

# РОМАН ЗЫКОВ

---

# С DATA SCIENCE

КАК МОНЕТИЗИРОВАТЬ  
БОЛЬШИЕ ДАННЫЕ



# РОМАН ЗЫКОВ

---

# С DATA SCIENCE

КАК МОНЕТИЗИРОВАТЬ  
БОЛЬШИЕ ДАННЫЕ



Санкт-Петербург • Москва • Минск

2021

ББК 32.973.233.02  
УДК 004.62  
3-96

## **Зыков Роман**

3-96 Роман с Data Science. Как монетизировать большие данные. — СПб.: Питер, 2021. — 320 с.: ил. — (Серия «IT для бизнеса»). ISBN 978-5-4461-1879-3

Как выжать все из своих данных? Как принимать решения на основе данных? Как организовать анализ данных (data science) внутри компании? Кого нанять аналитиком? Как довести проекты машинного обучения (machine learning) и искусственного интеллекта до топового уровня? На эти и многие другие вопросы Роман Зыков знает ответ, потому что занимается анализом данных почти двадцать лет. В послужном списке Романа — создание с нуля собственной компании с офисами в Европе и Южной Америке, ставшей лидером по применению искусственного интеллекта (AI) на российском рынке. Кроме того, автор книги создал с нуля аналитику в Ozon.ru.

Эта книга предназначена для думающих читателей, которые хотят попробовать свои силы в области анализа данных и создавать сервисы на их основе. Она будет вам полезна, если вы менеджер, который хочет ставить задачи аналитике и управлять ею. Если вы инвестор, с ней вам будет легче понять потенциал стартапа. Те, кто «пилит» свой стартап, найдут здесь рекомендации, как выбрать подходящие технологии и набрать команду. А начинающим специалистам книга поможет расширить кругозор и начать применять практики, о которых они раньше не задумывались, и это выделит их среди профессионалов такой непростой и изменчивой области. Книга не содержит примеров программного кода, в ней почти нет математики.

**16+** (В соответствии с Федеральным законом от 29 декабря 2010 г. № 436-ФЗ.)

ББК 32.973.233.02  
УДК 004.62

Иллюстрации Владимира Вышванюка.

Все права защищены. Никакая часть данной книги не может быть воспроизведена в какой бы то ни было форме без письменного разрешения владельцев авторских прав.

Информация, содержащаяся в данной книге, получена из источников, рассматриваемых издательством как надежные. Тем не менее, имея в виду возможные человеческие или технические ошибки, издательство не может гарантировать абсолютную точность и полноту приводимых сведений и не несет ответственности за возможные ошибки, связанные с использованием книги. Издательство не несет ответственности за доступность материалов, ссылки на которые вы можете найти в этой книге. На момент подготовки книги к изданию все ссылки на интернет-ресурсы были действующими.

ISBN 978-5-4461-1879-3

© ООО Издательство «Питер», 2021  
© Серия «IT для бизнеса», 2021  
© Роман Зыков, 2021

# ОГЛАВЛЕНИЕ

<b>Об авторе .....</b>	<b>8</b>
<b>Благодарности .....</b>	<b>10</b>
От издательства.....	11
<b>Введение .....</b>	<b>12</b>
Для кого эта книга .....	14
Как читать эту книгу.....	15
<b>Глава 1. Как мы принимаем решения .....</b>	<b>17</b>
Четыреста сравнительно честных способов .....	20
Чему можно научиться у Amazon?.....	21
Аналитический паралич.....	23
Погрешности — правило штангенциркуля.....	25
Принцип Парето .....	26
Можно ли принимать решения только на основе данных?.....	27
<b>Глава 2. Делаем анализ данных .....</b>	<b>31</b>
Артефакты анализа данных .....	33
Бизнес-анализ данных .....	34
Гипотезы и инсайты .....	34
Отчеты, дашборды и метрики.....	37
Артефакты машинного обучения.....	41
Артефакты инженерии.....	43
Кто анализирует данные .....	45
Идеальная кнопка .....	46

Продать аналитику внутри компании.....	48
Конфликт исследователя и бизнеса .....	49
Недостатки статистического подхода в аналитике .....	51
<b>Глава 3. Строим аналитику с нуля .....</b>	<b>55</b>
Первый шаг .....	56
Выбираем технологии .....	57
Поговорим об аутсорсе .....	61
Наем и увольнения .....	64
Кому подчиняются аналитики .....	67
Должен ли руководитель аналитики писать код .....	68
Управление задачами.....	71
Как управлять романтиками .....	76
<b>Глава 4. Делаем аналитические задачи .....</b>	<b>79</b>
Как ставить задачи аналитикам .....	80
Как проверять задачи.....	83
Как тестировать и выкладывать изменения в рабочую систему .....	87
Как защищать задачу перед инициатором .....	87
Нужно ли уметь программировать? .....	88
Датасет.....	90
Описательная статистика .....	91
Графики .....	94
Общий подход к визуализации данных .....	98
Парный анализ данных.....	100
Технический долг.....	101
<b>Глава 5. Данные .....</b>	<b>103</b>
Как собираются данные.....	104
Big Data.....	105
Связность данных .....	106
Много данных не бывает .....	107
Доступ к данным .....	108

Качество данных.....	110
Как проверяется и контролируется качество данных .....	112
Типы данных .....	114
Форматы хранения данных .....	116
Способы получения данных.....	120
<b>Глава 6. Хранилища данных.....</b>	<b>121</b>
Зачем нужны хранилища данных.....	122
Слои хранилища данных.....	125
Какие бывают хранилища.....	126
Как данные попадают в хранилища .....	130
Hadoop и MapReduce.....	132
Spark .....	136
Оптимизация скорости работы .....	140
Архивация данных и устаревание .....	141
Мониторинг хранилищ данных .....	143
Личный опыт.....	144
<b>Глава 7. Инструменты анализа данных.....</b>	<b>147</b>
Электронные таблицы.....	148
Сервисы блокнотов .....	149
Инструменты визуального анализа .....	151
Пакеты статистического анализа данных.....	153
Работа с данными в облаках.....	154
Что такое хорошая отчетная система .....	156
Сводные таблицы .....	159
OLAP-кубы.....	164
Корпоративные и персональные BI-системы.....	168
Мой опыт .....	170
<b>Глава 8. Алгоритмы машинного обучения .....</b>	<b>171</b>
Типы ML-задач.....	174
Метрики ML-задач.....	178

ML изнутри .....	182
Линейная регрессия .....	183
Логистическая регрессия .....	184
Деревья решений.....	186
Ошибки обучения .....	188
Как бороться с переобучением .....	190
Ансамбли.....	194
<b>Глава 9. Машинное обучение на практике .....</b>	<b>197</b>
Как изучать машинное обучение .....	198
Соревнования по ML .....	200
Искусственный интеллект.....	202
Необходимые преобразования данных .....	204
Точность и стоимость ML-решения .....	207
Простота решения .....	208
Трудоемкость проверки результата .....	210
Mechanical Turk / Yandex Toloka .....	210
ML и большие данные .....	212
Recency, Frequency и Monetary.....	212
Последний совет.....	215
<b>Глава 10. Внедрение ML в жизнь:</b>	
<b>гипотезы и эксперименты.....</b>	<b>217</b>
Гипотезы .....	218
Планируем тест гипотезы .....	220
Что такое гипотеза в статистике .....	222
Статистическая значимость гипотез.....	226
Статистические критерии для р-значений .....	229
Бутстрэп .....	232
Байесовская статистика .....	234
А/Б-тесты в реальности .....	237
А/А-тесты .....	240
Еще несколько слов о А/Б-тестах.....	242

Что делать перед А/Б-тестом .....	244
Конвейер экспериментов.....	245
<b>Глава 11. Этика данных.....</b>	<b>247</b>
Как за нами следят.....	248
Хорошее и плохое использование данных .....	255
Проблема утечки данных.....	258
Этика использования данных.....	260
Как защищают пользовательские данные .....	263
<b>Глава 12. Задачи и стартапы .....</b>	<b>267</b>
Веб-аналитика в рекламе .....	268
Внутренняя веб-аналитика .....	271
Маркетинг на основе баз данных .....	276
Стартапы.....	280
Личный опыт.....	284
<b>Глава 13. Строим карьеру.....</b>	<b>291</b>
Старт карьеры.....	292
Как искать работу.....	294
Требования к кандидатам.....	296
Вы приняли оффер.....	298
Как развиваться и работать.....	298
Когда менять место работы .....	302
Нужно ли все знать? .....	304
<b>Эпилог .....</b>	<b>308</b>
<b>Список литературы .....</b>	<b>309</b>

# ОБ АВТОРЕ

Роман Владимирович Зыков, 1981 года рождения, в 2004 году получил степень бакалавра, а затем магистра прикладной физики и математики в МФТИ (Московском физико-техническом институте).

В 2002 году начал свой карьерный путь в аналитике данных (Data Science) в качестве технического консультанта в компании StatSoft Russia, российского офиса одноименной американской компании-разработчика пакета статистического анализа данных STATISTICA. В 2004 году был принят на должность руководителя аналитического отдела интернет-магазина Ozon.ru, где создавал аналитические системы с нуля, в том числе веб-аналитику, аналитику баз данных, управленческую отчетность, внес вклад в систему рекомендаций.

В 2009 году консультировал ряд проектов инвестиционного фонда Fast Lane Ventures и гейм-индустрии.

В 2010 году возглавил отдел аналитики в интернет-ритейлере Wikimart.ru.

В конце 2012 года стал сооснователем и совладельцем маркетинговой платформы для интернет-магазинов RetailRocket.ru. На текущий момент компания является безусловным лидером на рынке в России и успешно работает на рынках Чили, Голландии, Испании и других.

С 2007-го вел блог «Аналитика на практике» (KPIs.ru — ныне не существует), где евангелизировал анализ данных в применении к бизнес-задачам в электронной коммерции. Выступал на отрас-

левых конференциях, таких как РИФ, iMetrics, Гес 2014 вместе с Аркадием Воложем (Yandex), бизнес-конференциях в Дублине и Лондоне, в посольстве США (AMC Center), университете Сбербанка. Печатался в технологическом прогнозе PwC, ToWave, «Ведомостях», «Секрете фирмы».

В 2016 году прочитал мини-лекцию в концертном зале MIT в Бостоне о процессах тестирования гипотез.

В 2020 году был номинирован на премию CDO Award.

# БЛАГОДАРНОСТИ

Я посвящаю эту книгу своей жене Екатерине и моим детям — Аделле и Альберту. Катя придала мне решимости написать книгу и приняла большое участие в редактировании текстов. За что я ей очень благодарен.

Также я благодарен своим родителям, которые вырастили и воспитали меня в очень непростое время. Отдельная благодарность моему отцу Владимиру Юрьевичу за то, что привил мне любовь к физике.

Я благодарен всем на моем долгом пути в аналитику данных. Илье Полежаеву, Большакову Павлу и Владимиру Боровикову за грамотное руководство, когда я только пришел в StatSoft. Бернару Люке, тогда генеральному директору Ozon.ru, а также коллегам в Ozon.ru: Александру Перчикову, Александру Алехину, Валерию Дьяченко — за совместное написание рекомендательной системы. Марине Туркиной и Ирине Коткиной — с вами было замечательно сотрудничать. Основателям проекта Wikimart.ru Камиллю Курмакаеву и Максиму Фалдину — те знакомства в Калифорнии очень сильно повлияли на меня. Александру Аникину — ты очень крутой был тогда, а сейчас вообще звезда. Основателям проекта Ostrovok.ru — Кириллу Махаринскому и Сержу Фаге, а также Жене Курьшеву, Роману Богатову, Феликсу Шпильману — с вами очень интересно было работать, я узнал много нового о разработке.

Я благодарен сооснователям Retail Rocket — Николаю Хлебинскому и Андрею Чижу. Отдельная благодарность венчурному фонду Impulse VC (Кириллу Белову, Григорию Фирсову, Евгению Пошибалову) — за то, что поверили в нас. Всем сотрудникам Retail

Rocket, особенно моим ребятам Александру Анохину и Артему Носкову — вы лучшие.

Я благодарен психологу Елене Клюстер, с которой работаю уже несколько лет, за осознание своих собственных границ и своих истинных желаний. Благодарен Андрею Гузю, моему тренеру по плаванию, за аналитический подход к тренировкам. Оказывается, так можно, и не только профессионалам, но и любителям.

Выражаю благодарность всем моим виртуальным рецензентам, особенно Артему Аствацатурову, Александру Дмитриеву, Аркадию Итенбергу, Алексею Писарцову. Роману Нестеру — за рецензию на главу по этике данных.

Благодарен всем, кто способствовал изданию этой книги. Прежде всего Алексею Кузменко, который помог мне быстро найти издательство, минуя бюрократические препоны. Отдельная благодарность Владимиру Вышванюку за ироничные иллюстрации кота Вилли. Юлии Сергиенко и Наталье Римичан, которые делали все, чтобы эта книга вышла в свет.

## ОТ ИЗДАТЕЛЬСТВА

Ваши замечания, предложения, вопросы отправляйте по адресу [comp@piter.com](mailto:comp@piter.com) (издательство «Питер», компьютерная редакция).

Мы будем рады узнать ваше мнение!

На веб-сайте издательства [www.piter.com](http://www.piter.com) вы найдете подробную информацию о наших книгах.

# ВВЕДЕНИЕ

Дайте мне точку опоры, и я переверну Землю.

*Архимед*

Дайте мне данные, и я переверну всю вашу жизнь.

*Data Scientist Архимед*

Данные повсюду — начиная от алгоритмов «Тиндера», который «матчит» вас с далеко не случайными людьми, и заканчивая информационными войнами, которые ведут политики. Никого уже не удивляет, что за каждым нашим шагом пристально следят: будь то история запросов в браузере телефона или ваши действия в офлайне. Задержитесь на секунду у витрины спортивного магазина — и ждите его таргетированную рекламу в соцсетях с минуты на минуту. Расскажите коллеге, что натворил ваш кот, — и вот сухие корма и наполнители уже тут как тут в вашей ленте.

Особо впечатлительные могут впасть в паранойю — но данные в этом не виноваты. Все зависит от того, в чьи руки они попадут. С анализом данных связано очень много мифов, а data scientist — одна из самых перспективных и «сексуальных» профессий будущего. В своей книге я намерен развенчать мифы и рассказать, как все обстоит на самом деле. Надеюсь, читатель, ты, как и я, окажешься на «светлой» стороне силы.

Я окончил МФТИ в начале нулевых и тогда же возглавил аналитический отдел интернет-магазина Ozon.ru, где создавал аналитические системы с нуля. Я консультировал инвестиционные фонды, гигантов

ритейла и гейм-индустрии, а восемь лет назад стал сооснователем и совладельцем маркетинговой платформы для интернет-магазинов RetailRocket.ru. Сейчас компания не просто является безусловным лидером на рынке в России, но и успешно работает на рынках Чили, Голландии, Испании и Германии. В 2016 году я прочитал лекцию в концертном зале MIT в Бостоне про процессы тестирования гипотез. В 2020 году номинировался на премию CDO Award.

Считается, что нужно потратить 10 000 часов для того, чтобы стать очень хорошим специалистом в своей области. Анализом данных я занимаюсь с 2002 года, когда это не было так популярно и хайпово. Так вот, чтобы получить эти заветные 10 000 часов, нужно проработать  $10\,000 \text{ часов} / 4 \text{ часа в день} / 200 \text{ дней в году} = 12.5 \text{ лет}$ . Я в полтора раза превысил эту цифру, поэтому, надеюсь, получилось написать книгу, которая будет очень полезна для вас, дорогие читатели.

Эта книга о том, как превращать данные в продукты и решения. Она основывается не на академических знаниях, а на моем личном опыте анализа данных длиной почти в двадцать лет. Сейчас существует очень много курсов по анализу данных (data science) и машинному обучению (machine learning, ML). Как правило, они узкоспециализированы. Отличие этой книги в том, что она, не утомляя частностями, дает цельную картину и рассказывает о том:

- как принимать решения на основе данных;
- как должна функционировать система;
- как тестировать ваш сервис;
- как соединить все в единое целое, чтобы на выходе получить «конвейер» для ваших данных.

## ДЛЯ КОГО ЭТА КНИГА

Эта книга предназначена для думающих читателей, которые хотят попробовать свои силы в области анализа данных и создавать сервисы на их основе.

Она будет вам полезна, если вы менеджер, который хочет ставить задачи аналитике и управлять ею. Если вы инвестор, с ней вам будет легче понять потенциал стартапа. Те, кто «пилит» свой стартап, найдут здесь рекомендации, как выбрать подходящие технологии и набрать команду. А начинающим специалистам она поможет расширить свой кругозор и начать применять практики, о которых вы раньше не задумывались — и это выделит вас среди профессионалов такой непростой и изменчивой области.

## КАК ЧИТАТЬ ЭТУ КНИГУ

Я писал эту книгу так, чтобы ее можно было читать не последовательно. Краткое содержание каждой главы:

Глава 1 «Как мы принимаем решения» описывает общие принципы принятия решения, как данные влияют на них.

Глава 2 «Делаем анализ данных» вводит общие понятия — с какими артефактами мы имеем дело, когда анализируем данные. Кроме того, с этой главы я начинаю поднимать организационные вопросы анализа данных.

Глава 3 «Строим аналитику с нуля» рассказывает об организации процесса построения аналитики: от первых задач и выбора технологии, заканчивая наймом.

Глава 4 «Делаем аналитические задачи» — полностью о задачах. Что такое хорошая аналитическая задача, как ее проверить. Технические атрибуты таких задач — датасеты, описательные статистики, графики, парный анализ, технический долг.

Глава 5 «Данные» о том, что говорят о данных — объемы, доступы, качество и форматы.

Глава 6 «Хранилища данных» рассказывает, зачем нужны хранилища, какие они бывают, также затрагиваются популярные системы для Big Data — Hadoop и Spark.

Глава 7 «Инструменты анализа данных», полностью посвящена наиболее популярным способам анализа от электронных таблиц в Excel до облачных систем.

Глава 8 «Алгоритмы машинного обучения» является базовым введением в машинное обучение.

Глава 9 «Машинное обучение на практике» является продолжением предыдущей главы: даются лайфхаки, как изучать машинное обучение, как работать с машинным обучением, чтобы оно приносило пользу.

Глава 10 «Внедрение ML в жизнь: гипотезы и эксперименты» рассказывает о трех видах статистического анализа экспериментов (статистика Фишера, байесовская статистика и бутстрэп) и об использовании А/Б-тестов на практике.

Глава 11 «Этика данных». Я не смог пройти мимо этой темы, наша область начинает все больше и больше регулироваться со стороны государства. Здесь поговорим о причинах этих ограничений.

Глава 12 «Задачи и стартапы» рассказывает об основных задачах, которые я решал в e-commerce, а также о моем опыте сооснователя проекта Retail Rocket.

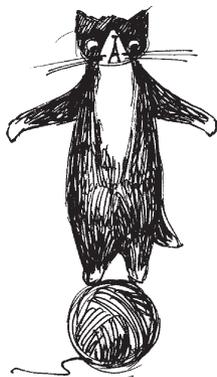
Глава 13 «Строим карьеру» больше предназначена для начинающих специалистов — как искать работу, развиваться и даже когда уходить дальше.

Внизу страниц книги вы встретите-QR коды. Они позволяют быстро перейти к источнику информации с помощью камеры смартфона. Номер источника указан рядом в квадратных скобках. Этот номер в таких же квадратных скобках указан в тексте на странице. Я рекомендую пользоваться этими источниками информации, если вы хотите глубже изучить вопрос.



1

## КАК МЫ ПРИНИМАЕМ РЕШЕНИЯ



«Итак, главный принцип — не дурачить самого себя. А себя как раз легче всего одурачить. Здесь надо быть очень внимательным. А если вы не дурачите сами себя, вам легко будет не дурачить других ученых. Тут нужна просто обычная честность.

Я хочу пожелать вам одной удачи — попасть в такое место, где вы сможете свободно исповедовать ту честность, о которой я говорил, и где ни необходимость упрочить свое положение в организации, ни соображения финансовой поддержки — ничто не заставит вас поступиться этой честностью. Да будет у вас эта свобода».

*Нобелевский лауреат Ричард Фейнман,  
из выступления перед выпускниками Калтеха  
в 1974 году*

Монетизация данных возможна лишь тогда, когда мы принимаем на основе этих данных правильные решения. Однако делать выбор, руководствуясь только статистикой, — плохая идея: как минимум нужно уметь читать их между строк и слушать свою интуицию (gut feeling). Поэтому в первой главе я расскажу про принципы, которыми я пользуюсь, принимая решения на основе данных. Я проверял на своем опыте — они работают.

Решения принимать непросто, ученые даже придумали новый термин «усталость от решений» (decision fatigue) [7]. Мы накапливаем стресс, совершая выбор каждый день сотни раз: и в какой-то момент, когда уже полностью вымотаны необходимостью принимать решения, можем махнуть рукой и начать действовать наугад.

Я не зря привел в начале этой книги цитату выдающегося физика, нобелевского лауреата Ричарда Фейнмана. Она напрямую касается как аналитики данных, так и вообще нашей жизни.

Как принимать верные решения, оставаясь честным с собой?

В книге «Биология добра и зла. Как наука объясняет наши поступки» профессор Стэнфордского университета, нейробиолог Роберт Сапольски [1] пишет, что на наши поступки, а значит и решения, влияет множество факторов: среда, в которой мы выросли, детские травмы, травмы головы, гормональный фон, чувства и эмоции. На нас всегда влияет множество факторов, которые мы даже не осознаем. Мы необъективны!

Лично я принял как данность, что гораздо легче принять необъективное и срезать углы, чем объективное, потому что для второго нужны серьезные усилия.

Вспомните об этом, когда будете предоставлять цифры кому-либо для принятия решения. И даже мои сотрудники указывали мне на то, что я сам нарушаю принципы объективности при утверждении результатов некоторых А/Б-тестов. Тогда я возвращался к реальности и соглашался с ними — объективность важнее моих априорных решений до проведения эксперимента.

В современном мире решения мы вынуждены принимать быстро и в условиях неопределенности. Но это не катастрофа. В квантовой физике, в отличие от классической, мы не знаем точно, где находится электрон, но знаем вероятность его нахождения. И вся квантовая физика базируется на этих вероятностях. С решениями точно так же — никто не знает истины, мы просто пытаемся угадать «правильное» с определенной долей успеха. И именно для этого нужны данные — увеличить вероятность успеха ваших решений!

## ЧЕТЫРЕСТА СРАВНИТЕЛЬНО ЧЕСТНЫХ СПОСОБОВ

Остап Бендер знал четыреста сравнительно честных способов отъема денег у населения. Профессиональный аналитик знает примерно столько же способов «повернуть» цифры в сторону «нужного» решения. К сожалению, это очень распространено в политике: вспомните, как государства рапортовали о количестве зараженных во время пандемии вируса COVID-19. Показатели смертности в России были занижены [6]. Оказалось, что если человек болел коронавирусом и умер от сопутствующего заболевания, то в соответствующую статистику не попадет. В большинстве же западных стран одной положительной пробы на коронавирус было достаточно, чтобы попасть в статистику. Если копнуть глубже, мы видим, что у всех разная методика и разные цели. Существуют объективные и субъективные причины неточности таких цифр.

Первая причина — объективная: много бессимптомных носителей вируса, они не обращаются к врачам. Здесь требуется «ковровое» тестирование населения, которое подразумевает случайную выборку из всей популяции определенной местности. Тестирование добровольное, значит, кто-то не придет. Некоторые — потому, что у них есть симптомы коронавируса, и если это будет обнаружено в процессе тестирования, то их запрут дома на двухнедельный карантин. А это может привести к потере заработка. В итоге мы получим выборку, смещенную в сторону здоровых людей, а значит, и заниженную оценку количества заболевших.

Вторая причина тоже объективная — нет денег на массовое тестирование населения.

А вот третья причина — субъективная: власти хотят уменьшить официальную статистику заболевших, чтобы снизить панику

среди населения и успокоить международное сообщество. Умение понимать эти причины и читать данные между строк — важное качество аналитика, которое позволяет ему делать более объективные выводы.

В работе я постоянно с этим сталкивался. Сейчас все живут на KPI, поэтому руководитель будет не очень-то рад плохим цифрам — премия висит на волоске. Возникает искушение найти показатели, которые улучшились. Нужно быть очень сильным руководителем, чтобы принять отрицательные результаты и внести коррекцию в работу. Аналитик данных как исследователь несет личную ответственность за результат своих цифр.

## ЧЕМУ МОЖНО НАУЧИТЬСЯ У AMAZON?

Мне всегда нравились письма Джеффа Безоса (основателя Amazon.com) акционерам. Например, еще в 1999 году он писал про важность систем персональных рекомендаций на сайте, которые сейчас стали стандартом в современной электронной коммерции. Меня заинтересовали два его письма: 2015 [2] и 2016 [3] годов.

В первом из них Безос писал про «Фабрику изобретений» (Invention Machine). Он точно знает, о чем говорит, — само провидение вело Amazon через тернии электронной коммерции. Попутно в компании изобретали много вещей, абсолютно новых для рынка: система рекомендаций, А/Б-тесты (да-да, именно они были пионерами тестирования гипотез для веба), AWS (Amazon Web Services), роботизация склада, кнопки на холодильник для мгновенного заказа порошка и многое другое.

Так вот, в первом письме он рассуждает о том, как в больших компаниях принимаются решения об изобретении новых продуктов. Часто процесс утверждения выглядит так: все участники процесса (как правило, руководители департаментов компании) проставляют свои «визы». Если решение положительное, идея или гипотеза отправляются на реализацию. Здесь Безос предупреждает, что

есть два типа решений и они не должны проходить один и тот же процесс утверждения.

Первый тип — решения, у которых нет или почти нет обратной дороги. Это как дверь, в которую можно войти, но нельзя выйти. Здесь нужно действовать очень внимательно и осторожно.

Второй тип — решения, у которых есть обратный ход. Дверь, в которую можно войти и выйти. Здесь он предлагает утверждать идею достаточно быстро, не мучая ее долго бюрократическими процедурами.

В письме 2016 года Безос противопоставляет компанию Дня 1 (Day 1), где сохраняется живая атмосфера создания компании и новых продуктов, компании Дня 2 (Day 2), которая статична и, как следствие, приходит к своей ненужности и смерти. Он выделяет 4 фактора, которые определяют компанию Дня 1:

- истинная одержимость покупателем (customer obsession);
- скепсис относительно моделей (a skeptical view of proxies);
- стремительное освоение внешних трендов (the eager adoption of external trends);
- стремительное принятие решений (high-velocity decision making).

Последний пункт мне кажется особенно важным в контексте этой книги. Для поддержания атмосферы компании Дня 1 требуется принимать быстрые и качественные решения. Мой шестилетний сын в таких случаях восклицает: «Но как?» Вот правила Безоса:

1. Никогда не использовать один-единственный процесс принятия решений (есть два типа решений, про которые я написал выше). Не дожидаться получения 90 % всей информации, нужной для принятия решения, — 70 % уже достаточно. Ошибаться не так страшно, если вы умеете быстро исправляться. А вот промедление, скорее всего, влетит вам в копейку.

2. Не соглашайся, но позволяй. Когда руководителю предлагают идею талантливые и успешные сотрудники, а он не согласен с ней — ему стоит просто позволить им ее реализовать, а не тратить их усилия на то, чтобы убедить. Безос рассказал, как дали зеленый свет одному из сериалов Amazon Studios. Он считал, что запускать этот проект рискованно: Безосу эта история казалась сложной в производстве и не слишком интересной. Но команда с ним не соглашалась. Тогда он сказал — хорошо, давайте пробовать. Им не пришлось убеждать Безосу в своей правоте, и они сэкономили уйму времени. Сам он подумал так: эти ребята уже привезли домой одиннадцать премий «Эмми», шесть «Золотых Глобусов» и три «Оскара» — они знают, что делают, просто у нас разные мнения.
3. Быстро находите причины несогласия и эскалируйте их наверх вашим руководителям. Разные команды могут иметь разные взгляды на решение. Вместо того чтобы тратить время на изматывающих совещаниях в попытках договориться — лучше эскалировать проблему наверх.

## АНАЛИТИЧЕСКИЙ ПАРАЛИЧ

Поспешишь — людей насмешишь. Все самые страшные ошибки я совершил, когда торопился — например, когда 15 лет назад пришел в Ozon.ru, чтобы поднять аналитику с нуля и должен был каждую неделю делать огромную простыню метрик о деятельности всей компании без нормальных проверок. Из-за давления менеджмента и спешки в этом регулярном еженедельном отчете было множество ошибок, с последствиями которых мне еще долго пришлось разбираться.

Современный мир живет на бешеных скоростях, но расчет метрик нужно делать очень аккуратно, а значит, не быстро. Конечно, не стоит впадать в другую крайность — «аналитический паралич», когда на каждую цифру будет уходить очень много времени. Иногда попытки сделать правильный выбор приводят к тому, что

я называю «аналитическим параличом» — когда уже пора принять решение, но не получается. Слишком высока неопределенность результата или рамки слишком жесткие. В аналитический паралич легко впасть, если пытаться принять решение чисто рационально, руководствуясь только логикой.

Яркий пример — книга «Проект Розы» Грэма Симсиона (кстати, одна из любимых книг Билла Гейтса и его жены). Молодой успешный ученый-генетик Дон ищет жену, но ни разу еще не продвинулся дальше первого свидания. Сочтя традиционный способ поиска второй половинки неэффективным, Дон решает применить научный подход. Его проект «Жена» начинается с подробнейшего 30-страничного вопросника, призванного отсеять всех неподходящих и выявить одну — идеальную. Понятно, что человека, который соответствовал бы такому списку требований, просто не существует. А потом он знакомится с девушкой, у которой нет ничего общего с его идеалом. Что из этого вышло — догадайтесь сами.

Второй пример — покупка машины. Когда я в последний раз делал это, то составил целую таблицу в Excel с техническими параметрами машин, вплоть до размера багажника в сантиметрах. Потом я целый год думал, ходил, смотрел, а в результате купил ту, которой и близко не было в моем списке, по велению сердца. Но на самом деле это было не веление сердца — просто за целый год поисков и анализа я понял, что в этом списке было по-настоящему важно для меня, а что нет.

Третий пример из моей профессиональной практики связан с гипотезами, точнее с тестами. Представьте себе, что вы вместо старого алгоритма рекомендаций разработали новый и хотите его протестировать. У вас есть 10 сайтов, где можно выполнить сравнение. В итоге вы получили: 4 выигрыша, 4 ничьи и 2 проигрыша. Стоит ли заменить старый алгоритм на новый? Все зависит от критериев решения, которые сформулировали перед тестом. Новый алгоритм должен победить на всех сайтах? Или вероятность выигрыша должна быть больше вероятности проигрыша? В первом случае очень высока вероятность того, что вы закопаетесь в бесконечных итерациях,

«полирую» свой алгоритм до совершенства, особенно учитывая то, что тесты займут не одну неделю. Это типичная ситуация «аналитического паралича». Во втором — условие кажется легким. Хотя из практики скажу, что даже его выполнить бывает очень непросто.

Я считаю, что в решениях нужно идти на осознанный риск, даже если нет всей информации. В наше время, конечно, мир меняется слишком быстро, чтобы иметь роскошь долго делать выбор. Если решение не примете вы, это сделает кто-то за вас, например ваш конкурент.

## ПОГРЕШНОСТИ — ПРАВИЛО ШТАНГЕНЦИРКУЛЯ

Следующая вещь, с которой я столкнулся, — это точность цифр. Я много занимался анализом маркетинговой деятельности, в том числе маркетинговых акций. Моя задача заключалась в том, чтобы как можно более точно оценить их влияние на бизнес. Вообще реакция менеджеров на цифры разная — все радуются положительным результатам, не проверяя их; но когда видят отрицательные — сразу ищут ошибку. И скорее всего, «найдут». Видите ли, все метрики содержат ошибку. Вспомните лабораторные работы по физике в школе или институте, сколько мы мучились и считали погрешности. Системные, случайные... Сколько времени мы тогда тратили на то, чтобы подогнать результат под нужную закономерность?

В бизнесе и науке так делать нельзя, особенно если вы хотите быть хорошим аналитиком и не пользоваться вышеупомянутыми «сравнительно честными способами» повернуть цифры туда, куда нужно. Сейчас погрешность измерений веб-аналитики (системы измеряют посещаемость веб-сайтов) составляет около 5%. Когда я еще работал в Ozon.ru, погрешность всей аналитической системы тоже была около 5% (расхождение с данными бухгалтерии). У меня был серьезный случай — я обнаружил ошибку в коммерческой системе веб-аналитики Omniture Sitecatalyst (ныне Adobe Analytics): она не считала пользователей с браузером Opera. В результате погрешность измерений была очень большой — около 10% всех совершенных заказов система, за которую мы платили более 100 тысяч долларов

в год, безнадежно потеряла. С такой погрешностью ей тяжело было доверять — но, к счастью, когда я обнаружил ошибку системы и сообщил о ней в Omniture, их разработчики ее устранили.

При работе с погрешностями я вывел правило, которое называю **Правилом штангенциркуля**. Есть такой инструмент для измерения размеров деталей с точностью до десятых долей миллиметра. Но такая точность не нужна при измерении, например, размеров кирпича — это уже за пределами здравого смысла, достаточно линейки. **Правило штангенциркуля** я бы сформулировал так:

Погрешность есть в любых измерениях, этот факт нужно принять, а саму погрешность — зафиксировать и не считать ее ошибкой (в одной из следующих глав я расскажу, как ее мониторить).

Задача аналитика — в разумной мере уменьшить погрешность цифр, объяснить ее и принять как данность. Как правило, в погоне за сверхточностью система усложняется, становится тяжелой с точки зрения вычислений, а значит, и более дорогой — ведь цена изменений становится выше.

## ПРИНЦИП ПАРЕТО

Итальянский экономист и социолог Вильфредо Парето в 1897 году, исследуя структуру доходов итальянских домохозяйств, выяснил, что 80 % процентов всех их доходов приходится на 20 % из них.

Универсальный принцип, названный в его честь, был предложен в 1951 году, и сейчас принцип Парето звучит так: «20 % усилий дают 80 % результата».

Опираясь на свой опыт, я бы так сформулировал его на языке данных:

- 20 % данных дают 80 % информации (data science);
- 20 % фич или переменных дают 80 % точности модели (machine learning);

- 20 % из числа успешных гипотез дают 80 % совокупного положительного эффекта (тестирование гипотез).

Я почти 20 лет работаю с данными и каждый день убеждаюсь в том, что эта закономерность работает. Это правило лентяя? Только на первый взгляд. Ведь чтобы понять, какие именно 20 % позволяют добиться результата, нужно потратить 100 % усилий. Стив Джобс в интервью Business Week в 98-м году сказал: «Простое сделать труднее, чем сложное: вам придется усердно поработать, чтобы внести ясность в ваши мысли, и тогда станет понятно, как сделать проще. Но это стоит того: как только вы достигнете этого, вы сможете свернуть горы».

Приведу пример того, как применяется правило Парето в машинном обучении. Для проекта обычно готовится ряд фич (входных параметров модели), на которых будет тренироваться модель. Фич может получиться очень много. Если выводить такую модель в бой, она будет тяжелой, требовать для своего поддержания много строк программного кода. Для такой ситуации есть лайфхак — посчитать вклад каждой фичи (feature importance) в результирующую модель и выбросить из модели фичи с минимальным вкладом. Это прямое использование правила Парето — 20 % фич дают 80 % результата модели. В большинстве случаев лучше модель упростить, пожертвовав небольшой долей ее точности, при этом проект будет в разы меньше исходного. На практике можно экономить время, подсмотрев фичи в решениях какой-нибудь схожей задачи на kaggle.com. Взять оттуда самые сильные из них и реализовать в первой версии собственного проекта.

## МОЖНО ЛИ ПРИНИМАТЬ РЕШЕНИЯ ТОЛЬКО НА ОСНОВЕ ДАННЫХ?

Можно, но не всегда и везде. Области, где можно принимать решение только на основе данных, уже захвачены компьютерными алгоритмами. Они не устают и очень хорошо масштабируются. Тот

же самый автопилот — уже относительно недалекое будущее: алгоритмы принимают решение на основе данных, поступающих к ним от датчиков, и управляют автомобилем.

Человек — универсальное существо, способное решать множество задач. Если задачу достаточно сузить, то можно сделать алгоритм, который будет работать быстрее тысячи человек. Но в отличие от человека, алгоритм не способен сделать ни шага в сторону от заданной схемы: его придется дорабатывать, внося каждое изменение. В этом и заключается вся суть автоматизации: сделать дешевле, быстрее и без участия человека. Поэтому все так одержимы идеей искусственного интеллекта.

На решения, принимаемые людьми, влияет много факторов. Один из них — так называемые когнитивные искажения, то есть систематические ошибки в восприятии и мышлении. Например, систематическая ошибка выжившего. Во время Второй мировой войны нью-йоркскому математику Абрахаму Вальду поручили исследовать пробоины на самолетах-бомбардировщиках, возвратившихся из боя, чтобы понять, в каких местах нужно усилить броню. Первое «логичное» решение — усилить броню в местах, поврежденных вражескими зенитками и пулеметами. Но Вальд понимал, что не может изучить все самолеты, включая те, что погибли. Проанализировав проблему как математик, он предложил бронировать те места, которые остались целыми, ведь самолеты с такими повреждениями не возвращались на базу, а значит, это самые уязвимые места.

Ошибку выжившего допустить очень легко. Чему нас учит пример Вальда? Тому, что нужно думать о всей генеральной совокупности. Ошибка выжившего является одной из форм когнитивных искажений.

В анализе данных ошибка выжившего — это учет известного и пренебрежение неизвестным, но существующим. С этой ошибкой очень легко столкнуться, когда у нас есть какие-то данные, на основе которых нужно сделать вывод. Любые данные — это выборка, ограниченное число. Сама выборка сделана из генеральной совокупности.

Если выборка сделана случайно и она достаточно большая, то все хорошо — большая часть закономерностей будет зафиксирована в выборке, и выводы будут объективными. Если же выборка была не случайной, как в нашем случае с самолетами, где в ней отсутствовали сбитые машины, — то, скорее всего, выводы будут ошибочными.

Например, в среднем только 1 из 100 посетителей сайта интернет-магазина совершает покупку. Если мы захотим улучшить свой сайт, чтобы больше покупателей покупали, то с какими посетителями нужно работать? Обычно дизайнеры и продуктологи обращают внимание на существующих покупателей из-за того, что с ними можно пообщаться, есть контактная информация из заказов, по ним есть хорошая статистика. Но эта выборка составляет всего лишь 1% от всей генеральной совокупности посетителей; с остальными почти невозможно связаться — это «сбитые самолеты». В итоге будет смещение выводов в сторону «выживших», а значит, выводы анализа не будут работать для всех посетителей.

Еще одно когнитивное искажение — предвзятость результата (outcome bias). Представьте себе — вам предлагают два варианта на выбор:

- Сыграть в «Орла или решку» — если выпадет орел, получите 10 000 рублей.
- Сыграть игральной костью с шестью гранями — если выпадет 6, получите 10 000 рублей.

Какой вариант выберете? Естественно первый, в котором шанс выиграть 1 к 2, во втором варианте значительно хуже — 1 к 6. Монету подбросили — выпала решка, вы ничего не получили. Тут же бросили кость, выпала шестерка. Будет ли обидно? Да, будет. Но было ли наше решение правильным?

Этот пример я взял из поста «Фокусируйтесь на решениях, а не на результате» [5] Кэсси Козырьков (Cassie Kozyrkov), которая работает директором по принятию решений [4] (Decision Intelligence) в Google. Она советует всегда оценивать верность решения, учитывая, какой

именно информацией вы обладали в момент его принятия. Многие люди жалеют, что они не уволились с работы раньше и только потеряли время, откладывая это решение, — я и сам в свое время так думал. И это отличный пример предвзятости результата — мы понимаем, что нужно было уволиться раньше, только обладая той информацией, которая у нас есть на данный момент. Например, что с тех пор зарплата так и не выросла, а интересный проект, который мы предвкушали, так и не был запущен. Оценивая последствия своего решения (особенно неудачного), в приступе самокопания мы не должны забывать, что принимали решение в условиях неопределенности.

# 2

## ДЕЛАЕМ АНАЛИЗ ДАННЫХ



Когда я работал в компании Wikimart.ru, основатели этого проекта познакомили меня с Ди Джейем Патилом (DJ Patil). Ди Джей был тогда одним из ангелов-инвесторов проекта, он руководил аналитикой в LinkedIn, затем был ведущим аналитиком данных (Chief data scientist) Белого дома в Вашингтоне при администрации Барака Обамы, тогдашнего президента США. Встречался я с Ди Джейем несколько раз в Москве и в Кремниевой долине в Калифорнии. В Москву он приезжал для презентации своей мини-книги «Building Data Science Teams» («Построение команд аналитиков данных») [9], выпущенной издательством O'Reilly. В книге он обобщил опыт ведущих компаний Кремниевой долины. Очень рекомендую вам эту книгу, так как ее мысли мне близки, и их я проверил на практике. Вот как автор определяет организацию, управляемую данными:

«A data-driven organization acquires, processes, and leverages data in a timely fashion to create efficiencies, iterate on and develop new products, and navigate the competitive landscape».

«Организация, управляемая данными, своевременно получает, обрабатывает и использует данные для повышения эффективности, итераций и разработки новых продуктов, а также навигации в конкурентной среде».

Далее Ди Джей указывает на принцип «Если ты не можешь измерить, ты не можешь это исправить» («if you can't measure it, you can't fix it»), который объединяет самые сильные организации, эффективно использующие свои данные. Вот рекомендации Патила, которые следуют из этого принципа:

- Собирайте все данные, какие только возможно. Вне зависимости от того, строите ли вы просто отчетную систему или продукт.
- Продумывайте заранее и делайте вовремя измерение метрик проектов.
- Позвольте как можно большему количеству сотрудников знакомиться с данными. Множество глаз поможет быстрее выявить очевидную проблему.
- Стимулируйте интерес сотрудников задавать вопросы относительно данных и искать на них ответы.

Эти мысли я еще озвучу в главе про данные. А теперь самое время поговорить о том, что мы получаем на выходе анализа данных.

## АРТЕФАКТЫ АНАЛИЗА ДАННЫХ

Здесь и далее под артефактами я буду понимать осязаемый результат, физический или виртуальный объект.

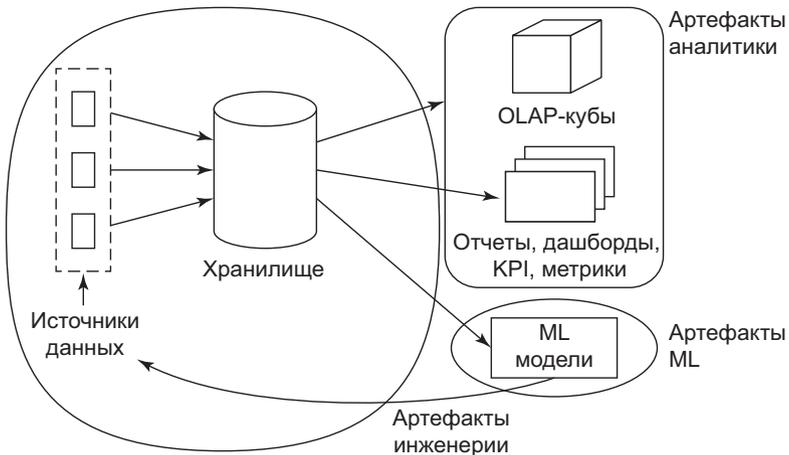


Рис. 2.1. Артефакты аналитики

Их можно разделить на три вида (рис. 2.1):

- артефакты бизнес-анализа данных (business intelligence);
- артефакты машинного обучения (machine learning);
- артефакты инженерии данных (data engineering).

Поговорим о них подробнее.

## БИЗНЕС-АНАЛИЗ ДАННЫХ

Бизнес-анализ данных (Business Intelligence, BI) — термин уже устоявшийся. Вот какое определение дает Википедия:

«Business Intelligence — это обозначение компьютерных методов и инструментов для организаций, обеспечивающих перевод транзакционной деловой информации в человекочитаемую форму, пригодную для бизнес-анализа, а также средства для работы с такой обработанной информацией».

Под бизнес-анализом я подразумеваю объединение контекста бизнеса и данных, когда становится возможным бизнесу задавать вопросы к данным и искать ответы. Первыми артефактами являются так называемые инсайты и гипотезы, вторыми — отчеты или дашборды, метрики и ключевые показатели (Key Performance Indicator). Поговорим подробнее об инсайтах и гипотезах.

## ГИПОТЕЗЫ И ИНСАЙТЫ

Инсайт (insight) в переводе с английского — понимание причин. Именно за этим обращаются к аналитикам. В поиске инсайтов помогают аналитика и статистика:

- Цель аналитики заключается [10] в помощи формулирования гипотезы.

- Цель статистики [10] в том, чтобы эту гипотезу проверить и подтвердить.

Это требует пояснений. В бизнесе, да и в жизни тоже, мы ищем причину проблемы, задавая вопрос «почему?». Не зная причины, мы не можем принять решение. В игру вступает аналитика — мы формулируем список возможных причин: это и есть гипотезы. Чтобы это сделать, нужно задать несколько вопросов:

- Не происходило ли что-нибудь подобное раньше? Если да, то какие тому были причины? Тогда у нас будет самая первая и самая вероятная гипотеза.
- Обращаемся к бизнес-контексту: не происходило ли каких-либо неординарных событий? Часто как раз параллельные события влияют на возникновение проблемы. Еще плюс пара гипотез.
- Описательный анализ данных (exploratory data analysis): смотрим данные в аналитической системе (например, кубах OLAP), не видно ли каких-либо аномалий на глаз? Например, какие-либо распределения изменились во времени (типы клиентов, структура продаж и т. д.). Если что-то показалось подозрительным — дополняем список гипотез.
- Использование более сложных методов поиска аномалий или изменений, например, как описано здесь [11].

Наша цель — накидать как можно больше гипотез, не ограничивая фантазию, затем отсортировать их по списку в порядке убывания вероятности, чтобы найти верную гипотезу как можно быстрее. Или даже воспользоваться бритвой Оккама, выстроив гипотезы по возрастанию сложности проверки. Иначе можно столкнуться с аналитическим параличом: превратить задачу

в научную работу, когда проверяются все гипотезы без исключения. Такого в реальной жизни не бывает, у нас всегда есть ограничения в ресурсах — как минимум во времени. Как только гипотезы готовы, приходит очередь статистики, с помощью методов которой они проверяются. Как это сделать — расскажу в главе про эксперименты в ML.

Когда я был директором по аналитике Retail Rocket (сервис рекомендаций для интернет-магазинов), мне и аналитикам часто приходилось заниматься расследованиями, ведь бизнес довольно большой — больше 1000 клиентов, и странности, с которыми приходится разбираться, случаются часто. Много приходится работать с так называемыми А/Б-тестами: это тесты, где аудитория сайта делится на две части случайным образом — первой части пользователей показывается одна версия сайта, второй — другая. Такие тесты обычно используют, чтобы оценить влияние изменений на бизнес-метрики сайта, когда первая версия — это старая версия или контрольная группа, а вторая — новая версия. Если это интернет-магазин — это, скорее всего, будут продажи. Далее к результатам теста применяются статистические критерии, которые подскажут достоверность изменений.

Такие тесты хорошо выявляют проблемы: например, версия сайта с обновленными рекомендациями Retail Rocket проиграла старой версии рекомендаций. Как только это становится известным, начинается расследование. Проверка начинается с интеграции, и это первая гипотеза: правильно ли передаются нам данные от интернет-магазина. Обычно на этом этапе решается 60–70 % проблем. Далее мы пытаемся найти отличие этого магазина от остальных в такой же тематике, например магазины одежды. Это вторая гипотеза. Третья гипотеза — возможно, мы изменили дизайн сайта таким образом, что полезная информация опустилась ниже на странице сайта. Четвертая гипотеза — тест мог отрицательно повлиять на определенные категории товаров. Собрав набор таких гипотез, мы начинаем их проверять примерно в такой последовательности, как я описал. Довольно

часто мы находим причину проблем, но иногда это не удается, его величество случай играет с нами в кошки-мышки, и эту мышку очень сложно найти.

Однажды клиент — магазин «Дочки-Сыночки» — тестировал наш сервис и сервис одного из наших российских конкурентов, чтобы выбрать лучший, и это превратилось в настоящий детектив [12]. Чтобы точно не проиграть в тесте, конкурент перемещал некоторое число пользователей, которые были близки к покупке, (например, добавили товар в корзину) из конкурентных (наших) сегментов в свой — причем делалось это не на постоянной основе, а в отдельные дни и часы. Основной метрикой сравнения была конверсия: процент пользователей, совершивших покупку. Ясно, что в той «мошеннической схеме» такой процент будет выше там, куда перетянули пользователей. Здесь компания Retail Rocket пошла на принцип! Мы стали копать. Через два месяца были обнаружены и опубликованы [12] факты подтасовки результатов. В итоге прошел ряд судебных процессов, и справедливость восторжествовала.

## ОТЧЕТЫ, ДАШБОРДЫ И МЕТРИКИ

Понятие самого отчета очень широкое, здесь я подразумеваю под ним табличное или иное графическое представление данных. Отчеты могут быть разными:

- Просто таблица с «сырыми» данными или так называемые «выгрузки», например, таблица с заказами клиентов.
- Отчет с «агрегированными» данными. Под агрегацией я подразумеваю суммы, количество и иные статистики. Например, таблица с именами клиентов и количеством заказов, который каждый из них совершил.
- Дашборды (dashboards) содержат ключевые показатели и метрики.

Первые два относительно просты и делаются через специальные системы, которые могут генерировать отчеты по запросу. Я стараюсь максимально оставить эту задачу на откуп пользователям. Почему? Потому, что тратить на это время высококвалифицированных сотрудников — значит стрелять из пушки по воробьям. Кстати, этим могут заняться стажеры-аналитики — отличный способ наработать опыт и понять бизнес-контекст. Как мотивировать пользователей стараться самостоятельно? Во-первых, они сэкономят время, которое обычно тратят на постановку задачи и ожидание результата. Во-вторых, получают возможность самим вносить правки и изменения — а значит творить. По моему опыту, обычно этим занимаются очень перспективные сотрудники, которые не боятся освоить новый инструмент, чтобы делать свою работу лучше. Остальным придется пройти через стандартный цикл планирования задач: а это время (дни, а иногда недели) и очень четкая формулировка технического задания. И кстати, все генеральные директора (Ozon.ru, Wikimart.ru, Ostrovok), с которыми я работал, пользовались OLAP-кубами со своих компьютеров. С их помощью они всегда могли ответить на простые вопросы, а если не получалось — обращались к аналитикам.

Теперь взглянем на дашборды и начнем с определения из Википедии:

«Дашборд — это тип графического интерфейса, который делает возможным быструю оценку ключевых показателей для конкретной цели или бизнес-процесса. Часто подразумевается, что это просто другое название отчета о прогрессе или просто отчета».

Как правило, дашборд состоит из ключевых показателей и метрик, выраженных с помощью графических инструментов (графики, диаграммы и т. д.):

- Ключевой показатель (key performance indicator, KPI) — это индикатор, который показывает, насколько далеко мы находимся от цели, например отставание/опережение плана.

- Метрика — это цифра, которая характеризует процесс, обычно используется как справочная информация.

Главное различие между метрикой и ключевым показателем — наличие цели. В ключевых показателях она есть, в метриках обычно нет. Как правило, ключевые показатели используются в дашбордах компаний, где уже и продукт, и бизнес-процесс «устоялись». Уже накоплено некоторое множество данных, что делает возможным прогнозирование целей. Метрики я обычно вношу туда, где нет достаточно устойчивых процессов. Их обычно ставят, когда цель пока не прогнозируема. Зрелость дашборда можно определить по соотношению количества ключевых показателей и метрик: чем метрик больше, тем более незрелый дашборд мы получаем на выходе.

Обычно дашборды характеризуют какой-то бизнес-процесс (далее слово «процесс» без «бизнес»), например эффективность рекламы, складские остатки, продажи и т. д. Есть еще важные характеристики дашбордов:

- не является «простыней цифр»;
- показывает, где возникла проблема, но не дает ответа на вопрос почему.

Часто велико искушение сделать огромную простыню цифр, закрывающую все аспекты бизнеса. И я понимаю владельцев/менеджеров компаний — на старте проекта по построению внутренней аналитической системы всегда хочется большего. Я наблюдал это и в Ozon.ru, и в Ostrovok.ru. К слову, эти строки написаны по мотивам письма, которое я писал восемь лет назад операционному директору Ostrovok.ru, — он хотел получить от аналитиков ту самую «простыню». А я считаю такое цифровым «микроменеджментом», в нем легко запутаться, самые важные показатели похоронены среди второстепенных. С первого взгляда будет сложно понять, где возникла проблема, а это основная функция дашбордов. Борьба с этим можно, например, через внедрение OKR — цели и ключе-

вые результаты (Objectives and Key Results) [13] — или системы сбалансированных показателей (Balanced Scorecard). В этой книге я не буду подробно останавливаться на этих методиках, но рекомендую вам с ними ознакомиться. Также можно чаще пользоваться графическими элементами, например, добавив на график линию тренда (с помощью семиточечного скользящего среднего, чтобы убрать недельную сезонность), будет легче заметить восходящий или нисходящий тренд.

Дашборд отвечает на вопрос, где есть проблема, а не почему она возникла. Может возникнуть искушение сделать огромный детальный отчет, чтобы быстро найти причину, — но тогда ваш дашборд превратится в простыню цифр, о которой я писал выше. В нем не будет интерактивности, и нужно будет «провалиться» внутрь этих цифр, чтобы проанализировать их, а для этого понадобятся совсем другие инструменты. Когда вам в следующий раз захочется это сделать, вспомните, удавалось ли вам хоть раз найти причину проблемы с помощью дашборда.

Никакой дашборд не заменит интерактивный анализ, для которого нужны соответствующая аналитическая система (SQL, OLAP, Google Data Studio, Tableau) и знание контекста. Мы никогда не сможем придумать ограниченный набор отчетов, которые будут отвечать на вопрос «почему». Максимум, что мы можем сделать, — наращивать (но не слишком) объем правильных метрик, исходя из инцидентов, за которыми будем следить.

Поэтому я всегда за лаконичные автоматические отчеты, которые будут отвечать на два вопроса: есть ли проблема и где она возникла. Если проблема есть, нужно лезть в интерактивные системы анализа данных.

Разработка дашбордов — это одна из самых нелюбимых работ у тех, кто занимается анализом данных. Когда я обсуждал этот вопрос с Ди Джейем Патилом, отметив, что 50 % времени аналитического отдела занимает работа над отчетностью, он сказал, что у них в LinkedIn тоже периодически накапливался пул таких задач и приходилось их закрывать. И взгрустнул. Но дашборды очень нужны — они помогают контролировать общее здоровье вашей системы — вверенных вам серверов и сетей, если вы системный администратор, или всей компании, если вы генеральный директор.

## АРТЕФАКТЫ МАШИННОГО ОБУЧЕНИЯ

Раньше компьютером можно было управлять только с помощью прямых команд или инструкций: поверни сюда, дай назад, сложи и т. д. Это обычное, так называемое детерминированное программирование — для нас понятен алгоритм в виде инструкций, мы его описали, и компьютер подчиняется ему. Машинное обучение предполагает совершенно другой подход к программированию — обучение на примерах. Здесь мы показываем системе что-то с помощью примеров, тем самым избавляем себя от самостоятельного написания инструкций, что бывает совсем не просто. Это становится работой по обучению алгоритма ML.

Для меня машинное обучение отличается от программирования так же, как квантовая физика отличается от классической. Там мы точно можем определить, где находятся планеты Солнечной системы, а в квантовой механике все есть вероятность — мы получим только вероятность нахождения электрона. Так и в машинном обучении вы будете работать с вероятностями — например, модель будет предсказывать вероятность того, что на фотографии кошка. Под моделью я подразумеваю компьютерную программу (далее программный код), который обладает рядом признаков.

- Функция обучения (`train`), в которую можно отправить данные для обучения признаки (они же фичи, независимые предикторы, независимые переменные) и правильный ответ (`output`). Сам результат обучения сохраняется внутри модели.
- Функция предсказания (`predict`), которая предсказывает результат для новых примеров.

Поясню на примере одной задачи. У нас есть много фотографий собак и кошек, и нам нужно их разделить: в одну папку сложить файлы с фотографиями кошек, в другую — фото собак. Фотографий очень много — миллионы, вручную не сделать. У вас есть размеченный набор данных для обучения — тысяча фотографий, для каждой указано, кошка там или собака. Вы берете нужную модель, «скармливаете» ей в функцию `train` набор с размеченными данными, и она учится на них. Сама модель выглядит для нас как черный ящик. Конечно, в него можно заглянуть, что мы и сделаем в главе про машинное обучение. Как только модель обучится, мы уже начинаем одна за другой скармливать ей фотографии, которые нужно разделить. Для каждой фотографии модель вернет нам вероятность того, кошка там или собака. Используя эти цифры, уже несложно разделить фотографии.

Этот пример я видел вживую, когда глубокое обучение нейронных сетей (`Deep Learning`) только набирало оборот. На одном из конкурсов `Kaggle.com` была точно такая же задача [14]. Чтобы поиграть с этой задачей, я нашел код в интернете, который не использовал нейронные сети. Естественно, ничего не получилось, мой алгоритм был настолько плох, что проще было бросить монетку и получить такой же результат. Первые места заняли исследователи, у которых результат был близок в 99 % (точность угадывания). Их модель была основана на сверточных нейронных сетях. Меня тогда по-

разил результат. Глубокое обучение нейронных сетей еще не было популярным, а ведь это было всего лишь в 2013 году. Вот так быстро меняются технологии!

Следующий постулат: данные, на которых обучена модель, — это часть кода. Это еще одно серьезное отличие от классического программирования. Чтобы сделать «тиражирование» программного кода, его текст можно опубликовать в Сети. Эта программа будет работать везде одинаково. Если вы захотите «поделиться» своей обученной моделью, то вам придется отправить не только код, но и весь получившийся черный ящик. Именно так исследователи и делятся своими обученными моделями. Например, модель нейронной сети Resnet 50 [15] была обучена на миллионах изображений. Она уже полностью готова к работе; просто показывая ей разные фотографии, вы получите названия предметов, которые там изображены.

## АРТЕФАКТЫ ИНЖЕНЕРИИ

Ничего нельзя сделать без инженерии аналитической системы. Даже для самых простых вещей «на коленке» нужно продумывать следующие вопросы:

- Откуда и с какой периодичностью брать данные и как туда получить доступ?
- Какова нагрузочная способность источников данных, чтобы и бизнес работал без сбоев, и данные как можно быстрее были доступны для анализа?
- Какую архитектуру хранилища сделать? Или, может, не делать его вовсе?
- Какую аналитическую систему выбрать?
- Как использовать в процессах обученную модель машинного обучения (далее ML-модель)?

Таких вопросов может быть очень много. Эти вопросы должны решаться и автоматизироваться. Артефактами инженерии будут:

- Архитектура аналитической системы.
- Программный код, который обеспечивает работу системы.

Если все сделано идеально, то этих двух артефактов достаточно, чтобы развернуть (подготовить) аналитическую систему за минимальное время. В крутых реализациях это можно сделать автоматически, нажатием одной кнопки. Это очень важно для устойчивой работоспособности аналитической системы. К сожалению, работа людей, которые этим занимаются (администраторы, инженеры), почти незаметна, особенно когда все хорошо работает. Их почти не замечают, не понимают, чем они занимаются, и поэтому часто не ценят.

Архитектура аналитической системы состоит из нескольких уровней:

- Физический — серверы и каналы связи между ними.
- Данные — хранилища данных.
- Приложения — программы, с помощью которых пользователи получают доступ к данным, а также публикуют модели ML.

За физический уровень отвечают системные администраторы. Они занимаются «железом», чтобы система была отказоустойчивой. Также администраторы постоянно мониторят здоровье системы. Знаете, как определить, что у вас хорошая система и администраторы? Вы о работе администраторов ничего не слышите, а система работает без серьезных сбоев.

За уровень данных отвечают инженеры данных (Data Engineers или ETL Engineers): их основная задача — сделать так, чтобы данные доставлялись от источников данных и сохранялись в хранилищах данных. Часто они же отвечают за предобработку данных и развертывание BI-систем (OLAP-кубы и отчетные системы).

За уровень приложений отвечают инженеры машинного обучения (ML engineers) и аналитики данных (data scientists). ML-инженеры занимаются созданием ML-моделей и иногда — их развертыванием, чтобы они работали на благо вашего бизнеса (хотя в больших компаниях развертыванием моделей «в бою» часто занимаются другие инженеры). Аналитики данных занимаются тестированием моделей и их оценкой. В небольших компаниях эти две роли часто совмещаются. Однажды я проходил собеседование в офисе компании Quora.com (социальная сеть) в Пало-Альто (Калифорния, США) и там выяснил, что местные ML-инженеры как раз и занимаются разработкой ML-моделей, а аналитики данных занимаются метриками, анализом данных и прочим, но не ML-моделями.

## КТО АНАЛИЗИРУЕТ ДАННЫЕ

Чем ближе анализ данных к точке принятия решений — тем лучше. Если вопрос возник у руководителя и у него есть полное понимание бизнес-контекста (какие события были и т. д.), а аналитическая система обладает хорошей интерактивностью, то большинство вопросов решаются на раз-два-три. До 80 % вопросов (вспомните правило Парето), если быть точным. В чем плюсы такого решения? Нет посредников — выше скорость! Пользователь даже может не иметь четко сформулированного вопроса, который точно понадобится, если ставить задачу аналитикам. Для этого очень важно внутри компании «продавать» аналитический подход и регулярно обучать пользователей.

Если бизнес-контекст размытый, находится вне компетенций или вопрос заказчика слишком сложный, то тут подключают в работу аналитика. Обычно я рекомендую в отделах, департаментах держать собственного «децентрализованного» аналитика, который в курсе дел этого департамента, то есть владеет бизнес-контекстом и при этом обладает развитыми аналитическими и техническими навыками. Это вторая линия обороны. Такой

«карманный» аналитик сможет решать вопросы внутри отдела/департамента быстрее центрального просто потому, что у него нет других задач.

Третий уровень — передаем задачу условному центральному отделу аналитиков данных, если:

- задача требует изменения ядра системы;
- задача технически сложна для аналитика какого-то отдела;
- требуется большая коллаборация между отделами для ее решения.

В Ozon.ru я не полностью ее реализовал, но уже в Wikimart.ru была сделана такая система: интерактивный анализ данных в OLAP-кубах дал возможность пользователям быстро решать свои вопросы, аналитики отделов решали проблемы анализа данных отделов, а центральный отдел создавал ядро всей аналитической системы. Кстати, многие бывшие пользователи OLAP-кубов в Ozon.ru потом писали мне, что им очень не хватает этих аналитических решений в других компаниях. К хорошему быстро привыкаешь.

## ИДЕАЛЬНАЯ КНОПКА

До Физтеха я вообще не знал английского — в школе у меня был немецкий, о чем я очень жалел. На Физтехе принято учить английский язык, поэтому сразу на первом курсе была сформирована группа начинающих, в которую попали всего 4 человека. На протяжении трех курсов у нас проходило 2 занятия в неделю. Это был один из самых моих любимых предметов, и он здорово мне пригодился. На четвертом курсе я устроился подрабатывать переводчиком книги с английского языка на русский. Это была книга о программе анализа данных STATISTICA компании StatSoft.

Я устроился туда стажером, переводил книгу, помню норматив — 15 000 знаков в день, от которого к вечеру пухла голова. Постепенно я втянулся и стал заниматься более интересными вещами: преподавал клиентам компании, проводил презентации для продаж, ездил в командировки и т. д. Тогда я постоянно консультировал клиентов и понял одну важную вещь: многие клиенты хотят получить кнопку и желательно на стуле — садись на нее, а она делает всю твою работу.

Кроме того, заказчику чаще всего лень вдаваться в детали, и он готов платить огромные деньги просто за яркую обертку. Этот феномен очень хорошо эксплуатируется продавцами IT-решений, консультантами всех мастей. Я наблюдал его, когда Ozon.ru выбирал решение для веб-аналитики между Omniture SiteCatalyst и Webtrends. Обе команды продавцов активно рассказывали о «светлом» будущем. Так как никто из принимающих решения не был особенно в теме (я, кстати, тоже), то выбрали тех, кто «поет» лучше. Презентация Omniture выглядела эффектней, они нам подарили радиоуправляемые машинки и всякие подарки. Поэтому выбор был сделан в их пользу, хотя я нахожу системы равнозначными, и стоили они почти одинаково. В продолжение истории — когда я пришел в Wikimart.ru, мне уже было понятно, что нужно пользователям от веб-аналитики. Я быстро накатал техническое задание, его реализовали разработчики, и через два месяца после моего прихода в компании была своя система веб-аналитики, ничуть не хуже Omniture. И экономия составляла порядка 100 тысяч долларов в год.

Я не утверждаю, что продавцы и консультанты плохи, я призываю вас самих не лениться. Прочитайте книгу, а лучше две по теме, дочитайте их до конца. Ищите независимых экспертов, которым сможете доверять. Главное — это погружаться в детали, именно там кроются и все проблемы, и их решения. Будьте скептически по отношению к своим эмоциям. Будьте скептически к докладам на конференциях, они часто однобоки и слишком позитивны, чтобы

быть правдой. Там есть интересные вещи, но мало кто рассказывает, чего стоило то или иное решение.

## ПРОДАТЬ АНАЛИТИКУ ВНУТРИ КОМПАНИИ

Для меня это очень непростой вопрос. В разделе «Кто анализирует данные» я упоминал, что аналитическую систему мне удалось поднять за два месяца (причем я работал тогда два дня в неделю). «Продажа» ее пользователям заняла гораздо больше времени, и только спустя 4 месяца системой начали более-менее пользоваться. Причем kick-off-презентацию я делал сразу после запуска: пригласил туда всех значимых сотрудников компании, включая основателей.

Мне легче работать на индивидуальном уровне: поговорить за обедом, обменяться парой фраз у кулера с водой, поинтересоваться чужими задачами, копнуть глубже. Затем представить в уме схему решения — что есть и чего не хватает. Прислать решение человеку, показать его лично. Приучать людей к новой системе лучше не навязывая, а обучая — так пользователи постепенно поймут, как она может ускорить решение их задач.

В Retail Rocket мы так внедряли аналитику на базе ClickHouse. Ранее данные были доступны только в SQL-интерфейсе к вычислительному кластеру на базе Spark/Hadoop (эти технологии мы обсудим в главе о хранилищах), Hive. Подобная схема используется в компании Facebook, они так дают доступ к данным внутри своей компании. Проблема этой технологии заключается в том, что она медленно считывает, запросы выполнялись до 30 минут, а данные доступны только до вчерашних суток. Пользовались этой системой только сотрудники технической поддержки. В одном из проектов мы попробовали аналитическую базу данных ClickHouse от Яндекса. Нам она понравилась: быстро считала, большая часть запросов — это секунды, можно было сделать систему, близкую к реальному времени. Вначале пересадили на нее техническую

поддержку, а в Retail Rocket это одно из самых сильных подразделений. Они очень быстро полюбили эту технологию за скорость и отказались от использования медленного Hive. Далее мы начали предлагать новую систему пользователям внутри компании. После обучающих презентаций многие сотрудники зарегистрировались в системе, но не стали ею пользоваться. Тогда мы пошли другим путем: все входящие задачи от сотрудников, которые можно было решить с помощью этой системы, начали раз за разом «отфутболивать» — возвращать под соусом «сделай сам», демонстрируя возможности системы. И часть пользователей стала работать с системой самостоятельно! Там многое еще можно сделать, но то, что уже сделано, я считаю успехом.

Вообще, если абстрагироваться от продаж аналитики внутри компании, в структуре бизнеса часто не хватает такой роли, как руководитель внутреннего продукта. Задачей которого было бы помогать сотрудникам работать эффективнее, лучше автоматизировать внутреннюю деятельность, избавляться от неэффективного «мартышкиного» труда. В компаниях часто любят внедрять процессы, чтобы забюрократизировать работу, но мало кто думает о внутреннем продукте, чтобы целенаправленно облегчить работу своим сотрудникам. Я думаю, причина в том, что сложно посчитать, сколько зарабатывает на этом компания. Но на самом деле это очень важная роль. И если она есть — продажа аналитики внутри компании происходит естественным образом.

## КОНФЛИКТ ИССЛЕДОВАТЕЛЯ И БИЗНЕСА

Работая уже много лет в области анализа данных, я заметил конфликт интересов, который в некотором роде похож на конфликт отцов и детей: молодые и дерзкие аналитики и инженеры хотят создать если не памятник, то что-то действительно значимое и красивое, о чем можно поведать миру, чем можно поднять самооценку или написать красивую строчку в резюме. Многие из них одержимы идеей применять машинное обучение в реальной

жизни — там, где это нужно и не нужно. Но в отличие от исследователей, у бизнеса менее романтические цели — в первую очередь это, как ни крути, деньги: в уставе почти любого российского ООО написано: «Целью деятельности Общества является достижение максимальной экономической эффективности и прибыльности».

Я много раз проводил собеседования и с новичками, и с опытными людьми, и знаю, что поиск интересной работы — главный тренд на рынке труда. Действующие специалисты говорят: «Надоело заниматься рутинной и всякой ерундой, хочу заниматься моделями машинного обучения». Новички: «Хочу заниматься компьютерным зрением и NLP (машинное обучение в лингвистике)». В целом людей объединяет любовь к машинному обучению, но для меня это звучит как любовь строителя к молотку, который пытается построить дом лишь с его помощью.

Эндрю Ён (Andrew Ng), которого я считаю одним из главных исследователей и популяризаторов машинного обучения, автор моего любимого курса на Coursera, в своей рассылке `deeplearning.ai` писал:

«Существует огромная разница между построением модели в блокноте Python (Jupyter Notebook) на компьютере в лаборатории и созданием реально работающих систем, которые создают ценность. Кажется, что сфера AI переполнена людьми, но по факту она широко открыта для профессионалов, которые знают, что делают».

Курсы по анализу данных и машинному обучению делают полезное дело, но их можно сравнить с игрушечными моделями кораблей — они далеки от настоящих примерно так же, так курсы — от реального применения ML в жизни.

Ноа Лоранг, аналитик данных из компании Basecamp, в своем блоге [16] пишет:

«Маленькая грязная тайна продолжающегося бума data science в том, что то, что обычно подразумевается под этим на самом деле, не нужно бизнесу. Бизнесу нужна точная и полезная информация для принятия решений: как тратить время и ресурсы компании. Очень небольшое подмножество задач в бизнесе может быть лучшим образом решено машинным обучением; большинство же из них нуждается в хороших данных и понимании их смысла, что может быть достигнуто простыми методами».

Я готов подписаться под каждым словом. К сожалению, проблема в хайпе вокруг нашей профессии. Курсы по анализу данных рекламируют видеоблогеры, президенты говорят об искусственном интеллекте, акции Tesla растут каждый день. Но, с одной стороны, есть молодые специалисты-романтики, которые хотят строить космические корабли, а с другой — компании, которые хотят зарабатывать деньги. О том, как примирить их интересы, пойдет речь в следующих главах.

## НЕДОСТАТКИ СТАТИСТИЧЕСКОГО ПОДХОДА В АНАЛИТИКЕ

Аналитика данных «усредняет» человека. На вопрос «Би-би-си» про индивидуальность человека Карл Юнг, основатель аналитической психологии, сказал [17]:

«Что тут скажешь: это — одно из следствий современной науки, которая основывается на статистическом усреднении. А для статистического усреднения человек как таковой совершенно не важен. Это — абстракция, а не конкретная личность.

Наше мировоззрение, тоже основанное на статистическом усреднении, является абстракцией, которая не имеет никакого отношения к тому, что происходит в реальном мире. В таком мировоззрении

индивидуум есть не что иное, как случайный феномен. Но в действительности индивидуум — это единственная реальность.

Если вы рассматриваете жизнь с позиций среднего арифметического, то у вас есть только некое представление о том, что такое “нормальный человек”. Но на самом деле такой “нормальный человек” просто не существует, и в жизни нам приходится иметь дело с конкретными людьми. И конкретному человеку, а не бесчисленным массам, приходится иметь дело с последствиями принятых решений».

Статистический подход, данные для которого по сути представляют «агрегаты» (суммы, количества, средние), убирает «слабые» сигналы индивидуальности. Для алгоритма человек — всего лишь строка с несколькими цифрами и ID. Все остальное не нужно, эти «фичи» не важны, потому что конкретная модель не смогла извлечь из них выгоду. Я размышлял над моделями машинного обучения, они все слишком обобщают и упрощают, часто видят лишь черное и белое, не различая оттенков. В итоге все сводится к банальности, к скору (score), к баллам на выходе модели, на основании которых принимают решение — дать кредит или не дать и т. д. Это касается моделей машинного обучения.

Еще один недостаток статистического подхода — измерение, которое лежит в его основе. Об этом пишет в своей книге «Тирания показателей» [18] Джерри Миллер, ученый, автор многочисленных статей для New York Times и Wall Street Journal:

«Есть вещи, которые можно измерить. Есть вещи, которые полезно измерять. Но поддающееся измерению не всегда оказывается тем, что нужно измерять. Измеряемое может не иметь никакого отношения к тому, что мы на самом деле хотим узнать. Затраты на измерение могут превышать приносимую пользу. Измерения могут отвлекать нас от действительно важных вещей».

Бездумное внедрение количественных показателей везде, где только можно, — зло. Я помню, как в школе на уроках физкультуры нас гоняли по нормативам. Вы тоже бегали на скорость стометровку и прыгали в длину? Но при этом никто не прививал культуру тренировок и привычку к ежедневной физической активности. Соответствие абстрактным нормативам оказалось важнее не только твоего личного прогресса (все мы разные — усреднять нельзя!), но и любви к спорту — а это в корне неправильно. Помню, читал пост выпускника Физтеха в соцсети: «1987 год. Мы уже поступили... А потом была какая-то контрольная по физкультуре. Надо было на время переплыть физтеховский 25-метровый бассейн. Заставили всех, а потом вывесили результаты. Помню, как я их изучал: 30 сек, 35 сек, 1 мин, 2 мин, 5 мин... Последней строкой значилось: “сошел с дистанции”. Куда сошел?»

Все мы знаем про «палочную» систему в силовых органах, которая доводит до абсурда. Саша Сулим, автор книги «Безлюдное место. Как ловят маньяков в России» [8], посвященной знаменитому делу ангарского маньяка, пишет в ней о том, как их на самом деле не ловят — милиция много лет не связывала убийства женщин в серию, игнорируя очевидные факты, чтобы избежать в отчетах «висяка» и непойманный маньяк не портил статистику раскрываемости.

Но хотя количественные оценки — это плохо, никто пока не придумал ничего лучше. И надо признать, что методы этих оценок эволюционируют, усложняясь. Десять лет назад я (как, вероятно, и большинство моих коллег), оценивая эффективность сайта, фокусировался на конверсии и лишь потом начал обращать внимание на другие метрики: средняя выручка на посетителя сайта, средняя стоимость заказа, среднее число товаров в заказе и даже маржа. Одновременно эти показатели нужно делить по верхним категориям товаров и группам пользователей (если достаточно данных). Одной количественной метрики — конверсии — оказалось недостаточно: экономика интернет-магазина сложнее.



# 3

## СТРОИМ АНАЛИТИКУ С НУЛЯ



В этой главе я изложу свой подход к построению аналитики в компании с нуля. За всю мою карьеру в найме я делал это дважды — в Ozon.ru, Wikimart.ru и один раз как сооснователь — в компании Retail Rocket. И еще помог сделать это нескольким компаниям в режиме консультирования, заодно поучаствовав в найме сотрудников.

## ПЕРВЫЙ ШАГ

Когда передо мной стоит задача сделать аналитическую систему или существенно расширить ее возможности, я всегда использую двусторонний подход: определяю, какие задачи и вопросы перед нами стоят, и выясняю, какие данные есть в источниках.

Чтобы сформировать список задач, необходимо провести интервью со всеми потенциальными потребителями информации, кого это может коснуться. Создавая дизайн системы для пользователей, нужно знать ответы на следующие вопросы:

- Какие метрики понадобится считать?
- Какие дашборды собрать?
- Какую информацию отправить в интерактивные системы?
- Будут ли тут задачи ML (машинное обучение)?

Сложность этого шага в том, что потребители (заказчики) не всегда представляют, какая именно информация им понадобится. И для того чтобы выстроить эффективную систему, аналитику необходимо самому обладать хотя бы минимальной экспертизой в том бизнесе, который он анализирует. После работы в интернет-магазинах мне поначалу было непросто в Ostrovok.ru (система бронирования отелей) — да, продажи идут тоже через интернет, но тут понадобились очень специфические знания отельного бизнеса. Ваша собственная экспертиза помогает вам во время интервью с заказчиком задавать правильные вопросы и на основе ответов формировать структуру данных, которые понадобятся для решения задач клиента.

Затем я иду к разработчикам и начинаю узнавать, а что же, собственно, у них есть — какие данные они собирают и где эти данные находятся. Во-первых, меня интересуют данные, которые помогут решать задачи клиента (мне важно увидеть не только схемы, но и живые примеры таких данных — строки таблиц и файлов). Во-вторых, для меня важны те данные, которые есть, а применения им пока нет — какие задачи они могли бы решить? К финалу этого этапа у меня уже есть:

- Список вопросов, которые покрываются текущими данными.
- Список вопросов без данных и понимание того, сколько усилий потребуется, чтобы их получить.
- Данные, которые пока не решают никаких актуальных задач.
- Источники данных и их примерные объемы.

И это только первая итерация. С этим списком я иду к заказчикам, общаюсь с теми же людьми, объясняю им, можно ли ответить на их вопросы, нужны ли дополнительные данные — а потом снова иду к разработчикам. Выглядит как челночная дипломатия, но именно так я и строю план проекта.

В итоге у меня есть: список требований к системе, список имеющихся данных и задач, которые нужно выполнить, чтобы получить недостающие цифры. Выглядит просто, но бывает, что на эти шаги уходят недели. Я не выгружаю бездумно все данные из хранилища, чтобы потом начать с ходу пытаться делать метрики и дашборды. Но пытаюсь решить эту задачу в уме. Это мне экономит силы, а заказчикам сэкономит нервы. Они заранее будут знать, что получится сразу, а что нет.

## ВЫБИРАЕМ ТЕХНОЛОГИИ

Это будет моим вторым шагом. Правильный технологический стек избавит вас от головной боли на несколько лет вперед. Детально технологии я буду обсуждать в следующих главах. Сейчас обри-

ую общую картину. Примерный список вопросов к технологиям звучит так:

- Собственное хранилище или облачное?
- Использовать ли open-source-технологии?
- Какой язык программирования использовать для артефактов инженерии?
- Можем ли отдать разработку аналитики стороннему подрядчику?
- Какую отчетную систему выбрать?
- Требуется ли где-нибудь скорость анализа, близкая к real-time?

Это самые базовые вопросы, но от них зависит многое. В том числе каких сотрудников нанимать, сколько придется инвестировать, как быстро запустится проект.

Насчет хранилища данных у меня обычно следующее правило: если компания собирается зарабатывать на данных существенную часть своей выручки, то лучше собственное хранилище. Если для компании аналитика — вспомогательный проект, то лучше использовать облачное хранилище.

Цель работы коммерческой компании — прибыль. Прибыль является разностью выручки и затрат, куда входит и себестоимость хранилища. И может быть довольно большой, если данные хранятся в облаке. Ее можно оптимизировать, создав собственное хранилище. Да, тут будут затраты на администрирование. Внимания такая система будет требовать больше. Но и способов снизить затраты у вас будет явно больше, система будет намного гибче. Если же аналитическая система не имеет такого прямого влияния на P&L (прибыли и убытки), то гораздо проще будет работать с облачным хранилищем. Тогда вам не придется думать об отказавших серверах — «облака» сделают за вас свою работу сами.

Технологии open-source (свободно распространяемое ПО с открытым исходным кодом) имеют очень большой вес в аналитике. Впервые я столкнулся с ними, когда учился на Физтехе. На втором курсе у меня появился компьютер, он имел очень слабую производительность даже по тем временам, поэтому я установил туда Linux. Часами компилировал ядро под свои нужды, учился работать в консоли. И это пригодилось мне ровно через десять лет. Именно тогда я посетил офис компании Netflix в Лос-Гатосе (Калифорния) и познакомился с директором по аналитике Эриком Колсоном. Он рассказал тогда об инструментах, которые используют его сотрудники в работе, и даже нарисовал маркерами на доске их названия. И как раз он много говорил об открытом ПО для анализа данных, таком как Python, Hadoop и R. До этого я пользовался только коммерческим софтом, но несколько месяцев спустя по следам этой встречи, летом, в пустом офисе, когда все сотрудники офиса Wikimart.ru отправились на корпоратив, я написал первые 9 строчек кода на языке Pig для платформы Hadoop (тут мне пригодилось знание Linux). На это ушло 4 часа. Тогда я еще не знал, что через несколько лет именно на этом языке и на этой платформе будет написан «мозг» рекомендательной системы Retail Rocket. К слову сказать, вся аналитическая система RR, как внутренняя для принятия решений, так и вычислительная для расчета рекомендаций, написана с использованием только open-source-технологий.

Сейчас, оборачиваясь в прошлое, я могу сказать, что Retail Rocket — это самое крутое, что я сделал в своей карьере: компания быстро вышла в прибыльность, успешно конкурирует с западными аналогами, и сейчас там работает больше сотни сотрудников по всему миру с основными офисами в Москве, Тольятти, Гааге, Сантьяго, Мадриде и Барселоне. Российская компания развивается и создает рабочие места за рубежом! Сейчас вектор развития изменился: RR продает не только рекомендательную систему, но и много сопутствующих услуг для интернет-магазинов. Технологии анализа больших данных и машинного обучения, которые

мы создали в далеком 2013 году, актуальны до сих пор, и я очень горд, что мы были на голову выше наших конкурентов в технологическом плане.

Когда стоит связываться с коммерческим ПО? Ответ: когда на это есть деньги. Практически у любого коммерческого ПО есть open-source-аналог. Да, как правило, они хуже, особенно в каких-то деталях. Например, я так и не нашел достойный open-source-аналог для OLAP-кубов. Отчетные системы тоже выглядят недоделанными. Но что касается инженерных технологий, таких как Hadoop, Spark, Kafka, — то это очень надежные и мощные инструменты разработчиков. Они очень хорошо зарекомендовали себя в коммерческом применении.

Обсудим языки программирования, которые будут использоваться при разработке системы. Мой принцип — чем их меньше, тем лучше. До Retail Rocket мне удавалось обходиться одним SQL. Правда, для перекачивания данных (ETL) из источника в хранилище приходилось использовать специальные коммерческие инструменты от Microsoft. В Retail Rocket в свое время использовалось аж четыре языка программирования для создания рекомендаций: Pig, Hive, Java, Python. Потом мы заменили их все на Scala, так как он относится к семейству JVM, на котором написана Hadoop. Поэтому на нем очень легко программировать на платформе Hadoop/Spark, для последней он еще является родным. Но пару лет назад мы стали использовать Python и SQL. Здесь пришлось отойти от Scala — некоторые вещи на нем делать было неудобно.

Scala — прекрасный и изящный язык программирования, но мы уперлись в две проблемы. Во-первых, пользователям очень сложно было бы работать с ним в качестве интерфейса к данным, для этого намного лучше подходит SQL. Во-вторых, все современные библиотеки машинного обучения сейчас пишутся на Python. Сейчас Scala используется для разработки центрального ядра системы, агрегации и доставки данных, SQL для отчетов, Python для разработки моделей машинного обучения и несложных про-

тотипов. Обычно выбор языка программирования зависит от нескольких вещей:

- для какой системы он будет использоваться (например, SQL идеально подходит для баз данных);
- есть ли специалисты по этому языку в вашей компании и на рынке.

Например, заставлять пользователей вашей системы учить сложные в освоении языки программирования для доступа к данным — плохая идея. Для пользователей это вспомогательный инструмент, и много времени на его изучение они тратить не захотят.

Специалисты на рынке — моя головная боль. Scala — очень редкий язык, довольно непростой в изучении. Специалистов на рынке очень мало, а имеющиеся стоят дорого. Вот на Python работают очень многие. Хотя за одного Scala-разработчика я бы дал трех на Python. Здесь мы приняли сознательное решение: качество нашей работы для нас важнее, поэтому выбрали Scala. Нанимать готовых Scala-людей почти не получалось, поэтому мы сделали свой курс молодого бойца [19], когда новичок в течение полугода обучается программировать на нем.

## ПОГОВОРИМ ОБ АУТСОРСЕ

Обсудим возможность привлечения внешнего подрядчика для создания аналитической системы. Ему на откуп можно отдать разные аспекты:

- создание и поддержка технической части системы;
- аналитическая часть;
- выделенные задачи.

Когда требуется сократить время развертывания технической части проекта и получить качественный результат — нужен хороший подрядчик. Но попробуй его еще найди! Мало того что редкий подрядчик достаточно глубоко знает предмет — ситуация часто усугубляется тем, что заказчик не знает, чего хочет.

В одной из компаний, где я работал, была собрана команда для реализации проекта. Проект не аналитический, в теории он выглядел замечательно. К тому же командой руководил человек, который преподавал проектирование таких систем чуть ли не в топовом университете. Для технической реализации были выбраны самые «современные» технологии. В итоге три или четыре разработчика писали эту систему целый год. В попытке запустить ее потратили целые сутки... Не завелось, и всю систему выбросили на свалку. То же самое может случиться и с аналитикой. Теория очень сильно отличается от практики, тем более в нашем быстро меняющемся мире.

Риск уменьшится, если привлечь очень опытного аналитика, который не раз лично реализовывал подобные проекты. На вашем проекте он будет выступать в качестве независимого советника или даже арбитра. Это нужно, чтобы, с одной стороны, «приземлить» заказчика, с другой — ограничить подрядчика. Я считаю, что проект на старте лучше сильно урезать по «хотелкам», чтобы получить на выходе работающую версию как можно быстрее. На то есть несколько причин. Во-первых, после того как вы, заказчик, вживую поработаете с ней, вам гораздо легче будет сформулировать, что вы действительно хотите. Это тяжело делать абстрактно на бумаге, конструируя сферического коня в вакууме. Вторая причина — драйв, лично для меня это очень важно. Когда время течет медленно, у команды, да и у заказчиков, постепенно угасает интерес. И на выходе мы

уже получаем вымученный проект, которым уже не так сильно хочется заниматься.

Если нет возможности найти советника — попробуйте хоть немного разобраться в вопросе самостоятельно, почитайте книгу, посмотрите видеозаписи конференций. Иначе велика вероятность, что проект просто не взлетит. А если и взлетит, то будет потрачено много времени и денег.

Хорошо, если можно отдать на аутсорс технологическую часть, но можно ли это сделать с аналитикой? Общий ответ — нет. Сторонние аналитики никогда не будут обладать всей полнотой бизнес-контекста. С другой стороны, аутсорс аналитики какого-то направления вполне возможен. Например, рекламного.

Еще один вариант аутсорса — отдать какую-то часть проекта целиком: вы отдаете данные, а на выходе получаете готовый продукт. Пример такого сотрудничества — компания Retail Rocket. Начали мы бизнес с товарных рекомендаций. Интернет-магазины отдавали нам данные и товарную базу, на выходе они получали готовые рекомендации. Лично у меня идея такого бизнеса зародилась во время работы в компании Wikimart.ru. Я сделал рекомендации для сайта компании и подумал: почему бы не запустить тиражируемое решение. Это бы сняло необходимость интернет-магазину нанимать инженеров машинного обучения и изобретать велосипед. Результат получался гораздо быстрее, буквально за неделю. Качество рекомендаций нашего сервиса гораздо лучше внутренней разработки. Если бы меня наняли сейчас в интернет-магазин, то, скорее всего, я бы привлек внешний сервис рекомендаций вместо того, чтобы делать собственную разработку.

Немного расскажу о своем личном опыте работы на аутсорсе. В 2009 году я ушел из Ozon.ru. В то время у меня был достаточно популярный блог по аналитике KPIs.ru, созданный за пару лет до этого. И оттуда ко мне стали приходить запросы на консалтинг по аналитике из самых разных сфер: разработчики игр, e-commerce,

венчурный фонд и т. д. Потихоньку я стал наращивать темп консультаций, одновременно работая на три компании. Первой я помог выбрать нужную технологию и нанять людей в команду, проводил собеседования. Второй — помогал растить стартапы. В третьей компании я поработал руками, подняв аналитическую систему. Мне этот опыт много дал — прежде всего я помогал компаниям, не отвлекаясь на корпоративные детали и бюрократию, как было бы, работая я в штате. Ну а компаниям моя работа позволила осуществлять быстрый старт проектов. Кстати, в третьей компании я в результате остался работать (это был Wikimart.ru): ее основатель предложил мне возглавить отдел аналитики — и я согласился, потому что в тот момент хотел быть ближе к данным и работать руками. На этом тогда закончился мой аутсорс.

## НАЕМ И УВОЛЬНЕНИЯ

Допустим, технологии выбраны, задачи понятны, есть информация по имеющимся данным. Возможно, даже «поднята» аналитическая система — бери и пользуйся. Поговорим о найме сотрудников.

Я уже описывал роли в прошлой главе. Их много, в идеале одна функция — один человек. На начальном этапе обычно происходит совмещение ролей: аналитик может и данные выгрузить, и ML-модель собрать. Я никогда не нанимаю лишних людей и придерживаюсь теории бережливого стартапа. Лучше последовательно нанимать и расширять отдел, чем бездумно нанять много людей, а потом не понимать, что с ними делать.

Будем считать, что вы определились со списком необходимых сотрудников. Теперь поделюсь своим опытом найма — за свою карьеру я собеседовал сотни специалистов, и у меня в голове есть некоторая картинка без лишних подробностей (оставим их HR-отделу). При найме любого сотрудника для меня прежде всего важно, чтобы кандидат был здравомыслящий, ищущий развитие и разделяющий мои ценности. Младших аналитиков, джуниоров,

стажеров на неполный рабочий день иногда получалось найти через групповое интервью. Делается это следующим образом. Даются объявления, в том числе в вузах, через вакансии. Рекрутер обзванивает кандидатов и приглашает всех собраться в одно время в один день. Сама встреча делится на несколько частей:

1. Вводное слово — рассказ о компании, работе и т. д. Пятнадцати минут достаточно.
2. Групповая работа — ребят и девушек разбиваем случайным образом на группы по 3–4 человека. Им дается простое аналитическое задание. Они обсуждают его группой в течение 30 минут — в это время рекомендую подходить к ним, слушать, как они рассуждают. Далее кто-то из группы озвучивает свое решение.
3. Индивидуальное задание — нужно предложить подход или решение к какой-либо задаче, можно письменно. На задание — полчаса.

В результате за два часа у вас сложится картина. Вам будет понятно, кого стоит дальше смотреть, а кого нет. Все познается в сравнении, вы сразу сравните кандидатов между собой, и это очень удобно. В следующий этап попадают несколько человек, с ними уже проводятся индивидуальные собеседования. С помощью этой несложной схемы я успешно нанял нескольких стажеров в двух компаниях. С ними я долго работал, почти все они выросли в отличных специалистов. Это была моя лучшая инвестиция времени в наем.

Со специалистами сложнее. В такую группу их не собрать, требования к их квалификации выше. А еще на рынке труда существует серьезный перекосяк. Совсем недавно мне нужно было нанять двух человек: инженера по данным и аналитика данных. Как вы думаете, на какую вакансию откликнулось больше кандидатов? Задам еще один вопрос: кого у нас в стране больше — гитаристов или барабанщиков? Я трижды играл на шоу #ROCKNMOB — это такой масштабный флешмоб для музыкантов-любителей: собирается толпа вокалистов, басистов, гитаристов и ударников, и банда

из трех сотен человек пилит рок-хиты, от Queen до Rammstein. На одно из шоу было заявлено 27 ударников и 151 гитарист. Эта статистика более-менее отражает распределение сил в природе: парень с гитарой — это сексуальный архетип (я уже написал, что играю на электрогитаре?), и выглядит он всегда круче барабанщика. А еще гитару купить проще, чем барабанную установку. Инженеры по данным проигрывают аналитикам в еще более грустной пропорции: 95 % откликов приходит на вакансию data scientist. Они прямо как гитаристы! При этом большинство имеют крайне низкую квалификацию и очень скромный послужной список, но чувствуют себя опытными «сержантами». В этом тоже виноват хайп!

Аналитиков данных я собеседую так: делаю первым звонок на 15 минут, задаю несколько несложных вопросов на понимание концепции машинного обучения. Если все ок, приглашаем на собеседование. Первое собеседование делится на две части: полчаса общаемся на тему машинного обучения, от азов до более сложных вещей. Во второй части задаем инженерные вопросы, например, какие-то вещи делаем на SQL. Потом устраиваем еще одно собеседование — решаем простейшую задачу машинного обучения. Буквально — садимся вместе за один компьютер, и кандидат выполняет задание, а я в это время задаю вопросы, чтобы убедиться, что он понимает, что и почему делает, действительно ли кандидат — практик. Обычно это сразу видно по скорости написания кода. В целом этих собеседований достаточно, чтобы оценить человека и сделать ему оффер.

Тема увольнения обычно стыдливо замалчивается, но оно даже важнее найма. Популистские высказывания в духе «нанимай медленно, увольняй быстро» я не поддерживаю. К сотрудникам нужно относиться по-человечески. Расставаться тоже нужно по-человечески, это важная часть корпоративной культуры. Увольнения происходят с двух сторон: по инициативе сотрудника и по инициативе работодателя. В моей практике первых было больше. Главная причина — мало машинного обучения, а ведь на курсах рассказывали, что этого будет много. Наука сильно расходится

здесь с жизнью. Не устаю повторять, что реального машинного обучения в проектах машинного обучения 5–10 % времени. После такого опыта я стал целенаправленно отсеивать таких кандидатов-мечтателей на этапе собеседования. Вторая причина — сотрудник сильно вырос или устал долго работать на одном проекте. В таких случаях я обычно помогаю ему найти новое место работы, используя свои связи.

Причины уволить сотрудника могут быть разными — откровенно лажает, не вписывается в нашу аналитическую культуру. Но я никогда не тороплю события, ведь я также могу ошибаться. Для начала советуюсь с командой, с каждым отдельно. Если получаю негативные отзывы — это практически всегда означает, что нужно расставаться. Можно попробовать поговорить, подкинуть проекты, но обычно это не работает. Я наблюдал за карьерой уволенных и обратил внимание, что часто эти сотрудники находят нормальную работу и приживаются там. То есть они не были плохими — просто они не подошли нам, и это нормально.

## КОМУ ПОДЧИНЯЮТСЯ АНАЛИТИКИ

В идеале аналитики должны быть независимы от менеджеров, которых они оценивают. Тут принцип — кто платит, тот и музыку заказывает. Не может сотрудник менеджера объективно оценить его работу. Решать задачи отдела может (помните про децентрализацию из прошлой главы?), но оценивать — нет. Здесь нужна независимость от операций. Я бы рекомендовал, чтобы центральный аналитический отдел подчинялся генеральному директору, финансам или ИТ. Список дан в порядке приоритета. У меня был опыт подчинения генеральному директору, директору по маркетингу и ИТ. Первый вариант был самым лучшим опытом — внешнее давление минимально. Но в этом есть и проблема: как правило, менеджеры не знают, как управлять аналитикой, а генеральному директору еще и времени не хватает. Руководителю аналитики придется проявить недюжинную самостоятельность. Я лично

получал задания в духе: «найди что-нибудь интересенькое». Эту книгу я начал писать в том числе и для того, чтобы ее прочитали топ-менеджеры, которым подчиняется аналитика. Мне бы этого очень хотелось!

## ДОЛЖЕН ЛИ РУКОВОДИТЕЛЬ АНАЛИТИКИ ПИСАТЬ КОД

Я всегда любил роль играющего тренера — управлять небольшой командой людей, учить стажеров, но при этом самому делать задачи. Скажу без ложной скромности: моя команда в компании — это всегда отряд спецназа, который решает сложные задачи за очень ограниченное время. Обычно в самом начале я всегда проектировал аналитическую систему и писал базовый код для ее фундамента наравне со всеми — всё как в стартапе. В какой-то момент я чувствовал, что ребята стали уже сильнее меня и можно им делегировать не только отдельные задачи, но и целые направления. Правда, что-то я продолжал делать сам — не хватало решимости отдать код полностью в чужие руки, и не хотелось, чтобы мои технические навыки полностью атрофировались. Но потом обстоятельства заставили меня посмотреть на работу руководителя аналитики с другой стороны.

Я попал на собеседование в Quora на должность менеджера. Послушав, чем я занимаюсь, директор по аналитике Шавье Аматриан (Xavier Amatriain) мне прямо сказал: ты ни то ни сё — не менеджер и не разработчик. И не принял меня на работу. Этот сигнал заставил меня задуматься: за двумя зайцами погонишься — ни одного не поймаешь. Что на самом деле очень сложно

совмещать работу сотрудника и менеджера и при этом быть эффективным во всем.

Однажды мне на глаза попался ответ Эрика Колсона (который тогда руководил аналитикой Netflix) на Quora [20]:

«...главная задача менеджера — наем (это действительно непросто — найти отличного аналитика данных). Далее организация команд — не только аналитиков, но и работа с другими командами в организации (Продукт, Инженеры, Маркетинг и т. д.). Затем коммуникация, координация, наставничество и т. д. Для менеджера не остается времени для решения аналитических задач, и, следовательно, это делегируется. Технические навыки лидера команды атрофируются».

И это действительно так. Вначале привлекает магия черных ящиков алгоритмов, потом хочется большего, ты становишься менеджером — и всему приходит конец, магия становится рутинной. Ты видишь только метрики, но код становится для тебя все менее читабельным. Моя история как раз об этом — роль играющего тренера очень нужна и полезна, но только на старте, в какой-то момент нужно делегировать все — иначе вы будете неэффективно делать и то и другое. Еще один факт — код или любые задачи, которые руководитель делает как исполнитель, обходятся гораздо дороже. Поначалу в Retail Rocket, собрав первую команду, я отошел от программирования к проверке (тестированию) всех выполненных ею задач. Потом, поддавшись влиянию партнера, вернулся к программированию — о чем сейчас жалею. Я согласен с Колсоном — менеджер на определенном этапе должен полностью отказаться от программирования и самостоятельного выполнения задач.

Важный аспект — мотивация менеджера. Я люблю цитировать конспект лекций «You and your research» [21] Ричарда Хэмминга. Ричард работал в лаборатории Bell Labs, в том числе с Клодом Шенноном (основателем теории информации). Как и многие знаменитые ученые того времени, Хэмминг работал над проектом первой атомной бомбы США. Сама лаборатория была очень

мощной: там изобрели первый транзистор, ученые лаборатории получили семь Нобелевских премий в разных областях. Его попросили сравнить исследовательскую работу и менеджмент, и вот что он ответил:

«Если вы хотите быть великим исследователем, вы не станете им, будучи президентом компании. Другое дело, если вы хотите быть президентом компании. Я не против того, чтобы быть президентом компании. Я просто не хочу. Я думаю, Иан Росс делает хорошую работу в качестве президента Bell Labs. Я не против этого; но вы должны четко понимать, чего хотите. Еще, когда вы молоды, вы можете захотеть быть великим ученым, но прожив больше, вы можете изменить мнение. Например, я пошел однажды к своему боссу, Боду, и спросил: “Почему ты вообще стал главой департамента? Почему ты не остался просто хорошим ученым?” Он ответил: “Хэмминг, у меня было видение того, какой должна быть математика в Bell Laboratories. И я понимал, что, чтобы это видение воплотилось, это должен был сделать я; я должен был быть главой департамента”. Когда вы можете в одиночку воплотить то, что хотите, тогда вам следует этим заниматься. Как только ваше видение того, что, как вы считаете, должно быть сделано, больше того, что вы можете сделать в одиночку, вам надо двигаться в менеджмент. И чем больше видение, тем дальше в менеджмент вам надо идти. Если у вас есть видение того, какой должна быть вся лаборатория или вся Bell System, вам надо идти туда, чтобы это осуществить. Вы не можете это осуществить легко снизу.

Это зависит от ваших целей и желаний. И по мере их изменения в жизни вы должны быть готовы меняться. Я выбрал избегать менеджмента, потому что предпочитал делать то, что могу делать в одиночку. Но это выбор, который сделал я, и он субъективен. Каждый человек имеет право на собственный выбор. Пусть ваш ум будет открыт. Но когда вы выберете путь, ради всего святого, осознавайте, что вы сделали и что решили. Не пытайтесь делать и одно, и другое».

Есть один минус полного перехода в менеджмент — атрофия технических навыков. Менеджеру будет сложно вернуться в работу руками, а такое иногда бывает, например, когда создают стартап. По этому поводу Колсон написал [20], что «истинный лидер ана-

литики никогда не утратит любопытства и поздними вечерами или утром в выходные может самостоятельно покопаться в данных, строить простейшие графики и самостоятельно сделать выводы. Это даст ему понимание данных, которое нельзя получить никаким другим способом, кроме как погрузив туда руки». Что повысит вашу самооценку и поможет мотивировать ваших сотрудников быть любознательными и любить свою профессию. Я очень люблю свое ремесло, даже когда управляю чем-то, мне всегда интересно посмотреть код — как это работает внутри черного ящика.

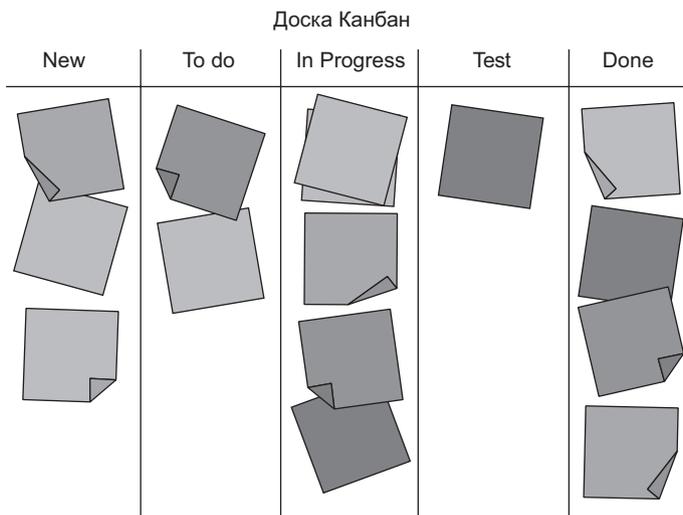
## УПРАВЛЕНИЕ ЗАДАЧАМИ

Почти все задачи аналитики делятся по направлениям, и к каждому нужен свой подход:

- Инженерные задачи.
- Найти причину какого-либо явления (инсайт).
- Проверить гипотезу или провести исследование.

Инженерные задачи включают в себя разработку дашбордов, метрик, добавление источников данных, оптимизацию вычислений, решение вопросов технического долга. Все эти задачи объединяет один очень важный фактор — у них четкий и понятный результат и, как правило, легко прогнозируемая трудоемкость. Их можно щелкать, как орешки, последовательно меняя статус (рис. 3.1):

- Задача пришла от заказчика (New).
- Задача получила приоритет, оценку трудоемкости и поставлена в очередь на исполнение (To Do).
- Задача взята в работу (In Progress).
- Задача проходит проверку на правильность реализации (Review).
- Задача проходит тестирование заказчиком (Test).
- Задача выполнена (Done).



**Рис. 3.1.** Доска Канбан

Вся эта схема типовая и вытекает из здравого смысла. Любая задача может быть принята к исполнению или отвергнута по разным причинам (это сделать мы не можем, нужна виза руководителя). Далее команда аналитиков берет такую задачу, обсуждает ее напрямую с заказчиком, оценивает ее, например, через покер планирования [22]. Задача становится в очередь на выполнение в соответствии со своим приоритетом. Причем у нас было правило: брать первую из стопки задач. Таким образом происходит рандомизация категорий задач, и специализация сотрудников размывается.

В чем плюс такой схемы рандомизации? В том, что все сотрудники понимают, как функционирует система в целом, а значит,

могут заменить друг друга в случае отпуска, болезни или увольнения. Это частично решает проблему фактора автобуса (сколько разработчиков вашей компании должен сбить автобус, чтобы компания все еще продолжала функционировать). До Retail Rocket я не заморачивался подобными вопросами. Если кто-то увольнялся — проект этого человека приостанавливался до найма сотрудника на его место. Но в той компании, соучредителем которой я являюсь сейчас, роль аналитики больше и ответственность выше.

У рандомизации задач есть минусы:

- У сотрудников есть сильные и слабые стороны: кто-то силен в инженерии, кто-то в построении моделей. Соответственно, у инженера будут проблемы с ML-моделями, а у аналитика — с инженерией.
- У сотрудников тоже есть профессиональный и личный интерес — брать задачи определенной категории. Рандомные задачи для них становятся деструктивными.
- Подавляется инициатива — сотрудники перестают сами предлагать интересные задачи: один предложил, другой не считает ее полезной. Если она достанется второму — весьма вероятен скрытый саботаж: человек будет работать не так усердно, как тот, кто предложил задачу.

Поэтому сама схема рандомизации не является панацеей.

Когда задача выполнена, ее забирает другой сотрудник отдела, чтобы проверить правильность реализации с точки зрения инженера: например, может сделать код-ревью (code review) архитектурных решений, проверить, написаны ли программные тесты. В следующем статусе задачу выводят в бой, заказчик проверяет, все ли хорошо сделано. И только после этого задача считается выполненной.

Именно такой подход к выполнению задач мы практикуем в Retail Rocket, правда, в реальности деталей и правил у нас гораздо больше.

В нашем случае получилась смесь методологий Scrum и Kanban. Но не стоит создавать из них карго-культ. Они зависят от размера команд, специфики задач и самое главное — степени готовности команды. Я начинал с самых простых столбцов со статусами попроще для ведения задач в Trello, потом пришел к схеме выше, но и ее не считаю совершенной. Не существует единой методологии, внедрив которую вы станете полностью счастливыми, главное — придерживаясь здравого смысла.

Следующий класс задач — уже из области анализа данных: поиск инсайтов. Обычно это задачи от менеджеров или клиентов. В тексте таких задач описывается какая-либо проблема, и необходимо найти ее причину. Мы пропускали такие задачи через те же самые статусы, что и инженерные. Но у них есть одно отличие — неизвестно, найдем мы причину или нет. Конечный результат неизвестен, значит, теоретически мы можем потратить бесконечно большое время на поиск причины. Поэтому при планировании такой задачи мы указываем максимальное время, которое готовы на нее потратить.

Третий класс задач — исследовательские, куда включена проверка гипотез и проведение экспериментов. Это самые сложные (но интересные) задачи с непредсказуемым результатом. Их обожают люди, которые любят постоянно учиться и экспериментировать, это их основной мотиватор. У таких задач следующие характеристики: непредсказуемый результат и очень долгое время его ожидания.

Управление гипотезами совсем непростая штука, как кажется на первый взгляд. Например, у нас в Retail Rocket только три из 10 гипотез по улучшению рекомендаций дают положительный результат. Чтобы провести эксперимент с одной гипотезой, требуется минимум полтора месяца. Это очень дорогое удовольствие. Что обычно понимается под гипотезой? Какое-либо изменение, которое приведет к улучшению чего-либо. Обычно это рационализаторское предложение, направленное на улучшение определенной метрики. Метрика — обязательный атрибут. На старте

работы компании это была конверсия сайта (процент посетителей, которые сделали покупку). Потом мы пошли дальше: захотели повысить заработок в расчете на одного посетителя сайта (Revenue Per Visitor), увеличить средний чек покупки, среднее количество заказов в товаре и даже визуальную привлекательность рекомендуемых товаров. Рационализаторские предложения могут быть разными: от исправления ошибки в алгоритме до внедрения алгоритма машинного обучения на нейронных сетях. Мы старались все изменения алгоритмов прогонять через гипотезы. Потому что даже исправление несложной ошибки в реальной жизни может привести к ухудшению метрики.

Гипотезы, как и задачи, имеют свой жизненный цикл. Во-первых, все гипотезы нужно очень четко приоритизировать, поскольку трудоемкость огромная и результат на практике появится далеко не сразу. Ошибка в приоритизации будет дорого стоить. Я считаю, что приоритизация гипотез должна быть извне: цели должен определять бизнес. Обычно в интернет-компаниях это делает отдел продукта. Они общаются с клиентами и знают, что будет лучше для них. Моя персональная ошибка в Retail Rocket была в том, что я первые годы приоритизировал гипотезы сам. Аналитики варились в собственном соку, придумывали гипотезы, приоритизировали их, экспериментировали. Да, мы неплохо оптимизировали алгоритмы, этот задел нам пригодился в конкурентной борьбе. Но если бы мы тогда больше думали о том, чего хочет клиент, то добились бы большего. Я списываю это на то, что аналитики в какой-то момент стали слишком квалифицированными (*overqualified*) и бизнес за нами не поспевал. Оценить гипотезу, понять ее потенциальную пользу, найти баланс между трудоемкостью и ее эффектом — это искусство.

Интересно, что на Западе такие проблемы тоже актуальны. В 2016 году я подал заявку на доклад «Тестирование гипотез: как уничтожить идею как можно быстрее» [23] на международную конференцию RecSys по рекомендательным системам. Туда очень сложно попасть, все доклады проходят инспекцию несколькими учеными. Предыдущую нашу заявку на доклад [24] отклонили, но

в этот раз моя тема оказалась достаточно актуальной, чтобы доклад приняли в программу. Я выступил в концертном зале MIT в Бостоне. В докладе был рассказ о том, как мы проверяем гипотезы. Помню, что страшно волновался, текст учил чуть ли не наизусть. Но все прошло хорошо, я даже получил лично положительный отзыв от Шавье Аматриана, экс-руководителя аналитики Netflix, он был одним из организаторов конференции. Тогда Аматриан пригласил меня на собеседование в офис компании Quora, топ-менеджером которой он был в то время — видимо, мой рассказ о тестировании гипотез произвел впечатление.

## КАК УПРАВЛЯТЬ РОМАНТИКАМИ

Идеальный менеджер в моем представлении:

- идет напрямую к цели;
- относится человечно к людям;
- делает из любого хаоса, даже творческого, рутину;
- удовлетворяет страсть сотрудников к интересным и развивающим задачам.

На последнем пункте я бы хотел остановиться подробнее. В прошлой главе я описал конфликт исследователя и бизнеса: исследователь хочет сделать что-то значимое, используя самые последние разработки ML, бизнесу часто это не нужно. Как этим можно управлять? В нашей работе аналитиков и инженеров машинного обучения создание алгоритма занимает 5–10 % времени, а остальные 90 % уходят на то, чтобы заставить новый алгоритм приносить прибыль. Этот конфликт — основная причина, по которой я терял сотрудников.

Консервативный бизнес не хочет оплачивать дорогостоящие исследования с непонятным результатом. Чем крупнее компания, тем ей проще это делать; в больших компаниях есть даже такая должность — инженер по исследованиям (research scientist). Но с ними

другая проблема — наука есть, а жизни нет: не видят исследователи реального применения, и это их демотивирует. Поэтому важно найти баланс. Обсудим роль менеджера аналитики в его достижении.

Как известно, бизнес должен быть устойчивым по отношению к персоналу. Эта устойчивость достигается автоматизацией, «автобусным» числом, построенными процессами — когда все творческое, хаотическое оборачивается в процессы и становится рутинной. Когда я пишу эти строки, представляю сборочный конвейер, у которого нет души, а люди подходят и на разных этапах крутят разные гайки. Все задачи максимально приземленные. И вот когда все отлажено до состояния идеально смазанной машины, нужно впускать в поток интересные для ваших сотрудников задачи. Мне лично всегда очень непросто их найти. Требования к ним я бы предъявил следующие:

- в перспективе результат должен принести пользу;
- техническая поддержка задачи может работать без ее создателя.

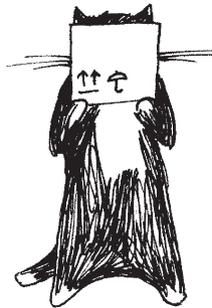
Иначе исследователь слепит огромный космический корабль, который не полетит, а после увольнения исследователя на его место придет другой и будет строить корабль заново. В любом случае лучше относиться к таким проектам как к венчурной инвестиции. И обычно результат достигается только тогда, когда уже вся компания подключилась к проекту, — а это уже посерьезнее, чем просто исследование.

Приведу пример. Один из внутренних продуктов компании Retail Rocket в аналитике по NLP-анализу (анализ семантики текста) написан на глубоких нейронных сетях (Deep Learning). Проект, который начинался как «игрушечный», оказался довольно слож-

ным и интересным с точки зрения развития. В процессе работы удалось доказать его эффективность, и сейчас он в строю. Бывали и проблемные проекты. Например, мы пытались сделать сопутствующие товары («С этим товаром часто покупают») для магазинов одежды, опираясь на стилевую сочетаемость [25]. Для проекта использовались сиамские нейронные сети, и у нас все получилось с точки зрения визуальных образов. Провели тестирование на коммерческих сайтах — улучшений не увидели. Пришлось признать гипотезу неудачной.

# 4

## ДЕЛАЕМ АНАЛИТИЧЕСКИЕ ЗАДАЧИ



Box & Plots

Глава состоит из двух частей: сначала я расскажу о самой организации анализа данных в виде задач, а потом непосредственно об анализе данных.

## КАК СТАВИТЬ ЗАДАЧИ АНАЛИТИКАМ

В прошлой главе я написал про доску статусов задачи. Сейчас мы подробно рассмотрим сами задачи. В идеале в тексте задачи перед ее планированием должны быть такие атрибуты:

- инициатор;
- причина возникновения задачи;
- ожидаемый результат;
- требуемые сроки и приоритет.

Инициатор — лицо, которому нужны результаты задачи для принятия решения. Очень важно, чтобы это не был посредник. Обычно руководители любят присылать такие задачи от лица своих сотрудников, в лучшем случае заместителей. Но лицом, принимающим решение, будет сам руководитель. Планирование исполнения задачи (трудоемкость, сроки и результат) — это переговоры, во время которых часто задача сильно меняется или даже отменяется, если появляются более простые пути ее решения. Часто у самого посредника не хватает власти, чтобы принять решение самостоятельно. Как вы думаете, можно ли договориться с посредником сразу? Нет, он пойдет к своему руководителю и потом вернется. Такой «футбол» отнимает много времени. Нужен ли инициатор задачи на таких переговорах при планировании? Да, потому что любая профессиональная коммуникация — это заключение контракта, а контракт подразумевает переговоры. Инициатор хочет получить от аналитиков обязательства по выполнению задачи, на которую они потратят много времени: часы и даже дни. Так почему же инициатор не может потратить 10 минут своего времени, чтобы договориться лично? Для меня это сигнал, что задача не так важна, и лучше заняться более приоритетными вещами.

Причину возникновения задачи необходимо обозначить. Она пригодится всем: инициатор осознает ее, аналитики понимают контекст, а значит, быстрее найдут способы решения. Есть еще один фактор — все могут попросту забыть эту причину. И когда через продолжительное время необходимо поднять результаты задачи, будет намного проще, если причина была описана в ней.

Ожидаемый результат — это форма ответа, которая устроит инициатора задачи. Результаты могут быть разными: просто текст причины с обоснованием, таблица с данными, графики, выгрузка данных для внешней системы. На встрече планирования инициатор должен объяснить, почему ему нужны результаты именно в таком виде и что он потом с ними сделает. Или хотя бы сообщить, что это его личное предпочтение для принятия решения. От формы результатов зависит трудоемкость задачи. Одно дело — написать короткое сообщение с парой цифр, другое — сделать большой отчет с графиками и выкладками. Обычно задачи для внутреннего пользования выглядят проще, чем, например, отчеты для клиентов.

Требуемые сроки и приоритет позволят тщательнее выстроить очередь исполнения задач. Никто не любит задачи, которые нужно было выполнить вчера, особенно если поставлены они были сегодня. Это и есть качество менеджмента: подумать и поставить задачу заранее. По соотношению таких задач можно судить об управленческих качествах менеджмента. На своей практике я часто видел, как такие задачи «перегорали» и были уже не интересны заказчику после их выполнения.

Давайте рассмотрим два примера задач: хорошая постановка и плохая. Начнем с хорошей. По электронной почте приходит письмо, текст задачи хорошо формализован:

- *Инициатор*: коммерческий директор Иванов И. И.
- *Причина*: продажи направления «Игрушки» упали, эта категория сильно отстает от плана. Возможная проблема в недостаточных расходах на рекламу.

- *Ожидаемый результат:* причина падения продаж — текстом с обоснованием причины цифрами.
- *Сроки:* готовы ждать 5 рабочих дней, иначе упустим время для принятия решений.

Здесь все очень четко — исполнителя вводят в курс дела, обозначают проблему и даже указывают возможную причину, сотруднику понятно, зачем он выполняет задачу. Ему доверяют и считают его профессионалом.

Пример плохой задачи:

- Инициатор: Сидоров А. по поручению Иванова И. И.
- Пришлите мне распределение продаж категории «Игрушки» по рекламным каналам как можно быстрее.

Здесь все плохо: есть посредник, нет причины, срок — вчера. Инициатор абсолютно уверен, что сам знает, в чем причина, и не считает нужным посвящать сотрудника в детали. В результате исполнитель оторван от контекста и просто не понимает, зачем он должен это делать. Конечно, аналитики выполняют эту задачу, но, скорее всего, она вернется, так как причина была не та, и гипотеза оказалась неверна. Такая постановка задач не оставляет пространства для творчества, а я по себе знаю, что это может очень демотивировать — чувствуешь себя калькулятором. Конечно, есть люди, которых устраивает такой подход. Но лучших сотрудников так не удержать. Они будут искать себе другое место, в котором полностью реализуют свой потенциал.

Планирование задач [22] — важный процесс, который может выглядеть по-разному: планировать может руководитель аналитики или вся команда. Можно делать это в текущем режиме, планируя

задачи по мере поступления, а можно и периодически, накапливая пул задач. Все эти способы я опробовал и теперь уверен, что лучше, когда в планировании участвуют все, как в Retail Rocket [22], и у этой встречи есть четкие календарные рамки. Мне лично бывает непросто спорить со своими же сотрудниками о вариантах и сроках исполнения. Часто хочется единолично принимать решения. Но есть формула — чем сильнее вы сами, тем лучших сотрудников вы нанимаете, тем больше свободы в принятии решений вы им даете. Так формируется команда профессионалов, у которых всегда будет возможность высказаться.

На встречах по планированию задач в Retail Rocket мы включаем диктофон. Аудиозаписи дисциплинируют и помогают решать спорные вопросы. Но еще лучше все договоренности прописывать прямо в тексте задачи — это гарантирует, что все всё поняли правильно, особенно если согласию предшествовали жаркие споры.

## КАК ПРОВЕРЯТЬ ЗАДАЧИ

Чтобы проверить задачу, нужно вспомнить, какие артефакты мы можем получить:

- инсайт, ответ на вопрос почему;
- автоматизированный отчет (дашборд);
- ML-модели;
- код системы анализа данных.

Почти все эти задачи объединяет наличие программного кода. Исключением может быть разве что инсайт, для поиска которого порой достаточно обычного Excel, а программирование могло не потребоваться.

Для проверки программного кода проводится код-ревью (code review). На этом этапе какой-либо сотрудник (не исполнитель)

изучает программный код, чтобы понять, насколько этот способ решения задачи корректен и соответствует стилистическому подходу, принятому в команде. Эта практика широко применяется в разработке ПО.

Когда пишете программу, всегда относитесь к ней как к тексту, который будет читать другой человек. Раньше, когда программу писал и поддерживал один человек, это было не так важно. Сейчас разработка ПО — это командная работа, в которой должно быть гарантировано качество. Компьютеру все равно, как выглядит ваша программа стилистически, а людям — нет. Те, кто будет работать с вашим кодом в дальнейшем — проверять его, оптимизировать скорость работы, переносить на другую платформу, — должны понимать его без лишних усилий. Если код вызывает вопросы, автора просят внести изменения так, чтобы текст стал читаемым и однозначным. Это одна из целей инспекции. Аналогичные стандарты работы действуют и в аналитике. Но есть несколько отличий от обычной разработки, расскажу о них далее.

В разработке используется система контроля версий, например Git. Через нее разработчики вносят изменения в аналитическую систему компании и проводят инспекцию. Я рекомендую весь код держать в системе контроля версий. Плюсы такого решения:

- все изменения будут прозрачны;
- в случае ухода разработчика/аналитика весь код останется у вас;
- если возникнут проблемы — легко откатить изменения, вернувшись к прошлой версии.

Инспекцию кода относительно легко сделать для всех артефактов аналитики, кроме инсайтов. С инсайтами не все так однозначно. Для их поиска и выкладок используются разные инструменты: Excel или его аналоги, графический интерфейс аналитической системы, SQL, блокноты Python или другого языка (например,

Jupyter Notebooks). В таких задачах обычно присутствует несколько этапов:

- получение данных;
- их очистка;
- анализ;
- выводы.

На каждом из этапов желательно проводить отдельную проверку. Получение данных — часто это код, например SQL, — проверить относительно легко: посмотреть, нужные ли данные были использованы. Кстати, при планировании очень полезно обсуждать, каким образом будет решаться задача, на что обратить внимание и какие данные могут понадобиться. При этом взять за основу можно похожие задачи из прошлого опыта. В процессе проверки будет легче соотнести решение задачи с тем вариантом, о котором договорились на планировании. Советую ограничивать время на такие задачи, иначе можно искать инсайт до бесконечности. Очистку данных и анализ проверить сложнее, но если там есть код, это упрощает дело.

Есть одна проблема с блокнотами (jupyter notebooks) — скрытые ошибки. В блокнотах выполняются разовые задачи (ad-hoc), и поэтому аналитики пренебрегают стандартами разработки — инспекциями кода и тестами. Как с этим бороться? Есть несколько способов проверить код и выводы.

Во-первых, проверяющий может очень внимательно просмотреть все решение на предмет ошибок. Это трудоемко, ведь по сути ему придется построить решение чуть ли не с нуля в своей голове. Во-вторых, можно воспользоваться другими источниками данных, которые хотя бы косвенно могли бы подтвердить вывод. В-третьих, можно последовать совету Кэсси Козырьков, директора по принятию решений в Google, из ее статьи «Самая мощная идея в анализе данных» [26]: сделать случайное разделение данных на два датасета (набора данных). По первому набору аналитик будет искать причи-

ну, а по второму проверяющий проверит выводы аналитика. Такой подход всегда используется в машинном обучении и называется валидацией (validation).

Хочу сделать важное замечание относительно решений, которые не используют код. В чем сложность их проверки? Представьте, что вы работаете в Excel и уже получили данные в виде файла. Вы должны загрузить его в Excel, проверить, почистить, написать формулы, построить таблицу или сводную таблицу (что удобнее для проверки). Теперь поставьте себя на место проверяющего. Часть операций в Excel делается мышью, данные можно копировать и вставлять блоками, протокола всех действий нигде нет. Чтобы посмотреть формулу — нужно кликнуть, а если таких формул много? И вы их «протягивали», а если ошиблись, исправили и не обновили все формулы? Чуть лучше с интерфейсами, где блоки выстраиваются графически и соединяются стрелками. Приходится щелкать по каждому блоку, проверять, все ли корректно. С кодом проверить все намного проще — все операции с данными написаны текстом! Не нужно никуда щелкать, все видно сразу. Еще один плюс кода — можно очень быстро пересчитать задачу — просто запустить код. В безкодовых решениях аналитику придется писать протокол — что и как он делал по шагам. Это облегчит проверку и даст возможность безболезненно повторить задачу в будущем. Конечно, Excel и другие визуальные инструменты очень ускоряют работу, я сам пользуюсь ими и не отговариваю вас. Моя задача обозначить плюсы и минусы этих подходов — что вам ближе, решать только вам.

Эти нюансы я понял, только когда стал работать в Retail Rocket, так как требования к качеству были значительно выше, чем на моих

предыдущих местах работы. Раньше я проверял только результат, а теперь — все решение целиком.

## КАК ТЕСТИРОВАТЬ И ВЫКЛАДЫВАТЬ ИЗМЕНЕНИЯ В РАБОЧУЮ СИСТЕМУ

Если задача вносит изменения в рабочую систему, то следующий шаг проверки — выкладка (deploy) изменений. Здесь все выглядит стандартно для разработки, и вы можете использовать практики, принятые у ваших разработчиков. В аналитике Retail Rocket мы использовали CI/CD на основе GitLab, когда все изменения выкладываются нажатием одной кнопки. Мы думали, кто это должен делать, и после различных экспериментов сошлись на том, что это должен делать исполнитель задачи. Как таковых инженеров тестирования у нас нет, поэтому исполнитель переводит задачу в статус тестирования (Testing). Далее делает выкладку, следит за тем, чтобы тесты были выполнены и изменения отразились на работе системы. Например, проверяет, что нужные отчеты работают и предоставляют информацию в требуемом виде. Цели выкладки: отразить изменения в рабочей системе, проверить, что все работает так, как этого требует задача.

## КАК ЗАЩИЩАТЬ ЗАДАЧУ ПЕРЕД ИНИЦИАТОРОМ

У задачи есть инициатор, который ее поставил, и только этот человек может дать разрешение перевести ее в статус выполненной. В статусе тестирования, после выполнения всех расчетов, исполнитель задачи обращается к инициатору с просьбой проверить результат. Это может быть инсайт, отчет или какое-то программное изменение системы. Тут инициатор должен либо согласиться с результатами задачи, либо нет. В случае отказа я рекомендую сравнить то, что требует инициатор по результатам проверки, с постановкой задачи. Разница между тем, чего хотят от вас сейчас, и тем, чего хотели

на этапе планирования задачи, может быть большой. Встречается такая ситуация довольно часто. Как с этим бороться, особенно если инициатор находится выше исполнителя в иерархии? Во-первых, правила игры должны быть известны всем и быть явно обозначены. Во-вторых, как я уже писал, нужно вести аудиозапись на встречах планирования. В-третьих, если условия задачи изменились существенно, то нужно признать, что результаты ее оказались ненужными и время было потрачено зря. А затем завести новую задачу, трудоемкость которой будет оценена отдельно.

Отдельная проблема — инициатор не выходит на связь и ничего не делает с полученными результатами. Это может свидетельствовать о том, что задача «перегорела» и больше не интересна, если, конечно, не было каких-либо форс-мажоров. Неплохо было бы узнавать такие новости до того, как на задачу были потрачены ресурсы. Что делать? Я боролся с этим пессимизацией приоритета последующих задач от таких инициаторов, но, откровенно говоря, смог позволить себе это только заняв позицию сооснователя компании.

## НУЖНО ЛИ УМЕТЬ ПРОГРАММИРОВАТЬ?

Да, нужно. В XXI веке понимать, как использовать программирование в своей работе, желательно каждому человеку. Раньше программирование было доступно только узкому кругу инженеров. Со временем прикладное программирование стало все более доступным, демократичным и удобным.

Я научился программировать самостоятельно в детстве. Отец купил компьютер «Партнер 01.01» в конце 80-х, когда мне было примерно одиннадцать лет, и я начал погружаться в программирование. Вначале освоил язык BASIC, потом уже добрался до ассемблера. Изучал все по книгам — спросить тогда было не у кого. Задел, который был сделан в детстве, мне очень пригодился в жизни. В то время моим главным инструментом был белый мигающий курсор на черном экране, программы прихо-

дилось записывать на магнитофон — все это не идет ни в какое сравнение с теми возможностями, которые есть сейчас. Азам программирования научиться не так сложно. Когда моей дочери было пять с половиной лет, я посадил ее за несложный курс по программированию на языке Scratch. С моими небольшими под-сказками она прошла этот курс и даже получила сертификат MIT начального уровня.

Прикладное программирование — это то, что позволяет автоматизировать часть функций сотрудника. Первые кандидаты на автоматизацию — повторяющиеся действия.

В аналитике есть два пути. Первый — пользоваться готовыми инструментами (Excel, Tableau, SAS, SPSS и т. д.), где все действия совершаются мышкой, а максимум программирования — написать формулу. Второй — писать на Python, R или SQL. Это два фундаментально разных подхода, но хороший специалист должен владеть обоими. При работе с любой задачей нужно искать баланс между скоростью и качеством. Особенно это актуально для поиска инсайтов. Я встречал и ярых приверженцев программирования, и упрямцев, которые могли пользоваться только мышкой и от силы одной программой. Хороший специалист для каждой задачи подберет свой инструмент. В каком-то случае он напишет программу, в другом сделает все в Excel. А в третьем — совместит оба подхода: на SQL выгрузит данные, обработает датасет в Python, а анализ сделает в сводной (pivot) таблице Excel или Google Docs. Скорость работы такого продвинутого специалиста может быть на порядок больше, чем одностаночника. Знания дают свободу.

Еще будучи студентом, я владел несколькими языками программирования и даже успел поработать полтора года разработчиком ПО. Времена тогда были сложными — я поступил в МФТИ в июне 1998 года, а в августе случился дефолт. Жить на стипендию было невозможно, денег у родителей я брать не хотел. На втором курсе мне повезло, меня взяли разработчиком в одну из компаний при МФТИ — там я углубил знание ассемблера и Си. Через какое-то

время я устроился в техническую поддержку компании StatSoft Russia — здесь я прокачал статистический анализ. В Ozon.ru прошел обучение и получил сертификат SAS, а еще очень много писал на SQL. Опыт программирования мне здорово помог — я не боялся чего-то нового, просто брал и делал. Если бы у меня не было такого опыта программирования, в моей жизни не было бы многих интересных вещей, в том числе компании Retail Rocket, которую мы основали с моими партнерами.

## ДАТАСЕТ

Датасет — это набор данных, чаще всего в виде таблицы, который был выгружен из хранилища (например, через SQL) или получен иным способом. Таблица состоит из столбцов и строк, обычно именуемых как записи. В машинном обучении сами столбцы бывают независимыми переменными (*independent variables*), или предикторами (*predictors*), или чаще фичами (*features*), и зависимыми переменными (*dependent variables, outcome*). Такое разделение вы встретите в литературе. Задачей машинного обучения является обучение модели, которая, используя независимые переменные (фичи), сможет правильно предсказать значение зависимой переменной (как правило, в датасете она одна).

Основные два вида переменных — категориальные и количественные. Категориальная (*categorical*) переменная содержит текст или цифровое кодирование «категории». В свою очередь, она может быть:

- Бинарной (*binary*) — может принимать только два значения (примеры: да/нет, 0/1).
- Номинальной (*nominal*) — может принимать больше двух значений (пример: да/нет/не знаю).
- Порядковой (*ordinal*) — когда порядок имеет значение (пример, ранг спортсмена, номер строки в поисковой выдаче).

Количественная (quantitative) переменная может быть:

- Дискретной (discrete) — значение подсчитано счетом, например, число человек в комнате.
- Непрерывной (continuous) — любое значение из интервала, например, вес коробки, цена товара.

Рассмотрим пример. Есть таблица с ценами на квартиры (зависимая переменная), одна строка (запись) на квартиру, у каждой квартиры есть набор атрибутов (независимы) со следующими столбцами:

- Цена квартиры — непрерывная, зависимая.
- Площадь квартиры — непрерывная.
- Число комнат — дискретная (1, 2, 3, ...).
- Санузел совмещен (да/нет) — бинарная.
- Номер этажа — порядковая или номинальная (зависит от задачи).
- Расстояние до центра — непрерывная.

## ОПИСАТЕЛЬНАЯ СТАТИСТИКА

Самое первое действие после выгрузки данных из хранилища — сделать разведочный анализ (exploratory data analysis), куда входит описательная статистика (descriptive statistics) и визуализация данных, возможно, очистка данных через удаление выбросов (outliers).

В описательную статистику обычно входят различные статистики по каждой из переменных во входном датасете:

- Количество непустых значений (non missing values).
- Количество уникальных значений.
- Минимум/максимум.

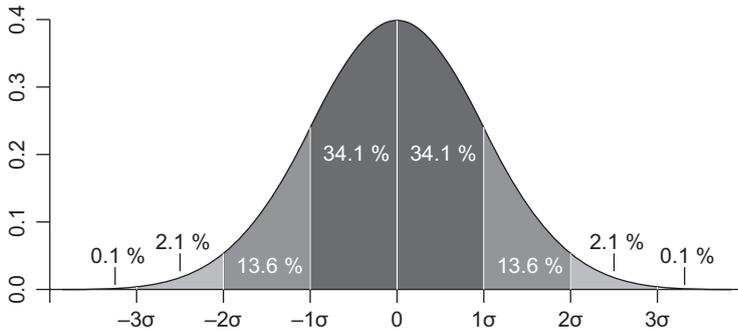
- Среднее значение.
- Медиана.
- Стандартное отклонение.
- Перцентили (percentiles) — 25 %, 50 % (медиана), 75 %, 95 %.

Не для всех типов переменных их можно посчитать — например, среднее значение можно рассчитать только для количественных переменных. В статистических пакетах и библиотеках статистического анализа уже есть готовые функции, которые считают описательные статистики. Например, в библиотеке `pandas` для Python есть функция `describe`, которая сразу выведет несколько статистик для одной или всех переменных датасета:

```
s = pd.Series([4-1, 2, 3])
s.describe()
count    3.0
mean     2.0
std      1.0
min      1.0
25%      1.5
50%      2.0
75%      2.5
max      3.0
```

Хотя эта книга не является учебником по статистике, дам вам несколько полезных советов. Часто в теории подразумевается, что мы работаем с нормально распределенными данными, гистограмма которых выглядит как колокол (рис. 4.1).

Очень рекомендую проверять это предположение хотя бы на глаз. Медиана — значение, которое делит выборку пополам. Например, если 25-й и 75-й перцентиль находятся на разном расстоянии от медианы, это уже говорит о смещенном распределении. Еще один фактор — сильное различие между средним и медианой; в нормальном распределении они практически совпадают. Вы будете часто иметь дело с экспоненциальным распределением, если анализируете поведение клиентов, — например, в `Ozon.ru` время между последовательными заказами клиента будет иметь



**Рис. 4.1.** Нормальное распределение и Шесть Сигм

экспоненциальное распределение. Среднее и медиана для него отличаются в разы. Поэтому правильная цифра — медиана, значение, которое делит выборку пополам. В примере с Ozon.ru это время, в течение которого 50 % пользователей делают следующий заказ после первого. Медиана также более устойчива к выбросам в данных. Если же вы хотите работать со средними, например, из-за ограничений статистического пакета, да и технически среднее считается быстрее, чем медиана, то в случае экспоненциального распределения можно его обработать натуральным логарифмом. Чтобы вернуться в исходную шкалу данных, нужно полученное среднее обработать обычной экспонентой.

Перцентиль — значение, которое заданная случайная величина не превышает с фиксированной вероятностью. Например, фраза «25-й перцентиль цены товаров равен 150 рублям» означает, что 25 % товаров имеют цену меньше или равную 150 рублям, остальные 75 % товаров дороже 150 рублей.

Для нормального распределения, если известно среднее и стандартное отклонение, есть полезные теоретически выведенные закономерности — 95 % всех значений попадает в интервал на расстоянии двух стандартных отклонений от среднего в обе стороны, то есть ширина интервала составляет четыре сигмы. Возможно, вы слышали такой термин, как Шесть сигм (six sigma, рис. 4.1), — эта

цифра характеризует производство без брака. Так вот, этот эмпирический закон следует из нормального распределения: в интервал шести стандартных отклонений вокруг среднего (по три в каждую сторону) укладывается 99.99966 % значений — идеальное качество. Перцентили очень полезны для поиска и удаления выбросов из данных. Например, при анализе экспериментальных данных вы можете принять то, что все данные вне 99-го перцентиля — выбросы, и удалять их.

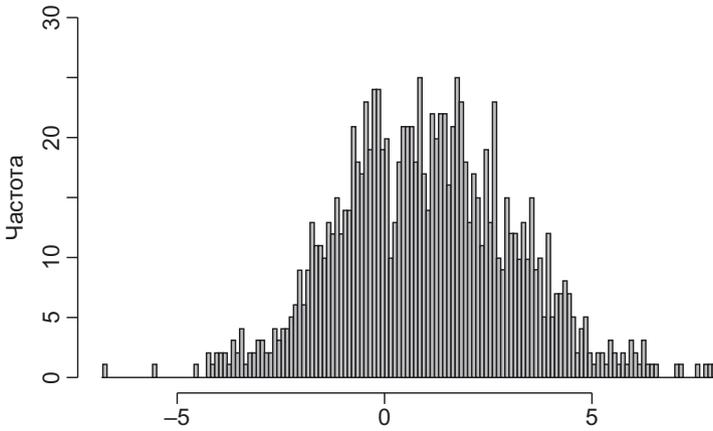
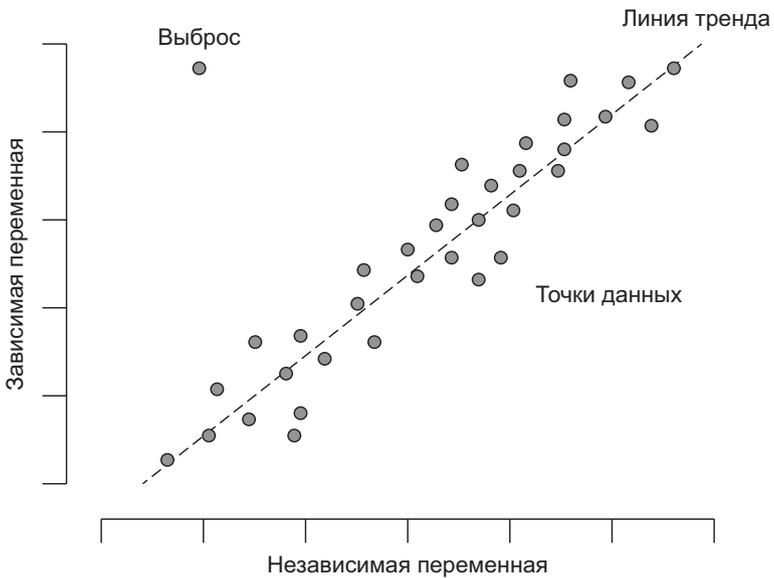
## ГРАФИКИ

Хороший график стоит тысячи слов. Основные виды графиков, которыми пользуюсь я:

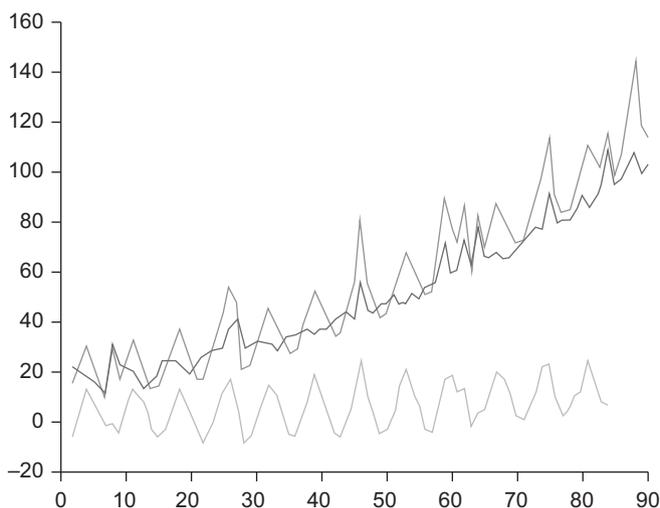
- гистограммы;
- диаграмма рассеяния (scatter chart);
- график временного ряда (time series) с линией тренда;
- график «ящики с усами» (box plot, box and whiskers plot).

Гистограмма (рис. 4.2) — наиболее полезный инструмент анализа. Она позволяет визуализировать распределение по частотам появления какого-то значения (для категориальной переменной) или разбить непрерывную переменную на диапазоны (bins). Второе используется чаще, и если к такому графику дополнительно предоставить описательные статистики, то у вас будет полная картина, описывающая интересующую вас переменную. Гистограмма — это простой и интуитивно понятный инструмент.

График диаграммы рассеяния (scatterplot, рис. 4.3) позволяет увидеть зависимость двух переменных друг от друга. Строится он просто: на горизонтальной оси — шкала независимой переменной, на вертикальной оси — шкала зависимой. Значения (записи) отмечаются в виде точек. Также может добавляться линия тренда. В продвинутых статистических пакетах можно интерактивно пометить выбросы.

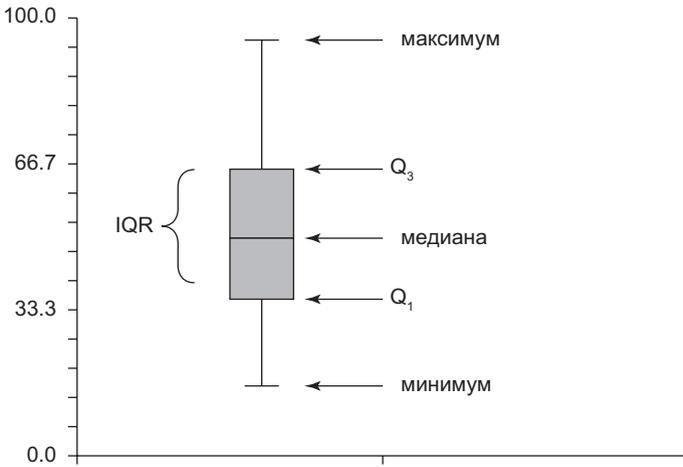
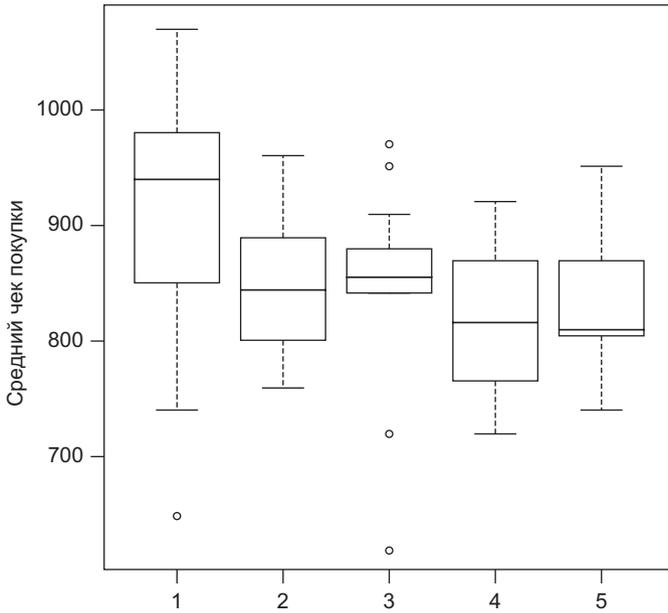
**Рис. 4.2.** Гистограмма**Рис. 4.3.** Диаграмма рассеяния

Графики временных рядов (time series, рис. 4.4) — это почти то же самое, что и диаграмма рассеяния, в которой независимая переменная (на горизонтальной оси) — это время. Обычно из временного ряда можно выделить две компоненты — циклическую и трендовую. Тренд можно построить, зная длину цикла, например, семидневный — это стандартный цикл продаж в продуктовых магазинах, на графике можно увидеть повторяющуюся картинку каждые 7 дней. Далее на график накладывается скользящее среднее с длиной окна, равной циклу, — и вы получаете линию тренда. Практически все статистические пакеты, Excel, Google Sheets умеют это делать. Если нужно получить циклическую компоненту, это делается вычитанием из временного ряда линии тренда. На основе таких простых вычислений строятся простейшие алгоритмы прогнозирования временных рядов.



**Рис. 4.4.** Временные ряды

График «Ящик с усами» (box plot, рис. 4.5) очень интересен; в некоторой степени он дублирует гистограммы, так как тоже показывает оценку распределения.

**Рис. 4.5.** Ящик с усами**Рис. 4.6.** Ящики с усами для разных экспериментов

Он состоит из нескольких элементов: усов, которые обозначают минимум и максимум, ящика, верхний край которого 75-й перцентиль, нижний — 25-й перцентиль. В ящике линия — это медиана, значение «посередине», которая делит выборку пополам. Этот тип графика удобен для сравнения результатов экспериментов или переменных между собой. Пример такого графика ниже (рис. 4.6). Считаю это лучшим способом визуализации результатов тестирования гипотез.

## ОБЩИЙ ПОДХОД К ВИЗУАЛИЗАЦИИ ДАННЫХ

Визуализация данных нужна для двух вещей: для исследования данных и для того, чтобы объяснить выводы заказчику. Часто для представления результатов используется несколько способов: простой комментарий с парой цифр, Excel или другой формат электронных таблиц, презентация со слайдами. Все эти три способа объединяют вывод и доказательство — то есть объяснение, как к этому выводу пришли. Доказательство бывает удобно выражать в графиках. В 90 % случаев для этого достаточно тех графиков, типы которых были описаны выше. Исследовательские графики и презентационные отличаются друг от друга. Цель исследовательских — найти закономерность или причину, их, как правило, много, и бывает, что они строятся наугад. Целью презентационных графиков является подведение ЛПР (лица, принимающего решения) к выводам в задаче. Тут важно все — и заголовок слайда, и их простая последовательность, которая ведет к нужному выводу. Важный критерий схемы доказательства вывода — как быстро заказчик поймет и согласится с вами. Необязательно это должна быть презентация. Лично я предпочитаю простой текст — пара

предложений с выводами, пара графиков и несколько цифр, доказывающих эти выводы, ничего лишнего.

Джин Желязны, который работает директором по визуальным коммуникациям в McKinsey & Company, в своей книге «Говори на языке диаграмм» утверждает [28]:

«Тип диаграммы определяют вовсе не данные (доллары или проценты) и не те или иные параметры (прибыль, рентабельность или зарплата), а ваша идея — то, что вы хотите в диаграмму вложить».

Рекомендую вам обращать внимание на графики в презентациях и статьях — доказывают ли они выводы автора? Все ли вам нравится в них? Могли бы они быть более убедительными?

А вот что пишет Джин Желязны про слайды в презентациях [28]:

«Широкое распространение компьютерных технологий привело к тому, что сейчас за минуты можно сделать то, на что раньше требовались часы кропотливой работы, — и слайды пекутся как пирожки... пресные и невкусные».

Я делал довольно много докладов: со слайдами и без, короткие, на 5–10 минут, и длинные — на час. Смею вас заверить, что мне намного сложнее сделать убедительный текст для короткого доклада без слайдов, чем презентацию в PowerPoint. Посмотрите на политиков, которые выступают: их задача убеждать, много ли из них показывают слайды на выступлениях? Слово убеждает сильнее, слайды — это всего лишь наглядный материал. И чтобы ваше слово было понятно и убедительно, требуется больше труда, чем для накидывания слайдов. Я себя поймал на том, что при составлении слайдов я думаю о том, как презентация выглядит. А при составлении устного доклада — насколько убедительны мои аргументы, как работать с интонацией, насколько понятна моя мысль. Пожалуйста, подумайте, действительно ли вам нужна презентация? Хотите ли вы превратить совещание в просмотр скучных слайдов вместо принятия решений?

«Совещания должны фокусироваться на кратких письменных отчетах на бумаге, а не на тезисах или обрывочных пунктах списка, проецируемых на стену», — утверждает Эдвард Тафти, видный представитель школы визуализации данных, в своей работе «Когнитивный стиль PowerPoint» [29].

## ПАРНЫЙ АНАЛИЗ ДАННЫХ

О парном программировании я узнал от разработчиков [30] Retail Rocket. Это техника программирования, при которой исходный код создается парами людей, программирующих одну задачу и сидящих за одним рабочим местом. Один программист сидит за клавиатурой, другой — работает головой, сосредоточен на картине в целом и непрерывно просматривает код, производимый первым программистом. Время от времени они могут меняться местами.

И нам удалось ее адаптировать для нужд аналитики! Аналитика, как и программирование, — творческий процесс. Представьте, что вам нужно построить стену. У вас есть один рабочий. Если вы добавите еще одного — скорость вырастет примерно в два раза. В творческом процессе так не получится. Скорость создания проекта не вырастет в два раза. Да, можно проект декомпозировать, но я сейчас обсуждаю задачу, которая не декомпозируется, и ее должен делать один человек. Парный же подход позволяет многократно ускорить этот процесс. Один человек за клавиатурой, второй сидит рядом. Две головы работают над одной проблемой. Когда я решаю сложные проблемы, я разговариваю сам с собой. Когда разговаривают две головы друг с другом — они ищут причину лучше. Мы используем схему парной работы для следующих задач.

- Когда нужно передать знания одного проекта от одного сотрудника другому, например, был нанят новичок. «Головой» будет сотрудник, который передает знания, «руками» за клавиатурой — кому передают.
- Когда проблема сложная и непонятная. Тогда два опытных сотрудника в паре решат ее намного эффективней одного. Будет сложнее сделать задачу анализа однобоко.

Обычно на планировании мы переносим задачу в категорию парных, если понятно, что она подходит под критерии таковой.

Плюсы парного подхода — время используется намного эффективней, оба человека очень сфокусированы, они друг друга дисциплинируют. Сложные задачи решаются более творчески и на порядок быстрее. Минус — в таком режиме невозможно работать больше нескольких часов, очень сильно устаешь.

## ТЕХНИЧЕСКИЙ ДОЛГ

Еще одна важная вещь, которой я научился у инженеров Retail Rocket [31], — работа с техническим долгом (technical debt). Технический долг — это работа со старыми проектами, оптимизация скорости работы, переход на новые версии библиотек, удаление старого программного кода от тестирования гипотез, инженерное упрощение проектов. Все эти задачи занимают добрую треть времени разработки аналитики. Приведу цитату технического директора Retail Rocket Андрея Чижа [31]:

«Я еще не встречал компаний за свою практику (а это более 10 компаний, в которых работал сам, и примерно столько же, с которыми хорошо знаком изнутри), кроме нашей, у которых в бэклоге были бы задачи на удаление функционала, хотя, наверное, такие существуют».

Я тоже не встречал. Видел «болота» программных проектов, где старье мешает создавать новое. Суть технического долга — все, что

вы сделали ранее, нужно обслуживать. Это как с ТО автомобиля — его нужно делать регулярно, иначе машина сломается в самый неожиданный момент. Программный код, в который давно не вносились изменения или обновления, — плохой код. Обычно он уже работает по принципу «работает — не трогай». Четыре года назад я общался с разработчиком Bing. Он рассказал, что в архитектуре этого поискового движка есть скомпилированная библиотека, код которой потерян. И никто не знает, как это восстановить. Чем дольше это тянется, тем хуже будут последствия.

Как аналитики Retail Rocket обслуживают технический долг:

- После каждого проекта тестирования гипотез мы удаляем программный код этой гипотезы везде, где только можно. Это избавляет нас от ненужного и неработающего хлама.
- Если происходит обновление каких-либо версий библиотек — мы делаем это с некоторым запозданием, но делаем регулярно. Например, платформу Spark мы апгрейдем регулярно, начиная с версии 1.0.0.
- Если какие-либо компоненты обработки данных работают медленно — ставим задачу и занимаемся ею.
- Если есть какие-то потенциально опасные риски — например, переполнение дисков кластера, тоже ставится соответствующая задача.

Работа с техническим долгом — это путь к качеству. Меня убедила в этом работа в проекте Retail Rocket. С инженерной точки зрения проект сделан как в «лучших домах Калифорнии».

# 5

## ДАННЫЕ



Данные — представление фактов, понятий или инструкций в форме, приемлемой для общения, интерпретации или обработки человеком или с помощью автоматических средств.

### *Википедия*

Прежде чем мы перейдем к собственно анализу данных, считаю необходимым рассмотреть предмет изучения. Цитата выше — это определение данных, которое дает Википедия. Оно очень сухое, но емкое. В моей книге я намеренно сузил это определение: под данными будут пониматься цифровые данные, которые могут быть прочитаны и обработаны ПО.

Данные бывают разными — это могут быть результаты медицинских анализов, фотографии, географические карты, описание и характеристики товаров, история посещения страниц сайта пользователями, списки клиентов и многое другое. В нашей области анализа у данных одна цель — помощь в принятии решений человеком и даже создание систем такой помощи.

- Медицинские анализы — помощь в постановке диагноза, принятие решений о выводе лекарства на рынок.
- Фотографии — поиск предметов, распознавание лиц.
- Товары — закупки нужных товаров на склад.
- История посещений сайта — рекомендательная система интересных страниц.
- Список клиентов — разбить их на группы, чтобы предложить разные скидки.
- Географические карты — навигация с учетом автомобильных пробок.

## **КАК СОБИРАЮТСЯ ДАННЫЕ**

Проведите несложный эксперимент: откройте браузер, откройте какой-нибудь новостной сайт, откройте инструменты разработчи-

ка, вкладку «Сетевые запросы» и обновите страницу. Вы увидите все сетевые взаимодействия вашего браузера. Количество таких сетевых запросов на одной странице может легко перевалить за 1000. Большая их часть — это скачивание картинок и скриптов, обеспечивающих визуализацию страницы у вас на экране. Но есть также запросы от трекеров и рекламных сетей, у них задача собрать ваш «профиль клиента» на одном или нескольких сайтах. Также все ваши запросы провайдер запишет на свои сервера, куда имеют доступ спецслужбы.

Второй пример — передвижение автомобиля по дорогам. Не секрет, что сервисы навигации активно используют данные нашего передвижения для построения своих карт пробок. Данные для этого они получают из своего приложения, а также с датчиков, установленных на дорогах.

Третий пример — мобильная геолокация. Наши передвижения, точнее, перемещения наших телефонов, аккуратно записываются сотовыми операторами в хранилища. На основе этих данных создаются разные сервисы. Один из них — определение лучшего места для открытия новой торговой точки.

## BIG DATA

Очень хайповый термин, который сейчас звучит из каждого утюга. Мне посчастливилось поработать в этой теме последние 8 лет и накопить достаточно большую экспертизу. Попробую дать собственное определение: большие данные (Big Data) — это такой объем данных, который невозможно обработать в требуемое время на одной машине (сервере).

Обычно когда говорят про большие данные, имеют в виду только их объем. Но на самом деле в коммерческом или научном применении очень важно время. Исследователь не может бесконечно ждать. Чем быстрее можно получить результат, тем лучше. Особенно когда мы работаем в условиях неопределенности результата и за-

прос к данным нужно итеративно уточнять, как правило, начиная с самых простых шагов. Современная техника, особенно скорость реакции приложений мобильного телефона, приучила нас к очень быстрой реакции на наши действия, и подсознательно мы этого ожидаем и от систем обработки данных.

Большой объем данных на самом деле получить очень несложно. Дам простой пример. Если каждую миллисекунду сохранять вашу геопозицию, например GPS координаты, то за сутки мы получим:  $1000 \text{ миллисекунд в секунде} \times 60 \text{ секунд} \times 60 \text{ минут} \times 24 \text{ часа} = 86\,400\,000$  событий. Цифра очень впечатляет, особенно если масштабировать ее на всех людей на Земле. Более подробно о больших данных я расскажу в главе про хранилища данных.

## СВЯЗНОСТЬ ДАННЫХ

Одна из важнейших характеристик данных — возможность связать разные источники данных. Например, если удастся связать затраты на интернет-рекламу и продажи через нее, то вы получаете инструмент эффективности. Далее добавляем данные по реально доставленным заказам, так как в некоторых e-commerce-бизнесах процент отказов очень высок. И на выходе мы получаем эффективность с поправкой на отказы. Фантазируем дальше, добавляем к данным категории товаров — получаем возможность видеть эффективность рекламы в разрезе категорий товаров. Продолжать можно до бесконечности. Кстати, именно этот пример иллюстрирует то, что называется «сквозной аналитикой».

Вышеприведенный пример показывает, что, добавляя новый источник данных, мы можем улучшить точность и увеличить число «степеней свободы», что можно делать с самими данными. Лично для меня это увеличивает ценность данных на порядок. Вот только есть одна заминка с тем, как эти данные связать. Для этого нужен «ключ», который должен быть в обоих источниках, и этот ключ не всегда бывает настолько точным, насколько требуется нам. Поясню на примере. Чтобы идеально связать затраты на интернет-рекламу

и покупки, нужен ключ — id пользователя. Но проблема в том, что, скорее всего, от рекламных систем вы не получите информации о том, сколько вы потратили денег на конкретного пользователя. Из-за этого приходится использовать набор ключей из ссылок, которые однозначно характеризуют рекламное объявление. Точность из-за этого страдает, но это реальная жизнь данных — лучше получить что-то, чем ничего.

## МНОГО ДАННЫХ НЕ БЫВАЕТ

Эту фразу я повторял, когда работал в Ostrovok.ru. Я точно не помню, с чем она была связана, возможно, требовалось расширить парк серверов. Считаю, что в эпоху облачных вычислений, дешевого хранения данных и хороших алгоритмов их сжатия нужно сохранять максимально много и подробно. Поверьте, когда понадобится найти ответ на какой-то вопрос и вы будете понимать, что данных нет, а могли бы быть, будет очень обидно. Рано или поздно собирать их все равно придется, почему бы не начать прямо сейчас?

**ВАЖНО!** Здесь хочу поднять одну проблему: в какой бы компании я ни работал — везде разработчики игнорируют аналитиков. При разработке какого-либо функционала или продукта анализ его функциональности через данные ставится на последнее место. В лучшем случае будет сделан сбор простейших метрик по усмотрению разработчика. Дальнейший сценарий такой — менеджеры проекта или продукта, владельцы бизнеса начинают активно интересоваться его судьбой. Что там с цифрами? Бегут к аналитикам, просят нарыть хоть что-нибудь. А что может сделать аналитик, если данных для статистических тестов не хватает и точность страдает? Только высосать информацию из пальца. С таким положением вещей я лично сталкивался десятки раз, а случаи, когда все было сделано как надо, могу по пальцам пересчитать.

Я предлагаю активно бороться с этим. Разработку можно понять — им нужно как можно быстрее выкатить новую «фичу» с очень хорошим качеством. Анализ метрик их не волнует, это лишние

строчки кода, это работа аналитиков. Что делать? Это сфера ответственности менеджера проекта/продукта, лица, от имени которого ставится задача разработке. Необходимо в процессе постановки подобных задач предусмотреть «визу» от аналитиков. Что в нее входит:

1. Отправка технического задания и примерного списка вопросов к эффективности новой разработки аналитикам.
2. Аналитики со своей стороны отдадут вам список метрик, а также встречное техническое задание для логирования (сбора метрик) данных проекта: что собирать и в каком формате.

Этот процесс не так прост, как кажется. Часто приходится в итеративном формате договариваться обо всех нюансах и ограничениях, в том числе с разработчиками. Происходит своеобразный торг, но он стоит того. Заранее хорошо продуманный результат не будет идеальным на 100 %, но если менеджмент получит ответы на 80 % своих вопросов в течение нескольких дней с момента запуска «фичи» — это успех. Ничто не играет против нас так, как время! И лучше его потратить до запуска, а не после, теряя деньги на неэффективном продукте.

## ДОСТУП К ДАННЫМ

Теперь коснемся доступа к данным внутри компании. Кто может получить его?

Отвлечемся на компанию Netflix, один из крупнейших поставщиков сериалов (мой любимый — «Карточный домик»). У компании очень интересная корпоративная культура [32]. Один из ее принци-

пов звучит так: «Share information openly, broadly, and deliberately» (обмениваемся информацией открыто, широко и сознательно).

У этого правила, правда, есть строгое исключение: они нетерпимо относятся к торговле инсайдерской информацией, а также платежной информацией клиентов, доступ к которой ограничен. Как этот принцип можно применить на практике? Не ограничивать своим сотрудникам доступ к информации, но ограничить доступ к персональным данным клиентов. Я иду обычно еще дальше, стараюсь максимально убрать барьер между сотрудниками-неаналитиками и данными. Просто я считаю, что должна быть не только свобода доступа к данным, но и минимум посредников между «спрашивающим» и данными. Это важно, потому что против нас играет время. Часто сами запросы данных выглядят довольно простыми, их можно сделать самостоятельно. «Дайте мне выгрузку таких-то данных» — не аналитическая задача: менеджер знает, что ему конкретно нужно, пусть сам получит это через несложный интерфейс. Для этого нужно обучить команду самостоятельно работать с данными. Посредник только создаст задержку, но если кто-то не хочет или не может действовать самостоятельно, пусть использует посредников. Этим вы убьете сразу двух зайцев — ваши аналитики не будут демотивированы примитивным скучным трудом по выгрузке данных, а ваши менеджеры смогут получать данные почти мгновенно, и значит, не будут терять драйв.

Конечно, все персональные данные клиентов должны быть обезличены. Это можно сделать, шифруя их личную информацию. Полностью лучше ее не удалять, тогда можно будет решать часть вопросов клиентской поддержки с помощью вашей системы анализа данных.

Я всегда стараюсь использовать этот подход во всех компаниях, где бы ни работал. Вы даже не представляете, насколько будут вам благодарны пользователи ваших аналитических систем, когда смогут получать данные самостоятельно. Самые умные и деятельные сотрудники являются самыми активными потребителями информации для принятия решений, и создавать им препятствия — это преступление.

## КАЧЕСТВО ДАННЫХ

Данные бывают грязными, очень грязными. Если вам встретятся «чистые» данные, то это, скорее всего, неправда. Но бывает, что в жизни сказочно везет. Аналитики данных тратят львиную долю своего времени на очистку данных от выбросов и прочих артефактов, которые могут помешать получить правильное решение. Мы все работаем в условиях неопределенности, и увеличивать ошибку из-за грязи в данных совсем не хочется.

Для меня качественные данные — это данные, которые могут быть использованы для решения конкретной задачи без каких-либо предварительных очисток. Я намеренно написал «конкретной задачи», потому что считаю: разные задачи требуют разной степени точности, так как последствия и уровень риска для компании разные. И мы движемся по лезвию бритвы, стараясь решить задачу как можно быстрее наименьшими усилиями, балансируем между трудоемкостью и ценой ошибки. Если это бухгалтерская задача, то она требует очень высокой степени точности, так как санкции налоговой службы могут быть весьма болезненными. Если управленческая и последствия не столь значимы, то некоторой степени точности можно пренебречь. Решение здесь за руководителем аналитики.

Основные причины плохого качества данных:

- человеческий фактор;
- техническая потеря данных;
- ошибка интеграции и выгрузки данных в хранилище;
- отставание в обновлении данных в хранилище.

Рассмотрим более подробно.

Часть данных приходит от людей напрямую: по разным каналам связи они отдают нам цифры. Для простоты будем считать, что периодически они заполняют какую-то форму и отправляют нам. Из школьного курса физики мы знаем про погрешность отсчета по шкале — при любых измерениях принимается погрешность

отсчета, равная половине цены деления. Для линейки с миллиметровой шкалой это полмиллиметра. То есть просто из-за того, что мы можем посмотреть не под тем углом, чуть сдвинуть линейку, — мы уже ошибаемся. Чего же ждать от людей, которые используют инструменты посложнее линейки?

Кроме того, не стоит забывать про намеренную фальсификацию данных. Давайте будем называть вещи своими именами: изменения, которые внесены в данные человеком, пусть даже из лучших побуждений, — это намеренная фальсификация. За примерами ходить далеко не нужно — выборы! Спасибо независимым исследователям, которые анализируют данные участков, ищут аномалии, выбросы и прочие «неслучайные» закономерности. В промышленности тоже есть методики поиска аномалий, например, с помощью статистических карт контроля качества.

Проблема технической потери данных очень актуальна в веб-аналитике, которая анализирует посещаемость сайтов. Не все данные с компьютера или смартфона долетают до аналитического сервера. Между сервером и клиентом может быть десяток маршрутизаторов, часть сетевых пакетов может потеряться, пользователь может закрыть браузер во время отправки. Как правило, этот процент потерь составляет около 5 %, и уменьшить без сложных ухищрений его практически невозможно. Один из способов — расположить блок с кодом вызова аналитической системы в самом верху веб-страницы, тогда данные отправятся раньше полной загрузки страницы, а значит, и потери немного уменьшатся.

Ошибки интеграции данных очень неприятны, они появляются в самый неожиданный момент, когда их не ждешь, их сложно диагностировать. Под ошибкой интеграции я понимаю потерю данных из-за неправильной работы процесса сбора данных. Она может быть обратимой и необратимой. Обратимая касается в основном ситуаций, когда мы забираем данные из какого-то источника, и чтобы исправить ошибку, достаточно данные перечитать из него. Необратимая ошибка связана с тем, что по факту данные «исчезают» после отправки к нам, то есть они идут потоком и нигде больше

не сохраняются. Я уже писал, как обнаружил, когда одна из самых продвинутых систем в мире не передавала данные по браузеру Орега. После исправления данные стали передаваться, но старые данные уже не восстановить. Бывает, что разработчики сложного решения не предусмотрели, забыли или допустили ошибку в реализации сбора статистической информации об использовании продукта. В таком случае вы получите статистику только после исправления, а со старыми данными можно попрощаться навсегда.

Отставание в обновлении данных в хранилище тоже может являться проблемой. По крайней мере, о ней нужно помнить, когда с этими данными идет работа. Есть разные схемы обновления хранилищ данных, о них мы поговорим в главе, посвященной хранилищам данных. Самое главное, что нужно помнить: заказчик анализа или даже вы сами можете ожидать, что данные в системе в точности соответствуют реальной картине мира, но это не так. Всегда есть отставание, разница между моментом, когда событие возникло, и моментом, когда информация об этом попала к вам в хранилище. Это могут быть секунды, а могут быть и дни. Я не считаю, что это ошибка, но всегда нужно понимать, какой источник данных в какой конкретный момент времени обновляется, и следить за выполнением этого расписания, чтобы не было сюрпризов.

## КАК ПРОВЕРЯЕТСЯ И КОНТРОЛИРУЕТСЯ КАЧЕСТВО ДАННЫХ

Для меня это сродни искусству, но существует несколько полезных практик, из тех, которые по принципу Парето дают 80 % результата при затраченных 20 % усилий:

- мониторинг выгрузки данных в хранилище;
- здоровый скептицизм к полученным результатам анализа;
- статистический анализ выбросов;
- особое внимание к недублированным источникам данным.

Первый шаг для любой аналитической системы — ее мониторинг, а именно мониторинг обновления данных в хранилище. Это несложно, зато очень эффективно с точки зрения максимально быстрого поиска проблем. Такой мониторинг можно разделить на две части: одна отслеживает, что данные поступают в систему вовремя, вторая убеждается, что они корректны. Первая часть очень проста, как правило она реализована на специально программе или системе-планировщике (scheduler). Главное — правильно подписаться на уведомления об ошибках и вовремя на них реагировать. Лучше это делать, как в армии: лампочка загорелась — бежим исправлять; стоит составить инструкции для персонала, как действовать в случае аварии.

Вторая часть значительно сложнее. Хорошие тесты на корректность данных дорогого стоят. Если есть доступ к источнику и способ запускать на нем несложные запросы, то можно сравнивать простые статистики данных в хранилище и источнике. Также можно проверять целостность данных, сравнивая их уже в хранилище, — действительно ли во всех справочниках есть информация и т. д.

Скептицизм к выводам анализа данных — очень полезная вещь. Он заключается в том, чтобы подвергнуть результат сомнению и проверке. Например, попробовать получить тот же самый вывод альтернативным способом или через другой источник данных — если результаты совпадут, получите плюс один к уверенности, что все правильно. Другой способ — тестировать данные на каждом шаге, проверять их распределение и соответствие здравому смыслу. Мне лично это помогло бы не допустить очень многих ошибок, которые я делал, когда хотелось побыстрее получить нужный результат.

Выбросы — это данные, которые не укладываются в нашу картину мира, а точнее в распределение, которое мы обычно наблюдаем: кто-то совершил очень крупную покупку в магазине, поступили странные данные от одного из избирательных участков, зачислена аномально большая сумма на счет. Все это выбросы, но удалять их из анализа просто так нельзя. Часто удаление выброса может привести к изменению выводов и решений на полностью противо-

положные. Считаю, что работа с выбросами данных является искусством. Более подробно это рассмотрим в главе 10.

Теперь поговорим про данные, которые невозможно восстановить повторным чтением из источника. Выше я писал, почему это может произойти, — в системе существует ошибка интеграции или разработчики не сделали сбор и отправку необходимых данных. Таким источникам данных необходимо уделять самое пристальное внимание, например, написать специальные тесты, чтобы как можно раньше заметить проблему. А вот с невнимательностью разработчиков лучше работать на уровне управления проектом внедрения или внося изменения в их культуру разработки. Об этом я писал в разделе «Много данных не бывает».

## ТИПЫ ДАННЫХ

Вот основные типы данных, с которыми приходится работать:

1. Состояние на определенный момент времени.
2. Лог изменений данных.
3. Справочники.

Разберем каждый тип отдельно на примере с банковским счетом. Итак, у вас есть счет, туда приходит зарплата, скажем, первого числа каждого месяца. Вы пользуетесь картой, привязанной к этому счету, для оплаты покупок. Так вот, остаток средств на вашем счете прямо сейчас — это состояние счета на определенный момент времени. Движение средств по счету — это так называемый лог изменений состояния счета (лог изменения данных). А справочником могут выступать категории покупок, которые банк предоставляет в онлайн-приложении для каждой покупки, например: продукты, авиабилеты, кинотеатр, ресторан. А теперь подробнее о каждом типе данных.

Состояние на определенный момент времени. Все мы имеем дело с разными объектами, как физическими, так и виртуальными. Эти объекты имеют свойства или атрибуты, которые могут изменяться во

времени. Например, координаты вашего местонахождения на карте, остаток средств на счете, цвет волос, который может измениться после посещения парикмахерской, рост и вес, которые меняются со временем, статус заказа в онлайн-магазине, ваша должность на работе. Это все объекты с каким-то свойством. Чтобы отследить изменения этих свойств, нужно их периодически запоминать, например, сделав «слепок» (snapshot) всех счетов клиентов в банке (табл. 5.1). Имея на руках два таких слепка, можно легко посчитать изменения. Но есть альтернативный способ отслеживать изменения.

**Таблица 5.1.** Пример «слепок» счетов клиентов

ID счета	Сумма средств
234	2000
245	5000
857	2000

Если запоминать, в какой момент времени какое свойство/атрибут объекта менялось, включая информацию о его новом значении, то мы получим так называемый лог изменений данных. Когда речь идет о таком типе данных, я обычно представляю себе таблицу (табл. 5.2), неизменяемыми атрибутами которой являются следующие столбцы:

- Дата и время изменения свойства — точность может быть очень высокой, вплоть до наносекунд.
- Указание на объект, который изменился. Например, номер банковского счета.
- Новое значение свойства или его изменение. Например, в этом поле можно сохранить новое значение суммы на счете, но обычно там находится сумма списания или зачисления на счет со знаком «плюс» или «минус».

Иногда может быть еще несколько необязательных полей: название атрибута, если их несколько, например вес или рост; старое значение атрибута, это может потребоваться для проверки целостности данных (табл. 5.2).

Таблица 5.2. Изменения счетов клиентов

Дата, время	ID счета	Изменение
20.06.2018 13:24:05.001	123	+1000
23.07.2018 12:20:23.034	245	-2000
10.08.2018 10:34:20.300	678	+4000

Что касается справочников, то они, как правило, содержат информацию, которая не изменяется часто и позволяет «расшифровать» или обобщить данные. Например, в таблице лога изменений или состояния может храниться не имя клиента, а его номер или идентификатор (ID), тогда в справочнике будет храниться соответствие этого идентификатора и имени клиента. Обобщение может быть нужно, чтобы агрегировать данные, например, по типу клиента — юридическое или физическое лицо. Для магазина это может быть категория товаров или даже дерево категорий, оформленное в специальной структуре. Сам справочник тоже может изменяться, и его тоже возможно представить и в виде таблицы состояния, и в виде лога изменений, если такое потребуется.

Большая часть данных в мире представлена вышеупомянутыми типами, их достаточно для выполнения подавляющего большинства задач анализа данных.

## ФОРМАТЫ ХРАНЕНИЯ ДАННЫХ

Аналитики в основном работают с двумя типами данных: файлы и базы данных.

Самый распространенный формат для анализа данных — файлы. Практически во всех открытых источниках публикуются именно они. Также системы хранения больших данных, как Hadoop, тоже используют файловый формат хранения. Вообще файл — это вариант передачи данных. Когда вы сохраняете что-то в файл из программы, вы «сериализуете» данные в нули и единицы (биты), которые бу-

дут храниться на диске. Когда вы читаете файл в программе, чтобы транслировать его в память, происходит «десериализация», то есть превращение последовательных нулей и единиц (битов) в структуры данных, понятных для обработки в программе. Запомните эти термины, они вам пригодятся при чтении литературы и статей про данные.

Формат файлов бывает текстовым и бинарным. Текстовый для обычных данных, главный признак — возможность его посмотреть через обычный текстовый редактор. Этот формат используется тогда, когда все данные можно представить в виде текста и букв. Бинарный формат используется уже для данных, которые нельзя представить в виде цифр и букв, например картинки и звук. Хотя есть способ представить бинарные данные в текстовом виде, например, закодировать специальными кодеками, такими как Base64. Тогда бинарные данные можно хранить и в текстовых файлах. Есть один недостаток — размер данных увеличивается примерно на 36%, что может быть существенным для больших данных.

Среди текстовых форматов файлов наиболее распространены три:

- CSV (comma-separated values) или TSV (tab-separated values).
- JSON (Java Script Object Notation).
- XML (eXtensible Markup Language).

Файлы CSV самые удобные и простые, выглядят они как обычная таблица, где разделитель между столбцами запятая (хотя это может быть табуляция, точка с запятой и т. д.), разделитель между строками (записями) — перенос строки. Парсить (прочитать и разметить поля в собственной программе) такие файлы одно удовольствие. Их очень легко просмотреть в любом редакторе или консоли. Есть у них только пара минусов. Если используется символ разделителя (запятая или иной), то в полях этот символ нужно экранировать, например, обернув все поле кавычками. В то же время в самом поле тоже могут быть кавычки. Придется это поддерживать в парсере (программа или код, которая будет читать это файл). Это усложняет работу с такими файлами и иногда приводит к появлению битых записей, которые не удалось распарсить. Второй недостаток — то,

что таблица плоская, поэтому невозможно использовать данные со сложной структурой или же придется упаковывать их в сложный формат, например JSON, и размещать в полях как обычный текст.

```
Year,Make,Model,Description,Price
1997,Ford,E350,"ac, abs, moon",3000.00
1999,Chevy,"Venture ""Extended Edition""",",",4900.00
1999,Chevy,"Venture ""Extended Edition, Very Large""",,5000.00
1996,Jeep,Grand Cherokee,"MUST SELL! air, moon roof,
loaded",4799.00
```

JSON — формат гораздо более сложный, он является стандартом де-факто для обмена данными в интернете между сервисами. Главная его особенность в том, что там каждая ячейка имеет наименование. Это и плюс, и минус. Плюс — можно размещать сложные структуры, иерархии, очень удобно парсить, легко открыть в текстовом редакторе или браузере. Главный минус JSON — много лишней информации, в то время как в табличных данных именованы столбцы и нет смысла писать названия для каждого значения в таблице, что сильно увеличивает объем файла.

```
{
  "firstName": "Иван",
  "lastName": "Иванов",
  "address": {
    "streetAddress": "Московское ш., 101, кв.101",
    "city": "Ленинград",
    "postalCode": 101101
  },
  "phoneNumbers": [5-
    "812 123-1234",
    "916 123-4567"
  ]
}
```

XML-формат менее распространен, чем JSON, в нем любят хранить конфигурации параметров каких-либо систем. Этот формат — конкурент JSON. Для данных я бы все-таки предпочел JSON, он легче и проще. В XML-формате, например, интернет магазины передают информацию о товарах: структуру их каталога, цены, названия, атрибуты товаров.

```
<person>
  <firstName>Иван</firstName>
  <lastName>Иванов</lastName>
  <address>
    <streetAddress>Московское ш., 101, кв.101</streetAddress>
    <city>Ленинград</city>
    <postalCode>101101</postalCode>
  </address>
  <phoneNumbers>
    <phoneNumber>812 123-1234</phoneNumber>
    <phoneNumber>916 123-4567</phoneNumber>
  </phoneNumbers>
</person>
```

Есть гораздо более экзотические форматы, с которыми вы можете столкнуться на практике:

- `pk1` — бинарные объекты Python, прочитав которые с помощью этого языка программирования вы получите в памяти сразу нужную структуру данных, не заморачиваясь с парсингом.
- `hdf` — иерархический формат структуры данных. В этот формат можно поместить разнородные данные, например товарный каталог магазина, продажи и т. д. В файле содержится метаинформация: названия, типы данных и т. д. Лично я с такими файлами никогда не работал, но они могут быть удобны, когда нужно передать данные сложного проекта другой команде или опубликовать в интернете.
- `parquet`, `avro` — это уже форматы, заточенные для больших данных. Как правило, они содержат схему данных (метаинформацию) о типе и названии полей и оптимизированы для использования в таких системах, как Hadoop. Оба формата — примеры бинарного хранения данных, хотя `avro` может опираться на JSON.

Что еще полезно знать о файлах хранения? Как они хранят метаинформацию. Если кто-то захочет передать файл с данными, то, скорее всего, в CSV-файле в первой строке будут названия полей, но информацию о типе (это число, текст, дата и т. д.) вы не получите, нужно будет дополнительно передать описание полей с файлом, иначе вам придется самим строить предположения. Если же вам

передадут JSON или XML, то там уже лучше с типами данных, в этом плане они удобнее.

Базы данных обсудим в главе про хранилища.

## СПОСОБЫ ПОЛУЧЕНИЯ ДАННЫХ

Есть три основных способа получить данные:

- прочитать файлы (обсуждали выше);
- сделать запрос к API;
- сделать запрос к базе данных.

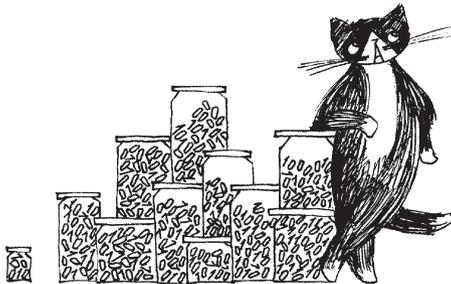
Прочитать файл — это самый простой способ: если это CSV-файл, его можно открыть в Microsoft Excel, Google Spreadsheet, OpenOffice и т. д. Все пакеты анализа данных, библиотеки любых языков программирования поддерживают данный формат. Он очень прост и удобен. С JSON и XML придется повозиться и, скорее всего, даже написать небольшой код (маленькая программа) по извлечению нужных вам данных.

Второй способ — сделать запрос к сетевому API (Application Programming Interface). Вы пишете запрос в требуемом API формате, на выходе вам приходит, как правило, JSON, который вы можете обработать, сохранить в файл и т. д. Это требует кодирования, зато работать с такими интерфейсами бывает очень интересно.

Третий способ — базы данных через использование языка программирования SQL. Для разных систем баз данных существуют свои диалекты этого языка. Обычно это связано с оптимизациями и расширением стандартного языка. Чтобы получить данные из БД, необходимо к ней подключиться через драйвер API по сети, написать запрос SQL, и если все хорошо — получить данные на выходе. В какой бы компании я ни работал — везде писал на SQL. Настоятельно рекомендую ознакомиться с этим языком программирования или хотя бы с его азами.

# 6

## ХРАНИЛИЩА ДАННЫХ



## ЗАЧЕМ НУЖНЫ ХРАНИЛИЩА ДАННЫХ

Хранилище данных содержит копию всех данных, необходимых для функционирования аналитической системы. Несколько лет назад появился модный термин Data Lakes (озеро данных) — это метод хранения данных системой или репозиторием в натуральном виде, то есть в формате, который предполагает одновременное хранение данных в различных схемах и форматах. Данные хранятся в том виде, в котором созданы: видеофайлы, изображения, документы, дампы таблиц из баз данных, CSV-файлы. Мое определение хранилища, которое я дал выше, очень сильно пересекается с озером данных. Также на кластере мы держали скачанные картинки, сырые и обработанные данные. Читателям я предлагаю меньше фокусироваться на терминах и не заморачиваться с ними, никто не даст вам четких инструкций, как хранить ваши данные. Это будет ваше решение, оно будет зависеть от ваших задач, которые предстоит решать именно вам.

Сейчас у хранилища данных гораздо больше функций, чем просто хранение данных для отчетов, — например, оно может выступать источником данных для обучения ML-моделей. Данные можно хранить не только в базе данных, но и в виде файлов, как делает Hadoop.

С моей точки зрения, хранилища данных:

- 1) являются цифровым архивом компании;
- 2) являются копией данных в источнике;
- 3) не изменяемы;
- 4) хранятся в виде, максимально приближенном к данным в источнике;
- 5) позволяют объединять данные из разных источников.

Относитесь к хранилищу как к архиву компании [34], ведь там хранятся данные с момента ее создания. Часть данных вы уже нигде не найдете, так как источники периодически чистятся. В Retail

Rocket, например, мы периодически архивируем все данные: товарные базы интернет-магазинов (они изменяются со временем), их структуры каталога, сами рекомендации. Ни в каких источниках их уже нет, но они есть в нашем хранилище и помогают решать важные задачи: искать причины проблем и моделировать новые алгоритмы рекомендаций.

Напрямую с источником данных не стоит работать по двум основным причинам. Во-первых, запросы к данным на чтение оказывают очень большую нагрузку на диски и увеличивают время ответа рабочих машин, и клиенты получают ответы ваших систем с задержкой. Во-вторых, может быть нарушена конфиденциальность данных, хранящихся в источниках. Не все данные нужно забирать оттуда в исходном виде, чувствительную информацию клиентов лучше не трогать или шифровать при загрузке в хранилище. Само хранилище проводит незримую границу между вашей рабочей системой, которая должна работать надежно, и данными, которые будут использованы для анализа. В [Ozon.ru](https://www.ozon.ru) у меня был один раз случай, когда мой сотрудник, обращаясь напрямую к источнику данных, повредил данные клиента — разработчики тогда очень разозлились.

Неизменяемость уже загруженных данных в хранилище гарантирует, что ваши аналитические отчеты не будут меняться. Я буду лукавить, если скажу, что так не бывает. Бывает, но обычно из-за технических ошибок или расширения перечня хранимых данных. Такие инциденты нужно минимизировать по двум причинам. Во-первых, перезагрузка данных бывает очень длительной и блокирующей аналитическую систему. Например, у нас в Retail Rocket такая операция между Hadoop и Clickhouse могла занимать дни и даже недели. Во-вторых, доверие пользователей к вашей системе

будет подорвано из-за изменения данных, а значит, отчетов и решений, которые были сделаны на их основе. Легко ли вам будет доверять данным, которые изменяются задним числом?

Я всегда стараюсь хранить данные в том виде, в котором они хранятся в источнике. Есть другой подход — делать преобразование данных при их копировании в хранилище. С моей точки зрения, второй подход имеет существенный недостаток: никто не может гарантировать, что преобразование или фильтрация пройдут без ошибок. Как минимум, данные в источнике изменятся, и преобразование «устареет». В какой-то момент вы заметите, что данные расходятся. Вам придется лезть в источник, возможно, скачивать их (а их может быть очень много) и построчно сравнивать. Такие ошибки крайне сложно искать. Поэтому удобно, когда исходные данные хранятся в сыром виде. Безусловно, преобразования нужны, но хранить измененные данные лучше в отдельных таблицах или файлах (в отдельном слое хранилища). Тогда сверка с источником будет заключаться лишь в сравнении числа строк между таблицами, а ошибки преобразований легко отыщутся внутри самого хранилища, так как исходники у вас имеются. Этот вывод был сделан мной на основе инцидентов по исправлению данных во всех компаниях, на которые я работал. Да, данных в хранилище может стать чуть ли не в два раза больше, но, учитывая сегодняшнюю низкую стоимость хранения и принцип «много данных не бывает», это все окупится.

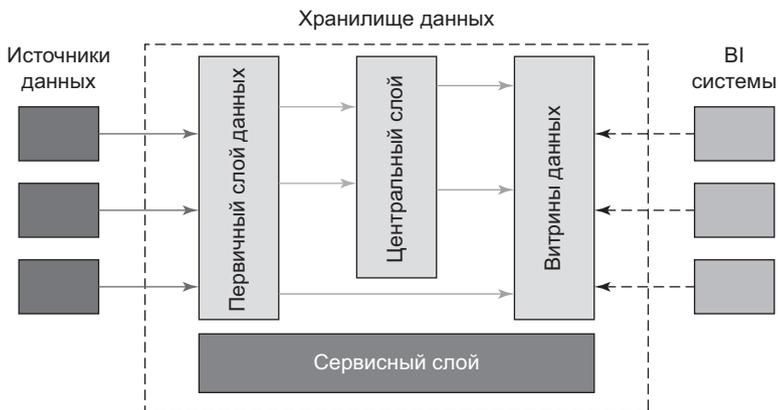
В главе 5 я уже писал про связность данных, что самые интересные инсайты находятся на стыке их разных источников. Данные объединяются через ключи. Сама операция называется соединением данных (join). Она очень ресурсоемкая, разработчики баз данных постоянно работают над ее ускорением. На одной из лекций в компании Microsoft я услышал, что для больших данных количество таких операций нужно минимизировать, а для этого нужно сразу соединить данные в хранилище и в таком виде хранить. Это было около десяти лет назад. Сейчас уже есть системы, которые это могут делать лучше традиционных баз данных, об этом позже в этой главе.

Однажды у меня был разговор с разработчиком из компании Netflix. Я рассказывал ему про хранилища данных, а он остановил меня и сказал: «А не проще ли восстановить базу из бэкапа и с ней работать?» Во-первых, если вы не пользуетесь облачными сервисами, как это делает Netflix, то восстанавливать данные из бэкапа не так легко. Во-вторых, хранилища часто содержат свои агрегаты (о них расскажу позже), которые нужно поддерживать. В-третьих, если источников несколько и это разные базы данных или хранилища файлов, то будет невозможно делать запросы с соединением этих источников.

## СЛОИ ХРАНИЛИЩА ДАННЫХ

Само хранилище может состоять из нескольких слоев (рис. 6.1) [34]:

- Данные из источников «как есть» (Первичный слой данных), или по-другому — сырых данных. Как я уже писал выше, лучше их хранить в виде, максимально совпадающем с источником.
- Данные, приведенные в целостную форму, независимую от источника (Центральный слой). Это означает, что мы уже можем работать с логической схемой, человеко-понятными терминами. Аналитику не придется думать, как соотносятся термины в разных источниках, все уже сделано в этом слое.
- Данные, представленные в виде витрин для определенного круга пользователей (Витрины данных). Деление происходит по предметной области: маркетинг, продажи, склад... Лучшим примером будет подготовка данных для OLAP-кубов. Данные для них приходится специально готовить. Подробнее об этом поговорим в следующей главе об инструментах анализа данных.



**Рис. 6.1.** Слои хранилища данных

Эти слои — логические, и они весьма условны; физически они могут размещаться как в разных системах, так и в одной.

В Retail Rocket я много работал с данными активности пользователей. Данные приходят в кластер в режиме реального времени в формате JSON и складываются в одну папку. Это первый слой — сырые данные. Сразу же делаем преобразование данных, чтобы было удобно работать с ними в формате CSV, складываем их во вторую папку. Это второй слой. Далее данные из второго слоя отправляются в аналитическую базу данных ClickHouse, где уже построены специальные таблицы для каких-либо задач — например, для отслеживания эффективности алгоритмов рекомендаций. Это третий слой витрины данных.

## КАКИЕ БЫВАЮТ ХРАНИЛИЩА

Если распределить все созданные хранилища по популярности использования, то я бы их расставил так:

1. Реляционные базы данных — PostgreSQL, Oracle, MS SQL Server и т. д.

2. «Колоночные» базы данных — Vertica, Greenplum, ClickHouse; облачные — Google BigQuery, Amazon Redshift.
3. Файловые — Hadoop.

Я не стал включать в список большие энтерпрайзные системы от IBM, Teradata и других вендоров, у них своя ниша, в которой я не работал.

Лично я работал с Microsoft SQL Server в Ozon.ru и Wikimart.ru, Postgres в Ostrovok.ru, Hadoop в Wikimart.ru и Retail Rocket, Clickhouse в Retail Rocket. У меня остались только положительные впечатления об этих технологиях. В последнее время я предпочитаю экономить и пользоваться открытыми технологиями (open-source).

Итак, реляционные базы данных (рис. 6.2) — рабочая лошадка многих бизнесов. Они все еще популярны, несмотря на атаку NoSQL-технологий. Данные в них хранятся в таблицах с колонками и строками, между таблицами есть «отношения», также можно выставлять логику при операциях с полями и т. д. Оптимизация производительности делается с помощью настройки индексов

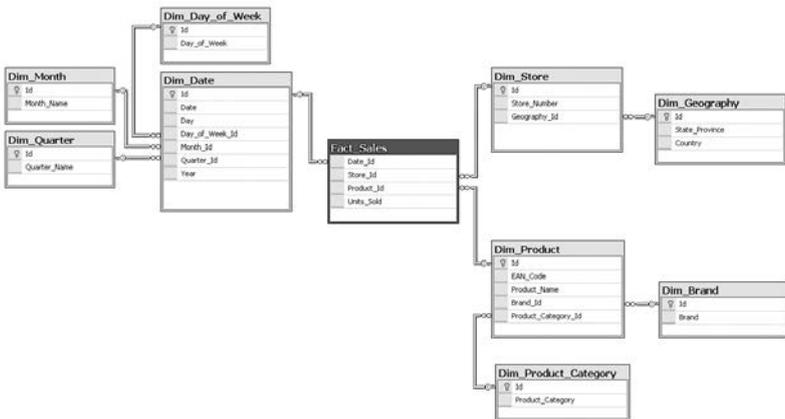


Рис. 6.2. Таблицы реляционной базы данных

и различных хаков, которые у каждого вендора свои. Они популярны, с ними умеют работать множество специалистов. Самое главное, что базы этого типа привнесли в мир, — это язык программирования SQL (Structured Query Language). Этот язык может использоваться как для внесения изменений, так и для получения данных. Минус реляционных баз — низкая производительность, когда идет работа с большими данными. И в Ozon.ru, и в Wikimart.ru я держал данные по статистике посещения сайтов в обычной таблице, а это миллиарды строк. Можно было сходить на обед, пока считался нужный запрос.

На помощь реляционным базам данных, в которых данные хранятся построчно [35], пришли колоночные базы. И совершили революцию. Вот пример таблицы реляционной базы (табл. 6.1).

**Таблица 6.1.** Пример данных реляционной таблицы

ID	Name	Type	Color
1	Стул	Кухонная мебель	Белый
2	Стол	Кухонная мебель	Красный
3	Стул	Гостиная	Синий

Чтобы найти все товары с типом (колонок Type) «Гостиная», придется прочитать все строки данных (если нет индекса по этому столбцу), что удручает, если их миллиарды и даже триллионы. В колоночной базе данные хранятся по столбцам отдельно, отсюда и название — колоночные:

ID: 1 (1), 2 (2), 3 (3)

Name: Стул (1), Стол (2), Стул (3)

Type: Кухонная мебель (1), Кухонная мебель (2), Гостиная (3)  
Color: Белый (1), Красный (2), Синий (3)

Теперь, если делать выборку товаров по типу «Гостиная», придется прочитать только нужный столбец. Но как найти нужные записи? Это не проблема: каждое значение столбца содержит номер записи, к которому оно относится. Дополнительно значения столбцов можно отсортировать и даже сжать, что делает такие базы еще быстрее. В них тоже есть минус — требовательность к «железу» сервера, на котором она работает. Колоночные базы идеально подходят для аналитических задач, которые изобилуют запросами по фильтрации и агрегации данных.

Если нужна «молотилка» для совсем больших данных — то это Hadoop. Она была разработана в компании Yahoo, которая сделала эту технологию общедоступной. Эта система работает на распределенной файловой системе HDFS и не требовательна к оборудованию. Для Hadoop компания Facebook разработала SQL-движок под названием Hive, позволяющий писать запросы к данным на языке SQL. Главное преимущество Hadoop — большая надежность. Минус — медленная скорость, зато там, где обычная БД не справится с каким-то запросом, Hadoop доведет дело до конца, медленно, как черепаха, но верно. Я использовал Hive и был доволен результатом, мне удавалось даже писать на нем простейшие алгоритмы рекомендаций.

Теперь главный вопрос — когда и какие технологии использовать?

Мой рецепт следующий:

- Если данных относительно немного и нет таблиц с миллиардами строк, то проще использовать обычную реляционную базу.
- Если данных больше миллиарда строк или требуется хорошая скорость для аналитических запросов (агрегация и выборки) — то лучше всего использовать колоночную базу данных.
- Если требуется хранить очень большой объем с сотнями миллиардов строк, вы готовы мириться с медленной скоростью или хотите иметь архив исходных данных — то Hadoop.

В Retail Rocket использовался Hive поверх Hadoop для служб технической поддержки, но запросы могли выполняться больше получаса. Поэтому требуемые ими данные были переведены в Clickhouse (колоночная база данных), и обработка запросов ускорилась в сотни раз. А скорость очень важна для интерактивных аналитических задач. Сотрудники Retail Rocket мгновенно полюбили новое решение за его скорость. Hadoop при этом остался в качестве рабочей лошади для расчета рекомендаций.

## КАК ДАННЫЕ ПОПАДАЮТ В ХРАНИЛИЩА

Чтобы данные оставались актуальными, их нужно периодически обновлять. Делать это можно путем полного или инкрементального обновления.

Полное обновление применяют к справочникам и состояниям на определенный момент времени (об этих типах я писал в главе 5). Выглядит оно следующим образом — данные скачиваются из источника в промежуточную таблицу или папку (staging) хранилища, и если операция прошла успешно, эти старые данные меняются на свежие. Плюсом такого типа обновления является полное соответствие данных источнику на момент скачивания. Первым минусом является время для обновления таблиц, которое напрямую зависит от размера исходных данных. Второй минус — потеря изменений после обновления данных. Представьте, что у вас есть справочник с товарами магазина, в исходной системе он всегда актуален, а в хранилище обновляется каждый день в полночь. Если какой-то товар был заведен и удален из справочника — вы этого не заметите в хранилище. Для аналитика это будет выглядеть так, будто такого товара не существовало. Заказы этого товара могли совершаться. Поэтому при анализе заказов возникнет ситуация нарушения целостности данных: в таблице этот заказ есть, в справочнике товаров — нет.

Инкрементальное обновление данных применяется для лога изменений (я писал о нем в главе 5). Конечно, можно его полностью перезагружать, но обычно это нецелесообразно из-за большого объема, а следовательно, медленной скорости обновления. Само инкрементальное обновление выглядит следующим образом: система смотрит в хранилище, какие данные были загружены последними (это может быть время или какой-то идентификатор). Далее делается запрос к источнику: «Отдай мне данные, которые поступили после момента X». После получения эти данные добавляются к старым в хранилище. Плюсом инкрементального обновления является скорость. Минусом — возможна рассинхронизация данных, когда были добавлены лишние записи, которые уже были, или, наоборот, произошла потеря данных.

Принципиально есть два типа инкрементального обновления: пакетный и потоковый.

Пакетный (batch) — данные загружаются большими блоками. Например, таблица с заказами клиентов обновляется раз в сутки в первый час ночи — происходит загрузка данных за последние сутки, и они добавляются в хранилище. Вариантов реализаций таких обновлений много — от специализированного софта до самописных систем. Я использовал оба подхода. В [Ozon.ru](https://www.ozon.ru) и [Wikimart.ru](https://www.wikimart.ru) использовался Microsoft Integration Services, которые входили в пакет MS SQL Server. В ней просто мышкой рисуются потоки данных между источником и хранилищем, программирования там минимум. В Retail Rocket использовалась самописная утилита без графического интерфейса, которая занималась скачиванием данных из Hadoop в ClickHouse.

Потоковый (stream) — данные загружаются по одной записи или мелкими пакетами по мере их поступления. Обычно это создается для данных, которые нужно поддерживать в «реальном времени». Если обратиться к примеру выше, то в таблицу с заказами клиентов новые заказы будут попадать по мере их появления. В таком случае будет небольшое запаздывание между созданием самого

заказа и его появлением в хранилище. Оно может варьироваться от миллисекунд до часов, и это время обязательно контролируется через мониторинг.

## HADOOP И MAPREDUCE

Сначала расскажу, как появился Hadoop. К этому приложил руку Google. На мой взгляд, у этой компании есть два великих изобретения: PageRank и MapReduce. Джеффри Дин и Санджай Гемават (Jeffrey Dean и Sanjay Ghemawat) в течение четырех месяцев 2003 года разрабатывали технологию MapReduce [38]. Эта идея пришла к ним, когда они в третий раз переписывали движок Google, отвечающий за скачивание и индексацию страниц. Эти два разработчика решили важную проблему: они нашли способ скоординировать работу огромного числа зачастую ненадежных компьютеров в разных дата-центрах по всему миру. Ведь отказ одного или нескольких компьютеров потребует повторного запуска расчетов, что очень проблематично, когда мы имеем дело с тысячей машин. Но самое замечательное, что Дин и Гемават не просто нашли решение частной проблемы — они создали философию. MapReduce позволил разработчикам Google обращаться к серверам их дата-центров как к одному огромному компьютеру.

До изобретения MapReduce любой разработчик должен был придумывать схему, как разделить и распределить данные, запускать расчет и самостоятельно разбираться с отказом оборудования. MapReduce предложил новый принцип решения задач. Алгоритм требовал разбивать задачу на два этапа. Этап «Мар» (предварительная обработка) — программист сообщает каждой машине, какую

предобработку данных выполнить, например, посчитать, сколько раз слово «котик» встретилось на веб-странице. Затем нужно написать инструкции для этапа «Reduce» (свертка), например, заставить машины вычислить суммарное количество «котиков» на всех веб-страницах мира.

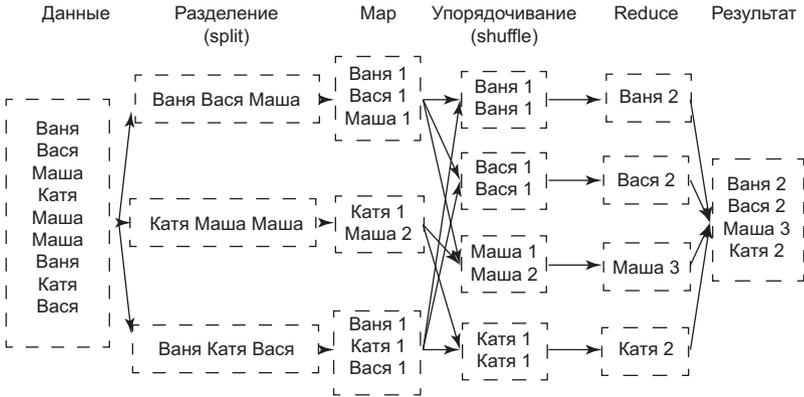
В 2004 году индексирующий движок Google был переведен на MapReduce. Затем эту технологию стали использовать для обработки видео и рендеринга карт Google Maps. Она была настолько проста, что ее стали использовать для широкого круга проблем. В том же году со стороны Google был заявлен патент [36] на MapReduce. Тогда же Джеффри и Санджай подумали, что было бы полезно познакомить астрономов, генетиков и других ученых, у которых очень много данных, с MapReduce. Они написали и опубликовали статью: «MapReduce: упрощенная обработка данных на больших кластерах» [37].

Статья произвела эффект разорвавшейся бомбы. Дешевое железо, рост числа веб-сервисов и подключенных устройств к Сети привели к «потопу» данных. На рынке было только несколько компаний с программными технологиями, которые могли справиться с этим. Дуг Каттинг и Майк Кафарелла (Mike Cafarella and Doug Cutting) работали над масштабированием своего поискового движка Nutch. Они были так впечатлены статьей, что на ее основе с нуля написали проект Hadoop. Затем Yahoo приглашает Каттинга продолжать работу над проектом внутри компании. В 2008 году начинается широкое применение Hadoop технологическими компаниями. Apache Hadoop сейчас распространяется под свободной лицензией [39].

Hadoop используется в большинстве технологических компаний, работающих с большими данными. Если не дистрибутив Apache,

то какой-нибудь коммерческий от MapR, Cloudera или другого вендора. Некоторые пошли своим путем и сделали собственную реализацию, например Яндекс.

Понять, как работает MapReduce, поможет иллюстрация (рис. 6.3).



**Рис. 6.3.** Подсчет числа слов в тексте

Слева у нас есть исходный текст, в каждой строке которого встречаются имена людей. Первая операция, Split, разрезает текст по строкам, каждая строка обрабатывается независимо от других. Вторая операция, Map, считает количество упоминаний каждого имени в строке. Ее мы можем проводить параллельно на разных машинах, так как строки независимы друг от друга. Третья операция, Shuffle, раскидывает одинаковые имена в группы. Четвертая операция, Reduce, считает сумму упоминаний каждого имени в разных строках. На выходе мы получаем число упоминаний каждого имени в тексте. Этот пример написан на трех строках, но с триллионом строк все операции были бы такими же.

MapReduce — это концепция. Hadoop — это программное обеспечение, которое реализует эту концепцию. Сам Hadoop состоит из двух главных компонент: распределенной файловой системы HDFS и планировщика ресурсов Yarn.

Файловая система HDFS (Hadoop Distributed File System) для пользователя выглядит как обычная файловая система с папками и файлами, которую вы привыкли видеть в своих компьютерах. Сама система располагается как минимум на одном компьютере. В ней есть две главные роли — name node (центральный узел имен) и data node (узел данных). Когда пользователь хочет записать файл в HDFS, происходит разбиение файла на блоки (размер блока зависит от настройки системы), name node возвращает data node, в который нужно сохранить блок. Клиент отправляет данные на data node, после записи данные реплицируются — копируются на другие ноды. По умолчанию коэффициент репликации составляет 3, то есть один блок данных будет на трех узлах данных. Как только процесс завершится и все блоки будут записаны, name node сделает соответствующую запись в своих таблицах (где какой блок хранится и к какому файлу относится). Это дает защиту от ошибок, например, когда сервер выходит из строя. С коэффициентом репликации 3 мы можем безболезненно потерять две ноды. Кстати, в таком случае HDFS самостоятельно обнаружит такие ноды и начнет реплицировать данные между «живыми» нодами, чтобы снова достичь нужного уровня репликации. Так мы достигаем устойчивости расчетов с точки зрения данных.

Планировщик ресурсов YARN отвечает за распределение вычислительных ресурсов на кластере Hadoop. Благодаря ему мы можем запускать на одном кластере несколько задач параллельно. Сами вычисления происходят, как правило, там же, где находятся данные, на тех же самых нодах с данными. Это экономит много времени, так как скорость чтения данных с диска гораздо выше, чем скорость копирования их по сети. При запуске задачи через Yarn ему явно нужно указать, сколько ресурсов для расчета вам нужно: сколько машин (executors) из кластера, сколько ядер процессора (cores) на каждой машине и сколько памяти. Сам Yarn также предоставляет отчет в реальном времени о выполнении задачи.

Про Hadoop мне рассказали в офисе Netflix в 2011 году. Я сразу стал искать и читать документацию по этому сервису, смотрел на

YouTube конференции о том, как с ним работать. В качестве эксперимента я развернул Hadoop на своем рабочем ноутбуке, выбрав в качестве дистрибутива версию от Cloudera. Удобство Hadoop в том, что его можно поставить хоть на ноутбуке — все сервисы будут крутиться на нем. По мере увеличения объема данных можно легко добавлять сервера, причем даже используя самые дешевые. Именно так я и поступил, когда начал писать рекомендательный движок Retail Rocket. Начал я с одного или двух серверов, пять лет спустя кластер Hadoop вырос до 50 машин и содержит порядка двух петабайт сжатых данных.

В начале пути я пользовался двумя языками программирования для Hadoop — Pig и Hive. На них была написана первая версия рекомендательного движка, затем мы перешли на Spark и стали писать на Scala.

Не пренебрегайте принципами MapReduce. Однажды мне они пригодились, когда я участвовал в одном из соревнований Kaggle. Датасет был очень большой, он не помещался полностью в память, пришлось писать предварительную обработку на чистом Python, используя подход MapReduce. Тогда у меня это заняло много времени, сейчас я бы поставил локально фреймворк Spark и не изобретал бы велосипед. Это сработает, ведь MapReduce-операции можно выполнять как параллельно на разных машинах, так и последовательно. Сами вычисления займут много времени, но они хотя бы посчитаются, и у вас не будет болеть голова, хватит ли памяти для расчетов.

## SPARK

С фреймворком Spark я познакомился в 2012 году, когда приобрел для корпоративной библиотеки Ostrovok.ru видеозаписи конференции по анализу данных Strata. Эту конференцию организует издательство O'Reilly в США. На одной из лекций я увидел, как Матей Захария (основной автор Spark) рассказывает о преимуще-

ствах Spark над чистой реализацией MapReduce на Hadoop. Самое главное преимущество в том, что Spark загружает данные в память, в так называемые отказоустойчивые распределенные датасеты RDD (resilient distributed dataset), и позволяет работать с ними в памяти итеративно. Чистый Hadoop же полагается на дисковую память — для каждой пары операций MapReduce-данные читаются с диска, затем сохраняются. Если алгоритм требует применения еще нескольких операций, то для каждой из них придется читать данные с диска и сохранять обратно. Spark же, совершив первую операцию, сохраняет данные в памяти, и последующие операции MapReduce будут работать с этим массивом, пока программа не прикажет явно сохранить их на диск. Это очень важно для задач машинного обучения, где используются итеративные алгоритмы поиска оптимального решения. Все это дало огромный прирост производительности, иногда в 100 раз быстрее классического Hadoop.

Эти идеи из лекции Матея Захарики мне пригодились через несколько лет, когда мы стали писать вторую версию рекомендательного движка Retail Rocket. Тогда как раз вышла версия 1.0.0. Летом 2014 года мы попробовали запустить Spark поверх Hadoop — и все получилось. В экспериментах мы достигли 3–4-кратного ускорения. Была, правда, проблема с производительностью на большом количестве мелких файлов. Мы написали небольшую библиотеку, которая склеивала их при загрузке, и проблема решилась [41]. Сейчас мы по-прежнему используем Spark на нашем Hadoop-кластере и ни о чем не жалеем.

При переходе с чистого Hadoop на «Spark поверх Hadoop» пришлось отказаться от большого количества кода на языке Pig, Hive, Java, Python — весь этот зоопарк доставлял большую головную

боль, ведь их все должны были знать. Для прототипирования задач машинного обучения мы использовали связку питоновских инструментов: IPython + Pyhs2 (hive-драйвер Python) + Pandas + Sklearn. Со Spark мы смогли начать пользоваться только одним языком программирования для прототипирования экспериментальных функций и для рабочего варианта. Это было очень большое достижение для нашей небольшой команды.

Spark поддерживает четыре языка программирования из коробки Python/Scala/Java/R через API. Сам Spark написан на Scala, поэтому мы выбрали его. В таком случае мы сможем свободно читать исходный код Spark и даже исправлять ошибки в нем (это нам не пригодилось). Кроме того, Scala принадлежит к семейству JVM-языков, что очень удобно при работе с файловой системой Hadoop напрямую через API, так как он написан на Java.

Ниже приведено сравнение языков программирования для Spark, где + и – означают плюсы и минусы языка соответственно.

### Scala:

- + функциональный, поэтому очень удобный для обработки данных любых объемов;
- + родной для Spark, это важно, если нужно понимать работу Spark изнутри;
- + основан на JVM, что делает его совместимым с Hadoop;
- + строгая типизация, тогда компилятор поможет найти часть ваших ошибок;
- труден в освоении, нам пришлось разработать свою программу обучения для новичков [19];

- сложный наем — разработчики на Scala дороже, чем на Java или Python;
- сам язык не настолько распространен, как Java или Python.

**Python:**

- + популярный;
- + простой;
- динамическая типизация, ошибки, которые мог обнаружить компилятор;
- производительность хуже, чем Scala;
- нет чистой функциональности, как у Scala.

**Java:**

- + популярный;
- + родной для Hadoop;
- + строгая типизация;
- нефункционален (на самом деле после Java 8 все стало намного лучше).

Когда писалась вторая версия рекомендательного движка Retail Rocket, решение выбрать Scala основным языком программирования далось непросто, потому что этот язык никто не знал. Если бы мне пришлось выбирать сейчас, возможно, я бы посмотрел в сторону Java 8 и выше (версий), поскольку Java-разработчиков проще найти, чем тех, кто знает Scala.

Сейчас Spark идет в сторону дата-фреймов и дата-сетов (dataframe dataset), моду на которые сделала библиотека pandas [42] для Python — самая популярная для анализа данных. На них проще писать, но есть один нюанс. Компилятор не может проверить корректность работы с внутренними переменными, что не очень хорошо для больших проектов.

## ОПТИМИЗАЦИЯ СКОРОСТИ РАБОТЫ

Если пользователя не устраивает скорость ответа аналитической системы, значит, пора оптимизировать скорость работы хранилища. Решение может быть как простым, так и сложным. Первое, что необходимо понять, — можно ли обойтись малой кровью. В реляционных базах данных можно добавить индексы. В базах данных для этого имеются так называемые профайлеры, которые на основе запросов пользователей предложат, какие именно индексы добавить. В колоночной базе данных Clickhouse можно сменить схему партиционирования или сделать сэмплирование, когда запрос будет использовать только часть данных. В Hadoop можно выделить больше ресурсов или даже оптимизировать программный код. Я практиковал оба способа.

Кроме этого, есть способ, работающий практически везде. Главный враг скорости запросов — это соединение данных (join) в разных таблицах. Например, у нас есть две таблицы: клиенты и заказы, обе таблицы соединяются через id клиента, имеющийся в обеих таблицах. Почему бы не делать это соединение данных периодически, например по ночам, и сохранять результат обратно в хранилище? Это называется материализованным представлением (materialized view). Тогда клиенты будут обращаться не к данным напрямую, а к их представлению. Скорость будет на порядок выше — это плюс. Минус такого решения — усложнение всей конструкции. Если что-то пойдет не так и таблицы-доноры будут иметь неверные данные, эти представления придется пересчитать. Это нужно будет всегда держать в голове.

Более радикальное решение — сменить технологию на ту, которая больше соответствует задачам. Я уже приводил пример, когда с Hadoop мы перевели пользователей на колоночную базу ClickHouse и получили ускорение в десятки и сотни раз.

Еще один дорогой способ ускорить работу аналитической системы — сделать апгрейд «железа сервера». В системах Hadoop и Clickhouse это легко сделать, просто добавив дополнительные

машины. Такая операция называется горизонтальным партиционированием (sharding) — данные разбиваются по записям и распределяются по серверам. Запросы к данным будут выполняться параллельно на нескольких серверах, а затем результаты объединяются. Такая схема теоретически может дать линейное ускорение: два сервера будут работать быстрее в два раза по сравнению с одним сервером и т. д.

## АРХИВАЦИЯ ДАННЫХ И УСТАРЕВАНИЕ

Следующая часто возникающая проблема с хранилищами данных — большой рост данных, который приводит к нехватке свободного места. Этим нужно постоянно заниматься. Много данных не бывает, но бывает мало бюджета. Какие стратегии существуют?

Первая и самая простая стратегия — удаление устаревших данных. Это могут быть данные старше двух лет, старше пяти лет — все зависит от ваших задач. В Hadoop можно использовать другую стратегию: менять фактор репликации. По умолчанию он равен трем — это означает, что для хранения одного терабайта данных вам понадобится три терабайта на дисках. Это цена надежности, в случае выхода из строя двух дата-нод одновременно вы не потеряете данные. Фактор репликации можно устанавливать индивидуально для файлов: мы можем его уменьшить для более старых. Например, данные младше двух лет — фактор равен трем, от двух лет до четырех — двум, старше четырех лет — удаляются. Подобный подход используется в Facebook. Некоторые компании архивируют старые данные на какой-нибудь дешевый носитель, а если данные понадобятся, они переносятся обратно. Я не поддерживаю такую схему — это как прятать вещи в чулан: потом о них забываешь, а если вспомнишь, то искать их там лень.

Второй способ — использование кодеков сжатия (табл. 6.2) [43]. Это очень актуально и эффективно работает в Hadoop и Spark.

Сжатие данных убивает двух зайцев — мы уменьшаем объем занимаемого места на дисках и ускоряем работу с данными: они в разы быстрее гоняются по сети между серверами кластера и быстрее читаются с диска. Чудес не бывает — чем сильнее жмет кодек, тем больше ему нужно ресурсов процессора для сжатия данных.

Мы на своем кластере используем кодеки: gzip, bzip2 и lzma. Lzma имеет самую высокую компрессию и используется для архивирования данных. Gzip используется для всех остальных данных, поступающих в кластер. От конкретного кодера сжатия зависит возможность «разрезания» (split) файла для операции Map без его распаковки. Как уже писалось ранее, для операции Map данные «нарезаются» на блоки размером не больше заданного в настройках Hadoop (block size). Если сжатый файл больше этого размера, то в случае делимого кодера (splittable codec) его можно разрезать и распаковать по частям на разных нодах кластера параллельно. В противном случае придется распаковывать этот огромный файл целиком — а это уже будет гораздо медленнее.

**Таблица 6.2.** Сравнение кодеров сжатия

Кодек	Расширение файла	Разделимый? (splittable)	Степень сжатия	Скорость сжатия
Gzip	.gz	Нет	Средняя	Средняя
Bzip2	.bz2	Да	Высокая	Медленная
Snappy	.snappy	Нет	Средняя	Быстрая
LZO	.lzo	Нет, без индекса	Средняя	Быстрая
LZMA	.xz	Да	Высокая	Очень медленная

## МОНИТОРИНГ ХРАНИЛИЩ ДАННЫХ

Пользователи вашей аналитической системы могут принимать очень важные решения на основе данных, поэтому важно обеспечить их надежными данными.

Однажды у меня произошел нехороший случай: в пятницу вечером я произвел изменения в системе пополнения данных в хранилище в Ozon.ru. И ушел в отпуск. Конечно, на выходных все упало. Ребята-аналитики получили в понедельник письмо от генерального директора на английском, которое начиналось словами: «I'm fed up...» (Я сыт по горло...). Они, конечно, нашли причину проблемы и исправили. Как следовало мне поступить? Во-первых, не делать никаких изменений в пятницу, тем более перед отпуском. Если бы я это сделал хотя бы в четверг, то в пятницу утром изменения «сломали» бы систему и у меня было бы время все исправить. Во-вторых, если бы была полноценная система мониторинга, то разработчики первыми получили бы сообщения о проблеме. У них была бы возможность предупредить пользователей до того, как те ее сами заметят.

Когда я проверял задачи по анализу данных или делал их сам, то периодически мучился вопросом: «А все ли в порядке с данными?» Иногда эти сомнения оправданны, и проблема действительно существует. Это заставляет нас первым шагом делать проверку. Но она не всегда бывает простой и может занять приличное время. Есть второй путь — автоматизация проверок данных и мониторинг. Вот об этом и поговорим.

Есть два параметра, которые нужно проверить:

- доступность всех данных, которые есть в источнике;
- целостность.

Доступность данных проверяется легче всего. Во-первых, проверяется дата последнего обновления, например файла или таблицы. А еще лучше воспользоваться полем с датой/временем — например

датой и временем заказа. Во-вторых, можно посчитать и сравнить количество записей в хранилище данных и в источнике. Если есть поле с датой и временем, можно сделать такое сравнение по дням. Конечно, в момент проверки данные источника и хранилища будут расходиться, потому что всегда есть дельта времени изменения данных в источнике и отражения этих изменений в хранилище. Если вы уверены, что с данными все в порядке, можно опытным путем найти допустимые пороговые значения относительной разницы в процентах. Для одних данных это может быть полпроцента, для других — все пять. Этот тип проверки закроет 80 % проблем с данными в хранилище. Это как раз те 20 % усилий по Парето, которые дают 80 % результата. Методика недорогая, доступна всем и нравится мне своей простотой.

Второй параметр — целостность данных — проверить сложнее. Под целостностью я подразумеваю наличие в одной таблице тех данных, которые есть во второй. Простейший пример — есть две таблицы, одна с клиентами, вторая с заказами. В таблице с заказами есть поле «клиент». Целостность таблицы с заказами будет означать, что все клиенты из этой таблицы должны быть также и в таблице — справочнике клиентов. Если это соответствие не соблюдается, то при соединении (join) двух таблиц либо возникнут «битые» данные (если соединение было сделано с учетом этой особенности — left join, right join, full outer join), либо эти данные просто исчезнут и эти продажи выпадут из отчетов (если соединение сделано через inner join). Оба этих результата потенциально могут привести к неверным решениям. Хорошо бы это контролировать через независимые тесты, например проверять относительный объем продаж клиентов, которых нет в таблице с клиентами.

## ЛИЧНЫЙ ОПЫТ

Не надо бояться. Свое первое хранилище я стал собирать в 2004 году в Ozon.ru. Мне в работе очень помогло обучение «MS SQL Server» в «Софтлайне», когда я еще работал в StatSoft. Этот

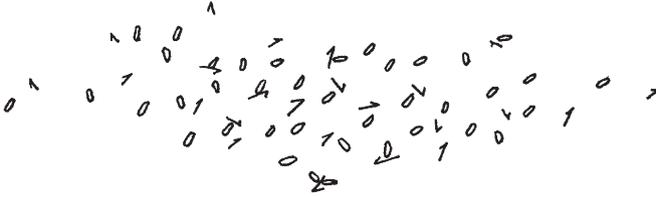
сертификат хранится у меня до сих пор. Я ничего практически не знал об этом, но знакомство с SQL Server и опора на здравый смысл сделали свое дело — я создал своего первого «паука», который закачивал данные в наспех собранное хранилище. Мне никто не помогал в этом, но никто и не мешал, что очень важно. Схема хранилища модифицировалась, но ее концепт, заложенный в самом начале, остался прежним. В Wikimart.ru я, работая два дня в неделю, собрал первую версию аналитической системы с полной внутренней веб-аналитикой всего за два месяца. Если вы хотите лучше узнать принципы построения хранилищ, рекомендую обратиться к трудам Ральфа Кимбалла — я в них почерпнул много полезного.

А теперь о сложностях. К моменту моего ухода из Ozon.ru расхождение данных о продажах в хранилище с бухгалтерией составляло 4–5 %. При этом бухгалтерия закрывала период в течение месяца, а данные в кубах аналитической системы были уже в первый день следующего месяца. После ухода из Ozon.ru я встречался с операционным директором «Связного» — целая небольшая команда пришла пообщаться со мной по поводу «строительства кубов». Они очень удивились тому, что я сделал весь основной движок в одиночку. Это не я такой крутой, это вопрос допустимой погрешности системы. Чем меньшего процента расхождения с бухгалтерией мы хотим достичь, тем сложнее его получить. Допустим, нужно уменьшить расхождение с 4 до 3 %. Это потребует большего вовлечения меня, найма одного-двух человек, усложнения системы, а следовательно, увеличения управленческой «энтропии». Если мы хотим продолжать дальше — спуститься до двух процентов, это потребует уже на порядок больше усилий. Каждое уменьшение будет требовать усложнения и удорожания по экспоненте. Но что мы теряем? Мы теряем гибкость, мы теряем маневренность. Не нужно молиться на нулевую погрешность, ее никогда не будет. Помните про правило Парето — 20 % усилий дают 80 % результата, и не факт, что остальные 80 % усилий стоит затрачивать. Возможно, стоит потратить их на что-то другое, что сделает нас ближе к цели, а не стремиться к идеально вылизанным цифрам.



# 7

## ИНСТРУМЕНТЫ АНАЛИЗА ДАННЫХ



Как вы помните из предыдущих глав, классическая аналитика данных делится на два этапа — поиск гипотез и их статистическая проверка. Для формирования гипотез нам понадобятся описательная статистика, визуализация данных и доменные знания, например, какие события в компании произошли.

Для первых двух пунктов существует много видов программного обеспечения, которое облегчает и ускоряет труд аналитика. Здесь я рассмотрю диаметрально разные подходы. Можно быть приверженцем только одного из них, но расширить кругозор полезно — вдруг альтернативный подход будет удобнее.

Я делю эти инструменты по способу взаимодействия с пользователем. Это деление весьма условно, так как некоторые категории могут пересекаться друг с другом.

- Электронные таблицы — Microsoft Excel, Open Office Calc, Google Docs Sheets.
- Программные сервисы блокнотов, например Jupyter Notebook, Netflix Polynote, Google Colab, R studio.
- Визуальные инструменты — Tableau, Google Data Studio, Yandex Data Lens, Microsoft Power BI, Amazon QuickSight.
- Специализированные статистические пакеты — SAS, SPSS, Statistica, Knime, Orange data mining.

## ЭЛЕКТРОННЫЕ ТАБЛИЦЫ

Электронные таблицы — одни из самых распространенных инструментов анализа данных. Впервые я познакомился с ними еще в 1997 году, когда делал таблицы для школьного реферата по географии в Quattro Pro, чем произвел впечатление на учительницу географии. Я много дней ходил на работу к отцу, который в то время занимался IT-технологиями, набирал текст на клавиатуре и масштабировал карты стран на ксероксе. В итоге получился полностью напечатанный реферат, что было очень примечательно

в те времена, особенно в Твери. Затем я работал с Microsoft Excel, потом в Google Sheets, которые сделали очень легкой совместную работу в облаке. В чем плюс электронных таблиц:

- низкий порог входа;
- все делается интуитивно и наглядно: добавление нового столбца или формулы;
- есть возможность анализа сводных таблиц (pivot) — самый мощный инструмент генерации гипотез;
- очень легко делать графики.

Основной минус — электронные таблицы не предназначены для автоматизации задач с использованием программирования. Когда такой проект усложняется, его поддержка превращается в ад для разработчиков. Внутри электронных таблиц появляется много программного кода, который находится вне системы контроля версий, поэтому отслеживать его изменения невозможно. У меня был опыт «изготовления» сложных отчетов для еженедельных собраний директоров в Ozon.ru и Wikimart.ru на основе электронных таблиц. В Ozon.ru к таблице подключались источники ручного ввода данных менеджерами компании, OLAP-кубы, SQL-скрипты. В определенный момент отчет обновлялся программным способом и сохранялся в папку с нужной датой. В Wikimart.ru плюс к тому аналитики вручную корректировали отчет и конвертировали его в PDF-файл для презентаций. Сейчас я бы по максимуму отказался от этого в пользу более управляемых решений, пусть и менее гибких, чем электронные таблицы. И договорился бы с менеджментом об изменении шаблона отчетов так, чтобы сделать его более инженерным способом.

## СЕРВИСЫ БЛОКНОТОВ

Программные инструменты, которые называют блокнотами (notebooks, рис. 7.1), могут быть очень гибкими и мощными в умелых руках. Они получили популярность благодаря широкому

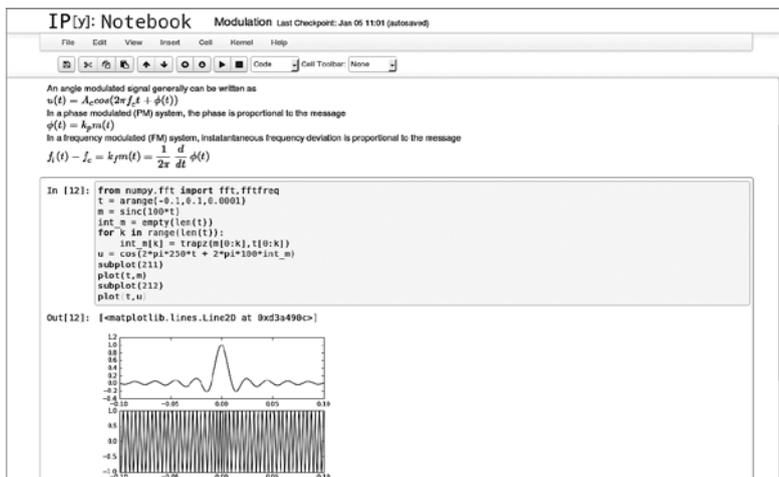


Рис. 7.1. Блокнот Jupyter

использованию языков программирования R и особенно Python для анализа данных. Блокнот запускается как веб-сервис на сервере или вашем компьютере. Он состоит из ячеек текста с программным кодом, ячейки можно запускать произвольно, весь вывод данных (графики, статистики, сообщения об ошибках) появляется под ячейкой. В ячейках можно писать тексты, делать свои заголовки. В общем, можно делать полноценные научные отчеты. Есть два наиболее известных публичных сервиса-блокнота — Google Colab и Kaggle Notebook, где вы можете попробовать этот инструмент совершенно бесплатно. В них также доступны мощные GPU-видеокарты, что позволяет делать задачи с использованием глубоких нейронных сетей (deep learning). Лично мне понравился сервис Google Colab своей простотой и мощностью, когда я проводил эксперимент по созданию deep-fake-видео.

### Плюсы:

- Гибкость. Доступны программные библиотеки на любой вкус.
- Блокнот очень легко запустить в облаке и не тратить ресурсы личного компьютера.

- Легко делиться и публиковать результаты.
- Поддержка разных языков программирования. Я в Retail Rocket использовал блокноты в Jupyter notebook на языке Scala.
- Можно взаимодействовать с любыми источниками данных, для которых есть драйверы.
- Чтобы повторить результат, блокнот достаточно перезапустить и выполнить все ячейки. Не со всеми инструментами это легко сделать. Например, в электронных таблицах могут поехать формулы. Здесь такого эффекта не будет.

### **Минусы:**

- Не считаю удачной идеей использование блокнотов как компонента рабочей системы, хотя много слышу о таких прецедентах (так делает даже Netflix [49]). Они созданы для исследовательской работы, а не построения рабочих процессов.
- Порог входа выше, чем в электронных таблицах. Как минимум нужны знания базовых основ программирования на выбранном языке.

## **ИНСТРУМЕНТЫ ВИЗУАЛЬНОГО АНАЛИЗА**

Я их также называю сервисом персональных дашбордов. Этот подход отличается от предыдущих тем, что вам практически не нужно программировать, вы можете работать через «тонкий клиент» (веб-браузер) и публиковать полученные дашборды на порталах. Самый простой из них — Google Data Studio — позволяет работать преимущественно с источниками данных в облаках Google, включая Google Sheets. Это больше чем просто средства визуализации. Power

BI и Tableau (рис. 7.2) пошли дальше — они реализовали средства ETL (Extract Transformation Layer), когда данные скачиваются из источников на машину пользователя или в облако. Power BI это делает с помощью языка программирования Power Query, Tableau через визуальный интерфейс (блоки и стрелки, их соединяющие).

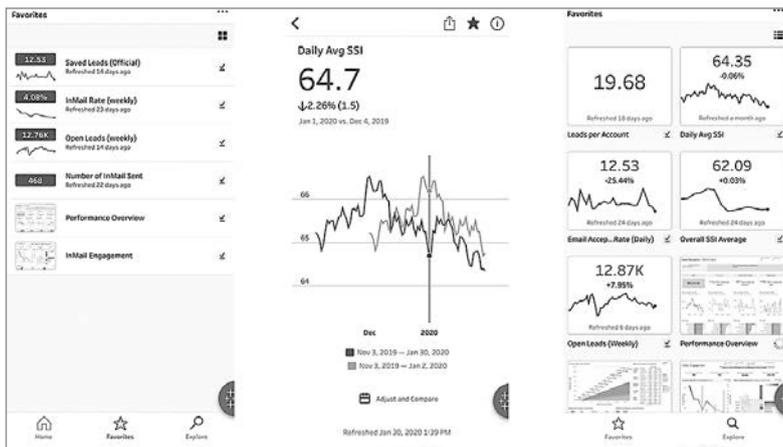


Рис. 7.2. Tableau

### Плюсы:

- Порог входа ниже, чем у блокнотов (notebooks).
- Визуализация гораздо лучше, чем у альтернативных инструментов (особенно у Tableau).
- Мощные средства интерактивного анализа — например, коррелированный анализ.
- Есть «толстые клиенты» (приложения), но не у всех вендоров.
- Можно собрать несложную BI-систему.

### Минусы:

- У бесплатных версий привязка к облаку вендора. Например, Google Dashboard привязан к облаку Google.

- Неполный набор коннекторов. Например, у Google Dashboard нет коннектора к Clickhouse. Но сама такая идея под большим вопросом из-за безопасности. Клиент Google работает из облака, а значит, придется открывать доступ к вашим внутренним БД из интернета, а это не самая лучшая идея.

## ПАКЕТЫ СТАТИСТИЧЕСКОГО АНАЛИЗА ДАННЫХ

Мое знакомство с анализом данных началось именно с этих инструментов, когда меня взяли на стажировку в компанию StatSoft. Электронные таблицы и системы визуального анализа очень слабы в статистическом анализе, а именно это является необходимым атрибутом анализа данных. Допустим, вы наблюдаете разницу в показателях — как определить, она действительно существует или случайна? Для этого нужно рассчитать ее статистическую значимость.

Пакеты стат-анализа данных обычно представляют собой десктопные приложения (рис. 7.3), в которых вычисления происходят локально. Данные загружаются в виде электронных таблиц. Как правило, есть несложный визуальный ETL, как в Tableau. Есть встроенный язык программирования для автоматизации действий.

### **Плюсы:**

- Очень богатые возможности для статистического анализа. Справка этих пакетов успешно конкурирует с учебниками по прикладному анализу данных. Сами статистические функции тщательно протестированы, в отличие от общедоступных статистических калькуляторов в интернете.
- Хорошие графические возможности.
- Внимание к деталям, что важно для научных исследований.
- С данными можно работать офлайн.

**Минусы:**

- Высокий порог входа. Вы должны понимать, что делать, какой именно статистический критерий использовать. Обязательно требуются базовые знания математической статистики.
- Коммерческие продукты стоят дорого.

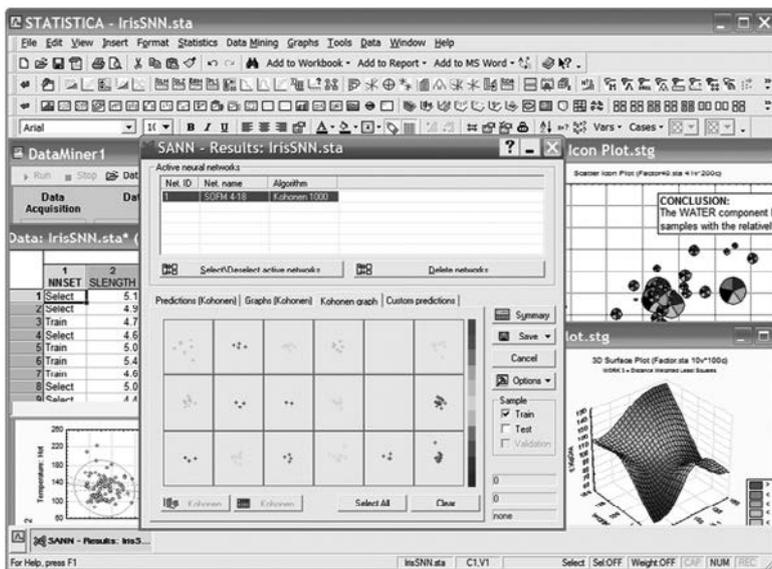


Рис. 7.3. STATISTICA

**РАБОТА С ДАННЫМИ В ОБЛАКАХ**

В эпоху развития удаленной работы все больше инструментов уходит в облака. Я связываю это с тем, что бизнесы, а значит источники данных, стали располагаться на облачных серверах. Перекачивать большие объемы данных по интернету то еще удовольствие. Согласно Гартнеру [46], к 2022 году публичные облачные сервисы закроют 90 % потребностей в анализе данных.

Уже практически все вендоры облаков разработали инструменты визуального анализа: Google Data Studio, Microsoft Power BI, Amazon Quick Sight, Yandex DataLens.

**Плюсы:**

- Данные и средства анализа находятся внутри одного периметра безопасности. Легко управлять доступом к данным. Не нужно явно подвергать себя риску и открывать доступ к данным через интернет.
- Данные доступны внутри сети одного облака — скорость работы выше.
- Нативная возможность совместной работы. Думаю, вы работали с сервисами наподобие Google Docs. Насколько удобнее получается совместная работа, чем работа со стандартным офисным пакетом.
- Тонкий клиент — все действия делаются в браузере. Не нужно ставить программы на ваш компьютер.
- Гибкое ценообразование — цена зависит от частоты использования и нагрузок.
- Расходы на администрирование системы меньше.

**Минусы:**

- Цена. Даже если облако предоставляет визуализацию бесплатно, за сами вычисления и агрегацию данных придется платить. Эта модель схожа с каршерингом: если вы очень активный пользователь, в какой-то момент становится выгоднее купить свой автомобиль. Так же и с облаками.

- Ваши данные находятся у одного вендора, а это порождает зависимость. Если объем информации составляет петабайты, то очень нелегко их перевести на свои сервера или облако другого вендора.

В целом мне нравится этот тренд — миграция данных и их анализа в облачные сервисы, это делает разработку аналитических систем легче и часто дешевле покупки корпоративных систем.

## ЧТО ТАКОЕ ХОРОШАЯ ОТЧЕТНАЯ СИСТЕМА

Опишу типичную ситуацию, которая возникает при запуске аналитической системы. В компании X появляется хранилище данных и аналитическая система к нему. Аналитики проводят первое общее собрание, показывают систему, демонстрируют, какие данные доступны. Самые любознательные сотрудники (берегите их), которым этого не хватало, начинают работать с новым хранилищем и системой, и вскоре от них начинают сыпаться комментарии: это неудобно, тут тормозит, здесь не хватает данных. Поговорим о минимальных требованиях к отчетной системе, которые я встречал на практике.

Для начала выделим две функции таких систем: предоставление дашбордов и служебных отчетов. О дашбордах я писал в прошлых главах. Служебные отчеты предназначены для автоматизации и упрощения задач сотрудника. Например, это могут быть контакты проблемных клиентов для прозвона, скоринг клиентов по эффективности внедрения системы рекомендаций на сайт, поисковые фразы с пустой страницей результатов. Эти отчеты даже встраивают как компонент в существующие бизнес-процессы.

Любой отчет, или дашборд, состоит из блоков: таблиц и графиков. Блоки часто бывают независимы друг от друга, но связаны общими параметрами. Отличный пример такого параметра — дата и время. Атрибут практически любого отчета — период, который

этот отчет охватывает. В хорошей отчетной системе этот параметр несложно «пробросить» на все блоки. Как это выглядит для пользователя: пользователь открывает в браузере нужный отчет, вводит период (дата начала и конца), ждет некоторое время и получает результат. Как это выглядит для разработчика: разработчик собирает несколько блоков в отчет, указывает имена общих параметров в каждом блоке, указывает имена параметров в общем отчете и публикует отчет. Выглядит просто, но не во всех отчетных системах это сделано удобно. Мой недавний пример из Retail Rocket: для хранилища на ClickHouse вначале выбрали SuperSet. Столкнулись с огромным количеством неудобств в параметрах. В итоге перешли на Metabase, где подобные параметрические отчеты делаются намного проще. Обе системы полностью бесплатны, с открытым исходным кодом.

Толстый или тонкий клиент? Толстый клиент означает наличие специальной программы на компьютере для просмотра отчетов, тонкий — вся работа идет через браузер. Обычно предпочитают работать через тонкий клиент из-за низкого порога входа: нужно авторизоваться через браузер и начать пользоваться системой. В толстых клиентах намного больше возможностей, но на их обучение придется потратить больше времени. Толстые клиенты важны для работы с мобильных телефонов, они адаптируют интерфейсы, пусть и урезанные.

Администрирование пользователей удобно, когда есть единая система учета. В таком случае пользователям не нужно помнить множество паролей, а администраторам легко регулировать доступ. По своему опыту скажу, что если компания использует, например, G Suite для бизнеса от Google, то система отчетности, которая может использовать ту же самую авторизацию, будет удобнее,

чем не использующая. Например, тот же Metabase [47] позволяет авторизоваться через Google, а SuperSet [48] нет.

Рассылка отчетов бывает периодической, когда она происходит по часам или определенным дням, и триггерной — в ответ на появление какого-либо события или изменения показателя. Триггерные рассылки часто используются в ИТ, например, чтобы поймать момент, когда падает какая-либо система или критически поднимается нагрузка. Для этого на определенный показатель системы выставляется пороговое значение, при превышении которого высылается соответствующее письмо. В бизнесе сложнее — там показатели не так быстро меняются: в интернет-магазине, например, можно поставить пороговые значения на количество заказов за последний час или трафик, чтобы как можно быстрее узнать о проблеме и избежать большой потери выручки. Отчеты могут присылаться в теле письма, что удобнее (вы сразу увидите результат), в приложенных файлах, например в формате Excel, или краткий отчет в теле письма, а расширенный доступен по ссылке. Удобство отчетов по электронной почте зависит от задач: если нужно быстро взглянуть на графики в мобильном телефоне, то лучше, когда отчет в теле письма; если с цифрами нужно будет работать — файл с электронной таблицей будет идеальным вариантом.

Что будет, если отчет запустить несколько раз подряд? Например, несколько пользователей с разницей в одну минуту запросят один и тот же отчет. Ждать очередные пять минут, пока он считается? Это зависит от схемы кэширования — в хорошей системе она есть. При публикации отчета выставляется период кэширования или сохранения прошлых результатов. Например, если выставить период в 30 минут, то после расчета данные отчета будут сохранены для последующих запросов ровно на 30 минут. И все последующие отчеты будут уже использовать их. Это очень полезно для тяжелых вычислений, пусть при кэшировании данные в отчете могут отставать от хранилища. В Ozon.ru одно время в системе back-office был отчет с текущими результатами дня. Отчет очень

часто обновляли сотрудники из азарта. Это привело к DoS (Denial of Service — отказ в обслуживании) — атаке, которая ухудшила производительность. Кэширование отчета на определенное время остудило пыл азартных любителей цифр и разгрузило систему приема заказов.

Интерактивный анализ — это когда вы исследуете данные, проваливаясь вглубь цифр и метрик; он де-факто считается стандартом любой аналитической системы. Есть графический тип анализа, хороший пример — Google Analytics: практически все тут можно сделать мышью. Второй тип — сводные таблицы. Я больше склонен именно к такому типу анализа. Делаю выборку данных, копирую ее в любую электронную таблицу, включаю анализ сводных таблиц (pivot table), а далее уже в интерфейсе «кручу» данные. На самом деле почти всегда, когда мы работаем с интерактивным анализом данных, мы работаем со сводными таблицами.

Если вкратце, то мои минимальные требования к отчетной системе такие:

- авторизация пользователей, желательна завязанная на корпоративную систему доступа;
- тонкий клиент, доступ через веб-браузер;
- возможность просмотра отчета, полученного по электронной почте, сразу на экране;
- несложная параметризация большого отчета, состоящего из множества блоков;
- кэширование результатов.

## СВОДНЫЕ ТАБЛИЦЫ

Сводные таблицы (pivot tables) — это самое лучшее, что было изобретено в разведочном анализе данных. Если аналитик хорошо владеет сводными таблицами, он всегда заработает на хлеб

с маслом. Сводная таблица избавляет нас от огромного числа бесполезных запросов к данным, когда нужно просто найти хоть какую-то зацепку. Я уже писал выше про свой личный шаблон интерактивного анализа данных: сделать выборку данных, скопировать данные в электронные таблицы, построить сводную таблицу и работать с ней. Этот способ сэкономил мне годы по сравнению с прямыми методами — подсчетом описательных статистик, построением простых графиков, то есть стандартными операциями анализа данных для любых аналитических инструментов. А теперь разберем по пунктам, как работать со сводными таблицами.

Во-первых, нужно подготовить данные. Они должны выглядеть как таблица фактов (fact table), которая делается на основе таблиц состояния на определенный момент или лога изменений данных (вспоминаем главу про данные). Если в таблице используются непонятные обычному человеку идентификаторы и у вас есть справочники на них, то лучше расшифровать это поле, присоединив (join или merge) данные справочника к таблице фактов. Поясню на примере. Мы ищем причину падения продаж. Пусть у нас есть таблица состояния заказов на определенный момент, у нее есть следующие поля:

- Дата и время создания заказа (например, 10 ноября 2020 года 12:35:02).
- ID типа клиента, который совершил заказ (1, 2).
- ID статуса клиента в программе лояльности (1, 2, 3).
- ID заказа (2134, 2135, ...).
- ID клиента (1, 2, 3, 4, ...).
- Сумма заказа в рублях (102, 1012, ...).

Эта таблица будет таблицей фактов, так как в ней записаны факты появления заказов. Аналитик хочет увидеть, как заказывали клиенты разных типов и статусов в программе лояльности. У него есть

гипотеза, что там находится основная причина изменения продаж. ID-поля нечитаемы и созданы для нормализации таблиц в учетной базе данных, но у нас есть справочники (табл. 7.1–7.2), которые полностью расшифровывают их.

**Таблица 7.1.** Справочник типа клиента

ID	Name
1	Физическое лицо
2	Юридическое лицо

**Таблица 7.2.** Статусы клиента в программе лояльности

ID	Name
1	VIP-клиент
2	Есть карта лояльности
3	Нет карты лояльности

После соединения (`join` или `merge`) таблицы фактов со справочниками мы получим обновленную таблицу (табл. 7.3) фактов:

- `datetime` — дата и время создания заказа (например, 10 ноября 2020 года 12:35:02).
- `client_type` — тип клиента, который совершил заказ (физическое или юридическое лицо).
- `client_status` — статус клиента в программе лояльности (VIP, есть карта лояльности, нет карты лояльности).
- `order_id` — ID заказа (2134, 2135, ...).
- `client_id` — ID клиента (1, 2, ...).
- `amount` — сумма заказа в рублях (102, 1012, ...).

**Таблица 7.3.** Пример объединения данных

datetime	client_type	client_status	order_id	client_id	amount
10-22-2020 12:35:02	Физическое лицо	VIP	2134	1	102 р.
10-22-2020 12:35:02	Юридическое лицо	Нет карты лояльности	2135	2	1012 р.

Что в этой таблице фактов хорошо — нет id полей, кроме двух — заказов и клиентов, но это полезные поля, они, возможно, понадобятся, чтобы посмотреть более подробно какие-то заказы во внутренней учетной системе. Аналитик получил выборку данных в указанном выше виде, поместил ее в электронную таблицу, например Microsoft Excel или Google Sheets. Построил над этой таблицей сводную (pivot table). Приступим к ее анализу.

В сводных таблицах есть два типа данных: измерения (dimensions) и показатели (или меры, measures). Измерения представлены в формате системы координат. Когда я слышу слово «измерения», я представляю себе три оси координат, выходящие из одной точки перпендикулярно по отношению друг другу — как нас учили на уроках геометрии. Измерений (осей) может быть гораздо больше трех. Их можно будет использовать в виде столбцов, строк или фильтров сводной таблицы, но их нельзя помещать в ячейки. Примеры измерений:

- Дата и время.
- Тип клиента.
- Статус клиента.

Показатели — это уже статистики, которые будут рассчитываться в сводной таблице, когда вы будете «вращать» или менять измерения. Они, как правило, агрегатные: суммы, средние, количество уникальных значений (distinct count), количество непустых значений (count). Примеры показателей для нашей задачи:

- Сумма заказов.
- Средний чек заказа.
- Количество заказов (уникальность здесь обеспечена тем, что одна строка — это заказ, дублей заказов нет).
- Количество уникальных клиентов (нужно считать число уникальных ID, так как один клиент может сделать несколько заказов, и его посчитают несколько раз).

ID заказов и ID клиентов могут быть как измерениями — тогда вы сможете считать статистику по конкретным заказам или клиентам, так и показателями — тогда можно просто посчитать количество заказов или клиентов. Это целиком зависит от вашей задачи, оба способа работают.

Аналитик определяет для каждого столбца, являются ли данные в нем измерениями или показателями, а также какие статистики по показателям ему нужны. Подготовительные работы закончены, теперь время сформулировать гипотезы и для каждой из них определить один или несколько срезов, которые подтвердят гипотезу или опровергнут. Понятие среза происходит из многомерной природы сводных таблиц. Представьте себе трехмерный предмет, имеющий следующие измерения: длину, ширину и высоту. Пусть это будет кусок сливочного масла. Вы берете нож, разрезаете его и получаете срез, причем плоскость среза перпендикулярна оси, которую вы фиксируете. То же самое вы проделываете, когда работаете со сводной таблицей — делаете срез многомерных данных. Осей может быть много, это число равно числу измерений — вот откуда берется многомерность. Место на оси (измерение), перпендикулярно которой режете, попадет в фильтр отчета как значение. Вы фиксируете его. Измерения, которые будут лежать в плоскости среза, будут столбцами и строками нашей таблицы. Если фильтр отчета не используется, то все данные будут спроецированы на наш срез при помощи операции агрегации, которая для каждого показателя выбирается индивидуально (суммы, средние, количество).

Аналитик формулирует две гипотезы относительно падения продаж:

- Изменение поведения вызвано одним из типов клиента. Для этой гипотезы одно из измерений — тип клиента.
- Изменение поведения вызвано одной из групп лояльности. Для этой гипотезы одно из измерений — статус лояльности клиента.

Так как у нас произошли изменения во времени, то нам понадобится еще одно измерение — время. Итак, гипотеза и нужный срез данных сформулированы, а дальше дело техники: мышью перетащить нужные измерения, например, дату в столбцы, тип клиента в строки. Заполнить таблицу нужными показателями и проверить, подтверждается ли проверяемая гипотеза цифрами или нет. Правильность гипотезы желательно проверить подходящим статистическим критерием для гипотез, что в реальности делается довольно редко.

Гипотезы можно формулировать и проверять последовательно, а когда наработается опыт, то они будут формулироваться на уровне подсознания. Аналитик будет играть ими, чтобы найти самую вероятную причину проблемы или успеха: делать первый срез, а потом добавлять измерения, пересекая их со старыми, и изменять показатели.

Если бы не было электронных таблиц и средств визуального анализа на сводных таблицах, то скорость подобного типа анализа была бы в десятки раз ниже. Аналитику пришлось бы программировать каждый срез, например, через оператор `GROUP BY` в SQL или `pivot` в питоновской библиотеке `pandas`. Со сводными таблицами аналитик работает со скоростью своей мысли.

## OLAP-КУБЫ

Сводные таблицы бывают не только в электронных таблицах. Большие объемы данных туда не поместить — они будут очень медленно работать, если вообще туда поместятся. А мы ведь хотим,

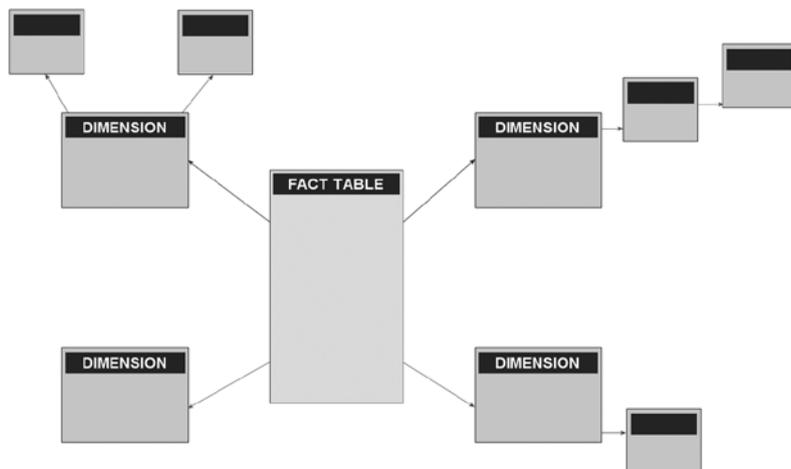
чтобы все работало со скоростью мысли, не правда ли? Для этого производители софта идут на всякие ухищрения, например, размещают данные в колоночной базе данных прямо на компьютере пользователя (о преимуществах колоночных баз данных уже написано в главе про хранилища). Второй способ — делать все вычисления на серверах, а пользователю предоставить туда доступ через интерфейс (толстый или тонкий клиент). Именно так были придуманы кубы OLAP (On-Line Analytical Processing — интерактивный анализ данных).

История их появления очень интересна как минимум тем, что к этому приложил руку наш бывший соотечественник — Михаил (Моша) Пасуманский. Михаил переехал в Израиль из Санкт-Петербурга в 1990 году. Там он написал аналитическое приложение «Панорама». В 1995 году они выпустили первую версию. В 1996 году компанию купила Microsoft, которой нужно было подобное решение для новой версии SQL Server. После интеграции системы в софт Microsoft появился язык программирования для работы с OLAP-кубами, который называется MDX (Multidimensional Expressions), чьим автором является Михаил Пасуманский. Этот язык является стандартом для работы с OLAP-кубами, и его поддерживают очень многие вендоры. Сервис OLAP-кубов теперь называется Analysis Services.

Мы уже рассмотрели, как работают сводные таблицы. Теперь посмотрим, как проблема производительности решается в OLAP-кубах, которые эти сводные таблицы умеют очень быстро рассчитывать. Я много работал с технологиями Microsoft по OLAP-кубам, поэтому буду опираться на свой опыт. Центральным звеном любого OLAP-куба является таблица фактов, которую мы рассмотрели на примерах построения сводных таблиц чуть ранее. Однако есть небольшое, но важное отличие: таблица фактов, как правило, не соединяется со справочниками, она загружается в кубы отдельно от них.

Для этого в хранилище данные готовятся по схеме «звезда» (рис. 7.4): таблица фактов соединяется по полям, содержащим ID

(ключи), со справочниками, как показано на рисунке. Существует правило — все измерения лучше держать в отдельных справочниках. Это сделано для того, чтобы можно было их обновлять независимо от таблицы фактов. После подготовки нужных данных в программе-дизайнере нужно отметить, какие таблицы являются таблицами измерений, а какие — таблицами фактов. Там же в настройках указывается, какие показатели необходимо рассчитать. Первичная обработка куба заключается в чтении всех данных из хранилища и помещении их в специальные структуры, которые очень быстро работают с расчетом сводных таблиц. Сначала читаются и обрабатываются все измерения, и только после этого таблица фактов. Но самое интересное происходит потом, когда нужно добавить в куб новые данные.



**Рис. 7.4.** Соединение таблиц по схеме «звезда»

Но что делать, если появились новые данные? Нужно обновить данные в нашей «звезде» — хранилище — и обновить куб на их основе. Обновление данных для уже обработанного рабочего куба заключается в полном прочтении всех измерений (справочников): если появляются новые элементы или они переиме-

новываются — все обновится. Поэтому справочники измерений для OLAP-кубов нужно сохранять и обновлять в первоизданном виде — данные оттуда удалять нельзя. Обновление таблицы фактов интереснее — можно выбрать полное обновление и пересчет куба, а можно стереть старые данные из таблицы фактов и залить туда новые, которых нет в кубе, и только после этого обновить куб. Схема с инкрементальным обновлением выгоднее с точки зрения времени обработки куба. В Ozon.ru полный процессинг куба мог занимать у меня четыре дня, а инкрементальное обновление всего двадцать минут.

Существует несколько популярных вариантов хранения и обработки данных в OLAP-кубах:

- MOLAP — та схема хранения, которую я описал выше. Данные хранятся в специальных структурах, которые очень быстро вычисляют сводные таблицы.
- ROLAP — данные никуда не помещаются, они находятся в хранилище. OLAP-куб транслирует запросы из сводных таблиц в запросы к хранилищу и отдает результат.
- HOLAP — данные частично находятся в MOLAP-, частично в ROLAP-схемах. Например, это может быть полезно для уменьшения времени отставания куба от реального времени. Куб обновляется раз в день по схеме MOLAP, а новые данные, которых там пока нет, существуют в схеме ROLAP.

Я всегда предпочитал пользоваться MOLAP-схемой, как наиболее быстрой. Хотя в связи с развитием быстрых колоночных баз данных ROLAP могут оказаться проще. Ведь ROLAP-схема не требует тщательного дизайна куба, как MOLAP, что сильно упрощает техническую поддержку OLAP-куба.

Самый идеальный клиент для OLAP-кубов — это электронные таблицы Microsoft Excel, где работа с кубами от Microsoft реализована очень удобно. Любые тонкие клиенты не обеспечивают того удобства и гибкости, которые предлагает Excel: возможность использования MDX, гибких формул, построения отчетов из отчетов.

К сожалению, эта функциональность поддерживается только в операционной системе Windows. Версия Excel для OS X (Apple) не поддерживает ее. Вы даже не представляете себе, сколько руководителей и специалистов шлют проклятья разработчикам MS за это. Ведь им приходится держать отдельные ноутбуки с Windows или удаленные машины в облаке только для того, чтобы работать с кубами Microsoft. Я считаю, что это самая большая ошибка Microsoft в нашей области.

## КОРПОРАТИВНЫЕ И ПЕРСОНАЛЬНЫЕ BI-СИСТЕМЫ

В построении аналитики в компании есть два подхода — корпоративный и персональный (self-service BI). О персональных средствах мы уже поговорили. Теперь посмотрим на системы корпоративного уровня. Обычно в корпоративную систему принятия решений входит ряд компонент, о которых уже написано выше:

- хранилище данных;
- система обновления данных в хранилище (ETL);
- система интерактивного анализа и отчетности.

Примером такой системы может быть Microsoft SQL Server: хранилищем выступает сам SQL-сервер, Integration Services — это ETL, Analysis Services — OLAP-кубы для интерактивных и статичных отчетов, Reporting Services — система отчетности через тонкий клиент. Подобная схема продуктов есть у многих вендоров BI-систем, в том числе у облачных.

С моей точки зрения, главное отличие корпоративных систем от персональных — их разработка осуществляется централизованно. Например, если нужно внести изменения в отчет или другую компоненту, то самостоятельно это не сделать и придется обратиться к разработчику. Часто доходит до крайности, когда

становится невозможным что-то изменить в системе без участия разработчика.

Полностью персональные системы автономны: данные берутся напрямую из источников данных и складываются в виде локальных файлов на компьютере аналитика. Им не нужен разработчик — аналитик может сделать все самостоятельно. Это другая крайность. Давайте сравним эти два подхода.

- *Скорость изменений* — персональные системы выигрывают у корпоративных. Любое изменение в корпоративной системе даст изменения у всех конечных пользователей. Такие изменения необходимо согласовывать и ждать, пока их внесут. В персональных системах пользователь просто вносит нужные ему изменения.
- *Качество данных* выше у корпоративных систем. Тут их консервативность играет нам на руку: практики разработки и дизайна системы позволяют следить за качеством данных. У персональных систем с этим плохо — они очень зависимы от навыков и внимательности пользователя, что повышает вероятность ошибок.
- *Производительность* — корпоративные системы сразу проектируют под нужную нагрузку, а если они не справляются, то есть технологии их масштабирования. С персональными системами вероятность столкнуться с плохой производительностью намного выше.
- *Смена вендора решения*: корпоративные системы — это деревья с сильными корнями, которые потом будет очень тяжело выкорчевать. Персональные намного проще.

С моей точки зрения, оба варианта имеют право на жизнь. Лучшие результаты можно получить, объединив два подхода. Ничто не мешает вам построить хранилище (или озеро данных), а затем использовать гибкие персональные инструменты для работы в «последней миле». Часть вопросов снимается, вероятность ошибки пользователя остается, но она становится меньше.

## МОЙ ОПЫТ

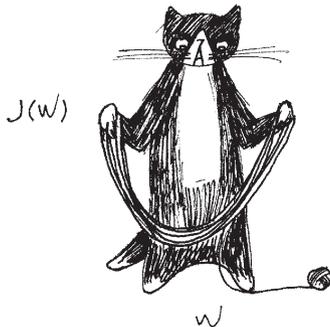
Я работал с разными системами: от файлов электронных таблиц до распределенных систем Hadoop/Spark с петабайтами данных. Могу только сказать: всему свое время. Архитектура аналитических систем — это искусство, требующее недюжинных талантов. В этой главе я рассказал о своих наработках, которые были проверены на практике. У любых решений есть плюсы и минусы, и знать их лучше до создания системы, а не после.

Вы всегда будете балансировать между скоростью и качеством, между стоимостью сейчас и потом. Но всегда полезно думать о пользователе. Чем быстрее и интуитивнее он сможет работать с данными, тем меньше эта работа будет вызывать отторжение и тем эффективнее он будет принимать решения. Я считаю, что именно скорость, даже при средних гарантиях качества, обеспечивает успех компаний. Выиграет та, которая приняла больше решений.

Аналитические системы позволяют ориентироваться в данных, но они не принимают решения — это делают пользователи. Давайте сделаем их жизнь лучше, упростив для них путь принятия решений.

# 8

## АЛГОРИТМЫ МАШИННОГО ОБУЧЕНИЯ

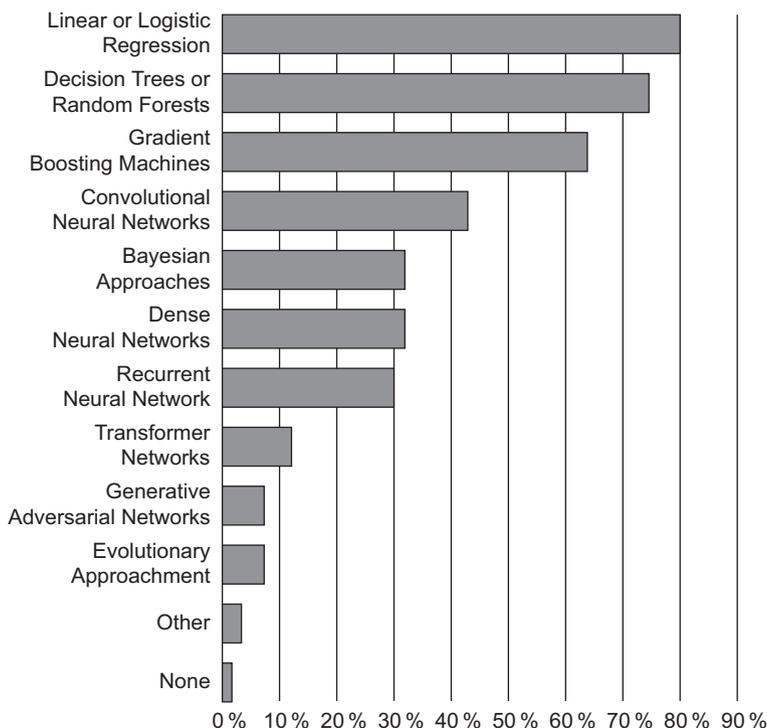


Как я уже писал, основное отличие машинного обучения от обычного программирования заключается в том, что программа обучается на примерах, а не на прямых инструкциях. Поэтому неотъемлемой частью какого-либо решения является обученная на данных (примерах) модель. Выглядит это как черный ящик: программный код, который представляет собой интерфейс взаимодействия обученной модели или черного ящика с внешним миром. В самом ящике находится набор коэффициентов и параметров модели, которые понятны только этому коду.

Польза ML заключается в автоматизации принятия решений, за счет этого данные обрабатываются гораздо быстрее. Мы, люди, обладаем настоящим универсальным интеллектом, но из-за этой универсальности количество задач, которые мы можем решить за единицу времени, очень ограничено. Чего не скажешь о ML, типовой алгоритм которого заточен на очень узкую задачу, но его можно масштабировать неограниченно. Хотите один миллион прогнозов в секунду — пожалуйста, предоставьте только необходимые ресурсы.

Моя книга — не учебник по ML. В этой главе я проговорю только основы, которые нам понадобятся в дальнейшем. Они покрывают 80 % того, что понадобится на практике работы со структурированными данными — правило Парето работает и здесь. Сами алгоритмы я выбрал из своей практики применения: линейная регрессия, логистическая регрессия, деревья решений и ансамбли (random forest, xgboost, catboost). Это самые популярные алгоритмы решения задач для структурированных данных. Что подтверждается исследованием, которое провел Kaggle в 2019 году [53]. На 18-й странице этого исследования есть график популярности методов машинного обучения (рис. 8.1). И в исследовании написан

вывод: «Участники опроса — фанаты простоты. Самые популярные методы — линейная и логистическая регрессии, за которыми идут деревья решений. Они не настолько мощные, как более сложные методы, но достаточно эффективны и их легко интерпретировать».



**Рис. 8.1.** Простые линейные методы — самые популярные

С нейронными сетями я впервые познакомился в 2002 году, когда стал работать в компании StatSoft. Тогда они не произвели на меня впечатления. Да, алгоритм красивый и интересный, но те задачи, которые ими решались, можно было решить более простыми методами. Золотой век нейронных сетей начался примерно 10 лет назад, когда стали применять сверточные нейронные сети (convolutional

neural networks). Они стали идеальным инструментом для работы с изображениями, звуком и прочей неструктурированной информацией. Есть попытки их использовать и в рекомендательных сервисах, но пока нет существенного прогресса [52]. Мы в Retail Rocket тоже пытались их использовать. В рекомендациях одежды существенной выгоды [25] не получили. А вот для удаления «плохих» рекомендаций они пригодились и используются прямо сейчас. Сама область довольно молодая, вычислительные чипы GPU, которые предназначены для ускорения вычислений на нейронных сетях глубокого обучения, становятся все мощнее и дешевле. Я ожидаю более широкого применения этих методов в течение ближайших 10 лет. Теоретически они будут способны заменить алгоритмы на универсальные даже для структурированных данных, что сильно упростит решение ML-задач. Сами нейронные сети — тема для отдельной книги.

## ТИПЫ ML-ЗАДАЧ

Классическое ML делится на три типа:

- обучение с учителем (supervised learning);
- обучение без учителя (unsupervised learning);
- обучение с подкреплением (reinforcement learning).

Обучение с учителем подразумевает под собой, что для каждого набора входных данных (независимые переменные или фичи) у вас есть величина (зависимая переменная), которую модель должна предсказать. Примеры таких задач представлены в табл. 8.1 [50].

Таблица 8.1. Типы задач в ML

Входные данные (независимые)	Что нужно предсказать (зависимая переменная)	Где применяется	Класс задачи
Фотография	Есть на фото лица людей? (0 или 1)	Для тегирования персон на фото	Классификация
Заявка на получение кредита	Заявитель выплатит кредит? (0 или 1)	Одобрение выдачи кредита в банке	Классификация
Аудиозапись	Текст из аудиозаписи	Распознавание речи	
Предложение на английском языке	Предложение на русском	Перевод с языка	
Технологические сенсоры в промышленном оборудовании	Когда оно сломается? (вероятность)	Контроль промышленного оборудования	Регрессия
Данные с камер автомобиля и других сенсоров	Позиция относительно других автомобилей	Самоуправляемые автомобили	

В таком типе задач есть два основных класса задач: задача регрессии, когда нужно прогнозировать показатель с непрерывной шкалой (например, деньги, вероятность); а также задача классификации, когда нужно понять, к какому классу принадлежит объект (есть ли на фотографии люди, человек болен или нет, кто изображен на фотографии). Есть еще ряд задач, связанных с переводом текста, распознаванием речи, геопозиционированием, которые получили большое распространение благодаря глубоким нейронным сетям.

Понятие «регрессия» возникло, когда сэр Френсис Гамильтон познакомился с книгой Дарвина «Происхождение видов». Он решил изучить, как рост детей зависит от роста родителей. В процессе исследования он выяснил, что дети очень высоких и очень низких родителей в среднем имеют менее высокий и, соответственно, менее низкий рост. Это движение роста назад в направлении к среднему Ф. Гамильтон назвал регрессией (to regress — двигаться в обратном направлении) [51]. В 1885 году он издал работу «Регрессия в направлении к общему среднему размеру при наследовании роста», через некоторое время это понятие стало применяться ко всем задачам с односторонней стохастической зависимостью.

Обучение без учителя (unsupervised learning) подразумевает, что в данных есть закономерность, но вы не знаете какая (нет зависимой переменной). Это может быть задача разделения датасета на кластеры (кластерный анализ), поиск аномалий, автоэнкодеры (например, для уменьшения размерности пространства фич), метод главных компонент (Principal Component Analysis), коллаборативная фильтрация (рекомендательные системы).

Обучение с подкреплением (reinforcement learning) — модель учится через взаимодействие со средой. Это частный случай модели с учителем, где учителем является не датасет, а реакция среды на какое-то действие, которое мы произвели. Часто применяется при разработке ботов для игр (не только стрелялки, но и шахматы), управлении роботами. В отличие от классического машинного обучения, датасета здесь нет. Агент (например, робот) производит какое-либо действие в среде, получает обратную связь, которая транслируется в награду (reward). Агент учится совершать такие действия, которые максимизируют его награду. Так, например, можно научить робота ходить или играть в шахматы.

Для каждого типа задач классического ML есть соответствующий алгоритм. Например шпаргалка (cheat sheet, рис. 8.2) для очень популярной библиотеки scikit learn выглядит так:

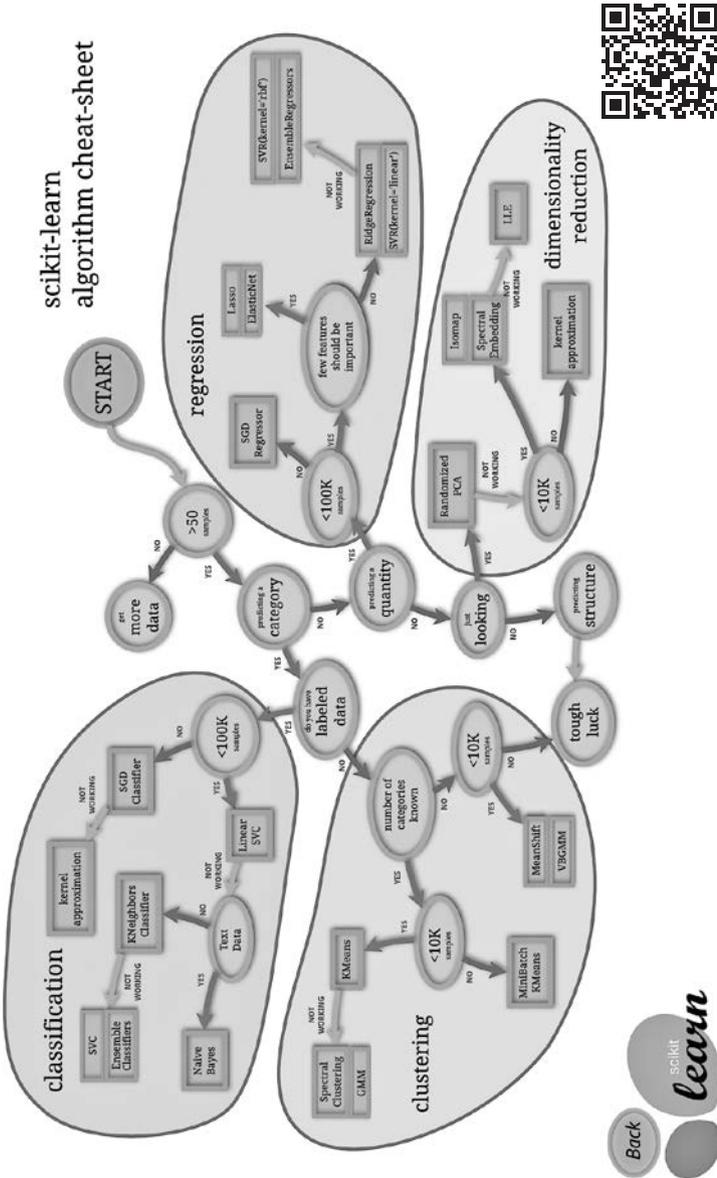


Рис. 8.2. Шапка методов ML ([https://scikit-learn.org/stable/tutorial/machine\\_learning\\_map/index.html](https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html))

В зависимости от задачи и данных, предлагается вариант модели, которая, скорее всего, будет успешной. Именно с этой библиотеки я начал свое изучение ML-моделей в Python, когда участвовал в соревнованиях на Kaggle.

## МЕТРИКИ ML-ЗАДАЧ

Если в данных есть неслучайная закономерность, выбрана соответствующая задаче модель и есть достаточный набор данных, то обучение ML-модели не выглядит чем-то сложным. Самый первый шаг — понять, как измерить эффективность модели. Это очень важно: если мы делаем модель, которая будет помогать в реальной жизни, у нее должно быть четкое определение цели. А цель уже выражается метрикой. Например, если нужно, чтобы она прогнозировала спрос на товары, то цель — уменьшить процент расхождения с реальностью (ошибка прогноза). Если по фотографии нужно определить, есть ли там изображение собаки, то цель — увеличить процент правильно угаданных фотографий. В любых задачах на Kaggle есть метрика, по которой выбирают победителя.

Этот шаг новички — и не только они — часто игнорируют или выполняют невнимательно. По опыту работы в Retail Rocket скажу, что ошибка в выборе правильной метрики может стоить очень дорого. Это фундамент всего ML. Сделать это не так просто, как кажется на первый взгляд. К примеру, есть несколько систем тестирования COVID-19 через мазок из горла. Одна из них дает больше ложноположительных результатов (больной на самом деле не болен), вторая — больше ложноотрицательных (пропускает больных, которые заразы). Какую систему выбрать? Если первую, то вы запрете дома много здоровых людей, это повлияет на экономику. Если вторую — то пропущенные больные будут распространять вирус. Такой выбор — это баланс плюсов и минусов. Аналогичные проблемы есть в разработке систем рекомендаций: пусть первый алгоритм выдает более логичные с точки зрения пользователя рекомендации, а второй дает магазину больший доход. (Это оз-

начает, что машина обучалась на слабых, но важных сигналах, которые кажутся нам нелогичными, и ее рекомендации приводят к росту продаж, хотя это и невозможно объяснить логикой в силу ограниченности человеческого ума.) Какой алгоритм выбрать? С точки зрения бизнеса — второй, но когда такую систему продают и показывают менеджерам, принимающим решение о покупке, им часто нравится первый вариант. Их можно понять, ведь сайт — это витрина, и она должна выглядеть привлекательно. В итоге приходится трансформировать работающий первый алгоритм, стараясь зафиксировать высокие финансовые показатели, но сделать так, чтобы рекомендации выглядели более логично. Фокус на нескольких метриках сильно усложняет разработку новых алгоритмов.

Любая метрика считается как разность между тем, что прогнозирует модель по входным данным (независимые переменные, или фичи), и тем, что есть на самом деле (зависимая переменная, или outcome). Существует много нюансов расчета метрик, но суть всегда именно в этой разнице. Типовые метрики ML-задач зависят от их класса.

Для регрессии — это среднеквадратичная ошибка (Mean Squared Error, MSE). Считается как сумма квадратов разностей прогнозируемого и действительного значений из датасета, деленное на число примеров в датасете:

$$MSE = \frac{1}{n} \sum (y - \hat{y})^2.$$

Есть еще ряд популярных метрик, таких как RMSE, MAE, R<sup>2</sup>.

Для задач классификации самые популярные метрики можно легко получить из матрицы ошибок классификации (misclassification или confusion matrix). Когда я собеседую кандидата на должность аналитика, то часто прошу нарисовать эту матрицу (табл. 8.2) и вывести из нее метрики.

Представьте, что вам нужно решить задачу классификации — определить, простужен человек или нет. У вас есть датасет со следующими фичами: температура тела, болит ли горло, есть ли насморк, чихание, светобоязнь. Для каждого примера у вас есть идентифи-

катор: 1, или True, — человек болен простудой, 0, или False, — нет. Вы строите модель, которая для каждого примера дает результат: 1 для больного, 0 для здорового. Чтобы понять ошибку вашей модели, нужно сравнить ее вывод и правильные значения. Допустим, мы это сделали и теперь можем составить такую матрицу ошибок классификации. В строках мы отметим предсказанные значения, в столбцах — действительные. В ячейках таблицы можем написать количество случаев для каждого класса (0 и 1) — когда прогноз совпал, а когда нет. Я заполнил таблицу, к примеру, 100 примеров, когда мы верно угадали единицу.

**Таблица 8.2.** Матрица ошибок классификации

	Действительно: 1 (True)	Действительно: 0 (False)
Предсказано: 1 (True)	100 (TP)	10 (FP)
Предсказано: 0 (False)	15 (FN)	90 (TN)

В скобках написаны обозначения, которые нам пригодятся для вывода метрик: TP = True Positive (правильно угаданные 1), TN = True Negative (правильно угаданные 0), FN = False Negative (ложно негативные, модель посчитала 0, а на самом деле 1), FP = False Positive (модель посчитала 1, а на деле 0). Давайте выведем метрики:

Accuracy (Точность) = точно угаданные / число примеров =  
 $= (TP + TN) / (TP + TN + FP + FN)$ .

Precision (Точность для 1) =  $TP / (TP + FP)$ .

Recall (Полнота) =  $TP / (TP + FN)$  (или сколько процентов 1 мы нашли правильно).

F мера =  $1 / (1/Precision + 1/Recall)$ .

Самые часто применяющиеся метрики для классификации — precision и recall. Они не зависят от несбалансированности классов, как accuracy. Эта ситуация возникает, когда соотношение единиц и нулей в датасете далеко от 50/50.

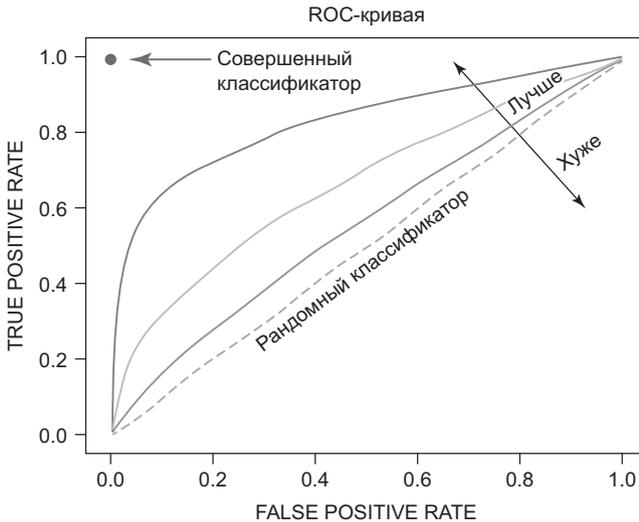
В метриках классификации есть еще одна интересная — AUC-ROC (Area Under Curve of Receiver Operating Characteristic). Она нужна тогда, когда алгоритм классификации выдает вероятность принадлежности к какому-либо классу — 0 или 1. Чтобы посчитать метрики Recall и Precision, нам придется делать разные пороговые значения для вероятностей (чтобы различать классы) и считать их. Как раз для этого и нужна AUC-ROC, которая хорошо показывает эффективность классификатора независимо от порогового значения. Чтобы ее построить, необходимо взять набор пороговых значений из отрезка  $[0;1]$  и для каждого значения порога вычислить два числа:

$$\text{TPR (True Positive Rate)} = \text{TP}/(\text{TP} + \text{FN}) \text{ и}$$

$$\text{FPR (False Positive Rate)} = \text{FP}/(\text{FP} + \text{TN}).$$

Отметить эти числа как точки в плоскости координат TPR и FPR и получить следующую кривую (рис. 8.3).

Площадь под ней и есть AUC. В случае если вы получаете AUC близкий к 0.5, ваш классификатор почти ничем не отличается от



**Рис. 8.3.** ROC-кривая

случайного подбрасывания монетки. Чем его значение ближе к 1, тем лучше. Этот показатель часто используется в научной литературе и соревнованиях Kaggle.

Для других задач, например для ранжирования результатов поиска и для рекомендаций, используются свои показатели качества работы, о них можно узнать в специальной литературе.

Итак, у нас есть метрика. Теперь с ее помощью мы сможем сравнивать разные модели друг с другом и понимать, какая из них лучше. Можно приступать к обучению.

## ML ИЗНУТРИ

Практически любая ML-модель для обучения с учителем сводится к двум вещам: определение функции потерь (loss function для одного примера, cost function для множества примеров) и процедуры ее минимизации.

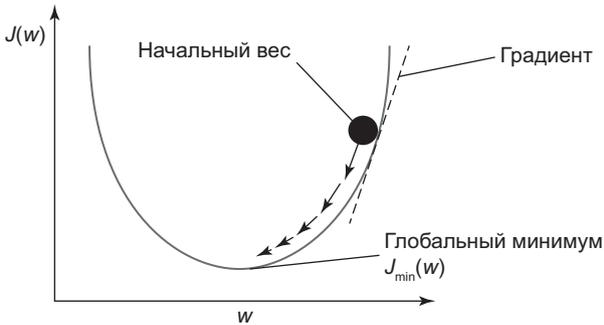
Например, для линейной регрессии это будет среднеквадратичная ошибка в том виде, в каком мы определили ее ранее. Чтобы найти минимум функции потерь, существуют различные процедуры оптимизации. Одна из них называется градиентным спуском (Gradient Descent), она широко применяется на практике.

Как правило, оптимизация выглядит следующим образом:

1. Коэффициенты (которые нужно подобрать) модели инициализируются нулями или случайно.
2. Вычисляется величина функции потерь (например, среднеквадратичное отклонение) и ее градиент (производная от функции потерь). Градиент нам нужен, чтобы понять, куда двигаться для минимизации ошибки.
3. Если функция потерь изменилась существенно и мы не достигли максимального числа повторений расчета, то пересчитаем коэффициенты, исходя из градиента, и идем к шагу 2.

4. Считаем, что оптимизация завершена, возвращаем модель с вычисленными коэффициентами.

Графически градиентный спуск выглядит как на рис. 8.4.



**Рис. 8.4.** Градиентный спуск

У нас есть функция потерь, и с произвольной точки мы двигаемся в сторону ее минимума последовательно, по шагам.

У этого алгоритма есть еще две версии: стохастический градиентный спуск (SGD) и пакетный (Mini Batch Gradient Descent). Первый используется для работы с большими данными, когда мы из всего датасета используем только один пример для одной итерации обучения. Альтернативной для больших данных является пакетная версия (batch) этого алгоритма, которая вместо одного примера использует подмножество датасета.

## ЛИНЕЙНАЯ РЕГРЕССИЯ

Самая простая и популярная модель регрессии. На самом деле ее мы затрагивали в школе, когда писали формулу линейной зависимости. Когда я учился в старших классах, она выглядела так:  $y = k \times x + b$ . Это так называемая простая линейная регрессия, в ней всего одна независимая переменная.

Обычно работают с множественной линейной регрессией (multivariate linear regression), формула которой выглядит так:

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip}, \quad i = 1, \dots, n.$$

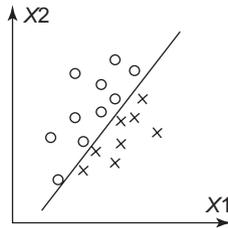
Она состоит из суммы произведений коэффициентов на значение соответствующей фичи и дополнительно свободного члена (intercept). Выглядит она как прямая в случае одной независимой переменной и как гиперплоскость в случае  $N$  фич. Когда происходит обучение линейной регрессии, то гиперплоскость строится таким образом, чтобы минимизировать расстояние от точек (из датасета) до нее, что является среднеквадратичным отклонением. Самый первый вопрос, который я задаю кандидатам на должность аналитика данных, звучит так: «У вас есть результат эксперимента, точки отмечены на плоскости с двумя осями. Кто-то провел линию, их аппроксимирующую. Как понять, оптимально ли построена прямая?» Это очень хороший вопрос на понимание сути линейной регрессии.

Если данные на входе линейной регрессии были нормализованы, то чем больше коэффициент у фичи, тем большее влияние на зависимую переменную она оказывает, а значит, и на результат. Положительный коэффициент — увеличение значения фичи увеличивает значение зависимой переменной (положительная корреляция). Отрицательная — это отрицательная корреляция или отрицательная линейная зависимость.

## ЛОГИСТИЧЕСКАЯ РЕГРЕССИЯ

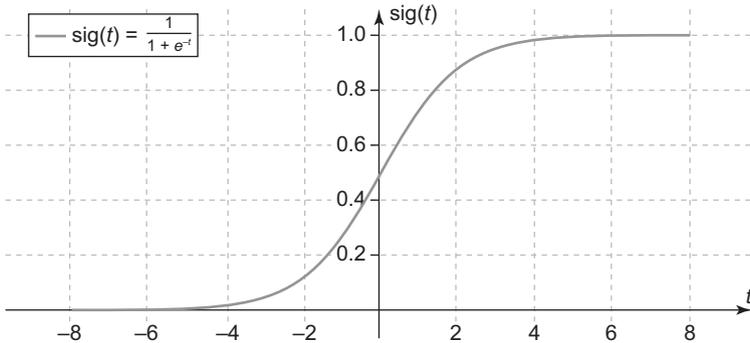
Это самая популярная модель решения задач бинарной (два класса) классификации.

Допустим, у нас есть задача — разделить два класса: крестики и нолики. Я их отметил на координатной сетке, по осям отложил значения фич  $X_1$  и  $X_2$  (рис. 8.5). Легко видеть, что между крестиками и ноликами можно провести прямую, которая их разделяет. Все, что выше прямой, — нолики, ниже — крестики. Так работает



**Рис. 8.5.** Разделяющая прямая в задаче классификации

логистическая регрессия — она ищет прямую или гиперплоскость, которая разделяет классы с минимальной ошибкой. Как результат она выдает вероятность принадлежности точки к классу. Чем ближе точка находится к разделяющей поверхности, тем менее модель уверена в своем выборе, вероятность будет приближаться к 0.5, чем дальше точка от поверхности — тем вероятность ближе к 0 или 1, в зависимости от класса. В задаче два класса, поэтому если вероятность принадлежности к одному классу равна 0.3, то ко второму  $1 - 0.3 = 0.7$ . Для вычисления вероятности в логистической регрессии используется сигмоида (рис. 8.6).



**Рис. 8.6.** Сигмоида

В этом графике в  $t$  подставляется значение из обычной линейной формулы с коэффициентами, как у линейной регрессии. Сама

формула является уравнением той разделяющей поверхности, о которой я писал выше.

По популярности это топовая модель как среди исследователей, которые любят ее за простоту и интерпретируемость (коэффициенты такие же, как у линейной регрессии), так и среди инженеров. На очень больших нагрузках, в отличие от других классификаторов, эта простая формула легко масштабируется. И когда вас догоняет в интернете баннерная реклама, скорее всего, за ней стоит логистическая регрессия, которая до недавнего времени использовалась, например, в компании Criteo, одной из самых больших ретаргетинговых компаний в мире [54].

## ДЕРЕВЬЯ РЕШЕНИЙ

Деревья решения (decision tree) дышат в спину линейным методам по популярности. Это очень наглядный метод (рис. 8.8), который может использоваться для задач классификации и регрессии. Самые лучшие алгоритмы классификации (Catboost, XGboost, Random Forest) основываются на нем. Сам метод нелинейный и представляет собой правила «если..., то...». Само дерево состоит из внутренних узлов и листьев. Внутренние узлы — это условия на независимые переменные (правила). Листья — это уже ответ, в котором содержится вероятность принадлежности к тому или иному классу. Чтобы получить ответ, нужно идти от корня дерева, отвечая на вопросы. Цель — добраться до листа и определить нужный класс.

Дерево строится совсем по иным принципам, чем те, которые мы рассмотрели в линейных методах. Мои дети играют в игру «вопрос-ответ». Один человек загадывает слово, а другие игроки

должны с помощью вопросов выяснить его. Допустимые ответы на вопрос только да/нет. Выиграет тот, кто меньшим числом вопросов угадает ответ. С деревом аналогично — начиная от корня дерева, правила строятся таким образом, чтобы за меньшее число шагов дойти до листа.

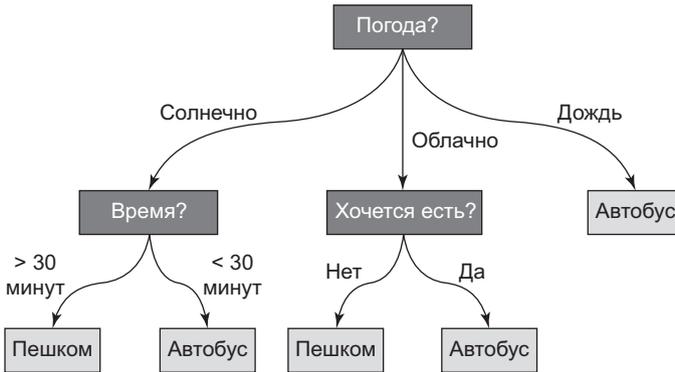


Рис. 8.7. Дерево решений

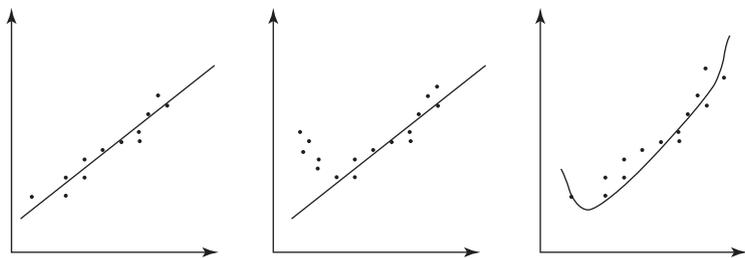
Для этого вначале выбирается фича. Разделив датасет по ее значению (для непрерывных подбираются пороги), мы получаем наибольшее уменьшение энтропии Шеннона (или наибольший информационный выигрыш). Для этого на каждом шаге происходит полный перебор всех фич и их значений. Этот процесс повторяется много раз, пока мы не достигнем ситуации, когда уже делить нечего, в выборке данных остались только наблюдения одного класса — это и будет листом. Часто это грозит переобучением — полученное дерево слишком сильно подстроилось под выборку, запомнив все данные в листьях. На практике при построении деревьев решений у них ограничивают глубину и максимальное число элементов в листьях. А если ничего не помогает, то проводят «обрезку» дерева (pruning или postpruning). Обрезка идет от листьев к корню. Решение принимается на основе проверки: насколько ухудшится качество дерева, если объединить эти два листа. Для этого используется отдельный небольшой датасет, который не участвовал в обучении [55].

## ОШИБКИ ОБУЧЕНИЯ

Модель в процессе обучения, если она правильно выбрана, пытается найти закономерности (patterns) и обобщить (generalize) их. Показатели эффективности позволяют сравнивать разные модели или подходы к их обучению путем простого сравнения. Согласитесь, что если у вас будут две модели, ошибка прогнозирования первой равна 15 %, а второй 10 %, то сразу понятно, что следует предпочесть вторую модель. А что будет, если при тестировании в модель попадут данные, которых не было в обучающем датасете? Если при обучении мы получили хорошее качество обобщения модели, то все будет в порядке, ошибка будет небольшой, а если нет, то ошибка может быть очень большой.

Итогом обучения модели могут быть два типа ошибок:

- модель не заметила закономерности (high bias, underfitting — недообучена);
- модель сделала слишком сложную интерпретацию, например, там, где мы видим линейную зависимость, модель увидела квадратичную (high variance, overfitting — переобучена).



**Рис. 8.8.** Правильное обучение, недообучение, переобучение

Попробую это продемонстрировать. На картинке (рис. 8.8) изображены результаты экспериментов в виде точек (вспомните лабы по физике в школе). Мы должны найти закономерности — построить линии, их описывающие. На первой картинке все

хорошо: прямая линия хорошо описывает данные, расстояния от точек до самой линии небольшие. Модель правильно определила закономерность. На второй — явно у нас зависимость нелинейная, например квадратичная. Значит, линия, проведенная по точкам, неправильная. Мы получили недообученную модель (underfitting), ошиблись порядком функции. На третьей картинке ситуация наоборот, модель выбрана слишком сложной для линейной зависимости, которая наблюдается по точкам. Выбросы данных исказили ее. Здесь налицо переобучение, нужно было выбрать модель попроще — линейную.

Я нарисовал относительно искусственную ситуацию — одна независимая переменная на горизонтальной оси и одна зависимая переменная на вертикальной оси. В таких простых условиях мы можем прямо на графике увидеть проблему. Но что будет, если у нас много независимых переменных, например десятков? Тут на помощь приходит подход для тестирования модели — валидация.

Она служит как раз для понимания таких ошибок, когда мы работаем с моделью как с черным ящиком. Самый простой подход — делим случайно датасет на две части: большую часть используем для обучения модели, меньшую — для ее тестирования. Обычно соотношение 80 к 20. Фокус здесь в том, что настоящая ошибка, когда модель выведем в бой, будет близка к ошибке, которую мы получим на тестовом датасете. Есть еще один вариант валидации, когда данные делятся не на две, а на три части: на первой части — обучается модель, на второй — происходит подбор гиперпараметров модели (настройки модели), на третьей уже получают тестовую оценку. Эндрю Бэн в своей книге «Machine learning Yearning» [60] считает эту модель валидации основной. Теперь обсудим сам алгоритм диаг-

ностики. Допустим, у нас есть две цифры — среднеквадратичные ошибки для обучающего датасета и тестового. Теперь сравним их:

- Тестовые и обучающие ошибки практически совпадают, сама ошибка минимальна и вас устраивает. Поздравляю, модель обучена правильно, ее можно выводить в бой.
- Тестовая ошибка существенно больше обучающей. При этом обучающая ошибка вас устраивает. Налицо переобучение — модель получилась слишком сложной для данных.
- Обучающая ошибка получилась высокой. Возникла ситуация недообучения. Либо выбранная модель слишком простая для этих данных, либо не хватает самих данных (объема или каких-то фич).

Более сложная версия валидации —  $k$ -fold cross validation ( $k$ -кратная перекрестная проверка). Ее активно применяют в серьезной работе, научных исследованиях и соревнованиях. Она заключается в случайном разделении датасета на  $k$  равных частей, например на 8 частей. Затем извлекаем первую часть из датасета, тренируем модель на оставшихся, считаем ошибки на обучающих данных и извлеченных данных (тестовая ошибка). Эту последовательность повторяем для всех частей. На выходе получаем  $k$  ошибок, которые можно усреднить. И делаем аналогичные сравнения, как описано выше.

## КАК БОРОТЬСЯ С ПЕРЕОБУЧЕНИЕМ

Для борьбы с переобучением есть несколько простых рецептов, которые применяются на практике. Во-первых, можно попытаться найти больше данных — привет, Капитан Очевидность! Это очень наивный совет, ведь обычно работают уже с максимально полным датасетом.

Второй способ — удалить выбросы в данных. Это можно сделать через анализ распределений: описательные статистики, гистограммы, графики «ящики и усы», диаграммы рассеяния будут полезны.

Третий вариант — удалить часть фич (независимых переменных). Это работает особенно хорошо для линейных методов, которые очень чувствительны к мультиколлинеарности фич. Мультиколлинеарность означает, что часть фич зависимы друг от друга. Природа этой зависимости может быть натуральной и искусственной. Естественная зависимость — число покупок и количество потраченных денег. Искусственная зависимость — когда аналитик добавил в датасет новые фичи как функцию от уже существующих. Например, возвел значение одной из них в квадрат, при этом старая фича осталась в датасете. В реальной работе эти ситуации встречаются сплошь и рядом.

Одним из негативных эффектов этого явления в линейных методах является резкое изменение коэффициентов, когда в модель добавляется новая, зависимая от уже включенных в нее фич. Например, аналитик использует линейную регрессию, чтобы понять, сделает ли покупатель еще одну покупку или нет. В модели у него уже была фича — число сделанных покупок, допустим, ее коэффициент равен 0.6. Следующим шагом он добавляет в модель объем средств, потраченных на совершенные покупки. Коэффициент этой фичи будет 0.5, при этом коэффициент числа покупок становится отрицательным:  $-0.1$ . Очень странная ситуация — понятно, что чем больше покупок клиент совершил в прошлом, тем больше вероятность, что он продолжит покупать. А тут мы видим, что число покупок влияет негативно. Это произошло из-за того, что корреляция (зависимость) между числом покупок и потраченными деньгами очень высокая и деньги оттянули на себя этот эффект. Сами коэффициенты могут быть важными, если вы пытаетесь понять причины какой-либо ситуации. С мультиколлинеарностью можно прийти к неверным выводам. Интересно, что у статистического анализа и машинного обучения разные цели. Для статистического анализа коэффициенты модели важны — они объясняют природу явления. Для машинного обучения важны не так, главное — достичь хорошей метрики, а как — не имеет значения.

Бороться с мультиколлинеарностью можно несколькими способами: удаление зависимых фич, сжатие пространства с помощью ана-

лиза главных компонент (Principal Component Analysis), гребневая регрессия (Ridge Regression). Для первого способа — используется пошаговое включение или пошаговое исключение фич. При пошаговом включении первым шагом выбирается фича, которая имеет наименьшую ошибку, если построить регрессию только на ней. Для этого нужно перебрать все фичи и выбрать только одну, вторым шагом выбрать следующую фичу и так далее. Остановиться нужно тогда, когда ошибка модели на тестовом датасете не уменьшается на приемлемую для вас величину. Аналогично работает пошаговое исключение.

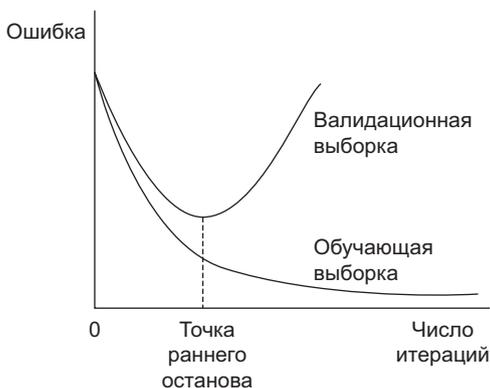
Метод главных компонент (PCA) — это линейный метод сжатия пространства, один из методов машинного обучения без учителя (unsupervised learning), работает только для линейных зависимостей. Сам метод звучит сложно: уменьшение числа фич с минимальной потерей информации путем проекции в ортогональное пространство меньшей размерности. Понять его нам поможет геометрическая интерпретация. Изобразим зависимость числа заказов от потраченных средств в виде диаграммы рассеяния. На графике видно вытянутое облако (рис. 8.9). Теперь изобразим на нем другую систему координат с осями  $X_1$  и  $X_2$ :



**Рис. 8.9.** Метод главных компонент (PCA)

X1 проведем вдоль облака, X2 перпендикулярно ему. Теперь значение каждой точки мы можем взять в этой новой системе координат, и скажу больше — мы можем оставить только значение по оси X1 и получить только одну фичу X1 вместо двух коррелирующих между собой. X2 мы можем выбросить, так как разброс (вариативность) значений по этой оси намного меньше, чем по X2. Таким образом, мы нашли совершенно новую фичу X1, которая несет информацию о прошлых покупках пользователя и хорошо заменяет две старые фичи. Примерно так и работает PCA, этому методу передается нужное число фич (размерность пространства), метод производит все необходимые операции и сообщает, сколько информации мы потеряли из-за этого преобразования (доля объясненной дисперсии). Подобрать размерность пространства можно по этому показателю.

Четвертый способ борьбы с переобучением — остановить обучение раньше (рис. 8.10).



**Рис. 8.10.** Досрочная остановка обучения

Для этого на каждом шаге итерации нужно считать ошибки на обучающем и тестовом датасете. Остановку сделать в момент «перелома» тестовой ошибки в сторону увеличения.

Пятый вариант — регуляризация. Регуляризация представляет собой сумму коэффициентов модели, умноженную на коэффициент регуляризации. Регуляризация добавляется к функции ошибки, которую оптимизирует ML-модель. Есть несколько типов регуляризаций: L1 — сумма модулей коэффициентов, L2 — сумма квадратов коэффициентов, Elastic Net — сумма L1 и L2 регуляризаций с отдельными коэффициентами. Задачей регуляризации является пессимизация коэффициентов с большими значениями, чтобы какая-то одна фишка не перетянула одеяло на себя. Коэффициенты регуляризации являются так называемыми гиперпараметрами модели, их тоже нужно подбирать таким образом, чтобы получить меньшую ошибку. На практике регуляризация L2 используется чаще.

И наконец, шестой способ — использовать ансамбли алгоритмов.

## АНСАМБЛИ

Теорема No Free Lunch (или по-нашему — халявы не бывает) гласит, что не существует единственного алгоритма, который будет самым точным для любых задач. Аналитики могут заниматься ручным трудом, подбирая все новые и новые модели, которые наилучшим образом решают проблему. Но что если попытаться объединить разные модели в одну большую, каким-либо образом аккумулируя результат? Тогда мы получим новый алгоритм — ансамбль алгоритмов, точность которого может быть очень высокой, даже если использовать внутри «слабые» алгоритмы, чья точность чуть выше обычного подбрасывания монетки. Развитие вычислительных мощностей (больше памяти, мощные процессоры) с легкостью позволило сделать это.

Способов объединения простых алгоритмов в ансамбли придумано много, но мы рассмотрим два наиболее известных типа — бэггинг (bagging) и бустинг (boosting). Бэггинг (Bagging = Bootstrap aggregating) был предложен Лео Брейманом в 1994 году. Суть

метода заключается в создании множества тренировочных датасетов, слегка отличающихся от исходного. Делать это можно двумя способами — случайно выбирая (сэмплируя) записи из датасета и случайно выбирая подмножество фич из датасета. Обычно эти два способа совмещают: случайно выбираются и записи, и фичи. Само сэмплирование данных осуществляется с замещением — мы не удаляем строки из исходного датасета, а значит, какие-то данные попадут в новый датасет несколько раз, а какие-то вообще не попадут.

Базовые алгоритмы для бэггинга должны обладать склонностью к переобучению — например, глубокие деревья решений с большим числом ветвей. Затем на каждом тренировочном датасете обучается базовый алгоритм. Для получения результата всего ансамбля результаты обучения на всех датасетах усредняются. Самый известный алгоритм — Random Forest, его несложно написать самому, но можно воспользоваться готовыми реализациями [56].

Бустинг строится совсем по иным принципам. В качестве базовых используются очень простые алгоритмы, точность которых чуть выше подбрасывания монетки. Это могут быть деревья решений с очень низкой глубиной ветвления. Принцип работы следующий: обучается первое дерево, затем на его ошибках обучается второе — и так далее, пока мы не достигнем требуемой точности. Все обученные деревья приобретают свой вес, пропорциональный их собственной точности. А когда нужно получить ответ от голосующей модели, используются эти веса: чем вес больше, тем выше влияние на результат (например, AdaBoost).

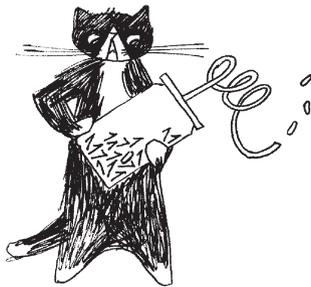
Самые известные алгоритмы на основе бустинга деревьев (Gradient Boosting Decision Tree) — XGBoost [57], LightGBM [58] от Microsoft

и CatBoost [59] от Яндекса. Они являются топовыми алгоритмами, с помощью которых побеждают на Kaggle. Сравним деревья на бустинге и бэггинге. Первый последовательный, второй параллельный, поэтому бэггинг быстрее считается. Его можно распараллелить по нодам кластера или по ядрам процессора. Это бывает важно, например, когда вы хотите быстро получить результат, Random Forest будет быстрее. По точности решения лидирует бустинг, но для этого его нужно долго учить (дни и даже недели), подбирать множество параметров модели (гиперпараметров). Random Forest из коробки учится проще.

Одной из вторичных функций этих ансамблей является список фич, отсортированных в порядке их влияния (Feature Importance). Это может быть полезно, если вы хотите использовать какую-либо другую модель, но в датасете слишком много столбцов (фич).

9

# МАШИННОЕ ОБУЧЕНИЕ НА ПРАКТИКЕ



## КАК ИЗУЧАТЬ МАШИННОЕ ОБУЧЕНИЕ

Когда я учился в Твери, в самой обычной школе, и целый год самостоятельно готовился к поступлению в МФТИ, то обкладывался учебниками и решал олимпиадные задачи по четыре часа в день. Два часа уходило на математику, два часа на физику. Я привык так учиться, по-настоящему хорошие книги — мои лучшие друзья.

Я считаю, что получить фундаментальные знания о предмете можно только из книги. Но сейчас время идет быстрее, чем 20 лет назад, и его часто не хватает, чтобы прочитать весь учебник от корки до корки. Приходится часть информации хватать по верхам — и вот тут пригодятся хорошие онлайн-курсы. Я лично пользуюсь Coursera — они первыми сделали хороший коммерческий продукт, за которым стояли профессора Стэнфорда. Я прошел там дюжину различных предметов и именно на Coursera нашел самый лучший курс по машинному обучению. Это курс Machine Learning [61], автором которого является Эндрю Ын. Сам Эндрю — сооснователь Coursera. Я думаю, что его курс сделал очень многое для популярности этой платформы. Цифра зарегистрировавшихся на Machine Learning на сегодняшний момент перевалила за 3 миллиона. Также у Эндрю есть книга по практике машинного обучения [60], она доступна бесплатно.

Отличие этого курса от остальных заключается в том, что в нем выдержан очень хороший баланс между практикой и теорией. У меня фундаментальное образование МФТИ по физике и математике, но я не люблю, когда практический курс изобилует формулами и выкладками. Курс должен давать минимальную теоретическую информацию, чтобы сделать первые практические шаги как можно быстрее. После курса слушатель должен не просто понимать, как

ему что-то сделать, но и почему. А глубоко изучать теорию нужно по книгам, а не по курсам.

Еще один большой плюс курса Эндрю Ына — вы будете сами программировать алгоритмы и поймете, как они устроены внутри. Это знание лишает вас страха перед машинным обучением — оно перестанет быть для вас магией и станет ремеслом. Я уверен, тот, кто усвоил школьный курс математики и знает технический английский, сможет без больших усилий пройти этот курс [61]. В нем не хватает только пары вещей — деревьев решений и ансамблей. Кстати, Machine Learning Эндрю Ына рекомендует и Шавье Амастриан на своей странице Quora.com.

Как еще можно освоить машинное обучение? Самостоятельно писать алгоритмы. Шавье рекомендует [62] следующую последовательность действий: взять хорошую книгу по машинному обучению, прочитать вводные главы, затем выбрать в книге алгоритм, который вам интересен, и написать его на том языке программирования, которым вы владеете. Если какие-то вещи непонятны, то можно подсмотреть готовую реализацию этого алгоритма в коде каких-либо библиотек (но лучше делать это по минимуму). Я давно заметил: чем сложнее даются знания, тем лучше они усваиваются. Не нужно бояться изобретать велосипед, эти знания пригодятся в дальнейшем. Я лично воспользовался советом Шавье несколько лет назад и написал свою небольшую библиотеку для машинного обучения на Scala [63]. В то время я читал книгу по дизайну программ на Scala и решил объединить эти знания и курс на Coursera. Для библиотеки использовал векторизованные вычисления так же, как это делал Эндрю Ын.

Если вам этот подход кажется слишком сложным из-за программирования, то Эндрю дал совет. Есть одна страшная программист-

ская тайна — мы часто ищем решение какой-либо проблемы на [stackoverflow.com](https://stackoverflow.com). Это база или форум с вопросами и ответами, которые интересуют разработчиков. Когда мы находим ответ, то часто просто копируем код из форума в собственную программу. Эндрю советует не копировать, а перепечатать на клавиатуре ответ, а еще лучше — разобраться, почему он правильный. Он считает, что так разработчик или аналитик быстрее вырастут в профессиональном плане. Нет ничего страшного в копировании, многие художники занимались этим, прежде чем стать великими, считает он.

## СОРЕВНОВАНИЯ ПО ML

Второго октября 2006 года компания Netflix объявила конкурс «Netflix Prize»: кто улучшит текущие рекомендации Netflix на 10 % по метрике RMSE, получит один миллион долларов призовых. В сентябре 2009 года команда победителей «BellKor's Pragmatic Chaos» получила свой миллион долларов. Соревнование длилось почти три года, задача была непростой.

Параллельно под эгидой научной конференции ACM SIGKDD такие соревнования проходили на платформе KDD Cup. Каждый год — это новое соревнование со своими организаторами, данными и правилами.

Все эти события привели к созданию платформы для коммерческих соревнований по машинному обучению — [Kaggle.com](https://kaggle.com). Компания была основана в 2010 году тремя людьми и поглощена Google в 2017 году. Сейчас Kaggle предоставляет много сервисов, но первый и самый главный — соревнования по машинному обучению с хорошими призовыми. Система полностью аналогична конкурсу от Netflix: какая-то компания публикует свои данные и правила участия, по которым будут выбираться победители. В день окончания баллы всех команд фиксируются. Победители получают призы, а компания — решение своей задачи. Часто решение и его описание потом публикуются на форуме.

В соревнованиях по машинному обучению прокачиваются навыки практического использования ML и создания фич на базе датасета. Там может поучаствовать каждый зарегистрировавшийся и получить очень хороший опыт. Все выглядит отлично, не правда ли? Но в них есть другая сторона — эти решения нельзя использовать в лоб, можно взять оттуда лишь некоторые идеи. Например, сам Netflix заявил [65], что алгоритм — ансамбль победителей состоял из 107 субалгоритмов, из которых только два дали самый значимый результат: факторизация матриц (SVD) и ограниченная машина Больцмана (RBM). В компании не без труда внедрили эти два алгоритма в рабочую систему. Сработало правило Парето: 20 % усилий (2 алгоритма) дали 80 % результата. Отмечу еще раз: они не стали внедрять всего монстра целиком, а взяли всего лишь два его элемента. Победивший алгоритм невозможно внедрить, он очень ресурсоемкий и сложный. Его поддержка стоила бы космических денег.

Это и есть основной недостаток решений, полученных на таких соревнованиях, — нет ограничений на вычисления и простоту результата. Такие решения часто будут нежизнеспособными конструктами. И тем не менее я все равно призываю вас участвовать в соревнованиях, это полезно. Подсматривайте решения на форумах и повторяйте их, учитесь делать фичи — это непросто, но в них заключено искусство ML. Не нужно занимать топовые места, достаточно, чтобы метрики ваших решений были процентов на пять хуже лидера. Даже если вы просто окажетесь выше медианы оценок — уже хорошо. Так вы научитесь многому.

Если бы у меня был выбор между двумя кандидатами: первый занимает призовые места на Kaggle и имеет за плечами десятки моделей, а второй реализовал всего две, но придумал задачу, решил, внедрил ее и доказал метриками, что она зарабатывает деньги для

компании, — я бы предпочел второго. Даже если ему не придется повторять на новом месте все эти этапы, я могу сделать вывод, что он способен видеть картину целиком, а значит, сможет говорить на одном языке с людьми, которые будут внедрять продукт его труда, без проблем будет понимать ограничения и требования смежных департаментов.

## ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ

Искусственный интеллект (AI) — очень модный термин, и я его ни разу не использовал в моей книге, хотя занимаюсь именно им. Словосочетание *data mining* я услышал еще в начале двухтысячных, когда работал в StatSoft. За этим маркетинговым термином кроется обычный анализ данных, сделанный из нескольких компонент. Мы с коллегами шутили, что весь этот *data mining* настоящие спецы делают на коленке. Через некоторое время возник новый термин — машинное обучение, он гораздо лучше зашел у специалистов, потому что действительно описывал новую область. Третий термин — большие данные, хайп вокруг которых сейчас уже поутих. Просто технология не оправдала слишком больших надежд, которые были на нее возложены. Я не помню, чтобы на конференциях ACM RecSys хоть раз слышал выражение *big data*, хотя часть игроков, которые там участвуют, обладают очень большими данными (Amazon, Google, Netflix). Компании используют их только для брендинга и продаж своих услуг, чтобы показать, что они в тренде. Иначе их обойдут конкуренты.

Об AI широко заговорили с появлением нейронных сетей глубокого обучения (Deep Learning). Принцип работы нейрона как строительной единицы нейронной сети был заимствован из биологии. Но согласитесь, это еще не повод считать нейронную сеть интеллектом, близким хотя бы к интеллекту насекомых. Пока те операции, которые реализуются нейронными сетями, очень примитивны по сравнению с тем, на что способны даже самые примитивные живые существа (например, синтезировать новую жизнь,

не говоря уже о полностью самостоятельном принятии решений). По моему мнению, человечество сможет приблизиться к созданию интеллекта, близкого к интеллекту животного, лишь тогда, когда сумеет синтезировать и обучать биологические нейронные сети, не полностью электронные.

Вместо абстрактного искусственного интеллекта я предпочитаю использовать более конкретные термины, например компьютерное зрение. Благодаря нейронным сетям именно в этой области произошел самый большой прорыв. Сейчас компьютерное зрение используется везде — от тегирования людей в мобильных телефонах и соцсетях до самоуправляемых автомобилей. Его используют и государства для выполнения полицейских функций, и коммерческие организации для решения своих задач. Мне лично нравятся примеры, когда дружба железа и софта приносит практическую пользу. Например, есть робот Stingray, который уничтожает вшей искусственно выращиваемого лосося с помощью компьютерного зрения и лазера [73]. Этот паразит является причиной массовой гибели рыбы при искусственном разведении. Например, компания «Русская аквакультура», крупнейший российский производитель искусственно выращенного лосося, в 2015 году потеряла больше 70 % рыбы, которая погибла из-за вспышки лососевой вши. Потери из-за мора компания оценивала в 1 млрд руб. [74]. А вот подводный робот позволяет решать проблему — заметив паразита на теле рыбы, он уничтожает его с помощью лазера.

Второе направление большого прорыва — роботизация. Здесь все не ограничивается только компьютерным зрением. Когда я был в музее MIT в Бостоне, то обратил внимание, что проект Boston Dynamics уходит корнями в 80-е, в лаборатории MIT. Уже тогда ученые этого лучшего университета мира занимались компьютерным зрением

и управлением роботов. В те годы у них уже был прыгающий на одной палке робот, который не падал. Boston Dynamics выделились из MIT в 1992 году. Сейчас компания известна своими роботами, которые давно стали героями YouTube и бьют рекорды просмотров. Недавно Boston Dynamics купила корейская Hyundai за 1 миллиард долларов. Если честно, в такие моменты я не понимаю наших сверхбогатых соотечественников — мне кажется, гораздо интереснее вкладывать деньги в такие проекты с перспективой стать вторым Илоном Маском, чем в футбольные клубы. Несмотря на то что новаторские проекты вроде Boston Dynamics пока плохо коммерциализированы, их время еще придет — ведь туда идет человечество.

Заменит ли AI людей? Думаю, что да. И это сделает бизнес. Сам по себе бизнес подчиняется жадным алгоритмам: если есть возможность сэкономить — это будет сделано. Когда-то с целью экономии многие западные компании начали размещать производства в Юго-Восточной Азии, где труд рабочих стоил намного дешевле. С внедрением роботизации число рабочих на единицу продукции уменьшается, логистические расходы в какой-то момент становятся выше трудовых, и тогда становится выгоднее производить товар в стране, где осуществляются продажи. Как пример — создание роботизированных фабрик Speedfactory компании Adidas. Были открыты две фабрики в Германии и США в 2016 и 2017 годах [74]. Целью было сделать производство ближе к покупателю. В 2019 году компания приняла решение закрыть эти фабрики. Несмотря на эту неудачу тенденция налицо — роботизация производства будет заменять все больше людей.

## НЕОБХОДИМЫЕ ПРЕОБРАЗОВАНИЯ ДАННЫХ

Перед тем как скармливать данные моделям ML, нужно провести над ними несколько важных преобразований:

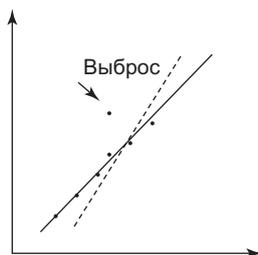
- стандартизацию данных (приведение к единой шкале);
- удаление выбросов;

- подготовку категориальных переменных;
- работу с пропущенными данными (missing data);
- сэмплинг несбалансированных классов.

Для линейных моделей можно нормализовать данные, так как часто сами данные представлены на разных шкалах. Например, в датасете есть две фичи: цена квартиры (2 000 000–100 000 000 рублей) и ее площадь (20–500 квадратных метров). Диапазон значений очень разный, поэтому коэффициенты модели теряют физический смысл. Будет невозможно сравнить влияние той или иной переменной на модель. Если использовать регуляризацию, также возникнет проблема — ненужная пессимизация коэффициентов. Есть разные варианты стандартизации, один из них — вычесть среднее и разделить на стандартное отклонение переменной. На выходе получится переменная со средним, равным нулю, и стандартным отклонением, равным 1. На ошибку линейной модели стандартизация не влияет (если без регуляризации), но есть некоторые типы методов, которые чувствительны к шкалам переменных, например метод главных компонент (РСА, о котором я рассказывал в прошлой главе).

Выбросы также могут вносить существенную ошибку в модель. Прямая линейной регрессии пройдет по-другому, если удалить выбросы, что особенно важно, когда данных мало. Удаление выбросов — непростая задача. Самый простой способ — удалить данные, которые лежат вне какого-либо перцентиля, например 99-го. На графике (рис. 9.1) представлен пример, как выброс изменил прямую, пунктиром показана прямая линейной регрессии для данных с выбросом, сплошной — без выброса. Видно, что точка выброса «поворачивает» прямую в свою сторону.

Категориальные переменные мы уже обсуждали в главе о данных. В их использовании есть нюансы. Обычно нет никаких проблем с бинарными переменными (да/нет, 0/1), нужно лишь свести их к значениям 0 и 1 (dummy variable), если работаете с линейными моделями. Сама операция называется label encoding. Что касается



**Рис. 9.1.** Выброс меняет поведение

деревьев решений, нужно смотреть документацию конкретного метода — в каком виде представить категориальные переменные. Когда идет работа с категориальной переменной, у которой три и более значений, в большинстве случаев требуется ее разбить на несколько бинарных. Например, если есть переменная с тремя значениями: Да/Нет/Не знаю, то ее нужно разбить на три переменные (по числу значений). Можно назвать эти переменные по названию значений: Да, Нет, Не знаю. Каждая переменная будет принимать значение 0 или 1. Например, если у исходной переменной было значение «Да», то эти переменные примут следующие значения: Да = 1, Нет = 0, Не знаю = 0. Эта операция кодирования называется *one-hot encoding*. Выполнять ее необходимо потому что, в отличие от непрерывных числовых переменных, взаимоотношения между значениями переменных (больше или меньше) не определены, а значит, операция сравнения значений невозможна. Некоторые методы поддерживают категориальные переменные со множеством значений, например, *catBoost* от Яндекса. Есть еще один вариант кодирования динамичных категориальных значений, например слов текста, — метод называется *hashing trick*. Он нужен, чтобы не заводить огромное количество переменных, когда число значений очень велико.

В работе с данными часто встречается ситуация, когда значения некоторых переменных пустые (*missing data*). В самих данных в зависимости от системы можно увидеть либо пустоту, либо одно из значений: *None* или *Null*. Для категориальных переменных эта проблема решается легко — достаточно просто завести новое значение

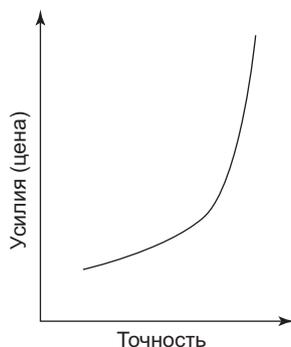
и назвать его «unknown». Для непрерывных числовых переменных это сделать сложнее — нужно понимать, не кроется ли за пустыми значениями какая-то закономерность. Если выяснится, что закономерности нет, можно или удалить данные, или заменить их на среднее значение. Подробно про работу с потерянными данными можно прочитать в книге Эндрю Гельмана [66].

В задачах классификации, когда мы обучаем модель различать два класса, часто возникает проблема несбалансированности этих классов. Это бывает в медицине при тестировании населения и диагностике редких болезней — если в датасете есть данные десяти тысяч людей и из них только десять заболевших, то алгоритму очень сложно обучиться. Другой пример — датасет показа рекламных баннеров в интернете: показов много, а кликов мало. Проблема в том, что модели проще не замечать данные малого класса, а просто предсказывать больший класс, ошибка точности при этом будет приближаться к 100 процентам. Что с этим делать? В курсе по машинному обучению от Google [67] советуют с помощью сэмплирования (downsampling) уменьшить самый большой класс, но назначить этим строкам пропорциональный вес. Это даст быструю сходимость, так как малый класс будет больше влиять на обучение, а веса дадут возможность сразу использовать вероятность, которую мы получим на выходе, для классификации.

## ТОЧНОСТЬ И СТОИМОСТЬ ML-РЕШЕНИЯ

Чем точнее изготовление какой-либо детали на производстве, тем оно дороже. То же самое можно сказать и про машинное обучение. Когда создается первая версия решения, получаем одну степень точности (рис. 9.2). Потом тратится очень много усилий и времени

на улучшение этого результата — к сожалению, рост результата не пропорционален усилиям, которые были на него затрачены (правило Парето никто не отменял!). Мой опыт создания рекомендательной системы говорит, что затраты на каждый процент улучшения растут по экспоненциальному закону. То же самое можно увидеть и в соревнованиях Kaggle.



**Рис. 9.2.** Зависимость стоимости решения от его точности

Это можно учитывать, когда создается первая версия продукта. Не нужно гнаться сразу за сверхточностью, лучше выбрать приемлемую точность. Второй способ — сделать, что получится, довести решение до рабочей системы, а уже затем решать, нужно бизнесу улучшать метрики или нет. Возможно, стоит потратить усилия на другой продукт, вместо того чтобы бесконечно полировать один — а клиенты этого не заметят и не оценят по достоинству.

## ПРОСТОТА РЕШЕНИЯ

Уильям Оккам сформулировал принцип: «Что может быть сделано на основе меньшего числа [предположений], не следует делать, исходя из большего». Этот принцип экономности под названием «брита Оккама» позволяет, например, выстроить цепь гипотез в порядке

возрастания сложности, и именно этот порядок в большинстве случаев окажется удачным. Его также можно использовать при выборе ML-модели. Всегда лучше идти от простого к сложному, от линейных моделей к нелинейным, таким, как, например, нейронные сети.

Почему чем проще, тем лучше? Кроме точности любой ML-модели, есть стоимость ее содержания. Она складывается из вычислительных ресурсов — сложные модели требуют их больше. Специалисты для обслуживания сложных моделей требуются сильные, и у них выше заработная плата. И главное — внесение изменений в сложную систему будет сложным, а это приводит к потере времени и денег. Мы в Retail Rocket при проверке гипотезы всегда следовали простому правилу: если есть две модели, которые по эффективности почти равнозначны, то выбираем всегда модель проще.

В некоторых алгоритмах ML есть опция получения значимости фич (features importances). Этот принцип можно использовать и при отсеке фич. Я уже писал про трактовку коэффициентов линейных моделей — если они стандартизованы, то модуль (игнорируем знак) коэффициента говорит о силе вклада переменной в модель. Для нелинейной оценки можно воспользоваться алгоритмом Random Forests, чтобы получить значимость фич. Эти данные тоже можно использовать для отсека фич. Чем меньше фич будет использовано в модели, тем проще ее будет поддерживать. От большего их числа модель становится только сложнее. И если отсечь наименее значимые фичи, не сильно потеряв при этом в точности, то выводить в рабочую систему такую модель будет проще. Дело в том, что каждая фича требует внимания, отдельных строк кода. За этим нужно следить, и если получится прийти от 20 к 10 фичам, то поддержка будет дешевле, а источников ошибок будет меньше.

Простота системы и отличает рабочие варианты ML-моделей от конкурсов Kaggle. При этом, чтобы добиться простоты, возможно, понадобится приложить больше усилий, чем при разработке очень сложной модели на Kaggle. Стремление к сложности я часто наблюдаю у новичков, они пытаются строить космический корабль там, где можно обойтись самокатом.

## ТРУДОЕМКОСТЬ ПРОВЕРКИ РЕЗУЛЬТАТА

В восьмом уроке от рекомендательной системы Qooqa [68] говорится о том, как важно уметь отвечать на вопрос, почему рекомендательная система дала ту или иную рекомендацию. В Retail Rocket мы также сталкиваемся с такими вопросами. Однажды в качестве альтернативного товара к туалетной бумаге выступила наждачная. Кстати, алгоритм предложил ее из-за реального поведения клиентов — но, конечно, рекомендация выглядела как пранк, и нам было не смешно, ведь с каждым таким случаем приходится разбираться в ручном режиме. В какой-то момент мы написали скрипты и инструкции нашей технической поддержке, чтобы подобные казусы можно было оперативно решать или просто объяснять клиенту без привлечения аналитиков.

Почему так происходит? Когда продукт, в основе которого лежат данные, работает на внешний мир, могут возникать ситуации, неожиданные для пользователей. И если система проста и в ней заложены средства поиска и отладки неисправностей, разрулить их можно быстро.

## MECHANICAL TURK / YANDEX TOLOKA

Даже в проектах с использованием ML-моделей все средства хороши. Совсем недавно писали про компанию ScaleFactor, которая, как утверждалось, использовала искусственный интеллект, чтобы оказывать бухгалтерские услуги [69]. В компанию было вложено порядка 100 миллионов долларов. На деле оказалось, что всю работу делали традиционным способом обычные бухгалтеры.

Впервые о ручном труде в ML я услышал на видео с конференции STRATA, когда сотрудники LinkedIn рассказывали о проекте Skills. Для создания датасета они с помощью сервиса Amazon Mechanical Turk использовали труд тысяч людей, чтобы разметить данные для проекта. Сама модель у них была простая — логистическая регрессия, но датасет для нее нужен был качественный. Конечно, можно было использовать метод анализа текстов, но дешевле и с гарантированным качеством можно получить результат через такие сервисы.

Я уже писал, что одно из преимуществ ML — огромная скорость по сравнению с людьми. Так вот, сервисы, подобные Amazon Mechanical Turk, позволяют использовать труд тысячи людей для решения задачи. Это может быть разметка обучающих примеров, как сделал LinkedIn, или проверка миллионов рекомендаций магазина, как делали мы, — кстати, наши задания по рекомендациям исполнители любили, они были им интересны. Поисковые системы используют такие ресурсы для проверки своей поисковой выдачи. Яндекс вывел в свет свой сервис под названием «Толока» и сделал его общедоступным.

Подобные сервисы работают следующим образом. Заказчик загружает задание и датасет к нему, пишет инструкцию, назначает цену. В датасет можно добавить контрольные примеры. Они пригодятся, чтобы отсеять халявщиков, которые могут выполнять задания быстрее, случайно щелкая по ответам. Поэтому рекомендую обязательно использовать контрольные примеры. После выполнения всех формальностей исполнители получают возможность выполнять задания и получать за это деньги. По моему мнению, заработать там сложно, но как подработка — это вполне себе вариант (вот оно, рабство XXI века — люди на службе AI). От цены за задание зависит количество откликнувшихся исполнителей.

Для каждой задачи требуется свой инструмент. Если делать собственными силами долго, не хватает данных, то сервисы наподобие «Толоки» могут стать хорошим решением. Они позволяют очень хорошо масштабировать задачу и получать результат с приемле-

мым качеством. Да, за это придется заплатить, но сэкономленное время может с лихвой окупиться.

## ML И БОЛЬШИЕ ДАННЫЕ

В 2016 году на конференции ACM Recsys я обратил внимание, что компании Netflix и Quora не рекомендуют пользоваться распределенными системами машинного обучения [68]. Причина проста — они работают намного медленнее, чем параллельные вычисления на одной машине. Я сам столкнулся с этим, когда мы считали GBRT (Gradient Boosting Regression Tree) модель, используя наш вычислительный кластер и библиотеку MLLib в Spark. В тот момент мы пробовали это делать на одной машине, в память данные не поместились, поэтому воспользовались распределенным алгоритмом. Все бы хорошо, но он считал модель два часа. Это слишком долго, учитывая, что модель была совсем несложная. Тогда мы оптимизировали данные и попробовали посчитать на локальной библиотеке Smile на Java. Все посчиталось за пять минут.

Проблемы с распределенными алгоритмами происходят из-за медленной сетевой скорости. Различным нодам кластера приходится постоянно координироваться между собой, передавать данные и параметры по обычной локальной сети. Скорость работы с памятью примерно в 50 раз быстрее гигабитной сети, поэтому локальные вычисления на одной машине работают значительно быстрее. Да и одна машина стоит гораздо дешевле, чем использование дорогого кластера.

## REGENCY, FREQUENCY И MONETARY

Впервые с закономерностями поведения клиентов я познакомился в книге Джима Ново [71]. Джим рассказывал в книге о незнакомом мне тогда способе сегментации RFM: Recency (давность), Frequency (частота) и Monetary (деньги).

Recency — давность какого-либо действия клиента. Для сегментации очень важно эмпирическое свойство Recency — чем меньше времени прошло с момента последней активности клиента, тем вероятней, что он повторит действие. Например, пусть Recency — это давность последнего заказа клиента. Нужно сравнить двух клиентов: у первого давность последнего заказа — 30 дней (30 дней назад он сделал свой последний заказ), у второго — 70 дней. Как вы думаете, какой клиент с большей вероятностью повторит заказ? Правильно, первый (давность — 30 дней).

Frequency — количество действий, которые совершил клиент. Для нас важно свойство Frequency — чем больше каких-либо действий совершит клиент, тем больше вероятность того, что он повторит их в будущем. В литературе и на сайтах основателей этого метода не ограничивается временной интервал, в течение которого измеряется Frequency. По своему опыту скажу, что ограничивать этот интервал нужно. Например, считать Frequency только в течение 360 дней, предшествовавших дате анализа. Пусть Frequency — количество заказов, сделанных в течение 360 дней: у первого клиента — 10 заказов, у второго клиента — 5 заказов. Понятно, что у первого клиента вероятность сделать в будущем заказ выше, чем у второго.

Monetary — сумма денег, которую потратил клиент. Здесь все, как у Frequency, — нужно постараться ограничить время, в течение которого измеряется величина; и чем больше денег было потрачено, тем больше вероятность того, что клиент вновь сделает заказ. На практике Monetary обычно не используют, так как этот показатель сильно коррелирует с Frequency. Поэтому RFM-сегментация в большинстве случаев называется RF-сегментацией.

Итак, у нас есть два параметра для сегментации — Recency (далее R) и Frequency (далее F), оба эти параметра могут прогнозировать дальнейшее поведение клиента с определенной точностью. И если объединить их в один параметр RF — то точность прогноза повышается в разы. Далее я приведу последовательность шагов (по методике Джима Ново):

- Параметр R — бьется на пять частей, и появляются пять значений от 1 до 5. 5 — это когда заказ был сделан совсем недавно.
- Параметр F — бьется на пять частей, и появляются пять значений от 1 до 5. 5 — это когда клиент в течение определенного периода времени (этот период тоже нужно рассчитать) сделал очень много заказов.
- Строится RF-сетка (grid): в виде двузначной комбинации R и F. 55 — сегмент лучших клиентов, 11 — самых худших клиентов.
- Вычисляются вероятность совершения следующего действия для каждого сегмента.
- 25 RF сегментов объединяются по вероятностям (из прошлого шага) в большие сегменты.

С точки зрения RFM, самый лучший клиент — это тот (рис. 9.3), который совершил покупку совсем недавно, до этого сделал их много на хорошую сумму денег. Этот фундаментальный принцип помог создавать фиши, которые предсказывают вероятность совершения действий в дальнейшем. Его можно распространить на любые действия людей, кроме покупок: вероятность заболеть, вероятность вернуться на сайт, вероятность попасть в тюрьму, вероятность кликнуть на баннер. Всего лишь с помощью этих переменных и простой линейной модели на одном из конкурсов Kaggle я смог

получить очень неплохой результат. Для лучших результатов, кроме действительных цифр, я использовал бинарное кодирование. За базу можно взять сегментацию, о которой я написал выше. Можно брать отдельно переменные R и F или целиком RF.

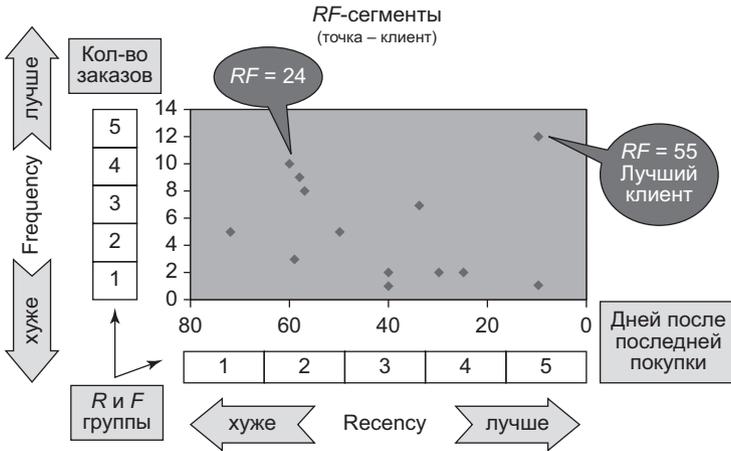


Рис. 9.3. RF-сегментация

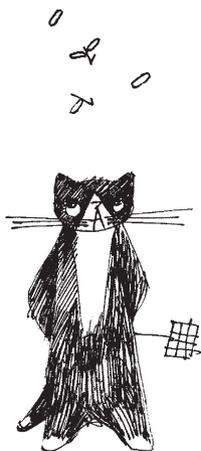
## ПОСЛЕДНИЙ СОВЕТ

Кроме каких-либо теоретических книг в качестве дополнительных источников знаний рекомендую два бесплатных ресурса: книгу Эндрю Ына [60] про практику машинного обучения и правила Google для инженерии ML-проектов [72]. Они помогут в дальнейшем совершенствовании.



# 10

## ВНЕДРЕНИЕ ML В ЖИЗНЬ: ГИПОТЕЗЫ И ЭКСПЕРИМЕНТЫ



Все эксперименты проводятся для того, чтобы дать фактам возможность опровергнуть нулевую гипотезу.

*Сэр Рональд Фишер,  
«Планирование экспериментов» (1935)*

Модели ML рождаются, живут и умирают. Жизнь меняется, это закон природы: если что-то долго не меняется, то оно умирает. Улучшая и оптимизируя модель, мы даем ей новую жизнь и надежду. Помочь нам в этом могут гипотезы (или идеи) и эксперименты, подтверждающие или отвергающие гипотезы. В 2016 году на сцене концертного зала MIT я рассказывал про то, как убивать гипотезы как можно раньше. Доклад зашел на ура, поэтому я решил изложить те идеи и выводы в этой главе.

## ГИПОТЕЗЫ

Гипотеза — это идея по улучшению продукта. Неважно, что это — сайт, товар или магазин. Существует даже должность менеджера по продукту, одной из задач которого является создание и поддержание списка таких гипотез, расстановка приоритетов их исполнения. Список гипотез еще называют бэклогом (backlog). Он является важным стратегическим элементом развития компании. Как придумывать гипотезы и расставлять их в порядке приоритетов — тема отдельной большой книги. Если кратко, идеальная ситуация выглядит так — продуктологи взаимодействуют с рынком, с существующими и потенциальными клиентами, изучают конкурентные решения, проводят фокус-группы, чтобы понять, сколько то или иное изменение (гипотеза) принесет компании денег. На основе этих исследований гипотезы попадают в список и приоритизируются. Бизнес требует денежных метрик для приоритизации гипотез, чем точнее они подсчитаны, тем лучше. Но в реальности с большинством гипотез сделать это очень сложно, и оценка происходит по принципу «пальцем в небо». Самые гром-

кие коммерческие успехи в истории были революционными, а не эволюционными — вспомните хотя бы появление первого iPhone.

Приоритизация гипотез служит главной цели — как можно быстрее достичь успеха. Если следовать этой логике, идеи, которые с большей вероятностью могут дать результат, должны быть первыми в списке на реализацию. Но у каждой гипотезы есть такая характеристика, как сложность ее реализации, — вот почему, приоритизируя гипотезы, важно оценивать их трудоемкость и стоимость инфраструктуры (сервера, наем дополнительного персонала). Допустим, первая гипотеза обещает принести примерно 10 млн рублей за год, при этом затраты на ее реализацию — это месяц работы двух разработчиков и одного аналитика данных. Вторая обещает 2 млн рублей за год, при этом реализовать ее смогут два человека за пять дней работы. Какую гипотезу выбрать первой? Это решение я оставляю за менеджментом, однозначного совета дать здесь не могу.

Что если гипотезы и их приоритизацию делать внутри отделов? С одной стороны, этот подход кажется правильным — минимум централизации, максимум скорости. Но давайте представим себе, что компания — это живой организм, а ее самый сильный отдел (например, IT) — это руки. У отдела хороший список гипотез, и приоритеты расставлены более правильно, чем у других отделов, — то есть руки прокачаны как следует. А теперь представим себе соревнования по триатлону — на олимпийской дистанции нужно проплыть 1500 метров, сразу после этого сесть на велосипед и проехать 40 км, а затем пробежать 10 км. Сильные руки пригодятся на первом этапе, но в двух других дисциплинах нужны уже сильные ноги. Если они не так хорошо натренированы, спортсмен проиграет гонку более сбалансированным соперникам или даже может сойти с дистанции. В бизнесе, как в спорте, невозможно сделать ставку на один отдел — нужен сбалансированный подход. Я сам проходил это в Retail Rocket — варился в собственном соку, приоритизировал свои гипотезы сам. Да, мы стали очень сильными в одной области, но остальные команды не успевали за нами. Если вернуться назад, я бы сделал ставку на совместную работу, продукт и рынок.

Все гипотезы из списка невозможно протестировать. Большинству уготовано так и остаться навеки гипотезами. Это нормально и даже хорошо — значит, более выгодные идеи реализуются раньше остальных. Каждая гипотеза потребляет ресурсы, они не бесконечны, поэтому невозможно протестировать все идеи. Скажу больше — 9 из 10 гипотез не принесут результата. Но понятно это может стать только на одном из многочисленных этапов ее тестирования. Моя теория заключается в том, что нужно убивать гипотезу как можно раньше, как только мы получим первый сигнал о том, что она не взлетит. Это экономит ресурсы — много ресурсов! — и даст шанс лучшим гипотезам, которые ожидают своей очереди.

Я сравнивал разные гипотезы и их отдачу. Эволюционные гипотезы, где один параметр слегка оптимизируется, в случае успеха дают меньший эффект по сравнению с революционными гипотезами, где подход принципиально иной. Но вероятность успеха как такового у эволюционной гипотезы выше.

## ПЛАНИРУЕМ ТЕСТ ГИПОТЕЗЫ

Пусть у нас есть готовая гипотеза, которую бизнес признал самой горячей. У нас есть все ресурсы, и мы готовы взять ее в работу. Какая еще информация нужна? Во-первых, цель гипотезы — какую количественную метрику она будет оптимизировать? Мы уже понимаем, что количественные метрики неидеальны, но нам она нужна для отслеживания изменений. Здесь метрика — это то число, значимо улучшив которое можно покупать ящик шампанского.

Во-вторых, нужно понимать, как мы будем тестировать гипотезу и где. В машинном обучении есть два вида тестирования: офлайн и онлайн. Офлайн дает метрики на уже существующих данных — о них я писал в главе 8 «Алгоритмы машинного обучения». В онлайн-тестировании нужно получить интересующие метрики и сравнить их с помощью статистических тестов.

Основоположник планирования экспериментов (тестирования гипотез) сэр Рональд Фишер в 1925 году написал монографию

«Статистические методы для исследователей», в которой изложил такие понятия, как статистический критерий значимости, правила проверки статистических гипотез, дисперсионный анализ, планирование эксперимента. Это определило наш сегодняшний подход к планированию экспериментов. Вы наверняка слышали про тестирование вакцины от COVID-19 — ее тестировали методом двойного слепого рандомизированного плацебо-контролируемого исследования. Это самое достоверное клиническое исследование, применяемое в доказательной медицине. Рандомизированное — значит распределение пациентов по опытной и контрольной группам происходит случайно. Для чистоты эксперимента крайне важно, чтобы исследователи не могли собрать более легких больных в опытную группу, а более тяжелых — в контрольную. Поэтому существуют специальные методы рандомизации (перемешивания), чтобы в итоге различия между группами стали статистически недостоверными, а результаты исследования более точными. Именно Фишер предложил способ планирования и проведения таких экспериментов. Он некоторое время работал в лаборатории сельского хозяйства в Ротамстеде. Планируя эксперимент с удобрениями [76], исследователь не знает о множестве факторов, которые могут повлиять на результат. Поэтому, пытаясь ответить на вопрос «Какое удобрение лучше?», нет смысла сравнивать рост растений в разных теплицах, в каждой из которых вносили свое удобрение. Сравнить нужно рост одного и того же растения, получившего два вида удобрений в одной теплице. Кроме того, даже в одной теплице солнечный свет будет падать под разным углом на разные участки, и влажность тоже может быть неравномерной. Поэтому при выборе удобрения А или удобрения Б для каждой лунки нужно подкидывать монетку — орел или решка. Фишер назвал такой подход к планированию эксперимента принципом рандомизации. Только в этом случае можно определить, является

ли разница между удобрениями значимой. И лишь соблюдая этот принцип, мы имеем право сказать, что два удобрения находились настолько в равных условиях, насколько это возможно, и почти все неконтролируемые различия устранены.

До Фишера распределение в таких экспериментах производилось систематически, что могло исказить результаты. Интересно, что многие ученые не сразу приняли его метод, считая свой систематический подход верным. Кроме обычных А/Б-тестов, Фишер предложил схемы для более сложных многофакторных тестов. На деле даже с обычными тестами с двумя группами часто возникают проблемы, и до многофакторных тестов, когда проверяется сразу несколько изменений одновременно, редко кто доходит. Поэтому в этой книге я буду фокусироваться на самых простых тестах с двумя группами.

Итак, для проведения теста нам нужны метрика и рандомизация. Тесты проводят с контрольной группой. В медицине группу пациентов делят случайно на две — первой группе дают исследуемое лекарство, второй — лекарство-пустышку под названием плацебо. В маркетинге делается аналогично. Во времена почтовой торговли промоскидки отправляли одной группе клиентов, письма-пустышки (без скидок) — второй. При рассылке email-сообщений интернет-магазина контрольной группе обычно не отправляют ничего. Amazon.com, который был пионером тестирования в интернете, использовал А/Б-тесты (split test) для показа одной группе пользователей старой версии сайта, а второй — новой, и сравнивал их поведение, чтобы выбрать лучшую версию. Перед запуском полноценного боевого теста нужно проверить весь механизм работы, делается это с помощью симуляционного и реального тестов. Также можно использовать А/А-тесты — расскажу о них далее.

## ЧТО ТАКОЕ ГИПОТЕЗА В СТАТИСТИКЕ

Для статистической проверки гипотез нам понадобится два важных понятия — генеральная совокупность и выборка. Генеральная совокупность (general population) — это все объекты, относи-

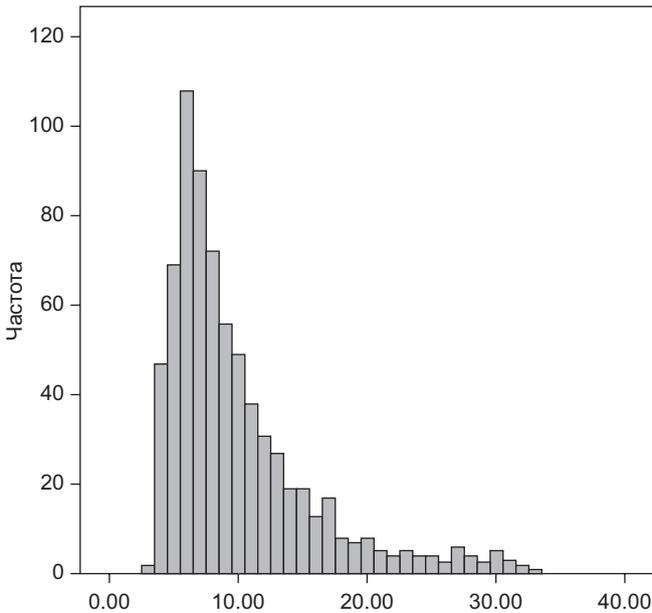
тельно которых нужно сделать выводы в исследовании. Выборка (sample) — это часть объектов генеральной совокупности, которые мы смогли пронаблюдать.

Пусть у нас есть огромный резервуар с шарами разного диаметра. В самом резервуаре сотни тысяч шаров. Средний диаметр неизвестен, и нам нужно его определить. Весь резервуар посчитать невозможно, слишком много работы нужно затратить. Для экономии средств и времени мы сделаем случайную выборку с замещением (возвращаем шар обратно после определения диаметра) определенного количества шаров. В этой задаче резервуар с шарами — это генеральная совокупность, средний диаметр шара — неизвестный параметр, который нам нужно определить, и мы сделаем это с помощью случайной выборки. Параметр в генеральной совокупности является истинным, параметр выборки является его оценкой.

Когда я слышу слово «распределение» — представляю себе гистограмму частот появления значений. В нашем примере это будет гистограмма с диаметрами шаров. Мы работаем с непрерывными числовыми значениями, вся шкала гистограммы разбивается на диапазоны, как правило, равной длины (0–10, 10.01–20, ...). На основе гистограммы сложно принимать решения, поэтому в гипотезах обычно оценивают какой-то отдельный параметр распределения, например среднее или медиану. Строим по ним гистограмму (рис. 10.1).

Такие гистограммы (распределения) очень сложно сравнить друг с другом, поэтому и используются числовые статистики распределений.

Генеральная совокупность имеет свое распределение шаров, выборка — свое. Чем больше выборочное распределение похоже на распределение генеральной совокупности — тем лучше. Случайность вытаскивания шаров очень важна для этого — ведь шары в резервуар могли насыпать сначала одного диаметра, потом другого. Тогда на поверхности могут оказаться самые большие шары, и если мы их будем брать преимущественно оттуда, то наше распределение шаров внутри выборки окажется смещенным в сторону большего



**Рис. 10.1.** Пример распределения

диаметра, поэтому наши выводы могут оказаться неверными. Возвращать шары нужно, чтобы работать с исходным распределением генеральной совокупности, так как каждое вытягивание будет независимо от предыдущих. Теперь давайте применим интуицию — чем больше шаров мы вытянем, тем лучше распределение выборки будет похоже на распределение в резервуаре, и тем выше точность оценки параметра в выборке мы получим. А сколько нужно вытянуть шаров, чтобы получить приемлемую точность? На этот вопрос уже ответит статистика — об этом чуть позже, а сейчас усложним задачу.

Теперь у нас есть два резервуара, нужно сравнить средний диаметр шаров между ними. Самое время перейти к формулировке гипотезы. Для этого нам понадобится сформулировать основную ( $H_0$ ) и альтернативную гипотезу ( $H_1$ ) с точки зрения статистики и проведения экспериментов:

- Нулевая гипотеза  $H_0$  (null hypothesis) утверждает, что метрика в эксперименте не изменилась и все наблюдаемые изменения случайны.
- Альтернативная гипотеза  $H_1$  (alternative hypothesis) утверждает, что метрика в эксперименте изменилась, наблюдаемые изменения не случайны.

Тестирование гипотез похоже на суд. Мы считаем, что обвиняемый невиновен, пока не будет найдено строгое доказательство, что он виновен. Аналогично с гипотезами [77], изначально считаем гипотезу  $H_0$  верной, пока не найдем доказательства, чтобы отклонить ее в пользу  $H_1$ .

Теперь переформулируем эти общие утверждения для нашей задачи с двумя резервуарами в виде двусторонней гипотезы:

Гипотеза  $H_0$  утверждает, что средние диаметры шаров в обоих резервуарах равны  $\mu_1 = \mu_2$ .

Гипотеза  $H_1$  утверждает, что диаметры в обоих резервуарах разные —  $\mu_1 \neq \mu_2$ .

Можно также сформулировать в виде односторонней гипотезы:

Гипотеза  $H_0$  утверждает, что средний диаметр в первом резервуаре меньше или равен среднему диаметру во втором резервуаре —  $\mu_1 \leq \mu_2$ .

Гипотеза  $H_1$  утверждает, что средний диаметр в первом резервуаре больше среднего диаметра во втором резервуаре —  $\mu_1 \geq \mu_2$ .

С моей точки зрения, лучше использовать односторонние гипотезы. Ведь проверяя какую-либо идею, мы стремимся улучшить

метрику, а значит, нас интересует вопрос, стало ли лучше (гипотеза  $H_1$ ). Далее посмотрим, как статистика делает сравнение.

## СТАТИСТИЧЕСКАЯ ЗНАЧИМОСТЬ ГИПОТЕЗ

Суд может ошибаться, тестирование статистических гипотез — тоже. Определим эти ошибки с помощью таблицы. Они бывают двух типов (табл. 10.1): ошибка первого рода, когда мы ошибочно отклонили нулевую гипотезу  $H_0$  (признали невиновного виновным), и ошибка второго рода, когда мы ошибочно приняли ее (признали виновного невиновным).

**Таблица 10.1.** Ошибки статистических гипотез

	Оставить верной $H_0$	Отклонить $H_0$
$H_0$ верна	$H_0$ верно приняли	$H_0$ неверно отвергнута (Ошибка 1-го рода)
$H_1$ верна	$H_0$ неверно принята (Ошибка 2-го рода)	$H_0$ верно отвергли

На языке статистики ошибки описываются вероятностями:

Вероятность ошибки 1-го рода:  $\alpha$ . Обычно исследователи используют  $\alpha = 0.05$  (5%).

Вероятность ошибки 2-го рода:  $\beta$ . Величина  $(1 - \beta)$  называется мощностью, которая является вероятностью найти улучшение, если оно есть.

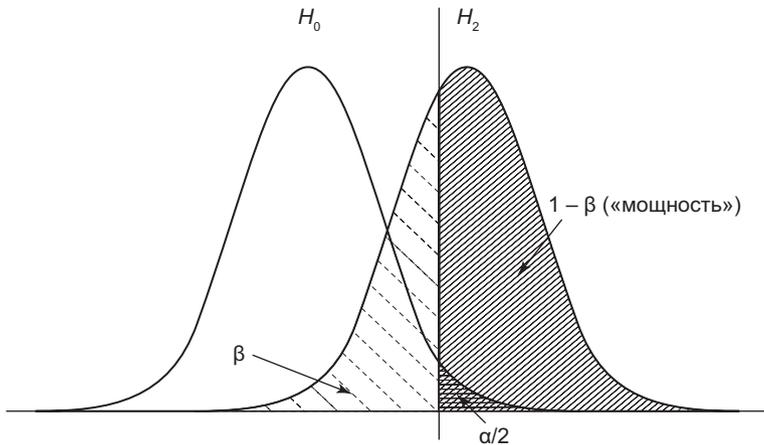
Для упрощения тестирования гипотез Фишер [76] ввел величину  $p$ -значение ( $p$ -value), которая является мерой доказательства против нулевой гипотезы  $H_0$ . Чем она меньше, тем сильнее доказательства против нулевой гипотезы. Важно заметить, что  $p$ -значение — это не вероятность правильности гипотезы  $H_0$ , оно работает только для ее отвержения.

В традиционной, или, как я ее называю, фишеровской статистике,  $p$ -значение — это универсальное число, которое понятно статистикам и позволяет отвергать нулевую гипотезу. До Фишера использовались конкретные статистики, а не  $p$ -значение. Согласно книге Ларри Вассермана «All of Statistics: A Concise Course in Statistical Inference» [77], исследователи обычно используют следующую трактовку  $p$ -значения (табл. 10.2) (для  $\alpha = 0.05$ ).

**Таблица 10.2.** Трактовка  $p$ -значений

$p$ -значение	Доказательство
$< 0.01$	очень сильное против $H_0$
$0.01-0.05$	сильное против $H_0$
$0.05-.10$	слабое против $H_0$
$> 0.01$	малое или нет доказательства против $H_0$

Теперь посмотрим на графическую интерпретацию двусторонней гипотезы. На рис. 10.2 изображено сравнение распределения



**Рис. 10.2.** Статистическая мощность

нулевой и альтернативной гипотез для нашего примера с двумя резервуарами. Каждое распределение представляет плотность вероятности. По сути это две гистограммы с площадью под каждой кривой, равной единице. На графике нулевой гипотезы мы отмечаем две вертикальные линии таким образом, что площадь каждой на хвосте была равна  $\alpha/2$ . В случае односторонней гипотезы строится только одна линия с площадью  $\alpha$ . Эта линия делит распределение альтернативной гипотезы на две части —  $\beta$  и  $(1 - \beta)$ , площади под ними как раз и равны соответственно ошибке второго рода и мощности критерия. Из графика наглядно видно, что чем дальше находятся пики (средние) этих распределений, тем выше мощность и ниже ошибка второго рода (неверное принятие нулевой гипотезы). И это очень логично — чем дальше средние распределений находятся друг от друга, тем становится явнее разница между гипотезами, а значит, нам легче отвергнуть  $H_0$ . С другой стороны, если «уже» распределения, то мощность растет, и нам также легче отвергнуть нулевую гипотезу. Увеличение числа данных в выборке (sample size) способствует «сжиманию» таких распределений.

Именно таким образом работают калькуляторы мощности, которые вычисляют необходимый объем данных для тестов. В калькулятор вводится минимальная детектируемая разность в значениях параметров, уровень  $\alpha$  и  $\beta$  ошибок. На выходе будет объем необходимых данных, которые нужно собрать. Закономерность здесь проста — чем меньшую разницу вы хотите детектировать, тем больше данных для этого нужно.

Альтернативой  $p$ -значению является доверительный интервал. Это интервал, внутри которого находится наш измеряемый параметр

с определенной степенью точности. Обычно используют 95 %-ную вероятность ( $\alpha = 0.05$ ). Если у нас есть два таких доверительных интервала для тестовой и контрольной группы, то по их пересечению можно понять, есть ли между ними отличие.  $P$ -значение и доверительные интервалы — это две стороны одной и той же медали. Интервал удобен для представления данных на графиках. Он часто используется в альтернативных методах оценок А/Б-тестов: байесовской статистике и бутстрэпе.

## СТАТИСТИЧЕСКИЕ КРИТЕРИИ ДЛЯ $P$ -ЗНАЧЕНИЙ

Как мы уже узнали,  $p$ -значение — универсальная метрика тестирования гипотез. Для ее расчета нужно следующее: нулевая гипотеза, статистический критерий, односторонний или двусторонний тест, данные.

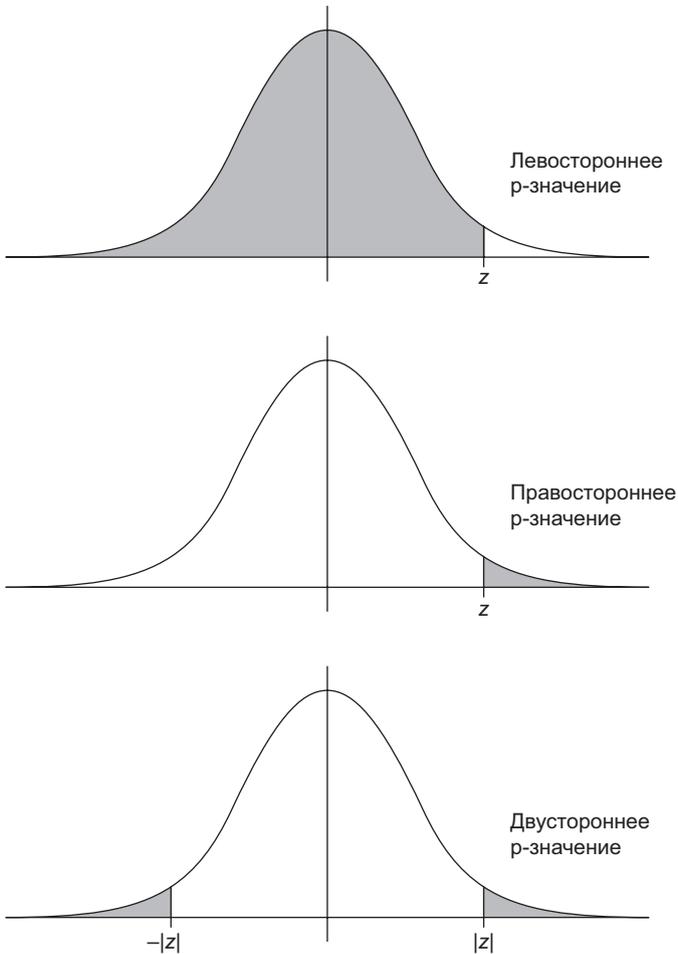
Чтобы определить  $p$ -значение, вам необходимо знать распределение выбранной статистики (статистического критерия), считая, что нулевая гипотеза верна. Далее с помощью кумулятивной функции распределения  $\text{cdf}$  (Cumulative Distribution Function) этой статистики мы можем вычислить  $p$ -значение, как проиллюстрировано на рисунке (рис. 10.3):

- Левосторонний тест:  $p$ -значение =  $\text{cdf}(x)$ .
- Правосторонний тест:  $p$ -значение =  $1 - \text{cdf}(x)$ .
- Двусторонний тест:  $p$ -значение =  $2 \times \min(\text{cdf}(x), 1 - \text{cdf}(x))$ .

Сейчас проще не изобретать велосипед, а пользоваться готовыми калькуляторами в статистических пакетах или программных библиотеках. Важно только выбрать правильный статистический критерий.

Выбор такого критерия зависит от задачи:

- $Z$ -тест для проверки среднего в нормально распределенной величине.



**Рис. 10.3.** Левосторонний, правосторонний, двусторонний тесты

- Т-тест Стьюдента — то же самое, что и z-тест, но для выборок малого объема ( $t < 100$ ).
- Хи-квадрат Пирсона для категориальных переменных и всеческих биномиальных тестов. Очень удобен для расчета конвер-

сий, например посетителей в покупателей, где нужен биномиальный тест — купил или нет.

- Тест Стьюдента для двух независимо распределенных выборок очень хорошо подходит для нашей задачи с двумя резервуарами или для сравнения средней суммы покупки.

У таких тестов есть одна проблема — они привязаны к распределению. Например, для тестов Стьюдента и  $z$ -теста нужны нормально распределенные данные. Форма таких данных формирует «колокол» на гистограмме. Например, распределение средних чеков покупок не образует такого распределения. Конечно, можно их преобразовать логарифмированием и собрать в форму колокола, но часто это неудобно. Первой альтернативой для ненормально распределенных данных являются непараметрические тесты.

Хотя согласно статистическому словарю STATISTICA [78] — непараметрические методы наиболее приемлемы, когда объем выборок мал. Если данных много (например,  $n > 100$ ), то не имеет смысла использовать непараметрические статистики. Дело в том, что когда выборки становятся очень большими, то выборочные средние подчиняются нормальному закону, даже если исходная переменная не является нормальной или измерена с погрешностью. Непараметрические тесты имеют меньшую статистическую мощность (менее чувствительны), чем их параметрические конкуренты, и если важно обнаружить даже слабые отклонения, следует особенно внимательно выбирать статистику критерия.

В нашей задаче с резервуарами можно применить тест Стьюдента для двух независимых выборок. Второй альтернативой является универсальный инструмент — бутстрэп.

## БУТСТРЭП

Это один из самых интересных способов оценки метрик в А/Б-тестах, мы с удовольствием используем его в Retail Rocket для непрерывных параметров, таких как стоимость средней покупки, средняя стоимость товара, средний доход на посетителя сайта (Revenue per Visitor, RPV).

Бутстрэп [79] (оригинальная статья) работает за счет многократных выборок из данных, по которым затем считаются статистики. Алгоритм выглядит следующим образом [80]:

1. Необходимо задать количество выборок  $k$ , которые мы сделаем из исходного датасета. Само число должно быть не меньше сотни. Больше — лучше.
2. При каждом повторении выборки (их всего будет  $k$ ) из исходного датасета случайно выбираются элементы с замещением, столько же, сколько было в исходном датасете (для сохранения вариации параметра [81]). В этой процедуре некоторые элементы исходного датасета будут выбраны несколько раз, некоторые — никогда.
3. Для каждой выборки вычисляется нужный нам параметр.
4. Теперь у нас есть  $k$  значений, которые можно использовать для вычисления доверительного интервала или статистического теста.

В А/Б-тестах мы работаем с двумя группами — контрольной и тестовой. По каждой группе нужно сделать свой бутстрэп. Считаем в каждой выборке и группе необходимую метрику. Для каждой выборки считаем разность метрик между группами. Таким образом мы получим  $k$  значений распределения разности в двух группах. Для вычисления значимости А/Б-теста нулевая гипотеза  $H_0$  формулируется так: две выборки одинаковы, поэтому разность между ними равна нулю. Если уровень нашей ошибки первого рода  $\alpha = 0.05$ , тест двусторонний, то нам просто нужно вычислить перцентили (квантили) для отрезка  $[\alpha/2, 100\% - \alpha/2]$ , то есть

[2.5 %, 97.5 %]. Это легко сделать самостоятельно, если отсортировать наш ряд  $k$  значений разницы метрик и определить значения перцентилей на концах. Если 0 будет находиться между двумя этими значениями, то нулевую гипотезу отвергнуть нельзя, если вне — отвергаем.

Вспомним наш пример с двумя резервуарами, у нас есть выборки по 1000 шаров из каждого. Напомню, что в задаче мы должны ответить на вопрос, есть ли разница в среднем диаметре шара между резервуарами. Для процедуры бутстрэпа делаем  $k = 300$  выборок для обеих групп, сразу считаем среднее в каждой выборке и разность между ними. В итоге мы получим 300 чисел. Сортируем по убыванию и выбираем два числа — одно на 2.5-й перцентиле ( $2.5 \% \times 300 = 7.5$  или на 7 позиция), второе на 97.5-м перцентиле ( $97.5 \% \times 300 = 292.5$  или 293-я позиция). Если оба числа оказались или положительными, или отрицательными, значит, разница статистически значима.

Само слово «бутстрэп» произошло от выражения «To pull oneself over a fence by one's bootstraps» (перебраться через ограду, потянув за ремешки на ботинках) — практически то же самое сделал барон Мюнхгаузен, когда вытянул сам себя за волосы из болота. Сейчас бутстрэпом называют такое «самовытягивание», когда мы получаем что-то бесплатное и полезное.

Плюсами бутстрэпа являются: независимость от распределения выборки, отсутствие параметров, кроме количества выборок, возможность легко подсчитать любую метрику. К минусам бутстрэпа относится очень высокая вычислительная требовательность. Создание тысяч выборок требует больших ресурсов. Третья альтернатива для А/Б-тестов — байесовская статистика.

## БАЙЕСОВСКАЯ СТАТИСТИКА

Впервые я познакомился с байесовским подходом для А/Б-тестов, когда прочитал статью Сергея Фельдмана на сайте нашего конкурента Richrelevance про этот тип тестов [82]. Одним из аргументов в пользу байесовских тестов для меня было сравнение двух формулировок итогов А/Б-тестов:

- мы отклоняем нулевую гипотезу, что  $A = B$ , с  $p$ -значением 0.043;
- с 85 %-ной вероятностью А лучше Б на 5 %.

Первая формулировка принадлежит традиционной фишеровской статистике, вторая — байесовской. В статье [82] Сергей обращал внимание на следующие два недостатка  $p$ -значений для работы с гипотезами:

- $P$ -значение — сложная концепция, ее приходится каждый раз объяснять. Что касается меня, то я был хорошо знаком с ней еще в 2002 году. Периодически мне приходится напоминать себе о ней, и тогда я обращаюсь к литературе.
- $P$ -значение использует бинарный подход — мы или оставляем нулевую гипотезу или отвергаем ее, сравнивая  $p$ -значение со значением  $\alpha = 0.05$ .

Классическая математическая статистика (frequentist approach) относится к параметру как к фиксированной неизвестной константе. Байесовская статистика относится к параметру как к вероятностной величине [83]. Это чем-то похоже на разность в подходах классической и квантовой физики. Мне лично больше нравится вероятностный подход байесовской статистики, он выглядит нагляднее и естественнее, чем  $p$ -значение. Меня он так заинтересовал, что я долго искал хорошую и понятную литературу по этой теме. Очень полезной книгой оказалось «Введение в байесовскую статистику» [83] Уильяма Больстарда. Я очень ценю хорошие книги и могу назвать автора Учителем с большой буквы. Больстард очень хорошо выстроил систему вывода формул и доказательств. Я прочитал его книгу от корки до корки, решил почти все задачи в ней и написал

первую версию программной библиотеки для А/Б-тестирования в Retail Rocket. Читая книгу Антонио Рохо о Рональде Фишере [76], я обнаружил интересный факт про байесовскую статистику — оказывается, она широко использовалась для оценки статистической значимости еще в дофишеровскую эпоху. Сторонники традиционного статистического подхода Фишера и сторонники байесовского подхода спорят до сих пор, какой метод лучше.

Сам преподобный Байес написал формулу так:

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)},$$

где:

- $P(A)$  — априорная информация, которая говорит о наших предположениях до проведения эксперимента. Это наши убеждения (может быть, даже интуитивные) до проведения эксперимента.
- $P(A|B)$  — апостериорная вероятность, когда формула суммирует убеждения ( $P(A)$ ) до эксперимента и данные  $B$ , приводя к новым выводам, которые называются апостериорными.
- $P(B|A)$  — (likelihood) вероятность наступления события  $B$  при истинности гипотезы  $A$ .
- $P(B)$  — полная вероятность наступления события  $B$ .

Формула Байеса позволяет «переставить причину и следствие»: по известному факту события вычислить вероятность того, что оно было вызвано данной причиной. Для оценки параметров формулу можно переписать в другом виде:

$$P(\theta|data) = \frac{P(data|\theta) \times P(\theta)}{P(data)}.$$

Мы хотим получить распределение параметра  $\theta$  (например, среднего диаметра шара) после получения данных (data) в нашем эксперименте, при этом до эксперимента мы считаем, что наш параметр подчиняется распределению  $P(\theta)$ . В [83] указаны все выкладки для биномиальных тестов, например, когда мы сравниваем конверсию посетителя в покупателя. Так и для непрерывных нормально распределенных величин, когда мы можем сравнить средний диаметр шаров в наших резервуарах или средний чек в экспериментах на интернет-магазинах. Обе эти задачи относительно легко считаются, так как там используются сопряженные (conjugate) распределения. Для расчета А/Б-теста нужно воспользоваться постериорными формулами и применить сэмплирование, это очень похоже на то, что мы делали в бустрепе.

Важная проблема в байесовской статистике — это выбор априорного суждения, именно к ней имеет претензии классическая статистика. У априорной информации есть свой «вес» (n equal sample size), выраженный в количестве точек данных. В той же книге есть также формулы для оценки «веса» априорных распределений, выраженных в количестве точек данных. Изучая литературу, я вывел для себя следующие правила. Если ничего не знаешь — используй равномерное (uniform) распределение. Если знаешь — то лучше использовать нормальное распределение, где априорное среднее — это ваше предположение, а априорное стандартное отклонение характеризует вашу уверенность в нем. «Вес» вашей уверенности лучше оценить по формулам во «Введении в байесовскую статистику» [83] — тогда вы будете понимать, сколько данных вам понадобится, чтобы изменить точку зрения. Я предпочитаю уверенность делать меньше, чтобы эксперимент быстрее сошелся. Ваши априорные суждения можно представить себе как увеличительное стекло, которое сфокусировано в точке вашей уверенности. Если данные не будут ее подтверждать, то фокус сам сместится ближе к правильному решению. Если подтвердят, то тест сойдется быстрее, так как фокус находился в нужном месте, вы не ошиблись. Например, когда тестируются разные версии рекомендательных алгоритмов, чтобы проверить, улучшилась ли конверсия посе-

тителей в покупателей, вы можете смело взять текущую цифру конверсии (до эксперимента) в качестве априорного среднего. Априорное стандартное отклонение не стоит делать очень узким.

Второй проблемой байесовской статистики является привязка к распределению исходной величины — оно должно быть вам известно. В этом плане бутстрэп лучше, но считается он гораздо дольше, чем байесовский метод.

## А/Б-ТЕСТЫ В РЕАЛЬНОСТИ

Я уже расписал основные плюсы и минусы алгоритмов тестирования. Более подробные советы можно найти в книге «Семь главных правил экспериментов на веб-сайтах» [84]. Хочу предупредить читателя: к сожалению, в интернете много советчиков-теоретиков (и даже целые школы), которые все очень усложняют. Но даже научные статьи порой изобилуют ошибками, особенно если не были опубликованы в научных журналах и не озвучивались на авторитетных научных конференциях. Что уж говорить про посты уважаемых блогеров. Я сторонник простоты и считаю, что в методиках тестирования и анализа можно разобраться самостоятельно. Просто начинать нужно с самого простого — с фишеровской статистики с  $p$ -значениями. Открою секрет — если ваш тест действительно значим и данных в выборках достаточно, то все три метода покажут статистическую значимость. А вот ошибки, с которыми я сталкивался:

- неверная конфигурация теста;
- плохой генератор частот;
- неверный статистический критерий;
- проблема подглядывания;
- отсутствие пост-анализа;
- принятие решения, когда нельзя отвергнуть нулевую гипотезу.

Неверная конфигурация теста — самая частая проблема. Допустим, вы придумали гипотезу, у вас есть метрика, вы написали техническое задание на проведение теста. Если это ML-модель, то вы заранее провели офлайн-тесты — все было хорошо. После реализации теста и выкладки его в рабочую систему его нужно обязательно проверить. Если это сайт, то прощелкать нужные ссылки, проверить, что обеим группам показываются нужные версии страниц. А теперь представим, что тест запущен и в нем есть ошибка. Прошел месяц, пора считать результаты. Наше сознание заставляет нас искать ошибки, если результаты получились плохими, и просто радоваться, если результаты положительные. Но если в тесте была ошибка, то много времени было потрачено впустую и тест придется перезапускать. У меня на практике такое было сплошь и рядом. В результате мы в Retail Rocket разработали целый бизнес-процесс по запуску тестов с инструкциями по проверке. Такие ошибки очень дорого обходятся.

Плохой генератор случайных чисел для разделения всех пользователей на тестовую и контрольную группы тоже может быть проблемой. Надежный способ обнаружения такой проблемы — A/A-тесты. Второй вариант — симуляция. Тогда вам нужно точно повторить код разработчиков, который назначает сегменты, и проверить его работу на старых логах пользователей, то есть произвести имитацию A/B-теста. С такими генераторами часто возникают проблемы, поэтому команда инженеров написала свой вариант и выложила его исходный код в сеть [85].

Неверный критерий тоже может дать свою погрешность. Я бы рекомендовал в целях проверки делать симуляционные тесты выбранного статистического критерия. Это можно делать как с помощью генераторов распределений, так и с помощью уже имеющихся

логов действий пользователя (если есть). Например, сделав два случайных генератора с одинаковыми исходными данными, нужно убедиться, что статистический критерий не показывает значимость. Затем сделать небольшую разницу между генераторами и убедиться, что статистическая значимость появилась. Также рекомендую сделать анализ мощности — сколько данных нужно, чтобы этот критерий показал статистическую значимость на какой-то минимальной для вас важности. Например, вы готовы внедрить новое улучшение только в том случае, если оно улучшает метрику на 1%. Тогда вы делаете два генератора с этой разностью и моделируете работу критерия, чтобы понять, сколько точек данных вам нужно, чтобы заметить эту разницу. Это и будет вашим минимальным объемом выборки данных.

Проблема подглядывания за результатами теста лично мне хорошо знакома. Она возникает, когда тест не набрал еще необходимых данных, а мы уже пытаемся увидеть его результаты. Статистическая значимость теста — это тоже случайная величина, и она «скачет» в первое время после запуска теста. Так происходит и на симуляциях тестов, и в реальных условиях. Я столкнулся с этим впервые в компании Ostrovok.ru, когда А/Б-тесты были выведены на дашборды офисных мониторов. Мне позвонил CEO с вопросом, почему результаты значимости недавно запущенного теста прыгают туда-сюда. Поэтому если вы примете решение в этот момент, признав тест успешным, то совершите ошибку, так как через некоторое время тест «устаканится» и будет показывать отсутствие статистической значимости. Я считаю, что единственный способ решить проблему подглядывания — определить минимально детектируемую разницу в метриках, которая вас устроит. По ней с помощью калькулятора мощности или симуляций вычисляется нужный объем выборки. И именно после достижения нужного объема данных можно смотреть на результат и принимать решения. Здесь вы столкнетесь с дилеммой — если разница слишком мала, то понадобится очень много данных, что плохо для бизнеса. Потому что чтобы получить много данных, придется долго ждать — тратить время. Рекомендую понять, сколько времени вы готовы ждать,

и уже исходя из этого определить минимально детектируемую разницу. Минимальная длительность теста также ограничена бизнес-циклом принятия решений клиентами. Например, если мы знаем, что среднее время принятия решения о покупке составляет три дня, то тест должен идти не меньше двух бизнес-циклов — шесть дней. До этого времени за тестом можно приглядывать, но только с целью обнаружить пропущенные технические ошибки.

Хороший пост-анализ эксперимента гипотезы может помочь понять, что еще можно сделать для улучшения метрики. Как я уже писал, желание обнаружить ошибку, когда тест новой версии продукта провалился, естественно, как и его отсутствие, когда тест выигран. В Retail Rocket мы действительно находили в тестах ошибки не только технического характера, но и идейного. Для этого обычно применяли сводные таблицы и искали проблемы во множестве срезов. Очень приятное чувство, когда находишь ошибку или придумываешь модификацию гипотезы и побеждаешь в следующем тесте. Это можно делать и для положительных тестов, чтобы искать новые идеи по улучшению продукта.

## A/A-ТЕСТЫ

Впервые про A/A-тесты я услышал от Ди Джея Патила — до этого я никогда к ним не прибегал. A/A-тест — это проверка последней мили, всего того, что вы сделали для теста: генератора случайных чисел, схемы сбора данных и выбранного статистического критерия для метрики. Сам тест запускается с реальным делением аудитории на две части, но в контрольной и тестовой группах используется одна и та же версия продукта. В финале вы должны получить сходящийся тест без опровержения нулевой гипотезы, так как версия продукта одна и та же.

Первое, что нужно проверить, — насколько хорошо работает генератор случайных чисел, по значениям которого будет происходить разделение на группы в тесте. Само назначение на группы можно делать двумя способами: через назначение случайного числа и че-

рез хеширование информации об объекте. Когда пользователь посещает сайт, обычно ему в куки пишут его идентификационный номер. Этот номер используется для того, чтобы узнать пользователя при повторном посещении. Для А/В-тестов этот номер хешируется, то есть его превращают из текста в число, далее берут две или три последние цифры для распределения по группам: 00–49 контрольная группа, 50–99 тестовая. Похожий принцип реализован в нашем проекте Retail Rocket Segmentator [85]. В А/А-тесте вы должны получить то же самое распределение, что и в тесте! Если распределение задано пополам, 50/50, то вы его и должны получить на выходе. Даже небольшие расхождения в 3 % в данных теста могут поставить под угрозу весь тест. Если в тесте есть 100 000 пользователей, вы хотите разделить их пополам, а в итоге получается в одной группе 48 000, а в другой 52 000 — это говорит о проблемах в «случайности» разбиения по группам. Эти распределения можно проверить и на симуляциях, когда вам точно известен алгоритм. Но моя практика показывает, что мелкие нюансы разработки, о которых мы не знаем, могут приводить к «сдвигам» распределений. Поэтому я больше доверяю А/А-тестам.

Второе, на что важно обратить внимание, — пользователи должны попадать в группы равномерно, не должно быть смещений по разным срезам пользователей. Например, в тесте участвуют две группы пользователей: юридические и физические лица, первых всего 10 %, а вторых 90 %. После разбиения на группы это соотношение изменилось — в контрольной группе 7 и 93 % соответственно, в тестовой — 12 и 88 %. У этого явления могут быть две причины. Первая — есть закономерность в назначении идентификаторов клиентов, и эти данные используются в назначении групп. Вторая — юридических лиц слишком мало в абсолютных цифрах, и выборка нужна больше. Последнюю причину проще отсечь — нужно попытаться собрать больше данных, если наблюдаемая разница исчезнет, то все в порядке. Если нет — нужно разбираться с процедурой назначения. Обратите внимание на то, что «срезы» лучше сходятся, когда используется разбиение 50/50, а не какое-нибудь экзотическое 90/10. В меньшую группу попадает всего 10 % пользователей.

И третье, что нужно иметь в виду, — на выбранной метрике ваш статистический критерий должен показывать отсутствие статистической значимости, ведь мы показываем пользователю одно и то же. Из опыта скажу — любые бинарные (биномиальные) тесты сходятся намного лучше и быстрее, чем тесты с непрерывной шкалой. Конверсия сайта (процент посетителей, сделавших покупку) сойдется лучше, чем средняя стоимость покупки (средний чек). Причин, с моей точки зрения, две. Первая — низкая вариабельность конверсии (только два значения — купил или нет), вторая — «выбросы» в метриках с непрерывной шкалой. Выброс в тестах — это редкое событие, например, очень дорогая покупка. В какую группу она попадет, там и будет сразу «улучшение» метрики. Согласитесь, такой результат никого не устроит. Поэтому есть определенная практика — срезать небольшой процент данных «сверху» (удаляем самые дорогие заказы), пока A/A-тест не сойдется. Эту практику применяют в Retail Rocket. Теоретически вместо арифметического среднего можно использовать медиану — она более устойчива к выбросам.

## ЕЩЕ НЕСКОЛЬКО СЛОВ О А/Б-ТЕСТАХ

Отдельно хочу рассказать про перекрывающиеся тесты (interleaving tests), множественные тесты и многоруких бандитов (multi-armed bandits). Перекрывающиеся тесты заключаются в смешивании выдачи двух групп в одну, при этом создатель теста знает, когда, кому, какие элементы из групп (тестовой или контрольной) были показаны. Такой способ применяют поисковые системы, когда дорабатывают алгоритмы. Пользователю показывается ответ на его поисковые запросы, где на части позиций (например, четных) показывается контрольный, а на остальных — тестируемый алгоритм. То есть здесь тестируют, разделяя не пользователей, а позиции в выдаче. И часто это делают случайно, сохраняя данные по каждому показу. Мы делали такие тесты для рекомендаций товаров. И как раз наши внутренние A/A-тесты нам помогли увидеть, что у нас была проблема с ге-

нератором случайных чисел, реализация которого отличалась от Retail Rocket Segmentator [85].

Множественные тесты мы тоже используем, но я согласен с авторами книги «Семь главных правил экспериментов на веб-сайтах» [84]: лучше делать тесты проще и сегменты «пожирнее», тогда на выходе получится выше надежность решений, да и приниматься они будут быстрее. Сравним хороший тест с двумя группами 50/50 и тест с четырьмя группами 25/25/25/25 (пропорции разбиения). Первый тест сойдется минимум в два раза быстрее, так как данных в каждом сегменте в два раза больше.

Многорукие бандиты (multi-armed bandits) — очень популярная тема в наши дни. Это направление вышло из обучения с подкреплением (reinforcement learning) [86]. Представьте себе казино: зал с игровыми автоматами, дергая за ручку которых можно получить выигрыш. На некоторых автоматах можно выиграть чаще. Алгоритм тестирования подразумевает использование стратегии «исследуй-эксплуатируй» (explore — exploit). Исследование заключается в том, что мы последовательно дергаем ручки всех автоматов. «Эксплуатация» заключается в том, чтобы дергать чаще ручки тех, которые дают больший выигрыш. Комбинируя эти две стратегии, мы можем найти лучшую комбинацию автоматов с индивидуальной частотой дерганья ручки — менее выгодные дергаются намного реже «счастливых». Например, первый автомат нужно дергать в 10 раз реже, чем второй. Это альтернатива А/Б-тестам, где мы находим только один выигрышный вариант. В этом есть свои плюсы — когда выводим новый алгоритм в бой, то просто добавляем его в список автоматов. Стратегия «исследуй» нужна потому, что среда меняется, и со временем «счастливые» автоматы могут стать «несчастливыми», и наоборот. В долгосрочном

варианте многорукие бандиты работают лучше, чем А/Б-тесты. Но я считаю, что это менее надежная схема. Поэтому рекомендую обращаться к ней только тогда, когда вы научитесь хорошо проводить А/Б-тесты — их можно использовать для калибровки многоруких бандитов.

## ЧТО ДЕЛАТЬ ПЕРЕД А/Б-ТЕСТОМ

Как-то со мной произошла интересная история. Мы в Retail Rocket искали инвестиции, и на встречу нас пригласил Яндекс.Маркет. Один парень из Яндекса спросил, используем ли мы офлайн-тесты, принятые в машинном обучении? Я ответил, что нет. Мои партнеры тогда взяли за голову. Сотрудничества с Яндексом так и не случилось. Почему они нам отказали — не знаю, но могу предположить, что они посчитали меня профаном. Офлайн-тесты у кабинетных исследователей рекомендаций — единственный способ оценить их эффективность. После этого случая я проштудировал всю литературу по теме. Мы вывели все формулы метрик, принятые научным сообществом. В итоге оказалось, что офлайн-тесты слабо коррелировали с нормальными А/Б-тестами, которые я проводил. Поэтому моя изначальная стратегия разработки алгоритмов, основанная только на А/Б-тестах, оказалась верной. Но зачем тогда вообще нужны офлайн-тесты?

Напомню, что такое офлайн-тест: это моделирование А/Б-теста на старых данных. Если у меня есть датасет (лог) действий старых пользователей, то я могу на одной его части обучить алгоритм, а на другой — проверить его эффективность. В идеале он должен хорошо сходиться с настоящим А/Б-тестом. Но почему в нашем случае рекомендаций товаров он расходится? С моей точки зрения, это происходит по двум причинам. Во-первых, товарные рекомендации изменяют действия пользователя, поэтому расчет метрик рекомендаций на старых данных обладает слабой предсказательной способностью. Во-вторых, нашей основной метрикой являются «угаданные» товары в покупках. Цикл принятия решений может

растянуться на дни. Если бы мы предсказывали клик на товар — все было бы намного проще.

В рекомендациях я использую офлайн-тесты в двух случаях. Во-первых, метрики должны измениться в соответствии с идеями, которые заложены в новый алгоритм. Например, если алгоритм улучшает «разнообразие» товаров в рекомендациях, то соответствующая метрика должна увеличиться. Если алгоритм «проталкивает» вверх новинки, то средний «возраст» товаров в рекомендациях должен уменьшиться. Во-вторых, если изменение метрик происходит не так, как мы ожидали, это свидетельствует об ошибке или в идее, или в ее реализации. Такие вещи нужно продумать заранее, чтобы перед просмотром метрик у вас уже были сложившиеся ожидания, иначе можно «проглядеть» ошибку. Например, новый алгоритм улучшит разнообразие товаров (будут показаны больше разных типов товаров), но ухудшит угадываемость. Если по метрикам произошло наоборот — нужно искать проблему.

И наконец, настоятельно рекомендую посмотреть на результат работы собственными глазами. Например, в рекомендациях можно сделать визуальный отчет — выбрать несколько десятков самых популярных и случайных товаров, построить по ним старые и новые рекомендации, вывести в единый отчет с картинками и названиями товаров. Посмотрите на него честно, дайте покритиковать другим. Что там нравится, а что нет? Можете найти товар, для которого новый алгоритм должен был сработать по-другому, стали ли рекомендации лучше? Я с помощью таких отчетов ищу ошибки, которые метрики иногда пропускают. Можно сказать, что они — истина в последней инстанции.

## КОНВЕЙЕР ЭКСПЕРИМЕНТОВ

Теперь мы знаем, что в компании должен быть список гипотез развития, выстроенных в порядке важности, которым управляют менеджеры по развитию бизнеса или продуктологи. Каждый раз

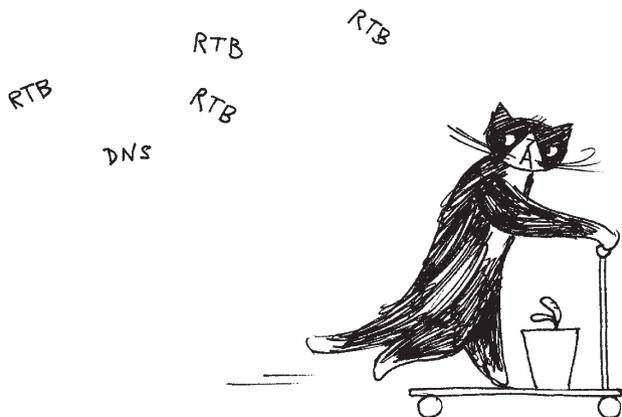
в работу берется первая гипотеза из списка, если нужно — она моделируется и проверяется с помощью офлайн-тестов, последний шаг — тестирование с помощью А/Б-теста, затем происходит пост-анализ результатов, по итогам которого принимается решение — внедряем гипотезу или нет.

Если вы сможете упорядочить этот процесс, то получите самый настоящий конвейер экспериментов. Он действительно похож на промышленный конвейер — гипотеза движется по статусам: принято в работу, моделирование, офлайн-тестирование, онлайн-тестирование, анализ, отклонена, внедрена. Это скорее механический, а не творческий процесс. У меня он был упакован в столбцы Trello, где карточка перемещалась слева направо. Такой подход позволяет масштабировать эксперименты, у него есть свои метрики, например «время между статусами», «взято в работу», «отклонено/внедрено».

В этот момент вы поймете, что время прохождения гипотезы от начала до конца конвейера — очень большое. Особенно время на А/Б-тесты. И скорее всего, сделаете вывод, что было бы неплохо «убивать» неудачные гипотезы до того, как они пройдут хотя бы половину пути [23]. Это очень здравая идея — как можно раньше отвергнуть неудачную гипотезу, чтобы не тратить время и силы на обреченный проект. Именно таким способом мне в Retail Rocket удалось уменьшить среднее время прохождения гипотез через наш конвейер экспериментов с 90 дней до 45.

# 11

## ЭТИКА ДАННЫХ



В наше время данные используются повсеместно — вопрос в том, насколько это безопасно для людей. В сфере программирования существует понятие «жадный алгоритм» — это алгоритм, ориентированный на получение сиюминутной краткосрочной выгоды. Так вот, коммерческие компании в большинстве случаев тоже руководствуются «жадными» алгоритмами и хотят извлекать прибыль из всего, что возможно. В том числе и данных, которые мы явно или неявно оставляем. Здесь я хочу поговорить об этической стороне вопроса. Я думаю, с каждым бывало — вы поговорили с кем-то, например, о стиральной машине с вертикальной загрузкой, а через несколько минут увидели в своей ленте в соцсети рекламу этих самых стиральных машин. Это означает, что кто-то подслушал ваш разговор и воспользовался вашими данными? Конечно, это миф, но сам факт слежения за нами дает пищу для ума. Законно ли это? И если да, то насколько этично?

## КАК ЗА НАМИ СЛЕДЯТ

Наши цифровые следы можно разделить на два вида: явные (explicit) и неявные (implicit). Явные данные — те, которые мы предоставляем сознательно: заполняем анкету на сайте, ставим галочку подтверждения обработки цифровых данных, подписываем заявление или договор. Неявные — это вся информация, которую мы о себе оставляем, так называемый цифровой след: наша геопозиция через мобильную связь и систему распознавания лиц на уличных или транспортных камерах, геопозиция наших автомобилей через распознавание номеров, наши контакты, посещенные страницы в интернете. Неявных данных намного больше, чем явных, а провайдеры и потребители относятся к ним беспечно. При этом если явные данные в нашей стране хоть как-то защищаются законом о персональных данных, то неявные — вне правового поля. В Европе они стали защищенными совсем недавно с введением европейского закона GDPR [107].

Яркий пример того, как используются неявные данные, — когда фото и видео с митингов распознают и арестовывают их участни-

ков, а тех, кто нарушил режим самоизоляции во время пандемии COVID-19, распознают по камерам наблюдения и штрафуют. Всего каких-то пятнадцать лет назад это казалось фантастикой. Эндрю Блн, главный эксперт, который стоит за алгоритмами распознавания по изображениям и которого я не раз упоминал в этой книге, сказал, что это двойственная технология: с одной стороны, она служит добру, с другой — ею легко злоупотреблять.

В наше время технология распознавания лиц уже отлично отработана, нужен всего лишь хороший датасет и доступ к камере. В статье «Мы создали “невероятную” систему распознавания лиц» [91] журналисты собрали небольшой датасет из публично доступных фотографий людей, работающих в районе Bryant Park. Они загрузили их в соответствующий сервис Amazon и буквально за 9 часов получили 2750 совпадений лиц с их датасетом. На все было потрачено всего 60 долларов. В принципе, такой датасет несложно собрать на основе социальных сетей — там есть сопоставление имени и фотографии. Раньше был условно-бесплатный сервис findface.ru (сейчас findface.pro), в который можно загрузить фотографию человека и получить его имя. В качестве датасета была использована социальная сеть «ВКонтакте».

Следующий источник неявных данных — считывание нашей точной геопозиции через смартфоны. Газета New York Times в конце 2019 года создала специальный проект «The Privacy Project» [87], где в серии статей освещаются разные вопросы сбора и использования наших данных. В статье из этой серии «One Nation Tracked» [88] рассказывается про то, как журналисты раздобыли очень большой датасет. В нем находится 50 миллионов геопозиций мобильных телефонов 12 миллионов американцев нескольких крупных городов США, включая Вашингтон, Нью-Йорк, Сан-Франциско

и Лос-Анджелес. Каждая строка датасета включает в себя точное местоположение отдельного смартфона в течение нескольких месяцев 2016–2017 годов. Журналисты сделали исследование датасета и шикарную анимацию этих данных. Вроде бы данные полностью анонимизированные и поэтому безопасны, но, к великому сожалению, это не так. Пол Ом (Paul Ohm), профессор права и исследователь конфиденциальности Джорджтаунского университета, заявил в статье, что попытка представить данные геопозиций как анонимные — совершенно ложное утверждение, которое было опровергнуто множеством исследований. «Действительно точную геолокацию невозможно анонимизировать», он также добавил: «ДНК — это единственная вещь, которую сложнее анонимизировать, чем геолокацию». В большинстве случаев перемещение смартфона между домом и работой позволяет идентифицировать человека. Стал бы еще какой-то другой смартфон перемещаться между вашим домом и работой, кроме вашего? Эта статья подтверждает мое мнение, что использование неявных данных плохо защищается.

До широкого появления смартфонов нас уже «посчитали» дома и на рабочем месте через наши веб-браузеры. Куки (cookies) — небольшой фрагмент данных, который сохраняется веб-сервером на компьютере пользователя в процессе просмотра страниц. Сами куки были придуманы в июне 1994 года сотрудником Netscape Communications Лу Монтулли. Тогда они стали решением проблемы надежной реализации виртуальной корзины покупок. В течение двух лет куки приобрели огромную популярность и стали стандартом. В настоящее время существует несколько видов кук, о которых подробно рассказывается в любых курсах веб-аналитики, меня интересуют только два из них:

- Постоянные (persistent first-party cookies) — постоянные куки, которые хранятся в основном домене просматриваемого сайта из адресной строки браузера. Например, вы зашли на ozon.ru, куки этого типа будут сохраняться в «папку» ozon.ru.
- Сторонние (persistent third party cookies) — постоянные куки, которые хранятся на стороннем домене, не совпадающем с адресной строкой. Обычно они сохраняются через сторон-

ний контент на странице, например через картинки с других доменов. Например, рекламная система doubleclick сохранит свою куку в папку doubleclick, несмотря на то что вы находитесь на сайте ozon.ru.

Первый используется для хранения ваших данных и авторизации, а также для веб-аналитики сайта. Например, когда вы заходите на сайт и авторизуетесь, то за счет куки второй тип — самый спорный. Сторонние куки можно использовать для трекинга вашего перемещения между сайтами, а также в интернет-рекламе и для передачи ваших данных сторонним ресурсам. Рассмотрим это на примере протокола RTB (Real Time Bidding) [89], который используется для мгновенного показа персонализированной рекламы через баннеры и видео. Часть мест на контентных сайтах, а это 2.5 миллиона из 4 миллионов сайтов рунета, выкупается большими компаниями (например, Google или Criteo), которые перепродают их своим клиентам по принципу аукциона. Упрощенная схема проста — кто больше дал ставку за показ, тот и будет показывать свой баннер. Сам аукцион выглядит следующим образом:

1. Вы приходите на веб-страницу, на которой есть такие рекламные блоки.
2. Еще до окончания загрузки страницы рекламная платформа отправляет вашу информацию [89]: IP, ваш ID из куки, ваши интересы (так делает Google), адрес страницы и ее тематику, характеристики вашего устройства и даже геокоординаты. И это не исчерпывающий список информации.
3. После отправки информации в течение пары сотен миллисекунд происходит вычисление ставок у рекламодателей. Серверы рекламодателя используют для этого свои математические модели, которые обрабатывают не только информацию от рекламной платформы, но и внутреннюю информацию, которой

обладают сами. Эту информацию можно накопить, анализируя действия пользователя на сайте, что, например, делает Criteo [90], продавая услуги по возврату потенциальных покупателей — пользователей, которые пришли на сайт, но так ничего и не купили.

4. Рекламная платформа по ответам проводит аукцион и выбирает победителя.
5. Победитель показывает свою рекламу.

Еще один тип рекламы — *programmatic*, когда рекламодатель выкупает у сервиса сегмент аудитории, например у Яндекса или Google, основываясь на внутренних (я выкупаю пользователей, которые покупали у меня подгузники) или внешних (я выкупаю пользователей, которые покупали подгузники в интернете) данных. Показ рекламы осуществляется через те же самые механизмы RTB. Такая реклама менее персонализирована, но бьет по нужным маркетологам сегментам. Например, крупный интернет-магазин может «продать» часть своей аудитории с маленькими детьми производителю подгузников. Тогда этой аудитории будет показана реклама подгузников.

Чтобы использовать внутреннюю информацию всей этой рекламной RTB-машины, требуются сторонние куки рекламодателя, которые нужно сопоставить (*cookie matching*) с куками рекламной сети. Для этого на странице (необязательно рекламной площадки), которую смотрит пользователь, нужно получить два сторонних куки — рекламодателя и рекламной площадки (например, Google). Сами куки получают путем запроса прозрачной, а потому невидимой пользователю картинки размером один пиксель. Обычно это делает небольшой JavaScript-код, который вызывается при просмотре страницы пользователем. Именно в момент сопоставления кук происходит сопоставление ID клиента между рекламодателем и рекламной площадкой. С этого момента у рекламодателя намного больше данных о пользователе. Например, интернет-магазин может передавать ID своего клиента в систему Google, чтобы увидеть его в аукционах RTB, который сам же Google и проводит. Далее по этому ID подтягивается необходимая информация из внутренней базы данных магазина, например, сколько покупок совершил кли-

ент, как давно была сделана последняя из них, какими категориями он интересуется. На основании этой информации магазин делает ставку — сколько он готов заплатить за показ своей рекламы этому клиенту, а также выбирает подходящий рекламный баннер. Если бы этой внутренней информации о покупках не было, экономика RTB-рекламы для магазина была бы значительно хуже.

Тот же механизм сопоставления кук используется при скрытом сборе и продаже данных клиента. Когда вы серфите в интернете — откройте список сетевых запросов в инструментах разработчика в браузере; вы будете удивлены, как много разных систем собирают о вас информацию. Там будут и социальные сети, которые ставят их кнопки и блоки с комментариями к статье, — все это используется для сбора информации. Именно поэтому сторонние куки находятся под ударом со стороны браузеров и законов. Согласно исследованию «The GDPR Is a Cookie Monster» [93], до введения GDPR закона в ЕС, в среднем одна страница оставляла около 80 сторонних кук, то есть порядка 80 сервисов аналитики и рекламных трекеров одновременно получали историю ваших действий в интернете.

Ваш ID в куках какого-либо сервиса — это святое. По этому ID сервис может найти у себя всю историю взаимодействий с вами. Сами куки-файлы — вещь ненадежная, и поэтому они периодически протухают. Они могут вытесняться из-за ограничений браузера или намеренно стираться пользователем. Поэтому сервисы стараются любой ценой повысить их живучесть, дублируя их хранение во всевозможных хранилищах браузера. Если JavaScript-код не находит основную куку, но находит информацию в таких хранилищах, то он восстанавливает ее из хранилища в куки. Следующая ступень — связывание всех ваших устройств в одно, так будет еще больше истории браузинга, а значит, сервис получит более полные данные. Самый простой способ это сделать — через логины: пользователь логинится на основной сайт с компьютера и с мобильного телефона. Так как это одна и та же учетная запись — то куки в основном и мобильном браузере привязываются к учетной записи сайта. А если пользователь намеренно стирает куки? Например, так делают интернет-мошенники всех мастей, которые

хотят получить кредит. Чтобы найти на них управу, созданы специальные сервисы — они используют цифровые отпечатки, которые работают без кук, только на основе той информации, что можно получить из браузера одновременно. Если цифровой отпечаток хорошо спроектирован сервисом, то он с высокой степенью сможет отличить одного пользователя от другого, а значит, и отследить потенциального мошенника. Например, так делает сервис `juicyscore.com`, который собирает сотню характеристик пользователя — от технических до поведенческих, когда даже вычисляется «ритмичность» ввода данных на клавиатуре.

Мы уже говорили про отслеживание и продажу данных геопозиций мобильных телефонов. Для интернет-рекламы в приложениях у смартфона есть свой ID, который является альтернативой кукам в браузерах — Mobile Advertising ID. Этот ID носит название AdID в Google Android и IDFA для устройств Apple. В принципе, это то же самое, что и куки, и у пользователя есть возможность сбросить этот ID, чтобы очистить свою историю. Эти ID недоступны из мобильных браузеров, только в приложениях [94].

Еще один источник данных — провайдеры интернета. До широкого внедрения защищенного протокола `https` они видели всю историю браузинга пользователей, всю информацию в адресной строке браузера. После внедрения этого протокола они видят только домены сайтов, которые вы посещаете, благодаря DNS-запросам [92], с помощью которых привычные нам имена доменов превращаются в IP-адреса, понятные маршрутизаторам. Я не думаю, что провайдеры оказывают существенное влияние на рынок данных, если только речь не идет о просмотре сайтов с очень специфичным контентом.

Голубая мечта любого ритейлера — видеть все данные потенциального покупателя и в нужный момент делать ему триггерное

сообщение, чтобы изменить его поведение. То, как это делается в онлайн, мы рассмотрели выше. В офлайне все сложнее — приходится использовать карты лояльности. Ритейлеры платят за это клиентам скидками — вы уже видите на ценниках две цены, с картой лояльности и без, а клиенты делятся информацией, позволяя связать отдельные покупки воедино. Кстати, когда вы логинитесь на веб-сайтах крупных ритейлеров — ваши онлайн-куки связываются с вашей картой лояльности. Это теоретически дает возможность отправлять вам персонализированную рекламу даже после покупки хлеба в соседнем магазине.

## ХОРОШЕЕ И ПЛОХОЕ ИСПОЛЬЗОВАНИЕ ДАННЫХ

Давайте подумаем, а является ли использование данных проблемой? В мультфильме «Козленок, который считал до десяти» звери обижались на козленка, который хотел их сосчитать. Но в итоге его счетные способности пригодились для подсчета количества пассажиров на корабле. Я всегда говорил, что из-за 5 % «плохих» парней страдают все остальные, так как небольшая группа авантюристов подрывает доверие ко всем.

Возьмем геопозицию мобильного телефона. Если обладать таким датасетом, то можно проектировать транспортное движение, находить лучшие места для магазинов или отслеживать на картах Google загруженность того или иного объекта по часам. Правда, полезное использование данных? Но если речь идет об отслеживании перемещения отдельных лиц без вовлечения контролирующих государственных органов, такое использование данных потенциально опасно. Даже если данные обезличены, теоретически можно выяснить, трек какого именно человека отслеживается.

Теперь о куках в интернет-браузерах. Главная цель их изобретения — обеспечить удобную работу с сайтами, например запомнить вас, чтобы не нужно было каждый раз логиниться. Следующим этапом стала веб-аналитика — каждому пользователю присваивался уникальный ID и записывался в куки. Когда пользователи

кликали на рекламе, приходили, уходили с сайта, затем возвращались и делали покупку, стало возможным считать эффективность рекламы. Затем сторонние куки стали использовать для персонализации сайта, а также персонализированного показа рекламы. Все эти способы применения данных вроде бы безобидны, но куки и ваша персональная информация дает мощные инструменты для «дискриминации по цене». Например, пользователям Mac или последних моделей iPhones некоторые сайты показывают цену выше, а новым посетителям сайта дают скидки побольше (Amazon). В статье «How Online Shopping Makes Suckers of Us All» [95] даже приводятся примеры, когда пользователям из пригородов Бостона показывали более высокую цену, а пользователям из самого Бостона — более низкую. Просто у бостонцев больше альтернатив, а у жителей пригородов меньше. Еще один плохой вариант — передача/продажа ваших данных сторонним сервисам.

Однажды на хакатоне в Ostrovok.ru я предложил идею — собрать дополнительные данные, сохранив у себя показы посетителям сайта персонализированной рекламы от Яндекс.Директ. Для этого пришлось разместить рекламу Яндекс.Директ в «подвале сайта». Потом был написан парсер, который сохранял тексты объявлений вместе с кукой нашего пользователя. Таким образом мы могли бы понять, что еще интересно нашему пользователю. И знаете, тогда все получилось. Подавляющая часть объявлений содержала рекламу других отелей и систем бронирования, что нам было неинтересно, но часть показов содержала рекламу услуг, которые не конкурировали с Ostrovok.ru. После проведения эксперимента и доказательства его жизнеспособности он был прекращен, а данные удалены. Мне лично это доказало, что приватные интересы пользователя можно перехватить, и это не такая сложная техническая задача, как кажется на первый взгляд. Это тоже пример плохого использования данных.

В российском сегменте интернета очень показательная история случилась со «счетчиком» Liveinternet.ru (li.ru). В статье «Почему крупнейшие сайты рунета убирают счетчик Liveinternet?» [96] приводится причина:

«Герман Клименко рассказывает про свой совместный проект с одним из банков. Из рассказа Германа примерно понятно, как работает его Fastscoring: если вы зашли на медицинский сайт, на котором стоит счетчик li.ru, и поискали какое-то лекарство от серьезной болезни или описание самой болезни, то банк, с которым работает Клименко, не выдаст вам кредит — никому не интересен тяжелобольной заемщик».

Меня в свое время это заявление тоже задело за живое — вызвало волну возмущения, и я тогда написал пост на своем сайте. Если задуматься, Герман Клименко выдал эту противоречивую информацию, чтобы пропиарить проект, но ведь есть куча сервисов, которые обладают ровно такой же информацией, просто молчат об этом. Кто гарантирует, что она не используется при определении платежеспособностей заемщика?

Я считаю, что главная причина злоупотребления данными пользователей — это то, что многие сервисы в интернете бесплатные. Практически 100 % контентных проектов и социальных сетей монетизируются за счет показа рекламы. Бесплатного ничего на самом деле нет — сервисам и сайтам необходимо оплачивать серверы, работу сотрудников. Есть, конечно, вариант брать деньги за подписку, как это делает YouTube, — если платишь, то рекламы нет, если не платишь, то есть. Подавляющее большинство пользователей не будут платить. У сервисов с рекламной моделью монетизации (а на рекламе много не заработать) возникает искушение заработать хоть как-то еще на данных пользователей.

Кроме того, утечка данных может произойти при установке на сайт «бесплатных» сервисов, например Google Analytics или Яндекс.Метрика, которые предоставляют аналитические услуги. Но я уверен, что они используют данные клиента не всегда по назначению — ведь должны они на чем-то зарабатывать. Любой

бесплатный сервис — это троянский конь, надо об этом помнить. В какой-то момент еще придет мода на «приватные» аналитические сервисы, но они будут стоить приличных денег, и не каждый сайт их сможет себе позволить.

## ПРОБЛЕМА УТЕЧКИ ДАННЫХ

Я уже писал, что в компании Netflix довольно демократично относятся к доступу к данным, кроме платежной информации клиента. А что вообще является персональными данными клиента — имя, адрес и телефон? Да, это явные персональные данные, по ним можно сопоставить реального человека и его виртуальный ID. С другой стороны, чтобы найти реального человека, достаточно его геопозиции — значит, и ее можно отнести к персональным данным. Другой пример — данные конкурса Netflix Prize убрали из публичного доступа после того как авторы статьи «How To Break Anonymity of the Netflix Prize Dataset» [97] доказали: сопоставление информации из этого датасета с публичными базами данных оценок фильмов раскрыло личность и оценки некоторых людей. Что привело к судебному иску от Федеральной торговой комиссии США (FTC — аналог нашего Роспотребнадзора и ФАСа) к Netflix, удалению датасета из публичного поля и отмене второго конкурса. Для нас в Retail Rocket это стало хорошим уроком — при публикации нашего датасета [33] на kaggle.com мы полностью зашифровали все текстовые описания. Конечно, это сильно затруднило работу исследователей с такими данными, но задача конфиденциальности была полностью решена.

Под утечкой данных я понимаю выход каких-либо персональных данных за периметр безопасности внутри компании. Причем сама утечка может произойти и внутри компании, когда доступ к данным

получают сотрудники, у которых его быть не должно. Посмотрим, как это получается. При проектировании хранилища данных контакты (телефон и email) и имя клиента часто хранятся в виде обычного текста. Чтобы минимизировать риски, эти данные должны быть зашифрованы, например хэш-функцией. Тогда не будет проблем с утечкой персональных данных из хранилища. Если же технической поддержке потребуется узнать причины проблемы по какому-либо из клиентов с известным email, им потребуется захешировать его, и тогда можно будет работать с хранилищем напрямую, используя этот хеш. Конечно, это очень неудобно — например, при составлении списка клиентов для рассылки придется предусмотреть отдельную операцию восстановления email. Но безопасность того стоит.

В нашу эпоху тотальной цифровизации проблема безопасности данных очень актуальна. Современный взломщик больше не ломает двери и не взрывает сейфы — ему нужно получить несанкционированный доступ и незаметно скопировать нужные данные. Причем необязательно это будут адреса, имена, пароли и явки — это могут быть простые тексты. Например, сообщение или письмо, содержащее информацию о человеке, и даже часть аудиозаписи из голосового помощника. Например, в статье «The Dark Side of Our Voice Assistants» [98] поднимается вопрос, что часть аудио, которое не было распознано автоматически, отправляется людям (сервис наподобие Яндекс.Толока) на расшифровку. Записи им вроде бы передаются в анонимизированном виде, но там можно встретить информацию, которую клиенты точно не хотели бы раскрывать. Обычно голосовые помощники активируются только после определенных слов (wake-word) и лишь после этого начинают транслировать ваш голос через интернет для распознавания. Чтобы проверить это, исследователи сделали проект и написали статью «LeakyPick: IoT Audio Spy Detector» [99]. Проект LickyPick

детектировал отправку даже зашифрованных аудиосообщений от голосовых помощников, его авторам удалось обнаружить еще 89 «ошибочно» активирующих слов для колонки Amazon.

Наверное, самый эпичный случай утечки персональных данных, который попадет в историю, — это расследование отравления Алексея Навального. Оно полностью базировалось на конфиденциальных данных перемещений лиц и их протоколов звонков. Что говорит о халатности в этих вопросах, несмотря на принятие закона о персональных данных.

## ЭТИКА ИСПОЛЬЗОВАНИЯ ДАННЫХ

Вопрос этики использования данных я бы разделил на две части: этические нормы и предотвращение утечек данных. Обе части требуют внимания и несут определенные риски, пусть даже не административные, но репутационные.

Экс-аналитик Amazon Андреас Вейгенд (Andreas Weigend) написал книгу «Data for the People», посвященную данным. Вот что он говорит о конфиденциальности данных пользователей:

«...конечная цель заключается в обучении пользователей. Я хочу, чтобы люди понимали, на какие компромиссы они идут... Я хочу, чтобы люди были осведомлены о доступных опциях и их последствиях. Я хочу позволить людям управлять их данными ответственно, включая решения, касающиеся их конфиденциальности. Компании должны уважать эти решения. Я ненавижу, когда компании пытаются манипулировать своими клиентами, вводя их в заблуждение».

Это цитата из интервью Андреаса 2005 года [100] — прошло полтора десятка лет, а воз и ныне там.

По сути между компаниями и их клиентами есть негласный договор на использование данных, пункты которого становятся все яснее со взрослением рынка и образованием клиентов, — я согласен

с Андреасом, что обучение необходимо. На мой взгляд, основы безопасности данных пора преподавать в школах. Законы о персональных данных заставляют компании выводить соглашения о конфиденциальности (privacy policy) на сайтах. Уже небезызвестный нам «Privacy project» провел исследование таких соглашений «Мы прочитали 150 соглашений конфиденциальности. Это непостижимая катастрофа» (We Read 150 Privacy Policies. They Were an Incomprehensible Disaster) [101]. Они проанализировали размер и читабельность соглашений о конфиденциальности 150 сайтов и приложений. Каковы были результаты этого исследования — ясно уже из заголовка статьи. Самое понятное соглашение оказалось у BBC — чтобы разобраться с ним, достаточно школьного образования и 15 минут времени. А вот соглашение, которое предлагает Airbnb, требует юридического образования и 35 минут на чтение. Сами соглашения стали читабельнее после введения закона GDPR в ЕС, но в целом «написаны юристами для юристов. Они не создавались как инструмент для пользователя» [101]. И часто в них есть техническая информация про сбор данных, но нет ни слова о передаче их третьей стороне, что напрямую затрагивает пользователя. Рекомендую ознакомиться с этой статьей, там отличная инфографика и анимация.

У аналитиков данных и ML-специалистов тоже есть свои негласные этические нормы. Самое первое из них — никогда не использовать данные для получения личной пользы или удовлетворения собственного любопытства. Единственная ситуация, в которой можно работать с конкретными персоналиями, — это техническая поддержка. Мы (аналитики данных) имеем довольно серьезный доступ к конфиденциальным данным клиентов, не составляет большого труда найти какую-то персону и поднять всю ее историю в данных. Есть такой термин «LOVEINT» (love intelligence) — он

возник, когда разразился скандал с сотрудниками NSA (Агентства Национальной Безопасности США) в 2013 году [102]. Выяснилось, что они использовали шпионские технологии для слежения за своими близкими или бывшими близкими. Так вот, никогда и ни при каких обстоятельствах так делать нельзя. Любые запреты и ограничения доступа к данным можно обойти, поэтому хорошо, когда есть внутренний этический запрет.

Кроме того, недопустимо халатное отношение к персональным данным. Это значит — нельзя передавать и хранить их внутри компании вне периметра безопасности, например, через публичные email-сервисы. Соблюдение этого правила сильно уменьшит вероятность непреднамеренной утечки.

И, конечно, не следует вносить в модели спорные фичи, такие как пол, расу, признаки тяжелых болезней, возраст и другие. В России эта повестка пока не полыхает так, как в Америке, но после Black Lives Matter и Time's Up специалисты, занимающиеся искусственным интеллектом, не могут ее обойти. На одном из семинаров Эндрю Ына рабочие группы тоже пытались сформулировать этические нормы [103]. Потому что причиной смещенности датасета часто оказываются предрассудки в обществе — интересный материал на эту тему был опубликован в Harvard Business Review «What Do We Do About the Biases in AI?» [104]. Там описан такой случай: в 1988 году Британская комиссия по расовому равенству обнаружила дискриминацию в высшем медицинском учебном заведении. Компьютерная программа, которая определяла, каких заявителей пригласить на интервью, пессимизировала женщин и людей с неевропейскими именами. При этом ее точность после обучения достигала 90–95 %. Проблема не в программе, а в датасете, который создали люди, раньше принимавшие решения, а машина просто обнаружила и повторила закономерность. Бо-

роться с этим смещением не так просто, но возможно. Например, добиваться диверсификации в собственной команде. В российских реалиях как минимум создавать равные условия при найме мужчинам и женщинам.

## КАК ЗАЩИЩАЮТ ПОЛЬЗОВАТЕЛЬСКИЕ ДАННЫЕ

Ситуация с конфиденциальностью становится лучше — и вот самые значимые события, которые повлияли на рынок защиты персональных данных, включая небезобидные куки.

Во-первых, появились блокировщики рекламы в настольных и мобильных браузерах. Они блокируют не только показ рекламы, но и часть сторонних кук (*third party cookies*), которые используются при передаче данных третьим лицам. Deloitte провели исследование [105] с говорящим названием «Уже почти половина россиян стали блокировать интернет-рекламу», согласно которому рекламу блокируют 44 % процента российских интернет-пользователей. В мобильных браузерах блокировать ее сложнее, поэтому распространенность блокировщиков там меньше. Я хочу только напомнить, что большинство контентных ресурсов живут за счет рекламы, и использование блокировщиков бьет по их доходам. На самом деле не так много сайтов использовали агрессивную рекламу в виде всплывающих окон, но эти агрессивные сайты повлияли на всю индустрию, потому что такая практика привела к массовой установке блокировщиков. Поисковые системы стали из-за этого терять доходы — и начали пессимизировать выдачу (опускать вниз списка) сайты с агрессивной рекламой, чтобы она меньше раздражала пользователей и те не ставили бы блокировщики.

Следующим шагом было введение в браузеры блокировки сторонних кук по умолчанию. В браузере от Apple уже сразу включен пункт «Prevent cross-site tracking». Бюро интерактивной рекламы — международная организация, которая разрабатывает стандарты в рекламной сфере и обеспечивает легальную поддержку индустрии онлайн-рекламы, — провело исследование «IAB Europe

Guide to the Post Third-Party Cookie Era» [106], согласно которому 30 % показов рекламы происходят через браузеры Safari и Firefox, в которых сторонние куки уже блокируются по умолчанию. Еще в 65 % показов будут заблокированы с Google Chrome, когда Google решится это сделать. В январе 2020 года Google объявил, что в течение двух лет прекратит поддержку сторонних кук. Но компания тянет с решением, потому что, в отличие от Apple (Safari) и Mozilla (Firefox), она зарабатывает деньги на рекламе, в том числе RTB, которой просто необходимы сторонние куки для обогащения информацией.

Это не касается рекламы в приложениях — Mobile Advertising ID по-прежнему будет работать. Но в любом случае есть возможность сбросить эти мобильные куки в начальное состояние через настройки системы. Еще я заметил, что с каждым обновлением мобильной операционной системы IOS приложениям дается все меньше и меньше прав на доступ по умолчанию к данным клиента. Например, сейчас можно изменить доступ к геоинформации (Никогда, Спросить в следующий раз, При использовании приложения) или отдельным фотографиям.

Хочу также обратить внимание на один технический нюанс — в современных системах полностью удалить данные очень сложно. Дело в том, что кроме основных рабочих баз данных, где ваши данные можно удалить по вашему ID, есть еще более низкоуровневые системы, например хранилище Nadoor и системы резервного копирования. Они оптимизированы для сохранения данных, но никак не для редактирования. Это делает удаление данных конкретного пользователя настолько сложным, что никто этим заниматься не будет. А если ваши данные все-таки удалили по вашему требованию, у компании остается возможность их восстановить, если она вдруг этого захочет.

Но самые интересные вещи произошли в сфере законодательства. Российские сайты должны публиковать информацию согласно Федеральному закону от 27.07.2006. № 152-ФЗ «О персональных данных». Сайты ЕС подчиняются GDPR (General Data Protection Regulation), который вступил в силу 25 мая 2018 года. Задачей закона является регулирование процесса обработки персональных данных и ее прозрачность для клиента. В самом законе есть пункт о том, что документ о защите данных на сайте компании должен быть написан в лаконичной, прозрачной и понятной форме, использующей ясный и понятный язык. На основе европейского закона были приняты аналогичные правила в других странах — например, закон CCPA (The California Consumer Privacy Act), который защищал резидентов Калифорнии. В статье «GDPR vs ФЗ-152» [107] сделано сравнение законов. Для меня было открытием, что куки и ip-адреса, согласно GDPR, являются персональными данными — российский закон так не считает. Я убежден, что это огромное ключевое различие в трактовке персональных данных, и мне после всех этих историй с торговлей данными подход GDPR кажется более правильным.

Сравним эффективность законов о защите персональных данных с точки зрения пользователя. Я открыл сайты «Декатлона» в британской юрисдикции ([decatlon.co.uk](http://decatlon.co.uk)) и российской ([decatlon.ru](http://decatlon.ru)). Российский сайт имеет ссылку «Защита данных» в подвале. По ссылке открывается скучный текст о том, как они используют данные, — без лишних подробностей. Внизу текста указан обычный офлайн-почтовый адрес, куда клиент может отправить письмо, если хочет, чтобы его персональные данные были удалены. Британская версия, которая пока подчиняется GDPR (Британия выходит из ЕС), выглядит намного круче. Во-первых, сразу при первом открытии сайта появляется перекрывающее страницу окно, где есть возможность согласиться с правилами обработки данных или пойти по ссылке и выставить галочками точно, какому сервису вы даете разрешение, а какому нет. Выглядит все, как в настройках безопасности приложений в смартфонах, а сами тексты намного понятнее, чем для ФЗ-152. Кстати, версия сайта в Нидерландах содержит кнопку отключения сервиса Retail Rocket — они являются

нашими клиентами, и у них все сделано по закону GDPR. А вот на сайтах Amazon и Target тексты соглашений ничем не лучше тех, которые я видел на российских сайтах, — об этом тоже писали исследователи Privacy Project [101]. Также я нажал ссылку на сайте Target.com, чтобы запретить продажу своих данных согласно Калифорнийскому закону CCPA. Сайт предложил заполнить большую форму с указанием, что запрет сработает, только если клиент является резидентом Калифорнии. Авторы статьи «This Article Is Spying on You» [108] провели эксперимент с публикацией на сайте The Times, посвященной теме влияния абортотерапии на фертильность. Если зайти на сайт из США, то данные отправятся 50 сервисам, если с IP-адресов ЕС — то всего шестнадцати. Число сторонних кук на версии сайта для США равно 100, а для ЕС — 28. Эффект GDPR налицо — данные стали намного меньше уходить налево. Также исследователи эффекта закона GDPR [93] произвели сравнение числа сторонних кук до введения закона и после и обнаружили, что их число упало в среднем на 22 %.

Я считаю, что преимущество GDPR относительно остальных законов очевидно, и надеюсь, что российские законы также подтянутся до его уровня.

# 12

## ЗАДАЧИ И СТАРТАПЫ



В этой главе я расскажу о проблемах, которые стоят перед современными компаниями в e-commerce, а также о способах их решения. Думаю, вам также пригодится мой опыт создания Retail Rocket в качестве одного из учредителей.

## ВЕБ-АНАЛИТИКА В РЕКЛАМЕ

Веб-аналитика — это предметная область, которая изучает поведение людей в интернете. Веб-аналитику я делю на две части: оценка эффективности рекламы и анализ взаимодействия пользователей с сайтом.

Начну с анализа эффективности интернет-рекламы. Есть крылатая фраза Джона Ванамейкера (1838–1922) — легендарного американского коммерсанта, революционера в торговле (он открыл первый универсам и первым применил ценники) и отца современной рекламы: «Я знаю, что половина моего рекламного бюджета расходуется впустую, вот только не знаю, какая именно».

Раньше я искренне считал, что именно интернет-реклама положит конец пустому расходованию денег и станет намного эффективнее рекламы на телевидении и в печати. Например, вы показали в телеэфире ролик — как теперь измерить его эффективность? Есть несколько способов: от изменения графика продаж в момент показа рекламы до опроса аудитории с целью узнать, насколько повысилась осведомленность. Для печатной рекламы, помимо этих методов, существует еще один, более точный — использование промокодов на скидку или подарок. По числу введенных промокодов можно определить условную эффективность рекламы.

С интернет-рекламой все стало проще. Все ссылки помечаются специальными тегами, например utm-метками. Обратите на них внимание, когда кликаете на рекламе. После перехода на сайт на компьютер пользователя записываются так называемые куки-файлы (cookies), по которым сайт узнает этого посетителя, когда он туда вернется. С помощью этого механизма можно отследить

покупки пользователя, сделанные через несколько дней или неделю после перехода с рекламы. Не правда ли, что это выглядит намного точнее, чем при традиционной офлайн-рекламе? Именно так я наивно и считал в далеком 2005 году, когда только начал заниматься оценкой эффективности рекламы в онлайн. Тогда не было такого количества рекламы и перекрестных переходов, поэтому ее влияние отслеживалось хорошо.

В наши дни рекламы стало не просто много, а очень много, и пользователь перед покупкой порой делает несколько переходов с разных источников рекламы. Вначале он может искать что-то в поисковике, перейти на сайт интернет-магазина с поисковой рекламы, сделать в магазине пару кликов, уйти с сайта. Через несколько дней он может вернуться на сайт с так называемой ретаргетинговой рекламой (например, этим занимается уже известная нам Criteo), зарегистрироваться в магазине, бросить товар в корзину и уйти с сайта. Скорее всего, через несколько часов или даже минут он (или она) получит письмо — «вы забыли оформить заказ, ваш товар уже в корзине». Пользователь возвращается на сайт магазина из письма и совершает заказ. Внимание, вопрос: благодаря какой рекламе пользователь сделал покупку? Кажется очевидным, что если бы не его первый переход из поисковой системы, магазин точно не получил бы заказ. Но как быть с остальными двумя — ретаргетинговой рекламой и письмом с просьбой завершить заказ? Действительно ли они повлияли на результат в этой цепочке переходов?

В стандартных инструментах веб-аналитики обычно выигрывает последний клик (last click attribution). В нашем примере это письмо о забытом заказе, но его бы не было без первых двух переходов. Это называется проблемой реатрибуции — когда разные источники рекламы «бьются» между собой за заказ. Как посчитать эффективность рекламы, если было несколько разных переходов с источников рекламы перед целевым действием, например заказом? Чтобы ответить на этот вопрос наверняка, нужно провести А/Б-тест — половине людей показывать ретаргетинг, другой — нет. Половине людей отправить письмо, другой — нет. А если эффективность ретаргетинга и email зависят друг от друга? В теории

можно было бы сделать сложный многофакторный тест — но на практике это невыполнимо. А/Б-тесты такого типа в интернет-рекламе — очень сложные и достаточно дорогие, так как приходится отключать часть интернет-рекламы, а это падение выручки. Многие великие умы бьются над созданием альтернативных способов расчета эффективности рекламы. Возможно, рано или поздно они выработают систему, в основе которой будет лежать некий вероятностный подход: например, давать больший вес начальным переходам. Чтобы построить такую модель, нужно сделать много А/Б-тестов, которые обойдутся очень дорого, но при этом все равно получить некую частную, а не общую модель, которую невозможно распространить на всю индустрию.

В рекламной веб-аналитике вы еще встретитесь с двумя терминами — сквозная аналитика и когортный анализ. Под сквозной аналитикой обычно понимают работу с клиентом на индивидуальном уровне: от показа рекламы до отгрузки заказа отслеживания последующих действий заказчика. Это делается с помощью уникальных идентификаторов клиента (ID), с помощью которых его «ведут» в разных системах, от рекламных до логистических. Благодаря этому можно считать затраты на рекламу и обработку заказов с точностью вплоть до индивидуального клиента, пусть и с некоторым приближением.

Когорта в маркетинге — это группа людей, которые совершили определенное действие в заданный промежуток времени. Под когортным анализом подразумевается отслеживание таких однородных групп клиентов. Самое главное его назначение — расчет LTV (Life Time Value), количества денег, которые приносит клиент за определенный промежуток времени. Предположим, вы определили, что этот период будет составлять три месяца, и решили считать LTV первого числа каждого месяца (рис. 12.1). Каждый расчетный месяц аналитик будет «смотреть» на клиентов, которые совершили свой первый заказ или регистрацию три месяца назад, и считать их покупки за эти три месяца, потом делить это число на число клиентов. Для такого расчета нельзя использовать клиентов, которые совершили первое действие четыре или два месяца назад.

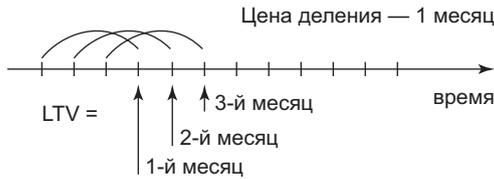


Рис. 12.1. Расчет LTV

## ВНУТРЕННЯЯ ВЕБ-АНАЛИТИКА

Внутренней веб-аналитике сайта уделяется не так много внимания, как рекламной — на рекламу тратится куда больше денег, чем на сайт, поэтому руководство хочет знать, насколько эффективно они потрачены. А ведь действия посетителя на сайте, которые как раз являются объектом внутренней аналитики, очень важны. В этот анализ входят: воронка продаж, анализ заполнения форм и анкет, мерчандайзинг, функционализм сайта, карты кликов, запись действий пользователя (например, Яндекс.Вебвизор). Используя эти инструменты, можно гораздо лучше понимать свою аудиторию.

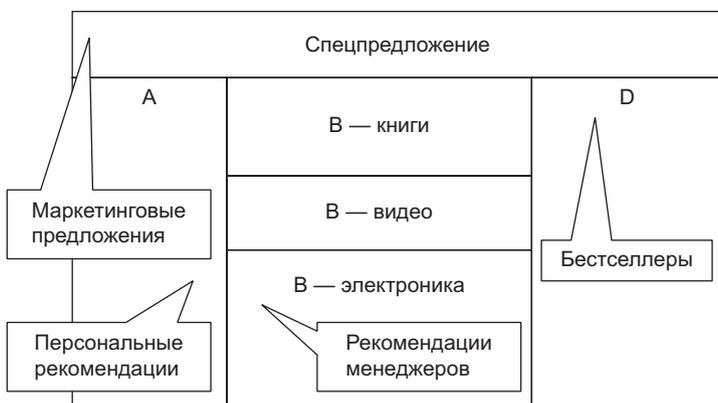
Воронка продаж выглядит почти как обычная воронка — посетитель сайта «проваливается» по ней, пока не достигнет целевого действия, например заказа. В среднестатистическом интернет-магазине конверсия посетителя в заказ составляет обычно один процент, то есть лишь каждый сотый посетитель доходит до дна воронки продаж и совершает покупку. Улучшению этого показателя уделяется очень много времени, ведь если растет конверсия сайта, то вы зарабатываете больше при тех же затратах на рекламу. Хотя реклама рекламе рознь: можно гнать на сайт небольшой поток почти готовых покупателей или большую толпу посетителей, подавляющее большинство которых уйдут с сайта сразу. В первом случае конверсия может быть высокой, во втором низкой, но и стоить первый вариант будет дороже. Поэтому я не сторонник «меряться» конверсиями, более важный показатель — средняя стоимость привлеченного заказа (Cost per Order). Он позволяет объективно сравнить экономики двух интернет-магазинов в первом

приближении. Воронку продаж можно также рассматривать как последовательность микрошагов из целевых действий:

1. Сделал хотя бы один клик после перехода (non-bounced visitor).
2. Добавил товар в корзину.
3. Нажал кнопку «оформить заказ» (checkout).
4. Оформил заказ.

Оптимизируя каждый шаг, можно увеличить число посетителей, которые доходят до конца воронки.

Анализ мерчандайзинга — это самое лучшее, что я узнал о внутренней веб-аналитике, когда изучал систему Omniture (ныне Adobe) SiteCatalyst. Анализ мерчандайзинга — это способ оценки эффективности виртуальных полок интернет-магазина. Сайт интернет-магазина включает в себя несколько типов страниц: главная, поиск, страница категории товаров, страница информации о товаре, корзина, шаги заказа и личный кабинет пользователя. На каждом типе страниц размещаются блоки товаров (рис. 12.2) — например, горизонтальная линия из пяти ротлируемых товаров или большой блок списка товаров на страницах категории. В любом товарном



**Рис. 12.2.** Пример мерчандайзинга сайта интернет-магазина

блоке товар подается со следующими атрибутами: картинка, сниппет с небольшой информацией, цена, название товара, кнопка добавления в корзину или быстрого заказа. Что можно делать с дизайном подачи товара в блоке? Можно увеличить картинку, убрать какие-то элементы. А вот посчитать, что изменилось в метриках, можно с помощью анализа мерчандайзинга, где аналогом обычной полки в магазине будет блок товаров в интернет-магазине.

Сам анализ работает следующим образом: все ссылки на товары (картинки, названия, кнопка добавления в корзину) помечаются специальными невидимыми тегами, где могут быть указаны тип страницы (главная, поиск и другие), название блока (горизонтальный, листинг), тип ссылки (картинка, название, кнопка добавления в корзину). Для каждого клика на таком блоке система запоминает, на каком товаре в каком блоке какой пользователь кликнул. Затем система в течение заранее установленного времени (например, 24 часа) следит за пользователем, что он будет делать с этим товаром после клика. Если пользователь добавил его в корзину или заказал, то эта метрика будет приписана к тому невидимому тегу, который был при клике. На выходе вы можете получить следующую статистику (табл. 12.1).

**Таблица 12.1.** Расчет эффективности мерчандайзинга

Название тега	Клики	Добавления в корзину	Заказы
Главная / Рекомендации первой строки / Картинка товара	1000	350	20
Поиск / Листинг / Название товара	15 000	3 000	500
Страница товара / Рекомендации аналогов / Картинка товара	50 000	15 000	2000

Обычно я выгружаю такую статистику в Excel, разбиваю тег на три поля (тип страницы, тип блока и тип ссылки) и получаю возможность легко решать следующий круг задач:

- Каков вклад в продажи каждого типа страницы? Например, 15 лет назад я вычислил, что страница поиска Ozon.ru дает половину от всех добавлений в корзину на сайте.
- Каков вклад рекомендательных блоков в продажи? На момент моего ухода из Ozon.ru система рекомендаций обеспечивала около 38 % всех добавлений в корзину.
- Откуда чаще покупают — после клика на картинке товара или на его названии? Тогда я выяснил, что чаще кликают на изображении, но названия товаров дают больше продаж.

Когда аналитик может это считать, у компании появляется неограниченное поле для экспериментов «а что, если»: увеличить картинки товаров, убрать картинки из поиска, поменять местами блоки товаров, изменить алгоритм рекомендаций в блоке товаров. Если у вас есть метрики мерчандайзинга, появляется гораздо больше возможностей для модификации сайта.

Напишу про некоторые нюансы этого типа анализа. Во-первых, там есть такая же проблема реатрибуции тегов, как и в рекламе: пользователь через поиск на сайте кликнул на товаре, через некоторое время он кликнул на том же товаре в блоке рекомендаций и купил его. К чему атрибуцировать товар — к странице поиска или блоку рекомендаций на сайте? Есть две стратегии: выиграл первый и выиграл последний. В первом случае этот заказ получит страница поиска, во втором — блок рекомендаций. Однозначного ответа на вопрос, какая стратегия лучше, нет. Я лично предпочитаю вариант «выиграл первый». Во-вторых, вычисления для анализа мерчандайзинга намного более затратны по сравнению с анализом рекламы. Из-за этого Omniture SiteCatalyst отказался поднимать время слежения за действиями пользователя с 24 часов до 7 дней, и мне пришлось пользоваться метрикой добавления в корзину, а не заказа, потому что в течение двадцати четырех часов после первого визита на сайт человек, как правило, не делает заказ, но успевает положить товар в корзину. Обращайте внимание, как вендоры веб-аналитики работают с мерчандайзингом: у Яндекс.Метрики такого нет и не планируется, у Google Analytics есть Enhanced

Ecommerce, у Adobe Analytics есть анализ мерчандайзинга [114]. Я изучал документацию по внедрению двух последних систем и могу сказать, что в Adobe Analytics это сделано намного лучше, чем в Google Analytics. Я сам заимствовал эту идею и написал свой алгоритм расчета, который используется и по сей день компанией Retail Rocket для вычисления эффективности рекомендаций на сайтах клиентов.

Карта кликов на странице — интересный инструмент, но ее нужно очень серьезно настраивать, если работа идет с динамическими блоками, когда товары там ротируются. Я обычно старался заменять ее на анализ мерчандайзинга, а саму карту рисовать в редакторе. Это позволяло мне сделать усредненную карту кликов для страницы товара, когда самих товаров около 500 тысяч. Никакая карта кликов сама по себе с этим не справится, а анализ мерчандайзинга может.

Еще один полезный инструмент — «видеозапись» действий пользователя. Его умеет делать Яндекс.Метрика, сам инструмент называется вебвизор. Он сохраняет все действия небольшой части пользователей, включая движения мыши. Потом вы можете просмотреть такие записи в интерфейсе программы. Это напомнило мне книгу Пако Андерхилла «Как заставить их покупать». В этой книге автор рассказывает, как он расставляет огромное количество камер в магазинах клиентов, сутками смотрит видеозаписи, дает рекомендации, как изменить пространство магазина, чтобы больший процент посетителей совершили покупку. Точно так же можно использовать и вебвизор. К сожалению, инструмент недооценен либо по причине слабой информированности, либо из-за неудобства в использовании. Этот способ — хорошая альтернатива дорогим системам юзабилити, например трекерам глаз.

По веб-аналитике я могу дать важный совет — читайте не инструкции для пользователя, а инструкции по внедрению. Именно там зашита вся суть — когда вы их прочитаете, у вас будут появляться свои мысли о том, как использовать какие-то хитрые фишки системы веб-аналитики, с которой вы работаете. Второй совет — думайте фундаментально. Мне очень понравился подход функционализма [109] в веб-аналитике. Даже на самых простых инструментах, используя подобный подход, можно принимать более системные решения.

## МАРКЕТИНГ НА ОСНОВЕ БАЗ ДАННЫХ

Электронную почту изобрели давно, мы все ею до сих пор пользуемся. До эпохи интернета была развита торговля по почтовым каталогам, которая зародилась в США в XIX веке. После этого появился директ-маркетинг, задачей которого является продажа через прямые каналы коммуникаций (почта, электронная почта, телефон). Это очень отличается от обычной рекламы, которая «бьет по площадям». В директ-маркетинге идет работа с клиентом на индивидуальном уровне. Работу можно разделить на два типа: продажа уже накопленной базе клиентов и рекрутирование новых. Первый тип — это маркетинг на основе баз данных.

Представьте, что у вас есть некоторая существующая база данных клиентов: там есть контакты, факты отправок корреспонденции и факты заказов, эти люди регулярно получают каталоги товаров по обычной почте. Вам поручено сделать очередную рассылку, чтобы получить от клиентов дополнительные заказы. Самый простой подход — отправить всем клиентам одно и то же, например предложение со скидкой. Это довольно дорогое удовольствие — каждое письмо стоит денег, да и скидка уменьшит маржинальность компании. Также нужно учесть, что часть клиентов и так сделает заказ, даже без скидки. Тогда возникает идея разбить клиентов на группы, те из них, кто сделает заказ без промоскидки, не получит ничего — в крайнем случае, простое предложение без скидки. А те,

кто менее склонен к покупке, получают скидку. Клиенты, которые совсем давно не делали заказ, не получают ничего.

Чтобы воспроизвести такую схему, понадобится механизм скоринга клиентов. Для нашей маркетинговой акции необходимо рассчитать вероятность покупки у каждого клиента в нашей базе данных. Затем, используя эту шкалу, разбить их на несколько групп, у каждой из них будет свое промопредложение. Например:

- Группа лучших клиентов (вероятность от 70 %) получит скидку 3 %.
- Группа хороших клиентов (вероятность от 40 до 70 %) получит скидку 10 %.
- Группа плохо покупающих (вероятность от 20 до 40 %) получит купон на 500 рублей.
- Группа давно не покупавших (вероятность ниже 20 %) не получит ничего.

Такую модель можно собрать на обычной логистической регрессии, используя RFM-фичи, о которых я уже писал в главе 9. На данном этапе мы разбиваем всех клиентов по группам и начинаем делать план теста.

Наша модель может ошибаться, несмотря на хорошие метрики, полученные на уже существующих данных. Чтобы это проверить, нужно провести А/Б-тест с контрольной группой. Для этого в каждой группе клиентов случайно выберем 20 % из них и разобьем получившуюся группу пополам. Одна часть группы получит промопредложение — это будет тестовая группа. Вторая получит простое письмо или не получит ничего, в зависимости от маркетинговой акции — это будет контрольная группа. Так нужно сделать для всех групп, может быть, за исключением давно не покупавших. После подготовки плана, верстки и печати писем производится их отправка клиентам. Аналитики выжидают некоторое время — месяц или два — и считают результаты: продажи и все расходы, которые можно учесть. Для каждой группы должна получиться следующая таблица (табл. 12.2).

Таблица 12.2. Расчет прибыли от тестовой рассылки

Группа	Отправки		Покупатели		Прибыль (gross profit)	
	1	0	1	0	1	0
Лучшие	1000	1000	120	100	110 000	100 000
Хорошие	3000	3000	170	150	140 000	130 000
Остальные	5000	5000	180	150	110 000	120 000

1 — тестовая; 0 — контрольная.

В итоге по лучшим клиентам стоит сделать скидку, по хорошим тоже, а вот по плохо покупающим — нет. С ними произошло следующее — купон на 500 рублей привлек много покупателей, но их средний чек покупки стал меньше, чем в контрольной группе. Это не окупило маркетинговую акцию, результат (прибыль) в контрольной группе оказался лучше. Из моего опыта: клиенты лучше реагируют на купон с фиксированной суммой, чем на скидку, но средний чек у «купонщиков» всегда ниже «скидочников». Эта тема довольно неплохо и понятно изложена в моей любимой книге Джима Ново «Drilling Down» [71], другой источник — книга «Маркетинг на основе баз данных» [110] Артура Хьюза. В реальной жизни дизайн может быть сложнее — для группы тестируют сразу несколько типов скидок, купонов и подарков. Я делал такие акции для давно не покупающих клиентов в Ozon.ru, и мне удавалось получать не одну сотню тысяч долларов дополнительных продаж.

С течением времени директ-маркетинг стал ассоциироваться с рассылкой спама, хотя это не та область, где бьют по большим площадям. Сейчас его, конечно, стало меньше в связи с переходом на электронную почту. Экономически доставить обычное письмо

стоит намного дороже, чем электронное. Что привело к взрывному росту количества сообщений в наших электронных почтовых ящиках. Но давайте порассуждаем: если заваливать вас потоком сообщений — вы будете больше покупать? В краткосрочном периоде компания получает рост продаж, но в долгосрочном — происходит такое явление, как «эрозия» базы данных клиентов. Компания теряет кредит доверия покупателей — постоянная рассылка раздражает, и недовольные получатели либо отписываются от нее, либо (что чаще) не выдерживают и нажимают кнопку «в спам». А это влияет на репутацию компании и ведет к тому, что в спаме окажутся и последующие рассылки. Но чаще всего люди просто перестают реагировать на сообщения. В Ozon.ru я тестировал оптимальную плотность и лучший день для рассылок. Какие-то клиенты получали письма каждый день, какие-то — каждую неделю. В итоге мы выяснили, что лучше рассылку отправлять раз в неделю по вторникам. Там же, в Ozon.ru, мы каждое утро делали рассылку с книжными новинками, пока однажды не решили, что это слишком часто, и стали отправлять клиентам подборки книг один раз в неделю. Наши адресаты начали писать в поддержку — почему письма стали приходить реже? Один клиент написал, что у него на работе нашу рассылку привыкли читать каждый день всем офисом. После этого мы добавили в подписку опцию: клиент сам решал, как часто он хочет получать письма — раз в неделю или каждый день. Одни хотят получать рассылку часто, другие — редко, и эту информацию желательно учитывать — например, снижать информационное давление на тех клиентов, которые не совершали действий в результате прошлых рассылок. Каждое новое электронное письмо, которое клиент даже не открывает, ведет к тому, что чаша его терпения однажды переполнится.

Следующей ступенью эволюции email-маркетинга стали триггерные письма и цепочки взаимодействия с клиентом на их основе. Триггерное письмо — это реакция на какое-либо действие или бездействие клиента. Примеры триггерных писем:

- Отправляем письмо клиенту, который зарегистрировался на сайте, но не сделал заказ в течение трех дней. При этом из ста-

истики у нас есть информация, что с каждым днем вероятность не сделать заказ повышается на 10 %.

- Отправляем письмо клиенту, который положил товар в корзину, но не сделал никаких действий в течение часа.
- Отправляем письмо со ссылками на сверла клиенту, который месяц назад купил дрель. Из статистики мы знаем, что 50 % клиентов покупают сверла в течение 30 дней после покупки дрели. Сверла — расходный материал.

Таких вариантов может быть много, поэтому их тестируют и собирают в единую цепочку взаимодействия с клиентами. Впервые я познакомился с такими схемами еще в Ozon.ru, тогда их предложили некие французские консультанты тогдашнего инвестора PPE Group. Конструктор такой системы рассылки реализован и в Retail Rocket с использованием данных взаимодействия клиентов с сайтом.

## СТАРТАПЫ

Некоторое время назад тема стартапов стала популярной; мой собственный карьерный путь связан как раз с работой в стартапах, а не в глобальных корпорациях. Я стоял у истоков одного из них как сооснователь и считаю, что имею право высказаться на эту тему. Что движет людьми, которые хотят создать новый бизнес? Для стартаперов очень важен их собственный профессиональный опыт в той области, в которую они хотят влезть, — с ним есть очень хорошие шансы запустить бизнес, не привлекая мегаинвестиции. А вот отсутствие опыта порой приходится заливать деньгами. Но вне зависимости от опыта забыть про выходные и свободное время приходится всем, кто параллельно с основной работой пилит стартап. Обычно его доводят до стадии MVP (Minimum Viable Product) и потом начинают активно искать инвестиции и питчить проект на конференциях перед инвесторами.

Венчурные инвесторы ищут проект, стоимость которого будет расти в экспоненциальной прогрессии. Они вкладываются во множество

стартапов, из которых, возможно, выстрелит и окупит все расходы только один. Поиск инвестора — это приключение: обязательно найдутся персонажи, которые захотят пожить за ваш счет. У меня был такой случай — один достаточно известный человек из e-commerce попросил 5 % компании за успешное знакомство с инвестором. Найти хорошего инвестора — большая удача. Хорошим инвестором я называю того, кто не просто даст деньги на выгодных условиях, но и не будет вмешиваться в прямое управление, а также будет помогать своими связями, например, чтобы найти потенциальных клиентов. Также хороший инвестор может знакомить основателей и менеджеров стартапа с менеджментом других портфельных компаний. Например, инвестор Ozon.ru Index Ventures возил нас в Лондон и знакомил с такими проектами, как Betfair, Last.fm и Lovefilm. А инвесторы и основатели Wikimart.ru организовывали поездки в Калифорнию и знакомили с Color, Netflix, eBay. Можно, конечно, назвать такие встречи хипстерским выпендрежем и напрасными расходами, но лично меня они очень воодушевляли на великие дела. Одно дело смотреть запись видеоконференции, другое — вживую разговаривать с людьми из легендарных компаний и видеть, что их проблемы похожи на твои, просто они масштабнее.

Когда происходит сделка, венчурный инвестор хочет дать меньше и получить больше, основатели же хотят продать меньше, но за большую цену. Условия сделки — вопрос переговоров, на которых уже начинают выстраиваться отношения между основателями и венчурным фондом. К сожалению, порой они оказываются такими, как у основателя «Хабрахабра» Дениса Крючкова и Mail.ru, почитайте его интервью про то, как он выкупал у этой компании долю [111]. Но бывает и по-другому — у Retail Rocket, например, очень хорошие отношения с венчурным фондом Impulse VC, который в нас инвестировал еще на старте проекта.

Какие доводы можно привести в пользу того, чтобы делать свой проект, вне зависимости от результатов? Они достаточно весомые — опыт создания стартапа дает понимание того, что важно, а что нет, и очень быстрое развитие, на порядок быстрее, чем в обычной корпоративной среде. А вот про обратную сторону медали мало кто говорит: из собственной компании нельзя вот так просто уйти — ты фактически привязан к ней золотой цепью. Это вопрос цены — кто раньше выходит, тот получает меньше. Я знаю людей, которые вышли из стартапов до их продажи стратегическим инвесторам и сильно от этого потеряли. Причины ухода могут быть разными: усталость, новое видение своего развития, разногласия с другими участниками проекта. Когда в управлении задействовано много людей — инвесторов и сооснователей, — часто может начаться история «лебедь, рак и щука»: на споры уходит больше времени, чем на сами действия. Плохо, когда бизнес превращается в политику, — но стартап тем и отличается от, скажем, семейного бизнеса: влияние фаундера будет неизбежно размываться, особенно с выходом компании на биржу, не говоря уже о продаже ее целиком стратегическому инвестору. Если основатель настолько любит свой проект, что ему трудно делить с кем-то свое детище, то схема с инвестициями не самая лучшая идея.

Когда я писал прошлую главу, на IPO вышла одна из компаний, в которых я работал, — Ozon.ru, а другая — Wikimart.ru — закрылась еще в 2016 году. Ozon.ru был основан в 1998 году, я пришел туда спустя шесть лет. Тогда он рос примерно на 40 % в год. Ни о какой культуре стартапа и речи не было, все просто делали свою работу. Wikimart.ru был основан в 2011 году двумя выпускниками Стэнфорда, которые прямо там придумали идею проекта и получили первые инвестиции, и культура в этой компании была совсем другой. Основатели нанимали умных людей без особого опыта работы в e-commerce — и это, на мой взгляд, было ошибкой. Невозможно организовать работу склада и доставки без соответствующего опыта — из-за этого экономика Wikimart.ru была хуже экономики Ozon.ru. Второе отличие: Wikimart.ru финансировался западным фондом Tiger Global, Ozon.ru — Barings Vostok Capital

Partners, который хоть и связан с Западом, но работает в нашей стране. Экономические санкции против России перекрыли поток западного капитала для Wikimart.ru — Tiger отказался от финансирования проекта. Третье — Wikimart.ru начинал как аналог Яндекс.Маркет, просто показывая разные предложения от продавцов, затем перешел к модели доставки от разных продавцов. К такой бизнес-модели рынок не был готов в тот момент, а сейчас эту идею реализовали многие крупные интернет-ритейлеры, например тот же Ozon.ru.

Еще одна особенность стартапа — массовые увольнения людей. Персонал — это одна из самых затратных статей бизнеса. Вместе с бурным ростом компании происходит рост штата, который напоминает деление клеток, и в какой-то момент он может стать неуправляемым: когда у каждого сотрудника появляется помощник, а у помощника — свой помощник, происходит расфокусировка действий компании. Но вот наступает очередной экономический кризис, инвесторы начинают давить на улучшение прибыльности и резать расходы, и тогда персонал приходится сокращать. В первый раз я столкнулся с этим в конце 2008 года, когда в Ozon.ru решили сократить 10 % сотрудников. Мне у себя увольнять было особенно некого, поэтому я просто сократил опубликованную вакансию. Помню, как генеральный директор Ozon.ru шел по спискам сотрудников, которые получил на проходной — там еще фиксировалось, во сколько ты пришел и ушел из офиса, — и увольнял тех, кто проводит меньше времени на работе. В Wikimart.ru на моих глазах тоже произошло сокращение штата — это был 2012 год, когда пришло печальное известие от Tiger. Там тоже нужно было сократить 10 % сотрудников, а в Ostrovok.ru в 2013 году мне и другим руководителям поставили задачу снизить бюджет каждого отдела на 30 % — и я сократил свою команду на треть. Надо отдать должное Ostrovok.ru, они сокращали по ТК, и люди получили несколько окладов. С моей точки зрения, сокращение персонала — это нормальное явление в стартапах для лечения болезней роста. Но, к сожалению, когда начинают резать расходы на персонал — увольняют часто без разбора. Учитывая, что в одних отделах штат

и правда раздут без причины, а в других подход к найму может быть все же более консервативным, принцип «сократить везде 10 %» правильным не назовешь.

## ЛИЧНЫЙ ОПЫТ

Стартапами я стал бредить, еще работая в Ozon.ru — именно тогда эта тема стала модной. Я тогда придумал проект по ремонту квартир с условным названием jobremont: пользователь публиковал бы там свой заказ, затем происходил аукцион исполнителей, и заказчик выбирал бы из них подходящего. Еще я хотел использовать для входа на сервис номер мобильного телефона, а не email. Авторизация через мобильный телефон станет популярной примерно через пять лет. Тогда я опубликовал пост на habr.ru, в целом получил положительные оценки и даже дважды встретился с потенциальными инвесторами. На второй встрече они предложили мне развивать другой проект, сообщив, что уже вложились в аналог remontnik.ru. Я отказался, но бригаду, которая делала потом ремонт в моей квартире, нашел на этом сайте. А через шесть лет возник новый проект — система рекомендаций для интернет-магазинов.

Заняться рекомендациями мы решили еще в Ozon.ru — у меня в аналитической базе оказались весьма ценные данные по трафику клиентов на сайте, и, как у любого любопытного инженера-аналитика из МФТИ или с мехмата МГУ, возник вопрос: как это можно использовать? Точно уже не вспомнишь, что было первым — курица или яйцо, данные или идея рекомендаций. Но факт в том, что нам — аналитикам и инженерам Ozon.ru — это было дико интересно, и мы были тогда молоды и красивы. А еще нам не давала покоя информация от Amazon.com, что 35 % продаж идет за счет рекомендаций. Ну а если человеку (а тем более группе людей), владеющему технологиями, что-то интересно, то его не остановить.

Вообще, Amazon еще в далеком 1999 году писал о рекомендательных сервисах в своих годовых отчетах для акционеров —

слово «recommendation» там появляется 10 раз. У нас в то время и интернет-магазинов толком не было. Amazon настолько крут, что статья Грега Линдена, который создавал там систему рекомендаций, «Две декады рекомендательных систем в Амазоне» [112] — одна из самых цитируемых статей, согласно сервису Google Scholar.

На меня большое впечатление произвела встреча в офисе Ozon.ru с Андреасом Вейгендом, который работал одним из ведущих аналитиков Amazon. Сейчас он преподает в Беркли и Стэнфорде, занимается консультированием гигантов e-commerce. Мне не давали покоя его слова: «Последний клик пользователя даст вам больше информации, чем вы о нем знали раньше». На тот момент я уже понимал, что социально-демографические данные ничто по сравнению с данными поведенческими. В своей статье [100] Вейгенд сказал, что наши поисковые фразы очень много говорят о нас. («We are what we search for ... that a powerful compression of people's lives is encoded in the list of their search queries».) Эта информация пригодилась мне в дальнейшем в Retail Rocket для написания «краткосрочных» персональных рекомендаций.

Итак, на сайте Ozon.ru на тот момент уже были какие-то рекомендации, созданные в свое время разработчиками, и мы решили расширить их функционал. Конечно, на тот момент отличным примером реализации рекомендаций был сайт Amazon.com. Там было очень много идей — например, выводить веса рекомендаций прямо в виджете, чтобы дать понять потенциальному покупателю, какой процент людей купили рекомендованный товар. Сейчас этого функционала на сайтах Ozon.ru и Amazon.com нет, но совсем недавно я обнаружил такой пример на немецком сайте по продаже электроники thomann.de, когда искал электронные барабаны.

В тот период были сделаны или доработаны следующие типы рекомендаций:

- с этим товаром часто покупают;
- те, кто смотрел эту страницу, затем купили;
- поисковые рекомендации;
- персональные.

С алгоритмом «С этим товаром часто покупают» произошла интересная история. Алгоритм статистически в лоб плохо работает. Тогда один из разработчиков нашел статью «Коллаборативная фильтрация товар-товар» [113] того же Грега Линдена и сам реализовал «косинус» на C#, используя мультипоточное программирование. Потому что на SQL-сервере писать косинус векторов интересов клиентов то еще удовольствие. После этого опыта я и уверовал в Великий Косинус в  $n$ -мерном пространстве векторов сессий клиентов, и это пригодилось мне в будущем.

Одной из сложных задач было измерить эффективность блоков рекомендаций, но у нас был уже купленный инструмент веб-аналитики Omniture Sitecatalyst (ныне Adobe Analytics), и с помощью анализа мерчандайзинга, о котором я писал выше, мы смогли с этим справиться. Именно это вы можете увидеть на 23-м слайде моей презентации о рекомендациях на Ozon.ru [115] — презентация не потеряла своей актуальности даже спустя 10 лет. Кстати, тогда мы достигли цифры 38 % от всех добавлений в корзину через рекомендательные блоки.

На некоторое время я забросил рекомендации, пока не оказался в Wikimart.ru — тогда поездка в офис Netflix и рассказ Эрика

Колсона об их технологиях сподвигли меня на полную смену технологий. На самом деле Wikimart.ru это было не нужно, все мои старые наработки успешно работали и на базах данных. Но именно Nadoor на тот момент открыл для меня границы невозможного — масштабирование вычислений вплоть до тысячи компьютеров одновременно. Мне не нужно было переписывать алгоритмы, ускорять их — достаточно было просто добавить еще компьютеров в кластер. Примерно через два года, в октябре 2012-го, я написал текст, оригинал которого представлен ниже.

## **СЕРВИС РЕКОМЕНДАЦИЙ ДЛЯ ИНТЕРНЕТ-МАГАЗИНОВ**

### **Цель**

Создать простой и быстрый облачный сервис рекомендаций товаров, который можно встраивать на сайт магазина без вмешательства во внутреннюю архитектуру сайта.

### **Монетизация**

Магазины могут платить по следующей схеме в порядке приоритета:

- оплата за заказы. Есть идея использовать анализ мерчандайзинга для точного определения покупки рекомендованного товара;
- оплата за клики;
- оплата за допсервисы, такие как предоставление персональных рекомендаций в рассылки или офлайн-данные по рекомендациям.

### **Типы рекомендаций**

Сервис может предоставлять следующие типы рекомендаций:

- поисковые рекомендации (внешний и внутренний поиск);
- наиболее популярные товары;
- товарные рекомендации (После просмотра купили, С этим товаром часто покупают);
- персональные рекомендации на сайте и в рассылках.

### Техническое описание

Сервис должен быть разделен на четыре абсолютно независимых блока:

- сервис будет собирать данные на сайте интернет-магазина с помощью JS-трекеров. Данные логируются на отдельные серверы;
- отдельно скачиваются данные (если возможно) по наличию товаров в магазине, то есть обычные xml-файлы;
- раз в сутки или чаще данные обрабатываются при помощи MapReduce. Рассчитываются рекомендации и помещаются в БД или файлы;
- отдельный веб-сервис выдает рекомендации на сайте магазина. Важно: нужно показывать только те товары, которые есть в наличии в магазине.

### Внедрение

Типовое внедрение должно включать установку JS-кода на сайт:

- трекера JS для сбора данных. Если установлен Google Analytics, то отдача важных событий на сайте (транзакция, добавление в корзину и т. д.);
- скрипта, который будет тянуть данные из веб-сервисов.

Это была полностью законченная идея создания сервиса рекомендаций для любого числа магазинов. В ноябре 2012-го на конференции ко мне подошел Николай Хлебинский и предложил делать такой сервис. Самое интересное, что моего текста он до этого не видел, но его заинтересовала моя презентация [115], а летом, за пару месяцев до этого, он писал мне письма с вопросами о моих наработках, которые я оставлял без ответа, потому что не хотел раскрывать свои идеи. Но наша встреча была вопросом времени и Колиного упорства. Следующим шагом мы образовали партнерство, написали наши доли и подписали простое соглашение между собой: в итоге Николай Хлебинский становился генеральным директором компании, Андрей Чиж техническим директором, а я директором по аналитике. Я до конца декабря уже сделал первые алгоритмы, начальные строчки которых были написаны в фудкорте ТЦ «Гага-

ринский» в Москве. За восемь месяцев до этого у меня родилась дочь — я шел гулять с коляской, брал с собой раскладную табуретку и писал код на планшете, сидя на улице в мороз. А уже в марте 2013-го мы запустились [116]. Мы знали, что параллельно с нами идет разработка похожего проекта crossss.ru, и хотели запуститься до них — кто первый встал, того и тапки. И нам это удалось. В то время партнеры Retail Rocket хорошо дополняли друг друга, каждый занимался своим делом и делал его на отлично, а через какое-то время мы получили инвестиции от Impulse VC — я бы рекомендовал всем хорошим проектам обращаться к ним за инвестициями.

Вначале мы опасались конкурентной борьбы, но она оказалась интересной — самым главным нашим конкурентом оказалась внутренняя разработка в компаниях. Какой-то сотрудник или команда в интернет-магазине вызывались самостоятельно написать такой сервис — обычно через год такой клиент возвращался к нам с контрактом. Ближе к 2020 году тренд изменился, конкурентная борьба между сервисами усилилась. Магазины начали доверять внешним сервисам и аутсорсить автоматизацию маркетинга и рекомендации. Я считаю, что мы сделали большой вклад в развитие этого доверия на рынке.

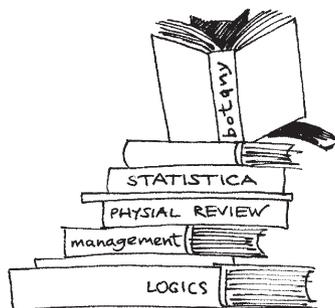
Важный вывод, который я сделал для себя: работа на две компании — это плохо: днем ты работаешь в офисе, вечером и в выходные вкалываешь на свое дело. В первые месяцы я работал по 80 часов в неделю, пока не уволился из Ostrovok.ru. Самое главное — когда пишешь вечером после целого рабочего дня до глубокой ночи, то весь следующий день, а то и два приходится исправлять ошибки. Больше скажу — некоторые недочеты были исправлены только через несколько лет. Именно тогда я решил, что впредь всегда буду заниматься только одним делом, поэтому, когда сел писать эту книгу, ушел в неоплачиваемый отпуск на пять месяцев.

Не все заказчики могут адекватно оценить ваши усилия по созданию качественного продукта. А в рекомендательных системах результат этих усилий не лежит на поверхности. Например, вы выяснили в результате А/Б-теста, что две рекомендательные системы дают одинаковый прирост продаж. Какую из них выбрать? При этом первая система дает намного более логичную картинку рекомендаций, чем вторая, — в первой в визуальную часть вложены усилия, чтобы у пользователя в голове не возникало диссонансов. У рекомендаций есть одна серьезная сложность: если начать улучшать «картинку», то результаты в А/Б-тестах ухудшаются. Я потратил очень много времени и вложил немало труда, чтобы сделать все красивее и не просесть в метриках. Сделали ли это конкуренты? Едва ли. Но клиенты обычно ориентируются на цену — какая стоит дешевле, ту и возьмем. Ну и между российским и американским решением обязательно выбирают второе. Что поделать, в импортное мы верим больше — об этом хорошо знают российские производители обуви, которые притворяются итальянскими и немецкими. Жизнь несправедлива — объяснить заказчику преимущества своего продукта порой невозможно, хотя мы все время бьемся над этим.

Следующий вывод, который я сделал для себя, — необходимо избавляться от токсичных клиентов, они не принесут вам ничего, кроме головной боли. Есть такой показатель — NPS (Net Promoter Score), который демонстрирует лояльность клиента компании. Сам NPS вычисляется посредством вопроса клиенту: «Какова вероятность, что вы порекомендуете нашу компанию друзьям или знакомым?» Ответ клиент обычно дает по десятибалльной шкале. Так вот, аналогичную оценку неплохо ввести и для самих клиентов — насколько сотрудникам приятно работать с ними. Я видел разных клиентов, от суперлояльных до предельно токсичных. Токсичные клиенты демотивируют сотрудников, которые вынуждены с ними работать. Нормальная деловая коммуникация — важный критерий качества работы с клиентом. Когда я снова буду создавать свою компанию, я пропишу в договоре с клиентами свое право на его расторжение на основе оценок, которые получу от своих сотрудников.

# 13

## СТРОИМ КАРЬЕРУ



## СТАРТ КАРЬЕРЫ

Цикл карьеры — это образование, старт, развитие и увольнение. Рекомендую начинать заниматься вопросом трудоустройства задолго до окончания вуза. Сам я начал работать со второго курса — но это был 1999 год, денег не было от слова «совсем». В то время зарплаты платили в долларах — помню, как радовался, когда получил свои первые две сотни. В тот момент я устроился программистом, но спустя год ушел, решив попробовать что-нибудь другое и ликвидировать долги по учебе. В StatSoft я устроился летом после третьего курса, прочитав небольшое бумажное объявление в читальном зале библиотеки МФТИ. В нем было написано: требуется переводчик с английского языка для локализации программного обеспечения. В StatSoft была очень хорошая и уютная атмосфера. Я работал в очень хорошей компании умных и амбициозных людей, студентов и аспирантов МФТИ и МГУ. Когда я закончил с переводами, стал потихоньку осваивать статистический пакет и теорию, которая за ним стоит, и участвовать в презентациях продукта. Первая презентация была в Центральном банке РФ. До сих пор помню, как пот струился по всему телу и сильно крутило живот. После нескольких таких презентаций я перестал есть перед ними. Лучше быть голодным, чем выступать с большим животом из-за нервов. Для меня, человека, которому сложно общаться с людьми, это была самая лучшая школа презентаций. Потом были командировки в Киев, Череповец, Красноярск — я участвовал в продажах и преподавал. Для меня это стало фундаментальной школой навыков, которых у меня тогда не было.

Конечно, это была счастливая случайность, что я попал на такую работу. Но для успеха одной удачи мало — надо ею правильно распорядиться. Одновременно со мной в StatSoft на должность стажера пришел еще один физтех с самого элитного факультета общей и прикладной физики (ФОПФ). Он ушел через месяц — ему не понравилось заниматься переводами, а возможно, еще через пару месяцев у него тоже появились бы интересные задачи. Кстати, на

сайте StatSoft.ru до сих пор есть хорошие вакансии для студентов старших курсов.

На старте карьеры я не рекомендую гнаться за большими деньгами — гораздо важнее не бояться попытать удачу, и, если она вам улыбнулась, воспользоваться ею. Мне повезло — моя работа мне понравилась сразу. Очень много физтехов не связывает свою дальнейшую жизнь с наукой, а мне приятно, что я занимаюсь прикладной наукой. Ищите стажировки, причем необязательно в больших «уважаемых» компаниях. Лучше даже начать с небольших фирм: там вы получите широкий кругозор и самостоятельность, а не будете просто мелким винтиком в гигантской корпорации, где вы максимально далеки от конечного продукта. Работая в небольшой фирме, вы будете больше влиять на результаты ее работы, на продажи и прибыль — что, с моей точки зрения, гораздо интереснее.

На что обратить внимание при выборе первого (а также второго, третьего и всех остальных) места работы? Во-первых, чему там научат? Я бы рекомендовал напрямую задавать такой вопрос на собеседовании. Если его сочтут за дерзость — значит, это не ваш работодатель. Во-вторых, личность руководителя — важная часть вашего становления как специалиста. Если он/она сильный — и как коллега, и как менеджер, — считайте, что вытянули счастливый билет. В-третьих, оказавшись на испытательном сроке, присмотритесь к корпоративной культуре — насколько она добра или токсична к новичкам. Я не считаю, что с ними нужно нянчиться, но должна существовать хотя бы самая простая программа введения в курс дела. Я сам обычно попадал в равнодушную среду — меня особо не тащили, но и не мешали, и этого мне было достаточно. Были моменты, когда мне передавали дела и я сталкивался с ревностью сотрудников. Но с этим должен уже разбираться руководитель. Еще один момент — не игнорируйте свою интуицию, когда проходите собеседования или уже работаете на испытательном сроке. Если вам кажется, что что-то не так, — возможно, вы правы.

## КАК ИСКАТЬ РАБОТУ

Есть три способа: обратиться в компанию напрямую, через профессиональных рекрутеров или использовать личные связи. Последний вариант лучше всего. Один сотрудник компании Netflix мне так и сказал — сюда нельзя попасть, если никто тебя не знает. В очень серьезные компании типа FAANG (Facebook, Apple, Amazon, Netflix, Google (Alphabet)) через знакомых проще получить доступ к рекрутеру. Самые лучшие предложения о найме я получил от тех, с кем когда-то работал: например, в Ozon я попал через своего бывшего руководителя из StatSoft, в Ostrovok.ru меня позвала моя бывшая сотрудница, которая устроилась туда работать. Поддерживайте хорошие отношения с коллегами, они вам еще пригодятся. Ответить на вакансию на сайте компании или hh.ru — менее популярный путь. У него есть минус по сравнению с трудоустройством через знакомых — о вас нет априорной информации (никто вас не рекомендовал). Кроме того, будем честны — самые интересные вакансии зачастую даже не публикуются на сайтах компаний. А вот поиск работы через профессиональных рекрутеров в моей практике ни разу не сработал — я никогда не находил работу и сам не нанял ни одного человека с их помощью. На мой взгляд, у такого посредничества есть только один плюс — иногда рекрутеры работают по скрытым вакансиям компаний (ищут топ-менеджеров), которые их наняли.

Как попасть на собеседование и подготовиться к нему? В первую очередь нужно внимательно прочитать весь текст объявления о вакансии, лучше несколько раз — почти каждое слово в нем несет смысл. Второй шаг — переписать резюме таким образом, чтобы рекрутер, просто просмотрев его, сразу понял(а), что вас стоит позвать. Не игнорируйте сопроводительное письмо — оно правда работает. Я в это не верил, пока сам не начал нанимать сотрудников. Напишите, почему вам интересна вакансия и почему вы считаете себя подходящим кандидатом. Для меня как руководителя это очень сильный сигнал, что человек очень хочет попасть к нам. Я обязательно обращаю внимание на такое резюме,

даже если у меня есть кандидаты посильнее, но без сопроводительного письма.

Вы получили приглашение на собеседование — обычно это короткий звонок по телефону, где попытаются выяснить вашу адекватность. Хороший рекрутер по одному телефонному разговору может даже дать вам характеристику, которая пойдет дальше по всем этапам собеседования. Также у вас попробуют узнать ваши зарплатные ожидания — тут правило такое: кто первый назвал цифру — тот и проиграл. Поэтому лучше всего отложить этот вопрос до последнего этапа переговоров. К собеседованию обязательно стоит подготовиться. У меня был такой опыт с Facebook и Quora. Quora провели пару вводных собеседований по видеосвязи. Затем я прилетел на два дня в их офис в Пало-Альто, где прошел 13 собеседований за первый день, и еще пять за второй. А Facebook даже присылал специальные памятки перед каждым собеседованием, как к нему подготовиться. Перед важными собеседованиями я всегда искал (или набрасывал сам) список вопросов, которые могли бы мне задать, и писал краткие ответы в блокнот. Благодаря записям вы будете себя уверенно чувствовать на интервью. Не на все вопросы во время него удастся ответить, но обязательно запишите и проработайте те, которые вызвали затруднения. Это можно сравнить с тренировкой — на собеседовании в другой компании некоторые из них вам могут задать снова. Обычно собеседование делится на три части — что вы делали раньше, теория и практика.

В первой части вы рассказываете о своем опыте. Вам будут задавать вопросы, и самый важный из них — почему вы сделали так или иначе. На хорошем собеседовании вас об этом обязательно спросят, и стоит заранее обдумать ответ, а еще лучше — попросить кого-то отрепетировать с вами собеседование. Тогда будет понятно, где вы «плывете». Я также обращаю внимание, как кандидат оценивал свой вклад в результаты компании.

Вторая часть обычно теоретическая — задают вопросы на простую теорию. Например, как работает тот или иной алгоритм, когда и какой статистический критерий применять. Как ни странно, многие

кандидаты не могут ответить на очень простые вопросы даже из моей главы об экспериментах. Если выучите минимум, вы сразу будете выгодно выделяться на фоне остальных кандидатов. Раньше было принято загадывать всякие абсурдные загадки (из серии «почему люки круглые») — особенно этим славился Google, — но потом вышло исследование, что правильные ответы не являются хорошим предиктором успешности кандидата. И загадки на собеседованиях прекратились.

Третья часть — практическая. Мы вместе с соискателем садимся за один компьютер и решаем несложную задачу его любимым способом. От кандидата на должность аналитика данных требуется обработать данные и сделать выводы. Здесь важна скорость — если есть практические навыки, то он справится быстро. Однажды меня позвали в одну группу поиграть на гитаре. Тогда я играл по табам (схемы, где обозначено, какую струну и когда зажать), аккордов не мог строить, но знал, как они устроены, теоретически. Чтобы что-то сыграть, мне нужно было построить и выучить каждый аккорд. Это отнимало много времени, я тормозил. Конечно, меня туда не взяли — у музыкантов не было времени ждать, пока я заиграю. Также с аналитикой и инженерией — простые вещи нужно уметь делать быстро. Если их делать медленно, то это будет тормозить рабочий процесс и всю команду. Этому можно научиться только на практике, и хорошая школа для развития таких навыков — Kaggle.

## ТРЕБОВАНИЯ К КАНДИДАТАМ

Приведу здесь базовые требования к аналитику данных (здесь я подразумеваю, что машинным обучением занимается инженер ML, а не аналитик данных).

Хороший аналитик должен разбираться в области, которой он занимается — будь то маркетинг, логистика, финансы, веб-аналитика или что-то еще. Обладать этим доменным знанием важно, потому что оно поможет быстро разобраться в рабочих вопросах.

Технические средства для аналитика — всего лишь инструменты. Я выступаю за разнообразие и начал бы, как ни странно, с Excel. Это действительно очень мощный инструмент и стандарт в финансах. На втором месте SQL — язык для работы с базами данных. Это самый популярный способ получения данных и их первоначальной обработки. Третью строчку моего личного рейтинга занимает базовая теория вероятностей и математическая статистика: среднее, медиана, дисперсия, корреляция, статистические критерии проверки гипотез. На четвертом — инструменты программирования: блокноты на Python (Jupyter Notebooks) или R.

Иногда в некоторых вакансиях требуется знание определенного софта. Не стоит переживать, если вы его не знаете, — когда есть база, отдельные навыки приобретаются легко.

Инженеров я бы разделил на две категории — инженеры по данным, задачей которых является обеспечение работы системы, и инженеров ML, которые работают над ML-моделями. С моей точки зрения, инженер по данным должен:

- уметь работать с Unix/Linux Shell;
- знать принципы MapReduce;
- уметь работать с Hadoop в случае необходимости;
- уметь работать с Kafka или другим стримовым софтом;
- работать с DAG-софтом (AirFlow, Oozie, ...) — системой, которая стоит граф расчетов.

Инженер ML должен:

- владеть базовыми алгоритмами машинного обучения так, чтобы он мог самостоятельно написать их. Иногда необходимо выходить за рамки стандартных библиотек ML и писать свое;
- владеть искусством создания фич (feature engineering) — этому можно научиться только на практике, например, решая задачи на Kaggle;

- уметь пользоваться системой контроля версий Git;
- уметь работать с контейнеризацией моделей и средствами ML Ops, например с ML Flow.

Кроме того, и те и другие инженеры должны хорошо владеть двумя языками программирования — Python и любым компилируемым из списка (C++, C#, Java, Scala), а также знать SQL

## ВЫ ПРИНЯЛИ ОФФЕР

Поздравляю! Помните, что испытательный срок, который обычно составляет два или три месяца, является испытательным и для компании тоже. Если вы попали в ситуацию, когда слова, сказанные при найме, сильно расходятся с делами, — уходите. Еще и поэтому не стоит рвать отношения с другими потенциальными работодателями (и даже с прошлым) сразу, когда вы принимаете предложение. Вполне нормально продолжать ходить на собеседования, выйдя на испытательный срок в новой компании. Это не нравится работодателям, но что поделать — вы должны вести свою игру. Может оказаться, что работа вам не подходит или вы не подойдете работодателю. У меня были сотрудники, которые не принимали мой оффер, уходили в другое место, но через несколько недель все-таки возвращались.

И еще. Любые отношения с работодателем выстраиваются в самом начале. Их сложно менять с течением времени — как вы себя поставите на испытательном сроке, так, скорее всего, и будет.

## КАК РАЗВИВАТЬСЯ И РАБОТАТЬ

В хорошей компании вас не оставят один на один с непонятной работой, в идеале вам должны назначить наставника и написать программу на испытательный срок. Выполнив ее, вы останетесь

работать в компании. Сейчас это стандарт для работы со стажерами и новичками, но на заре моей карьеры так было только в крупных компаниях.

И вот вы, пройдя испытательный срок, стали полноценным членом команды. Что дальше? Было бы хорошо договориться со своим менеджером об индивидуальном плане работы, из которого было бы понятно, где вам развиваться. Мы практикуем это в Retail Rocket — я убежден, что составление такого плана — работа менеджера, и если она не выполняется, можно его об этом попросить. В этот план могут быть включены навыки и знания, которые в дальнейшем должны привести к повышению вашего дохода или должности. Не нужно стесняться своих профессиональных амбиций — о них нужно заявлять. Те, кто это делает, добиваются большего. У менеджера много работы, он не обязан думать о ваших карьерных устремлениях. В нашей области нужно постоянно развиваться, проходит буквально несколько лет, и стандарты работы меняются. Ваш план развития не позволит вам выпасть из обоймы. Кроме того, нужно соблюдать баланс работы и развития, иначе легко застрять на одном месте. У меня очень большая занятость, но время на то, чтобы читать хорошие книги по специальности и проходить онлайн-курсы на Coursera, я стараюсь находить всегда.

Нам всем нужна обратная связь, чтобы понимать, где нужно улучшиться и насколько. Для этого придумали встречи в формате один на один (one-to-one) с вашим менеджером, которые проводятся в одно и то же время раз в неделю. Если в компании процесс управления задачами настолько отлажен, что для их постановки такие встречи не требуются, — все равно настаивайте на их проведении с вашим менеджером. Обычно все совещания коллективные, и там о многом политкорректно молчат. На встречах one-to-one намного проще говорить открыто и даже дать свою обратную связь руководителю (если вам, конечно, настолько повезло с руководителем). Я сам был и в роли подчиненного, и в роли менеджера, работал в тех компаниях, где был принят формат one-to-one, и в тех, где это

не практиковали, — так что я могу сравнивать. Я уверен, что такие встречи однозначно полезны и даже необходимы.

Простой исполнитель делает то, что ему скажут, — но когда речь идет об интеллектуальном труде, это не работает. Интеллектуальный труд сродни искусству — одну и ту же задачу можно сделать по-разному. И здесь, когда вам ставят задачу, иметь свое собственное мнение о ней нужно, даже если вы его не озвучиваете. Когда ваш репутационный вес вырастет, эта привычка вам пригодится.

Чем опытный сотрудник отличается от неопытного? Первому нужно сказать, что сделать, а второму — как делать. Чем выше вы растете как профессионал, тем реже вам говорят, как делать. А если и говорят, то вы уже, ориентируясь на собственное мнение, сможете ответить: «Я лучше сделаю так-то и так-то, потому что...» Так вы будете себя чувствовать не калькулятором, а полноценной интеллектуальной единицей.

Наша аналитическая работа отличается от физической тем, что кассир или сотрудник склада заканчивает свой рабочий день и прекращает думать о работе. С интеллектуальным трудом все иначе — мы продолжаем работать головой, даже когда рабочий день закончен, особенно если увлеклись интересной задачей. Часто гениальные решения приходят не сразу. В моем случае работает поговорка «утро вечера мудренее», а еще меня часто осеняет, когда я мою посуду. Поэтому у нас дома нет посудомоечной машины, иначе вместо озарений был бы скроллинг соцсетей. Ричард Хэмминг считает, что мозг накануне нужно плотно загрузить хорошей проблемой, тогда на следующий день проще будет найти решение [21]. Вы спросите, к чему я клоню? К тому, что невозможно эффективно решать задачи, сидя на рабочем месте

по двенадцать часов. Я работал на двух работах одновременно, когда строил Retail Rocket, все выходные и вечера, и заметил такую закономерность — если поздно вечером пишешь код, то потом весь следующий день вычищаешь в нем ошибки. Мы работаем неэффективно, когда устаем.

Сейчас я не могу больше трех часов подряд напряженно работать над проблемой — нужно делать перерыв и отдыхать. Причем под отдыхом подразумеваю полное отключение от рабочих проблем. В последние годы я люблю ездить на работу на велосипеде, даже в условиях нашей зимы — как только температура опускается ниже нуля, переобуваю шипованную резину и достаю непромокаемые штаны, которые заодно спасают от грязи на московских дорогах. Мне это очень помогает отключаться от рабочих проблем. Если еду на общественном транспорте, то голова гудит, а на велосипеде приезжаю домой свежий и бодрый — потому что был сосредоточен на дороге и больше ни о чем не думал.

Теперь немного о трудоголизме. О том, что это такое, я узнал на четвертом курсе, познакомившись с руководительницей одной из базовых кафедр в МФТИ. Нам, студентам, она рассказывала, что встает в пять утра и упорно работает допоздна. В итоге она в какой-то момент исчезла с радаров — поговаривали, что у нее возникли серьезные проблемы с психическим здоровьем. Я знаю людей, которые очень много работают и, живя в постоянном и сильном стрессе, вынуждены принимать психотропные средства, начиная от антидепрессантов и заканчивая транквилизаторами. Я уверен, что ни одна работа в мире не стоит психического здоровья, и считаю, что мы мало о нем заботимся.

Еще один важный аспект рабочего процесса — он не должен слишком зависеть от нашей персоны. Я называю это человеконезависимостью. Хороший менеджер всегда должен знать, что он будет делать, если сотрудник уйдет по тем или иным причинам. Компания должна продолжать работать, сильная зависимость от одного человека — это всегда большой риск. Я сам строю системы таким образом, чтобы они работали без моего участия, и компания, в слу-

чае чего, продолжала функционировать и без меня. Когда ушел из Ozon.ru — мои OLAP-кубы работали там еще много лет. Уходил из Wikimart.ru — рекомендательная система работала автономно. Конечно, это связано с риском для сотрудника — когда все хорошо сделано и автоматизировано, в какой-то момент ты становишься не нужен, а значит, тебя могут уволить. Но давайте посмотрим на это с другой стороны. Вы сделали свою работу, ваша система и процессы работают, пусть не идеально, но без серьезных ошибок. Возможно, у вас даже есть другой человек, который вас может легко заместить. Это и есть идеальная работа, вы можете спокойно развивать свою карьеру как в этой компании, так и в другой. А собравшись уходить, вы будете спокойны за свою репутацию, и вас с радостью порекомендуют ваши бывшие коллеги и руководители. Все это я испытал на своей шкуре.

## КОГДА МЕНЯТЬ МЕСТО РАБОТЫ

Менять работу нужно тогда, когда вектор вашего развития или амбиции уже не совпадают или даже вступают в конфликт с вектором движения компании. Понять, что это произошло, нетрудно — вы почувствуете, что вам больше не интересно то, что вы делаете. У меня так тоже было. Например, в 2005 году, когда я только пришел в Ozon.ru, — я сделал систему аналитики на основе MS SQL Server и в какой-то момент делать мне стало нечего. В это время из компании ушел мой руководитель, а в жизнь Ozon.ru вмешались новые акционеры, которым было не до меня. Я оказался предоставлен сам себе и стал заниматься на работе своим хобби — уже давно по вечерам я разрабатывал сигнализацию на основе датчика движения и мобильного телефона. Мне было двадцать три года, я снял первую в своей жизни квартиру и начитался в интернете историй про то, как хозяева приходят туда без спроса в отсутствие жильца. На этот случай я придумал технологию: в квартире лежал старый мобильный телефон с датчиком движения, и если бы кто-то пришел — он отзвонил бы на

мой номер, а я смог бы послушать, что происходит в квартире. Для разработки устройства мне нужно было написать программу для контроллера. И чтобы не бить баклуши в интернете, я начал писать эту программу прямо на рабочем месте. Это, конечно, очень плохая ситуация — неправильно повели себя и менеджеры, и я сам, но тогда она не стала причиной для увольнения. Когда все кадровые перестановки утряслись, задачи вновь появились, и я переключился на работу. Кстати, если вам интересно — хозяйка квартиры так ни разу и не пришла ко мне без спроса. А вот уволился из Ozon.ru я 5 лет спустя, когда у меня уже был свой личный кабинет по соседству с генеральным директором. В какой-то момент я почувствовал, что мне стало совершенно неинтересно работать, что я так могу и состариться на этой работе. Я попробовал себя в другой роли — вел проект по смене системы лояльности Ozon.ru на баллы, но меня это не зацепило. Тогда и стало окончательно понятно, что пора уходить, — и я ушел.

Известно, что самый быстрый рост карьеры происходит при относительно частой смене работ. Раньше считалось, что оптимальный срок работы на одной позиции в одной компании — два года. Кроме того, на решение об уходе влияет тип амбиций сотрудника. Я для себя выделил два: первый — новаторы, люди, которые любят придумать и сделать что-то новое с нуля, но их не привлекает процесс «полировки» их решения разными оптимизациями. Второй тип — оптимизаторы, которые могут вдохнуть новую жизнь в существующее решение, но придумать и создать что-то с нуля им сложно. Как только новатор запускает процесс — он достиг результата, ему становится скучно и он ищет новых вызовов. Я точно знаю, что отношусь к первому типу новаторов — пришел, увидел, победил: создал аналитику с нуля, нанял персонал, выстроил процессы, пошел дальше. Для развития и понимания карьеры в аналитике данных (да и в других областях) неплохо было бы попробовать себя и в роли новатора, и в роли оптимизатора и понять, что больше нравится. Тогда станет намного легче принимать карьерные решения.

Вот что сказал Ричард Хэмминг о том, как строить карьеру [21]:

«Примерно каждые семь лет значительно, если не полностью, меняйте область своей работы. К примеру, я переключался с вычислительной математики на аппаратное обеспечение, оттуда на программное обеспечение, и так далее, потому что есть тенденция к расходованию своих идей...

Вы должны меняться. Вы со временем устаете; вы расходуете свою оригинальность в одной области. Вам надо найти что-то рядом. Я не говорю, чтобы вы переключались с музыки на теоретическую физику, а там на английскую литературу; я подразумеваю, что в своем поле вам следует переключаться между областями, чтобы не застаиваться».

Хэмминг говорил об исследовательских работах, но я бы применил их и к области анализа данных. Никто не мешает вам перемещаться между «доменами» от веб-аналитики в машинное обучение, от аналитики к программированию и наоборот.

Если вы уже давно чувствуете, что вам неинтересно и нечего делать на работе, поговорите со своим менеджером. Если во время или после разговора станет понятно, что ничего не изменится, то лучше уходить. Впоследствии вас не будет мучить совесть, что вы засиделись и потеряли драгоценное время. К сожалению, оно не бесконечно.

## НУЖНО ЛИ ВСЕ ЗНАТЬ?

Однажды уже известному нам ученому Ричарду Хэммингу задали вопрос: «Сколько сил должно уходить на работу в библиотеке?» И вот что он ответил [21]:

«Это зависит от области. Я вот что скажу об этом. Был парень в Bell Labs. Очень-очень умный парень. Он всегда был в библиотеке, он читал все. Если вы хотели ссылок, вы шли к нему и он давал вам всякие разные ссылки. Но в процессе формирования этих теорий я сформулировал утверждение: его именем в долгосрочной перспективе не будет названо ни одного эффекта. Он уже ушел из Bell Labs и является адъюнкт-профессором. Он был очень ценен; это несомненно. Он написал некоторые очень хорошие статьи в *Physical Review*; но его именем не было названо ни одного эффекта, потому что он читал слишком много».

Все знать невозможно, чтение отнимает очень много времени. Я очень люблю читать, но на это у меня мало времени. Книга ждет своего часа, иногда десять лет, пока я не возьму ее с полки. Когда у меня стоит выбор между пассивными (теория) и активными (практика) действиями, я выбираю активное. В пассивное включаются чтение, просмотр видео с конференций, занятия на онлайн-курсах. При этом я не могу назвать себя неучем — у меня с десяток сертификатов Coursera. Под активным действием я подразумеваю решение какой-либо задачи, доведение какого-то проекта (пусть даже личного) до конца. Особенно меня интересуют рабочие задачи, а не учебные. Важно обладать эрудицией и знаниями, но гораздо важнее их умелое применение. А чтобы применить знание, не нужно разбираться в каком-то предмете на 100 %, бывает достаточно всего 20 % (правило Парето!). Помните, в главе про машинное обучение я сослался на Шавье Аматриана, который предлагал просто прочитать введение книги по ML, открыть нужный алгоритм и закодировать его. Именно практика покажет, что важно, а что нет. Невозможно играть хорошую музыку, лишь зная ее теорию.

Какой сотрудник лучше: тот, кто прошел штук 20 разных курсов и сделал очень много учебных заданий, или тот, кто пару несложных проектов довел до конца, от идеи до реализации? В 95 % случаев я выберу второго. Моя личная практика показала, что есть люди-теоретики, а есть практики. Я нанимал теоретика на одну из наших задач, думая, что если человек хорошо разбирается

в теории, то он разберется и в практике. Ничего хорошего из этого не получилось.

Хорошо, можно много не читать, но как тогда получать информацию? Вот что написал Хэмминг:

«Если вы все время читаете, что сделали другие люди, вы будете думать, как думали они. Если вы хотите думать новые мысли, отличающиеся мысли, тогда делайте то, что делают многие креативные люди: сформулируйте задачу достаточно ясно и отказывайтесь смотреть какие-либо ответы, пока основательно не продумаете задачу – как вы будете решать ее, как вы можете немного изменить ее, чтобы формулировать ее правильно. Поэтому да, вам надо быть в курсе. Вам надо быть в курсе больше, чтобы узнавать, какие есть задачи, чем читать, чтобы находить решения. Чтение, чтобы находить решения, не кажется путем к значимым исследованиям. Поэтому я дам два ответа. Вы читайте; но имеет значение не сколько, а как вы читаете».

В свое время меня критиковал мой преподаватель физики: «Зачем ты изобретаешь велосипед и выводишь эту формулу?» Я никогда не любил учить наизусть, просто ненавидел это. Мне было проще знать базовые принципы и несколько формул, все остальное выводится через них — пускай это и дольше, чем пользоваться готовыми. Вы знаете, что на письменном экзамене по физике на Физтехе можно пользоваться любой литературой? Хоть целый рюкзак учебников приноси. Как вы думаете, это сильно помогало? Нет, потому что без понимания базовых принципов сложнейшие физические задачи не решить. Принципы решения задач отрабатываются на задачах, а не на чтении решений и заучивании формул. Невозможно все знать, и бывает проще найти свой подход, чем копать горы литературы в поисках очередных «кейсов». Кейсы сейчас воспринимаются как рецепты — применишь, и все получится. Но чего-то нестандартного из кейсов уже не сочинишь.

Еще Хэмминг заметил, что ученые в Bell Labs, у которых была приоткрыта дверь в кабинет, добились большего, чем ученые, дверь которых была закрыта. И это тоже альтернатива чтению. Откры-

тость ума и широта взглядов помогают добиться успеха. Что мы можем для этого сделать? Как минимум не отмахиваться от коллег, как от назойливых мух, и не запирается от них в своих кабинетах с магнитными замками. Да, они могут отвлекать своими вопросами и просьбами, но они также и опускают вас с небес ваших мыслей на земную твердь реальности. Коллеги — источник информации, которую вы нигде больше не получите. Заодно они могут участвовать в брейншторме ваших идей — в информационном вакууме заточить их невозможно. Будучи сооснователем и директором по аналитике Retail Rocket, я иногда садился со своим ноутбуком в других отделах, просто сидел, слушал в фоновом режиме, что там происходит, и именно так узнавал о проблемах, о которых не подозревал и никогда бы не узнал, если бы задавал вопросы в лоб. И это давало мне полезную информацию, которую я мог воплотить в своих решениях.

# ЭПИЛОГ

Цель этой книги — дать практические советы. И если у вас получится применить в работе хотя бы несколько моих идей, для меня, как для автора, это будет успехом.

Напоследок еще один совет: постоянно задавайте себе вопрос «все ли я выжимаю из данных?» Где-то аналитику я организовывал сам (Ozon.ru, Wikimart.ru, Retail Rocket), где-то консультировал («Технониколь», «Иннова», «Купивип», Fastlane Ventures) и понял, что дело не только в цифрах. Чтобы использовать данные максимально эффективно, нужно, во-первых, следить за качеством самих данных, а во-вторых, правильно организовать взаимодействие людей внутри компании, приоритизацию гипотез и использование технологий. Все эти направления я постарался подробно разобрать в главах этой книги.

Мы получаем знания о жизни методом проб и ошибок: дети экспериментируют больше, а взрослые меньше. Точно так же организации, чтобы развиваться, нужно экспериментировать: генерировать идеи, проверять их на практике, получать результат и повторять этот цикл снова и снова, даже если результат не так хорош, как вы ожидали, и у вас опускаются руки. Не бойтесь неудач — именно через эксперименты происходят все улучшения.

Вы всегда можете связаться со мной через сайт поддержки книги <https://topdatalab.ru/book> или написать мне по адресу электронной почты [rykov@todatalab.ru](mailto:rykov@todatalab.ru). Буду рад ответить на ваши вопросы!

# СПИСОК ЛИТЕРАТУРЫ

Список литературы для книги сделан в основном на ссылках. Спустя какое-то время часть ссылок перестает работать. Поэтому я сделал механизм поддержки работающих ссылок. Он построен на URL: [http://topdatalab.ru/ref?link=\[Номер ссылки\]](http://topdatalab.ru/ref?link=[Номер ссылки]). Номер ссылки означает номер из списка литературы (например, для номера 1 — <https://topdatalab.ru/ref?link=1>). Если какая-то ссылка или QR-код в книге перестает работать и мне становится об этом известно, то я восстанавливаю ее работоспособность. Читателю для этого ничего делать не нужно, кроме как уведомить меня по email ([rzykov@topdatalab.ru](mailto:rzykov@topdatalab.ru)), что ссылка перестала работать.

1. Роберт Сапольски. «Биология добра и зла. Как наука объясняет наши поступки».
2. Письмо Amazon.com акционерам, 2015 год. <https://www.dropbox.com/s/c2m4zvcv0bxqb6f/2015-Letter-to-Shareholders.PDF?dl=0>
3. Письмо Amazon.com акционерам, 2016 год. <https://blog.aboutamazon.com/company-news/2016-letter-to-shareholders>
4. What is decision intelligence. <https://towardsdatascience.com/introduction-to-decision-intelligence-5d147ddab767>
5. Focus on decisions not outcomes. <https://towardsdatascience.com/focus-on-decisions-not-outcomes-bf6e99cf5e4f>
6. В России очень низкая смертность от коронавируса. Как ее считают? <https://www.bbc.com/russian/news-52641008>
7. Understanding Decision Fatigue. <https://topdatalab.ru/ref?link=7>

8. Саша Сулим, «Безлюдное место. Как ловят маньяков в России».
9. «Building Data Science Teams» by DJ Patil. <https://www.dropbox.com/s/9scdtqmi8k2lb5y/Building%20Data%20Science%20Teams.pdf?dl=0>
10. What's the difference between analytics and statistics? <https://towardsdatascience.com/whats-the-difference-between-analytics-and-statistics-cd35d457e17>
11. «Поиск причин проблем». <https://multithreaded.stitchfix.com/blog/2016/03/23/debunking-narrative-fallacies/>
12. «Атака на АБ-тест: рецепт 'R'+t(101)+'es46». <https://habr.com/ru/company/retailrocket/blog/330012/>
13. Измеряйте самое важное. Как Google, Intel и другие компании добиваются роста с помощью OKR | Дорр Джон.
14. Dogs vs. Cats: Create an algorithm to distinguish dogs from cats. <https://www.kaggle.com/c/dogs-vs-cats>
15. ResNet-50 is a convolutional neural network. <https://github.com/matlab-deep-learning/resnet-50>
16. Data scientists mostly just do arithmetic and that's a good thing. <https://m.signalvnoise.com/data-scientists-mostly-just-do-arithmetic-and-thats-a-good-thing/>
17. Интервью для BBC Карл Густав Юнг, основатель аналитической психологии, 1955 год. <https://www.bbc.com/russian/features-53475033>
18. Тирания показателей, Джерри Мюллер. <https://www.alpinabook.ru/catalog/book-542297/>
19. Курс молодого бойца для Spark/Scala. <https://habr.com/ru/company/retailrocket/blog/302828/>
20. Задачи руководителя аналитики. <https://www.quora.com/How-do-I-move-from-data-scientist-to-data-science-management>

21. Ричард Хэмминг. «Вы и ваша Работа». <https://habr.com/ru/post/209100/>
22. Planning Poker: как сделать процесс постановки задач максимально прозрачным и четким. <https://habr.com/ru/company/retailrocket/blog/334256/>
23. Hypothesis Testing: How to Eliminate Ideas as Soon as Possible. Roman Zykov. <https://recsys.acm.org/recsys16/industry-session-3/#content-tab-1-1-tab>
24. Application of Kullback-Leibler divergence for short-term user interest detection. <https://arxiv.org/abs/1507.07382>
25. Нужен ли магазину «Стильный кросселл»: опыт Retail Rocket в анализе изображений для формирования рекомендаций. <https://habr.com/ru/company/retailrocket/blog/441366/>
26. The most powerful idea in data science. <https://towardsdatascience.com/the-most-powerful-idea-in-data-science-78b9cd451e72>
27. Элементарные понятия статистики. <http://statsoft.ru/home/textbook/esc.html>
28. Джин Желязны. «Говори на языке диаграмм».
29. Эссе «The Cognitive Style of Powerpoint: pitching out corrupts within» Edward R. Taftay.
30. Retail Rocket: 9 советов по увеличению эффективности парного программирования. <https://habr.com/ru/company/retailrocket/blog/339358/>
31. Retail Rocket: работа с бэклогом задач с точки зрения проектного менеджера в Retail Rocket. <https://habr.com/ru/company/retailrocket/blog/329346/>
32. Корпоративная культура Netflix. <https://jobs.netflix.com/culture>
33. Ecommerce dataset для рекомендательных систем компании Retailrocket. <https://www.kaggle.com/retailrocket/ecommerce-dataset>

34. «Антихрупкость архитектуры хранилищ данных». <https://habr.com/ru/post/281553/>
35. Колоночные базы данных. <https://ruhighload.com/Колоночные+базы+данных>
36. Патент Google на MapReduce. <http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PALL&p=1&u=/netahtml/PTO/srchnum.htm&r=1&f=G&l=50&s1=7,650,331.PN.&OS=PN/7,650,331&RS=PN/7,650,331>
37. MapReduce: Simplified Data Processing on Large Clusters. <https://www.dropbox.com/s/azf00wnjwnqd2x8/mapreduce-osdi04.pdf?dl=0>
38. The Friendship That Made Google Huge. <https://www.newyorker.com/magazine/2018/12/10/the-friendship-that-made-google-huge>
39. Apache Hadoop. <https://hadoop.apache.org/>
40. Apache Spark. <http://spark.apache.org/>
41. Решение проблемы загрузки мелких файлов на Spark. <https://github.com/RetailRocket/SparkMultiTool>
42. Маккини Уэс «Python и анализ данных» (Python for Data Analysis. Wes McKinney).
43. Cloudera Hadoop — Choosing and Configuring Data Compression. [https://docs.cloudera.com/documentation/enterprise/6/6.3/topics/admin\\_data\\_compression\\_performance.html](https://docs.cloudera.com/documentation/enterprise/6/6.3/topics/admin_data_compression_performance.html)
44. Google colab. <https://colab.research.google.com/>
45. Kaggle notebooks. <https://www.kaggle.com/notebooks>
46. Gartner Top 10 Trends in Data and Analytics for 2020. <https://www.gartner.com/smarterwithgartner/gartner-top-10-trends-in-data-and-analytics-for-2020/>
47. Metabase. <https://www.metabase.com/>

48. SuperSet. <https://superset.apache.org/>
49. Beyond Interactive: Notebook Innovation at Netflix. <https://netflixtechblog.com/notebook-innovation-591ee3221233>
50. What Artificial Intelligence Can and Can't Do Right Now. <https://hbr.org/2016/11/what-artificial-intelligence-can-and-cant-do-right-now>
51. Фёрстер Э., Рёнц Б. Методы корреляционного и регрессионного анализа. Руководство для экономистов. Перевод с немецкого и предисловие В. М. Ивановой. М.: Финансы и статистика, 1983 г.
52. Are We Really Making Much Progress? A Worrying Analysis of Recent Neural Recommendation Approaches. <https://arxiv.org/abs/1907.06902>
53. Kaggle's State of Data Science and Machine Learning 2019. <https://www.kaggle.com/kaggle-survey-2019>
54. Unity is strength — A story of model composition. <https://medium.com/criteo-labs/unity-is-strength-a-story-of-model-composition-49748b1f1347>
55. Introduction to Machine Learning, Second Edition. Ethem Alpaydin.
56. Scikit learn Ensemble methods. <https://scikit-learn.org/stable/modules/ensemble.html>
57. XGBoost: Introduction to Boosted Trees. <https://xgboost.readthedocs.io/en/latest/tutorials/model.html>
58. LightGBM. <https://lightgbm.readthedocs.io/>
59. Catboost. <https://catboost.ai/>
60. Andrew Ng. Machine learning Yearning.
61. Coursera Machine Learning. <https://www.coursera.org/learn/machine-learning>

62. How do I learn machine learning? <https://qr.ae/pN9vA4>
63. Fastml4j on Scala. <https://github.com/rzykov/fastml4j>
64. Netflix prize. <https://www.netflixprize.com>
65. Netflix Recommendations: Beyond the 5 stars (Part 1). <https://netflixtechblog.com/netflix-recommendations-beyond-the-5-stars-part-1-55838468f429>
66. Andrew Gelman, Jenifer Hill «Data Analysis Using Regression and Multilevel/Hierarchical Models». <https://www.dropbox.com/s/a82wwn6l74j5qka/Gelman-missing.pdf?dl=0>
67. Google Course of ML: Imbalanced Data. <https://developers.google.com/machine-learning/data-prep/construct/sampling-splitting/imbalanced-data>
68. «10 уроков рекомендательной системы Quora». <https://habr.com/ru/company/retailrocket/blog/341346/>
69. «Как разрушился стартап, годами выдававший армию бухгалтеров за искусственный интеллект». <https://www.forbes.ru/tehnologii/405589-kak-razrushilsya-startap-godami-vydavavshiy-armiyu-buhgalterov-za-iskusstvennyy>
70. Yandex Toloka. <https://toloka.ai/ru/>
71. Drilling Down: Turning Customer Data into Profits with a Spreadsheet - Third Edition, Jim Novo.
72. Google Rules of Machine Learning: Best Practices for ML Engineering. <https://developers.google.com/machine-learning/guides/rules-of-ml>
73. В Норвегии с паразитами рыб борются при помощи подводных роботов с лазерами. <https://habr.com/ru/post/402797/>
74. «Русская аквакультура» взялась за разведение лосося на Дальнем Востоке. <https://www.rbc.ru/business/01/09/2020/5f4cb2159a7947347c5c16a9>

75. Adidas backpedals on robotic shoe production with Speedfactory closures. <https://techcrunch.com/2019/11/11/adidas-backpedals-on-robotic-factories/>
76. Рохо, Антонио. Возможно да, возможно нет. Фишер. Статистический вывод // Наука. Величайшие теории. — М.: Де Агостини, 2015. — Вып. 47. — ISSN 2409-0069.
77. Larry Wasserman, All of Statistics: A Concise Course in Statistical Inference (Springer Texts in Statistics), Springer (December 1, 2010).
78. Непараметрические критерии. <http://statistica.ru/theory/непараметрические-критерии/>
79. B. Efron, Bootstrap Methods: Another Look at the Jackknife. <https://doi.org/10.1214/aos/1176344552>
80. Bootstrap confidence intervals. [https://www.dropbox.com/s/6dbqxrccmfxyvp/MIT18\\_05S14\\_Reading24.pdf?dl=0](https://www.dropbox.com/s/6dbqxrccmfxyvp/MIT18_05S14_Reading24.pdf?dl=0)
81. Criteo Labs: Why your A/B-test needs confidence intervals. <https://medium.com/criteo-labs/why-your-ab-test-needs-confidence-intervals-bec9fe18db41>
82. Bayesian A/B tests. <https://richrelevance.com/2013/05/21/bayesian-ab-tests/>
83. William Bolstard, Introduction to Bayesian Statistics.
84. Ron Kohavi, Alex Deng, Roger Longbotham, and Ya Xu. Seven Rules of Thumb for Web Site Experimenters. <https://exp-platform.com/rules-of-thumb/>
85. Retail Rocket Segmentator. <https://github.com/RetailRocket/RetailRocket.Segmentator>
86. Reinforcement Learning: An Introduction. <https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf>
87. The Privacy Project. <https://www.nytimes.com/interactive/2019/opinion/internet-privacy-project.html>

88. One Nation tracked. <https://www.nytimes.com/interactive/2019/12/19/opinion/location-tracking-cell-phone.html>
89. Google Authorized Buyers, Real-time Bidding. <https://developers.google.com/authorized-buyers/rtb/start>
90. Explained: Data in the Criteo Engine. <https://www.criteo.com/blog/explained-data-in-the-criteo-engine/>
91. We Built an ‘Unbelievable’ (but Legal) Facial Recognition Machine. <https://www.nytimes.com/interactive/2019/04/16/opinion/facial-recognition-new-york-city.html>
92. What ISPs Can See, Upturn, March 2016. <https://www.upturn.org/reports/2016/what-isps-can-see/>
93. The GDPR Is a Cookie Monster. <https://content-na1.emarketer.com/the-gdpr-is-a-cookie-monster>
94. IAB. Cookies on Mobile 101. <https://www.iab.com/wp-content/uploads/2015/07/CookiesOnMobile101Final.pdf>
95. How Online Shopping Makes Suckers of Us All. <https://www.theatlantic.com/magazine/archive/2017/05/how-online-shopping-makes-suckers-of-us-all/521448/>
96. Почему крупнейшие сайты рунета убирают счетчик Liveinternet? <https://vc.ru/flood/1822-pochemu-krupneyshie-saytyi-runeta-ubirayut-schetchik-liveinternet>
97. How To Break Anonymity of the Netflix Prize Dataset. <https://arxiv.org/abs/cs/0610105>
98. ‘Alexa, are you invading my privacy?’ – the dark side of our voice assistants. <https://www.theguardian.com/technology/2019/oct/09/alexa-are-you-invading-my-privacy-the-dark-side-of-our-voice-assistants>
99. LeakyPick: IoT Audio Spy Detector. <https://arxiv.org/abs/2007.00500>

100. I Search, Therefore I AM, Andreas Weigend. <https://www.dropbox.com/s/xk6w60suzuq6dpeh/WeigendFOCUS2004-en.pdf?dl=0>
101. We Read 150 Privacy Policies. They Were an Incomprehensible Disaster. <https://www.nytimes.com/interactive/2019/06/12/opinion/facebook-google-privacy-policies.html>
102. 5 Americans who used NSA facilities to spy on lovers. <https://www.washingtonpost.com/news/the-switch/wp/2013/09/27/5-americans-who-used-nsa-facilities-to-spy-on-lovers/>
103. Pie & AI Asia: On Ethical AI with Andrew Ng. <https://www.deeplearning.ai/blog/pie-ai-asia-on-ethical-ai-with-andrew-ng/>
104. What Do We Do About the Biases in AI? <https://hbr.org/2019/10/what-do-we-do-about-the-biases-in-ai>
105. Уже почти половина россиян стали блокировать интернет-рекламу. <https://www2.deloitte.com/ru/ru/pages/about-deloitte/deloitte-in-press/2019/blokirovka-internet-reklamy.html>
106. IAB Europe Guide to the Post Third-Party Cookie Era. <https://iab europe.eu/knowledge-hub/iab-europe-guide-to-the-post-third-party-cookie-era/>
107. GDPR vs ФЗ-152. <https://vc.ru/flood/42581-gdpr-vs-fz-152>
108. This Article Is Spying on You. <https://www.nytimes.com/2019/09/18/opinion/data-privacy-tracking.html>
109. Functionalism: A New Approach to Web Analytics. [https://www.dropbox.com/s/a75hmjzekf006ia/wpaper\\_005.pdf?dl=0](https://www.dropbox.com/s/a75hmjzekf006ia/wpaper_005.pdf?dl=0)
110. Артур М. Хьюс. «Маркетинг на основе баз данных».
111. «Как Денис Крючков выкупил Хабр у Mail.ru». <https://habr.com/ru/company/roem/blog/193488/>
112. Two Decades of Recommender Systems at Amazon.com. <https://www.amazon.science/publications/two-decades-of-recommender-systems-at-amazon-com>

113. Item-to-Item Collaborative Filtering, Greg Linden, Brent Smith, and Jeremy York. <https://www.dropbox.com/s/dctxbv8dk8wrsmw/Amazon-Recommendations.pdf?dl=0>
114. How to use Merchandising eVars in Adobe Analytics. <https://dmpg.co.uk/how-to-use-merchandising-evars-in-adobe-analytics-product-modules>
115. Роман Зыков. «Итоги запуска системы рекомендаций на Ozon.ru». <https://www.dropbox.com/s/68kixkhk5yqjwve/kibrzykov24thapr2009.pdf?dl=0>
116. Стартап RetailRocket — сервис для рекомендации товаров в интернет-магазинах. <https://vc.ru/story/1285-retailrocket-ecommerce>
117. Шпаргалка методов машинного обучения. [https://scikit-learn.org/stable/tutorial/machine\\_learning\\_map/index.html](https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html)

*Роман Зыков*

**Роман с Data Science.  
Как монетизировать большие данные**

Заведующая редакцией	<i>Ю. Сергиенко</i>
Ведущий редактор	<i>К. Тульцева</i>
Художественный редактор	<i>В. Мостипан</i>
Литературный редактор	<i>Е. Зыкова</i>
Корректоры	<i>В. Беляева, Н. Сидорова</i>
Верстка	<i>Л. Егорова</i>

Изготовлено в России. Изготовитель: ООО «Прогресс книга».

Место нахождения и фактический адрес: 194044, Россия, г. Санкт-Петербург,

Б. Сампсониевский пр., д. 29А, пом. 52. Тел.: +78127037373.

Дата изготовления: 06.2021. Наименование: книжная продукция.

Срок годности: не ограничен.

Налоговая льгота — общероссийский классификатор продукции ОК 034-2014, 58.11.13 —  
Книги печатные для детей.

Импортер в Беларусь: ООО «ПИТЕР М», 220020, РБ, г. Минск, ул. Тимирязева, д. 121/3, к. 214,  
тел./факс: 208 80 01.

Подписано в печать 18.05.21. Формат 60×90/16. Бумага офсетная. Усл. п. л. 20,000.  
Доп. тираж. Заказ 0000.



## ВАША УНИКАЛЬНАЯ КНИГА

Хотите издать свою книгу?

Книга может стать идеальным подарком для партнеров и друзей или отличным инструментом продвижения личного бренда. Мы поможем осуществить любые, даже самые смелые и сложные, идеи и проекты!

### МЫ ПРЕДЛАГАЕМ

- издание вашей книги
- издание корпоративной библиотеки
- издание книги в качестве корпоративного подарка
- издание электронной книги (формат ePub или PDF)
- размещение рекламы в книгах

### ПОЧЕМУ НАДО ВЫБРАТЬ ИМЕННО НАС

В 2021 году исполнится 30 лет, как «Питер» издает полезные и интересные книги. Наш опыт — гарантия высокого качества. Мы печатаем книги, которыми могли бы гордиться и мы, и наши авторы.

### ВЫ ПОЛУЧИТЕ

- услуги по обработке и доработке вашего текста
- современный дизайн от профессионалов
- высокий уровень полиграфического исполнения
- продажи книги в крупнейших книжных магазинах страны
- продвижение книги (реклама в профильных изданиях и местах продаж; рецензии в ведущих СМИ; интернет-продвижение)

Мы имеем собственную сеть дистрибуции по всей России и в Белоруссии, сотрудничаем с крупнейшими книжными магазинами страны и ближнего зарубежья. Издательство «Питер» — постоянный участник многих конференций и семинаров, которые предоставляют широкие возможности реализации книг. Мы обязательно проследим, чтобы ваша книга имелась в наличии в магазинах и была выложена на самых видных местах. А также разработаем индивидуальную программу продвижения книги с учетом ее тематики, особенностей и личных пожеланий автора.

**Свяжитесь с нами прямо сейчас:**

Санкт-Петербург — Анна Титова, (812) 703-73-73, [titova@piter.com](mailto:titova@piter.com)