ГЛАВА 1
Основные этапы процесса машинного обучения

Работа аналитика начинается с контактов с лицами, заинтересованными в решении определенной бизнес-задачи (например, обнаружение мошенничества с банковскими картами или предсказание курса биткойна). Зачастую эти бизнес-партнеры (или конечные пользователи) являются коллегами аналитика из других отделов предприятия, таких как финансы или маркетинг, но они могут представлять и другую организацию. После этого аналитик может приступать к работе, используя при написании своего кода общедоступные библиотеки машинного обучения.

Прежде чем приступать к описанию популярных библиотек, я хочу представить вам весь процесс ML с «высоты птичьего полета». На рис. 1.1 он показан в виде схемы.

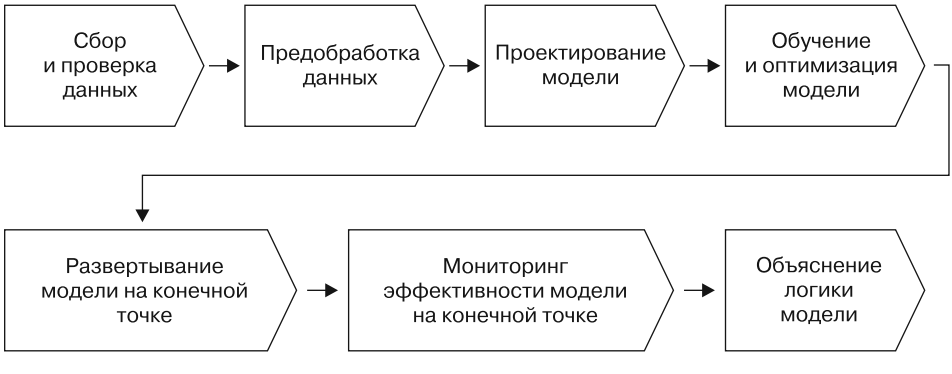


Рис. 1.1. Базовые этапы процесса машинного обучения

Процесс начинается со *сбора исходных данных для обучения модели*. Как правило, данные находятся вне проектируемой системы машинного обучения,

например на интернет-странице или в базе данных, установленной локально или же в облачном сервисе[1]. Данные бывают структурированными, слабоструктурированными и неструктурированными (см. приложение Б). В этой книге (чтобы не превращать ее в энциклопедию) я ограничусь только структурированными. Такие данные существуют в форме матрицы или таблицы, строки которой содержат *записи* (records) или *примеры* (examples), а столбцы — *признаки* (features), характеризующие эти записи или примеры. Например, записью может являться информация о личном автотранспорте владельца, где идентификатором является номер автомобиля, а признаками — его марка, цвет, год регистрации, пробег в тысячах километров и т. д.

Данные, загруженные или считанные из источника, часто несовершенны по качеству. Поэтому аналитики применяют *визуализацию* и *профилирование* для качественной и количественной оценки имеющихся данных соответственно. При визуализации исследуемые данные графически отображаются в выбранных аналитиком представлениях, где дефекты и несоответствия ожиданиям могут быть зрительно обнаружены. Профилирование дополняет визуализацию и позволяет с минимальным написанием кода, генерирующего отчет, вычислить набор количественных (и часто немногих графических) характеристик с целью обнаружить «отклонения от ожидаемой нормы» в данных.

Изучив графики и отчет по профилированию, аналитик приступает к «очистке» исходных данных, которая выполняется на этапе *предобработки*. Предобработка играет роль «чистки», удаляя «грязь» из данных и приводя их к форме, пригодной для дальнейшего анализа. Различные дефекты данных проявляются в виде:

- пропусков (так называемые NULL-значения);
- некорректных форматов значений (дата должна быть в формате ГОД-МЕСЯЦ-ДЕНЬ, а она записана как ДЕНЬ-МЕСЯЦ-ГОД);
- нереальных значений (возраст человека меньше 18 лет, когда предполагаются только совершеннолетние);
- неправильного типа (идентификатор — целое число, а не строка) и т. д.

Многие из таких дефектов проникают в данные из-за того, что люди, отвечающие за сбор информации, отчитываются за объем, но не качество собираемых данных. Это, в свою очередь, происходит потому, что отделы сбора данных и их аналитики (Data Science) принадлежат разным подразделениям и управляются разными руководителями. Мой богатый опыт позволяет мне

[1] Описание источников данных см. в приложении А. — *Примеч. ред.*

утверждать, что такое разделение не приводит ни к чему хорошему. Хотя отдел сбора данных возникает (намного) раньше отдела аналитики данных (но не бизнес-аналитики (Business Intelligence, BI)), часто высшее руководство организаций думает об аналитике как о средстве подготовки различных отчетов. Такой подход работает в краткосрочной перспективе, но по мере роста объемов данных количество ошибок в них растет в геометрической прогрессии. В результате нередко возникает трение по вопросу, кто отвечает за качество данных вообще и для отчетов в частности.

Во избежание возникновения большого процента «грязных» данных я бы советовал объединить инженеров, ответственных за сбор данных, и аналитиков данных в одном отделе. Тогда все претензии к аналитикам данных со стороны бизнес-аналитиков будут во многих случаях сняты, и организация получит более правдоподобные отчеты, так же как и нормальный рабочий (свободный от взаимных упреков и обвинений) климат в коллективе. Независимо от того, кто отвечает за качество данных, дефекты следует устранить, желательно до того, как бизнес-аналитики или другие пользователи данных заметят их и примчатся с гневным возгласом: «У вас опять ЧП с данными!!!» Иными словами, подход к поддержанию надлежащего качества данных обязан быть проактивным, а не реактивным! Для этого инженеры должны наладить всевозможный контроль поступающих данных до того, как те отправятся к потребителям. Иначе не избежать неприятностей и проблем. Если же контроль и исправление дефектов ложится на плечи аналитиков данных, то они могут потратить до 80 % своего рабочего времени на «очистку». Использовать высококвалифицированных специалистов в качестве «дворников» — не очень продуктивное занятие (это на заметку тем руководителям среднего и высшего звена, кто считает, что где-то существует супермодель машинного обучения, которая способна справиться с любыми «мусорными» данными без их предварительной очистки; такие люди не знают о принципе GIGO (garbage in, garbage out)).

На этапе *проектирования модели* выбирают модель для решаемой задачи, а также метрику, по которой будет оцениваться эффективность обученной модели. Выбор модели, конечно же, важен, но выбор правильной метрики может быть даже важнее. В настоящее время большие объемы данных во многом нивелируют необходимость поиска «супермодели»: бывает, что и простая модель удовлетворяет потребностям бизнеса организации, особенно когда качество данных достаточно высокое. Тем не менее метрика, удовлетворяющая запросам конечных пользователей, была, есть и будет в цене, так как по ее показателям руководство организации/компании оценивает полезность машинного обучения в целом. Поэтому мой совет аналитикам и инженерам, проектирующим системы машинного обучения, — думайте о метрике до того, как приступите к обучению модели. К тому же выбор

метрики должен быть согласован с бизнесом, чтобы не совершить так называемую ошибку третьего рода («смертный грех» для любого специалиста по машинному обучению!) — решение задачи, бесполезной для конечных пользователей, то есть не той задачи, которую эти пользователи ожидали от вас!

Имея модель, выбранную для решения задачи, можно переходить на следующий этап — приступать к *обучению этой модели и ее оптимизации согласно выбранному критерию*, например такому, как количество совершенных ошибок. Здесь критерий оптимизации модели должен либо в точности совпадать с метрикой, выбранной на предыдущем этапе, либо быть похожим (по поведению) на нее; иначе модель будет оптимизировать одну величину, а оцениваться по другой величине, с отличным от первой поведением. Корреляция двух величин укажет, изменяются они синхронно (обе одновременно возрастают или убывают при изменении одних и тех же факторов) или нет.

Следующий этап предполагает помещение модели в том месте, где она будет «работать». Это называется *развертыванием модели на конечной точке*. Пользователи модели будут «вызывать» ее, обращаясь к адресу конечной точки. Но это еще не все!

Необходимо *непрерывно отслеживать эффективность развернутой модели*, так как со временем обученная модель может «устареть» из-за того, что входные данные и/или их качество изменятся. Такие моменты необходимо своевременно обнаруживать и оперативно реагировать на них, переобучая модель на новых данных и устраняя дефекты в уже существующих.

Почему периодическое переобучение необходимо? Допустим, ваша компания занимается продажами товаров в интернете. Для этого она применяет модель рекомендации продуктов потенциальным покупателям. В прошлом целевая аудитория компании состояла из людей в возрасте 40–50 лет, но в последний месяц она резко изменилась (например, появилось много молодежи до 30 лет) и продажи среди покупателей 40–50 лет упали. Произошел *дрейф* (data drift) в характеристике данных (возрасте), и текущая модель уже не в состоянии адекватно реагировать на него — требуется переобучить ее с учетом изменившейся популяции клиентов.

Помимо дрейфа в одном или нескольких признаках данных, может произойти *дрейф концепции* (concept drift). Он происходит, когда распределение выходных классов, которые предсказывает модель, меняется. Например, микрофинансовая компания обнаруживает внезапное увеличение количества неплательщиков по микрозаймам, так что вместо приблизительно сбалансированного соотношения между двумя классами (плательщиков и неплательщиков) она видит, что это соотношение резко сдвинулось в сторону неплательщиков и составляет 90:10.

Наконец, проблемы с моделью могут возникнуть из-за *задержки* (latency) поступления данных от внешних источников: данные поступают в хранилище компании, но позднее во времени, и модель не получает их, когда делает предсказание. Такую задержку часто сложно обнаружить, особенно если модель нечувствительна к пропускам в данных, то есть абсолютно ничего не случится (модель продолжит «работу» как ни в чем не бывало), если вместо числового значения модель «увидит» `NULL` (поверьте, такие модели существуют, и отсутствие сигнала тревоги при обнаружении пропуска — негативная черта, которая при другом рассмотрении может стать и позитивной). В базе данных на момент запоздалой проверки тоже будет число, а не коварный пропуск.

На практике мониторинг часто приводит к возврату к одному или нескольким предыдущим этапам процесса машинного обучения и повторению всех последующих шагов. Так что будьте готовы к итерациям!

В наше время, когда конечные пользователи часто желают понять логику принятия решений развернутой моделью, уже недостаточно для модели быть «черным ящиком» со входом и выходом. В ряде областей, например таких, как медицина и юриспруденция, законодательство и отраслевые акты все чаще требуют понимания того, как модели принимают решения. Доктор хочет знать, почему модель, анализирующая медицинский снимок, предсказала, что пациент болен; ей недостаточно слепо доверять модели, поскольку цена ошибки высока. Адвокат, полагающийся на модель — интеллектуального помощника, нуждается в подкреплении своих доводов в суде, со ссылками на настоящие законы и решения по прошлым судебным делам; он не может просто выиграть судебный процесс, заявив, что его клиент невиновен, ссылаясь на предсказание модели (тут тоже неправильный прогноз модели будет иметь крайне негативные последствия). Поэтому понимание, почему модель выдает тот или иной прогноз, становится неотъемлемым этапом процесса ML.

Цель этой книги — предоставить введение в современные библиотеки ML на Python на основании вышеперечисленных этапов процесса ML. Я нашел, что многие темы, затронутые в книге, недостаточно хорошо освещены в отечественных и зарубежных публикациях или же они разбросаны по многим источникам. Это подвигло меня написать книгу, которую вы, уважаемые читатели, только что открыли.

В следующей главе мы рассмотрим примеры данных и задачи, связанные с ними, которые я буду использовать для объяснения материала в книге.

ГЛАВА 2
Примеры данных и практических задач, рассмотренные в книге

Как я уже говорил, эта книга была бы чисто теоретической без примеров кода, который можно было бы опробовать на данных. В большинстве случаев я решил объяснять концепции на искусственно сгенерированных данных, поскольку им можно придать любые характеристики. Такие данные подобны сырой глине, которой гончар придает желаемую форму. В то же время, чтобы не создавать у неофитов впечатление, что машинное обучение — это всего лишь красивая теория, я добавил в эту главу пару реальных задач и ассоциированные с ними данные:

- несбалансированную классификацию (обнаружение мошенничества);
- прогнозирование временных рядов и выявление аномалий (курс биткойна).

Различные типы задач машинного обучения описаны в приложении Г.

2.1. Обнаружение мошенничества

Первая задача принадлежит к типу несбалансированной классификации. Этот класс задач часто возникает, когда требуется своевременно обнаружить мошенничество в действиях людей и компаний. Классификация означает, что имеется два класса данных, соответствующих нормальному и мошенническому поведению. Эти классы присутствуют в обучающей и тестовой

выборках. Возможен сценарий, когда в обучающей выборке есть только данные нормального поведения. Но в рассматриваемом примере данных имеются оба класса.

Зачастую данных, описывающих мошенническое поведение, — мало. Точнее, размер «нормального» класса намного больше размера «ненормального»; поэтому классификация называется *несбалансированной*. Соотношение двух классов может быть 10:1, 100:1, 1000:1 и более. Чем выше дисбаланс, тем больше вероятность, что количество случаев ошибочного распознавания нормального поведения как мошеннического будет увеличиваться. Но это ожидаемо: ведь главное — обнаружить все или как можно больше случаев мошенничества, поскольку необнаруженная махинация означает чьи-то финансовые потери, в то время как небольшой процент ошибок распознавания нормального поведения как мошеннического допусти́м, если беспокойство из-за этого допущения не является значительной проблемой. Как правило, процент ошибки задается изначально до обучения модели, и эффективность обнаружения ею мошенничества определяется при достижении этого процента.

Скачать данные для первой задачи можно с сайта Kaggle: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud (для этого потребуется зарегистрироваться на платформе Kaggle!). Этот набор данных описывает задачу обнаружения мошенничества с кредитными картами и состоит из 284 807 примеров нормальных транзакций и 492 примеров махинаций (соотношение двух классов — приблизительно 579:1). Из-за требования конфиденциальности большинство исходных числовых признаков закодированы с помощью линейного преобразования (кажется, это старый добрый анализ главных компонент (Principal Component Analysis, PCA)). Таким образом, в наборе есть 28 закодированных числовых признаков и 2 незакодированных — `Time` и `Amount`.

Time обозначает время в секундах, прошедшее между каждой операцией с кредитной картой и первой по времени операцией во всем наборе данных. `Amount` соответствует сумме каждой операции.

Кроме того, в наборе есть еще один признак — `Class`. Он обозначает легитимность операции: значение `0` — операция легитимна, значение `1` — нет. Поскольку `Class` — это цель предсказания, никакой кодировки не требуется.

Обратите внимание: для этой задачи можно генерировать дополнительные данные. Авторы указанного набора привели ссылку на симулятор данных.

2.2. Прогнозирование временных рядов и выявление аномалий

Второй пример состоит из двух задач:

1) регрессии или предсказания на основе временного ряда значений;
2) выявления аномалий во временны́х данных.

Реальные данные — ежедневный курс биткойна (BTC) к американскому доллару (USD). Биткойн — мировой лидер среди криптовалют. Его история кратко описана на сайте https://bytwork.com/articles/btc-chart-history, а динамика курса по отношению к американскому доллару показана на рис. 2.1.

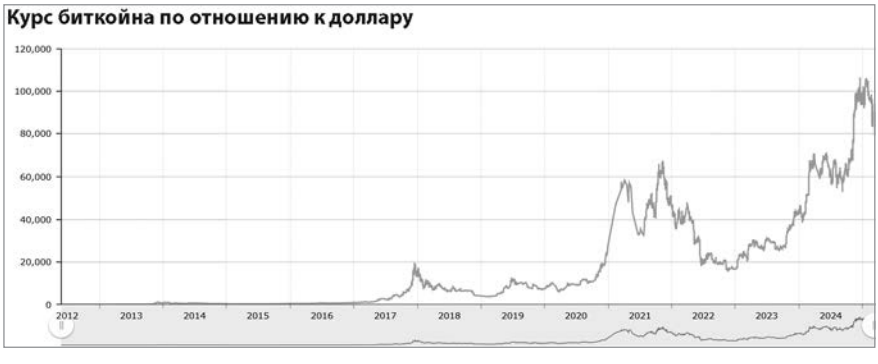


Рис. 2.1. Курс биткойна по отношению к американскому доллару за всю историю его существования (источник: https://mainfin.ru/crypto/chart-bitcoin)

Исходные данные — одномерные; одно значение на каждый день. Как видно из графика, сначала курс был очень низким и почти не колебался, но за последние несколько лет его волатильность резко возросла вместе с популярностью этой криптовалюты. Из-за поведения курса я и выбрал его для анализа — прогнозирование усложняется волатильностью и есть масса «аномалий». Мой выбор не продиктован персональными соображениями, и все выводы из анализа не могут считаться советом по инвестициям.

Весь временной интервал неинтересен для анализа. Поэтому я решил сосредоточиться на отрезке «весна 2017 года — сегодняшний день» (10 марта 2025 года на момент написания этого текста). Многие ключевые события, повлиявшие на курс биткойна, произошли в 2024 году, поэтому на рис. 2.2 я привел еще один график. Вы можете заметить, что прогноз по будущему

курсу значительно усложняется тем, что на курс оказывают влияние такие (сторонние) факторы, как возврат украденной криптовалюты или выборы в США, а также психологическое чувство комфорта или дискомфорта владельцев криптовалюты. В общем, торговля «криптой», по моему мнению, — это «форекс-трейдинг на стероидах» (риск, возведенный в квадрат).

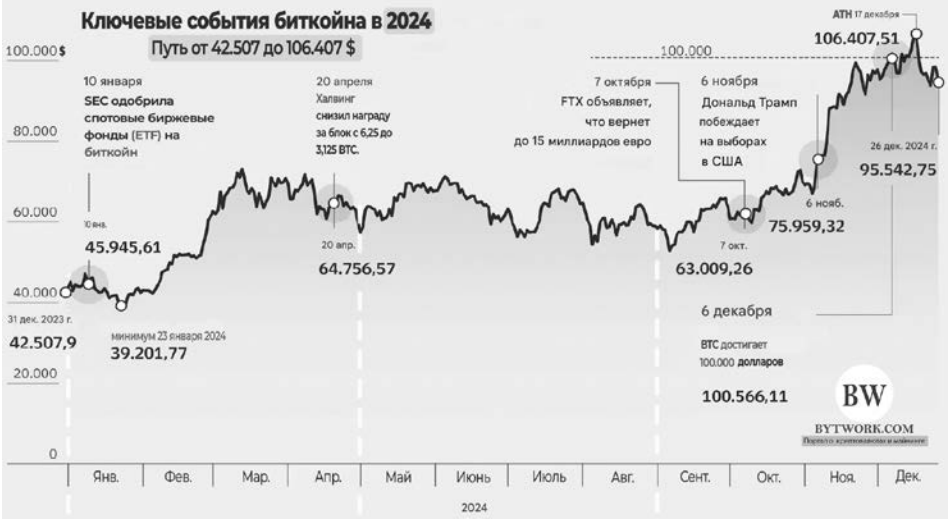


Рис. 2.2. Ключевые события, повлиявшие на курс биткойна в 2024 году (источник: https://bytwork.com/articles/btc-chart-history)

Данные этого примера я вручную скачал с сайта https://www.marketwatch.com/investing/cryptocurrency/btcusd/download-data?mod=mw_quote_tab восемью порциями из-за ограничения на размер получаемых данных, а затем собрал их в один файл. Его можно найти на сайте издательства «Питер»: https://storage.piter.com/addons/978544614561/Biblioteki_ML_Code.zip.

Прогноз или предсказание будущего курса финансового актива (не только биткойна, но и, например, акций компании) — задача обучения с учителем (см. приложение Г). В такой задаче имеющиеся данные делятся на две-три группы или выборки. Разделение всегда происходит по дате, а не случайным образом; ведь нам надо предсказать «будущее» значение, а это значит, что наша регрессионная модель не должна знать ничего о будущих во времени значениях в момент их предсказания или до него.

Обнаружение аномалии (одной или нескольких) похоже на задачу из предыдущего раздела. Однако мы будем использовать другие подходы, поскольку при анализе курса биткойна имеем дело с одномерным сигналом.

ЧАСТЬ II
Подготовка к программированию на Python

В этой части мы рассмотрим вопросы организации работы с Python. Вы получите детальные инструкции и рекомендации, позволяющие начать программировать без проблем. В главе 3 подробно описано, как установить последнюю рабочую версию Python и интегрированную среду разработки программ, в которой мы будем писать код и запускать его на выполнение. Кроме того, в этой же главе вы познакомитесь с рекомендуемой организацией папок нашего проекта, которые будут наполняться кодом и данными по мере вашего продвижения от главы к главе.