

# Содержание

Об авторах	17
<b>Предисловие</b>	20
Начало работы с машинным обучением	20
Практика и теория	20
Почему был выбран язык Python?	21
Исследование области машинного обучения	21
Для кого предназначена эта книга?	22
Что рассматривается в этой книге?	22
Что необходимо при работе с этой книгой?	26
Соглашения	26
Загрузка кода примеров	28
Ждем ваших отзывов!	28
<b>Глава 1. Наделение компьютеров способностью обучения на данных</b>	29
Построение интеллектуальных машин для трансформирования данных в знания	30
Три типа машинного обучения	30
Выработка прогнозов о будущем с помощью обучения с учителем	31
Решение интерактивных задач с помощью обучения с подкреплением	34
Обнаружение скрытых структур с помощью обучения без учителя	36
Введение в основную терминологию и обозначения	38
Обозначения и соглашения, используемые в книге	39
Терминология, связанная с машинным обучением	41
Дорожная карта для построения систем машинного обучения	42
Предварительная обработка — приведение данных в приемлемую форму	43
Обучение и выбор прогнозирующей модели	44
Оценка моделей и прогнозирование на не встречавшихся ранее образцах данных	45
Использование Python для машинного обучения	45
Установка Python и необходимых пакетов	46
Использование дистрибутива Anaconda и диспетчера пакетов	46
Пакеты для научных расчетов, науки о данных и машинного обучения	47
Резюме	48
<b>Глава 2. Обучение простых алгоритмов МО для классификации</b>	49
Искусственные нейроны — беглое знакомство с ранней историей машинного обучения	50
Формальное определение искусственного нейрона	51
Правило обучения персептрона	53

Реализация алгоритма обучения перцептрона на Python	56
Объектно-ориентированный API-интерфейс перцептрона	56
Обучение модели перцептрона на наборе данных Iris	60
Адаптивные линейные нейроны и сходимость обучения	67
Минимизация функций издержек с помощью градиентного спуска	68
Реализация Adaline на Python	71
Улучшение градиентного спуска посредством масштабирования признаков	75
Крупномасштабное машинное обучение и стохастический градиентный спуск	78
Резюме	84
<b>Глава 3. Обзор классификаторов на основе машинного обучения с использованием scikit-learn</b>	85
Выбор алгоритма классификации	86
Первые шаги в освоении scikit-learn — обучение перцептрона	87
Моделирование вероятностей классов посредством логистической регрессии	93
Понятие логистической регрессии и условные вероятности	94
Выяснение весов функции издержек для логистической регрессии	98
Преобразование реализации Adaline в алгоритм для логистической регрессии	101
Обучение логистической регрессионной модели с помощью scikit-learn	106
Решение проблемы переобучения с помощью регуляризации	109
Классификация с максимальным зазором с помощью методов опорных векторов	113
Понятие максимального зазора	113
Обработка нелинейно сепарабельного случая с использованием фиктивных переменных	115
Альтернативные реализации в scikit-learn	117
Решение нелинейных задач с применением ядерного метода опорных векторов	118
Ядерные методы для линейно сепарабельных данных	118
Использование ядерного трюка для нахождения разделяющих гиперплоскостей в пространстве высокой размерности	121
Обучение моделей на основе деревьев принятия решений	124
Доведение до максимума прироста информации — получение наибольшей отдачи	126
Построение дерева принятия решений	131
Объединение множества деревьев принятия решений с помощью случайных лесов	135
Метод $k$ ближайших соседей — алгоритм ленивого обучения	139
Резюме	143
<b>Глава 4. Построение хороших обучающих наборов — предварительная обработка данных</b>	145
Решение проблемы с недостающими данными	146
Идентификация недостающих значений в табличных данных	146

Исключение обучающих образцов или признаков с недостающими значениями	148
Условный расчет недостающих значений	149
Понятие API-интерфейса оценщиков scikit-learn	150
Обработка категориальных данных	151
Кодирование категориальных данных с помощью pandas	152
Отображение порядковых признаков	153
Кодирование меток классов	154
Выполнение унитарного кодирования на именных признаках	155
Разбиение набора данных на отдельные обучающий и испытательный наборы	159
Приведение признаков к тому же самому масштабу	162
Выбор значимых признаков	165
Регуляризация L1 и L2 как штрафы за сложность модели	166
Геометрическая интерпретация регуляризации L2	166
Разреженные решения с регуляризацией L1	169
Алгоритмы последовательного выбора признаков	173
Оценка важности признаков с помощью случайных лесов	180
Резюме	183
<b>Глава 5. Сжатие данных с помощью понижения размерности</b>	185
Понижение размерности без учителя с помощью анализа главных компонент	186
Основные шаги при анализе главных компонент	186
Выделение главных компонент шаг за шагом	188
Полная и объясненная дисперсия	191
Трансформация признаков	193
Анализ главных компонент в scikit-learn	197
Сжатие данных с учителем посредством линейного дискриминантного анализа	200
Анализ главных компонент в сравнении с линейным дискриминантным анализом	200
Внутреннее устройство линейного дискриминантного анализа	202
Вычисление матриц рассеяния	203
Выбор линейных дискриминантов для нового подпространства признаков	205
Процирование образцов в новое подпространство признаков	208
Реализация LDA в scikit-learn	209
Использование ядерного анализа главных компонент для нелинейных отображающих функций	211
Ядерные функции и ядерный трюк	212
Реализация ядерного анализа главных компонент на языке Python	217
Процирование новых точек данных	225
Ядерный анализ главных компонент в scikit-learn	229
Резюме	231

<b>Глава 6. Освоение практического опыта оценки моделей и настройки гиперпараметров</b>	233
Модернизация рабочих потоков с помощью конвейеров	234
Загрузка набора данных Breast Cancer Wisconsin	234
Объединение преобразователей и оценщиков в конвейер	236
Использование перекрестной проверки по $k$ блокам для оценки эффективности модели	238
Метод перекрестной проверки с удержанием	239
Перекрестная проверка по $k$ блокам	240
Отладка алгоритмов с помощью кривых обучения и проверки	245
Диагностирование проблем со смещением и дисперсией с помощью кривых обучения	246
Решение проблем недообучения и переобучения с помощью кривых проверки	250
Точная настройка моделей машинного обучения с помощью решетчатого поиска	252
Настройка гиперпараметров с помощью решетчатого поиска	253
Отбор алгоритма с помощью вложенной перекрестной проверки	255
Использование других метрик оценки эффективности	257
Чтение матрицы неточностей	258
Оптимизация точности и полноты классификационной модели	260
Построение кривой рабочей характеристики приемника	263
Метрики подсчета для многоклассовой классификации	266
Решение проблемы с дисбалансом классов	267
Резюме	271
<b>Глава 7. Объединение разных моделей для ансамблевого обучения</b>	273
Обучение с помощью ансамблей	273
Объединение классификаторов с помощью мажоритарного голосования	278
Реализация простого классификатора с мажоритарным голосованием	279
Использование принципа мажоритарного голосования для выработки прогнозов	285
Оценка и настройка ансамблевого классификатора	289
Бэггинг — построение ансамбля классификаторов из бутстрэп-образцов	296
Коротко о бэггинге	297
Применение бэггинга для классификации образцов в наборе данных Wine	298
Использование в своих интересах слабых учеников посредством адаптивного бустинга	302
Как работает бустинг	303
Применение алгоритма AdaBoost с помощью scikit-learn	308
Резюме	312

---

<b>Глава 8. Применение машинного обучения для смыслового анализа</b>	313
Подготовка данных с рецензиями на фильмы IMDb для обработки текста	314
Получение набора данных с рецензиями на фильмы	314
Предварительная обработка набора данных с целью приведения в более удобный формат	315
Модель суммирования слов	317
Трансформирование слов в векторы признаков	318
Оценка важности слов с помощью приема <code>tf-idf</code>	320
Очистка текстовых данных	323
Переработка документов в лексемы	325
Обучение логистической регрессионной модели для классификации документов	328
Работа с более крупными данными — динамические алгоритмы и внешнее обучение	331
Тематическое моделирование с помощью латентного размещения Дирихле	335
Разбиение текстовых документов с помощью LDA	336
Реализация LDA в библиотеке <code>scikit-learn</code>	337
Резюме	341
<b>Глава 9. Встраивание модели машинного обучения в веб-приложение</b>	343
Сериализация подогнанных оценщиков <code>scikit-learn</code>	344
Настройка базы данных SQLite для хранилища данных	348
Разработка веб-приложения с помощью Flask	350
Первое веб-приложение Flask	351
Проверка достоверности и визуализация форм	353
Превращение классификатора рецензий на фильмы в веб-приложение	360
Файлы и подкаталоги — дерево каталогов	362
Реализация главного приложения как <code>app.py</code>	363
Настройка формы для рецензии	366
Создание шаблона страницы результатов	367
Развертывание веб-приложения на публичном сервере	370
Создание учетной записи PythonAnywhere	370
Загрузка файлов для приложения классификации рецензий на фильмы	370
Обновление классификатора рецензий на фильмы	372
Резюме	375
<b>Глава 10. Прогнозирование значений непрерывных целевых переменных с помощью регрессионного анализа</b>	377
Ведение в линейную регрессию	378
Простая линейная регрессия	378
Множественная линейная регрессия	379
Исследование набора данных Housing	381
Загрузка набора данных Housing в объект <code>DataFrame</code>	381

---

Визуализация важных характеристик набора данных	383
Просмотр взаимосвязей с использованием корреляционной матрицы	385
Реализация линейной регрессионной модели с использованием обычного метода наименьших квадратов	388
Использование градиентного спуска для выяснения параметров регрессии	389
Оценка коэффициентов регрессионной модели с помощью scikit-learn	393
Подгонка надежной регрессионной модели с использованием RANSAC	396
Оценка эффективности линейных регрессионных моделей	399
Использование регуляризованных методов для регрессии	403
Превращение линейной регрессионной модели в криволинейную — полиномиальная регрессия	405
Добавление полиномиальных членов с использованием scikit-learn	405
Моделирование нелинейных связей в наборе данных Housing	407
Обработка нелинейных связей с использованием случайных лесов	410
Регрессия на основе дерева принятия решений	411
Регрессия на основе случайного леса	413
Резюме	417
<b>Глава 11. Работа с непомеченными данными — кластерный анализ</b>	419
Группирование объектов по подобию с применением алгоритма K-Means	420
Кластеризация K-Means с использованием scikit-learn	420
Более интеллектуальный способ размещения начальных центроидов кластеров с использованием алгоритма K-Means++	426
Жесткая или мягкая кластеризация	427
Использование метода локтя для нахождения оптимального количества кластеров	430
Количественная оценка качества кластеризации через графики силуэтов	431
Организация кластеров в виде иерархического дерева	436
Группирование кластеров в восходящей манере	437
Выполнение иерархической кластеризации на матрице расстояний	439
Прикрепление дендрограмм к тепловой карте	443
Применение агломеративной иерархической кластеризации с помощью scikit-learn	445
Нахождение областей высокой плотности с помощью DBSCAN	446
Резюме	452
<b>Глава 12. Реализация многослойной искусственной нейронной сети с нуля</b>	455
Моделирование сложных функций с помощью искусственных нейронных сетей	456
Краткое повторение однослойных нейронных сетей	458
Введение в архитектуру многослойных нейронных сетей	461
Активация нейронной сети посредством прямого распространения	464
Классификация рукописных цифр	467

---

Получение и подготовка набора данных MNIST	468
Реализация многослойного персептрона	476
Обучение искусственной нейронной сети	488
Вычисление логистической функции издержек	488
Выработка общего понимания обратного распространения	491
Обучение нейронных сетей с помощью обратного распространения	493
О сходимости в нейронных сетях	498
Несколько слов о реализации нейронных сетей	499
Резюме	500
<b>Глава 13. Распараллеливание процесса обучения нейронных сетей с помощью TensorFlow</b>	501
TensorFlow и производительность обучения	502
Проблемы, связанные с производительностью	502
Что такое TensorFlow?	504
Как мы будем изучать TensorFlow	506
Первые шаги при работе с библиотекой TensorFlow	506
Установка TensorFlow	506
Создание тензоров в TensorFlow	507
Манипулирование типом данных и формой тензора	508
Применение математических операций к тензорам	509
Расщепление, укладывание стопкой и объединение тензоров	511
Построение входных конвейеров с использованием <code>tf.data</code> — API-интерфейса <code>Dataset</code> библиотеки TensorFlow	513
Создание объекта <code>Dataset</code> из существующих тензоров	514
Объединение двух тензоров в общий набор данных	515
Тасование, создание пакетов и повторение	516
Создание набора данных из файлов на локальном диске	520
Извлечение доступных наборов данных из библиотеки <code>tensorflow_datasets</code>	524
Построение нейросетевой модели в TensorFlow	530
API-интерфейс Keras в TensorFlow ( <code>tf.keras</code> )	530
Построение линейной регрессионной модели	531
Обучение модели с помощью методов <code>.compile()</code> и <code>.fit()</code>	536
Построение многослойного персептрона для классификации цветков в наборе данных Iris	538
Оценка обученной модели на испытательном наборе данных	542
Сохранение и повторная загрузка обученной модели	543
Выбор функций активации для многослойных нейронных сетей	544
Краткое повторение логистической функции	545
Оценка вероятностей классов в многоклассовой классификации через многопеременную логистическую функцию	547
Расширение выходного спектра с использованием гиперболического тангенса	548

---

Активация на основе выпрямленного линейного элемента	551
Резюме	553
<b>Глава 14. Погружаемся глубже — механика TensorFlow</b>	<b>555</b>
Ключевые средства TensorFlow	556
Вычислительные графы TensorFlow: переход на TensorFlow v2	558
Понятие вычислительных графов	558
Создание графа в TensorFlow v1.x	559
Перенос графа в TensorFlow v2	560
Загрузка входных данных в модель: стиль TensorFlow v1.x	561
Загрузка входных данных в модель: стиль TensorFlow v2	561
Увеличение вычислительной мощности с помощью декораторов функций	562
Объекты <code>Variable</code> библиотеки TensorFlow для хранения и обновления параметров модели	565
Расчет градиентов посредством автоматического дифференцирования и <code>GradientTape</code>	569
Расчет градиентов потери по отношению к обучаемым переменным	570
Расчет градиентов по отношению к необучаемым тензорам	571
Сохранение ресурсов для множества вычислений градиентов	572
Упрощение реализаций распространенных архитектур посредством API-интерфейса Keras	573
Решение задачи классификации XOR	577
Увеличение гибкости построения моделей с помощью функционального API-интерфейса Keras	583
Реализация моделей на основе класса <code>Model</code> библиотеки Keras	584
Реализация специальных слоев Keras	586
Оценщики TensorFlow	590
Работа со столбцами признаков	591
Машинное обучение с использованием готовых оценщиков	596
Использование оценщиков для классификации рукописных цифр MNIST	601
Создание специального оценщика из существующей модели Keras	604
Резюме	606
<b>Глава 15. Классификация изображений с помощью глубоких сверточных нейронных сетей</b>	<b>609</b>
Строительные блоки сверточных нейронных сетей	610
Понятие сетей CNN и иерархий признаков	611
Выполнение дискретных сверток	613
Слой подвыборки	624
Группирование всего вместе — реализация сверточной нейронной сети	626
Работа с множественными входными или цветовыми каналами	627
Регуляризация нейронной сети с помощью отключения	631



---

Функции потерь для классификации	635
Реализация глубокой сверточной нейронной сети с использованием TensorFlow	638
Архитектура многослойной сверточной нейронной сети	638
Загрузка и предварительная обработка данных	639
Реализация сверточной нейронной сети с использованием API-интерфейса Keras библиотеки TensorFlow	641
Классификация полов по изображениям лиц с использованием сверточной нейронной сети	648
Загрузка набора данных CelebA	648
Трансформация изображений и дополнение данных	649
Обучение классификатора полов, основанного на сверточной нейронной сети	656
Резюме	662
<b>Глава 16. Моделирование последовательных данных с использованием рекуррентных нейронных сетей</b>	665
Понятие последовательных данных	666
Моделирование последовательных данных — вопросы порядка	666
Представление последовательностей	667
Категории моделирования последовательностей	668
Рекуррентные нейронные сети для моделирования последовательностей	670
Механизм организации циклов рекуррентной нейронной сети	670
Вычисление активаций в сети RNN	673
Рекуррентность скрытого слоя или рекуррентность выходного слоя	676
Сложности изучения долгосрочных взаимодействий	680
Ячейки долгой краткосрочной памяти	681
Реализация многослойных рекуррентных нейронных сетей для моделирования последовательностей в TensorFlow	684
Проект номер один — прогнозирование отношения в рецензиях на фильмы IMDb	685
Подготовка данных с рецензиями на фильмы	685
Слои вложений для кодирования предложений	691
Построение модели на основе рекуррентной нейронной сети	694
Проект номер два — моделирование языка на уровне символов в TensorFlow	702
Понимание языка с помощью модели “Преобразователь”	716
Механизм самовнимания	717
Многоголовое внимание и блок “Преобразователь”	720
Резюме	722
<b>Глава 17. Порождающие состязательные сети для синтеза новых данных</b>	723
Понятие порождающих состязательных сетей	724
Начало работы с автокодировщиками	725
Порождающие модели для синтеза новых данных	727
Генерирование новых образцов с помощью порождающих состязательных сетей	729

Функции потерь сетей генератора и дискриминатора в модели GAN	731
Реализация порождающей состязательной сети с нуля	733
Обучение моделей GAN в среде Google Colab	734
Реализация сетей генератора и дискриминатора	737
Определение обучающего набора данных	742
Обучение модели GAN	744
Повышение качества синтезированных изображений с использованием сверточной сети GAN и сети GAN Вассерштейна	753
Транспонированная свертка	753
Пакетная нормализация	756
Реализация генератора и дискриминатора	759
Меры несходства между двумя распределениями	766
Практическое использование расстояния Вассерштейна для порождающих состязательных сетей	770
Штраф градиента	771
Реализация порождающей состязательной сети Вассерштейна со штрафом градиента для обучения модели DCGAN	772
Коллапс мод	777
Другие приложения порождающих состязательных сетей	778
Резюме	780
<b>Глава 18. Обучение с подкреплением для принятия решений в сложных средах</b>	<b>781</b>
Введение — обучение на опыте	782
Понятие обучения с подкреплением	782
Определение интерфейса “агент–среда” системы обучения с подкреплением	785
Теоретические основы обучения с подкреплением	786
Марковские процессы принятия решений	787
Математическая формулировка марковских процессов принятия решений	787
Терминология обучения с подкреплением: отдача, политика и функция ценности	791
Динамическое программирование с использованием уравнения Беллмана	796
Алгоритмы обучения с подкреплением	797
Динамическое программирование	798
Обучение с подкреплением с помощью метода Монте-Карло	801
Обучение методом временных разностей	804
Реализация первого алгоритма обучения с подкреплением	808
Введение в комплект инструментов OpenAI Gym	808
Решение задачи с миром сетки с помощью Q-обучения	818
Обзор глубокого Q-обучения	823
Резюме по главе и по книге	832
<b>Предметный указатель</b>	<b>835</b>

# ВСТРАИВАНИЕ МОДЕЛИ МАШИННОГО ОБУЧЕНИЯ В ВЕБ-ПРИЛОЖЕНИЕ

**В** предшествующих главах вы ознакомились со многими концепциями и алгоритмами МО, которые могут содействовать принятию лучших и более рациональных решений. Однако приемы МО не ограничиваются автономными приложениями и анализом; они стали прогнозирующими механизмами веб-служб. Например, популярные и удобные случаи употребления моделей МО в веб-приложениях включают обнаружение спама в отправляемых формах, поисковые механизмы, системы выдачи рекомендаций для медийных и продающих порталов, а также многие другие.

В этой главе вы узнаете, как встраивать модель МО в веб-приложение, которое способно не только классифицировать, но и учиться на данных в реальном времени.

В главе будут раскрыты следующие темы:

- сохранение текущего состояния обученной модели МО;
- использование баз данных SQLite для хранилищ данных;
- разработка веб-приложения с применением популярного веб-фреймворка Flask;
- развертывание приложения МО на публичном веб-сервере.

## Сериализация подогнанных оценщиков `scikit-learn`

Как было показано в главе 8, обучение модели МО может оказаться довольно затратным с точки зрения вычислений. Ведь не хотим же мы обучать модель заново каждый раз, когда закрыли интерпретатор Python и желаем выработать прогноз или перезагрузить веб-приложение?

Одним из вариантов обеспечения постоянства моделей является модуль `pickle` для Python (<https://docs.python.org/3.7/library/pickle.html>). Он делает возможными сериализацию и десериализацию структур объектов Python с целью сжатия байт-кода, чтобы мы могли сохранить классификатор в текущем состоянии и опять загрузить его, когда нужно классифицировать новые непомеченные образцы, не заставляя модель учиться на обучающих данных еще раз. Прежде чем выполнять следующий код, удостоверьтесь в том, что обучили внешнюю логистическую регрессионную модель из последнего раздела главы 8 и обеспечили ее готовность в текущем сеансе Python:

```
>>> import pickle
>>> import os
>>> dest = os.path.join('movieclassifier', 'pkl_objects')
>>> if not os.path.exists(dest):
...     os.makedirs(dest)
>>> pickle.dump(stop,
...             open(os.path.join(dest, 'stopwords.pkl'), 'wb'),
...             protocol=4)
>>> pickle.dump(clf,
...             open(os.path.join(dest, 'classifier.pkl'), 'wb'),
...             protocol=4)
```

В приведенном выше коде мы создаем каталог `movieclassifier`, где позже будем хранить файлы и данные для нашего веб-приложения. Внутри каталога `movieclassifier` мы создаем подкаталог `pkl_objects`, чтобы сохранять на локальном жестком диске или твердотельном накопителе сериализованные объекты Python. С помощью метода `dump` модуля `pickle` мы затем сериализуем обученную логистическую регрессионную модель, а также набор стоп-слов из библиотеки NLTK, чтобы устранить необходимость в установке глоссария NLTK на сервере.

Метод `dump` принимает в своем первом аргументе объект, который мы хотим законсервировать. Во втором аргументе предоставляется открытый

файловый объект, куда будет записываться объект Python. Посредством аргумента `wb` внутри функции `open` мы открываем файл в двоичном режиме для консервирования и устанавливаем `protocol=4`, чтобы выбрать самый последний и эффективный протокол консервирования, который появился в Python 3.4 и совместим с последующими версиями Python. В случае возникновения проблем с использованием `protocol=4` проверьте, работаете ли вы с самой последней версией Python 3 — в этой книге рекомендуется Python 3.7. Или же можете обдумать вопрос применения протокола с меньшим номером.

Также имейте в виду, что если вы имеете дело со специальным веб-сервером, то должны также проверить, совместима ли установленная на нем копия Python с версией протокола.



На заметку!

### Сериализация массивов NumPy с помощью библиотеки `joblib`

Наша логистическая регрессионная модель содержит несколько массивов NumPy, таких как весовой вектор, а более рациональный способ сериализации массивов NumPy предусматривает использование альтернативной библиотеки `joblib`. Чтобы обеспечить совместимость с серверной средой, которая будет применяться позже в главе, мы воспользуемся стандартным подходом к консервированию. Дополнительные сведения о библиотеке `joblib` доступны по ссылке <https://joblib.readthedocs.io>.

Консервировать объект `HashingVectorizer` нет никакой необходимости, потому что он не нуждается в подгонке. Взамен мы можем создать файл сценария Python, из которого импортировать векторизатор в текущий сеанс Python. Скопируем следующий код и сохраним его в файле `vectorizer.py` внутри каталога `movieclassifier`:

```
from sklearn.feature_extraction.text import HashingVectorizer
import re
import os
import pickle

cur_dir = os.path.dirname(__file__)
stop = pickle.load(open(os.path.join(
    cur_dir, 'pkl_objects', 'stopwords.pkl'),
    'rb'))
```

```
def tokenizer(text):
    text = re.sub('<[>]*>', '', text)
    emoticons = re.findall('(?::|;|=) (?:-)?(?:\)|\(|D|P)',
                           text.lower())
    text = re.sub('[\W]+', ' ', text.lower()) \
           + ' '.join(emoticons).replace('-', '')
    tokenized = [w for w in text.split() if w not in stop]
    return tokenized

vect = HashingVectorizer(decode_error='ignore',
                        n_features=2**21,
                        preprocessor=None,
                        tokenizer=tokenizer)
```

После консервирования объектов Python и создания файла `vectorizer.py` имеет смысл перезапустить интерпретатор Python или ядро Jupyter Notebook, чтобы проверить, можем ли мы безошибочно десериализовать объекты.



На заметку!

### Консервирование может быть сопряжено с риском в плане безопасности

Тем не менее, имейте в виду, что расконсервирование данных из не заслуживающего доверия источника может быть сопряжено с риском в плане безопасности, т.к. модуль `pickle` не защищен против злонамеренного кода. Поскольку `pickle` проектировался для сериализации произвольных объектов, процесс расконсервирования выполнит код, который был сохранен в консервированном файле. Таким образом, если вы получаете консервированные файлы из не заслуживающего доверия источника (скажем, загружая их из Интернета), тогда предусмотрите дополнительные меры предосторожности — расконсервируйте элементы в виртуальной среде и/или на второстепенной машине, где отсутствуют важные данные, к которым никто кроме вас не должен иметь доступ.

В терминальном окне перейдем в каталог `movieclassifier`, запустим новый сеанс Python и выполним приведенный ниже код для проверки, можем ли мы импортировать `vectorizer` и расконсервировать классификатор:

```
>>> import pickle
>>> import re
>>> import os
>>> from vectorizer import vect
```

```
>>> clf = pickle.load(open(os.path.join(
...     'pk1_objects', 'classifier.pkl'),
...     'rb'))
```

После успешной загрузки `vectorizer` и расконсервирования классификатора мы можем использовать эти объекты для предварительной обработки образцов документов и выработки прогнозов об их отношениях:

```
>>> import numpy as np
>>> label = {0: 'негативный', 1: 'позитивный'}
>>> example = ["I love this movie. It's amazing."]
>>> X = vect.transform(example)
>>> print('Прогноз: %s\nВероятность: %.2f%%' %\
...     (label[clf.predict(X)[0]],
...     np.max(clf.predict_proba(X))*100))
Прогноз: позитивный
Вероятность: 95.55%
```

Поскольку наш классификатор возвращает спрогнозированные метки классов как целые числа, мы определили простой словарь Python для отображения таких целых чисел на их отношения ('негативный' или 'позитивный'). Хотя в итоге получилось простое приложение только с двумя классами, необходимо отметить, что подход с отображением посредством словаря также обобщается на многоклассовые конфигурации. Кроме того, такой словарь отображения должен архивироваться вместе с моделью.

В этом случае из-за того, что определение словаря включает лишь одну строку кода, мы не будем прилагать усилия по его сериализации с применением `pickle`. Однако в реальных приложениях с более обширными словарями отображения вы можете задействовать те же самые команды `pickle.dump` и `pickle.load`, которые использовались в предыдущем примере кода.

Продолжая обсуждение предыдущего примера кода, мы затем применили `HashingVectorizer` для трансформации простого примера документа в вектор слов `X`. Наконец, мы использовали метод `predict` классификатора на основе логистической регрессии для прогнозирования метки класса, а также метод `predict_proba` для возвращения соответствующей вероятности прогноза. Обратите внимание, что в результате вызова метода `predict_proba` возвращается массив со значениями вероятностей для всех уникальных метод классов. Так как метка класса с наибольшей вероятностью соответствует

метке класса, возвращаемой методом `predict`, для возвращения вероятности спрогнозированного класса мы применили функцию `np.max`.

## Настройка базы данных SQLite для хранилища данных

В текущем разделе мы настроим простую базу данных SQLite для сбора дополнительных отзывает о прогнозах от пользователей веб-приложения. Мы можем использовать такую обратную связь для обновления классификационной модели. SQLite — это механизм баз данных SQL с открытым кодом, не требующий для своей работы отдельного сервера, что делает его идеальным вариантом при разработке небольших проектов и простых веб-приложений. По существу базу данных SQLite можно считать одиночным самодостаточным файлом базы данных, который позволяет напрямую обращаться к хранилищу.

Кроме того, SQLite не требует конфигурирования, специфичного для системы, и поддерживается во всех распространенных операционных системах. Механизм SQLite известен своей высокой надежностью и применяется популярными компаниями, среди которых Google, Mozilla, Adobe, Apple, Microsoft и многие другие. Дополнительная информация о механизме SQLite доступна на официальном веб-сайте <http://www.sqlite.org>.

К счастью, согласно принятой в Python философии “батарейки в комплекте” стандартная библиотека Python предлагает API-интерфейс `sqlite3`, который позволяет работать с базами данных SQLite. (Больше сведений о модуле `sqlite3` можно получить по ссылке <https://docs.python.org/3.7/library/sqlite3.html>.)

Выполнив следующий код, мы создадим внутри каталога `movieclassifier` новую базу данных SQLite и сохраним в ней два образца рецензий на фильмы:

```
>>> import sqlite3
>>> import os

>>> conn = sqlite3.connect('reviews.sqlite')
>>> c = conn.cursor()
>>> c.execute('DROP TABLE IF EXISTS review_db')
>>> c.execute('CREATE TABLE review_db\'
...           ' (review TEXT, sentiment INTEGER, date TEXT)')
```



```
>>> example1 = 'I love this movie'
>>> c.execute("INSERT INTO review_db\"
...           " (review, sentiment, date) VALUES\"
...           " (?, ?, DATETIME('now'))", (example1, 1))
>>> example2 = 'I disliked this movie'
>>> c.execute("INSERT INTO review_db\"
...           " (review, sentiment, date) VALUES\"
...           " (?, ?, DATETIME('now'))", (example2, 0))
>>> conn.commit()
>>> conn.close()
```

В коде мы создаем подключение (`conn`) к файлу базы данных SQLite, вызывая метод `connect` из библиотеки `sqlite3`, который создает в каталоге `movieclassifier` файл базы данных `reviews.sqlite`, если он пока не существует.

Затем посредством метода `cursor` мы создаем курсор, который даст возможность проходить по записям базы данных, используя универсальный синтаксис SQL. Далее с применением первого вызова `execute` мы создаем новую таблицу базы данных `review_db`, которую будем использовать для сохранения и последующего доступа к записям. В таблице `review_db` мы создаем три столбца: `review`, `sentiment` и `date`. Они будут применяться для хранения примеров рецензий на фильмы и соответствующих меток классов (отношений).

С использованием SQL-команды `DATETIME('now')` мы добавляем к записям дату и отметку времени. Знаки вопроса (?) применяются для передачи методу `execute` текстов рецензий на фильмы (`example1` и `example2`) и связанных с ними меток классов (1 и 0) в виде позиционных аргументов как членов кортежа. В заключение мы вызываем метод `commit` для сохранения изменений, внесенных в базу данных, и закрываем подключение посредством метода `close`.

Чтобы проверить, корректно ли сохранились записи в базе данных, мы снова откроем подключение к базе данных и с помощью SQL-команды `SELECT` извлечем из таблицы все строки, зафиксированные в период между началом 2017 года и текущей датой:

```
>>> conn = sqlite3.connect('reviews.sqlite')
>>> c = conn.cursor()
>>> c.execute("SELECT * FROM review_db WHERE date\"
...           " BETWEEN '2017-01-01 00:00:00' AND DATETIME('now')")
```

```
>>> results = c.fetchall()
>>> conn.close()
>>> print(results)
[('I love this movie', 1, '2020-04-15 17:53:46'), ('I disliked
this movie', 0, '2020-04-15 17:53:46')]
```

В качестве альтернативы мы могли бы также воспользоваться бесплатным приложением DB Browser for SQLite (Программа просмотра баз данных SQLite), доступным по ссылке <https://sqlitebrowser.org/dl/>; оно предлагает удобный графический пользовательский интерфейс для работы с базами данных SQLite (рис. 9.1).

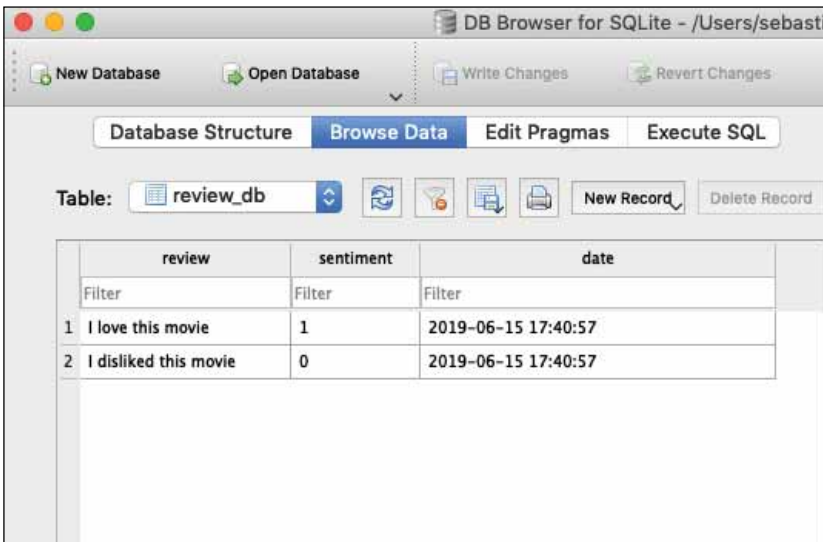


Рис. 9.1. Приложение DB Browser for SQLite

## Разработка веб-приложения с помощью Flask

Теперь, когда код для классификации рецензий на фильмы готов, давайте обсудим основы веб-фреймворка Flask, необходимого для разработки нашего веб-приложения. После выпуска Армином Ронахером первоначальной версии Flask в 2010 году этот фреймворк с годами обрел огромную популярность и применяется в таких известных приложениях, как LinkedIn и Pinterest. Поскольку фреймворк Flask написан на языке Python, он снабжает программистов на Python удобным интерфейсом для встраивания существующего кода Python, подобного классификатору рецензий на фильмы.