

# Содержание

<b>Предисловие</b> .....	10
<b>Глава 1. Установка и начало работы</b> .....	13
Подготовка к установке .....	13
Установка R .....	13
Установка Python .....	14
Конфиденциальность .....	14
Установка Java .....	15
Установка H2O при помощи R (CRAN) .....	15
Установка H2O при помощи Python (pip).....	16
Наша первая задача машинного обучения.....	17
Обучение и предсказания в Python.....	21
Обучение и предсказания в R .....	23
Производительность и предсказания .....	25
Если вам не повезло .....	26
Веб-интерфейс Flow .....	27
Данные .....	28
Модели.....	29
Предсказания .....	31
Дополнительные сведения об интерфейсе Flow .....	32
Резюме.....	32
<b>Глава 2. Импортирование и экспортирование данных</b> .....	33
Требования к памяти .....	33
Подготовка данных.....	34
Загрузка данных в H2O.....	35
Загрузка файлов в формате CSV.....	35
Загрузка файлов в других форматах .....	37
Загрузка данных из R .....	38
Загрузка данных из Python .....	39
Операции с данными .....	40
«Ленивость», присвоение имен и удаление .....	40
Итоговые статистики .....	42
Операции со столбцами .....	42
Агрегирование строк.....	43
Индексация .....	44
Разделение данных в кластере H2O .....	46
Строки и столбцы .....	49
Выгрузка данных из H2O .....	52
Экспорт таблиц .....	52
Формат ROJO .....	53
Файлы моделей.....	54
Сохранение всех моделей .....	54
Резюме.....	55

<b>Глава 3. Наборы данных</b> .....	56
Набор данных об энергетической эффективности .....	56
Настройка и загрузка .....	57
Переменные .....	58
Разделение данных.....	59
Изучение данных.....	60
О наборе данных.....	64
Набор данных: рукописные цифры.....	64
Настройка и загрузка .....	65
Изучение данных.....	66
Как можно «помочь» модели .....	68
О наборе данных.....	70
Набор данных: результаты футбольных матчей .....	70
Корреляции .....	73
Пропущенные данные.....	76
Как обучать и тестировать? .....	77
Настройка и загрузка .....	77
Третий источник данных .....	78
Снова про пропущенные данные .....	80
Настройка и загрузка (снова).....	80
О наборе данных.....	83
Резюме.....	83
<b>Глава 4. Общие параметры моделей</b> .....	84
Поддерживаемые метрики .....	84
Метрики для регрессии .....	85
Метрики для классификации.....	85
Бинарная классификация .....	86
Основы .....	88
Объем выполняемой работы .....	89
Оценка и проверка .....	90
Ранняя остановка.....	90
Контрольные точки .....	92
Перекрестная проверка.....	94
Взвешивание наблюдений.....	95
Выборки и обобщающая способность.....	98
Регрессия.....	99
Контроль вывода результатов.....	100
Резюме.....	100
<b>Глава 5. «Случайный лес»</b> .....	101
Решающие деревья .....	101
«Случайный лес».....	103
Параметры .....	103
Энергоэффективность зданий: «случайный лес» с настройками по умолчанию .....	105

Поиск по сетке .....	107
Полный перебор .....	108
Случайный поиск.....	110
Общая стратегия.....	112
Энергоэффективность зданий: настроенный «случайный лес» .....	113
MNIST: «случайный лес» с настройками по умолчанию.....	114
MNIST: настроенный «случайный лес» .....	116
Дополненные данные.....	119
Футбол: «случайный лес» с настройками по умолчанию .....	120
Футбол: настроенный «случайный лес» .....	122
Резюме.....	124
<b>Глава 6. Градиентный бустинг.....</b>	<b>125</b>
Бустинг .....	125
Хорошее, плохое... и непонятное .....	126
Параметры .....	127
Энергоэффективность зданий: градиентный бустинг с настройками по умолчанию .....	128
Энергоэффективность зданий: настроенный градиентный бустинг .....	130
MNIST: градиентный бустинг с настройками по умолчанию .....	133
MNIST: настроенный градиентный бустинг .....	134
Футбол: градиентный бустинг с настройками по умолчанию .....	137
Футбол: настроенный градиентный бустинг.....	138
Резюме.....	140
<b>Глава 7. Линейные модели .....</b>	<b>141</b>
Параметры GLM.....	141
Данные об энергоэффективности: GLM с настройками по умолчанию.....	145
Данные об энергоэффективности: настроенная GLM .....	147
MNIST: GLM с настройками по умолчанию .....	151
MNIST: настроенная GLM.....	153
Футбол: GLM с настройками по умолчанию .....	155
Футбол: настроенная GLM .....	156
Резюме.....	157
<b>Глава 8. Глубокое обучение (нейронные сети) .....</b>	<b>158</b>
Что такое нейронные сети?.....	159
Количественные и категориальные переменные .....	160
Слои нейронной сети .....	161
Функции активации .....	163
Параметры .....	164
Регуляризация .....	164
Оценка качества .....	165
Энергоэффективность зданий: модель глубокого обучения с настройками по умолчанию .....	168
Энергоэффективность зданий: настроенная модель глубокого обучения.....	168

MNIST: модель глубокого обучения с настройками по умолчанию.....	174
MNIST: настроенная модель глубокого обучения .....	175
Футбол: модель глубокого обучения с настройками по умолчанию .....	179
Футбол: настроенная модель глубокого обучения .....	180
Резюме .....	185
Приложение: дополнительные параметры .....	185
<b>Глава 9. Обучение на неразмеченных данных.....</b>	<b>187</b>
Кластеризация методом k-средних.....	188
Автокодировщики .....	191
Вложенные автокодировщики.....	193
Метод главных компонент.....	194
GLRM.....	196
Пропущенные данные.....	196
GLRM.....	200
Избавляемся от R.....	200
Резюме.....	203
<b>Глава 10. Все остальное.....</b>	<b>204</b>
Документация.....	204
Установка актуальной версии .....	204
Сборка из исходных кодов .....	204
Запуск из командной строки .....	205
Кластеры.....	205
EC2 .....	206
Другие облачные провайдеры .....	206
Hadoop .....	207
Spark / Sparkling Water .....	207
Наивный байесовский классификатор.....	207
Ансамбли.....	208
Стекинг: h2o.ensemble.....	208
Ансамбли для классификации.....	210
Резюме.....	210
<b>Глава 11. Эпилог .....</b>	<b>211</b>
Результаты для данных об энергоэффективности .....	211
Результаты для набора данных MNIST.....	213
Результаты для данных о футбольных матчах .....	214
Как далеко вы готовы зайти.....	216
Чем больше, тем лучше .....	217
Еще больше данных.....	218
Отбор сложных примеров.....	219
Автокодировщик .....	219
Сверточные сети .....	220
Ансамбли.....	221
Результаты.....	222
Резюме.....	223

<b>Приложение 1. Deep Water</b> .....	224
Установка.....	224
Сборка из исходных кодов .....	224
Amazon Machine Image .....	224
Образ Docker .....	224
Примеры данных .....	224
Обзор библиотеки Deep Water .....	224
Глубокое обучение в библиотеке H2O .....	225
Современные тенденции в глубоком обучении.....	225
Почему нужно использовать Deep Water .....	225
Начало работы: набор данных MNIST .....	226
Бекенды .....	227
CPU и GPU.....	227
Классификация изображений.....	229
Данные .....	229
Параметры изображений.....	229
Предварительно созданные архитектуры .....	229
Архитектуры, создаваемые пользователем.....	230
Предварительно обученные нейросети .....	230
Веб-интерфейс Flow .....	230
Поиск по сетке .....	233
Полный перебор .....	233
Случайный поиск.....	234
Контрольные точки .....	235
Ансамбли.....	237
Признаки скрытых слоев и меры сходства.....	238
Поддержка нескольких GPU.....	239
Развертывание моделей.....	240
МОJO.....	240
Prediction Service Builder .....	240
 <b>Приложение 2. Ансамбли (стекинг моделей).....</b>	 241
Вступление .....	241
Стекинг / Super Learner .....	241
Алгоритм.....	242
Вложенные ансамбли в библиотеке H2O.....	242
Пример .....	243
На языке R.....	243
На языке Python.....	245
Вопросы и ответы .....	247
Дополнительная информация .....	248
Список литературы.....	248
 <b>Краткий предметный указатель.....</b>	 249

# Предисловие

Похоже, что машинное обучение наконец-то достигло стадии зрелости. У него было длинное детство, охватывающее период с 1950-х гг., когда были созданы первые программы (для игры в шашки), обучающиеся на основе опыта, а также первые нейронные сети. Исследователи в области искусственного интеллекта так часто обещали скорый прорыв в этой сфере, что мы давно перестали их слушать. Но, возможно, они были на верном пути все это время, и нужно было лишь увеличить вычислительные мощности еще на порядок, или же немного доработать алгоритмы, чтобы перейти к их полезному и продуктивному использованию.

В начале 90-х нейронные сети называли новым прорывом в области искусственного интеллекта. Я пытался применить их для игры в го, но они работали очень плохо даже по сравнению с результатами (также весьма посредственными), которые были получены инженерными методами на основе знания предметной области с использованием усеченных деревьев поиска. Также они плохо масштабировались. И когда 20 лет спустя я услышал о потрясающих успехах глубокого обучения в этой игре, я был смущен отличиями от нейросетей, от использования которых я тогда отказался. «Недостаточно много» – вот ответ; иногда нужно предоставить алгоритму больше вычислительных ресурсов (в данном случае на 5–6 порядков больше), чтобы он раскрыл свой потенциал.

H2O – это программное обеспечение для машинного обучения и анализа данных. Я начал использовать данную библиотеку из-за интереса к возможностям глубокого обучения (хотя в ней реализованы и другие алгоритмы) и сразу же был впечатлен. Ее особенности:

- является открытым (open source) продуктом, распространяется на условиях лицензии Apache;
- проста в использовании;
- масштабируется для работы с большими данными;
- хорошо документирована и имеет коммерческую поддержку;
- актуальной является уже третья версия, что говорит о зрелой архитектуре;
- поддержка различных операционных систем и языков программирования.

При разработке H2O целью было обеспечение эффективной работы при масштабировании на больших объемах данных.

В этой книге рассматривается практический аспект использования библиотеки H2O с минимумом математики и теоретических основ применяемых методов.

Разумеется, библиотека H2O не лишена недостатков, и вот некоторые из них. В ней нет поддержки вычислений на GPU, что, в частности, могло бы повысить эффективность работы алгоритмов глубокого обучения<sup>1</sup>. Кластеры предназначены для работы с большими объемами данных, но для обработки относительно небольших выборок со сложной структурой вам может понадобиться один многоядерный сервер. Для кластеров не реализована поддержка высокой доступности. Библиотека H2O написана на языке Java и хорошо оптимизирована, но теоретически реализации на C++ могут работать быстрее. Не реализован алгоритм SVM (ма-

---

<sup>1</sup> В разработке находится проект DeepWater, который добавляет поддержку других библиотек для глубокого обучения, использующих GPU.

шины опорных векторов). Наконец, необходимость обеспечивать совместимость с несколькими платформами замедляет разработку.

## Кто и зачем использует H2O?

Многие известные компании ([www.h2o.ai/customers/](http://www.h2o.ai/customers/)) используют H2O для обработки своих данных, и сейчас их количество превышает 5000. В самой компании H2O.ai работает свыше 80 сотрудников, большая часть – разработчики.

Эта статистика может впечатлить вашего руководителя, но не вас самих как программистов. Для разработчиков на языках R или Python, которые уже используют необходимые им библиотеки машинного обучения, H2O предоставляет простой и эффективный способ работы с объемами данных, слишком большими для обработки на одном сервере. Алгоритмы, реализованные в H2O, работают значительно быстрее, по сравнению со SparkML (хотя в этой библиотеке реализовано больше методов, чем в H2O). Кроме того, продуманный выбор значений по умолчанию для параметров моделей позволяет писать компактный и понятный код. Одной из целей этой книги было научить вас настраивать модели, но, как мы увидим, иногда не получается улучшить результат, который дают настройки по умолчанию.

## Для кого эта книга

Чтобы ограничить объем книги парой сотен страниц, я позволил себе некоторые вольности. Подразумевается, что вы знаете R или Python. Продвинутое знание языков не используются, поэтому знаний других языков должно хватить для понимания примеров, но сами примеры приводятся только на R или на Python. Пользователи Python выиграют от знакомства с библиотекой pandas, и не в последнюю очередь благодаря тому, что она облегчает работу с данными.

Также я предполагаю наличие у вас определенной ментальной гибкости: чтобы не повторять каждый пример дважды, я предположил, что пользователи R смогут понять примеры на Python, и наоборот. Полезно ознакомиться с презентацией: <http://bit.ly/2gl4GRx> (в том числе пользователям R).

У вас должен быть определенный опыт выполнения операций с данными, хотя бы при помощи электронных таблиц или SQL. Также вы должны иметь общее представление о машинном обучении и искусственном интеллекте, об их применении в жизни. Возможно, вы читаете данную книгу, потому что хотите участвовать в этом процессе и желаете, чтобы происходящие изменения были этичными и благоприятными для всех. Если это так, я приветствую такой подход.

Также я рассчитываю, что вы немного знаете статистику. Ничего страшного знать не потребуется – название книги содержит слово «практическое» неспроста, и изложение теоретических основ алгоритмов машинного обучения сведено к минимуму, достаточному для настройки моделей, но недостаточному для самостоятельной реализации алгоритмов с нуля. Используйте Википедию или поисковые системы, чтобы узнать больше. Но вы точно должны знать, чем медиана отличается от среднего и от моды, а также что такое стандартное отклонение и нормальное распределение.

Более того, я надеюсь, что вы понимаете: статистика может вводить в заблуждение (хорошая иллюстрация: <https://xkcd.com/882/>), а модели машинного обучения могут переобучаться (<https://en.wikipedia.org/wiki/Overfitting>).

Также нужно уточнить, что подразумевается под «моим ПК», который часто упоминается в контексте времени работы алгоритмов. Это ноутбук средней мощности с 8 Гб ОЗУ и 4 ядрами (8 потоков). Этого достаточно для запуска примеров из книги; на самом деле должно хватить и 4 Гб ОЗУ. Тем не менее иногда я «жульничал» и запускал обработку на серверах облачных провайдеров. Это сделано исключительно из рациональных соображений: не хотелось ждать по 24 часа до получения результатов, прежде чем написать о них.

## ПРИНЯТЫЕ ОБОЗНАЧЕНИЯ

В книге использованы следующие типографские обозначения:

- *курсивом* выделены имена и расширения файлов, а также параметры функций при их перечислении в списках;
- моноширинный шрифт использован для программного кода и его элементов (имен переменных и функций и др.), включенных в текст.



Предупреждения появляются в этом поле.



Советы появляются тут.



Примечания появляются в этом поле.

## БЛАГОДАРНОСТИ

Прежде всего выражаю огромную благодарность техническим рецензентам: избитая фраза о том, что эта книга стала лучше благодаря вам, является правдой. Другая избитая фраза – все ошибки в книге остаются на моей совести. Итак, хочу сказать спасибо Кэтрин Джармул (Katharine Jarmul), Юлин Джуан (Yulin Zhuang), Хьюго Матиен (Hugo Mathien), Эрин ЛеДелл (Erin LeDell) и Тому Кралевичу (Tom Kraljevic); извините, если не принял каких-то из ваших предложений. Помимо Эрин и Тома, множество других людей из H2O.ai помогло мне своими ответами на вопросы, поэтому я благодарю Арно Кандела (Arno Candell), Томаса Никодима (Tomas Nykodym), Михала Курка (Michal Kurka), Навдипа Джилл (Navdeep Gill), СриСатиша Амбати (SriSatish Ambati), Лорен ДиПерна (Lauren DiPerna) и всех, кого я забыл упомянуть (извините!).

Спасибо Николь Таше (Nicole Tache) за то, что была редактором в течение первой половины процесса создания книги, и Дабби Хардин (Debbie Hardin) за то, что ее заменила (Николь решила, что единственный способ покинуть проект – это завести ребенка). Спасибо им обоим за их спокойствие, когда я так увлекался созданием моделей, что забывал про необходимость соблюдать сроки.

Спасибо моей семье за то, что отнеслись с пониманием к тому, сколько времени я вложил в эту книгу.

Наконец, спасибо всем остальным: людям, которые отвечают на вопросы на StackOverflow, размещают сообщения в блогах и создают видеокурсы, пишут книги, пополняют Википедию. Они помогли мне заполнить пробелы в моих знаниях. Не стесняйтесь сообщать мне об ошибках, найденных в книге, и о том, как ее можно улучшить.



# Глава 1

## Установка и начало работы

Вас очень обрадует простота установки библиотеки H2O. Сначала я покажу, как ее установить при помощи R, используя CRAN, а затем – как это сделать при помощи Python, используя pip<sup>1</sup>.

После этого мы погрузимся в наш первый проект по машинному обучению, который включает загрузку данных, создание модели и предсказаний на ее основе, а также оценку наших успехов. К этому моменту вы сможете похвастаться, что вы в некотором роде являетесь экспертом по глубокому обучению.

После обзора того, как влияние случайности может ввести нас в заблуждение, в этой главе более детально будет рассмотрен веб-интерфейс Flow, который поставляется вместе с H2O.

### ПОДГОТОВКА К УСТАНОВКЕ

В этой книге будут примеры и на языке R, и на языке Python, поэтому у вас должен быть установлен хотя бы один из них<sup>2</sup>. Вам также будет нужна платформа Java. Если есть выбор, я рекомендую всегда использовать 64-битные версии, в том числе 64-версию ОС (на странице загрузки 64-битная версия часто обозначена как «x64», а 32-битная – как «x86»).

Вы можете спросить, имеет ли значение выбор R или Python. Нет, и вскоре будет объяснено, почему. Также не существует преимуществ по производительности в случае использования скриптов вместо более дружественных к пользователю графических сред, таких как Jupiter или RStudio.

### Установка R

В ОС Linux менеджер пакетов вашего дистрибутива делает установку очень простой: `sudo apt-get install r-base` для Debian/Ubuntu/Mint/... или `sudo yum install R` для Red-Hat/Fedora/Centos/...

Пользователи Mac должны ознакомиться с <https://cran.r-project.org/bin/macosx/> и следовать данной инструкции.

<sup>1</sup> В главе 10 показаны некоторые альтернативные способы установки H2O (см. «Установка актуальной версии»), включая сборку из исходников. Вам это может понадобиться, если вы столкнулись с багом, который был исправлен только в последней версии для разработчиков, или если вы хотите начать самостоятельно вносить изменения в H2O.

<sup>2</sup> Здесь и далее под установкой языка подразумевается установка соответствующего интерпретатора. – *Прим. перев.*

При использовании ОС Windows скачайте и запустите установочный файл с <http://cran.rstudio.com/bin/windows/>, после чего следуйте инструкциям по установке. На странице **Выбор компонентов (Select Components)** предлагается установка 32- и 64-битных версий; я выбираю установку только 64-битной, но можно установить и обе.

Необязательным следующим шагом после установки R является установка RStudio; вы *можете* делать все необходимое для запуска H2O из командной строки, но RStudio значительно упрощает работу (особенно в Windows, где командная строка осталась на уровне 1995 года). Скачать и установить RStudio можно по адресу <https://www.rstudio.com/products/rstudio/download/>.


## Установка Python

H2O одинаково хорошо работает и с Python 2.7, и с Python 3.5, примеры из книги также должны работать. Если вы используете более раннюю версию Python, возможно, потребуется выполнить ее обновление. Также будет нужен pip – менеджер пакетов для Python. Для Red-Hat/Fedora/Centos/... точный вид команды может быть различным, поэтому следуйте инструкциям по адресу [https://packaging.python.org/en/latest/install\\_requirements\\_linux/](https://packaging.python.org/en/latest/install_requirements_linux/).

В ОС Linux нужно выполнить команду `sudo apt-get install python-pip` для Debian/Ubuntu/Mint/... или `sudo apt-get install python3-pip` в случае использования Python3.

Для установки на Mac следуйте руководству «Использование Python на Macintosh» (<http://bit.ly/2gn4HF5>).

При использовании ОС Windows следуйте аналогичному руководству «Использование Python в Windows» (<http://bit.ly/1RCJ7VR>). Не забудьте выбрать 64-битную версию (если только вы не пользуетесь до сих пор 32-битной версией Windows, разумеется).

 Вы также можете захотеть испытать дистрибутив Anaconda (<https://www.continuum.io/downloads>), который содержит практически все пакеты, необходимые специалисту по анализу данных. Дополнительным преимуществом является возможность установки обычным пользователем, что будет полезным при отсутствии у вас root-привилегий. Доступны версии для Linux, Mac и Windows.

## Конфиденциальность

H2O содержит код<sup>1</sup> для вызова сервиса Google Analytics при каждом запуске. Происходит анонимный сбор данных, поскольку отслеживается всего лишь используемая версия, но если это вас беспокоит или нарушает корпоративную политику безопасности, то создание пустого файла `.h2o_no_collect` в вашей домашней папке (`C:\Users\YourName\` в Windows) остановит данный процесс. Вы будете знать, что это сработало, увидев сообщение «Opted out of sending usage metrics.» в логах. Другой способ отключения сбора информации представлен в разделе «Запуск из командной строки» главы 10.

<sup>1</sup> <http://bit.ly/2f96Hyu> по состоянию на июнь 2016-го.

## Установка Java

У вас должна быть установлена платформа Java, которую вы можете получить на странице загрузки <http://bit.ly/16mhlmY>. Выберите вариант JDK<sup>1</sup>. Если вы думаете, что у вас уже есть Java JDK, но не уверены в этом, то можете двигаться дальше и установить H2O, а затем при необходимости вернуться к этому пункту и (пере)установить Java.

## УСТАНОВКА H2O ПРИ ПОМОЩИ R (CRAN)

Если вы не используете R, можете перейти к разделу «Установка H2O при помощи Python (pip)».

Запустите R и введите команду `install.packages("h2o")`. Да, когда я говорил о простоте установки, я не шутил. Эта команда также установит все зависимости.

Если вы впервые используете CRAN<sup>2</sup>, нужно будет выбрать используемое зеркало для скачивания (лучше всего в месте, ближайшем к вам, но можно выбрать и такое, куда бы вы хотели поехать и сделать селфи).

Если хотите установить H2O для всех пользователей на данном компьютере, запустите R с правами суперпользователя командой `sudo R`, затем введите команду `install.packages("h2o")`.

Давайте проверим, все ли работает, набрав команду `library(h2o)`. Если нет никаких тревожных сообщений, переходим к следующему шагу: `h2o.init()`. При должном везении вы увидите большое количество информации о запуске H2O, а также о вашем кластере, подобно представленной на рис. 1.1. В противном случае сообщение об ошибке укажет на отсутствующие зависимости или другие проблемы.

Версия H2O на CRAN может отставать на месяц или два от самой свежей и самой лучшей. Но не беспокойтесь об этом до тех пор, пока не столкнетесь с багом, исправленным в этой новейшей версии.

`h2o.init()` по умолчанию будет использовать только два ядра на вашем ПК и, возможно, четверть оперативной памяти<sup>3</sup>. Используйте команду `h2o.shutdown()`, которая делает именно то, о чем вы догадались. Затем запустите H2O снова, но с использованием всех доступных ядер: `h2o.init(nthreads = -1)`. Чтобы использовать все ядра и, скажем, 4 Гб ОЗУ: `h2o.init(nthreads = -1, max_mem_size = "4g")`.

<sup>1</sup> Варианты «Server JRE» или «JRE» *могут* работать с H2O, но я рекомендую всегда устанавливать JDK.

<sup>2</sup> CRAN является репозиторием пакетов для R. Дополнительную информацию можно найти на <https://cran.r-project.org/>.

<sup>3</sup> См. <http://bit.ly/2gn5h6e>, чтобы узнать, как выполнить запрос и определить значения по умолчанию для вашей системы.

```

> h2o.init()

H2O is not running yet, starting it now...

Note: In case of errors look at the following log files:
      /tmp/Rtmp6btQEF/h2o_darren_started_from_r.out
      /tmp/Rtmp6btQEF/h2o_darren_started_from_r.err

java version "1.7.0_101"
OpenJDK Runtime Environment (IcedTea 2.6.6) (7u101-2.6.6-0ubuntu0.14.04.1)
OpenJDK 64-Bit Server VM (build 24.95-b01, mixed mode)

Starting H2O JVM and connecting: ..... Connection successful!

R is connected to the H2O cluster:
  H2O cluster uptime:      3 seconds 535 milliseconds
  H2O cluster version:    3.8.2.2
  H2O cluster name:       H2O_started_from_R_darren_rge683
  H2O cluster total nodes: 1
  H2O cluster total memory: 1.71 GB
  H2O cluster total cores: 8
  H2O cluster allowed cores: 2
  H2O cluster healthy:    TRUE
  H2O Connection ip:      localhost
  H2O Connection port:    54321
  H2O Connection proxy:   NA
  R Version:               R version 3.2.5 (2016-04-14)

Note: As started, H2O is limited to the CRAN default of 2 CPUs.
      Shut down and restart H2O as shown below to use all your CPUs.
      > h2o.shutdown()
      > h2o.init(nthreads = -1)

> |

```

Рис. 1.1 ❖ Запуск `h2o.init()` (в R)

## УСТАНОВКА H2O ПРИ ПОМОЩИ PYTHON (PIP)

Если вы не используете Python, можете перейти к разделу «Наша первая задача машинного обучения».

Введите в командной строке `pip install -U h2o`. Все готово! Это невероятно просто.

Опция `-U` говорит о том, что нужно также обновить все зависимости. В Linux вам, вероятно, нужно будет обладать правами суперпользователя, поэтому используйте команду `sudo pip install -U h2o`. Или же можно выполнить установку для отдельного пользователя: `pip install -U -user h2o`.

Для проверки запустите Python, напечатайте `import h2o`, а затем, при отсутствии предостережений, `h2o.init()`. Будет выведена некоторая информация с таблицей в конце, где среди прочего будут указаны количество узлов (nodes), общее количество памяти и количество доступных ядер, как показано на рис. 1.2<sup>1</sup>. При написании сообщения о баге не забудьте указать всю информацию из данной таблицы.

<sup>1</sup> Возможно, вы видите какие-то сообщения с предостережениями? Для данного скриншота они были отключены при помощи `import warnings; warnings.filterwarnings("ignore", category = DeprecationWarning)`, но обычно я просто игнорирую их.

```

In [3]: import h2o

In [4]: h2o.init()
Checking whether there is an H2O instance running at http://localhost:54321.... not found.
Attempting to start a local H2O server...
  Java Version: java version "1.7.0_111"; OpenJDK Runtime Environment (IcedTea 2.6.7) (7u111-
ild 24.111-b01, mixed mode)
  Starting server from /usr/local/h2o_jar/h2o.jar
  Ice root: /tmp/tmpsNCJ_v
  JVM stdout: /tmp/tmpsNCJ_v/h2o_darren_started_from_python.out
  JVM stderr: /tmp/tmpsNCJ_v/h2o_darren_started_from_python.err
  Server is running at http://127.0.0.1:54321
Connecting to H2O server at http://127.0.0.1:54321... successful.
-----
H2O cluster uptime:          02 secs
H2O cluster version:        3.10.0.7
H2O cluster version age:    5 days
H2O cluster name:           H2O_from_python_darren_a9x6hb
H2O cluster total nodes:    1
H2O cluster free memory:    1.710 Gb
H2O cluster total cores:    8
H2O cluster allowed cores:  8
H2O cluster status:         accepting new members, healthy
H2O connection url:         http://127.0.0.1:54321
H2O connection proxy:
Python version:             2.7.6 final
-----

```


Рис. 1.2 ❖ Запуск `h2o.init()` (в Python)

По умолчанию ваш экземпляр (instance) H2O может использовать все доступные ядра и (как правило) 25% оперативной памяти. Обычно это подходит для работы, но что, если вы хотите использовать ровно 4 Гб памяти и лишь два из восьми ядер? Сначала остановите H2O при помощи команды `h2o.shutdown()`, затем введите `h2o.init(nthreads=2, max_mem_size=4)`. Ниже приведена выдержка из таблицы, подтверждающая, что команда работает:

```

...
H2O cluster total free memory:
H2O cluster total cores:
H2O cluster allowed cores:
...

```

 Виртуальные окружения `virtualenv` не работают с H2O<sup>1</sup>. Точнее, установка происходит, но запуск H2O невозможен. Если вы действительно хотите установить H2O таким способом, следуйте инструкциям по запуску H2O из командной строки в главе 10. В таком случае `h2o.init()` и все остальное будет работать.

## НАША ПЕРВАЯ ЗАДАЧА МАШИННОГО ОБУЧЕНИЯ

Теперь, когда установлено все необходимое, давайте займемся делом. API для Python и R настолько подобны, что в этом примере мы будем рассматривать их параллельно. Если вы используете Python, изучите программный код 1.1, а если R – программный код 1.2. Они повторяют операции `import/library` и `h2o.init`, которые мы выполнили ранее. Не волнуйтесь, это не навредит.

<sup>1</sup> По крайней мере, для версии 3.10.x.x и более ранних.

Я подробно все объясню на следующих нескольких страницах, но хочу подчеркнуть, что это полный скрипт: он загружает данные, подготавливает их, создает модель – многослойную нейронную сеть (т. е. это глубокое обучение), которая сопоставима с наилучшей на данный момент для этого набора данных, и делает предсказания с ее помощью.

### Набор данных Iris

Если вы раньше не слышали о наборе данных Iris, то это, должно быть, ваша первая книга по машинному обучению! Набор содержит 150 наблюдений для растений рода Ирис с результатами четырех измерений (длина и ширина чашелистиков и лепестков) и с указанием принадлежности к тому или иному виду. Представлены три вида, по 50 наблюдений для каждого из них.

Это очень популярный набор данных для машинного обучения, поскольку он достаточно маленький, чтобы обучение модели происходило быстро и его было удобно анализировать графически, и достаточно большой, чтобы быть интересным. Для него также не существует тривиального решения: ни одна из четырех переменных не позволяет идеально разделить классы.

#### Программный код 1.1 ❖ Глубокое обучение в Python с набором данных Iris

```
import h2o
h2o.init()

datasets = "https://raw.githubusercontent.com/DarrenCook/h2o/bk/datasets/"
data = h2o.import_file(datasets + "iris_wheader.csv") ❶
y = "class" ❷
x = data.names
x.remove(y)
train, test = data.split_frame([0.8]) ❸

m = h2o.estimators.deeplearning.H2ODeepLearningEstimator() ❹
m.train(x, y, train)
p = m.predict(test) ❺
```

#### Программный код 1.2 ❖ Глубокое обучение в R с набором данных Iris

```
library(h2o)
h2o.init(nthreads = -1)

datasets <- "https://raw.githubusercontent.com/DarrenCook/h2o/bk/datasets/"
data <- h2o.importFile(paste0(datasets, "iris_wheader.csv")) ❶
y <- "class" ❷
x <- setdiff(names(data), y)
parts <- h2o.splitFrame(data, 0.8) ❸
train <- parts[[1]]
test <- parts[[2]]

m <- h2o.deeplearning(x, y, train) ❹
p <- h2o.predict(m, test) ❺
```

Пункты ❶, ❷, ❸ отвечают за подготовку данных, ❹ – за обучение модели, ❺ – за использование модели.

❶ служит примером главной концепции, требующей понимания при использовании H2O: все данные находятся в кластере (сервере), а не на стороне клиента. *Даже если клиент и кластер являются одним и тем же компьютером.*

Следовательно, всякий раз, когда мы хотим обучить модель или сделать предсказание, мы должны разместить данные в кластере H2O; мы рассмотрим эту тему более детально в главе 2. Сейчас достаточно понимать, что эта строка создает таблицу с именем *iris\_wheader.hex* в кластере. Эта команда распознает, что первая строка csv-файла содержит заголовки, которые автоматически используются в качестве имен столбцов. Она также понимает (проанализировав данные), что столбец *class* является категориальной переменной, а это значит, что мы будем решать задачу многоклассовой классификации, а не регрессии (см. врезку «Жаргон и соглашения»).

② определяет пару вспомогательных переменных: *y* будет именем целевой переменной, т. е. переменной, которую мы хотим научиться предсказывать, а *x* будет содержать имена переменных, на которых будет происходить обучение; в данном случае это означает все остальные переменные. Иными словами, мы будем пытаться использовать результаты четырех измерений (*sepal\_len*, *sepal\_wid*, *petal\_len*, *petal\_wid*) для предсказания, к какому виду принадлежит растение.

### Жаргон и соглашения

Ваши данные представлены в виде *строк* (которые также называют *наблюдениями*) и *столбцов*. Они организованы в *таблицу* (*frame*, *data frame*)<sup>1</sup>. Если вы знакомы с электронными таблицами или таблицами SQL, то таблицы в H2O – это примерно то же самое. Аналогом в R является объект класса *data.frame*, а в Python (библиотека *pandas*) – *DataFrame* (или словарь *dict* из списков *list* равной длины).

В H2O столбцы могут иметь следующие типы:

- *real* – число с плавающей точкой, то есть *numeric* в R, *float* в Python и *double* во многих других языках;
- *int* – целое число;
- *enum* – набор категорий или классов (*factor* в R или *categorical* в *pandas*);
- *time* – 64-битное целое число, миллисекунды от начала эры Unix (1 января 1970 г.). Может быть получено из разных форматов времени;
- *string* – текст. Практически все, что вы можете с ним сделать в H2O, – это конвертировать в тип *enum*; текстовые данные не могут непосредственно использоваться при построении моделей.

Решение об использовании типов *int* или *real* принимается системой H2O после анализа данных в столбце; вы можете лишь выбирать, использовать ли тип *enum* для числовых данных.

В случае *обучения с учителем* или *обучения на размеченных данных* (*supervised machine learning*) один из этих столбцов будет содержать то, что мы хотим научиться предсказывать. Такой столбец называют откликом, зависимой переменной, выходной переменной, правильным ответом и другими терминами. В этой книге я буду сохранять имя данного столбца в переменной *y*<sup>2</sup>. В случае *обучения без учителя* или *обучения на неразмеченных данных* (*unsupervised machine learning*) переменная *y* не будет задана.

Некоторые из остальных или все остальные столбцы в таблице являются данными, на которых происходит обучение. Они называются по-разному: независимые пере-

<sup>1</sup> Здесь и далее термины *frame* и *data frame* переведены как «таблица», поскольку в русскоязычной литературе нет устоявшегося понятия «кадр данных» и речь всегда идет именно о таблицах. – *Прим. перев.*

<sup>2</sup> В API для R и Python она также называется *y*; в REST API ей соответствует *response\_column*.

менные, признаки, входные данные, предикторы. Мы договоримся хранить список имен таких столбцов в переменной  $x^1$ .

Еще несколько соглашений: все наши данные будут содержаться в переменной `data`; выборка, используемая для тренировки/обучения (`train`), будет находиться в переменной `train`; выборка для проверки (`validation`) – в переменной `valid`; выборка для тестирования (`test`) – в переменной `test`<sup>2</sup>. И запомните, все эти переменные являются лишь указателями (`pointers`) для данных, которые на самом деле хранятся в кластере. В Python используется класс-обертка для указателя (`handle`), где также хранятся некоторые итоговые статистики; в R используется тот же принцип с реализацией в виде окружения (`environment`).

Я использовал короткие имена: это книга, и переносы в программном коде выглядят некрасиво; некоторые могут даже читать ее с телефона. Я рекомендую вам применять в своем коде осмысленные имена, то есть `premierLeagueScores2005_2015_train` вместо `train`. Когда ваш скрипт занимает тысячи строк и вы имеете дело с десятками наборов данных, это сэкономит ваше психическое здоровье.

❷ (разбивка на обучающую и тестовую выборки) является еще одним принципом, суть которого состоит в борьбе с переобучением (`overfitting`). В общих чертах: мы случайным образом отбираем 80% наших данных для обучения, а затем пытаемся использовать модель на оставшихся 20%, чтобы увидеть, насколько она хороша. В реально работающей системе эти 20% представляют садовников, которые приходят с новыми растениями и спрашивают, к какому виду они относятся.

Напомню, что в Python разбивка данных выглядит следующим образом. `split_frame()` является одной из функций (методов) класса `H2OFrame`. Указание `[0.8]` приводит к тому, что в первую часть попадают 80% данных, а остальные данные – во вторую:

```
train, test = data.split_frame([0.8])
```

В R функция `h2o.splitFrame()` принимает таблицу и возвращает список, элементы которого для удобства присвоены переменным `train` и `test`:

```
parts <- h2o.splitFrame(data, 0.8)
train <- parts[[1]]
test <- parts[[2]]
```

Разделение, которое происходит случайным образом для каждой строки, приводит к пропорции примерно 120/30 строк, но вы можете получить чуть больше строк в обучающей или тестовой выборке<sup>3</sup>.

Давайте подведем итоги. Как показано на рис. 1.3, клиент хранит лишь указатели для данных, находящихся в кластере H2O.

<sup>1</sup> Эта переменная во всех API называется  $x$ , но если вы когда-нибудь решите работать с REST API, то найдете там дополнение к  $x$ : список имен столбцов, которые *не нужно* использовать, с именем `ignored_columns`.

<sup>2</sup> Если вы незнакомы с этими концепциями, в главе 3 объясняются различия между проверочной и тестовой выборками, а также между проверкой и перекрестной проверкой (`cross-validation`).

<sup>3</sup> Поэкспериментируйте с аргументом `seed` для получения точной разбивки. Например, мне повезло с `h2o.splitFrame(data, 0.8, seed = 99)`. В Python: `data.split_frame([0.8], seed = 99)`.



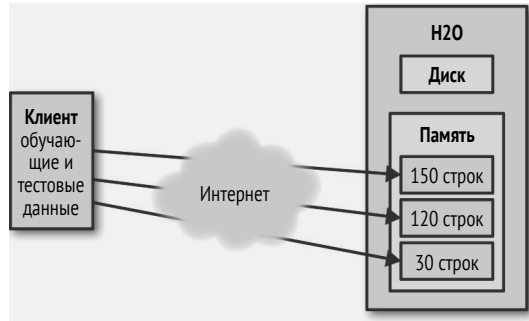


Рис. 1.3 ❖ Местоположение данных

Разумеется, наш «кластер» находится на локальном компьютере, то есть там же, где и клиент, в одной и той же оперативной памяти. Но вы должны думать о них как о находящихся на противоположных сторонах земного шара. Также представьте, что данные могут содержать миллиарды строк, слишком много, чтобы они могли поместиться в памяти клиента. Путем добавления компьютеров в кластер, если его общий объем памяти будет достаточно большим, можно обеспечить возможность загрузки и анализа таких данных клиентом, которым может выступать обычный ноутбук.

## Обучение и предсказания в Python

Наконец, мы добрались до пункта ④ – машинного обучения. В Python процесс состоит из двух стадий:

1. Создадим объект для вашего алгоритма машинного обучения, задав при необходимости параметры для него:

```
m = h2o.estimators.deeplearning.H2ODeepLearningEstimator()
```

2. Запустим обучение, указав, какие данные использовать:

```
m.train(x, y, train)
```

Если вы предпочитаете стиль библиотеки `scikit-learn`, можете вместо этого использовать следующие команды:

```
from h2o.estimators.deeplearning import H2ODeepLearningEstimator
m = H2ODeepLearningEstimator()
m.train(x, y, train)
```

Поскольку мы не задаем никаких параметров конструктора, модель строится со значениями по умолчанию. Это значит (помимо прочего), что обучается нейросеть с двумя скрытыми слоями по 200 нейронов каждый в течение 20 эпох. В главе 8 будут даны определения понятий «нейроны» и «эпохи», но сейчас не будем об этом. Важно отметить, что со значениями по умолчанию обучение проходит очень быстро, оно занимает всего несколько секунд для этого набора данных.

Так же, как и в случае с данными, `m` является экземпляром класса-обертки указателя для модели, хранящейся в кластере H2O. Если вы выведете `m` на экран, то увидите много информации о том, как проходило обучение. Также можно использовать методы класса, чтобы получить только ту информацию, которая вас инте-

ресует, – например, `m.mse()` сообщит мне, что MSE (mean squared error – средне-квадратичная ошибка) равна 0.01096. Здесь есть элемент случайности, так что ваше значение, вероятно, будет немного отличаться<sup>1</sup>.

`m.confusion_matrix(train)` возвращает *матрицу несоответствий* (confusion matrix), которая содержит информацию о правильно и неправильно определенных классах. Ниже представлены результаты для 120 наблюдений обучающей выборки:

Iris-setosa	Iris-versicolor	Iris-virginica	Error	Rate
42	0	0	0	0 / 42
0	37	1	0.0263158	1 / 38
0	1	39	0.025	1 / 40
42	38	40	0.0166667	2 / 120

В данном случае я вижу, что все 42 экземпляра вида *setosa* были определены безошибочно, но один экземпляр вида *versicolor* был определен как *virginica*, и один экземпляр *virginica* – как *versicolor*.

Последняя строка программного кода – **5** `p = m.predict(test)`. Она делает предсказания на основе модели и присваивает результат переменной `p`. Ниже приводится несколько примеров этих предсказаний. В первом столбце указан предсказанный класс, в остальных – вероятности принадлежности к каждому из классов. Как вы можете видеть, модель уверена в своих ответах более чем на 99.5%:

predict	Iris-setosa	Iris-versicolor	Iris-virginica
Iris-setosa	0.999016	0.000983921	1.90283E-019
Iris-setosa	0.998988	0.00101178	1.40209E-020
Iris-versicolor	5.22035E-005	0.997722	0.00222536
Iris-versicolor	0.000275126	0.995354	0.00437055

И здесь тоже таблица хранится в кластере H2O, поэтому вы видите лишь превью – первые 10 строк. Чтобы увидеть все 30 предсказаний, их нужно загрузить при помощи команды `p.as_data_frame()`. Если у вас не установлена библиотека `pandas`, вы увидите вложенный список:

```
[['predict', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica'],
 ['Iris-setosa', '0.9990160791818314', '9.83920818168421E-4',
 '1.9028267028039464E-19'], ['Iris-setosa', '0.9989882189908829', ...
 ..., ['Iris-virginica', '1.72617432126E-11', '1.0197263306598747E-4',
 '0.9998980273496721']]]
```

Вы можете анализировать данные в таком виде. Тем не менее API H2O для Python интегрирован с `pandas`. Если вы используете Python для работы с данными, то наверняка знакомы с этой библиотекой (а если нет, установите ее при помощи `pip install pandas`). При наличии установленной библиотеки `pandas` результат работы функции `p.as_data_frame()` будет таким:

<sup>1</sup> Я использовал `h2o.deeplearning(x, y, train, seed = 99, reproducible = TRUE)` для получения воспроизводимых значений.

```

      predict  Iris-setosa  Iris-versicolor  Iris-virginica
0  Iris-setosa  9.990161e-01      0.000984    1.902827e-19
1  Iris-setosa  9.989882e-01      0.001012    1.402089e-20
...

```

Что еще мы можем сделать? Этот вопрос детально рассматривается в главе 2. Мы можем получить долю правильных ответов при помощи `p["predict"] == test["class"]`. А команда `p["predict"].cbind(test["class"]).as_data_frame()` выведет результат в виде двух столбцов с предсказаниями и правильными метками классов:

```

      predict      class
0  Iris-setosa  Iris-setosa
1  Iris-setosa  Iris-setosa
...
11 Iris-versicolor Iris-versicolor
12 Iris-virginica Iris-versicolor
13 Iris-versicolor Iris-versicolor
14 Iris-virginica Iris-versicolor
15 Iris-versicolor Iris-versicolor
...

```

## Обучение и предсказания в R

В R этап машинного обучения ❹ состоит из единственного вызова функции, в котором одновременно указываются параметры и используемые данные: `m <- h2o.deeplearning(x, y, train)` (на самом деле я запускал команду `m <- h2o.deeplearning(x, y, train, seed = 99, reproducible = TRUE)` для получения воспроизводимых результатов, но вам не стоит этого делать, потому что таким образом будет использоваться всего одно ядро, и это займет больше времени).

И вновь `m` является указателем для модели, хранящейся в кластере H2O. `h2o.mse(m)` сообщит мне, что MSE равна 0.01097. `h2o.confusionMatrix(m)` вернет матрицу несоответствий (по умолчанию – для обучающей выборки):

```

Confusion Matrix: vertical: actual; across: predicted
      setosa  versicolor  virginica  Error  Rate
setosa      42           0           0  0.0000 = 0 / 42
versicolor  0           37           1  0.0263 = 1 / 38
virginica   0           1           39  0.0250 = 1 / 40
Totals      42           38           40  0.0167 = 2 /120

```

Видим, что был получен идеальный результат для `setosa`, но по одному экземпляру вида `versicolor` и `versicolor` были определены ошибочно. Число в правой нижней ячейке соответствует общей доле ошибок – 1.67% (на обучающих данных).

Последняя строка программного кода – ❺ `p = h2o.predict(m, test)`. Она делает предсказания на основе модели. Как обычно, `p` является ссылкой на таблицу в кластере H2O. Если я выведу на экран `p`, то получу только первые шесть предсказаний. Чтобы увидеть их все, мне нужно загрузить данные. Будьте внимательны, работая с кластерами или с большими данными (Big Data™): вы должны сначала решить, как много данных вам нужно иметь на локальном компьютере, сколько времени уйдет на загрузку и справится ли вообще ваш компьютер с ними.

Команда `as.data.frame(p)` выведет все 30 предсказаний (ниже показаны некоторые из них):

```
predict      Iris-setosa Iris-versicolor Iris-virginica
-----
Iris-setosa  0.999016   0.0009839   1.90283e-19
Iris-setosa  0.998988   0.0010118   1.40209e-20
Iris-setosa  0.999254   0.0007460   9.22466e-19
...
Iris-virginica 1.5678e-08  0.3198963   0.680104
Iris-versicolor 2.3895e-08  0.9863869   0.013613
...
Iris-virginica 3.9084e-14  2.192105e-06  0.999998
```

Столбец `predict` содержит предсказанные метки классов, остальные три столбца отображают меру уверенности в предсказаниях. Мы видим, что модель имеет разную степень уверенности в тех или иных предсказаниях.

Следующий вопрос, который вы можете задать: делает ли ошибки модель, созданная в H2O? Правильные названия видов указаны в `test$class`, а предсказания — в `p$predict`. Чтобы их сопоставить, можно применить два различных подхода:

```
as.data.frame( h2o.cbind(p$predict, test$class) )
```

и

```
cbind( as.data.frame(p$predict), as.data.frame(test$class) )
```

В первом случае `test$class` и `p$predict` объединяются и создают новую таблицу в кластере. Затем эта таблица из двух столбцов загружается клиентом. Во втором случае сначала загружаются `test$class` и `p$predict`, затем объединяются в таблицу на стороне клиента. Как правило, предпочтение следует отдавать первому способу.

В моем случае (у вас результаты могут слегка отличаться) получается следующее (звездочкой отмечены ошибочные предсказания):

```
predict      class
1      setosa  setosa
2      setosa  setosa
3      setosa  setosa
4      setosa  setosa
5      setosa  setosa
6      setosa  setosa
7      setosa  setosa
8      setosa  setosa
9  versicolor versicolor
10 versicolor versicolor
11 versicolor versicolor
12 versicolor versicolor
13  virginica versicolor *
14 versicolor versicolor
15  virginica versicolor *
16 versicolor versicolor
17 versicolor versicolor
18 versicolor versicolor
```

```

19 versicolor versicolor
20 virginica virginica
21 virginica virginica
22 virginica virginica
23 virginica virginica
24 virginica virginica
25 virginica virginica
26 virginica virginica
27 virginica virginica
28 virginica virginica
29 virginica virginica
30 virginica virginica

```

Другим способом анализа является расчет доли правильных ответов. В R мы можем сделать это при помощи `mean(p$predict == test$class)`. Результат равен 0.933, иными словами, модель верно определяет класс 93.3% неизвестных (тестовых) наблюдений и ошибается в 6.7% случаев. Как мы увидим в разделе «Если вам не повезло», почти наверняка вы получите значение 0.900 (3 ошибки), 0.967 (2 ошибки) или 1.000 (идеальный результат).

## Производительность и предсказания

Мы можем найти долю правильных ответов и другим способом: в R используем `h2o.performance(m, test)` вместо `predict()`; в Python используем `m.model_performance(test)`. Таким образом мы не получим отдельных предсказаний, но зато увидим большое количество сводных данных<sup>1</sup>:

```

H2OMultinomialMetrics: deeplearning

Test Set Metrics:
=====

MSE: (Extract with `h2o.mse`) 0.05968686
RMSE: (Extract with `h2o.rmse`) 0.2443089
Logloss: (Extract with `h2o.logloss`) 0.1843311
Mean Per-Class Error: 0.03921569
Confusion Matrix: Extract with `h2o.confusionMatrix(<model>, <data>)`
=====
Confusion Matrix: vertical: actual; across: predicted
      Iris-setosa Iris-versicolor Iris-virginica Error Rate
Iris-setosa      9              0              0 0.0000 = 0 / 9
Iris-versicolor  0              4              0 0.0000 = 0 / 4
Iris-virginica   0              2             15 0.1176 = 2 / 17
Totals           9              6             15 0.0667 = 2 / 30

Hit Ratio Table: Extract with `h2o.hit_ratio_table(<model>, <data>)`

```

<sup>1</sup> На момент перевода актуальной была версия H2O 3.10.0.8, в которой формат вывода результатов оценки производительности значительно изменен, по сравнению с версией, использованной при написании оригинала; в текст включен обновленный вариант. К сожалению, в первоисточнике не были заданы начальные значения для генератора случайных чисел при разбивке данных на обучающую и тестовую выборки, поэтому результаты не являются воспроизводимыми, и некоторые числа в этом фрагменте отличаются от полученных ранее. – *Прим. перев.*

```
=====
Top-3 Hit Ratios:
```

```
  k hit_ratio
1 1  0.933333
2 2  1.000000
3 3  1.000000
```

«Коэффициент попаданий» (hit ratio) дает нам то же самое число 0.933 (1 во второй строке означает, что среди двух классов, предсказанных с наибольшей вероятностью, обязательно будет истинный – будет получена доля верных ответов, равная 100%).

**!** Рассмотрев степень уверенности (вероятности) наших корректных предсказаний, мы заметим, что в большинстве случаев она выше 0.99, а наименьшие значения равны примерно 0.97. А как насчет ошибочных ответов модели? Для строки 13 тестовых данных это значение составило 0.45 против 0.55, для строки 15 – 0.07 против 0.93.

Это очень хорошо, ведь мы можем считать результаты с вероятностью меньше 0.97 сомнительными. В сфере медицины такие результаты могут означать необходимость выполнения альтернативного диагностического теста; в финансовой сфере это можно интерпретировать как показатель слишком рискованной операции.

Но! Вы могли что-то заподозрить сразу после слов «очень хорошо». Мы выбрали порог отсечения 0.97 на основании результатов на *тестовых* данных, уже зная правильные ответы. *Все* параметры, используемые для интерпретации результатов на тестовых данных, *должны* быть выбраны исключительно на основе обучающих данных. В разделе «Проверочные и тестовые данные» главы 2 мы также рассмотрим использование проверочных данных для этих целей.

## Если вам не повезло

Сейчас самое время обсудить влияние случайных эффектов на результат. Чтобы изучить это влияние, я попробовал переделать модель 100 раз (используя начальные значения для генератора случайных чисел от 1 до 100). 52 раза модель сделала две ошибки и 48 раз – одну. В зависимости от вашей точки зрения случайность оказала или небольшое влияние (93% правильных ответов против 97%), или двукратное (две ошибки против одной). Выше мы рассматривали один из «неудачных» случаев.

А что насчет случайного разделения данных на обучающую и тестовую выборки? Как это влияет на результат? Я выполнил 25 таких разделений (получив пропорции от 111/39 до 130/20) и построил 20 моделей для каждого из них. Создание 500 моделей заняло примерно 20 минут на моем компьютере, так что такие эксперименты, к сожалению, нецелесообразны для больших наборов данных, которые мы будем использовать далее в этой книге.

Кажется, что случайность в процессе разбивки данных оказывает большее влияние, чем случайность при построении модели<sup>1</sup>, потому что один вариант

<sup>1</sup> Это более характерно для маленьких наборов данных, подобных набору Iris. Для больших наборов данных такая ситуация менее вероятна, кроме случаев, когда, например, категориальная переменная имеет одну категорию со значительно меньшим числом наблюдений, чем в других категориях.

разбивки дает нам идеальный результат для всех 20 моделей (вариант со 129 наблюдениями в обучающем наборе и 21 в тестовом), в то время как другой вариант приводит к среднему результату на уровне 90% правильных ответов (вариант со 114 наблюдениями в обучающей выборке и 36 в тестовой). Вы думаете, что чем больше данных используется для обучения, тем лучше? Тем не менее разделение со 130 наблюдениями в обучающей выборке тоже дает 90% правильных ответов для большинства моделей.

Но и это не все! Важнейший урок, который я вынес из этого эксперимента: 85 моделей из 500 (17%) дают идеальный результат на тестовых данных. Обычно мы используем *одно* разделение и получаем *одну* модель. Это означает, что в 17% случаев мы ошибочно будем считать, что нашли идеальные параметры модели.

Пару лет назад была опубликована новость, что в психологии 64% экспериментов (опубликованных в уважаемых журналах) не воспроизводятся. Я думаю, что подобный вид невезения<sup>1</sup> имел место в некоторых из них.

## ВЕБ-ИНТЕРФЕЙС FLOW

Flow – это веб-интерфейс, который является частью системы H2O (отдельная установка не требуется). По сути, это еще одно клиентское приложение, написанное на CoffeeScript (JavaScript-подобном языке), которое создает такие же веб-запросы к серверной части (backend) H2O, что и R- или Python-клиенты. Оно обладает исчерпывающей функциональностью в том смысле, что вы можете выполнять следующие действия:

- просматривать на стороне клиента загруженные данные;
- загружать данные;
- просматривать на стороне клиента созданные модели (и те модели, которые создаются в данный момент!);
- создавать модели;
- просматривать на стороне клиента сделанные предсказания;
- делать предсказания.

Вы можете найти этот интерфейс по адресу <http://127.0.0.1:54321>. Разумеется, если вы запустили H2O на нестандартном порту, замените 54321 на правильный номер порта, а если H2O запущен на удаленном кластере – укажите имя одного сервера в кластере (*публичное* DNS-имя или IP-адрес, а не *приватное*, если у сервера есть оба). Когда вы впервые запустите Flow, то увидите меню, показанное на рис. 1.4.

<sup>1</sup> Подразумевается невезение с точки зрения более здорового и счастливого общества. Разумеется, это была большая удача для того, чьей целью была публикация в журнале!

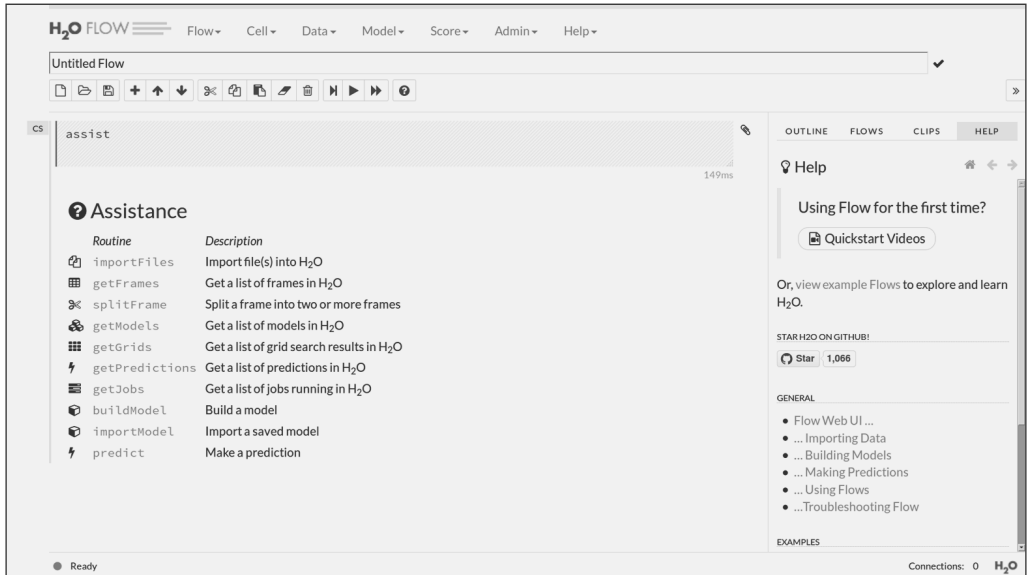


Рис. 1.4 ❖ Меню Flow

## Данные

Давайте импортируем тот же набор данных Iris, что и в примерах с R и Python. Нажмите на ссылку **importFiles**, или выберите **Data** » **Import Files** в верхнем меню. Вставьте ссылку на расположение файла в формате csv в строку поиска (**Search**), нажмите на появившуюся ссылку в графе **Search Results**, затем нажмите кнопку **Import** – см. рис. 1.5.

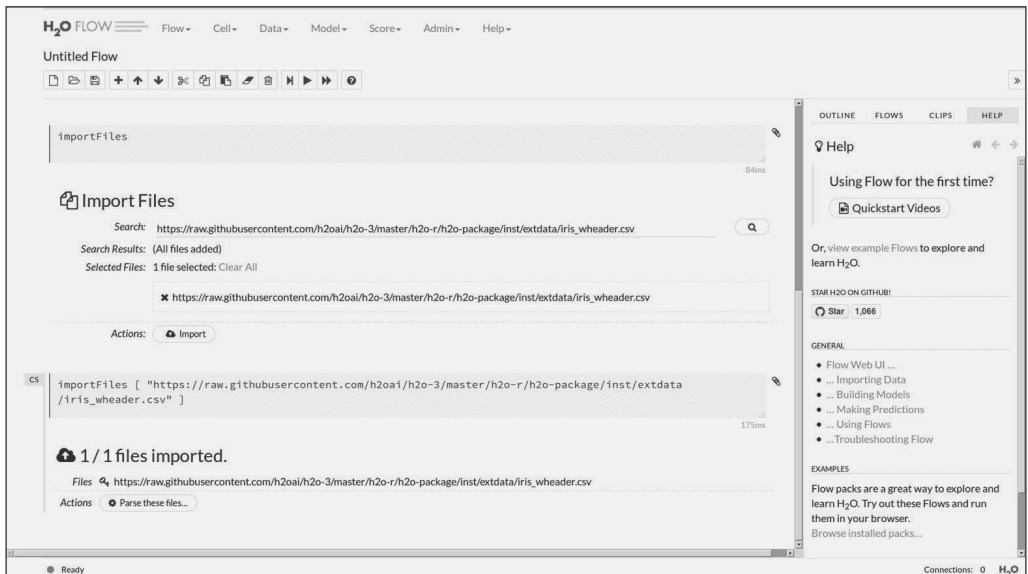


Рис. 1.5 ❖ Импортирование файлов



Теперь нажмите кнопку **Parse These Files**, после чего у вас будет возможность задать опции, показанные на рис. 1.6. В данном случае достаточно принять значения по умолчанию.

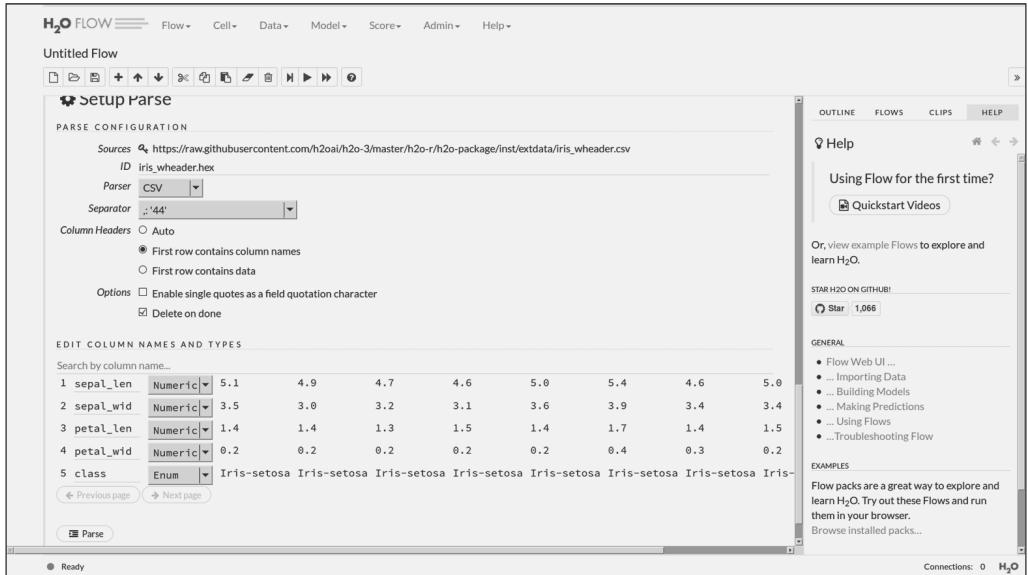


Рис. 1.6 ❖ Парсинг файла в интерфейсе Flow

Если вы выберете ссылку **getFrames** в самой верхней ячейке после выполнения предыдущих шагов или после загрузки данных в R или Python, то увидите запись, которая оканчивается на `iris_wheader.hex` и содержит 150 строк и 5 столбцов. Нажав на нее, вы увидите нечто похожее на рис. 1.7.

Вы можете видеть кнопку для разделения данных на обучающую и тестовую выборки, кнопку для создания модели, а также список столбцов. Важно отметить, что столбец `class` был автоматически распознан как данные типа `enum`, то есть его можно сразу использовать для решения задачи классификации. Если мы захотим заняться регрессией, нужно будет нажать на **Convert to numeric**.

Нажмите на **Split**, затем измените 0.75 на 0.8. В столбце **Key** назовите новые наборы данных `train` и `test`.

## Модели

Нажмите на **train**, затем на **Build Model** и выберите алгоритм **Deep Learning**. Вы увидите большое количество параметров, но нужно задать только некоторые из них: в выпадающем меню **response\_column** выберите `class` и нажмите кнопку **Build Model**, поскольку остальные параметры по умолчанию нас устраивают. Вы увидите что-то похожее на рис. 1.8.

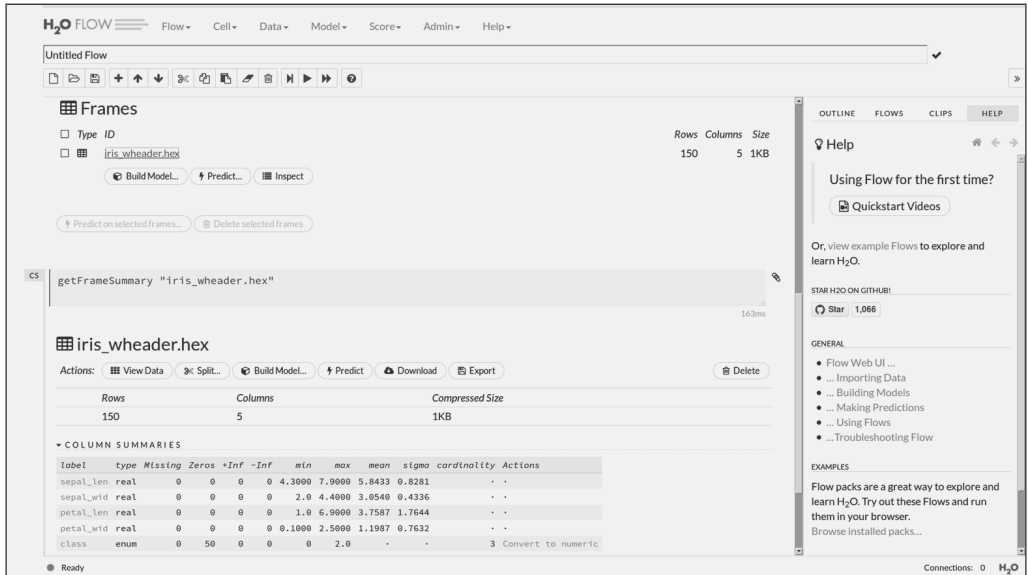


Рис. 1.7 ❖ Просмотр таблицы в интерфейсе Flow



Рис. 1.8 ❖ Модель глубокого обучения в интерфейсе Flow

Нажмите кнопку **View**. Другой способ: если вы предварительно построили модель в R, Python или Flow, выберите **Model** » **List All Models** в главном меню и нажмите на интересующую вас модель. Как видно на рис. 1.9, вы получите графический результат; другие доступные пункты – параметры модели и информация о том, как происходило обучение.

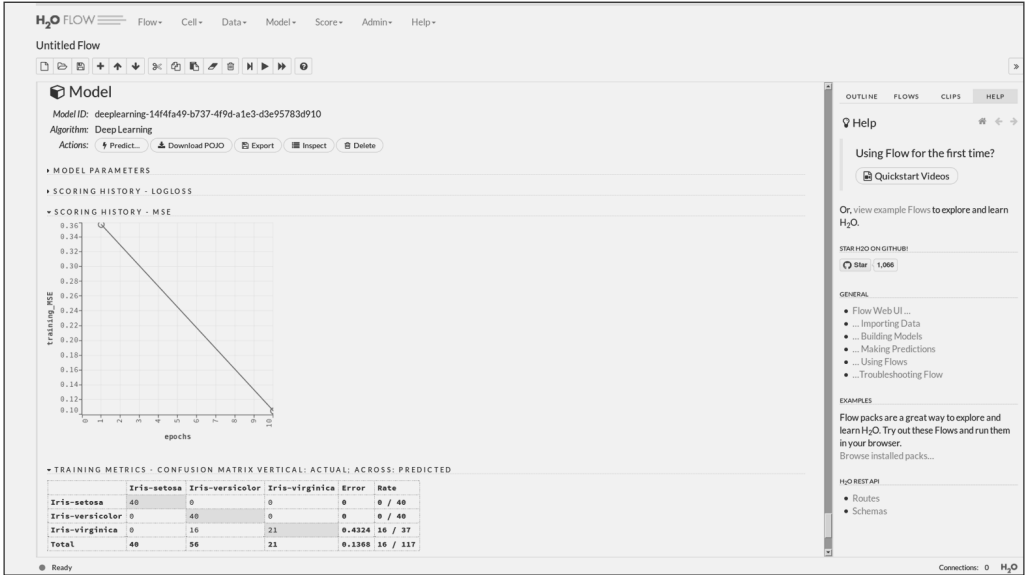


Рис. 1.9 ❖ Изучение модели в интерфейсе Flow

## Предсказания

Вы можете выполнять в интерфейсе Flow весь цикл загрузки-моделирования-предсказания. Нажав на кнопку **Predict**, вы получите возможность указать набор данных (**test** в нашем случае), для которого будет предсказана целевая переменная (также можно выбрать в главном меню **Score** » **Predict**). Результаты соответствуют представленным на рис. 1.10.

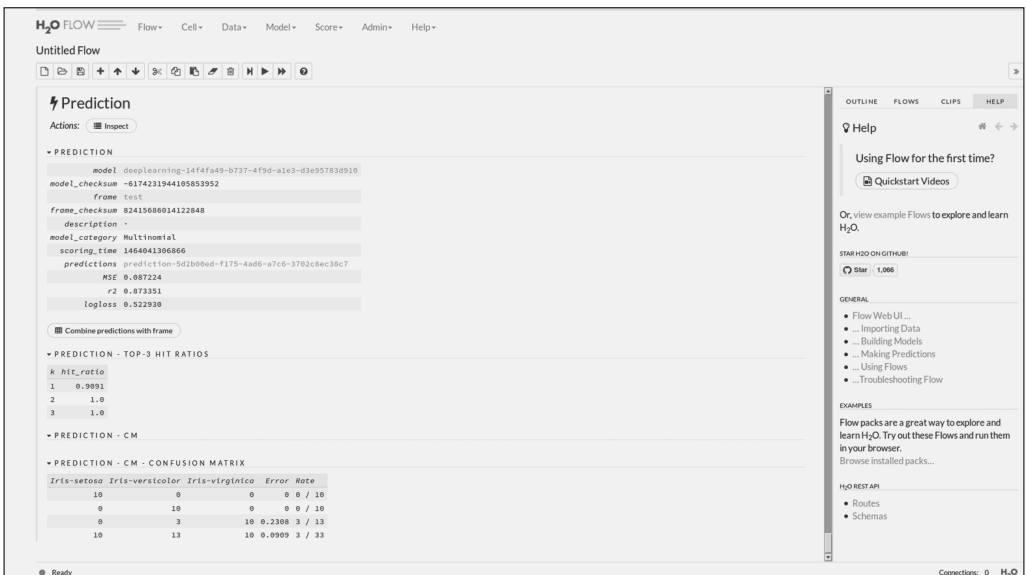


Рис. 1.10 ❖ Предсказания в интерфейсе Flow

## Дополнительные сведения об интерфейсе Flow

Команды в интерфейсе Flow, которые вы видите, могут быть сохранены как скрипт и вновь загружены позднее. Но есть вещи, которые вы можете делать в R или Python, но не в интерфейсе Flow, – прежде всего это объединение таблиц и манипуляции с данными (будут рассмотрены в главе 2).

Таким образом, для некоторых пользователей Flow может делать все, что им нужно, но большинство из нас выберет R или Python. Далее в этой книге я не буду рассматривать примеры с использованием Flow, хотя полученные вами знания могут быть использованы и для работы во Flow.

Тем не менее Flow может быть полезен для вас, даже если вы намереваетесь использовать только R или лишь Python. Данные, загруженные в R/Python, можно просматривать во Flow (и наоборот). Более того, вы можете запустить длительное обучение модели в R/Python и следить за процессом обучения во Flow. Увидев неожиданные показатели быстродействия, вы можете понять, что что-то упустили, и остановить задачу, избежав ненужных затрат процессорного времени. А пункт **Water Meter** в меню **Admin** позволит вам увидеть загрузку каждого процессора на вашем кластере.

## РЕЗЮМЕ

В первой главе мы изучили основы и рассмотрели:

- установку H2O для R и Python;
- импортирование данных, создание моделей и предсказаний;
- и все это при помощи R, Python или веб-интерфейса Flow.

Мы упустили из рассмотрения много возможностей и в оставшейся части книги будем углублять наши знания.