O'REILLY®

Машинное обучение с использованием библиотеки H2O

Даррен Кук

ДМК ПРЕСС
ИЗДАТЕЛЬСТВО

УДК 004.85H2O
ББК 32.813
К89

Кук Д.
К89 Машинное обучение с использованием библиотеки H2O / пер. с англ. А. Б. Огурцова. – М.: ДМК Пресс, 2018. – 250 с.: ил.

ISBN 978-5-97060-508-0

H2O – простая в использовании и открытая библиотека, которая поддерживает большое количество операционных систем и языков программирования, а также масштабируется для обработки больших данных. Эта книга научит вас использовать алгоритмы машинного обучения, реализованные в H2O, с упором на наиболее важные для продуктивной работы аспекты. Рассмотрены глубокое обучение, случайный лес, обучение на неразмеченных данных и ансамбли моделей.

В российское издание добавлены дополнительно два приложения, описывающих новейшие модули H2O – Deep Water и Stacked Ensemble. Их также можно найти в репозитории https://github.com/statist-bhfz/h2o_book_translate.

Издание предназначено для специалистов по анализу данных, желающих изучить и применять на практике относительно новый, но многообещающий инструмент – библиотеку H2O.

УДК 004.85H2O
ББК 32.813

ISBN 978-1-491-96460-6 (анг.)
ISBN 978-5-97060-508-0 (рус.)

Содержание

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[1].

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[2]. 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.

---

[1] 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.

[2] 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/.

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/2gn4HFs).

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[1] 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.

---

[1] 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[1]. 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[2], 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[3]. 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")`.

---

[1] Варианты «Server JRE» или «JRE» *могут* работать с H2O, но я рекомендую всегда устанавливать JDK.

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

[3] 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 неимоверно 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[1]. При написании сообщения о баге не забудьте указать всю информацию из данной таблицы.

[1] 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[1]. 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.

[1] 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»).

❷ 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)[1]. 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`[2]. 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-

---

[1] 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.*

[2] В API для R и Python она также называет `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`[2]. 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 сбережет ваше психическое здоровье.

❸ (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[3].

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

---

[1] 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`.

[2] 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).

[3] 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)`.

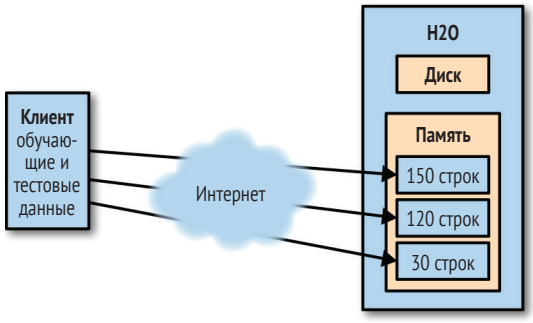

Рис. 1.3 ❖ Местоположение данных

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 ❹ – 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 – среднеквадратичная ошибка) равна 0.01096. Здесь есть элемент случайности, так что ваше значение, вероятно, будет немного отличаться[1].

`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 – ❺ `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:

---

[1] 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"]).mean()`. 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[1]:

```
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>)`
```

[1] 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.*