

高级大数据 解析

Lecture 6, play with Spark
(Python)
付彦伟



Link Spark with IPython Notebook

Assume you had installed ipython or Jupyter

```
echo "export PATH=$PATH:/Users/yanwei/software/spark-2.1.0-bin-hadoop2.7/bin" >> .profile  
echo "export PYSARK_DRIVER_PYTHON=ipython" >> .profile  
echo "export PYSARK_DRIVER_PYTHON_OPTS='notebook' pyspark" >> .profile
```

Now you can source it to make changes available in this terminal

```
source .profile
```

Run IPython Notebook

```
cd Documents/my_spark_folder  
pyspark
```

You should see something like this

```
In [1]: sc  
Out[1]: <pyspark.context.SparkContext at 0x1049bdf90>
```

Or adding spark bin into \$PATH

```
echo "export PATH=$PATH:/Users/yanwei/software/spark-2.1.0-bin-hadoop2.7/bin"
```

adding to .profile or .bash_profile



Hello World Example

```
cd /Users/yanwei/Dropbox/courses/大数据解析/  
lecture_slides/Fu_Chap6_上机/  
machine_learning_with_spark/Chapter_01  
  
spark-submit pythonapp.py
```

```
17/04/04 16:15:00 INFO Executor: Finished task 1.0 in stage 4.0 (TID 9). 2045 bytes result sent to driver  
17/04/04 16:15:00 INFO TaskSetManager: Finished task 1.0 in stage 4.0 (TID 9) in 17 ms on localhost (executor 1)  
17/04/04 16:15:00 INFO Executor: Finished task 0.0 in stage 4.0 (TID 8). 2053 bytes result sent to driver  
17/04/04 16:15:00 INFO TaskSetManager: Finished task 0.0 in stage 4.0 (TID 8) in 21 ms on localhost (executor 0)  
17/04/04 16:15:00 INFO TaskSchedulerImpl: Removed TaskSet 4.0, whose tasks have all completed, from pool  
17/04/04 16:15:00 INFO DAGScheduler: ShuffleMapStage 4 (reduceByKey at /Users/yanwei/Dropbox/courses/大数据解析/Chapter_01/pythonapp.py) finished in 0.022 s  
17/04/04 16:15:00 INFO DAGScheduler: looking for newly runnable stages  
17/04/04 16:15:00 INFO DAGScheduler: running: Set()  
17/04/04 16:15:00 INFO DAGScheduler: waiting: Set(ResultStage 5)  
17/04/04 16:15:00 INFO DAGScheduler: failed: Set()  
17/04/04 16:15:00 INFO DAGScheduler: Submitting ResultStage 5 (PythonRDD[13] at collect at /Users/yanwei/Dropbox/courses/大数据解析/Chapter_01/pythonapp.py:14), which has no missing parents  
17/04/04 16:15:00 INFO MemoryStore: Block broadcast_6 stored as values in memory (estimated size 6.3 KB, 10.221.174.238:51842) (size 6.3 KB)  
17/04/04 16:15:00 INFO MemoryStore: Block broadcast_6_piece0 stored as bytes in memory (estimated size 3.1 KB, 10.221.174.238:51842) (size 3.1 KB)  
17/04/04 16:15:00 INFO BlockManagerInfo: Added broadcast_6_piece0 in memory on 10.221.174.238:51842 (size 3.1 KB, 10.221.174.238:51842)  
17/04/04 16:15:00 INFO SparkContext: Created broadcast 6 from broadcast at DAGScheduler.scala:996  
17/04/04 16:15:00 INFO DAGScheduler: Submitting 2 missing tasks from ResultStage 5 (PythonRDD[13] at collect at /Users/yanwei/Dropbox/courses/大数据解析/Chapter_01/pythonapp.py:14)  
17/04/04 16:15:00 INFO TaskSchedulerImpl: Adding task set 5.0 with 2 tasks  
17/04/04 16:15:00 INFO TaskSetManager: Starting task 0.0 in stage 5.0 (TID 10, localhost, executor driver 0)  
17/04/04 16:15:00 INFO TaskSetManager: Starting task 1.0 in stage 5.0 (TID 11, localhost, executor driver 1)  
17/04/04 16:15:00 INFO Executor: Running task 0.0 in stage 5.0 (TID 10)  
17/04/04 16:15:00 INFO Executor: Running task 1.0 in stage 5.0 (TID 11)  
17/04/04 16:15:00 INFO ShuffleBlockFetcherIterator: Getting 2 non-empty blocks out of 2 blocks  
17/04/04 16:15:00 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 0 ms  
17/04/04 16:15:00 INFO ShuffleBlockFetcherIterator: Getting 1 non-empty blocks out of 2 blocks  
17/04/04 16:15:00 INFO ShuffleBlockFetcherIterator: Started 0 remote fetches in 0 ms  
17/04/04 16:15:00 INFO PythonRunner: Times: total = 2, boot = -20, init = 22, finish = 0  
17/04/04 16:15:00 INFO PythonRunner: Times: total = 1, boot = -21, init = 22, finish = 0  
17/04/04 16:15:00 INFO Executor: Finished task 0.0 in stage 5.0 (TID 10). 2150 bytes result sent to driver  
17/04/04 16:15:00 INFO Executor: Finished task 1.0 in stage 5.0 (TID 11). 2230 bytes result sent to driver  
17/04/04 16:15:00 INFO TaskSetManager: Finished task 1.0 in stage 5.0 (TID 11) in 10 ms on localhost (executor 1)  
17/04/04 16:15:00 INFO TaskSetManager: Finished task 0.0 in stage 5.0 (TID 10) in 11 ms on localhost (executor 0)  
17/04/04 16:15:00 INFO TaskSchedulerImpl: Removed TaskSet 5.0, whose tasks have all completed, from pool  
17/04/04 16:15:00 INFO DAGScheduler: ResultStage 5 (collect at /Users/yanwei/Dropbox/courses/大数据解析/Chapter_01/pythonapp.py:14) finished in 0.014 s  
17/04/04 16:15:00 INFO DAGScheduler: Job 3 finished: collect at /Users/yanwei/Dropbox/courses/大数据解析/Chapter_01/pythonapp.py:14 in 0.014 s  
Total purchases: 5  
Unique users: 4  
Total revenue: 39.91  
Most popular product: iPhone Cover with 2 purchases  
17/04/04 16:15:00 INFO SparkUI: Stopped Spark web UI at http://10.221.174.238:4040  
17/04/04 16:15:00 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!  
17/04/04 16:15:00 INFO MemoryStore: MemoryStore cleared  
17/04/04 16:15:00 INFO BlockManager: BlockManager stopped  
17/04/04 16:15:00 INFO BlockManagerMaster: BlockManagerMaster stopped  
17/04/04 16:15:00 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped  
17/04/04 16:15:00 INFO SparkContext: Successfully stopped SparkContext  
17/04/04 16:15:01 INFO ShutdownHookManager: Shutdown hook called  
17/04/04 16:15:01 INFO ShutdownHookManager: Deleting directory /private/var/folders/6w/j18kjkv9397gxhsfg/  
17/04/04 16:15:01 INFO ShutdownHookManager: Deleting directory /private/var/folders/6w/j18kjkv9397gxhsfg/
```



Two much INFO!!!

You may find the logging statements that get printed in the shell distracting. You can control the verbosity of the logging. To do this, you can create a file in the *conf* directory called *log4j.properties*. The Spark developers already include a template for this file called *log4j.properties.template*. To make the logging less verbose, make a copy of *conf/log4j.properties.template* called *conf/log4j.properties* and find the following line:

```
log4j.rootCategory=INFO, console
```

Then lower the log level so that we show only the WARN messages, and above by changing it to the following:

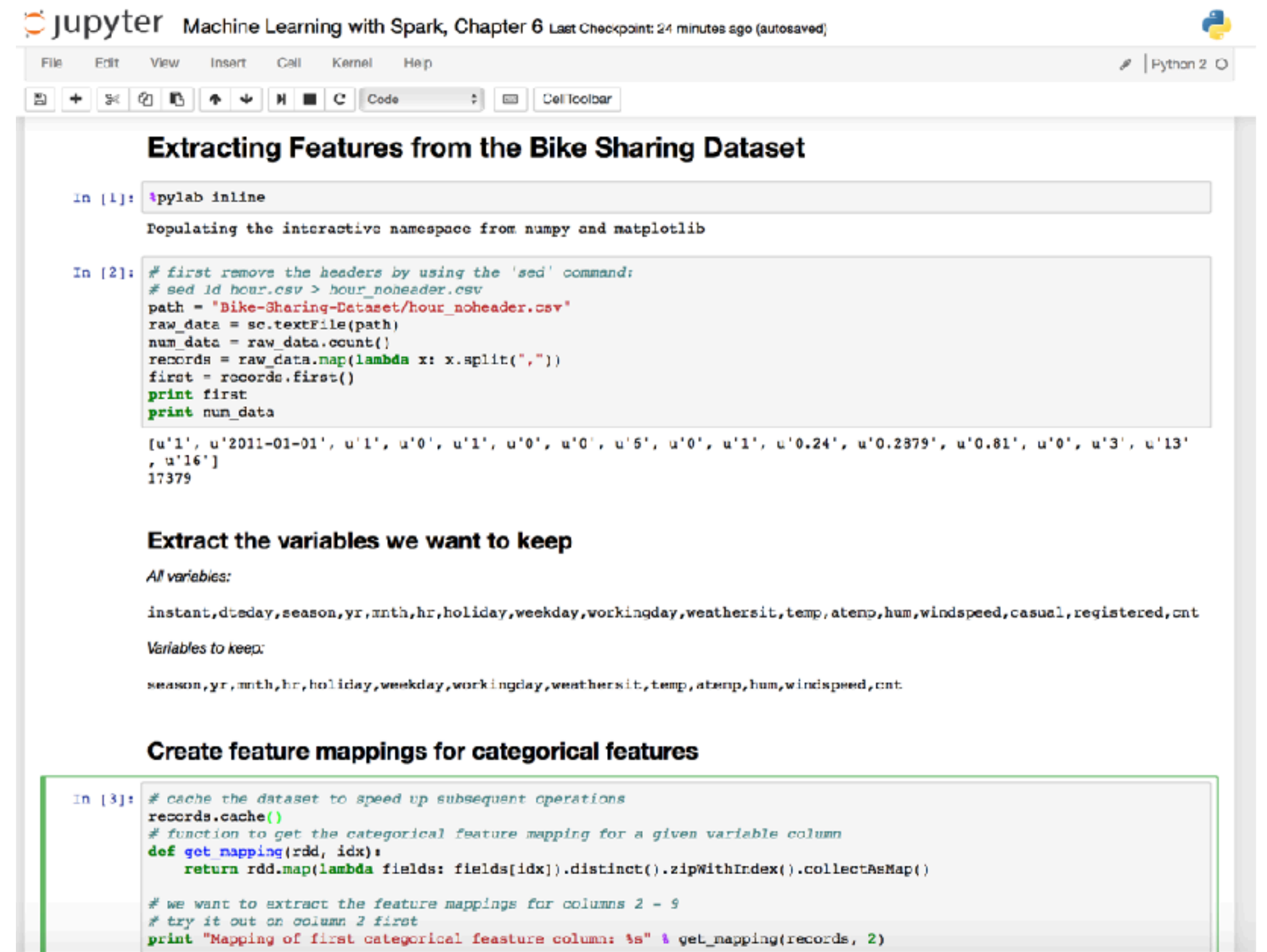
```
log4j.rootCategory=WARN, console
```

```
MacBook-Pro-5:Chapter_01 yanwei$ spark-submit pythonapp.py
17/04/04 16:20:52 WARN Utils: Your hostname, 127.0.0.1 resolves to a loopback address: 1
17/04/04 16:20:52 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
17/04/04 16:20:53 WARN NativeCodeLoader: Unable to load native-hadoop library for your p
/Users/yanwei/software/spark-2.1.0-bin-hadoop2.7/python/lib/pyspark.zip/pyspark/worker.p
ing unequal
/Users/yanwei/software/spark-2.1.0-bin-hadoop2.7/python/lib/pyspark.zip/pyspark/worker.p
ing unequal
Total purchases: 5
Unique users: 4
Total revenue: 39.91
Most popular product: iPhone Cover with 2 purchases
MacBook-Pro-5:Chapter_01 yanwei$
```

Much Better!



Example: Building a Regression Model with Spark



The image shows a Jupyter Notebook titled "Machine Learning with Spark, Chapter 6" with a last checkpoint 24 minutes ago. The notebook is running Python 2. It contains three code cells. The first cell, titled "Extracting Features from the Bike Sharing Dataset", uses `%pylab inline` to populate the interactive namespace with `numpy` and `matplotlib`. The second cell, titled "Extract the variables we want to keep", shows the process of reading a CSV file, removing headers, and printing the first record and the number of data points. The third cell, titled "Create feature mappings for categorical features", shows how to cache the dataset and create a function to map categorical features to numerical values.

```
In [1]: %pylab inline
Populating the interactive namespace from numpy and matplotlib

In [2]: # first removes the headers by using the 'sed' command:
# sed -i hour.csv > hour_noheader.csv
path = "Bike-Sharing-Dataset/hour_noheader.csv"
raw_data = sc.textFile(path)
num_data = raw_data.count()
records = raw_data.map(lambda x: x.split(","))
first = records.first()
print first
print num_data

[u'1', u'2011-01-01', u'1', u'0', u'1', u'0', u'0', u'5', u'0', u'1', u'0.24', u'0.2379', u'0.81', u'0', u'3', u'13', u'16']
17379

Extract the variables we want to keep

All variables:
instant, dteday, season, yr, mnth, hr, holiday, weekday, workingday, weathersit, temp, atemp, hum, windspeed, casual, registered, cnt

Variables to keep:
season, yr, mnth, hr, holiday, weekday, workingday, weathersit, temp, atemp, hum, windspeed, cnt.

Create feature mappings for categorical features

In [3]: # cache the dataset to speed up subsequent operations
records.cache()
# function to get the categorical feature mapping for a given variable column
def get_mapping(rdd, idx):
    return rdd.map(lambda fields: fields[idx]).distinct().zipWithIndex().collectAsMap()

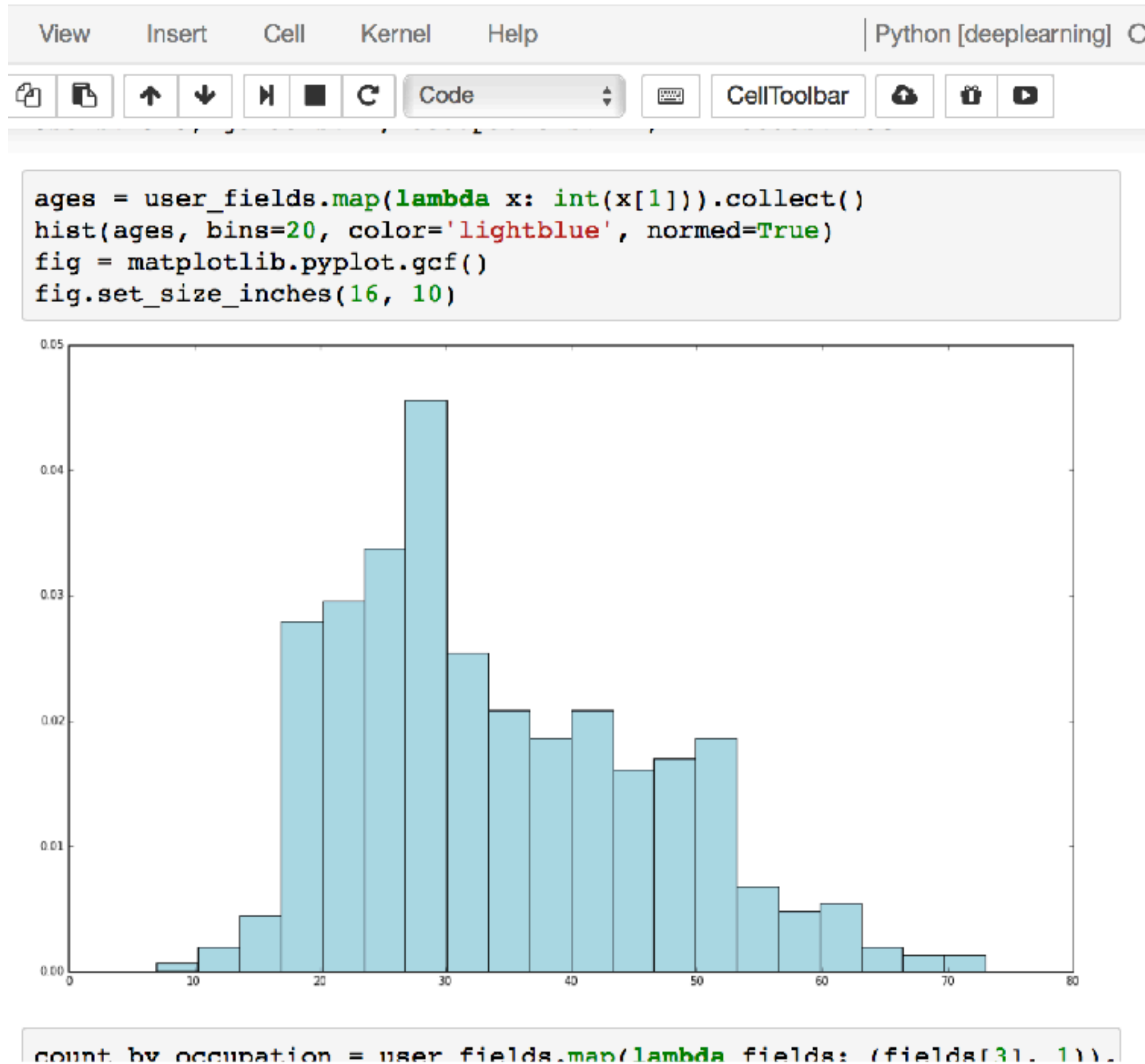
# we want to extract the feature mappings for columns 2 - 9
# try it out on column 2 first
print "Mapping of first categorical feature column: %s" % get_mapping(records, 2)
```


Today's codes

Optional subtitle

Chapter 3: one example of exploring dataset

Learning with Spark, Chapter 3. MovieLens 100k Analysis.



Chapter 6: one example of regression model

upyter Machine Learning with Spark, Chapter 6

The screenshot shows a Jupyter Notebook interface with a menu bar (Edit, View, Insert, Cell, Kernel, Help) and a toolbar. The code cell contains the following Python code:

```
[0.12015, 0.01, 0.10]
Decision Tree feature vector length: 12
```

Training a Regression Model

```
In [7]: from pyspark.mllib.regression import LinearRegressionWithSGD
from pyspark.mllib.tree import DecisionTree
help(LinearRegressionWithSGD.train)
```

Help on method train in module pyspark.mllib.regression:

train(cls, data, iterations=100, step=1.0, miniBatchFraction=1.0, initialWeights=None, regParam=0.0, regType=None, intercept=False, validateData=True, convergenceTol=0.001) method of __builtin__.type instance

Train a linear regression model using Stochastic Gradient Descent (SGD). This solves the least squares regression

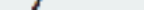
Debug&Errors



```
"/bin/spark-shell".
```

It has the error below. I also tried to install different versions of Spark but all have the same error. This is the second time I'm running Spark. My previous run works fine.

```
log4j:WARN No appenders could be found for logger (org.apache.hadoop.metrics2.lib.Mu
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Using Spark's repl log4j profile: org/apache/spark/log4j-defaults-repl.properties
To adjust logging level use sc.setLogLevel("INFO")
Welcome to
```

 version 1.6.0

```
Using Scala version 2.10.5 (Java HotSpot(TM) 64-Bit Server VM, Java 1.7.0_79)
```

Type in expressions to have them evaluated.

```
Type :help for more information.
```

[illegible]

```
adding:      sudo hostname -s 127.0.0.1
```


Optional subtitle

```
-- -- -- -- --  
at java.lang.Thread.run(Thread.java:745)  
Caused by: java.sql.SQLException: Failed to start database 'metastore_db' with class loader org.apache.  
anon$1@1be7eb93, see the next exception for details.  
at org.apache.derby.impl.jdbc.SQLExceptionFactory.getSQLException(Unknown Source)  
at org.apache.derby.impl.jdbc.SQLExceptionFactory.getSQLException(Unknown Source)  
at org.apache.derby.impl.jdbc.Util.getNextException(Unknown Source)  
at org.apache.derby.impl.jdbc.EmbedConnection.bootDatabase(Unknown Source)  
at org.apache.derby.impl.jdbc.EmbedConnection.<init>(Unknown Source)  
at org.apache.derby.jdbc.InternalDriver$1.run(Unknown Source)  
at org.apache.derby.jdbc.InternalDriver$1.run(Unknown Source)  
at java.security.AccessController.doPrivileged(Native Method)  
at org.apache.derby.jdbc.InternalDriver.getNewEmbedConnection(Unknown Source)  
at org.apache.derby.jdbc.InternalDriver.connect(Unknown Source)  
at org.apache.derby.jdbc.InternalDriver.connect(Unknown Source)  
at org.apache.derby.jdbc.AutoloadedDriver.connect(Unknown Source)  
at java.sql.DriverManager.getConnection(DriverManager.java:664)  
at java.sql.DriverManager.getConnection(DriverManager.java:208)  
at com.jolbox.bonecp.BoneCP.obtainRawInternalConnection(BoneCP.java:361)  
at com.jolbox.bonecp.BoneCP.<init>(BoneCP.java:416)
```

使用hive默认的Derby数据库，由于是内嵌的文件数据库，只支持一个用户的操作访问。否则会抛：

Another instance of Derby may have already booted the database /home/.../metastore_db. 异常。

```
ps -ef | grep spark-shell  
kill -9 Spark-Shell-processID ( example: kill -9 4848)
```

