Xi'an Jiaotong-Liverpool University
西交利物浦大学

DTS303TC Big Data Security and Analytics

Laboratory Session 2 – Setting Up Jupyter and Spark – Weather Dataset

(Duration: 2 hours)

Overview:

Data exploration is the initial step in data analysis, where users explore a large data set in an unstructured way to uncover initial patterns, characteristics, and points of interest. This process isn't meant to reveal every bit of information a dataset holds, but rather to help create a broad picture of important trends and major points to study in greater detail.

In this activity, you will be able to perform the following in Spark:

- Read CSV files into Spark Dataframes.
- Generate summary statistics.
- Compute correlation coefficients between two columns.

PART 1: Instructions for Downloading Hands On Datasets

Task Instructions

This part describes the steps to download the dataset and scripts necessary for the Spark Hands On exercises in this lab. Before proceeding, make sure have you downloaded and installed VirtualBox and the Cloudera virtual machine. Instructions for this process can be found in the Reading called Downloading and Installing the Cloudera VM.

Step1: Start the Cloudera VM. The Spark Hands On exercises in this lab use the Cloudera Virtual Machine, so we will download the weather data onto the VM. Start the VM in VirtualBox and perform the remaining steps in the VM.

Step 2: Download the dataset from 'Resources for Lab session 2' by first downloading the file *big-data-4.zip*

Step 3: Un-compress the dataset. Open a terminal shell by clicking on the terminal shell icon in the top toolbar. In the terminal, run: *cd Downloads* and *unzip -o big-data-4.zip*

Step 4: Install tools. Change directories to *cd big-data-4* and run *./setup.sh* to install tools and libraries.

NOTE: If set up failed. Do these steps. First run *vi setup.sh.* Then click and download http://repo.continuum.io/archive/Anaconda3-4.0.0-Linux-x86_64.sh file and move file to the big-data-4 folder. Next comment out these two lines of text. Final, run *./setup.sh*.

```
#!/bin/bash

# upgrade spark python to work with python 3
sudo yum upgrade -y spark-python hive

# remove old derby jar so only one version is on spark's classpath
sudo rm /usr/lib/flume-ng/lib/derby-10.8.2.2.jar

# decompress datasets
gzip -d *.csv.gz

# download and install anaconda for pandas, jupyter
#rm -f Anaconda3-4.0.0-Linux-x86_64.sh
#wget http://repo.continuum.io/archive/Anaconda3-4.0.0-Linux-x86_64.sh
bash Anaconda3-4.0.0-Linux-x86_64.sh

# add spark-csv jars to classpath
echo export SPARK_CLASSPATH="$(pwd)/lib/spark-csv_2.10-1.5.0.jar:$(pwd)/lib/commons-csv-1.1.jar" >> ~/.bashrc

# set environment variables to load spark libs in jupyter
echo "export PYSPARK_DRIVER_PYTHON_OPTS=\"notebook\"" >> ~/.bashrc
echo "export PYSPARK_DRIVER_PYTHON=jupyter"  >> ~/.bashrc

echo "Run 'source ~/.bashrc' to complete the setup."
```

During the setup process, Anaconda will give you a series of prompts. First, press enter to continue the installation:

```
[cloudera@quickstart big-data-4]$ ./setup.sh
Loaded plugins: fastestmirror, security
Loading mirror speeds from cached hostfile
YumRepo Error: All mirror URLs are not using ftp, http[s] or file.
 Eg. Invalid release/repo/arch combination/
removing mirrorlist with no valid mirrors: /var/cache/yum/x86_64/6/base/mirrorlist.txt
Error: Cannot find a valid baseurl for repo: base
rm: cannot remove `/usr/lib/flume-ng/lib/derby-10.8.2.2.jar': No such file or directory
gzip: *.csv.gz: No such file or directory

Welcome to Anaconda3 4.0.0 (by Continuum Analytics, Inc.)

In order to continue the installation process, please review the license
agreement.
Please, press ENTER to continue
>>>
```

Next, read and accept the license:

```
Please answer 'yes' or 'no':
>>> yes
```

Next, press enter to accept the default installation location:

```
Anaconda3 will now be installed into this location:
/home/cloudera/anaconda3

  - Press ENTER to confirm the location
  - Press CTRL-C to abort the installation
  - Or specify a different location below

[/home/cloudera/anaconda3] >>>
```

Next, enter yes when it asks if you want to prepend the install location to PATH:

```
Do you wish the installer to prepend the Anaconda3 install location
to PATH in your /home/cloudera/.bashrc ? [yes|no]
[no] >>> yes
```

The setup of tools and datasets should continue.

Finally, run *source $HOME/.bashrc*

PART 2: Instructions for Starting Jupyter

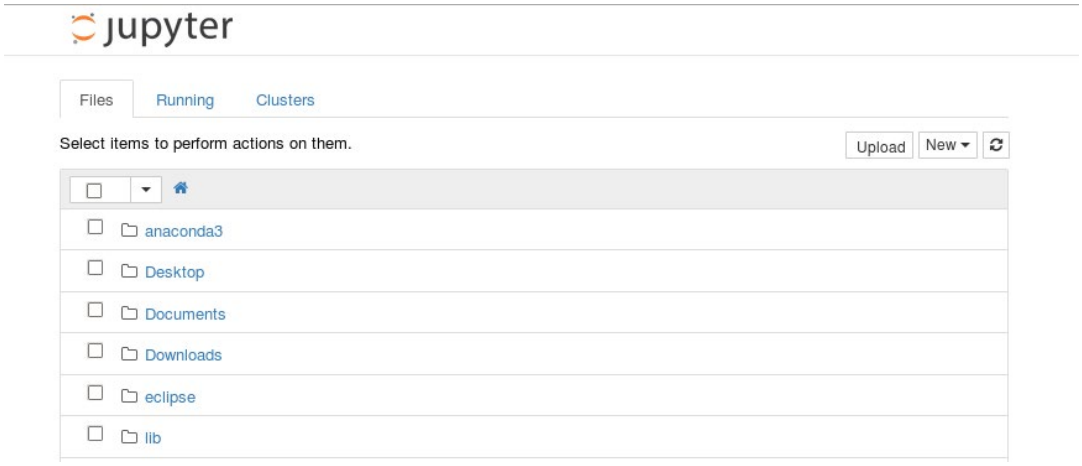
Task Instructions

Please use the following WordCount program Instructions on Cloudera VM in VirtualBox.

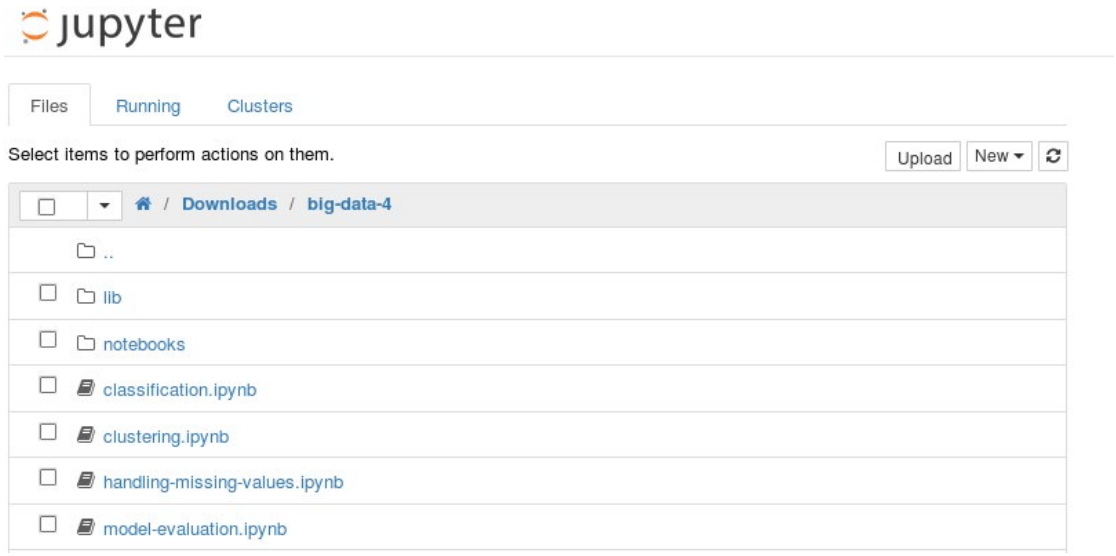
Step 1: All of the Spark Hands On exercises in this lab are executed in a Jupyter Notebook. To start Jupyter, open a new terminal window and run *pyspark*

```
[cloudera@quickstart ~]$ pyspark
[I 06:27:09.160 NotebookApp] Writing notebook server cookie secret to /home/clou
dera/.local/share/jupyter/runtime/notebook_cookie_secret
[I 06:27:09.219 NotebookApp] The port 8888 is already in use, trying another ran
dom port.
[I 06:27:09.224 NotebookApp] Serving notebooks from local directory: /home/cloud
era
[I 06:27:09.224 NotebookApp] 0 active kernels
[I 06:27:09.224 NotebookApp] The Jupyter Notebook is running at: http://localhos
t:8889/
[I 06:27:09.224 NotebookApp] Use Control-C to stop this server and shut down all
 kernels (twice to skip confirmation).
```

This will start Firefox (if not already running), and open a new tab showing the current directory:



You can find the notebooks for this lab in the Downloads/big-data-4 directory. Click on Downloads and then click on *big-data-4*.



You can open Jupyter Notebooks by clicking on the .ipynb files.

When you are done, Jupyter can be stopped by entering control-c in the terminal window and then entering yes.

PART 3: Description of Daily Weather Dataset

Task Instructions

This reading describes the daily weather dataset used in the hands-on activities in this lab.

The file daily_weather.csv is a comma-separated file that contains weather data. This data comes from a weather station located in San Diego, California. The weather station is equipped with

sensors that capture weather-related measurements such as air temperature, air pressure, and relative humidity. Data was collected for a period of three years, from September 2011 to September 2014, to ensure that sufficient data for different seasons and weather conditions is captured.

Sensor measurements from the weather station were captured at one-minute intervals. These measurements were then processed to generate values to describe daily weather. Since this dataset was created to classify low-humidity days vs. non-low-humidity days (that is, days with normal or high humidity), the variables included are weather measurements in the morning, with one measurement, namely relatively humidity, in the afternoon. The idea is to use the morning weather values to predict whether the day will be low-humidity or not based on the afternoon measurement of relatively humidity.

Each row in daily_weather.csv captures weather data for a separate day. Each row, or sample, consists of the following variables:

Variable	Description	Unit of Measure
number	unique number for each row	NA
air_pressure_9am	air pressure averaged over a period from 8:55am to 9:04am	hectopascals
air_temp_9am	air temperature averaged over a period from 8:55am to 9:04am	degrees Fahrenheit
avg_wind_direction_9am	wind direction averaged over a period from 8:55am to 9:04am	degrees, with 0 means coming from the North, and increasing clockwise
avg_wind_speed_9am	wind speed averaged over a period from 8:55am to 9:04am	miles per hour
max_wind_direction_9am	wind gust direction averaged over a period from 8:55am to 9:04am	degrees, with 0 being North and increasing clockwise
max_wind_speed_9am	wind gust speed averaged over a period from 8:55am to 9:04am	miles per hour
rain_accumulation_9am	amount of rain accumulated in the 24 hours prior to 9am	millimeters
rain_duration_9am	amount of time rain was recorded in the 24 hours prior to 9am	seconds
relative_humidity_9am	relative humidity averaged over a period from 8:55am to 9:04am	percent
relative_humidity_3pm	relative humidity averaged over a period from 2:55pm to 3:04pm	percent

PART 4: Data Exploration in Spark

Task Instructions

In this activity, you will be programming in a Jupyter Python Notebook. If you have not already started the Jupyter Notebook server, see the instructions in the Reading Instructions for Starting Jupyter.

Step 1: Create new Jupyter Python Notebook. Create a new Python Notebook by clicking on New, and then click on Python 3.

Step 2: Load data into Spark DataFrame. First, we need to import the SQLContext class:

```
In [1]: from pyspark.sql import SQLContext
```

Next, we create an SQLContext:

```
In [2]: sqlContext=SQLContext(sc)
```

Then we read the weather data into a DataFrame:

```
In [3]: df=sqlContext.read.load('file:///home/cloudera/Downloads/big-data-4/daily_weather.csv',
                                format='com.databricks.spark.csv',
                                header='true',inferSchema='true')
```

The first argument specifies the URL to the daily_weather.csv file, the second argument specifies the spark-csv format, the third argument says the first line in daily_weather.csv is the header, and the fourth argument says to infer the data types.

We use spark-csv to read CSV data directly into a Spark DataFrame. In Spark 1.x (the version on the Cloudera VM), spark-csv is an external package, but spark-csv is integrated with Spark 2.x.

Step 3: Look at data columns and types. We can see the columns in the DataFrame by looking at the columns attribute:

```
In [4]: df.columns

Out[4]: ['number',
         'air_pressure_9am',
         'air_temp_9am',
         'avg_wind_direction_9am',
         'avg_wind_speed_9am',
         'max_wind_direction_9am',
         'max_wind_speed_9am',
         'rain_accumulation_9am',
         'rain_duration_9am',
         'relative_humidity_9am',
         'relative_humidity_3pm']
```

The data type for each column by calling printSchema():

```
In [6]: df.printSchema()

root
 |-- number: integer (nullable = true)
 |-- air_pressure_9am: double (nullable = true)
 |-- air_temp_9am: double (nullable = true)
 |-- avg_wind_direction_9am: double (nullable = true)
 |-- avg_wind_speed_9am: double (nullable = true)
 |-- max_wind_direction_9am: double (nullable = true)
 |-- max_wind_speed_9am: double (nullable = true)
 |-- rain_accumulation_9am: double (nullable = true)
 |-- rain_duration_9am: double (nullable = true)
 |-- relative_humidity_9am: double (nullable = true)
 |-- relative_humidity_3pm: double (nullable = true)
```

Step 4： Print summary statistics. We can print the summary statistics for all the columns using the describe() method:

```
In [7]: df.describe().toPandas().transpose()
```

Out[7]:

	0	1	2	3	4
summary	count	mean	stddev	min	max
number	1095	547.0	316.0991405661627	0	1094
air_pressure_9am	1092	918.8825513138097	3.1827028973312683	907.9900000000024	929.32000000
air_temp_9am	1090	64.93300141287075	11.170386444290147	36.752000000000685	98.90599999
avg_wind_direction_9am	1091	142.23551070057584	69.10616647985127	15.500000000000046	343.4
avg_wind_speed_9am	1092	5.50828424225493	4.550728366514393	0.69345139999974	23.55497819
max_wind_direction_9am	1092	148.9535179651692	67.2072192586138	28.89999999999991	312.19999999
max_wind_speed_9am	1091	7.019513529175272	5.595642950535744	1.1855782000000479	29.84077959
rain_accumulation_9am	1089	0.20307895225211126	1.5932201151579855	0.0	24.01999999
rain_duration_9am	1092	294.1080522756142	1597.346890012345	0.0	17704.0
relative_humidity_9am	1095	34.24140205923539	25.460433064807265	6.090000000001012	92.62000000
relative_humidity_3pm	1095	35.34472714825902	22.51379213656124	5.3000000000006855	92.25000000

We can also see the summary statistics for just one column:

```
In [8]: df.describe('air_pressure_9am').show()
```

```
+-------+------------------+
|summary|  air_pressure_9am|
+-------+------------------+
|  count|              1092|
|   mean| 918.8825513138097|
| stddev|3.1827028973312683|
|    min| 907.9900000000024|
|    max| 929.3200000000012|
+-------+------------------+
```

Let's count the number of columns and rows in the DataFrame:

```
In [9]: len(df.columns)
Out[9]: 11

In [10]: df.count()
Out[10]: 1095
```

The number of rows in the DataFrame is 1095, but the summary statistics for air_pressure_9am says there are only 1092 rows. These are different since 1095 - 1092 = 3 rows have missing values.

Step 5: Drop rows with missing values. Let's drop the rows with missing values in the air_pressure_9am column:

```
In [11]: df2=df.na.drop(subset=['air_pressure_9am'])
```

Now let's see the total number of rows:

```
In [12]: df2.count()
Out[12]: 1092
```

The total number of rows and number of rows in the summary statistics are now the same.

Step 6： Compute correlation between two columns. We can compute the correlation between two columns in a DataFrame by using the corr() method. Let's compute the correlation between rain_accumulation_9am and rain_duration_9am:

```
In [13]: df2.stat.corr('rain_accumulation_9am','rain_duration_9am')
Out[13]: 0.7298253479609015
```