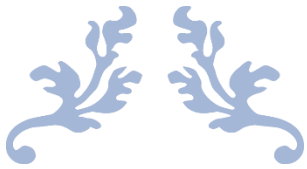


PRÁCTICA BIG DATA 2

Spark

ÁLVARO PÉREZ GARCÍA
UNIVERSIDAD DE HUELVA
Máster en Ingeniería Informática

Instalación

Para el desarrollo de la práctica se ha utilizado un sistema con las siguientes características:

Versión Linux

```
alvaro@alvaro-VirtualBox:~/Escritorio$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 16.04.3 LTS
Release:        16.04
Codename:       xenial
alvaro@alvaro-VirtualBox:~/Escritorio$
```

Versión Java

```
alvaro@alvaro-VirtualBox:~/Escritorio$ java -version
java version "1.8.0_161"
Java(TM) SE Runtime Environment (build 1.8.0_161-b12)
Java HotSpot(TM) Client VM (build 25.161-b12, mixed mode)
alvaro@alvaro-VirtualBox:~/Escritorio$
```

Version Python

```
alvaro@alvaro-VirtualBox:~/Escritorio$ python3 -V
Python 3.6.3 :: Anaconda custom (32-bit)
alvaro@alvaro-VirtualBox:~/Escritorio$
```

Dependencias

Ha sido necesario instalar Spark en el equipo, usando los siguientes comandos:

Instalación de Anaconda (Intérprete de Pyhton)

```
# Go to home directory
cd ~


# You can change what anaconda version you want at
# https://repo.continuum.io/archive/
wget https://repo.continuum.io/archive/Anaconda3-5.0.1-Linux-
x86_64.sh
bash Anaconda3-5.0.1-Linux-x86_64.sh -b -p ~/anaconda
rm Anaconda3-5.0.1-Linux-x86_64.sh
echo 'export PATH="~/anaconda/bin:$PATH"' >> ~/.bashrc


# Refresh basically
source .bashrc


conda update conda
```

Instalación de Spark

```
cd ~
wget https://archive.apache.org/dist/spark/spark-2.0.0/spark-2.0.0-bin-hadoop2.7.tgz
tar -zxvf spark-2.0.0-bin-hadoop2.7.tgz
nano .bashrc
```

Añadir al final del archivo este código

```
Function
snotebook
()
        {
        #Spark path (based on your computer)
        SPARK_PATH=~/spark-2.0.0-bin-hadoop2.7


        export PYSPARK_DRIVER_PYTHON="jupyter"
        export PYSPARK_DRIVER_PYTHON_OPTS="notebook"


        # For python 3 users, you have to add the line below or you will get an
        error
        #export PYSPARK_PYTHON=python3


        $SPARK_PATH/bin/pyspark --master local[2]
        }
```

```
source .bashrc
```

Instalación de Hadoop

```
Wget http://www-us.apache.org/dist/hadoop/common/hadoop-2.8.3/hadoop-2.8.3.tar.gz
tar -zxvf hadoop-2.8.0.tar.gz
export HADOOP_HOME=~/hadoop-2.8.3
```

Experimentación

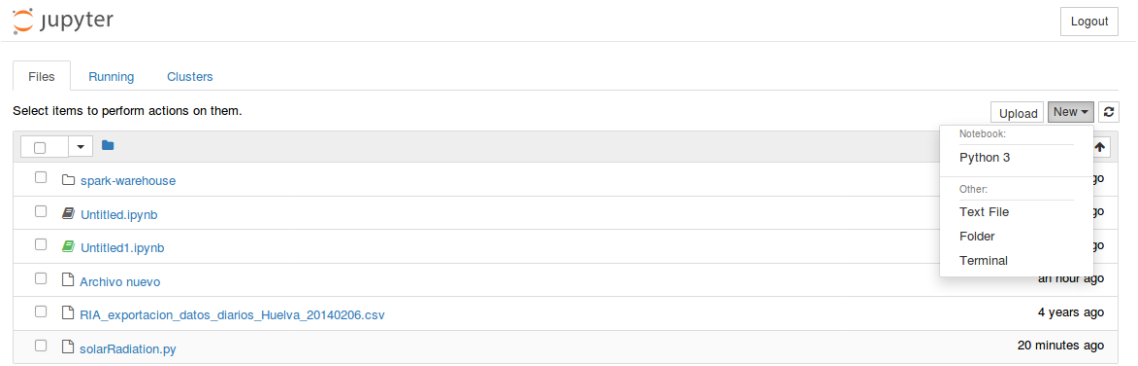
Se ha optado por utilizar una experimentación basada en un programa o script pyhton ejecutable desde el programa “Jupyter”, instalado previamente junto con Anaconda.

De tal modo que para realizar la experimentación procederemos de la siguiente manera:

En primer lugar ejecutaremos el siguiente comando:

```
jupyter netbook
```

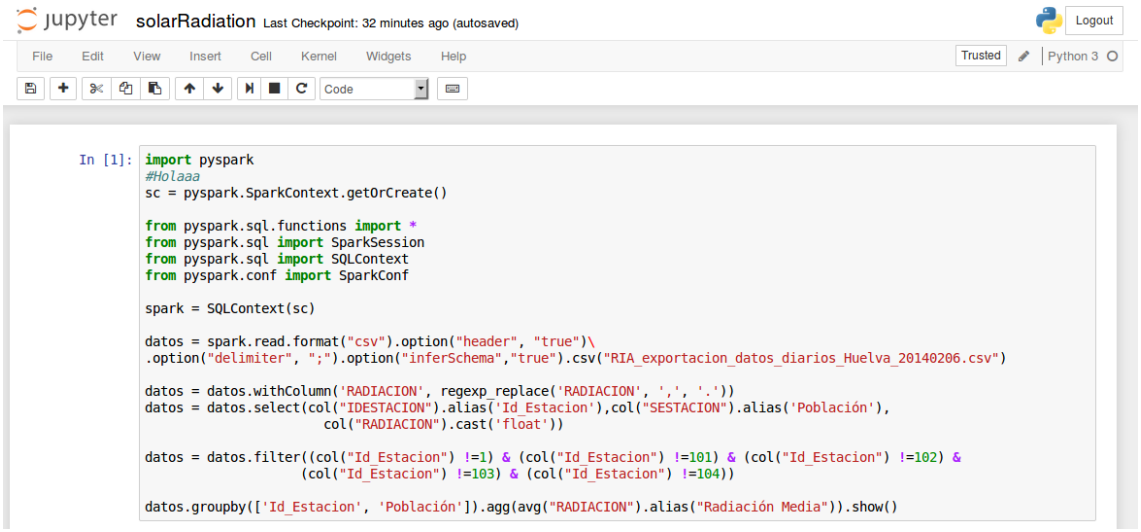
Esto abrirá un servidor local de Jupyter, un IDE web para el desarrollo de Python, con el cual podremos abrir un nuevo archivo que permitirá ejecutar código Python.



Una vez en el, ejecutaremos el código de nuestro programa, el cual ha sido diseñado de la siguiente manera:

```
#Importamos pyspark para realizar la experimentación
import pyspark

sc = pyspark.SparkContext.getOrCreate()

#Importamos la librería sql para tratar el csv como una tabla sql y facilitar el tratamiento
de datos
from pyspark.sql.functions import *
from pyspark.sql import SparkSession
from pyspark.sql import SQLContext
from pyspark.conf import SparkConf

spark = SQLContext(sc)

#Leemos el archivo indicando que el delimitador es “;” y que el archivo desde el que lee es el
que tiene los datos que queremos tratar
datos = spark.read.format("csv").option("header", "true")\
.option("delimiter",
";").option("inferSchema","true").csv("RIA_exportacion_datos_diarios_Huelva_20140206.csv")

#Reemplazamos en los valores de radiación las comas por puntos para realizar las operaciones
correspondientes y que no tengamos problemas de casting
datos = datos.withColumn('RADIACION', regexp_replace('RADIACION', ',', '.'))
#Seleccionamos las columnas que necesitamos para los resultados
datos =
datos.select(col("IDESTACION").alias('Id_Estacion'),col("SESTACION").alias('Población'),
                        col("RADIACION").cast('float'))
```

```
#Filtramos las estaciones que no queremos procesar
datos = datos.filter((col("Id_Estacion") !=1) & (col("Id_Estacion") !=101) &
(col("Id_Estacion") !=102) &
                    (col("Id_Estacion") !=103) & (col("Id_Estacion") !=104))

#Agrupamos por estación, calculamos la media de radiación correspondiente y lo mostramos por
pantalla
datos = datos.groupby(['Id_Estacion', 'Población']).agg(avg("RADIACION").alias("Radiación
Media")).show()
```



Esto nos mostrará el esquema de la tabla correspondiente:

```
+-----------+--------------------+------------------+
|Id_Estacion|           Población|   Radiación Media|
+-----------+--------------------+------------------+
|          9|La Palma del Condado|18.071459669487474|
|          3|           Gibraleón| 18.70950440160565|
|          2|               Lepe|18.879762046745903|
|          7| La Puebla de Guzmán|17.962850858576193|
|          5|              Niebla| 18.07869350540072|
|          4|              Moguer|18.341574146820463|
|         10|             Almonte|18.123927680463083|
|          6|              Aroche|17.919251145115762|
|          8|         El Campillo|18.261276060114252|
+-----------+--------------------+------------------+
```

Finalmente, ejecutando el siguiente comando nos exportará a .csv los resultados obtenidos:

```
datos.coalesce(1).write.option("header", "true").csv("radiacionmedia.csv")
```

Id_Estacion	Población	Radiación Media
9	La Palma del Condado	18.071459669487474
3	Gibraleón	18.70950440160565
2	Lepe	18.879762046745903
7	La Puebla de Guzmán	17.962850858576193
5	Niebla	18.07869350540072
4	Moguer	18.341574146820463
10	Almonte	18.123927680463083
6	Aroche	17.919251145115762
8	El Campillo	18.261276060114252