Install Java, Spark, and Findspark

Java 8, Apache Spark 2.2.1, FindSpark

```
!apt-get install openjdk-8-jdk-headless -qq > /dev/null
```

```
!java -version
```

```
openjdk version "11.0.3" 2019-04-16
OpenJDK Runtime Environment (build 11.0.3+7-Ubuntu-1ubuntu218.04.1)
OpenJDK 64-Bit Server VM (build 11.0.3+7-Ubuntu-1ubuntu218.04.1, mixed mode, s
```

```
!wget --no-cookies --no-check-certificate 'https://archive.apache.org/dist/spark/spa
```

```
--2019-05-17 18:25:49--  https://archive.apache.org/dist/spark/spark-2.2.1/spa
Resolving archive.apache.org (archive.apache.org)... 163.172.17.199
Connecting to archive.apache.org (archive.apache.org)|163.172.17.199|:443... c
HTTP request sent, awaiting response... 200 OK
Length: 200934340 (192M) [application/x-gzip]
Saving to: ‘spark-2.2.1-bin-hadoop2.7.tgz.1’

spark-2.2.1-bin-had 100%[===================>] 191.62M  32.4MB/s    in 6.8s

2019-05-17 18:25:57 (28.0 MB/s) - ‘spark-2.2.1-bin-hadoop2.7.tgz.1’ saved [200
```

```
!ls -l
```

```
total 393272
-rw-r--r--  1 root root    822526 May 17 18:25 CleanDataset.csv
drwxr-xr-x  1 root root      4096 May 15 16:23 sample_data
drwxrwxr-x 12 1001 1001      4096 Nov 24  2017 spark-2.2.1-bin-hadoop2.7
-rw-r--r--  1 root root 200934340 Nov 25  2017 spark-2.2.1-bin-hadoop2.7.tgz
-rw-r--r--  1 root root 200934340 Nov 25  2017 spark-2.2.1-bin-hadoop2.7.tgz.1
drwxr-xr-x  2 root root      4096 May 17 18:25 spark-warehouse
```

```
!rm -r spark-2.3.1-bin-hadoop2.7.tgz
```

```
rm: cannot remove 'spark-2.3.1-bin-hadoop2.7.tgz': No such file or directory
```

```
!rm -r spark-2.3.1-bin-hadoop2.7.tgz.1
```

```
rm: cannot remove 'spark-2.3.1-bin-hadoop2.7.tgz.1': No such file or directory
```

```
!tar xf  spark-2.2.1-bin-hadoop2.7.tgz
```

```
!ls
```

```
CleanDataset.csv            spark-2.2.1-bin-hadoop2.7.tgz
sample_data                 spark-2.2.1-bin-hadoop2.7.tgz.1
spark-2.2.1-bin-hadoop2.7   spark-warehouse
```

```
!which gzip
!gzip -V
```

```
/bin/gzip
gzip 1.6
Copyright (C) 2007, 2010, 2011 Free Software Foundation, Inc.
Copyright (C) 1993 Jean-loup Gailly.
This is free software.  You may redistribute copies of it under the terms of
the GNU General Public License <http://www.gnu.org/licenses/gpl.html>.
There is NO WARRANTY, to the extent permitted by law.

Written by Jean-loup Gailly.
```

```
!pip install -q findspark
```

Set Environment Variables

Setting the locations where Spark and Java are installed.

```
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = "/content/spark-2.2.1-bin-hadoop2.7"
```

Starting a SparkSession

This will start a local Spark session.

```
import findspark
findspark.init()
from pyspark.sql import SparkSession

spark = SparkSession.builder.master("local[*]").getOrCreate()
```

Creating And Displaying A Sample Dataframe

```
df = spark.createDataFrame([{"hello": "world"} for x in range(1000)])
df.show(30)
```

```
/content/spark-2.2.1-bin-hadoop2.7/python/pyspark/sql/session.py:336: UserWarr
  warnings.warn("inferring schema from dict is deprecated,"
+-----+
|hello|
+-----+
|world|
|world|
|world|
|world|
|world|
|world|
|world|
|world|
|world|
|world|
|world|
|world|
|world|
|world|
|world|
|world|
|world|
|world|
|world|
|world|
|world|
|world|
|world|
|world|
|world|
|world|
|world|
|world|
|world|
|world|
+-----+
only showing top 30 rows
```

Flint

```
!conda create -n flint python=3.5 pandas
```

```
/bin/bash: conda: command not found
```

```
#Importing libraries
from pyspark.ml.regression import LinearRegression
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.linalg import Vectors, VectorUDT
from pyspark.ml.feature import StandardScaler
from pyspark.ml import Pipeline
```

```
from pyspark.sql.functions import *
from pyspark import SparkContext
from pyspark.sql import SQLContext


#Loading the dataset
data = spark.read.csv(r'CleanDataset.csv',inferSchema=True,header=True)
data.show()
data.printSchema()
data.describe()
# inspect.getfullargspec(spark.read.csv)
```

```
+---------+---------+----------+----------+
|    S_avg|   Ds_avg|    Ws_avg|    Ot_avg|
+---------+---------+----------+----------+
|3.0899999|    88.18|       1.9| 5.3000002|
|1.8099999|    23.01|       0.2| 4.8899999|
|     1.89|38.110001|0.18000001| 4.8000002|
|   177.53|   1181.7|      5.04|      1.79|
|191.96001|  1200.48| 5.3099999|0.85000002|
|2.8900001|159.92999| 3.1500001|      6.73|
|  1259.62|  1800.15|       9.0| 4.6900001|
|95.709999|  1042.71| 4.5300002|     10.66|
|   921.19|   1781.9|      7.79| 5.8600001|
|805.09998|  1781.65| 7.6399999| 6.9299998|
|572.60999|1660.5601| 6.7199998| 7.1199999|
|   137.25|  1114.23| 4.9899998| 6.2600002|
|   143.42|1121.8199| 5.0100002| 6.6399999|
|162.32001|  1155.42| 4.6500001| 7.2199998|
|   129.45|  1086.96|      4.77| 6.4400001|
|2.1400001|     1.95|       0.1| 5.8800001|
|   144.64|  1130.66| 5.1999998| 8.2600002|
|76.980003|997.72998|      4.54| 6.4099998|
|129.35001|  1097.42| 4.9699998| 7.9699998|
|   172.92|   1177.1| 5.4899998| 6.1500001|
+---------+---------+----------+----------+
only showing top 20 rows

root
 |-- S_avg: double (nullable = true)
 |-- Ds_avg: double (nullable = true)
 |-- Ws_avg: double (nullable = true)
 |-- Ot_avg: double (nullable = true)

DataFrame[summary: string, S_avg: string, Ds_avg: string, Ws_avg: string, Ot_a
```

```
#Creating a features column to be used
vectorAssembler = VectorAssembler(inputCols=['S_avg','Ws_avg','Ot_avg'], outputCol='
output=vectorAssembler.transform(data)
```

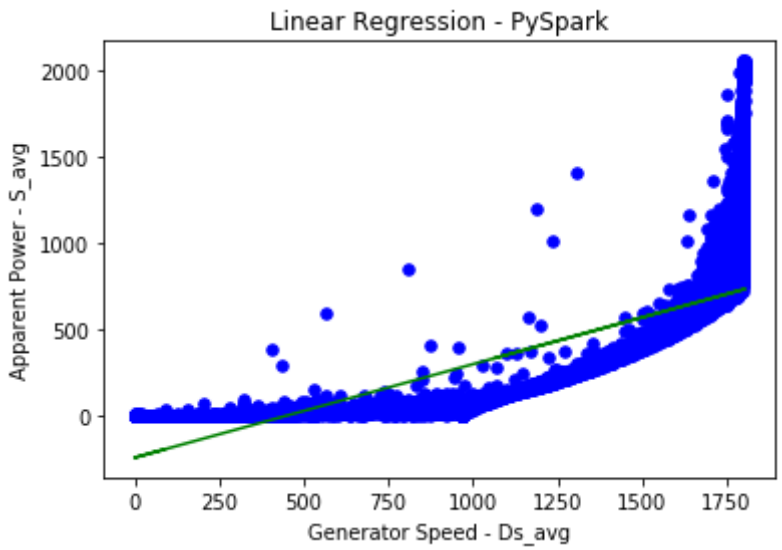
```
#Fitting the model
final_data=output.select('features','Ds_avg')

final_data.show()

train_data,test_data = final_data.randomSplit([0.7,0.3])


train_data.describe().show()
# test = test_data.describe().show()
test = test_data.describe()
test.show()
lr = LinearRegression(labelCol='Ds_avg')

lr_model=lr.fit(train_data)
print("Coefficients: " + str(lr_model.coefficients))
print("Intercept: " + str(lr_model.intercept))
trainingSummary = lr_model.summary
print("RMSE: %f" % trainingSummary.rootMeanSquaredError)
print("r2: %f" % trainingSummary.r2)
test_results = lr_model.evaluate(test_data)

test_results.residuals.show()

test_results.rootMeanSquaredError
test_results.r2
final_data.describe().show()

unlabeled_data=test_data.select('features')
unlabeled_data.show()

Predictions=lr_model.transform(unlabeled_data)
Predictions.show()
```

```
+--------------------+---------+
|            features|   Ds_avg|
+--------------------+---------+
|[3.0899999,1.9,5....|    88.18|
|[1.8099999,0.2,4....|    23.01|
|[1.89,0.18000001,...|38.110001|
|  [177.53,5.04,1.79]|   1181.7|
|[191.96001,5.3099...|  1200.48|
|[2.8900001,3.1500...|159.92999|
|[1259.62,9.0,4.69...|  1800.15|
|[95.709999,4.5300...|  1042.71|
|[921.19,7.79,5.86...|   1781.9|
|[805.09998,7.6399...|  1781.65|
|[572.60999,6.7199...|1660.5601|
|[137.25,4.9899998...|  1114.23|
|[143.42,5.0100002...|1121.8199|
|[162.32001,4.6500...|  1155.42|
|[129.45,4.77,6.44...|  1086.96|
|[2.1400001,0.1,5....|     1.95|
|[144.64,5.1999998...|  1130.66|
|[76.980003,4.54,6...|997.72998|
|[129.35001,4.9699...|  1097.42|
|[172.92,5.4899998...|   1177.1|
+--------------------+---------+
only showing top 20 rows

+-------+------------------+
|summary|            Ds_avg|
+-------+------------------+
|  count|             17861|
|   mean|1105.8873141969223|
| stddev| 589.0255868455521|
|    min|             -0.09|
|    max|           1803.16|
+-------+------------------+

+-------+------------------+
|summary|            Ds_avg|
+-------+------------------+
|  count|              7610|
|   mean|1110.0731074779228|
| stddev| 581.0646375602546|
|    min|               0.0|
|    max|           1803.37|
+-------+------------------+

Coefficients: [0.015828128463414186,182.93372311124952,-0.7159260530232182]
Intercept: 107.56443185886623
RMSE: 340.379108
r2: 0.666050
+-------------------+
|          residuals|
+-------------------+
|-100.35505650492242|
| -47.23209894174102|
| -99.47446745970386|
| -101.4125365031896|
```

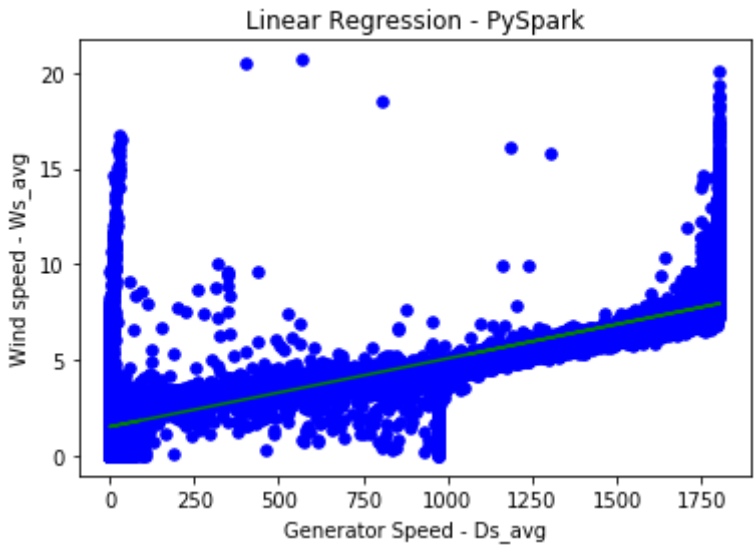
```
|-121.19792536193748|
|-248.45439047206898|
| -409.2481742166878|
|-506.05641049704616|
| -529.6261935032808|
| -816.8884254787154|
|-105.93227873925792|
|-105.81057131024397|
|-100.51271823150174|
|  -90.0480404990164|
|  -98.8517700748583|
| -90.79760197834709|
| -90.46111673342618|
| -78.67127174754728|
| -96.76900992311343|
|-115.87853793468484|
+-------------------+
only showing top 20 rows

+-------+------------------+
|summary|            Ds_avg|
+-------+------------------+
|  count|             25471|
|   mean|1107.1379084754542|
| stddev| 586.6501388637288|
|    min|             -0.09|
|    max|           1803.37|
+-------+------------------+

+--------------------+
|            features|
+--------------------+
|     [0.0,0.0,10.07]|
|     [0.0,0.0,10.13]|
|      [0.0,0.0,11.3]|
|    [0.0,0.13,12.45]|
|    [0.0,0.22,26.43]|
|    [0.0,0.83,15.26]|
|[0.0,1.6900001,10...|
|     [0.0,2.23,13.2]|
|    [0.0,2.57,23.98]|
|    [0.0,3.99,18.37]|
|     [0.01,0.0,2.28]|
|     [0.01,0.0,2.45]|
|[0.01,0.0,9.8500004]|
|    [0.01,0.0,10.96]|
|    [0.01,0.0,12.17]|
|    [0.01,0.0,23.42]|
|    [0.01,0.0,23.89]|
|    [0.01,0.0,28.15]|
|    [0.01,0.04,25.3]|
|   [0.01,0.14,24.16]|
+--------------------+
only showing top 20 rows

+--------------------+------------------+
|            features|        prediction|
+--------------------+------------------+
```

```
|      [0.0,0.0,10.07]|100.35505650492242|
|      [0.0,0.0,10.13]|100.31210094174102|
|       [0.0,0.0,11.3]| 99.47446745970386|
|     [0.0,0.13,12.45]| 122.4325365031896|
|     [0.0,0.22,26.43]|128.88792536193748|
|     [0.0,0.83,15.26]|  248.474390472069|
|[0.0,1.6900001,10...| 409.2481742166878|
|      [0.0,2.23,13.2]|506.05641049704616|
|     [0.0,2.57,23.98]| 560.5361935032807|
|     [0.0,3.99,18.37]| 824.3184254787153|
|      [0.01,0.0,2.28]|105.93227873925792|
|      [0.01,0.0,2.45]|105.81057131024397|
|[0.01,0.0,9.8500004]|100.51271823150174|
```

```
# Ds_avg vs S_avg

from pyspark import SQLContext, SparkConf, SparkContext
import matplotlib.pyplot as plt
import numpy as np
from numpy import polyfit
%matplotlib inline

conf = SparkConf().setMaster('local').setAppName('ML_learning')
x1 = data.toPandas()['Ds_avg'].values.tolist()
y1 = data.toPandas()['S_avg'].values.tolist()

plt.scatter(x1, y1, color='Blue', s=30)
plt.xlabel('Generator Speed - Ds_avg')
plt.ylabel('Apparent Power - S_avg')
plt.title('Linear Regression - PySpark')
p1 = polyfit(x1, y1 , 1)
plt.plot(x1, np.polyval(p1,x1), 'g-' )
plt.show()
```



```
# Ds_avg vs Ws_avg

x1 = data.toPandas()['Ds_avg'].values.tolist()
y1 = data.toPandas()['Ws_avg'].values.tolist()

plt.scatter(x1, y1, color='Blue', s=30)
plt.xlabel('Generator Speed - Ds_avg')
plt.ylabel('Wind speed - Ws_avg')
plt.title('Linear Regression - PySpark')
p1 = polyfit(x1, y1 , 1)
plt.plot(x1, np.polyval(p1,x1), 'g-' )
plt.show()
```



```
# Ds_avg vs Ot_avg

x1 = data.toPandas()['Ds_avg'].values.tolist()
y1 = data.toPandas()['Ot_avg'].values.tolist()

plt.scatter(x1, y1, color='Blue', s=30)
plt.xlabel('Generator Speed - Ds_avg')
plt.ylabel('Outdoor Temperature - Ot_avg')
plt.title('Linear Regression - PySpark')
p1 = polyfit(x1, y1 , 1)
plt.plot(x1, np.polyval(p1,x1), 'g-' )
plt.show()
```