PySpark Hands-On

Mai H. Nguyen, Ph.D.

Overview

- **Weather station measurements**
- **Data Exploration**
  - Load into Spark DataFrame
  - Describe schema
  - Show summary statistics
  - Calculate correlation between features
- **Cluster to identify different weather patterns**
  - Spark k-means
  - Parallel plots

Get Latest from Github Repo

- **Clone/Update repo on comet**
- **If haven’t cloned Summer Institute repo**
  - git clone <URL>
- **If already cloned Summer Institute repo**
  - git pull <URL>
- **<URL>**

https://github.com/sdsc/sdsc-summer-institute-2020

Server Setup

- **Set up server**
  - In terminal window: start_python_sparkr_cpu
  - Should get something like this:

```
Your notebook is here:
          https://unkind-illicitly-mutt.comet-user-
content.sdsc.edu?token=6615bbdb1a8e0fbe3ad948fb52678133
Submitted batch job 35032027
```

- **Connect to jupyter notebook**
  - In browser, paste URL of notebook from above step
- **Check queue**
  - *squeue –u $USER*

Data Setup

- **In terminal window, do the following:**

- **Link to data files**

```
cd <SI2020_dir>/datasci3_scalable_machine_learning/pyspark
ln –s ~/ML-data/*_weather.csv .
cd ../sparkR
ln –s ~/ML-data/wine*.csv .
```

- **Go to spark directory**

```
cd <SI2020_dir>/datasci3_scalable_machine_learning/pyspark
```

Dataset Description

- **Measurements from weather station on Mt. Woodson, San Diego**
- **Air temperature, humidity, wind speed, wind direction, etc.**
- **Three years of data: Sep. 2011 - Sep. 2014**
- ***minute_weather.csv***
  - measurement every minute
- ***daily_weather.csv***
  - aggregated measurements

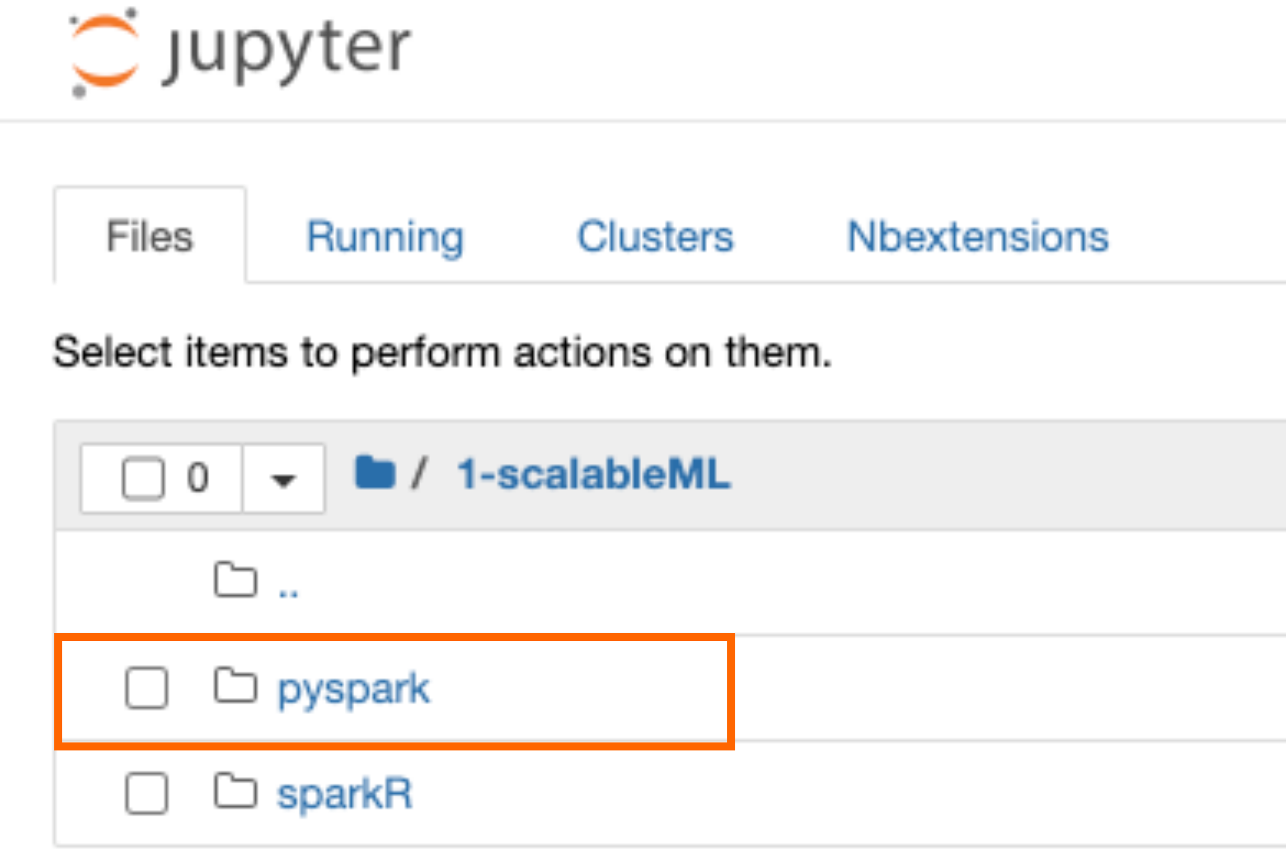
Dataset Description

- **Measurements from weather station on Mt. Woodson, San Diego**
- **Air temperature, humidity, wind speed, wind direction, etc.**
- **Three years of data: Sep. 2011 - Sep. 2014**
- ***minute_weather.csv***
  - measurement every minute
- ***daily_weather.csv***
  - aggregated measurements

Clustering

Data Exploration

Go to sparkR folder



Select items to perform actions on them.

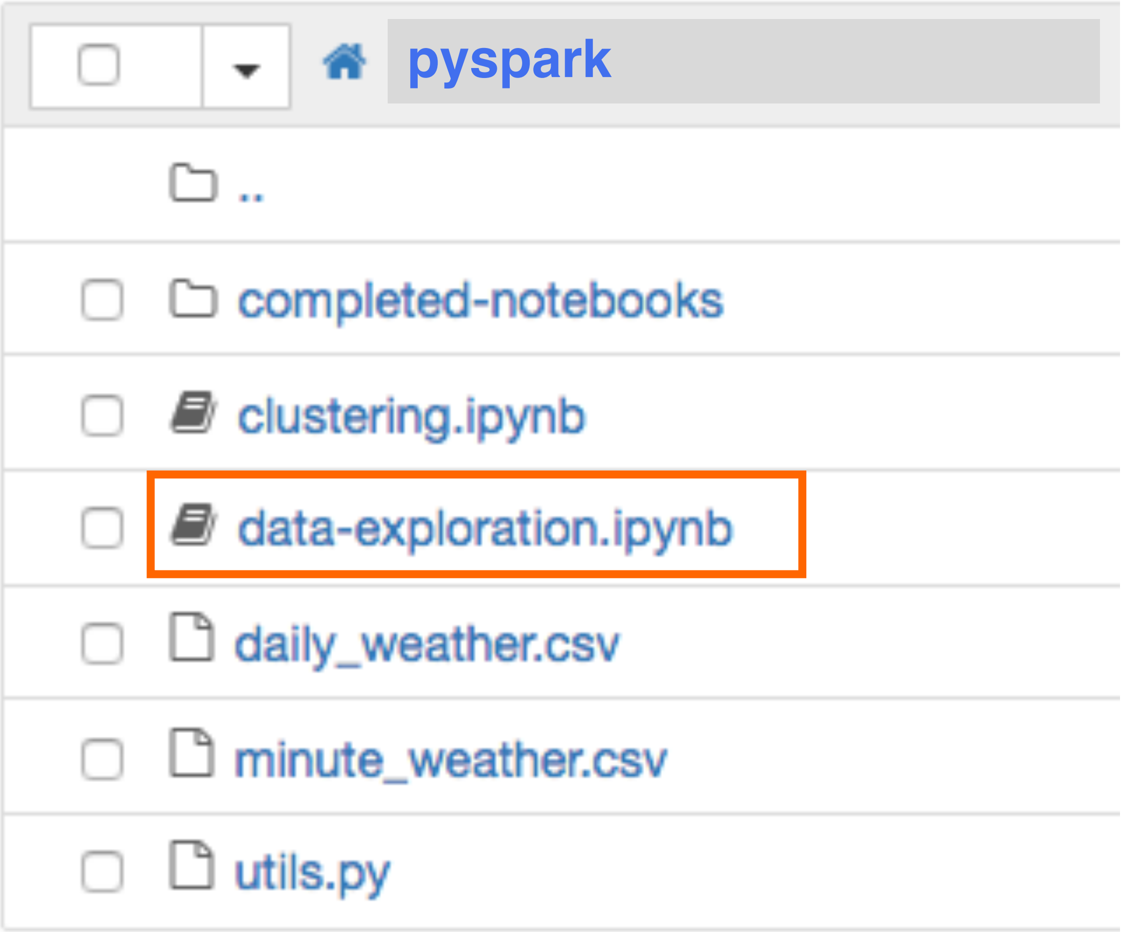
☐ 0 ▾ / 1-scalableML

..

☐ pyspark

☐ sparkR

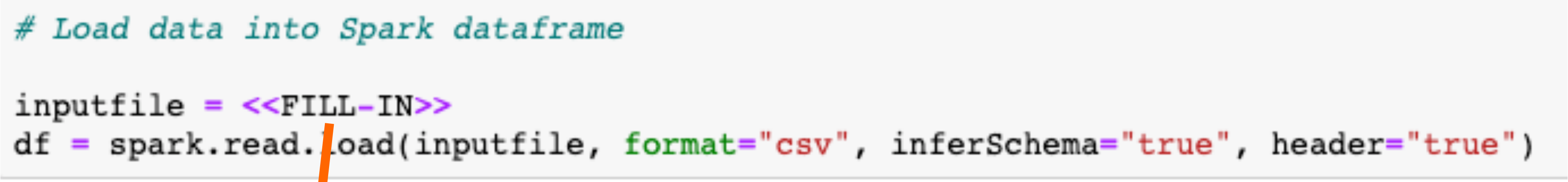
Open Data Exploration Notebook



- ..
- □ completed-notebooks
- □ clustering.ipynb
- □ data-exploration.ipynb
- □ daily_weather.csv
- □ minute_weather.csv
- □ utils.py

Load Data into Spark DataFrame

- Start the Spark session
- Read the daily weather data into a Spark DataFrame

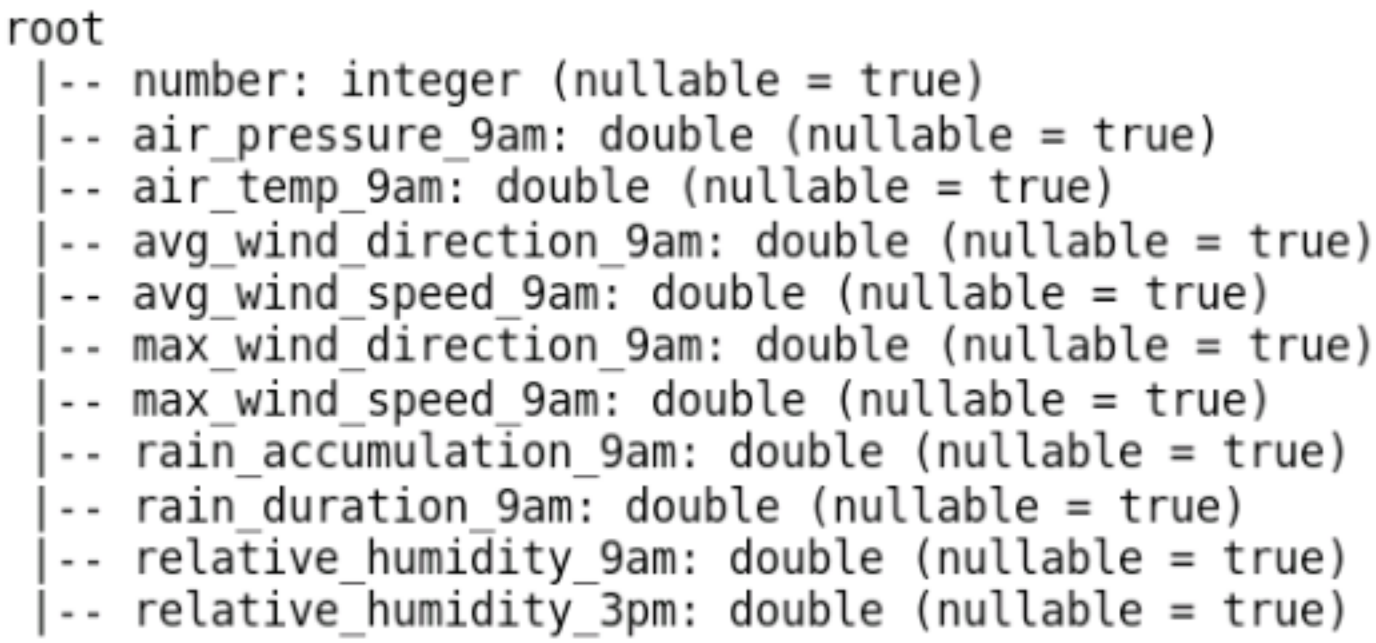


```
# Load data into Spark dataframe

inputfile = <<FILL-IN>>
df = spark.read.load(inputfile, format="csv", inferSchema="true", header="true")
```

- Replace with data filename (with quotes):
  'daily_weather.csv'

Examine Schema

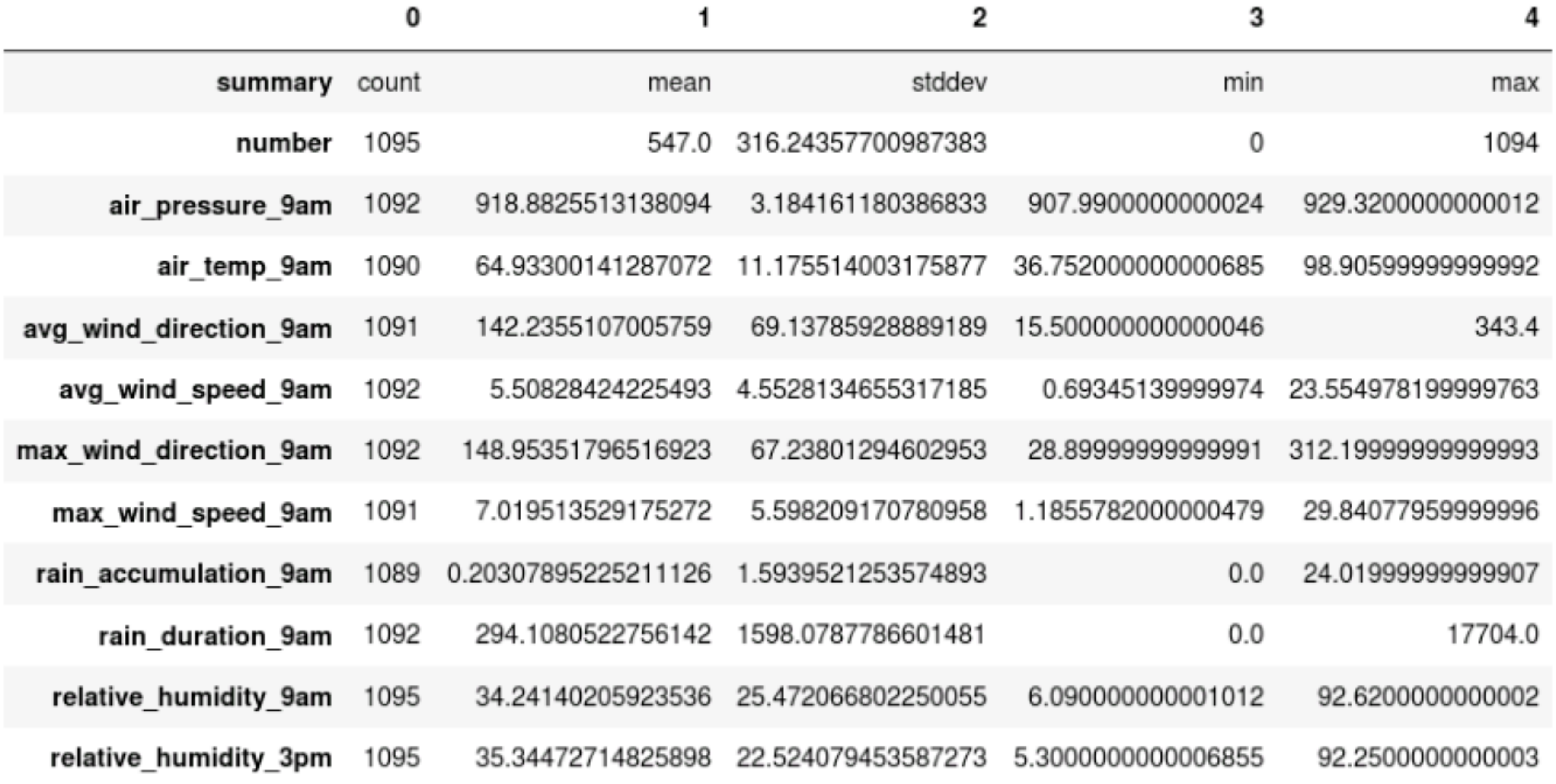
df.printSchema()



```
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)
```

Show Summary Statistics

df.describe().toPandas().transpose()



	0	1	2	3	4
summary	count	mean	stddev	min	max
number	1095	547.0	316.24357700987383	0	1094
air_pressure_9am	1092	918.8825513138094	3.184161180386833	907.9900000000024	929.3200000000012
air_temp_9am	1090	64.93300141287072	11.175514003175877	36.752000000000685	98.90599999999992
avg_wind_direction_9am	1091	142.2355107005759	69.13785928889189	15.500000000000046	343.4
avg_wind_speed_9am	1092	5.50828424225493	4.5528134655317185	0.69345139999974	23.554978199999763
max_wind_direction_9am	1092	148.95351796516923	67.23801294602953	28.89999999999991	312.19999999999993
max_wind_speed_9am	1091	7.019513529175272	5.598209170780958	1.1855782000000479	29.84077959999996
rain_accumulation_9am	1089	0.20307895225211126	1.5939521253574893	0.0	24.01999999999907
rain_duration_9am	1092	294.1080522756142	1598.0787786601481	0.0	17704.0
relative_humidity_9am	1095	34.24140205923536	25.472066802250055	6.090000000001012	92.6200000000002
relative_humidity_3pm	1095	35.34472714825898	22.524079453587273	5.3000000000006855	92.2500000000003

Number of Rows

```
df.count()
```

```
1095
```

First Two Rows

df.show(2)

```
+------+-----------------+-----------------+----------------------+------------------+----------------------+------------
------+---------------------+-----------------+---------------------+---------------------+
|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_hu
midity_9am|relative_humidity_3pm|
+------+-----------------+-----------------+----------------------+------------------+----------------------+------------
------+---------------------+-----------------+---------------------+---------------------+
|     0|918.0600000000087|74.82200000000041|                 271.1|
2.080354199999768|    295.39999999999986|
2.863283199999908|                  0.0|              0.0|    42.420000000000046|   36.16000000000049
4|
|     1|917.3476881177097|71.40384263106537|    101.93517935618371|2.443092157340217|
  140.47154847112498|3.533236016106238|                  0.0|              0.0|   24.3286972918022
07|     19.4265967985621|
+------+-----------------+-----------------+----------------------+------------------+----------------------+------------
------+---------------------+-----------------+---------------------+---------------------+
only showing top 2 rows
```

Number and Names of Columns

```
df.columns
```

```
['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’]
```

```
len(df.columns)
```

```
11
```

Correlation Between Air Temperature and Relative Humidity

```
df.stat.corr(“air_temp_9am”, “relative_humidity_9am”)
```

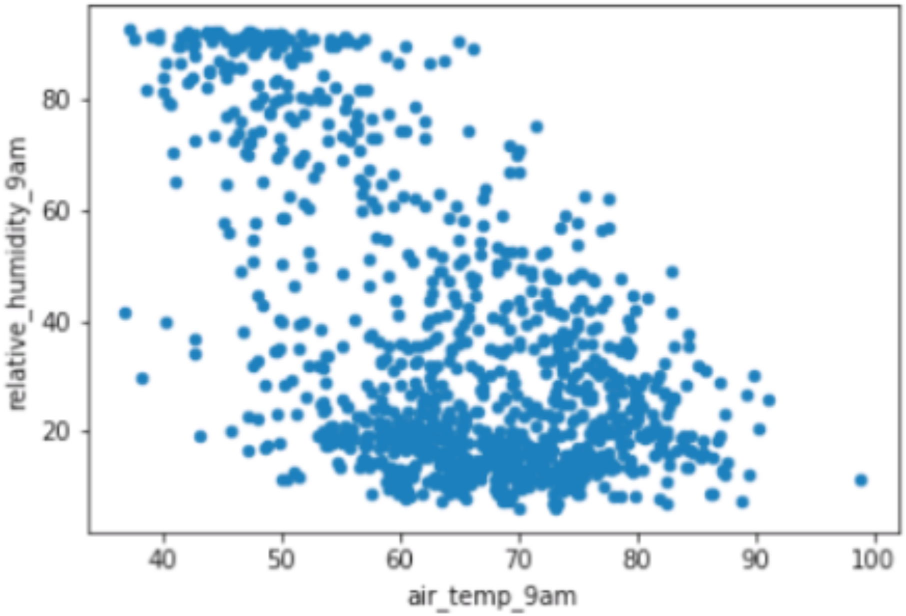
```
-0.536670…
```

Show Plots in Notebook

```
%matplotlib inline
```

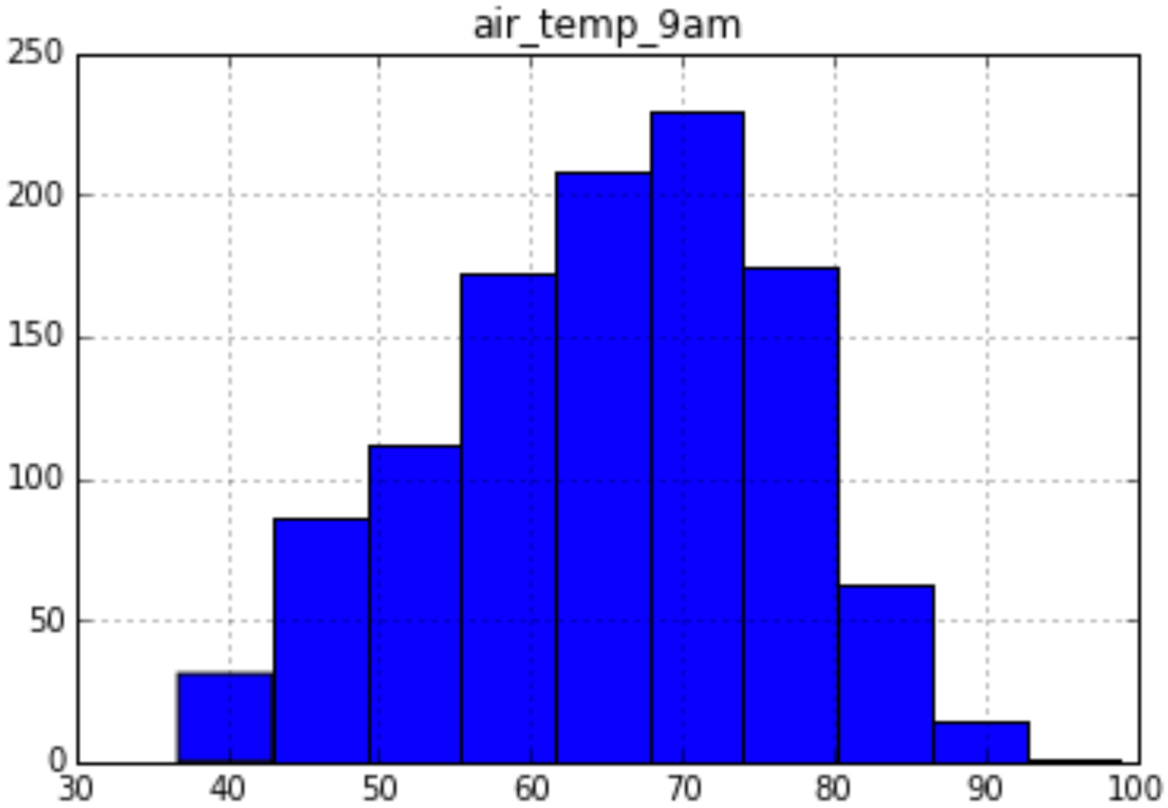
Scatter Plot of Air Temperature vs Humidity

```
df.select(‘air_temp_9am’, ‘relative_humidity_9am’)
.toPandas()
.plot.scatter (x=‘air_temp_9am’,
               y=‘relative_humidity_9am’)
```



Histogram of Air Temperature

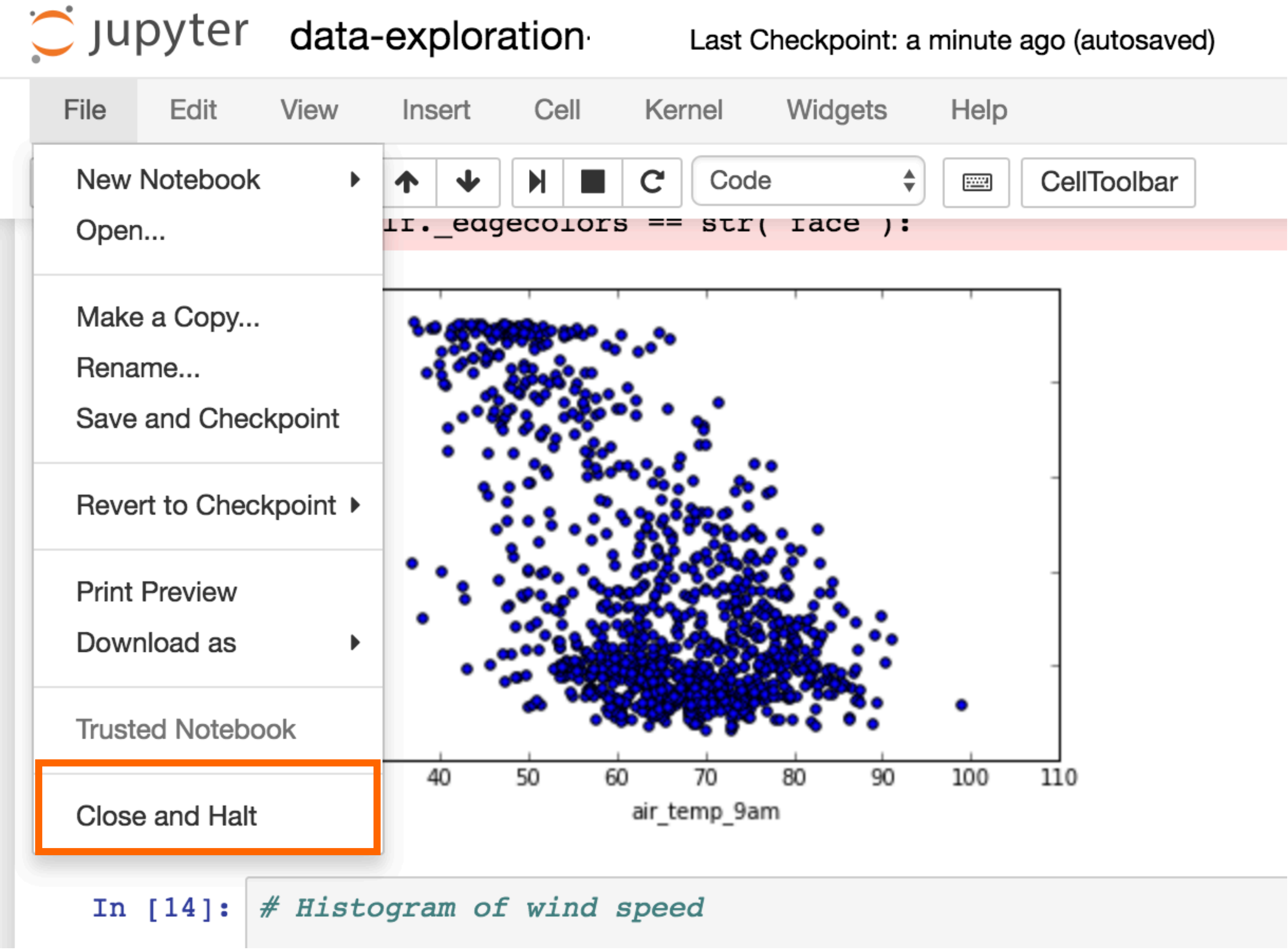
df.select('air_temp_9am').toPandas().hist()



Stop Spark Session

spark.stop()

Save and Exit Notebook



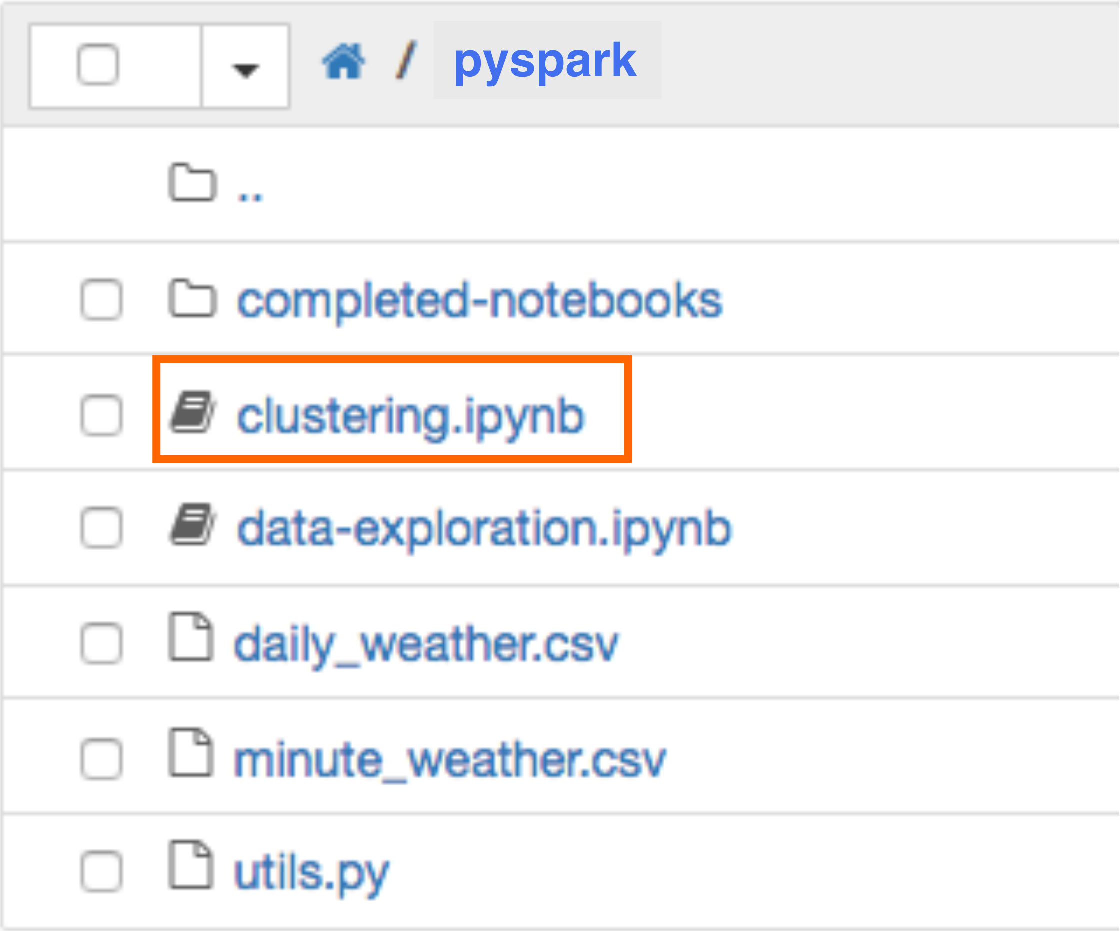
```
if._edgecolors == str( face ):
```

```
In [14]: # Histogram of wind speed
```

Clustering to Identify Santa Ana Conditions

- **Strong, dry winds in Southern California**
  - wind speed > 30mph
  - wind direction between 10 & 110 degrees (from east)
  - relative humidity < 10%
- **Extreme fire danger**
  - May 2014, swarm of 14 wildfires in San Diego County
  - 2008, Witch Fire, ~200,000 acres
  - 2003, Cedar Fire, ~280,000 acres

Open Clustering Notebook



/ pyspark

..

completed-notebooks

clustering.ipynb

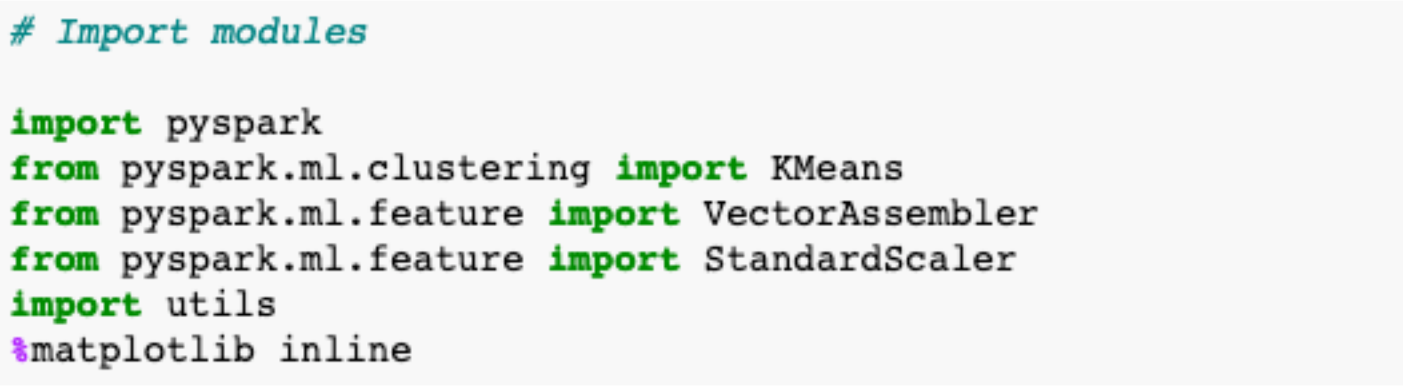
data-exploration.ipynb

daily_weather.csv

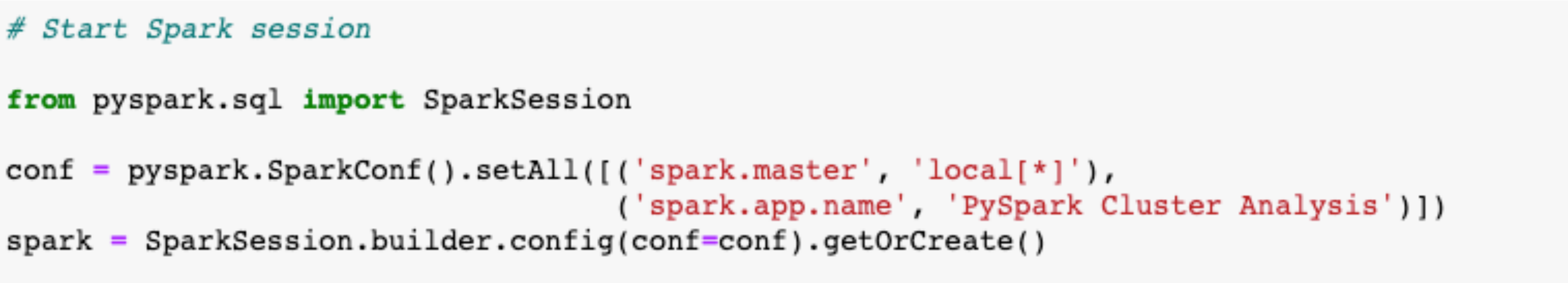
minute_weather.csv

utils.py

Import Modules & Start Spark Session

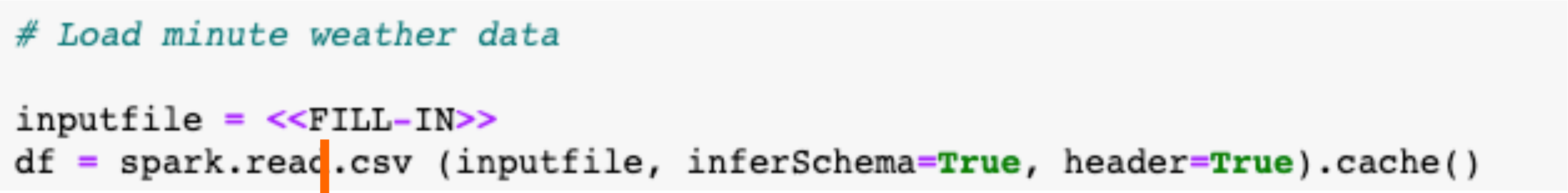


```
# Import modules

import pyspark
from pyspark.ml.clustering import KMeans
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.feature import StandardScaler
import utils
%matplotlib inline
```



```
# Start Spark session

from pyspark.sql import SparkSession

conf = pyspark.SparkConf().setAll([('spark.master', 'local[*]'),
                                   ('spark.app.name', 'PySpark Cluster Analysis')])
spark = SparkSession.builder.config(conf=conf).getOrCreate()
```

Load Modules & Minute Weather Data

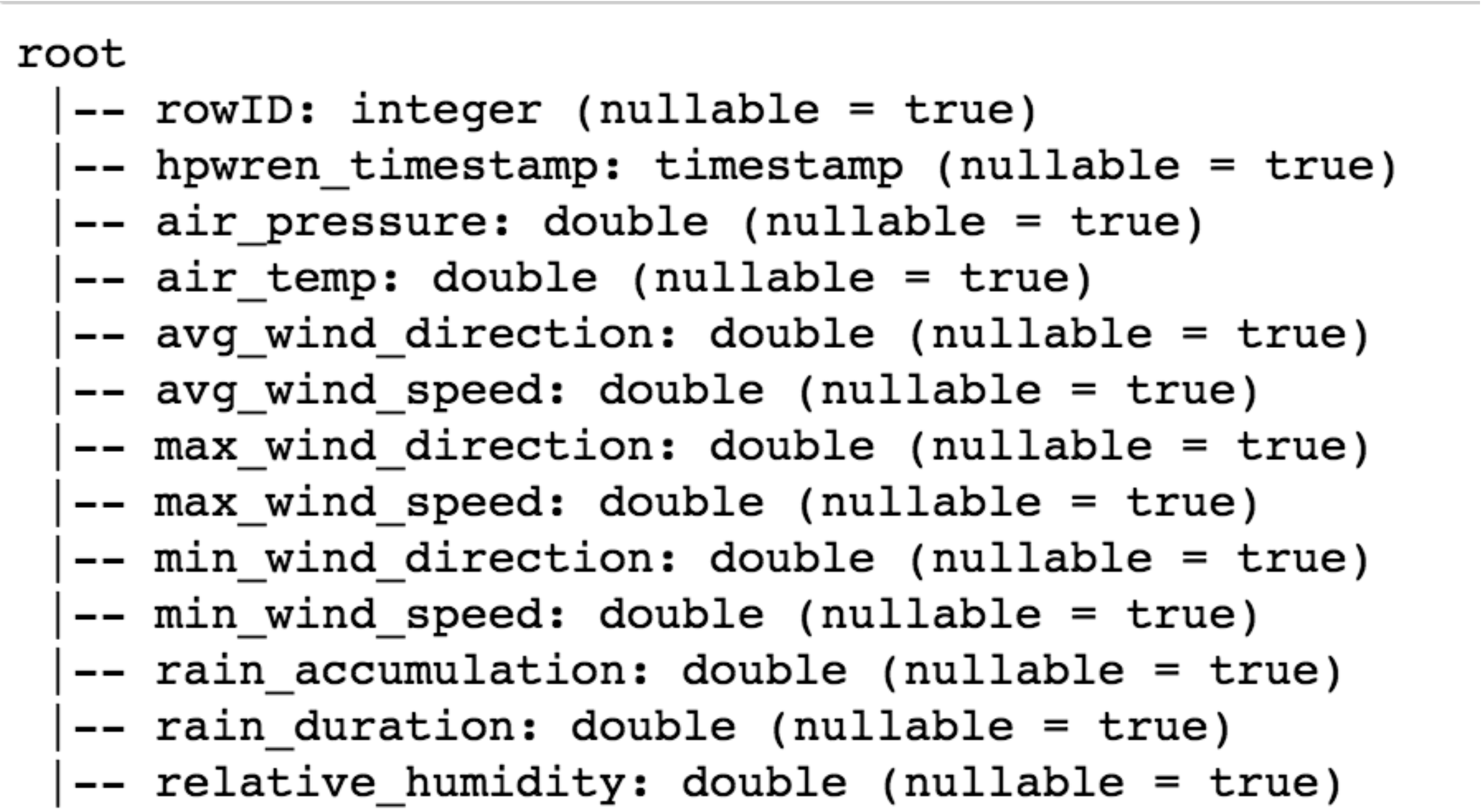


```
# Load minute weather data

inputfile = <<FILL-IN>>
df = spark.read.csv (inputfile, inferSchema=True, header=True).cache()
```

- Replace with data filename (with quotes):
  - ‘minute_weather.csv’

Examine Schema

df.printSchema()



```
root
 |-- rowID: integer (nullable = true)
 |-- hpwren_timestamp: timestamp (nullable = true)
 |-- air_pressure: double (nullable = true)
 |-- air_temp: double (nullable = true)
 |-- avg_wind_direction: double (nullable = true)
 |-- avg_wind_speed: double (nullable = true)
 |-- max_wind_direction: double (nullable = true)
 |-- max_wind_speed: double (nullable = true)
 |-- min_wind_direction: double (nullable = true)
 |-- min_wind_speed: double (nullable = true)
 |-- rain_accumulation: double (nullable = true)
 |-- rain_duration: double (nullable = true)
 |-- relative_humidity: double (nullable = true)
```

Count Rows and Filter Data

- **Count rows**

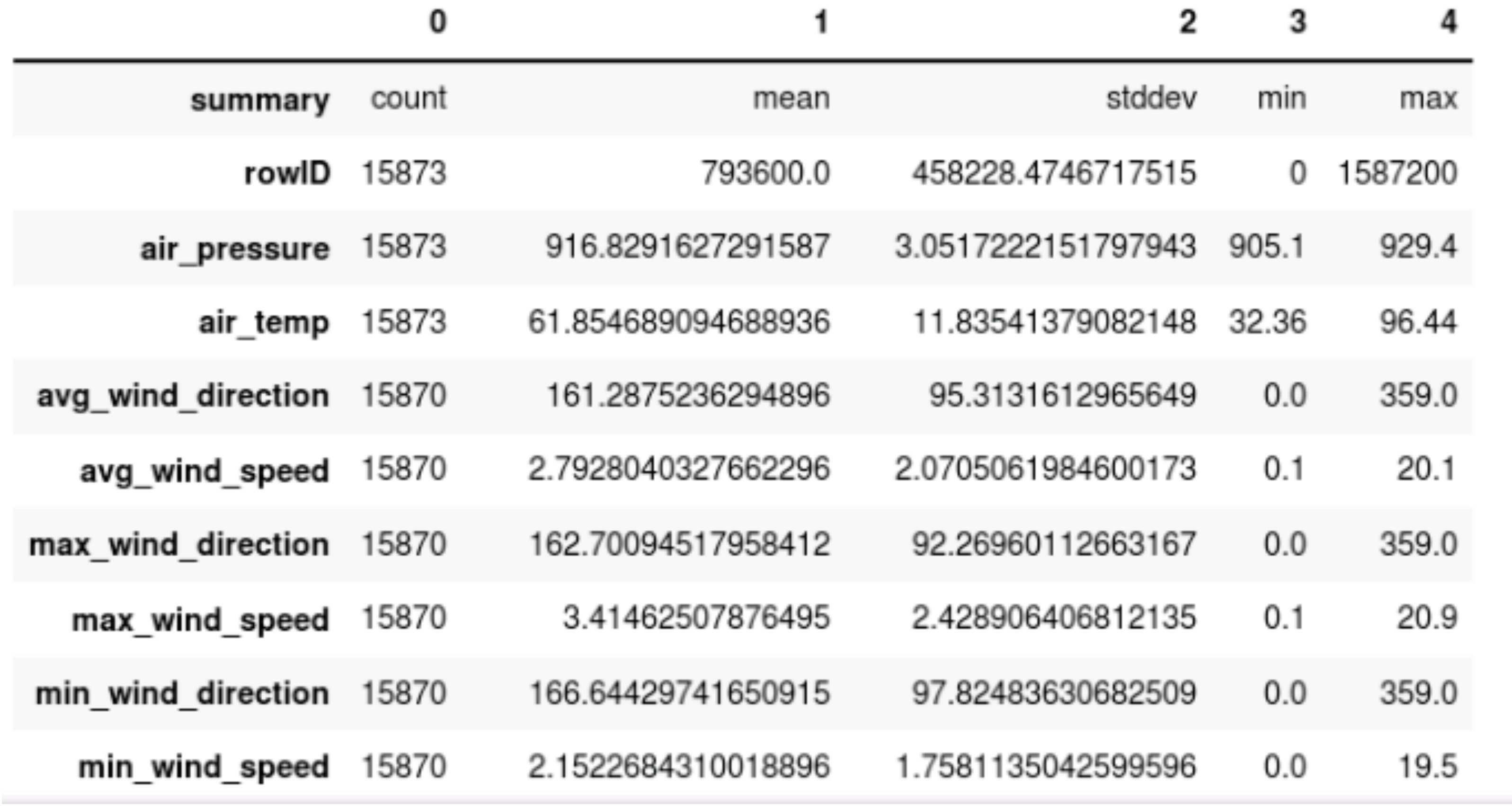
```
df.count()
= 1587257
```

Filter data

```
filteredDF = df.filter((df.rowID % 100) == 0)
filteredDF.count()
= 15873
```

Show Summary Statistics

filteredDF.describe().toPandas().transpose()



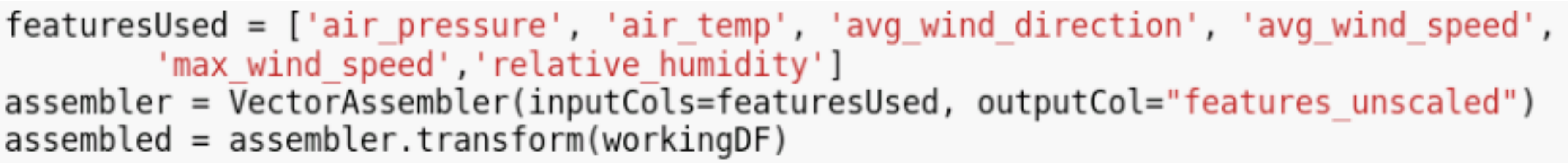
	0	1	2	3	4
summary	count	mean	stddev	min	max
rowID	15873	793600.0	458228.4746717515	0	1587200
air_pressure	15873	916.8291627291587	3.0517222151797943	905.1	929.4
air_temp	15873	61.854689094688936	11.83541379082148	32.36	96.44
avg_wind_direction	15870	161.2875236294896	95.3131612965649	0.0	359.0
avg_wind_speed	15870	2.7928040327662296	2.0705061984600173	0.1	20.1
max_wind_direction	15870	162.70094517958412	92.26960112663167	0.0	359.0
max_wind_speed	15870	3.41462507876495	2.428906406812135	0.1	20.9
min_wind_direction	15870	166.64429741650915	97.82483630682509	0.0	359.0
min_wind_speed	15870	2.1522684310018896	1.7581135042599596	0.0	19.5

Drop Samples with Null Values

```
workingDF = filteredDF.na.drop()
workingDF.count()
```

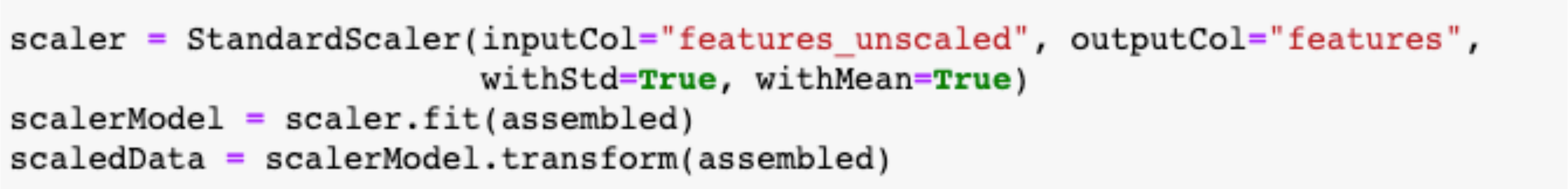
= 15869

Create Feature Vector



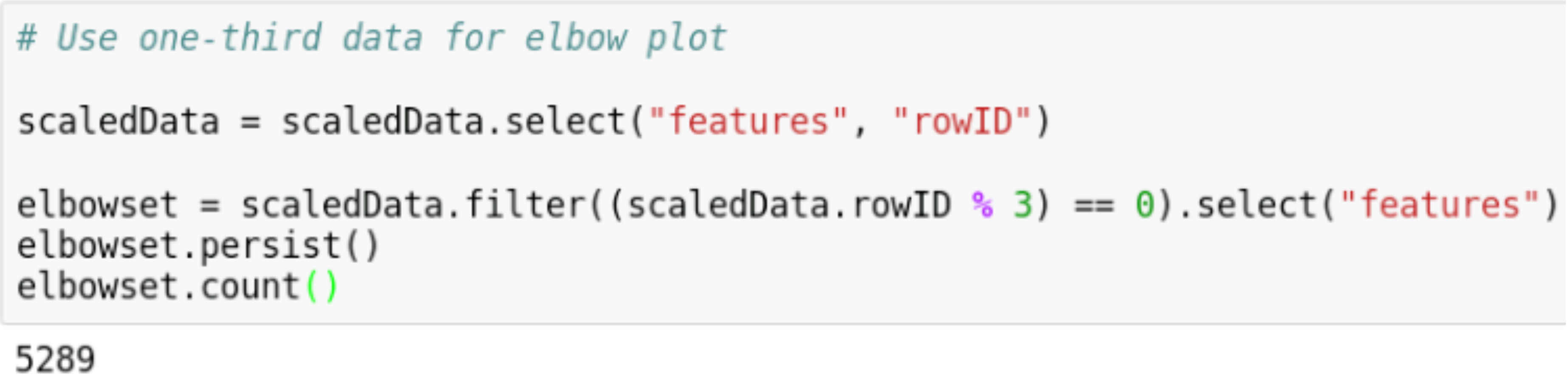
```
featuresUsed = ['air_pressure', 'air_temp', 'avg_wind_direction', 'avg_wind_speed',
        'max_wind_speed','relative_humidity']
assembler = VectorAssembler(inputCols=featuresUsed, outputCol="features_unscaled")
assembled = assembler.transform(workingDF)
```

Scale Data



```
scaler = StandardScaler(inputCol="features_unscaled", outputCol="features",
                        withStd=True, withMean=True)
scalerModel = scaler.fit(assembled)
scaledData = scalerModel.transform(assembled)
```

Generate Elbow Plot



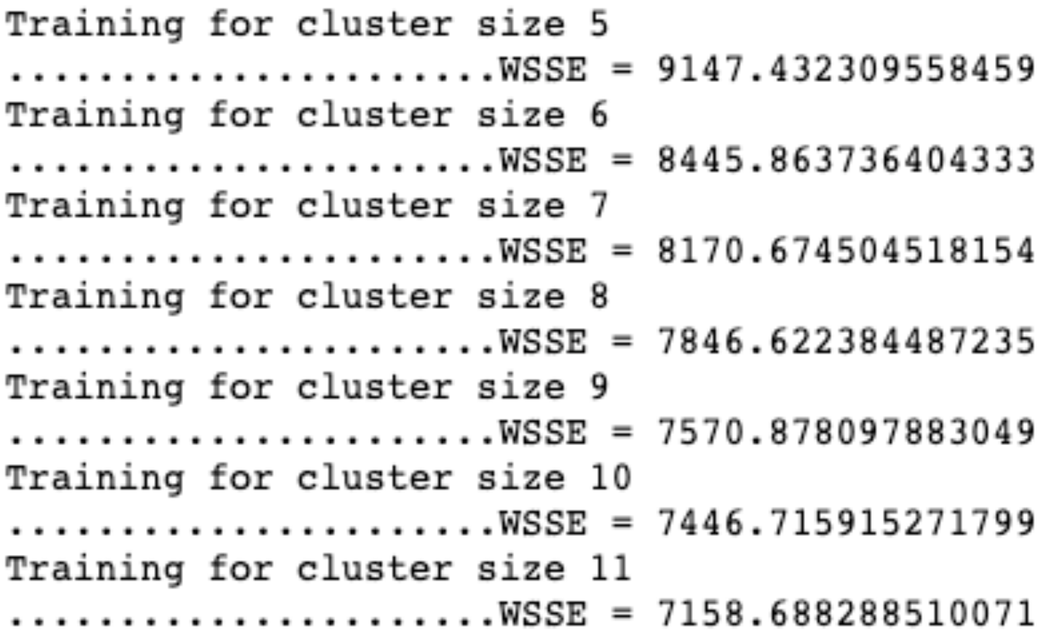
```
# Use one-third data for elbow plot

scaledData = scaledData.select("features", "rowID")

elbowset = scaledData.filter((scaledData.rowID % 3) == 0).select("features")
elbowset.persist()
elbowset.count()
```

```
5289
```

Generate Clusters for Elbow Plot

clusters = range(5, 15)

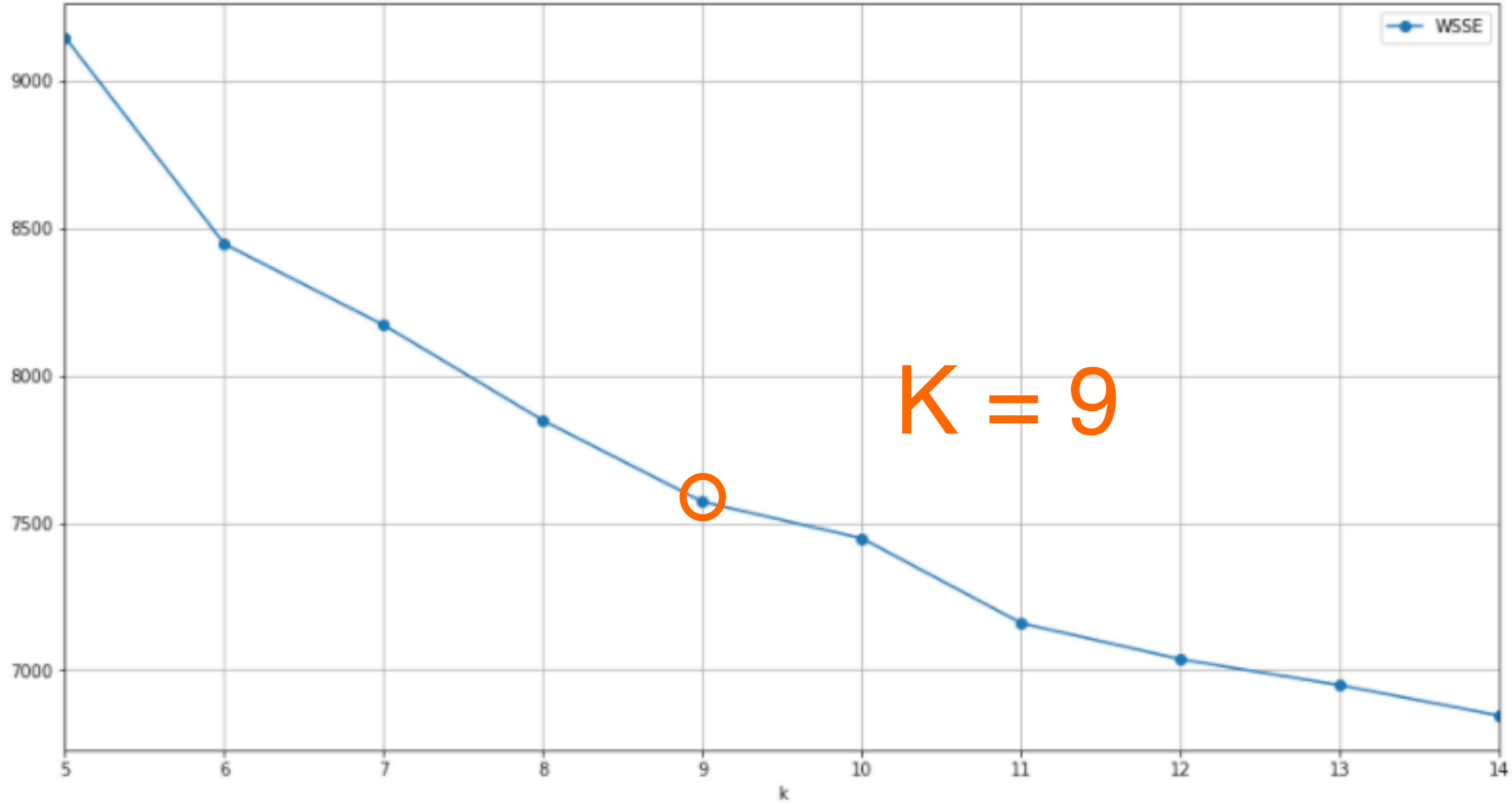
wsseList = utils.elbow(elbowset, clusters)



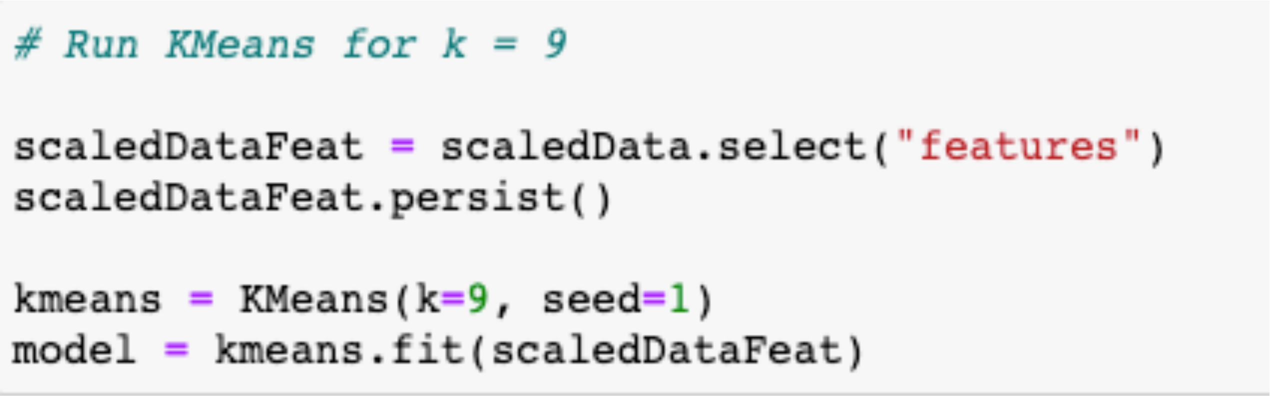
```
Training for cluster size 5
......................WSSE = 9147.432309558459
Training for cluster size 6
......................WSSE = 8445.863736404333
Training for cluster size 7
......................WSSE = 8170.674504518154
Training for cluster size 8
......................WSSE = 7846.622384487235
Training for cluster size 9
......................WSSE = 7570.878097883049
Training for cluster size 10
......................WSSE = 7446.715915271799
Training for cluster size 11
......................WSSE = 7158.688288510071
```

Show Elbow Plot

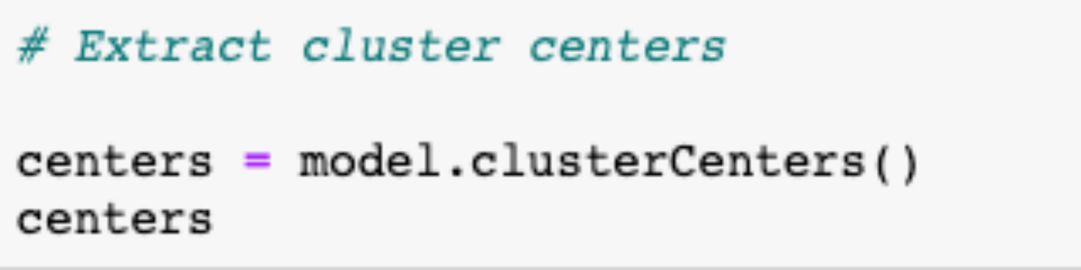
utils.elbow_plot(wsseList, clusters)



Run KMeans for k = 9 and Extract Cluster Centers



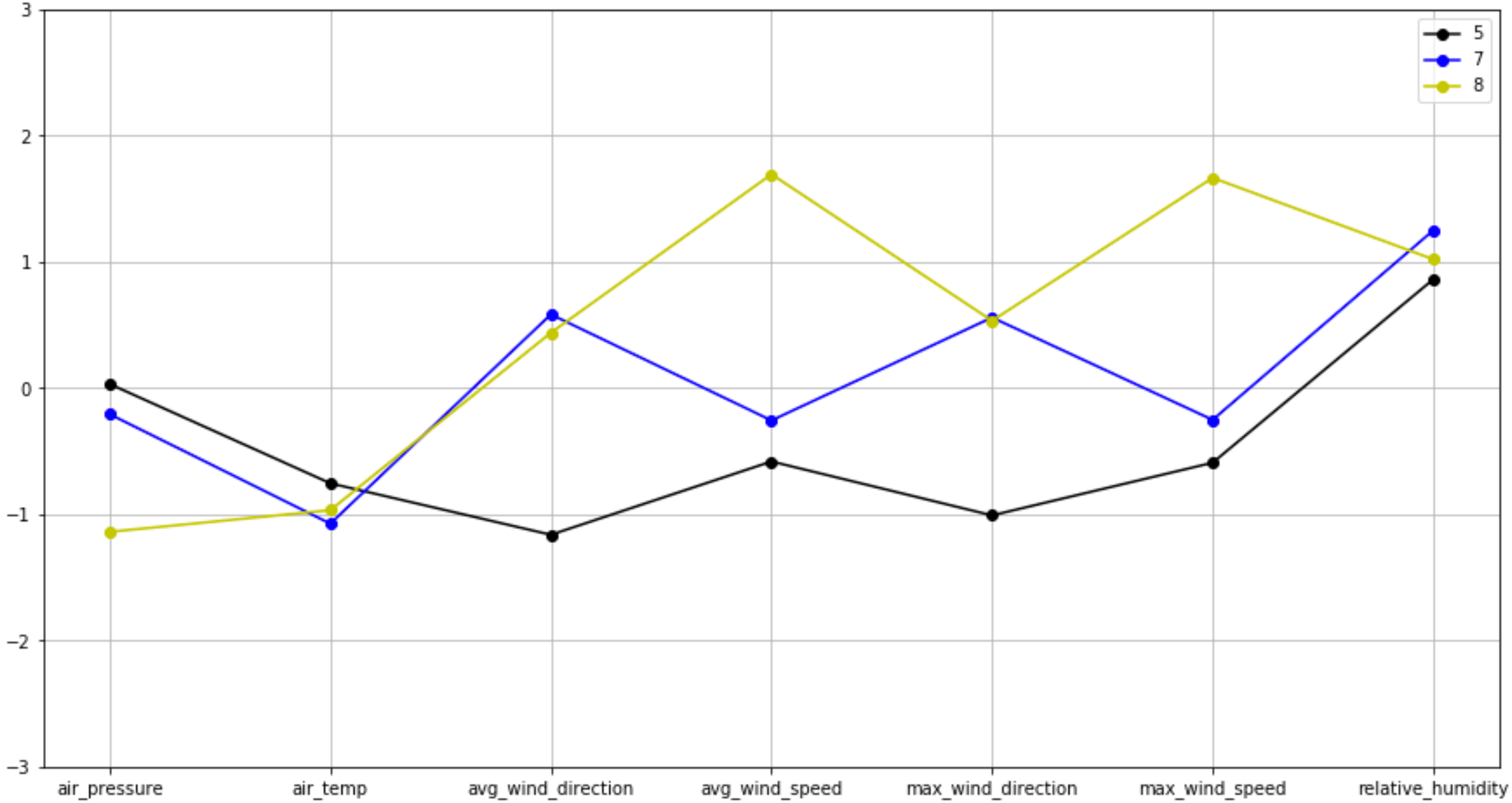
```
# Run KMeans for k = 9

scaledDataFeat = scaledData.select("features")
scaledDataFeat.persist()

kmeans = KMeans(k=9, seed=1)
model = kmeans.fit(scaledDataFeat)
```



```
# Extract cluster centers

centers = model.clusterCenters()
centers
```

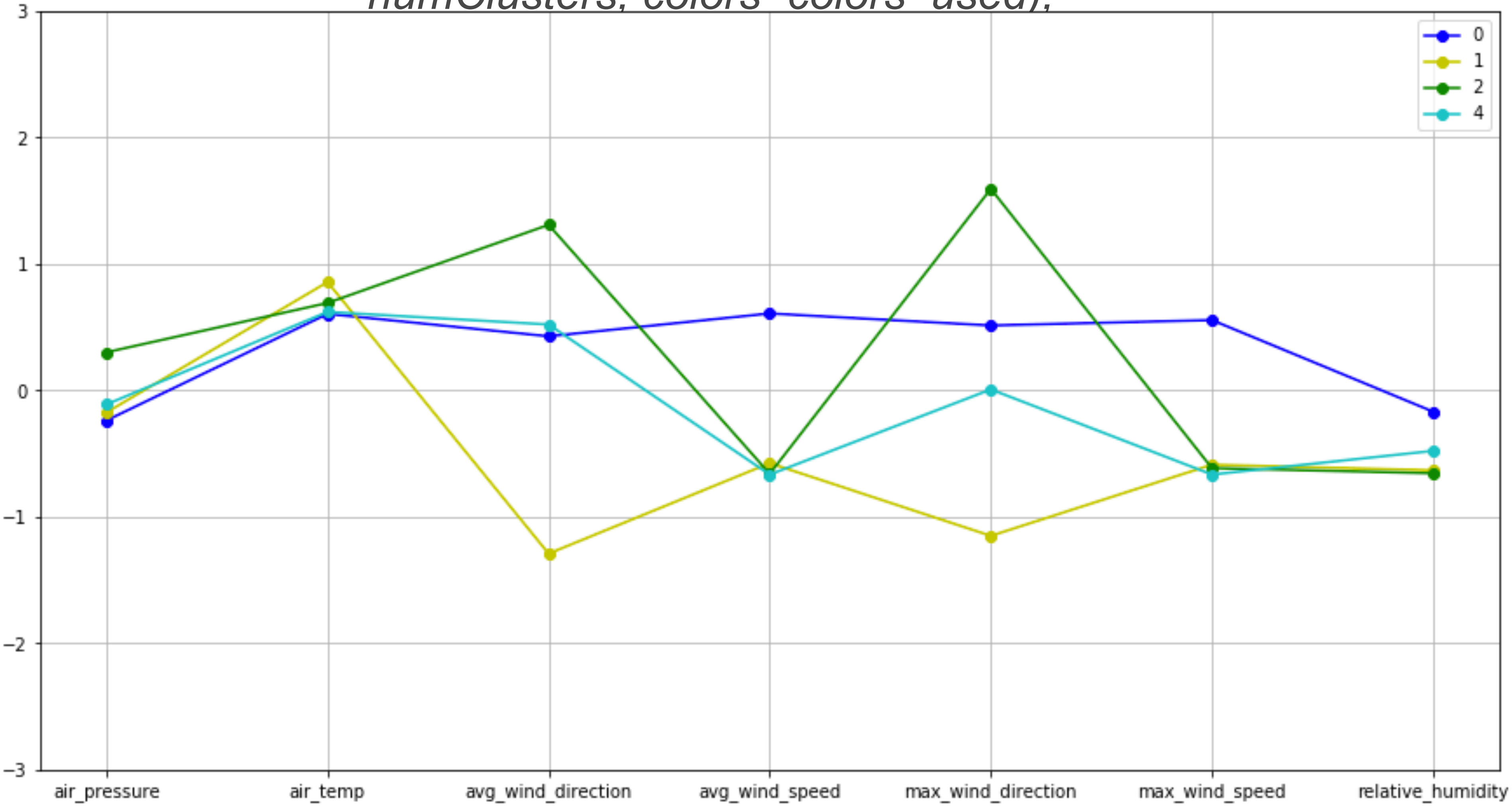
Cluster Capturing Humid Days

utils.parallel_plot(centersNamed[centersNamed['relative_humidity’] > 0.5], numClusters, colors=colors_used);



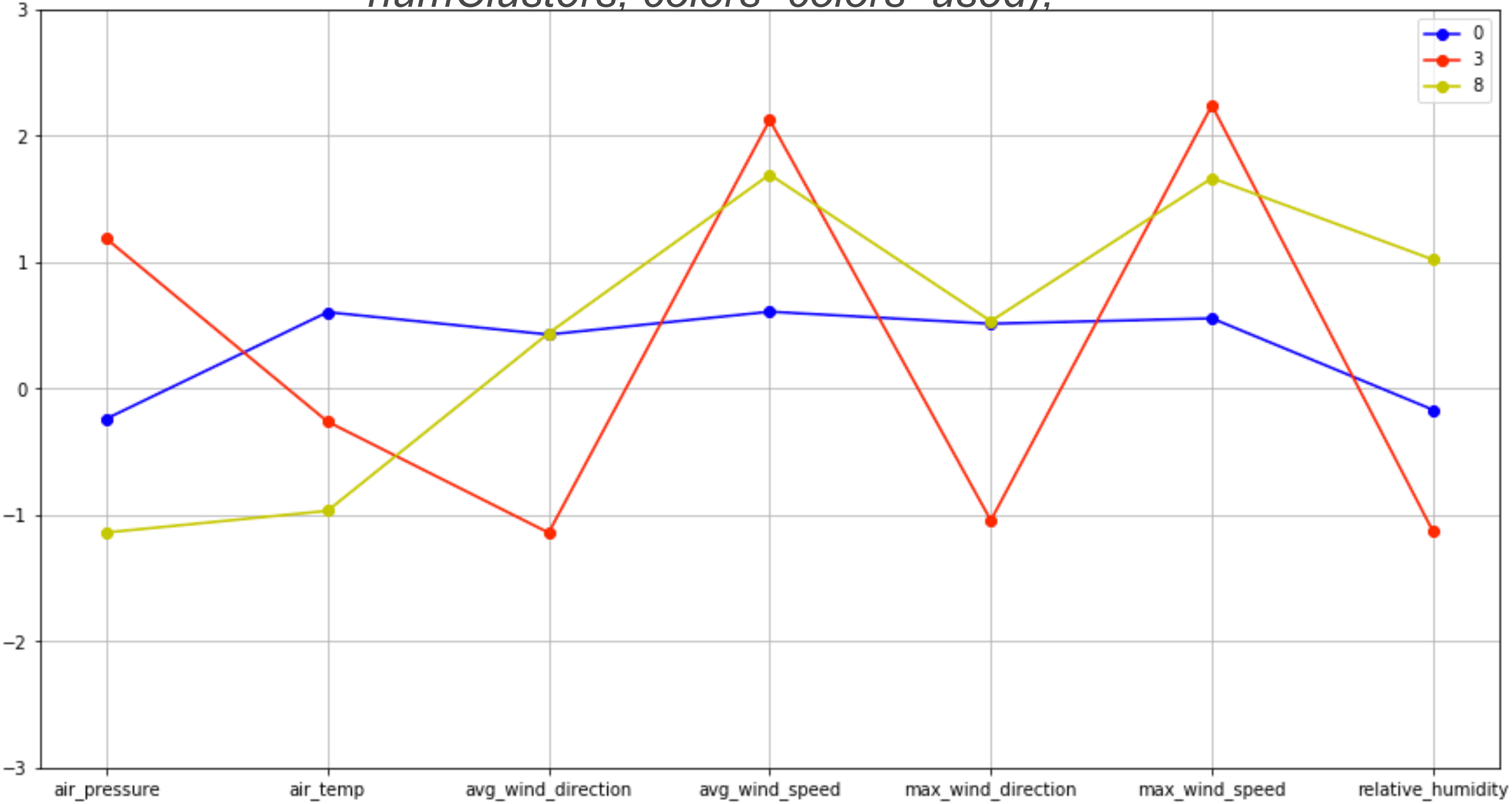
Cluster Capturing Hot Days

utils.parallel_plot(centersNamed[centersNamed[air_temp’] > 0.5], numClusters, colors=colors_used);



Cluster Capturing Windy Days

```
utils.parallel_plot(centersNamed[centersNamed[avg_wind_speed’] > 0.5],
numClusters, colors=colors_used);
```



Stop Spark Session

```
spark.stop()
```

Clean Up

- **Exit notebook**
  - File -> Close and Halt
- **Exit Jupyter Notebook**
  - Click on ‘Logout’

References

- **Spark**
  - https://spark.apache.org/
- **MLlib**
  - https://spark.apache.org/mllib/

Questions?