Lab_9 Sravan

```
import org.apache.spark.sql.functions._
import org.joda.time.format.DateTimeFormat
import org.apache.commons.io.IOUtils
import java.net.URL
import java.nio.charset.Charset
```

FINISHED

```
import org.apache.spark.sql.functions._
import org.joda.time.format.DateTimeFormat
import org.apache.commons.io.IOUtils
import java.net.URL
import java.nio.charset.Charset
```

Took 2 sec. Last updated by anonymous at March 30 2017, 7:34:48 PM.

```
%pyspark
from pandas import Series, DataFrame
import pandas as pd
import numpy as np
```

FINISHED

Took 0 sec. Last updated by anonymous at March 30 2017, 7:35:03 PM.

```
%pyspark
start = timeit.timeit()

inputPath =  "/Users/sborra/Downloads/raw_weather_data_aarhus"

dewpoint = pd.read_csv(inputPath+"/dewptm.csv")

humidity = pd.read_csv(inputPath+"/hum.csv")

pressure = pd.read_csv(inputPath+"/pressurem.csv")


temp = pd.read_csv(inputPath+"/tempm.csv")

winddirection = pd.read_csv(inputPath+"/wdird.csv")
end = timeit.timeit()
print("Time taken", end - start)
```

FINISHED

```
('Time taken', -0.0020341873168945312)
```

Took 0 sec. Last updated by anonymous at March 30 2017, 8:32:07 PM.

```
%pyspark
weather = pd.concat([dewpoint,humidity.ix[:,1],pressure.ix[:,1],temp.ix[:,1],winddirection.ix|
```

FINISHED

Took 0 sec. Last updated by anonymous at March 30 2017, 7:37:12 PM.

```
%pyspark
```

FINISHED

```
Dewpoint          34708.0
Humidity         583993.0
Pressure        8314873.0
Temperature       74365.0
Winddirection   1486090.0
dtype: float64
```

Took 0 sec. Last updated by anonymous at March 30 2017, 7:37:13 PM.

```
%pyspark
weather.ix[:,1:].sum(axis=1)
```

FINISHED

```
0       1225.0
1       1233.0
2       1232.0
3       1223.0
4       1242.0
5       1242.0
6       1229.0
7       1243.0
8       1242.0
9       1230.0
10      1242.0
11      1256.0
12      1241.0
13      1256.0
14      1256.0
15      1252.0
16      1256.0
17      1255.0
```

Took 0 sec. Last updated by anonymous at March 30 2017, 7:37:35 PM.

```
%pyspark
weather.ix[:,1:].mean(axis=1,skipna=False)
```

FINISHED

```
0       245.0
1       246.6
2       246.4
3       244.6
4       248.4
5       248.4
6       245.8
7       248.6
8       248.4
9       246.0
10      248.4
11      251.2
12      248.2
13      251.2
14      251.2
15      250.4
16      251.2
17      251.0
```

Took 0 sec. Last updated by anonymous at March 30 2017, 7:38:03 PM.

```
%pyspark
weather.isnull().any()
```

FINISHED

```
DateTime         False
Dewpoint          True
Humidity          True
Pressure          True
Temperature       True
Winddirection     True
dtype: bool
```

Took 0 sec. Last updated by anonymous at March 30 2017, 7:38:32 PM.

```
%pyspark
#In this step, we will try to update null values.
#filling null values could be complicated.As we seen in previous data exploration steps
#that 116 was the maximum null values and total datasize is 12563. Since, maximum percent of n
#So, null values will be replaced by mean of the particular parameter.
def updatenullvalues(dataset):
    for col in dataset.ix[:,1:]:
        if dataset[col].isnull().any:
            mean = dataset[col].mean()
            dataset[col].fillna(mean,inplace=True)
    return dataset
```

FINISHED

Took 0 sec. Last updated by anonymous at March 30 2017, 7:39:11 PM.

```
%pyspark
start = timeit.timeit()
#Let's update null values in our dataset.
weather = updatenullvalues(weather_dataset)
#verify is there still any null value left in the dataset
weather.isnull().sum()
end = timeit.timeit()
print("Time taken", end - start)
```

FINISHED

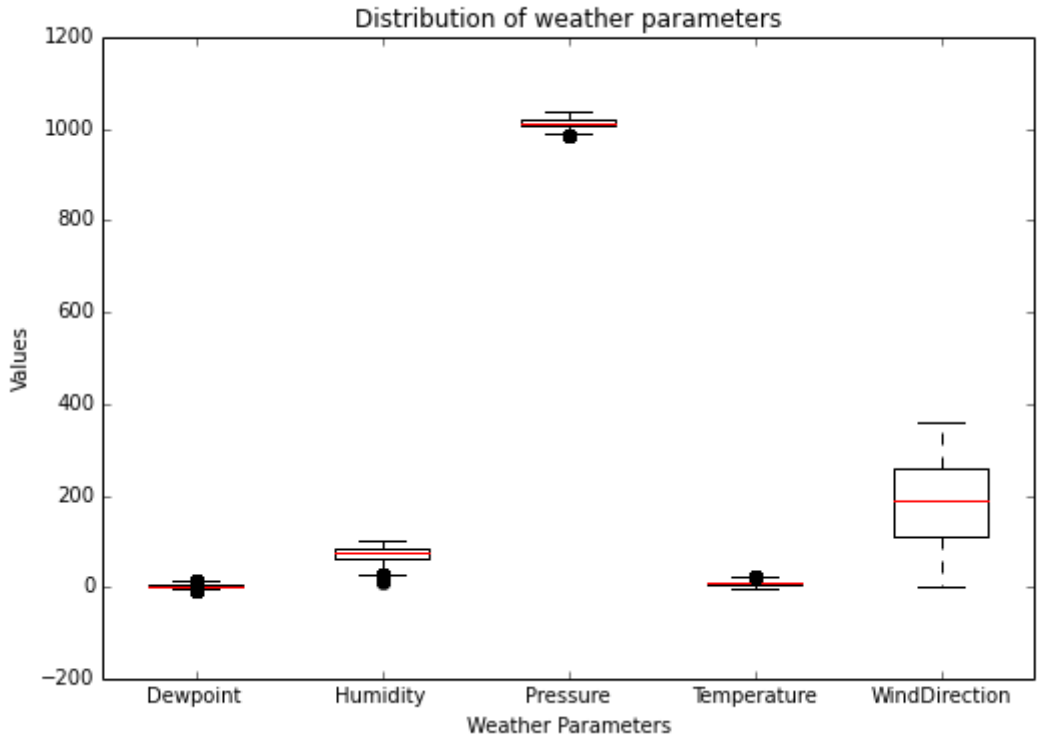
```
('Time taken', -0.004943132400512695)
```

Took 0 sec. Last updated by anonymous at March 30 2017, 8:30:24 PM.

```
%pyspark
start = timeit.timeit()
import matplotlib.pyplot as plt

#now we are in good state as our null values are vanished.
#in this code step, we will check distribution of our data.

data = [weather.ix[:,1], weather.ix[:,2], weather.ix[:,3], weather.ix[:,4], weather.ix[:,5]]

parameter_names = ['Dewpoint', 'Humidity', 'Pressure', 'Temperature', 'WindDirection']

fig, axis = plt.subplots()
axis.set_title("Distribution of weather parameters")
axis.set_xlabel('Weather Parameters')
axis.set_ylabel('Values')
```

FINISHED

```
day_plot = plt.boxplot(data, sym='o', vert=1, whis=1.5)
plt.setp(day_plot['boxes'], color = 'black')
plt.setp(day_plot['whiskers'], color = 'black')
plt.setp(day_plot['fliers'], color = 'black', marker = 'o')
axis.set_xticklabels(parameter_names)
plt.show()
end = timeit.timeit()
```



('Time taken', -0.0022840499877929688)

Took 0 sec. Last updated by anonymous at March 30 2017, 8:30:05 PM.

FINISHED

```
%pyspark
start = timeit.timeit()
# column-wise and Multiple Function Application
grouped_pressure = weather.groupby(['Pressure'])
end = timeit.timeit()
print("Time taken", end - start)
```

```
('Time taken', -0.0006330013275146484)
```

Took 0 sec. Last updated by anonymous at March 30 2017, 8:29:40 PM.

FINISHED

```
%pyspark
start = timeit.timeit()

print(grouped_pressure.apply(lambda weather: weather['Humidity'].corr(weather['Temperature'])))
end = timeit.timeit()
print("Time taken", end - start)
```

```
Pressure
986.000000    -0.866025
987.000000    -0.707428
988.000000    -0.808061
989.000000    -0.909697
990.000000    -0.867790
991.000000    -0.594529
992.000000     0.327185
993.000000     0.311631
994.000000     0.255130
995.000000     0.024581
996.000000    -0.172178
997.000000    -0.553450
998.000000    -0.589844
999.000000    -0.661983
1000.000000   -0.331268
1001.000000   -0.376863
1002.000000   -0.417666
```

Took 0 sec. Last updated by anonymous at March 30 2017, 8:28:39 PM.

FINISHED

```
%pyspark
start = timeit.timeit()
import statsmodels.api as sm
def regression(data, yvar, xvars):
  Y = data[yvar]
  X = data[xvars]
  X['intercept'] = 1.
  result = sm.OLS(Y,X).fit()
  return result.params
end = timeit.timeit()
print("Time taken", end - start)
```

```
('Time taken', -0.0017590522766113281)
```

Took 0 sec. Last updated by anonymous at March 30 2017, 8:28:01 PM.

FINISHED

```
%pyspark
start = timeit.timeit()

#regression(weather_dataset,'Humidity','Temperature')
print(grouped_pressure.apply(regression,'Humidity',['Temperature']))
# ['Dewpoint', 'Humidity', 'Pressure', 'Temperature', 'WindDirection']
end = timeit.timeit()
print("Time taken", end - start)
```

Pressure	Temperature	intercept
986.000000	-7.500000	123.500000
987.000000	-6.402778	114.472222
988.000000	-6.416451	114.394296
989.000000	-5.805901	109.166149
990.000000	-6.048119	105.118985
991.000000	-3.661290	93.661290
992.000000	0.797475	71.931089
993.000000	0.651748	74.142330
994.000000	0.384854	77.341932
995.000000	0.075142	77.804207
996.000000	-0.563072	84.290997
997.000000	-2.240947	96.214336
998.000000	-2.317430	94.806914
999.000000	-2.369485	92.758444
1000.000000	-1.297309	85.976496
1001.000000	-1.581189	89.527090

Took 0 sec. Last updated by anonymous at March 30 2017, 8:27:41 PM.

```
%pyspark

start = timeit.timeit()

pearsonr(weather['Humidity'], weather['Temperature'])
print("Pearson's correlation coefficient, between humidity & temperature",pearsonr(weather['Hu

end = timeit.timeit()
print("Time taken", end - start)
```

FINISHED

```
("Pearson's correlation coefficient, between humidity & temperature", -0.53273625238587075)
('Time taken', -0.0013527870178222656)
```

Took 0 sec. Last updated by anonymous at March 30 2017, 8:23:01 PM.

```
%pyspark
```

READY