Clustering with scikit-learn

In this notebook, we will learn how to perform k-means lustering using scikit-learn in Python.

We will use cluster analysis to generate a big picture model of the weather at a local station using a minute-graunlarity data. In this dataset, we have in the order of millions records. How do we create 12 clusters our of them?

NOTE: The dataset we will use is in a large CSV file called *minute_weather.csv*. Please download it into the *weather* directory in your *Week-7-MachineLearning* folder. The download link is: https://drive.google.com/open?id=0B8iiZ7pSaSFZb3ItQ1l4LWRMTjg (https://drive.google.com/open?id=0B8iiZ7pSaSFZb3ItQ1l4LWRMTjg)

Importing the Necessary Libraries

In [1]:

```
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
import pandas as pd
import numpy as np
from itertools import cycle, islice
import matplotlib.pyplot as plt
from pandas.plotting import parallel_coordinates

%matplotlib inline
```

Creating a Pandas DataFrame from a CSV file

In [2]:

```
data = pd.read_csv('./weather/minute_weather.csv')
```

Minute Weather Data Description

The **minute weather dataset** comes from the same source as the daily weather dataset that we used in the decision tree based classifier notebook. The main difference between these two datasets is that the minute weather dataset contains raw sensor measurements captured at one-minute intervals. Daily weather dataset instead contained processed and well curated data. The data is in the file **minute_weather.csv**, which is a comma-separated file.

As with the daily 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.

Each row in **minute_weather.csv** contains weather data captured for a one-minute interval. Each row, or sample, consists of the following variables:

- **rowID:** unique number for each row (*Unit: NA*)
- **hpwren_timestamp:** timestamp of measure (*Unit: year-month-day hour:minute:second*)
- **air_pressure:** air pressure measured at the timestamp (*Unit: hectopascals*)
- **air_temp:** air temperature measure at the timestamp (*Unit: degrees Fahrenheit*)
- **avg_wind_direction:** wind direction averaged over the minute before the timestamp (*Unit: degrees, with 0 means coming from the North, and increasing clockwise*)
- **avg_wind_speed:** wind speed averaged over the minute before the timestamp (*Unit: meters per second*)
- **max_wind_direction:** highest wind direction in the minute before the timestamp (*Unit: degrees, with 0 being North and increasing clockwise*)
- **max_wind_speed:** highest wind speed in the minute before the timestamp (*Unit: meters per second*)
- **min_wind_direction:** smallest wind direction in the minute before the timestamp (*Unit: degrees, with 0 being North and inceasing clockwise*)
- **min_wind_speed:** smallest wind speed in the minute before the timestamp (*Unit: meters per second*)
- **rain_accumulation:** amount of accumulated rain measured at the timestamp (*Unit: millimeters*)
- **rain_duration:** length of time rain has fallen as measured at the timestamp (*Unit: seconds*)
- **relative_humidity:** relative humidity measured at the timestamp (*Unit: percent*)

In [3]:

```
data.columns
```

Out[3]:

```
Index(['rowID', 'hpwren_timestamp', 'air_pressure', 'air_temp',
       'avg_wind_direction', 'avg_wind_speed', 'max_wind_direction',
       'max_wind_speed', 'min_wind_direction', 'min_wind_speed',
       'rain_accumulation', 'rain_duration', 'relative_humidity'],
      dtype='object')
```

In [4]:

```
data.shape
```

Out[4]:

```
(1587257, 13)
```

In [5]:

```
data.head()
```

Out[5]:

	rowID	hpwren_timestamp	air_pressure	air_temp	avg_wind_direction	avg_wind_speed	max_v
0	0	2011-09-10 00:00:49	912.3	64.76	97.0	1.2	
1	1	2011-09-10 00:01:49	912.3	63.86	161.0	0.8	
2	2	2011-09-10 00:02:49	912.3	64.22	77.0	0.7	
3	3	2011-09-10 00:03:49	912.3	64.40	89.0	1.2	
4	4	2011-09-10 00:04:49	912.3	64.40	185.0	0.4	

Data Sampling

Lots of rows, so let us sample down by taking every 10th row.

In [6]:

```
sampled_df = data[(data['rowID'] % 10) == 0]
sampled_df.shape
```

Out[6]:

```
(158726, 13)
```

Statistics

In [7]:

```
sampled_df.describe()
```

Out[7]:

	rowID	air_pressure	air_temp	avg_wind_direction	avg_wind_speed	max_w
count	1.587260e+05	158726.000000	158726.000000	158680.000000	158680.000000	15
mean	7.936250e+05	916.830161	61.851589	162.156100	2.775215	
std	4.582039e+05	3.051717	11.833569	95.278201	2.057624	
min	0.000000e+00	905.000000	31.640000	0.000000	0.000000	
25%	3.968125e+05	914.800000	52.700000	62.000000	1.300000	
50%	7.936250e+05	916.700000	62.240000	182.000000	2.200000	
75%	1.190438e+06	918.700000	70.880000	217.000000	3.800000	
max	1.587250e+06	929.500000	99.500000	359.000000	31.900000	

In [8]:

```
sampled_df.describe().transpose()
```

Out[8]:

	count	mean	std	min	25%	50%	
rowID	158726.0	793625.000000	458203.937509	0.00	396812.5	793625.00	11904
air_pressure	158726.0	916.830161	3.051717	905.00	914.8	916.70	9
air_temp	158726.0	61.851589	11.833569	31.64	52.7	62.24	
avg_wind_direction	158680.0	162.156100	95.278201	0.00	62.0	182.00	2
avg_wind_speed	158680.0	2.775215	2.057624	0.00	1.3	2.20	
max_wind_direction	158680.0	163.462144	92.452139	0.00	68.0	187.00	2
max_wind_speed	158680.0	3.400558	2.418802	0.10	1.6	2.70	
min_wind_direction	158680.0	166.774017	97.441109	0.00	76.0	180.00	2
min_wind_speed	158680.0	2.134664	1.742113	0.00	0.8	1.60	
rain_accumulation	158725.0	0.000318	0.011236	0.00	0.0	0.00	
rain_duration	158725.0	0.409627	8.665523	0.00	0.0	0.00	
relative_humidity	158726.0	47.609470	26.214409	0.90	24.7	44.70	

In [9]:

```
sampled_df[sampled_df['rain_accumulation'] == 0].shape
```

Out[9]:

```
(157812, 13)
```

In [10]:

```
sampled_df[sampled_df['rain_duration'] == 0].shape
```

Out[10]:

```
(157237, 13)
```

We have about 157,000 rows out of 158,000. So, it looks like a ssignificant number of rows are zero compared to the size of data. So, let's drop these columns.

Drop all the Rows with Empty rain_duration and rain_accumulation

In [11]:

```
del sampled_df['rain_accumulation']
del sampled_df['rain_duration']
```

In [12]:

```
rows_before = sampled_df.shape[0]
sampled_df = sampled_df.dropna()
rows_after = sampled_df.shape[0]
```

How many rows did we drop ?

In [13]:

```
rows_before - rows_after
```

Out[13]:

```
46
```

In [14]:

```
sampled_df.columns
```

Out[14]:

```
Index(['rowID', 'hpwren_timestamp', 'air_pressure', 'air_temp',
       'avg_wind_direction', 'avg_wind_speed', 'max_wind_direction',
       'max_wind_speed', 'min_wind_direction', 'min_wind_speed',
       'relative_humidity'],
      dtype='object')
```

Select Features of Interest for Clustering

In [15]:

```
features = ['air_pressure', 'air_temp', 'avg_wind_direction', 'avg_wind_speed', 'max
        'max_wind_speed','relative_humidity']
```

In [16]:

```
select_df = sampled_df[features]
```

In [17]:

```
select_df.columns
```

Out[17]:

```
Index(['air_pressure', 'air_temp', 'avg_wind_direction', 'avg_wind_spe
ed',
       'max_wind_direction', 'max_wind_speed', 'relative_humidity'],
      dtype='object')
```

In [18]:

```
select_df
```

Out[18]:

	air_pressure	air_temp	avg_wind_direction	avg_wind_speed	max_wind_direction	max_v
0	912.3	64.76	97.0	1.2	106.0	
10	912.3	62.24	144.0	1.2	167.0	
20	912.2	63.32	100.0	2.0	122.0	
30	912.2	62.60	91.0	2.0	103.0	
40	912.2	64.04	81.0	2.6	88.0	
50	912.1	63.68	102.0	1.2	119.0	
60	912.0	64.04	83.0	0.7	101.0	
70	911.9	64.22	82.0	2.0	97.0	
80	911.9	61.70	67.0	3.3	70.0	
90	911.9	61.34	67.0	3.6	75.0	
100	911.8	62.96	95.0	2.3	106.0	
110	911.8	64.22	83.0	2.1	88.0	
120	911.8	63.86	68.0	2.1	76.0	
130	911.6	64.40	156.0	0.5	203.0	
140	911.5	65.30	85.0	2.2	92.0	
150	911.4	64.58	154.0	1.3	176.0	
160	911.4	65.48	154.0	0.9	208.0	
170	911.5	65.66	95.0	1.1	109.0	
180	911.4	65.66	155.0	1.1	167.0	
190	911.4	67.10	157.0	1.2	172.0	
200	911.4	68.00	53.0	0.3	69.0	
210	911.3	67.64	167.0	1.5	196.0	
220	911.4	67.82	4.0	0.6	25.0	
230	911.4	66.74	172.0	1.3	192.0	
240	911.4	66.56	39.0	0.2	145.0	
250	911.4	65.66	56.0	1.9	67.0	
260	911.5	65.66	74.0	0.8	101.0	
270	911.4	66.92	147.0	0.9	174.0	
280	911.3	64.76	73.0	1.0	82.0	
290	911.3	64.94	164.0	1.3	176.0	
...	...	...	...	...	...	
1586960	914.7	76.46	247.0	0.6	264.0	
1586970	914.8	76.28	208.0	0.7	216.0	

	air_pressure	air_temp	avg_wind_direction	avg_wind_speed	max_wind_direction	max_v
1586980	914.8	76.10	209.0	0.7	216.0	
1586990	914.9	76.28	339.0	0.5	350.0	
1587000	914.9	75.92	344.0	0.4	352.0	
1587010	915.0	75.56	323.0	0.3	348.0	
1587020	915.1	75.56	324.0	1.1	347.0	
1587030	915.1	75.74	1.0	1.3	13.0	
1587040	915.2	75.38	355.0	0.9	1.0	
1587050	915.3	75.38	359.0	1.4	11.0	
1587060	915.4	75.38	11.0	1.1	21.0	
1587070	915.5	75.38	13.0	1.4	24.0	
1587080	915.6	75.20	18.0	1.0	24.0	
1587090	915.6	75.20	356.0	1.7	1.0	
1587100	915.7	75.38	13.0	1.5	24.0	
1587110	915.7	75.02	19.0	1.2	28.0	
1587120	915.7	74.84	25.0	1.4	35.0	
1587130	915.8	74.84	23.0	1.3	30.0	
1587140	915.8	74.84	32.0	1.4	41.0	
1587150	915.8	75.20	23.0	1.1	31.0	
1587160	915.8	75.38	16.0	1.2	28.0	
1587170	915.7	75.38	347.0	1.2	353.0	
1587180	915.8	75.74	326.0	1.2	337.0	
1587190	915.9	75.92	289.0	0.7	309.0	
1587200	915.9	75.74	335.0	0.9	348.0	
1587210	915.9	75.56	330.0	1.0	341.0	
1587220	915.9	75.56	330.0	1.1	341.0	
1587230	915.9	75.56	344.0	1.4	352.0	
1587240	915.9	75.20	359.0	1.3	9.0	
1587250	915.9	74.84	6.0	1.5	20.0	

158680 rows × 7 columns

Scale the Features using StandardScaler

In [19]:

```
X = StandardScaler().fit_transform(select_df)
X
```

Out[19]:

```
array([[-1.48456281,  0.24544455, -0.68385323, ..., -0.62153592,
        -0.74440309,  0.49233835],
       [-1.48456281,  0.03247142, -0.19055941, ...,  0.03826701,
        -0.66171726, -0.34710804],
       [-1.51733167,  0.12374562, -0.65236639, ..., -0.44847286,
        -0.37231683,  0.40839371],
       ...,
       [-0.30488381,  1.15818654,  1.90856325, ...,  2.0393087 ,
        -0.70306017,  0.01538018],
       [-0.30488381,  1.12776181,  2.06599745, ..., -1.67073075,
        -0.74440309, -0.04948614],
       [-0.30488381,  1.09733708, -1.63895404, ..., -1.55174989,
        -0.62037434, -0.05711747]])
```

Use k-Means Clustering

In [20]:

```
kmeans = KMeans(n_clusters=12)
model = kmeans.fit(X)
print('model\n',model)
```

```
model
 KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=12, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)
```

What are the centers of 12 clusters we formed ?

In [21]:

```
centers = model.cluster_centers_
centers
```

Out[21]:

```
array([[-0.68615766,  0.55921626,  0.17911545, -0.58568732,  0.3490503
8,
        -0.59899828, -0.12575638],
       [-0.16298001,  0.86371192, -1.31042752, -0.59006978, -1.1660196
8,
        -0.60540644, -0.64180246],
       [ 0.1308897 ,  0.84517148,  1.41008333, -0.63838012,  1.6741413
,
        -0.58924197, -0.71341326],
       [-0.84058394, -1.1998107 ,  0.37605642,  0.38194641,  0.4747083
4,
         0.36947897,  1.35904098],
       [ 1.19074095, -0.25527402, -1.15497682,  2.12470392, -1.0534936
7,
         2.24158068, -1.13457013],
       [ 0.22599871, -0.99814305,  0.65373577, -0.54697175,  0.8430684
7,
        -0.52987798,  1.16893616],
       [ 1.36653543, -0.08109075, -1.20762783, -0.04842759, -1.0765005
7,
        -0.02802359, -0.97762623],
       [ 0.23392083,  0.31972724,  1.88788603, -0.65180058, -1.5517482
6,
        -0.57663256, -0.28329452],
       [ 0.74763015,  0.40918264,  0.28985322, -0.52859565,  0.4770019
6,
        -0.53463242, -0.76707048],
       [-1.1829521 , -0.86783561,  0.44678541,  1.98744135,  0.5380654
4,
         1.9484275 ,  0.90625578],
       [ 0.05814561, -0.78636062, -1.19632066, -0.57115324, -1.0423282
7,
        -0.58575501,  0.87699375],
       [-0.21370231,  0.63340097,  0.4082897 ,  0.73509622,  0.5164010
5,
         0.67299741, -0.15027853]])
```

Plots

Let us first create some utility functions which will help us in plotting graphs:

In [22]:

```
# Function that creates a DataFrame with a column for Cluster Number

def pd_centers(featuresUsed, centers):
    colNames = list(featuresUsed)
    colNames.append('prediction')

    # Zip with a column called 'prediction' (index)
    Z = [np.append(A, index) for index, A in enumerate(centers)]

    # Convert to pandas data frame for plotting
    P = pd.DataFrame(Z, columns=colNames)
    P['prediction'] = P['prediction'].astype(int)
    return P
```

In [23]:

```
# Function that creates Parallel Plots

def parallel_plot(data):
    my_colors = list(islice(cycle(['b', 'r', 'g', 'y', 'k']), None, len(data)))
    plt.figure(figsize=(15,8)).gca().axes.set_ylim([-3,+3])
    parallel_coordinates(data, 'prediction', color = my_colors, marker='o')
```

In [24]:

```
P = pd_centers(features, centers)
P
```

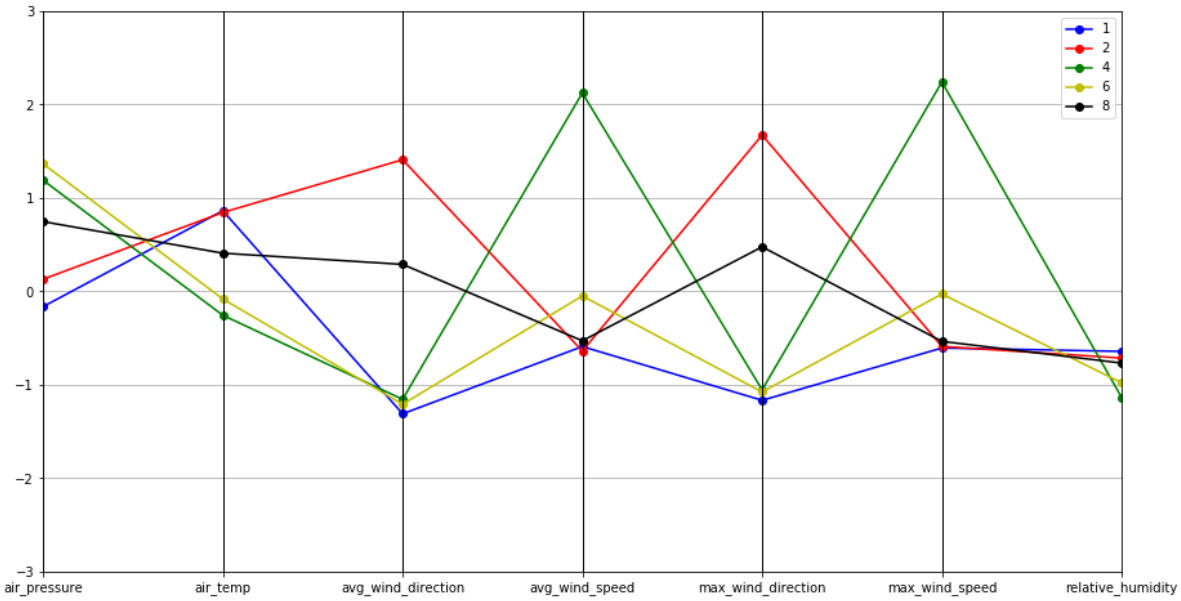
Out[24]:

	air_pressure	air_temp	avg_wind_direction	avg_wind_speed	max_wind_direction	max_wind_
0	-0.686158	0.559216	0.179115	-0.585687	0.349050	-0.5
1	-0.162980	0.863712	-1.310428	-0.590070	-1.166020	-0.6
2	0.130890	0.845171	1.410083	-0.638380	1.674141	-0.5
3	-0.840584	-1.199811	0.376056	0.381946	0.474708	0.3
4	1.190741	-0.255274	-1.154977	2.124704	-1.053494	2.2
5	0.225999	-0.998143	0.653736	-0.546972	0.843068	-0.5
6	1.366535	-0.081091	-1.207628	-0.048428	-1.076501	-0.0
7	0.233921	0.319727	1.887886	-0.651801	-1.551748	-0.5
8	0.747630	0.409183	0.289853	-0.528596	0.477002	-0.5
9	-1.182952	-0.867836	0.446785	1.987441	0.538065	1.9
10	0.058146	-0.786361	-1.196321	-0.571153	-1.042328	-0.5
11	-0.213702	0.633401	0.408290	0.735096	0.516401	0.6

Dry Days

In [25]:

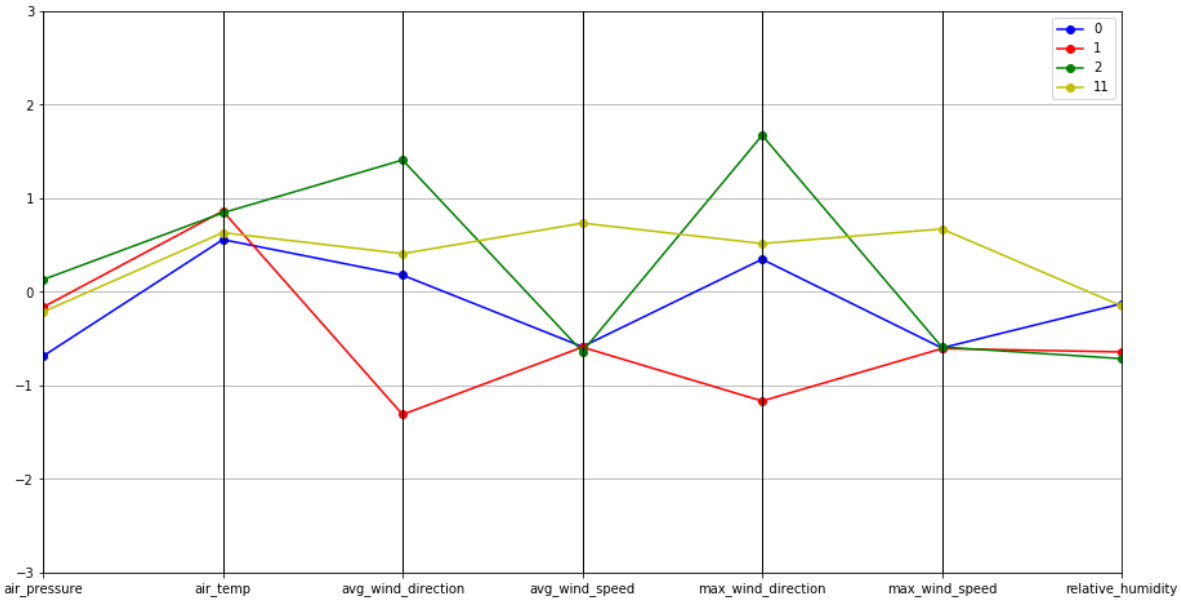
```
parallel_plot(P[P['relative_humidity'] < -0.5])
```



Warm Days

In [26]:

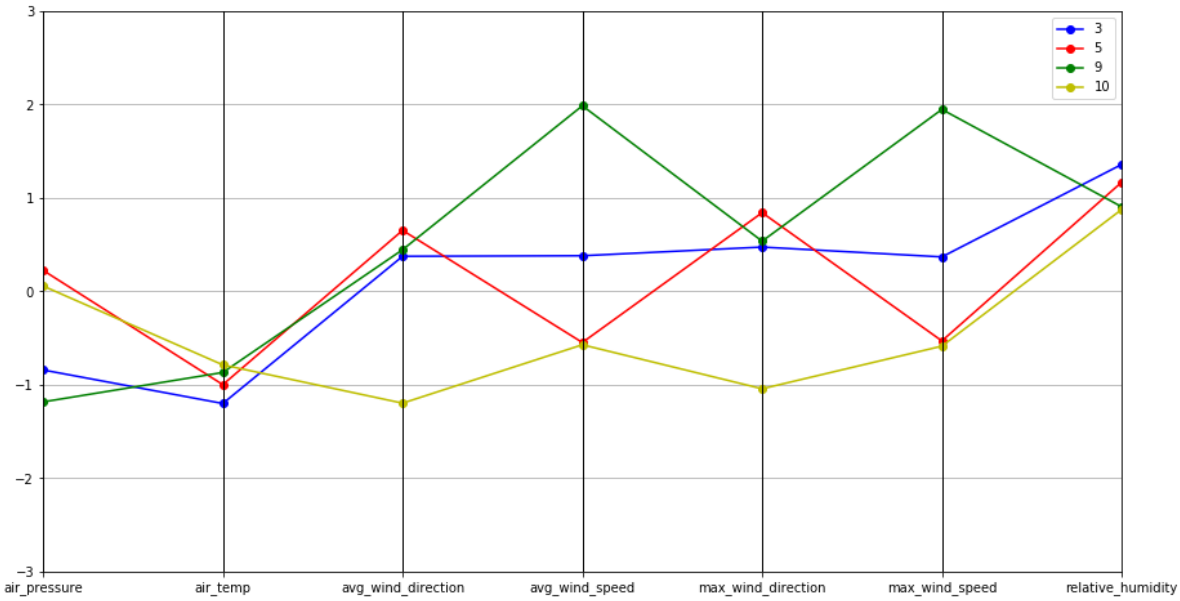
```
parallel_plot(P[P['air_temp'] > 0.5])
```



Cool Days

In [27]:

```
parallel_plot(P[(P['relative_humidity'] > 0.5) & (P['air_temp'] < 0.5)])
```



In [ ]:

```

```