Lab 5: Data Analysis & Visualization for Wind Energy

Lab Description

In this lab, you will use your new knowledge to analyze a real dataset of a wind turbine. To succeed, you will need to upload the dataset into Colab and import all the required libraries such as "Pandas" and "Seaborn", in addition to answering questions related to manipulation and visualization tasks to understand patterns in the data.

Using a smart device and Supervisory Control and Data Acquisition (SCADA) systems, the data is collected and aggregated into 10-minutes intervals.

Features of this dataset include:

- **Date/Time** - 10 min intervals.
- **LV Active Power (kW)** - Power generated by the turbine for that moment.
- **Theoretical Power (kWh)** - The theoretical power values that the turbine generates with that wind speed which is given by the turbine manufacturer. It represents the maximum possible power using the *Betz Law*.
- **Wind Speed (m/s)** - Speed of wind at the hub height of the turbine (the wind speed that the turbine uses for electricity generation).
- **Wind Direction** (°) - Direction of the wind at the hub height of the turbine (wind turbines turn to this direction automatically).

1 Questions

1.1 Load the data

- Your first assignment is to import the required Python libraries that you might need to complete this exercise.
- Then, load the corresponding data file ('T1.csv') into the `df_wind` data-frame. Which column do you have to use as the index? Why?

1.2 Review the data

In this part, we will review the dataset, in order to allow you first to understand what you have "in your hand".

(i) What is the size of the loaded data frame?

(ii) Print the first and last 5 rows of the `df_Wind` data frame. What can you conclude? For which year(s) the turbine has recorded data?

(iii) Print the data type of each column in this dataset.

(iv) Print the mean and standard deviation of **'Wind_speed'**.

1.3 Data Manipulation

(v) Change column names:

- **'Power'** instead of **'LV Active Power (kW)'** .
- **'Wind_speed'** instead of **'Wind Speed (m/s)'**.
- **'Theoretical_power'** instead of **Theoretical Power (kWh)**.
- **'Wind_direction'** instead of **Wind Direction** (°).

(vi) Plot the variation of the originally produced power (for 10 min intervals) versus date/time.

(vii) Create a new column, called *'loss'* that is equal to the difference between the theoretical power and the produced one. Now, plot the loss of energy versus date/time. What can you conclude?

(viii) In fact, a wind turbine starts generating electricity between 12-14km/h (3.3-3.8m/s). If the wind speed is higher than 3.3m/s and the power output is zero, we can assume that the turbine is under maintenance. Therefore, you should detect which rows verify this condition (where the wind speed is above 3.3 and the power is less than zero). Then, create a new data frame called *'df_wind_nm'* where times of maintenance are not included. This can be done by iterating the following operation.

```
df_wind_nm = df_wind[ ~((df_wind['Power']<=0)
& (df_wind['Wind_speed'] > 3.3))]
```

Now, plot the loss of energy of the *'df_wind_nm'* data frame versus date/time. What can you conclude in comparison to the previous plot?

(ix) Plot the difference of loss between maintenance and no maintenance data frames (**'df_wind'** and **'df_wind_nm'**). What can you conclude?

(x) Plot the kde of power and wind speed of the no maintenance data frame (**'df_wind_nm'**).

(xi) Next, we will use the following function to create two new columns (x and y components of wind direction).

```
def x_y_component(wind_direction, wind_speed):
    """
    Convert degrees to x,y components
    """
    #convert to radians
    radians = (wind_direction * np.pi)/180
    # give the x, y compenents
    x = wind_speed * np.cos(radians)
    y = wind_speed * np.sin(radians)

    return x,y
```

The x and y components of Wind Direction and Wind Speed should be appended to the dataset. The equations below show how both components are calculated in the previous function.

$$x = W_s \times cos(\frac{W_d * \pi}{180}) \tag{1}$$

and

$$y = W_s \times sin(\frac{W_d * \pi}{180}) \tag{2}$$

where W_s and W_d represent the Wind Speed and the Wind Direction, respectively.

(xii) Due to the random noise, the data will be re-sampled. Therefore, create an hourly, daily, weekly, and monthly data frame resampled by the mean from the no-maintenance dataset. To do it, you should use the *'resample'* method of Pandas.

```
X_df_wind = df_wind.resample('X').mean()
```

where X can be Hour ('H'), Day ('D'), Week ('W'), and Month ('M'). Then, plot the variation of hourly, daily, weekly, and monthly produced power versus date/time as sub-plots. You can use another technique, which is based on "asfreq" function. What is the distinction between these methods' "resample" and "asfreq" functions? What conclusions can you draw?

(xiii) Replace all Nan values with interpolated values for each column of the hourly, daily, weekly, and monthly data frames. You should use the '*interpolate()*' function for this purpose.

(xiv) You should add and use the following function(*'direction'*) to create a new categorical column called **'Direction'** to the obtained filtered data frames. In fact, this function uses the wind speed as an input (here is x). Add the direction column to the daily, weekly, and monthly (aggregated) data frame obtained from the no maintenance data frame (**'df_wind_nm'**).

```
def direction(x):
    if x > 348.75 or x<11.25: return 'N'
    if x < 33.75: return 'NNE'
    if x < 56.25: return 'NE'
    if x < 78.75: return 'ENE'
    if x < 101.25: return 'E'
    if x < 123.75: return 'ESE'
    if x < 146.25: return 'SE'
    if x < 168.75: return 'SSE'
    if x < 191.25: return 'S'
    if x < 213.75: return 'SSW'
    if x < 236.25: return 'SW'
    if x < 258.75: return 'WSW'
    if x < 281.25: return 'W'
    if x < 303.75: return 'WNW'
    if x < 326.25: return 'NW'
    else: return 'NNW'
```

(xv) Plot the variation of produced energy of the hourly data frame during December. Then, plot another one for the last two weeks of December.

(xvi) Plot the weekly and monthly average generated power in the same figure.

(xvii) Apply the following code. What can you conclude?

```
m_s_t_se = df_wind_nm.groupby('Wind_speed')['Theoretical_power'].mean()
m_s_p_se = df_wind_nm.groupby('Wind_speed')['Power'].mean()
plt.figure(figsize=(10,5))
plt.plot(m_s_t_se,label='Theoretical')
plt.plot(m_s_p_se,label='Actual Power')
plt.xlabel('Wind Speed (m/s)')
plt.ylabel('Power (kW)')
plt.title('Theoretical Power vs Actual Power Curve')
plt.grid()
plt.legend()
plt.show()
```

(xviii) Plot a pair-plot for the filtered data frame. What conclusions can you draw?

(xix) Plot the correlation heatmap. What can you conclude?

- Which variables have a stronger correlation with *'Power'*? Can you list them?
- Which variables do not have a correlation with *'Power'*? Can you list them?
- Can we obtain the same results if other correlation methods such as "Kendall Tau correlation coefficient" or"Spearman rank correlation" are used?

(xx) Plot the loss of energy versus the wind direction. What can you conclude?

(xxi) Plot the theoretical and experimentally produced energy in the function of the direction (bar graph).

(xxii) Create a 3D scatter plot with variables (x, y, z): where 'wind_speed', 'wind_direction', and 'Power' represent the axes x, y and z, respectively. What can you conclude from this result?