JACOBS UNIVERSITY

Advanced Project 1 (MRD005-340001)

Spring 2022

Analysis of Increment Statistic and PCA for Wind Power

(Project Report)

Author: *Ali Afsharian* Instructor: *Prof. Dr. Stefan Kettemann*

Abstract

Wind and solar power are notoriously uncertain, leading utilities to default to fossil fuels, which are available on demand. Predicting wind energy yields can reduce some of that uncertainty allowing utilities to benefit from renewable sources' advantages and easing dependence on fossil-fuel and nuclear sources. Making accurate predictions requires a solid understanding of wind characteristics and its intermittent behavior. This report provides an exploratory data analysis for wind speed and output power time series of ten wind turbines selected from a wind farm in the US. Different visualizations are deployed for wind velocity, output power and the increments of measurements. In addition, an EOF analysis is done on the sample dataset to find the temporal bases and corresponding spatial coefficients. This is a promising approach that can be useful for spatio-temporal prediction of climate and environmental data in a combination with machine learning techniques. All codes are presented at the end of the report.

Table of Content

1. Introduction 2
2. Main Topic 3
2.1 Data Description 3
2.2 Data Visualization 4
2.2.1 Time Series Plots 4
2.2.2 Heat Map 6
2.3 Analysis 7
2.3.1 Histograms 7
2.3.2 Increment Analysis 8
2.3.3 Scatter Plots and Correlation Matrix 9
2.3.4 EOF Analysis 14
3. Conclusion and Outlook 16
4. References 17
5. Code 18

1. Introduction

It is anticipated that significantly more renewable energy sources would be deployed as a result of a rising demand for energy services and attention to environmental issues. Due to decades of scientific discovery and technological advancement, wind energy is already becoming a common source of electricity. According to the WindEurope, 116 GW of new wind farms will be installed in Europe between 2022 and 2026. Moreover, Germany will have the largest wind market in Europe as a result of its strong expected onshore market's performance over the next five years (19.7 GW) and growing offshore installations (5.4 GW) [1].

The renewable sources, such as wind, are highly fluctuating across different time scale which could greatly influence the power generated from these sources [2]. This intermittent behavior could significantly affect the power system operation. In order to have a stable supply grid based on renewable energies and reduce the reserved capacity, deep understanding of random character of wind and the ability to predict the wind velocity and generated power are needed [3].

Different studies has been done on wind time series forecasting. The presented methods can be classified based on their methodology [4]. First, the physical approach that use weather features, such as temperature and pressure, to achieve highest prediction accuracy. The second is the statistical approach, like ARMA model, that is based on the numerus historical data and disregards weather conditions. Moreover, in many forecasting methods, both physical and statistical models are utilize. In hybrid approach uses weather forecasts and time series analysis simultaneously.

The Empirical Orthogonal Function analysis, usually just referred to as EOF analysis, is among the most popular and commonly used technique in geophysical sciences. The EOF terminology was introduced by Lorenz (1956) in a statistical weather prediction project at Massachusetts Institution of Technology [5]. The objective of the EOF analysis is to identify a new set of variables by linear combination of original variables that explains the maximum variance of the data.

The EOF analysis has a strong constraint on the orthogonality of spatial pattern (EOFs) and temporal coefficients (PCs) of extracted modes. Rotated EOF techniques (REOF) have been developed to address some of the prior limitations of EOF patterns such as physical interpretability while simplify spatial structures [6] [7]. In this report, EOF has been deployed on the wind speed data, and its ability to reconstruct the real data is evaluated.

2. Main Topic

2.1 Data Description

In this report, I use the NREL (National Renewable Energy Laboratory) downloaded from a repository of University of Oldenburg [8]. The NREL Western Wind Resources Dataset consists wind speed and power output measurements for 32,043 wind turbines across the United States for the whole year 2006. The location of all wind turbines in the dataset is presented in Fig. 1. The red circle on the map shows the selected wind farm region used for further analysis. The data resolution is ten minutes and there are 52,560 instances for each turbine. An ID number identifies each turbine. The metadata contains information about the locations of the turbines including Latitude, Longitude, and the State Code. A short sample of the data for one turbine is shown in Table 1.

Moreover, the dataset has been already cleaned without any missing values which means no further steps for data cleaning are required. Table 2 describes the dataset and missing values for each column.

Figure 1. Location of all turbines in the dataset

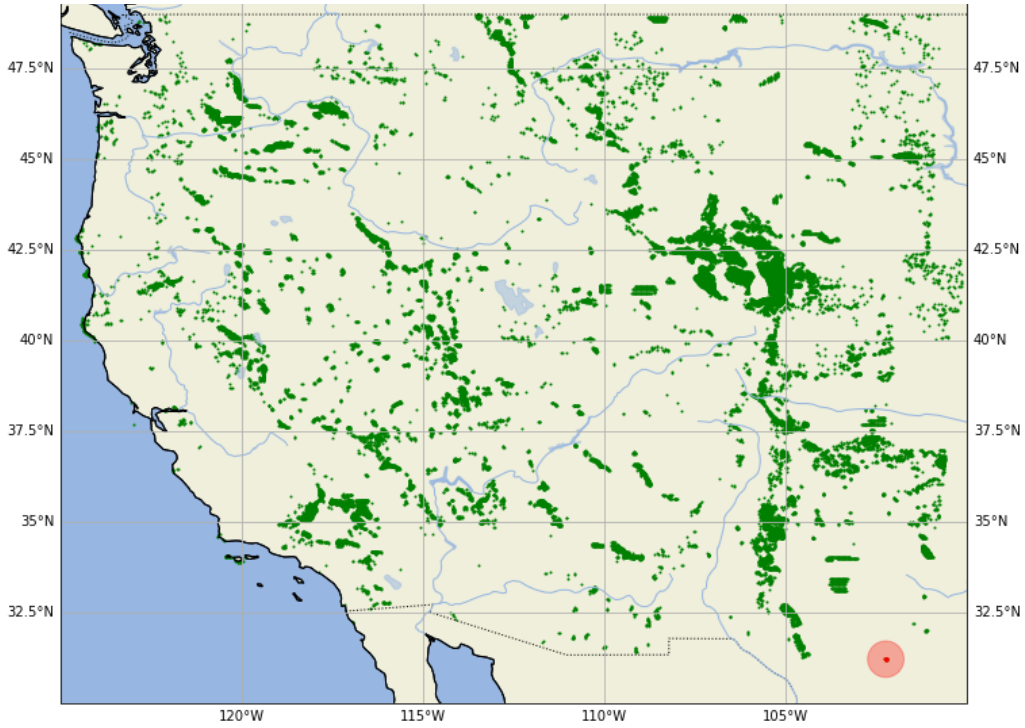


Table 1. First five rows in the dataset

Date(YYYY-MM-DD hh:mm:ss)	100m wind speed (m/s)	rated power output at 100m (MW)	SCORE-lite power output at 100m (MW)	CorrectedScore
2006-01-01 00:00:00	5.12	1.788	2.012	2.012
2006-01-01 00:10:00	5.12	1.788	2.227	2.227
2006-01-01 00:20:00	5.14	1.812	0.130	0.130
2006-01-01 00:30:00	5.17	1.848	2.311	2.311
2006-01-01 00:40:00	5.24	1.929	1.849	1.849

Table 2. Missing values for each column in the dataset

```
DatetimeIndex: 52560 entries, 2006-01-01 00:00:00 to 2006-12-31 23:50:00
Data columns (total 4 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   100m wind speed (m/s)                 52560 non-null  float64
 1    rated power output at 100m (MW)      52560 non-null  float64
 2    SCORE-lite power output at 100m (MW) 52560 non-null  float64
 3   CorrectedScore                        52560 non-null  float64
dtypes: float64(4)
memory usage: 2.0 MB
```

2.2 Data Visualization

2.2.1 Time Series Plots

The wind speed time series of turbine ID1 over the whole year and January are plotted in Fig. 2 and 3 respectively. We can clearly see that the wind speed fluctuates on different time scales, representing the uncertainty in wind behavior. Moreover, the output power measurements are shown for the same periods in Fig. 4 and 5. By looking at Fig. 5, some plateaus can be seen for the power values. This is because of the characteristic of the power curve in wind turbines. Fig. 6 depicts a typical power curve for a wind turbine. The power output is zero in the initial zone where wind speed is below than cut-in speed. The produced power increases between the cut-in and the rated speed. After that, for a wind speed higher than the rated speed, the turbine produces a constant power until the cut-off speed is reached. Beyond this speed, no electricity is produced and the turbine is shut down to prevent damage during strong winds [9].

Figure 2. Wind speed time series for whole year 2006

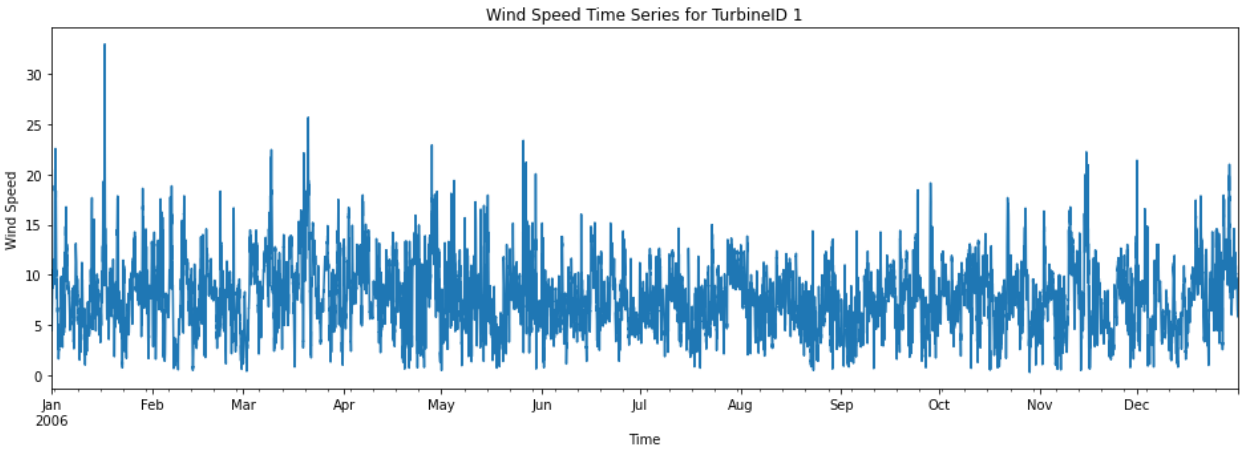


Figure 3. Wind speed time series for January

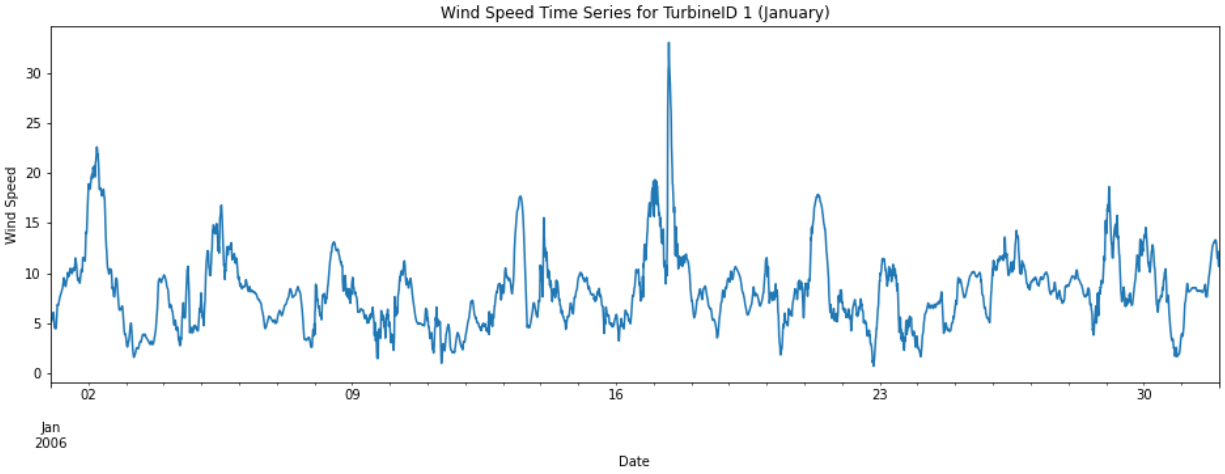


Figure 4. Output power time series for whole year 2006

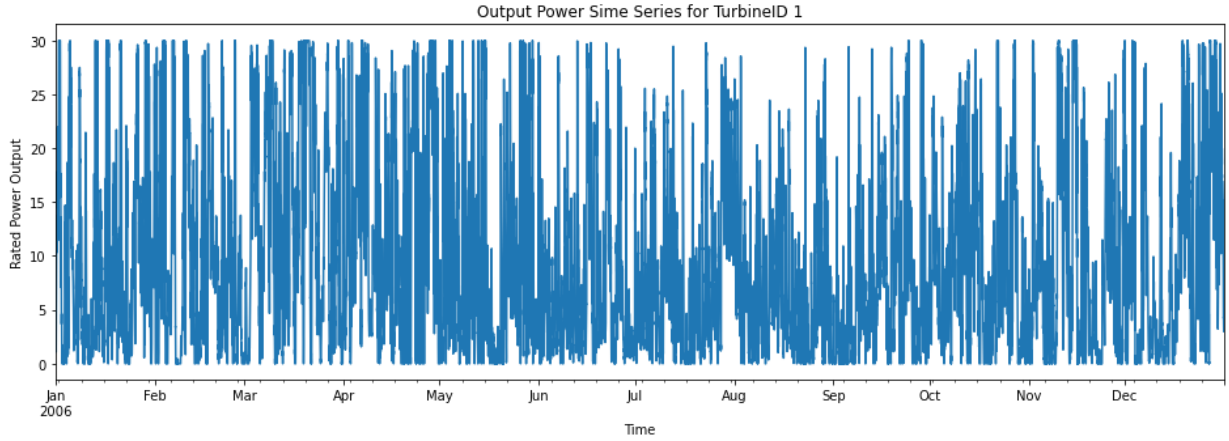


Figure 5. Output power time series for January

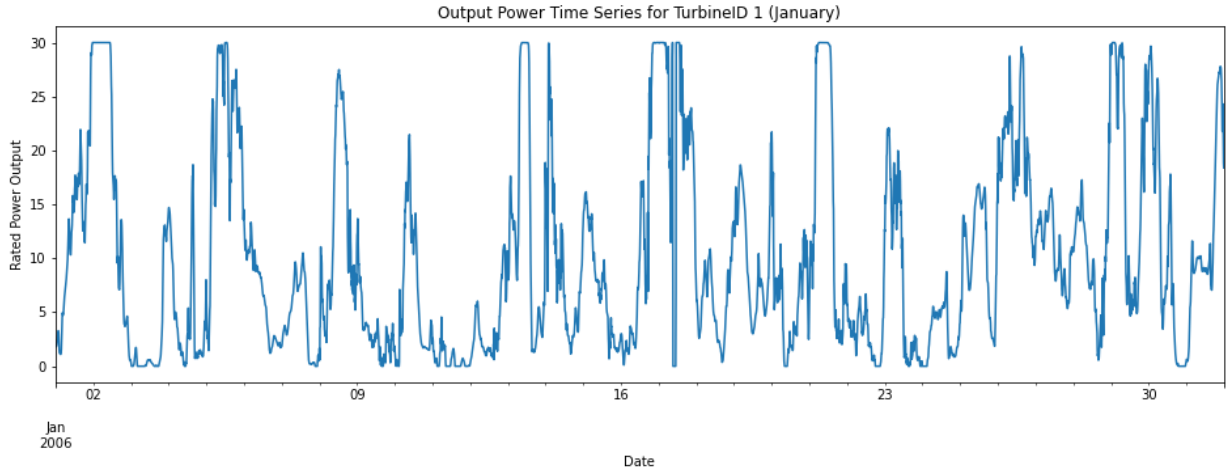
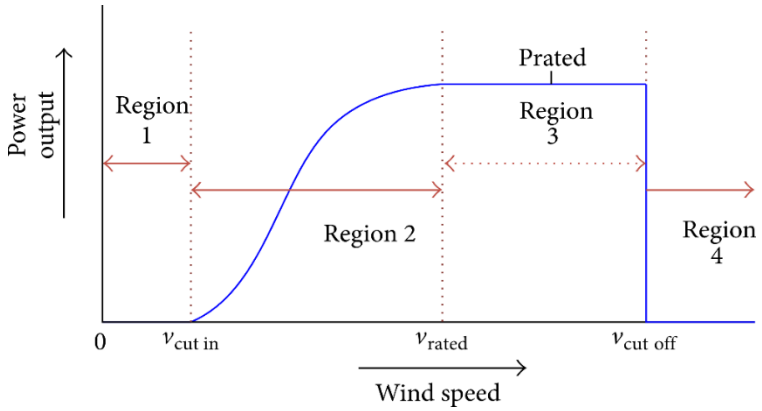


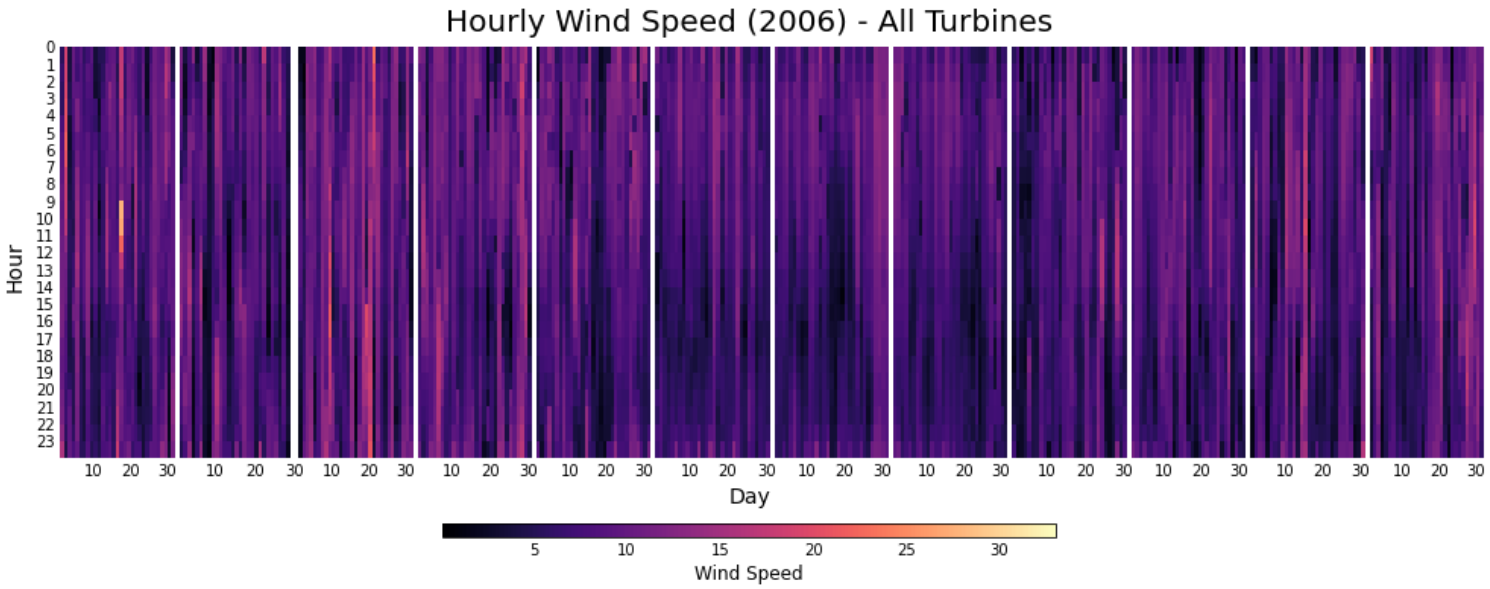
Figure 6. Power curve for wind turbine



2.2.2 Heat Map

In order to offer a comprehensive view, the average hourly wind velocity measurements of all turbines are shown as a heat map over the whole period in Fig. 7. Although no clear seasonality pattern can be seen from the heat map, in general, the amount of wind speeds during summertime is lower compared to the rest of the year. On average, March experienced winds with higher velocities than other months. Additionally, it is generally more likely for winds with higher speeds to blow from 12 PM until 12 AM especially in the summer season.

Figure 7. Heat map of hourly wind speed for whole time period



2.3 Analysis

2.3.1 Histograms

Fig. 8 (a) and (b) display the histogram of wind speed data for turbine ID1 and for all turbines respectively. Since the turbines are selected from one specific wind farm and are in the vicinity of each other, we can see the similar structure in wind speed data for turbine ID and all turbines. The results indicate that the data has right-skewed distribution and the majority of measured wind speeds are less than 15 m / s.

Moreover, when we look at the histogram of output power in Fig. 9, we can see two peaks, at 0 and 30 MW. This is rooted in the sigmoid shape of the power curve as discussed above. The output power measurements for turbine ID1 have also right-skewed distribution.

Figure 8. Histogram of wind speed

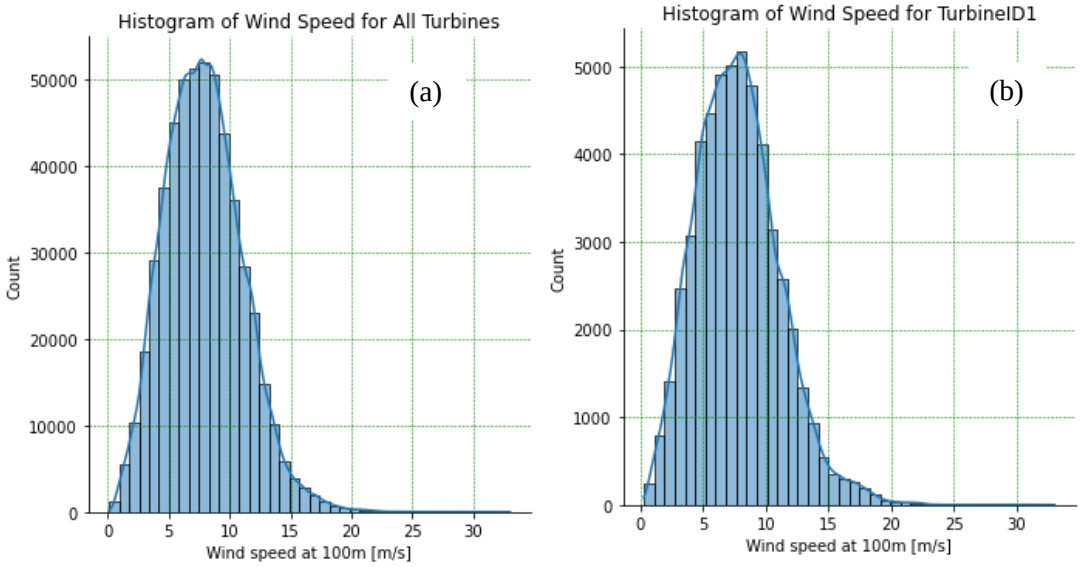
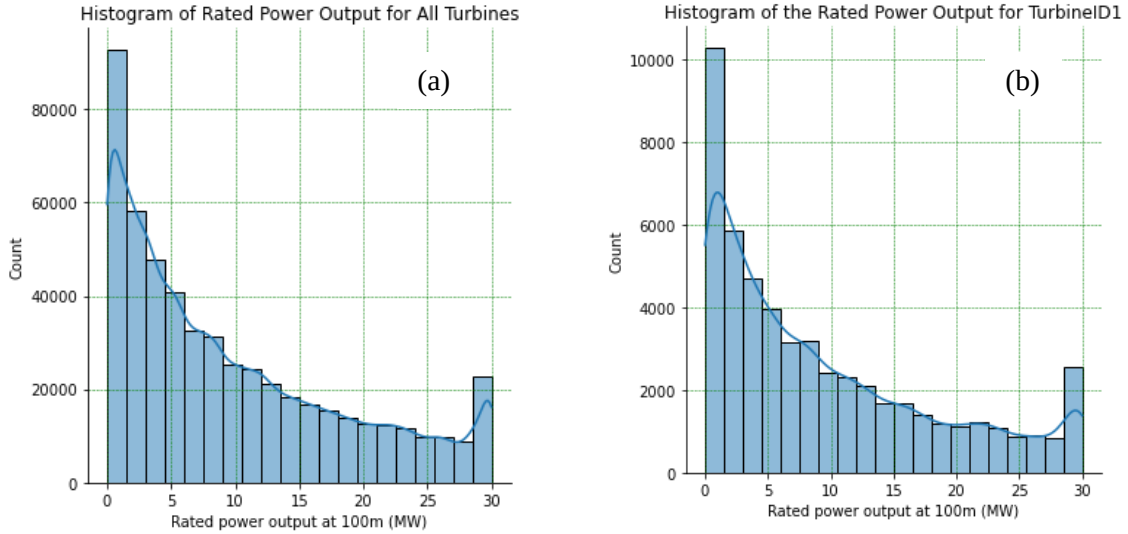


Figure 9. Histogram of output power



2.3.2 Increment Analysis

In this part, I calculated the increments of velocity and power (10 minutes and 1 hour) for turbine ID1 and plotted their corresponding kernel density estimation compared to the Normal distribution. The increments are basically achieved by differencing the values (wind velocity or output power) between two time points as in Eq. 1.

$$\Delta_r v = v(x) - v(x + r) \quad (1)$$

Fig. 10 (a) and (b) display the pdf of wind speed and Normal distribution with the same standard deviation for 10 minutes and 1 hour respectively. Calculated Fisher kurtosis for 10 minutes and 1-hour intervals are 62.45 and 16.75 respectively which are greater than three, the kurtosis for normal distribution. By looking at the results, we can find that both increments have leptokurtic distributions which means the distribution has frequent outliers with several extreme observations. Moreover, the increments of wind power with the same time intervals are plotted in Fig. 11 (a) and (b). As it is expected, the increments for power follow the same pattern as wind speed increments with Fisher kurtosis 64 and 13.59 for 10 minutes and 1 hour respectively. We can conclude that the 10 minutes and 1-hour increments for both wind speed and power have leptokurtic distribution with a greater likelihood of extreme events as compared to a normal distribution.

Figure 10. PDF of wind speed increments compared to normal distribution

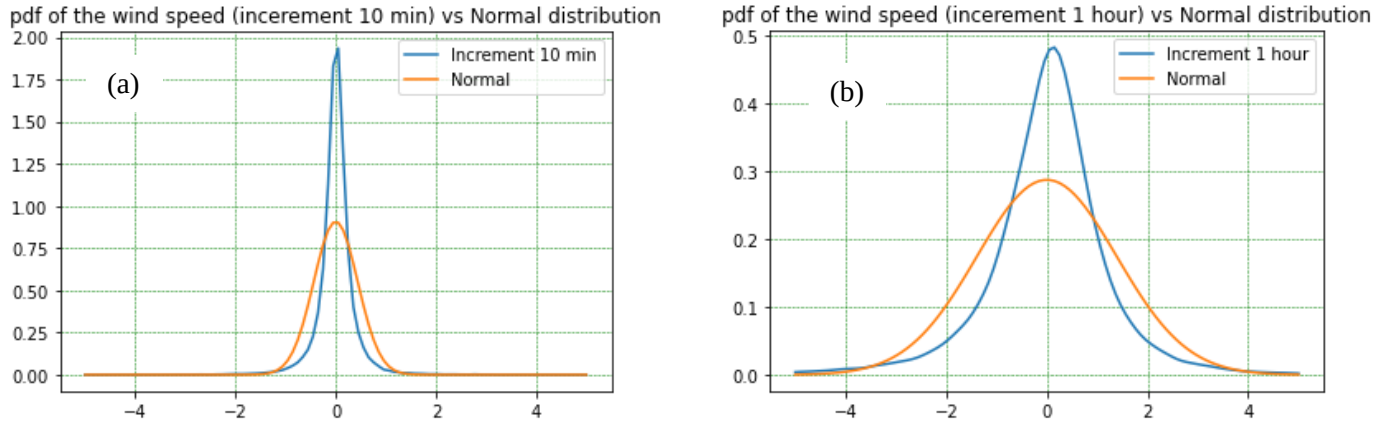
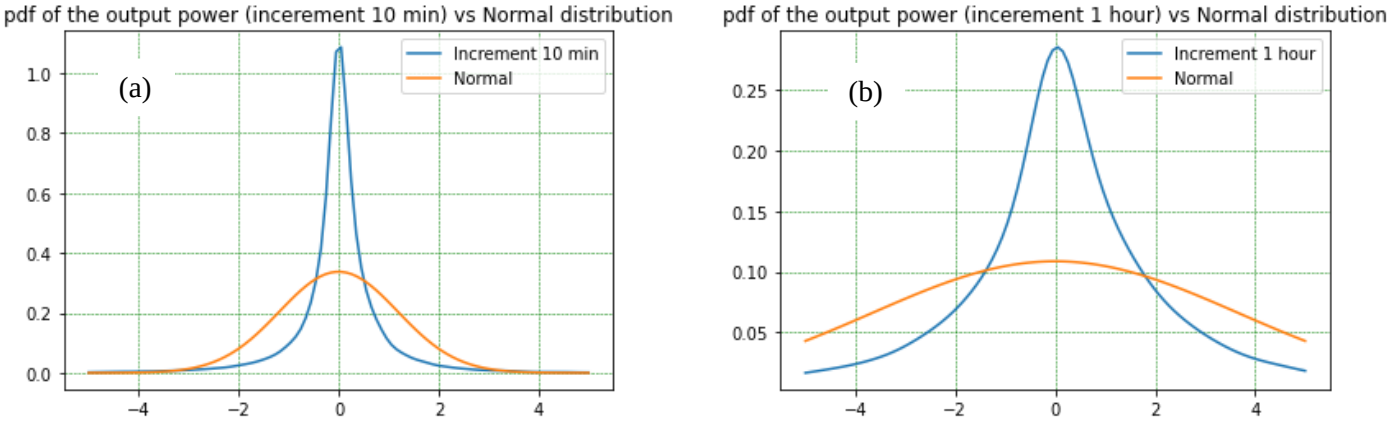


Figure 11. PDF of output power increments compared to normal distribution



2.3.3 Scatter Plots and Correlation Matrix

In order to investigate the correlation between all selected turbines in terms of wind speed and output power, the scatter plots and correlation matrix as a heat map are plotted. Fig. 12 shows the scatter plot for wind velocity for each pair of turbines in the selected wind farm and the distribution functions in the diagonal. As it is expected, the velocity measurements at different turbines are highly correlated because of the fact that the selected turbines are all in one wind farm and relatively close to each other. We can confirm the conclusion by looking at Fig. 13 which determines the Pearson correlation coefficients for wind speed among all turbines with the minimum of 0.93. Consequently, we expect a close relation in terms of the generated power. Fig. 14 and 15 display the scatter plot and correlation matrix of output power among all turbines respectively.

Figure 12. Scatter plot of wind speed for 10 turbines

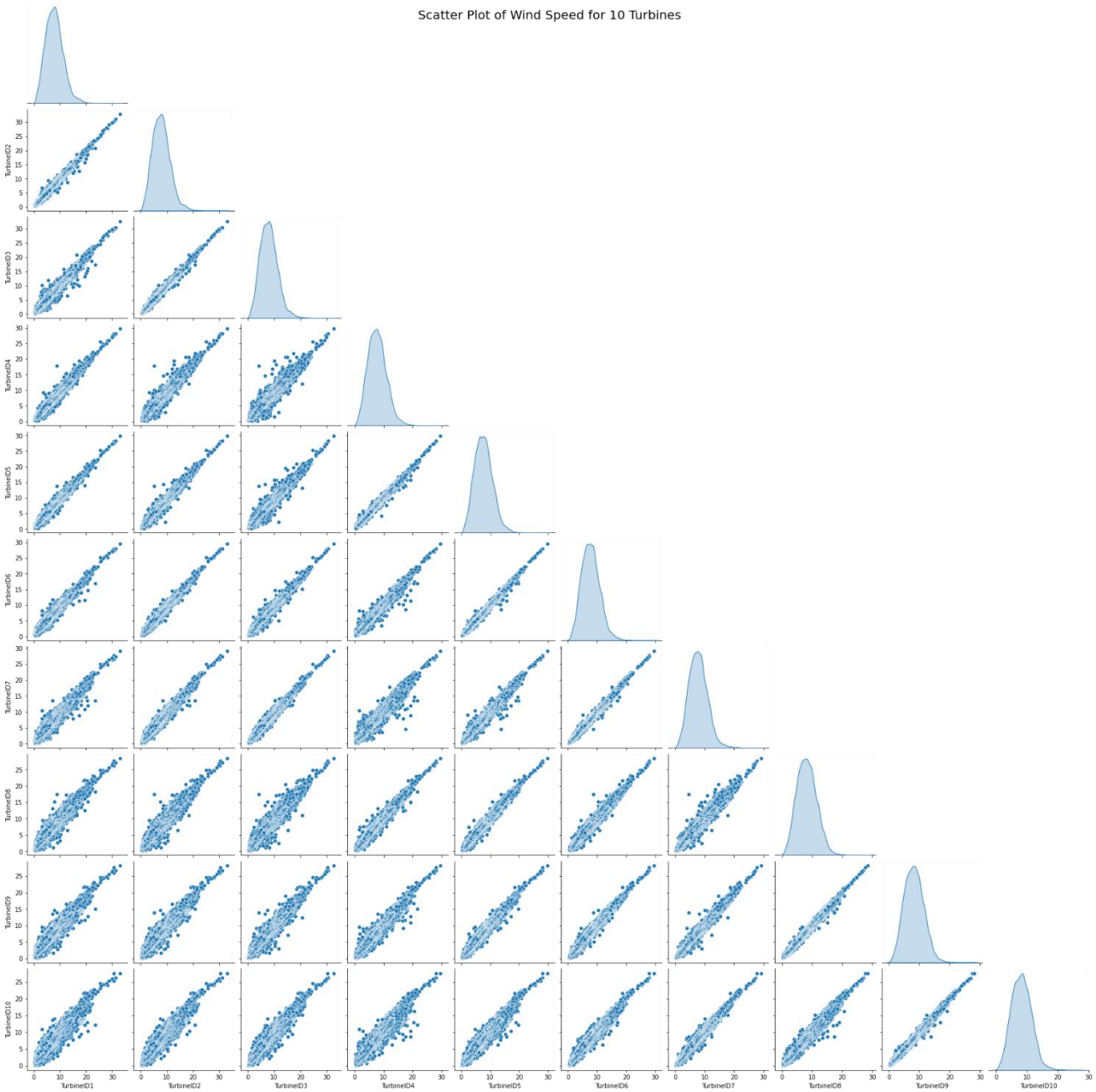


Figure 13. Correlation matrix of wind speed for 10 turbines

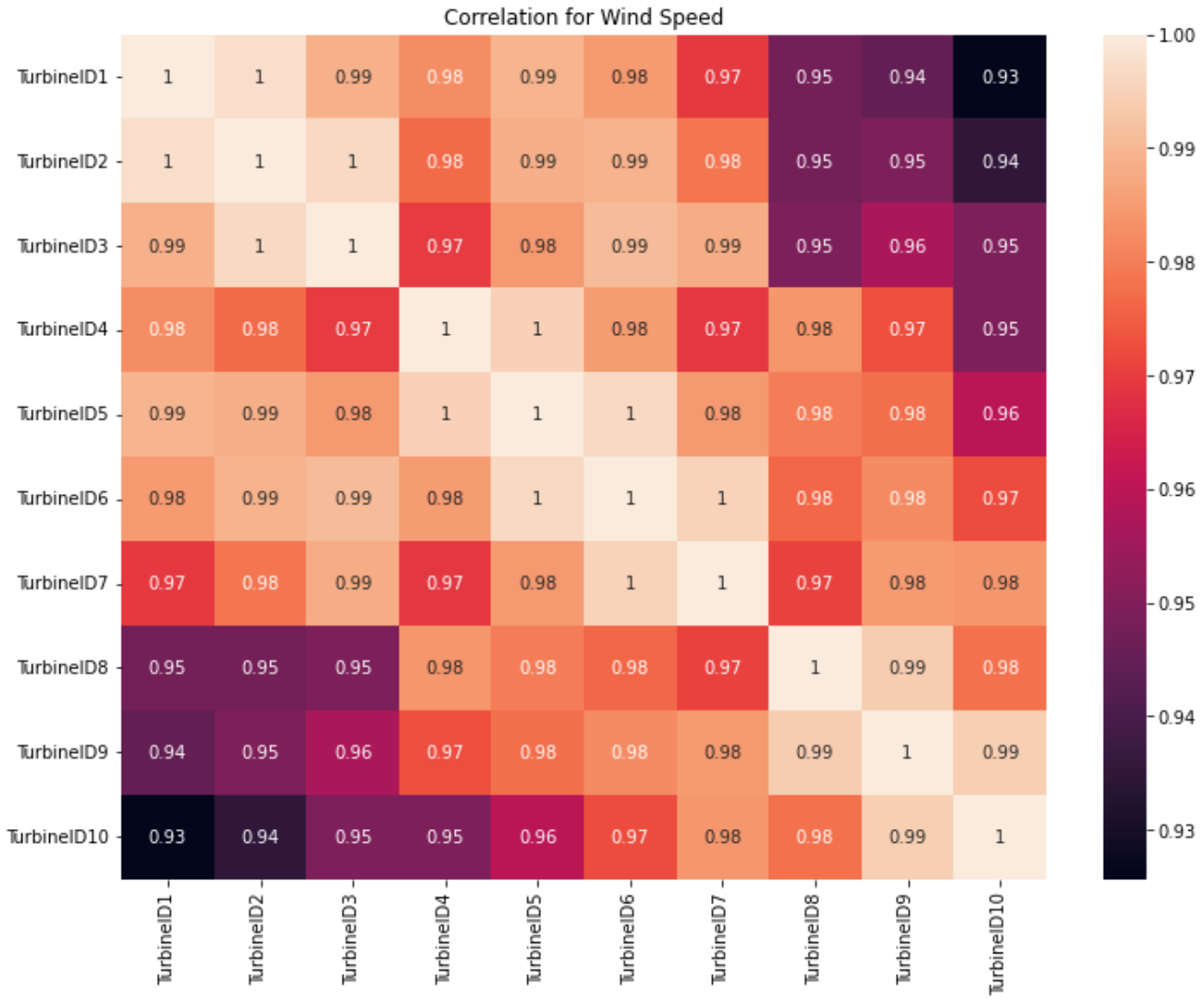


Figure 14. Scatter plot of output power for 10 turbines

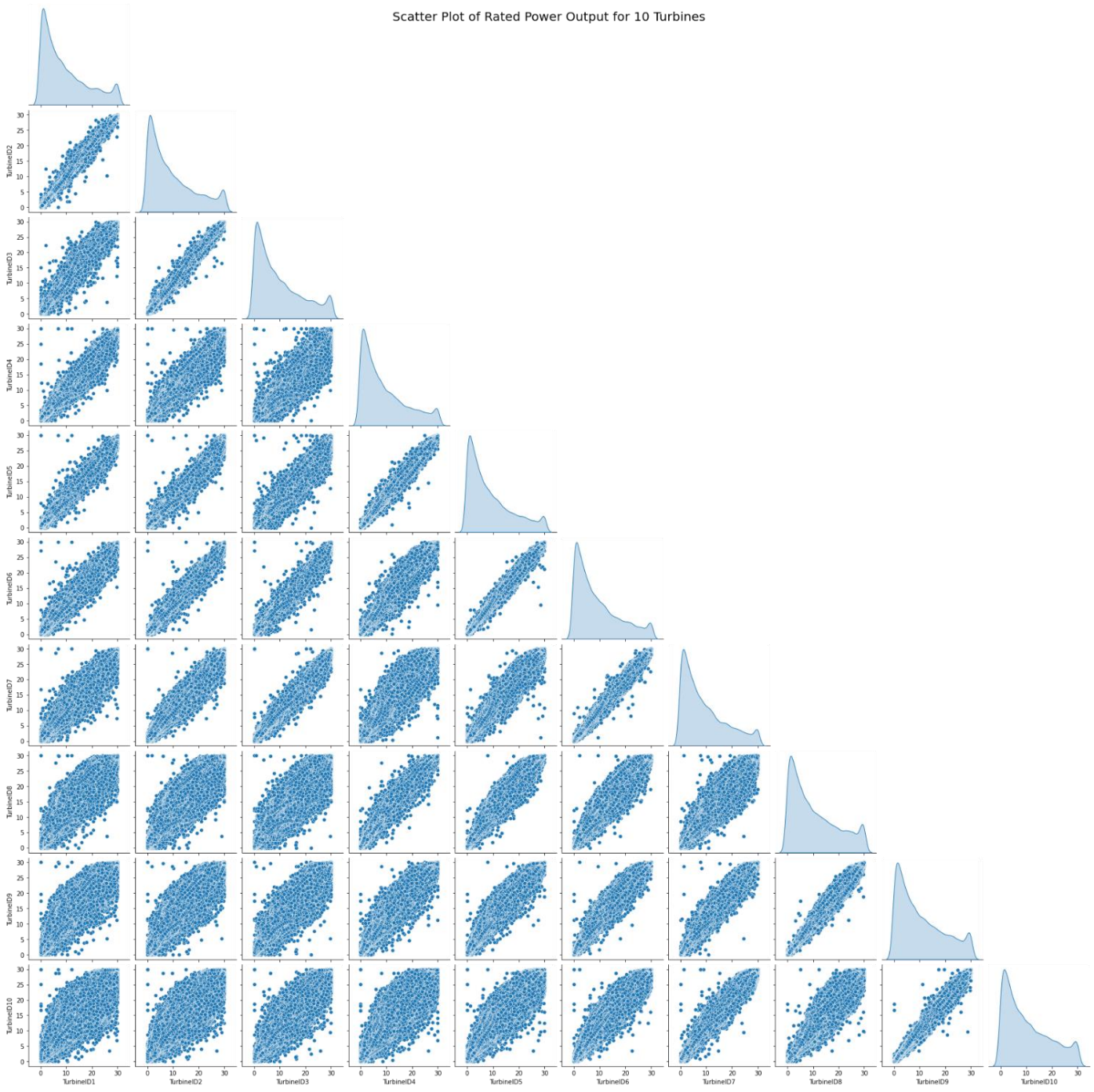
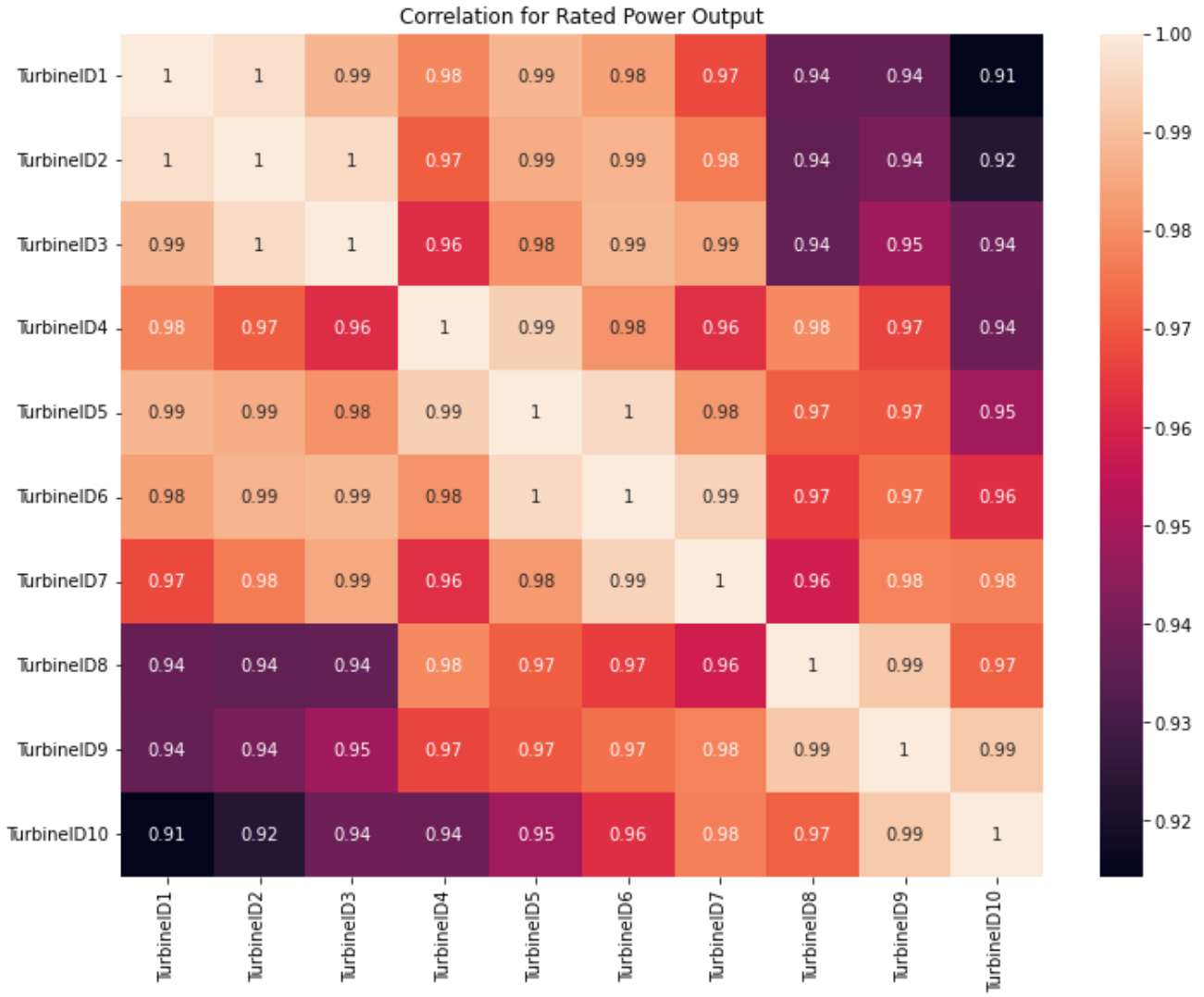


Figure 15. Correlation matrix of output power for 10 turbines



Moreover, to study the effect of spatial proximity on the correlation between turbines, I choose three turbines (ID 1, 11, 30510) from different wind farms that are marked on Fig. 16 with red, blue, and orange circles. Fig. 17 and 18 show the scatterplots and correlation matrix respectively. First, it can be seen that the wind speed values at the three turbines from different wind farms have weak correlations compared to the selected turbines from a single wind farm. Secondly, according to Fig. 18, turbines ID1 and ID11 have a Pearson coefficient of 0.43 which is considerably higher than the correlation coefficient between turbines ID1 and ID30510 or between turbines ID11 and 30510. The result could be due to the proximity of blue and red wind farms.

Figure 16. Location of all turbines in the dataset

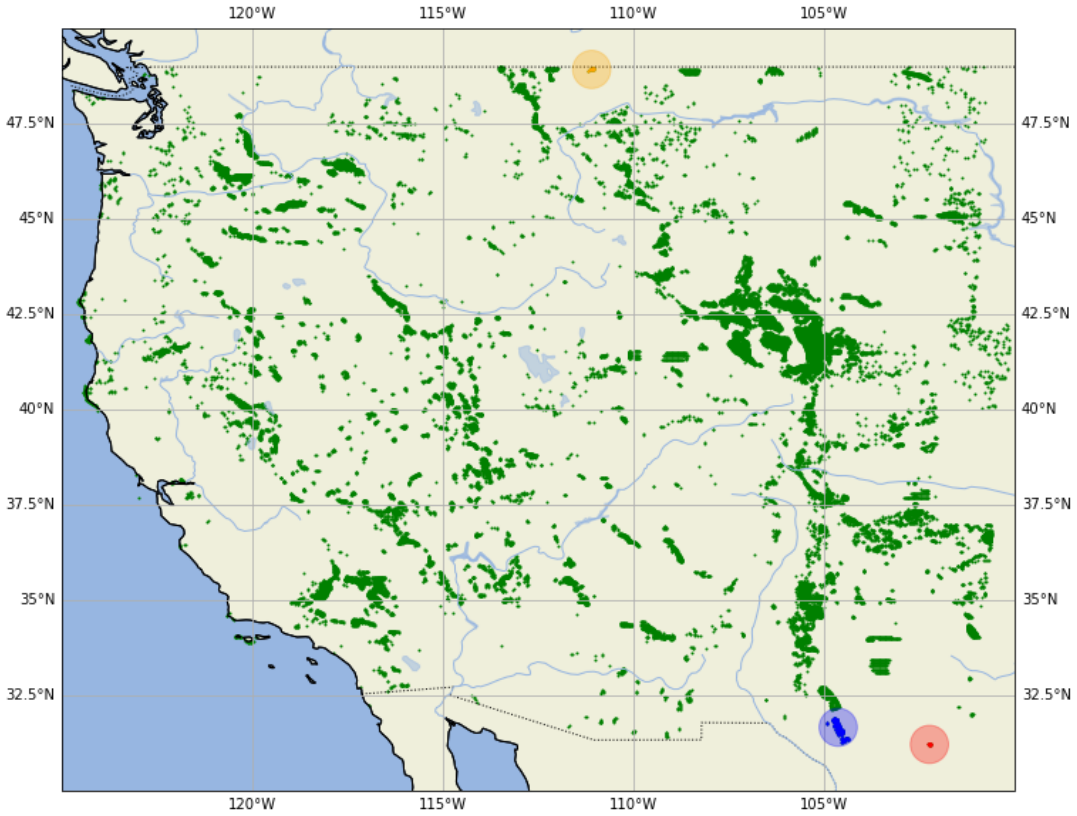


Figure 18. Scatter plot of wind speed for turbines ID 1, 11, 30510

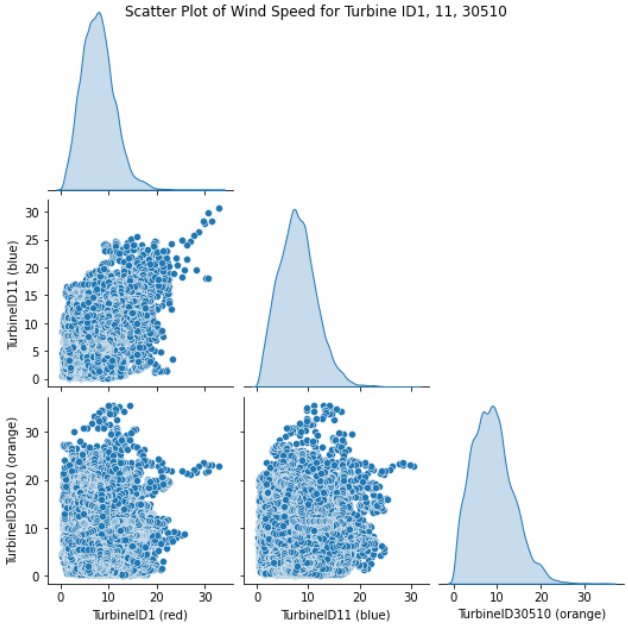
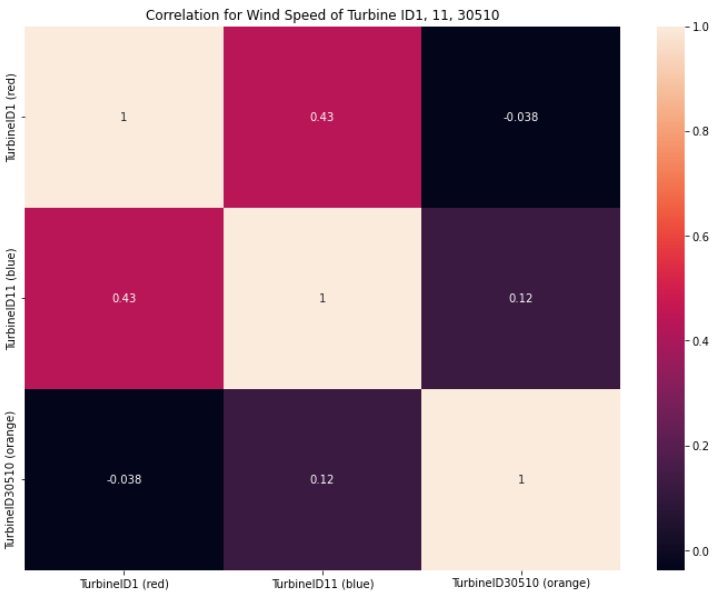


Figure 17. Correlation matrix of wind speed for turbines ID 1, 11, 30510



2.3.4 EOF Analysis

In this section of the report, the EOF analysis is implemented on wind velocity data for ten selected turbines using python library "eofs" [10]. First, the speed data for ten turbines are grouped by the location coordinates (latitude and longitude). Next, the two corresponding EOFs and PCs are calculated that are shown in tables 3 and 4. Fig. 19 demonstrates the results for EOF analysis. In the first row and the first column of the figure, we can see the time series of the average wind speed for the whole year. The second column shows the average wind speed for each turbine based on their location. The plots for calculated PCs and EOFs represent in the following rows. As it can be seen, the first mode of EOF captured 97.70% of the variance from the data while the remaining mode has a small contribution. The leading two EOFs/PCs explain together about 99.27% of the total variability. For the next step, I evaluate the ability of the EOF in reproducing the actual data. Fig. 20 shows the real wind speed data for turbine ID1 for January against the reconstructed data using two leading EOFs and PCs. Three evaluation indices, R-squared, RMSE, and MAE are calculated as metrics to measure the ability in reproducing data. The R^2 value, RMSE and MAE as an average on all turbines for the whole time period are 0.997, 0.132, and 0.206 respectively which indicates that first two EOFs and PCs shows great performance in reconstructing the real data.

Table 4. First two EOFs

	wind speed									
lat	31.192			31.208				31.225		
lon	-102.242	-102.225	-102.208	-102.258	-102.242	-102.225	-102.208	-102.242	-102.225	-102.208
EOF										
1	0.328645	0.328284	0.326031	0.312481	0.310651	0.309201	0.305441	0.317614	0.313950	0.308925
2	-0.435443	-0.398047	-0.285439	-0.053334	-0.090759	-0.038999	0.081072	0.363138	0.411105	0.500420

Table 3. First two PCs

	PC1	PC2
time		
2006-01-01 00:00:00	-8.260026	-0.160039
2006-01-01 00:10:00	-8.072653	-0.039084
2006-01-01 00:20:00	-7.834176	0.066072
2006-01-01 00:30:00	-7.589182	0.157604
2006-01-01 00:40:00	-7.246139	0.263593
...	...	...
2006-12-31 23:10:00	-5.931856	-1.042520
2006-12-31 23:20:00	-6.222377	-1.042768
2006-12-31 23:30:00	-6.706891	-0.970217
2006-12-31 23:40:00	-6.857650	-0.810155
2006-12-31 23:50:00	-6.771940	-0.624238

Figure 19. (First columns) Average wind speed and two leading PCs time series,

(Second column) Average wind speed and two leading EOFs based on location

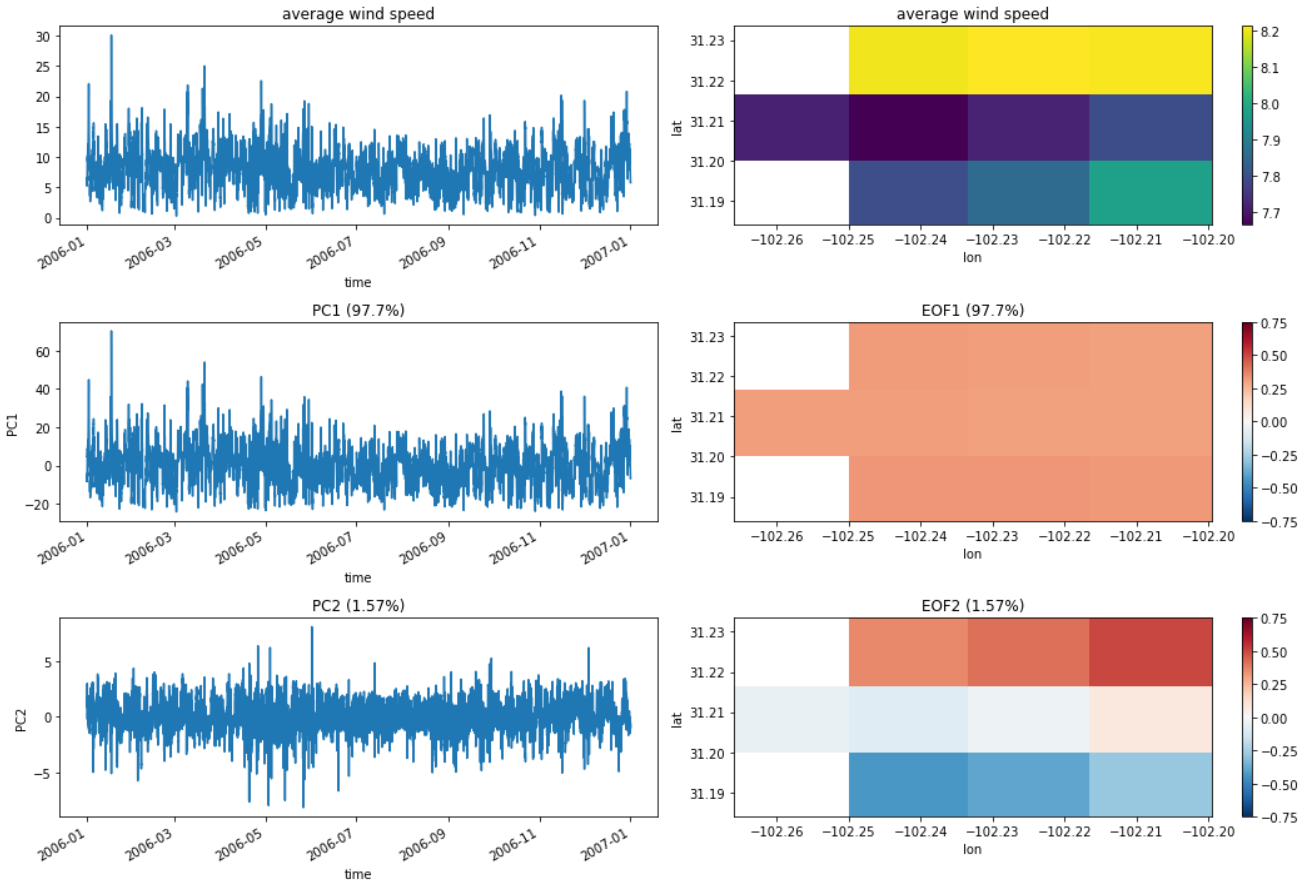
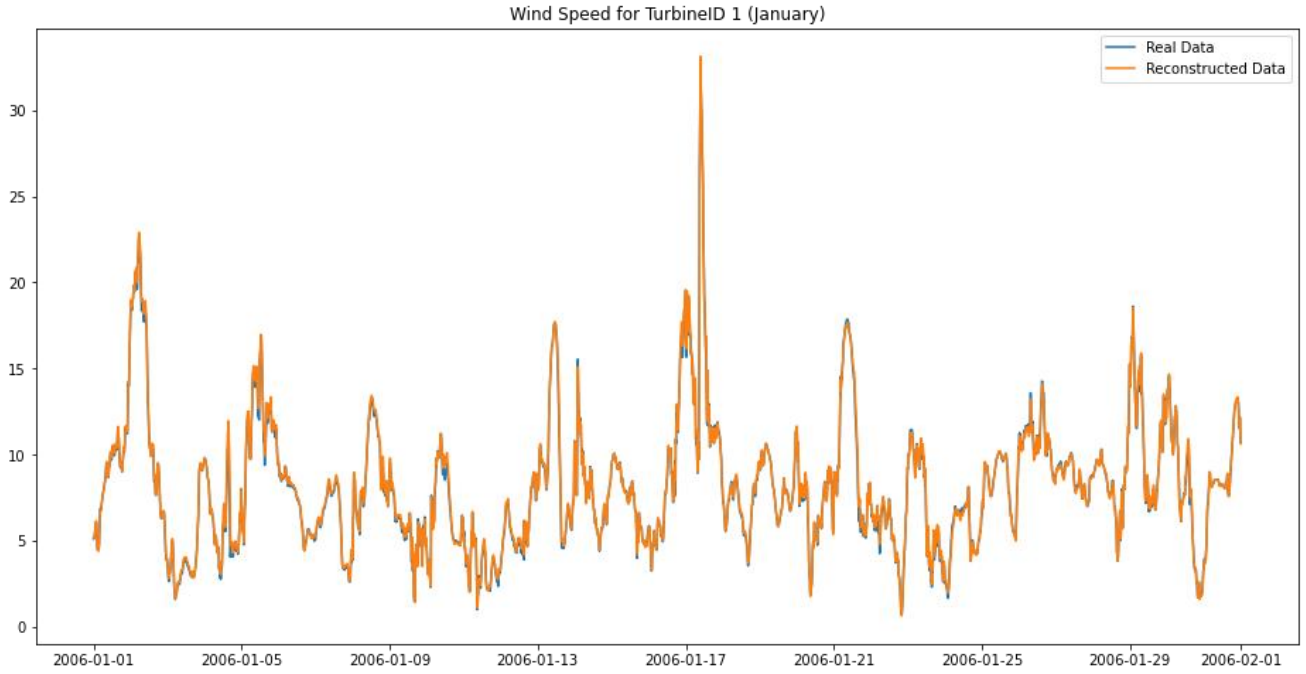


Figure 20. Wind speed time series of real data and reconstructed data for January



3. Conclusion and Outlook

This study focused on an exploratory data analysis of a small subset of the NREL dataset. I investigated the wind speed and output power time series for 10 turbines in a particular wind farm. Different visualization tools are used to get a better understanding of the data. In order to find seasonal patterns, the heat map of wind velocity is plotted at an hourly time scale for the whole year. Although we could not see any clear seasonal pattern from the heat map, some extracted information is explained. The histograms of wind speed and output power are presented and explained by the typical power curve of the turbine. In addition, distribution characteristics for 10 minutes and 1-hour increments of velocity and power are shown and discussed. Both distributions for velocity and output power have leptokurtic distributions which represent several outliers in the data. Moreover, the correlation between selected turbines and turbines from different wind farm are presented. As we expected, turbines selected from a wind farm are highly correlated in terms of wind speed and power because they are in close proximity. The results help to have a better understanding of the wind data and

Moreover, an EOF analysis is briefly introduced to find the temporal bases and spatial coefficient of wind speed data for selected turbines. The two leading EOFs and PCs explained 99.27% of the observed variance in the data and were able to accurately reconstruct the real data. Although the EOF analysis have been done on small part of the dataset, it is a promising approach to decompose the spatio-temporal data and ultimately predict climate data. For future work, we can focus on modeling the wind data using larger subset of the dataset and deploying machine learning techniques to achieve reasonable predictions.

4. References

[1] WindEurope, “Wind energy in Europe: 2021 Statistics and outlook for 2022-2026,” Feb. 2022. Accessed: Jun. 30, 2022. [Online]. Available: https://windeurope.org/intelligence-platform/product/wind-energy-in-europe-2021-statistics-and-the-outlook-for-2022-2026/#findings

[2] P. Milan, M. Wächter, and J. Peinke, “Turbulent character of wind energy,” *Phys Rev Lett*, vol. 110, no. 13, 2013, doi: 10.1103/PhysRevLett.110.138701.

[3] M. Lei, L. Shiyan, J. Chuanwen, L. Hongling, and Z. Yan, “A review on the forecasting of wind speed and generated power,” *Renewable and Sustainable Energy Reviews*, vol. 13, no. 4. 2009. doi: 10.1016/j.rser.2008.02.002.

[4] X. Wang, P. Guo, and X. Huang, “A review of wind power forecasting models,” in *Energy Procedia*, 2011, vol. 12. doi: 10.1016/j.egypro.2011.10.103.

[5] E. N. Lorenz, “Empirical Orthogonal Functions and Statistical Weather Prediction,” *Technical report Statistical Forecast Project Report 1 Department of Meteorology MIT 49*, vol. 1, no. Scientific Report No. 1, Statistical Forecasting Project. 1956.

[6] A. Hannachi, I. T. Jolliffe, and D. B. Stephenson, “Empirical orthogonal functions and related techniques in atmospheric science: A review,” *International Journal of Climatology*, vol. 27, no. 9. 2007. doi: 10.1002/joc.1499.

[7] A. Hannachi, *A primer for EOF analysis of climate data*. 2004.

[8] “NREL Western Wind Resources Dataset.” https://odysseus.informatik.uni-oldenburg.de/download/Data/NREL/2006/ (accessed Jun. 30, 2022).

[9] V. Sohoni, S. C. Gupta, and R. K. Nema, “A Critical Review on Wind Turbine Power Curve Modelling Techniques and Their Applications in Wind Based Energy Systems,” *Journal of Energy*, vol. 2016, 2016, doi: 10.1155/2016/8519785.

[10] A. Dawson, “eofs: A Library for EOF Analysis of Meteorological, Oceanographic, and Climate Data,” *J Open Res Softw*, vol. 4, no. 1, 2016, doi: 10.5334/jors.122.

5. Code

```
#Import libraries
import warnings # supress warnings
warnings.filterwarnings('ignore')

import glob
import numpy as np
import pandas as pd
import scipy.stats as stats
from scipy.stats import gaussian_kde
import math

import pyEOF
from pyEOF import *
import xarray as xr
from eofs.standard import Eof
from sklearn.preprocessing import StandardScaler

from statsmodels.tsa.seasonal import seasonal_decompose
from dateutil.parser import parse
from pylab import rcParams

from matplotlib import pyplot as plt
import matplotlib.colors as mc
from matplotlib.cm import ScalarMappable
import matplotlib.patches as mpatches
import seaborn as sns
import cartopy.crs as ccrs
import cartopy.feature as cfeature
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error



# Import data for all turbines
df from each file = [pd.read csv('https://odysseus.informatik.uni-
oldenburg.de/download/Data/NREL/2006/{}.csv'.format(i))
                     for i in range(1,11)]
data all = pd.concat(df from each file, ignore index=True)

# Import meta data
data_meta = pd.read_csv('https://odysseus.informatik.uni-
oldenburg.de/download/Data/NREL/site meta.csv',
                     header=0, index_col=0, parse_dates=True)


# Plot the location of all Turbines and selected wind farm

fig = plt.figure(figsize=(12, 10))
ax = plt.axes(projection=ccrs.PlateCarree())
ax.coastlines()
ax.set extent([-100, -125, 30, 50], ccrs.PlateCarree())
ax.add_feature(cfeature.LAND)
```

```
ax.add_feature(cfeature.OCEAN)
ax.add_feature(cfeature.COASTLINE)
ax.add_feature(cfeature.BORDERS, linestyle=':')
ax.add feature(cfeature.LAKES, alpha=0.5)
ax.add_feature(cfeature.RIVERS)
ax.gridlines(draw_labels=True)

m size = 1

lat = data_meta[' Latitude']
lon = data meta[' Longitude']

data_selected = data_meta[(data_meta[' Longitude']<-102) & (data_meta[' Longitude']>-103) &
                     (data meta[' Latitude']>31) & (data meta[' Latitude']<32)]


lat selected = data selected[' Latitude']
lon selected = data selected[' Longitude']

plt.scatter(lon,lat,color='green',s=m size)
plt.scatter(lon_selected ,lat_selected ,color='red',s=m_size)
ax.add patch(mpatches.Circle(xy=[np.mean(lon selected), np.mean(lat selected)],
                             radius=0.5, color='red', alpha=0.3, transform=ccrs.PlateCarree(), zorder=
30))

# Import the data for Turbine ID1
data1 = pd.read csv('https://odysseus.informatik.uni-oldenburg.de/download/Data/NREL/2006/1.csv',
                    header=0, index col=0, parse dates=True)
data1.head()
data1.info()

# Plot the wind speed time series for TurbineID1 - whole year
plt.figure(figsize=(16, 5))
data1["100m wind speed (m/s)"].plot()
plt.title('Wind Speed Time Series for TurbineID 1')
plt.ylabel('Wind Speed')
plt.xlabel('Time')


# Extract the data for January
data1_Jan = data1['2006-01-01':'2006-01-31']

# Plot the wind speed time series for Turbine ID1 - January
plt.figure(figsize=(16, 5))
data1_Jan["100m wind speed (m/s)"].plot()
plt.title('Wind Speed Time Series for TurbineID 1 (January)')
plt.ylabel('Wind Speed')
plt.xlabel('Date')

# Plot the output power time series for Turbine ID1 - January
plt.figure(figsize=(16, 5))
data1_Jan[" rated power output at 100m (MW)"].plot()
```

```
plt.title('Output Power Time Series for TurbineID 1 (January)')
plt.ylabel('Rated Power Output')
plt.xlabel('Date')


MIN_WS = data1['100m wind speed (m/s)'].min()
MAX WS = data1['100m wind speed (m/s)'].max()

# Plot heatmap for whole year 2006 for Turbine ID1
def single plot (data, month, ax):
    data = data[data.index.month == month]
    hour = data.index.hour
    day = data.index.day
    WindSpeed = data['100m wind speed (m/s)']

    d = {'hour':hour, 'day':day, 'windspeed':WindSpeed}
    df = pd.DataFrame (data=d)
    WindSpeed = df.pivot table(index="hour",columns="day",values="windspeed", aggfunc="mean")

    xgrid = np.arange(day.max() + 1) + 1
    ygrid = np.arange(25)

    ax.pcolormesh(xgrid, ygrid, WindSpeed, cmap="magma", vmin=MIN WS, vmax=MAX WS)
    # Invert the vertical axis
    ax.set ylim(24, 0)
    # Set tick positions for both axes
    ax.yaxis.set ticks([i for i in range(24)])
    ax.xaxis.set_ticks([10, 20, 30])
    # Remove ticks by setting their length to 0
    ax.yaxis.set tick params(length=0)
    ax.xaxis.set tick params(length=0)

    # Remove all spines
    ax.set frame on(False)


fig, axes = plt.subplots(1, 12, figsize=(14, 5), sharey=True)

for i in range(1,13):
    single_plot(data1, i, axes[i-1])

fig.subplots_adjust(left=0.05, right=0.98, top=0.9, hspace=0.08, wspace=0.04)


fig.subplots_adjust(bottom=0.15)

cbar_ax = fig.add_axes([0.3, 0.005, 0.4, 0.025])

# Create a normalizer that goes from minimum to maximum wind speed
norm = mc.Normalize(MIN_WS, MAX_WS)

# Create the colorbar and set it to horizontal
```

```
cb = fig.colorbar(
    ScalarMappable(norm=norm, cmap="magma"),
    cax=cbar_ax, # Pass the new axis
    orientation = "horizontal"
)

# Remove tick marks
cb.ax.xaxis.set tick params(size=0)

# Set legend label
cb.set label("Wind Speed", size=12)
#fig

# Set common labels for x and y axes
fig.text(0.5, 0.08, "Day", ha="center", va="center", fontsize=14)
fig.text(0.02, 0.5, 'Hour', ha="center", va="center", rotation="vertical", fontsize=14)

fig.suptitle("Hourly Wind Speed (2006) - Turbine 1", fontsize=20, y=0.97)
#fig


# Plot the wind speed histogram for Turbine ID1
sns.displot(data1 ["100m wind speed (m/s)"], kde=True, bins = 40)

plt.title('Histogram of Wind Speed for TurbineID1')
plt.xlabel('Wind speed at 100m [m/s]')
plt.ylabel('Count')
plt.grid(color = 'green', linestyle = '--', linewidth = 0.5)

from scipy.stats import kurtosis
kur = kurtosis(data1 ["100m wind speed (m/s)"])
print ('Kurtosis =', round(kur,3))


# Plotting the output power histogram for turbine 1
sns.displot(data1 [" rated power output at 100m (MW)"],kde=True,bins = 20)

plt.title('Histogram of the Rated Power Output for TurbineID1')
plt.xlabel('Rated power output at 100m (MW)')
plt.ylabel('Count')
plt.grid(color = 'green', linestyle = '--', linewidth = 0.5)


# Distribution of wind speed for 10 min increment Turbine ID1
increment 10 wind = list()
for i in range (len(data1)-1):
    increment_10_wind.append (data1['100m wind speed (m/s)'][i]-data1['100m wind speed (m/s)'][i+1])


# Plot Normal distribution vs pdf of increment
std 10 wind = np.std(increment 10 wind)
kur 10 wind = kurtosis(increment 10 wind, fisher=False)#/(std 10 wind**4)
print ('Standard Deviation =', round(std 10 wind,3))
print ('Kurtosis =', round(kur_10_wind,3))
```

```
mu = 0
sigma_10_wind = std_10_wind

plt.figure(figsize=(6, 4))

density 10 wind = gaussian kde(increment 10 wind)
x=np.linspace(-5,5,100)
plt.plot(x,density_10_wind(x))
plt.plot(x, stats.norm.pdf(x, mu, sigma_10_wind))

plt.title('pdf of the wind speed (incerement 10 min) vs Normal distribution')
plt.grid(color = 'green', linestyle = '--', linewidth = 0.5)
plt.legend(['Increment 10 min', 'Normal'])


# Distribution of wind speed for 1h increment Turbine ID1
increment 60 wind = list()
for i in range (len(data1)-6):
    increment_60_wind.append (data1['100m wind speed (m/s)'][i]-data1['100m wind speed (m/s)'][i+6])


# Plot Normal distribution vs pdf of increment
std 60 wind = np.std(increment 60 wind)
kur_60_wind = kurtosis(increment_60_wind, fisher=False)#/(std_60_wind**4)
print ('Standard Deviation =', round(std 60 wind,3))
print ('Kurtosis =', round(kur 60 wind,3))

mu = 0
sigma 60 wind = std 60 wind

plt.figure(figsize=(6, 4))

density 60 wind = gaussian kde(increment 60 wind)
x=np.linspace(-5,5,100)
plt.plot(x,density_60_wind(x))
plt.plot(x, stats.norm.pdf(x, mu, sigma_60_wind))

plt.title('pdf of the wind speed (incerement 1 hour) vs Normal distribution')
plt.grid(color = 'green', linestyle = '--', linewidth = 0.5)
plt.legend(['Increment 1 hour', 'Normal'])


# Distribution of output power for 10 min increment Turbine ID1
increment 10 power = list()
for i in range (len(data1)-1):
    increment 10 power.append (data1[" rated power output at 100m (MW)"][i]-
data1[" rated power output at 100m (MW)"][i+1])


# Plot Normal distribution vs pdf of increment
std 10 power = np.std(increment 10 power)
kur 10 power = kurtosis(increment 10 power,fisher=False)#/(std 10 power**4)
print ('Standard Deviation =', round(std_10_power,3))
```

```
print ('Kurtosis =', round(kur_10_power,3))

mu = 0
sigma 10 power = std 10 power

plt.figure(figsize=(6, 4))

density 10 power = gaussian kde(increment 10 power)
x=np.linspace(-5,5,100)
plt.plot(x,density_10_power(x))
plt.plot(x, stats.norm.pdf(x, mu, sigma 10 power))

plt.title('pdf of the output power (incerement 10 min) vs Normal distribution')
plt.grid(color = 'green', linestyle = '--', linewidth = 0.5)
plt.legend(['Increment 10 min', 'Normal'])


# Distribution of the output power for 1h increment Turbine ID1
increment 60 power = list()
for i in range (len(data1)-6):
    increment 60 power.append (data1[" rated power output at 100m (MW)"][i]-
data1[" rated power output at 100m (MW)"][i+6])


# Plot Normal distribution vs pdf of increment

std 60 power = np.std(increment 60 power)
kur 60 power = kurtosis(increment 60 power, fisher=False)#/(std 60 power**4)
print ('Standard Deviation =', round(std_60_power,3))
print ('Kurtosis =', round(kur 60 power,3))

mu = 0
sigma_60_power = std_60_power

plt.figure(figsize=(6, 4))

density_60_power = gaussian_kde(increment_60_power)
x=np.linspace(-5,5,100)
plt.plot(x, density_60_power(x))
plt.plot(x, stats.norm.pdf(x, mu, sigma_60_power))

plt.title('pdf of the output power (incerement 1 hour) vs Normal distribution')
plt.grid(color = 'green', linestyle = '--', linewidth = 0.5)
plt.legend(['Increment 1 hour', 'Normal'])


# Plotting histogram of wind speed for all turbines
sns.displot(data all ["100m wind speed (m/s)"], kde=True, bins = 40)

plt.title('Histogram of Wind Speed for All Turbines')
plt.xlabel('Wind speed at 100m [m/s]')
plt.ylabel('Count')
plt.grid(color = 'green', linestyle = '--', linewidth = 0.5)
```

```
# Plotting histogram of output power for all turbines
sns.displot(data all [" rated power output at 100m (MW)"], kde=True, bins = 20)

plt.title('Histogram of Rated Power Output for All Turbines')
plt.xlabel('Rated power output at 100m (MW)')
plt.ylabel('Count')
plt.grid(color = 'green', linestyle = '--', linewidth = 0.5)


# Construct Dataframe for Wind Speed of all Turbines
k = len(data1)
WindSpeed dic = {}
for i in range(10):
    WindSpeed dic['TurbineID{}'.format(i+1)] = list(data all['100m wind speed (m/s)'][i*k:(i+1)*k])

WindSpeed df = pd.DataFrame(data=WindSpeed dic)
#WindSpeed_df


# Construct Dataframe for Rated Power Output of all Turbines
k = len(data1)
PowerOut dic = {}
for i in range(10):
    PowerOut dic['TurbineID{}'.format(i+1)] = list(data all[' rated power output at 100m (MW)'][i*k:(i
+1)*k])

PowerOut_df = pd.DataFrame(data=PowerOut_dic)
#PowerOut_df


g = sns.pairplot(WindSpeed_df, diag_kind='kde', corner=True)
g.fig.suptitle('Scatter Plot of Wind Speed for 10 Turbines', fontsize = 20)


g = sns.pairplot(PowerOut df, diag kind='kde', corner=True)
g.fig.suptitle('Scatter Plot of Rated Power Output for 10 Turbines', fontsize = 20)

# Correlation matrix for Wind Speed
corrMatrix WS = WindSpeed df.corr()

# Plotting correlation matrix for wind speed
plt.figure(figsize=(12, 9))
sns.heatmap(corrMatrix WS, annot=True)
plt.title('Correlation for Wind Speed')


# Correlation matrix for rated output power
corrMatrix PO = PowerOut df.corr()

# Plotting correlation matrix for wind speed
plt.figure(figsize=(12, 9))
sns.heatmap(corrMatrix_PO, annot=True)
```

```
plt.title ('Correlation for Rated Power Output')


fig = plt.figure(figsize=(12, 10))
ax = plt.axes(projection=ccrs.PlateCarree())
ax.coastlines()
ax.set extent([-100, -125, 30, 50], ccrs.PlateCarree())
ax.add feature(cfeature.LAND)
ax.add_feature(cfeature.OCEAN)
ax.add feature(cfeature.COASTLINE)
ax.add feature(cfeature.BORDERS, linestyle=':')
ax.add feature(cfeature.LAKES, alpha=0.5)
ax.add feature(cfeature.RIVERS)
ax.gridlines(draw labels=True)

m size = 1

lat = data meta[' Latitude']
lon = data meta[' Longitude']

data selected1 = data meta[(data meta[' Longitude']>-103) & (data meta[' Latitude']<31.5)]
lat selected1 = data selected1[' Latitude']
lon selected1 = data selected1[' Longitude']

data selected2 = data meta[(data meta[' Longitude']<-104) & (data meta[' Longitude']>-105) &
                     (data meta[' Latitude']>31) & (data meta[' Latitude']<32)]
lat selected2 = data selected2[' Latitude']
lon_selected2 = data_selected2[' Longitude']

data selected3 = data meta[(data meta[' Longitude']<-110) & (data meta[' Longitude']>-111.5) &
                     (data meta[' Latitude']>48) & (data meta[' Latitude']<49)]
lat_selected3 = data_selected3[' Latitude']
lon selected3 = data selected3[' Longitude']

plt.scatter(lon, lat, color='green', s=m size)
plt.scatter(lon_selected1, lat_selected1, color='red', s=m_size)
plt.scatter(lon selected2, lat selected2, color='blue', s=m size)
plt.scatter(lon selected3, lat selected3, color='orange', s=m size)


ax.add_patch(mpatches.Circle(xy=[np.mean(lon_selected1), np.mean(lat_selected1)],
                             radius=0.5, color='red', alpha=0.3, transform=ccrs.PlateCarree(), zorder=
30))
ax.add_patch(mpatches.Circle(xy=[np.mean(lon_selected2), np.mean(lat_selected2)],
                             radius=0.5, color='blue', alpha=0.3, transform=ccrs.PlateCarree(), zorder
=30))
ax.add_patch(mpatches.Circle(xy=[np.mean(lon_selected3), np.mean(lat_selected3)],
                             radius=0.5, color='orange', alpha=0.3, transform=ccrs.PlateCarree(), zord
er=30))


data11 = pd.read csv('https://odysseus.informatik.uni-oldenburg.de/download/Data/NREL/2006/11.csv',
                  header=0, index_col=0, parse_dates=True)
```

```
data30510 = pd.read_csv('https://odysseus.informatik.uni-
oldenburg.de/download/Data/NREL/2006/30510.csv',
                  header=0, index_col=0, parse_dates=True)


WindSpeed_dic_compare = {}

WindSpeed dic compare['TurbineID{} (red)'.format(1)] = list(data1['100m wind speed (m/s)'])
WindSpeed_dic_compare['TurbineID{} (blue)'.format(11)] = list(data11['100m wind speed (m/s)'])
WindSpeed dic compare['TurbineID{} (orange)'.format(30510)] = list(data30510['100m wind speed (m/s)'])

WindSpeed_df_compare = pd.DataFrame(data=WindSpeed_dic_compare)


g = sns.pairplot(WindSpeed df compare, diag kind='kde', corner=True)
g.fig.suptitle('Scatter Plot of Wind Speed for Turbine ID1, 11, 30510', fontsize = 12)


# Correlation matrix for Wind Speed
corrMatrix WS compare = WindSpeed df compare.corr()

# Plotting correlation matrix for wind speed
plt.figure(figsize=(12, 9))
sns.heatmap(corrMatrix WS compare, annot=True)
plt.title('Correlation for Wind Speed of Turbine ID1, 11, 30510')

data all c = pd.DataFrame(columns=['time','wind speed','lat','lon'])
data_all_c['time'] = data_all['Date(YYYY-MM-DD hh:mm:ss)']
data_all_c['wind speed'] = data_all['100m wind speed (m/s)']

k = len(data1)
for i, t in enumerate(range(1,11)):
    data_all_c['lat'][i*k:(i+1)*k] = float(data_meta[data_meta.index == t][' Latitude'])
    data all c['lon'][i*k:(i+1)*k] = float(data meta[data meta.index == t][' Longitude'])

data_all_c['time'] =  pd.to_datetime(data_all_c['time'])
data_all_c
array_data = pd.DataFrame.to_numpy(df_data)


df data = pyEOF.get time space(data all c, time dim = "time", lumped space dims = ["lat","lon"])
df_data


# Implement Rotated EOF (REOF)
n = 2
pca = pyEOF.df_eof(df_data, pca_type="varimax", n_components=n, scaler=True)

eofs = pca.eofs(s=0, n=n) # get eofs
eofs da = eofs.stack(["lat","lon"]).to xarray() # make it convenient for visualization
pcs = pca.pcs(s=0, n=n) # get pcs
evfs = pca.evf(n=n) # get variance fraction

# create a function for visualization convenience
```

```
def visualization(da, pcs, eofs_da, evf):
    fig = plt.figure(figsize = (15,10))

    ax = fig.add subplot(n+1,2,1)
    da.mean(dim=["lat","lon"]).plot(ax=ax)
    ax.set_title("average wind speed")

    ax = fig.add subplot(n+1,2,2)
    da.mean(dim="time").plot(ax=ax)
    ax.set_title("average wind speed")

    for i in range(1,n+1):
        pc_i = pcs["PC"+str(i)].to_xarray()
        eof i = eofs da.sel(EOF=i)["wind speed"]
        frac = str(np.array(evf[i-1]*100).round(2))

        ax = fig.add subplot(n+1,2,i*2+1)
        pc i.plot(ax=ax)
        ax.set title("PC"+str(i)+" ("+frac+"%)")

        ax = fig.add subplot(n+1,2,i*2+2)
        eof i.plot(ax=ax,
                   vmin=-0.75, vmax=0.75, cmap="RdBu r",
                   cbar_kwargs={'label': ""})
        ax.set title("EOF"+str(i)+" ("+frac+"%)")

    plt.tight layout()
    plt.show()


# EOF with eofs library
solver = Eof(array_data)

s pcs = pd.DataFrame(data=solver.pcs(npcs=2, pcscaling=0),columns = pcs.columns,
                     index = pcs.index)


s eofs = pd.DataFrame(data = solver.eofs(neofs=2, eofscaling=0),columns = eofs.columns,
                     index = eofs.index)
s_eofs_da = s_eofs.stack(["lat","lon"]).to_xarray() # make it convenient for visualization

s evfs = solver.varianceFraction(neigs=2)

# plot
visualization(da, s_pcs, s_eofs_da, s_evfs)


# Reconstructing data for wind speed based on eofs and pcs
reconstructed data =  pd.DataFrame(np.dot(s pcs,s eofs))
mean = WindSpeed df.mean()
for i in range(10):
    reconstructed data[i] += mean[i]
```

```
#reconstructed_data

# Plotting reconstructed data for TurbineID 1 (January)

plt.figure(figsize=(16, 8))
plt.plot(data1_Jan.index, list(data1_Jan['100m wind speed (m/s)']))
plt.plot(data1 Jan.index, list(reconstructed data[0][:len(data1 Jan)]))
plt.legend(['Real Data', 'Reconstructed Data'])
plt.title('Wind Speed for TurbineID 1 (January)')

r2 score = r2 score(list(data1 Jan['100m wind speed (m/s)']), list(reconstructed data[0][:len(data1 Ja
n)]))
mae = mean_absolute_error(list(data1_Jan['100m wind speed (m/s)']), list(reconstructed_data[0][:len(da
ta1 Jan)]))
mse = mean squared error(list(data1 Jan['100m wind speed (m/s)']), list(reconstructed data[0][:len(dat
a1 Jan)]))
rmse = mse ** 0.5

print("R2 Score =", r2 score)
print("MAE =", mae)
print("RMSE =", rmse)
```