1 INTRODUCTION

1.1 Overview:

This project focussed on predicting the energy output of a wind farm based on weather condition.In this project we will be seeing the time series graph predicted energy output of a wind farm,an application to recommend the power grid and also graph model is provided for all the possible comparison with the given data set.

1.2 Purpose:

Our main motive of this project is to predict the accurate energy output of the wind turbine with the given data set. By predicting the energy output with the accuracy of 80 percentage, it will help us to predict the loss of energy more accurately and efficiently.

2 LITERATURE SURVEY

2.1 Existing problem:

Wind energy plays an increasing role in the supply of energy world-wide. The energy output of a wind farm is dependent on the wind conditions present at its site. If the output can be predicted more accurately, energy suppliers can coordinate the collaborative production of different energy sources more efficiently to avoid costly overproduction.Renewable energy, such as wind and solar energy, plays an increasing role in the supply of energy world wide. This trend will continue because global energy demand is increasing, and the use of nuclear power and traditional sources of energy such as coal and oil is either considered unsafe or leads to a large amount of CO2 emission. Wind energy is a key player in renewable energy. The capacity of wind energy production has been substantially increased during the last years. In particular, wind speed is crucial for energy production based on wind, and it may vary drastically over time. Energy suppliers are interested in accurate predictions, as they can avoid overproduction by coordinating the collaborative production of traditional power plants and weather-dependent energy sources.

2.2 Proposed solution:

Our aim is to map weather data to energy production. We wish to show that even data that is publicly available for weather stations close to wind farms can be used to give a good prediction of the energy output. Furthermore, we examine the impact of different weather conditions on the energy output of wind farms. We are particularly, interested in the correlation of different components that characterize weather conditions such as wind speed, pressure, and temperature. Statistical approaches use historical data to predict the wind speed on an hourly basis or to predict energy output directly. Short term prediction is often done based on meteorological data, and learning approaches are applied. Neural networks are a very popular learning approach for wind power forecasting based on given time series.

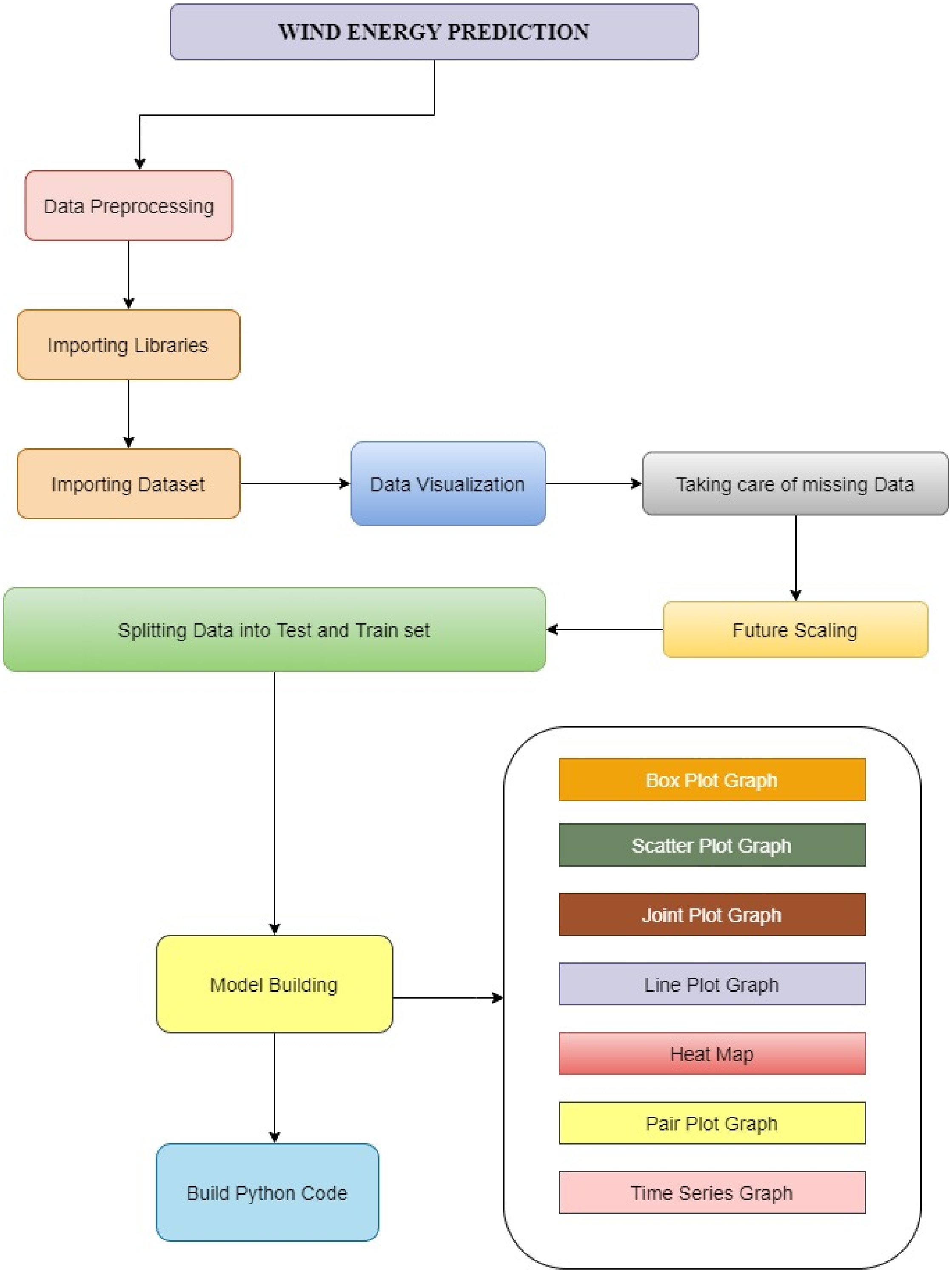
We have satisfied the above given criteria by creating a time series model to predict the power output of the wind farm by comparing LV Active power, Theoritical power, wind direction and Date/Time. we created a comparison graph models for each and every given data with seconds, minutes,hours, day and year. We have approached this problem in an modernized way which is by creating an web application. This web application is helpful for predicting the energy output for the specific input given by the user and also equivalent time series model will be displayed for better understanding.

3 THEORETICAL ANALYSIS

3.1 Website Designing:

Frontend is one of the important criteria for project display. In this project we have created our website using HTML, CSS and by collaging our machine learning using FLASK.

3.2 Flow Chart:



4 EXPERIMENTAL INVESTIGATIONS

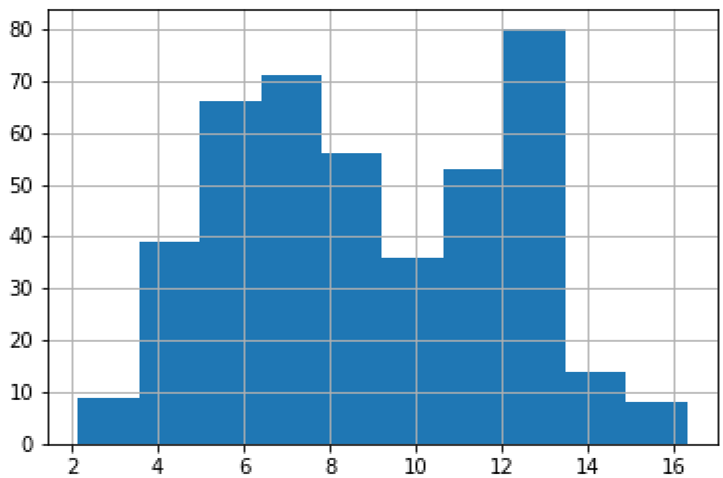
The project was first initiated by importing the libraries such as numpy, pandas, matplot, seaborn, pickle, etc..The second step is to import the dataset which is given in the problem statement. Once the dataset were imported it is splitted into dependent and independent variables and it is used for testing and training of dataset. Splitted dataset were used for Model building , Model evaluation and for plotting various graph models.

5 RESULT

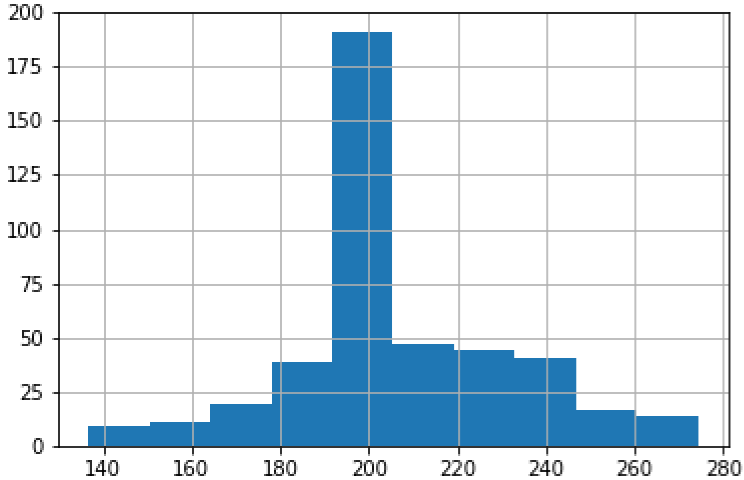
As a result of this the energy prediction according to weather condition for 72 hours is predicted successfully. Using the given dataset, Time series model were plotted for 72 hours.

6 GRAPH AND ITS EXPLANATIONS

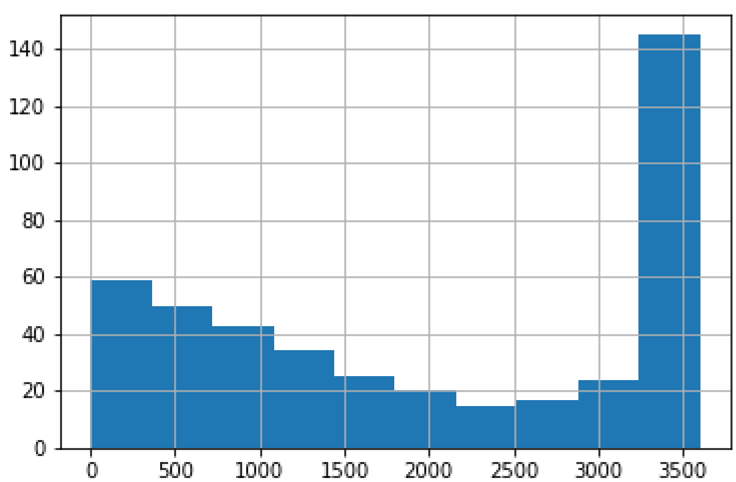
6.1 Data Visualization of Wind speed:



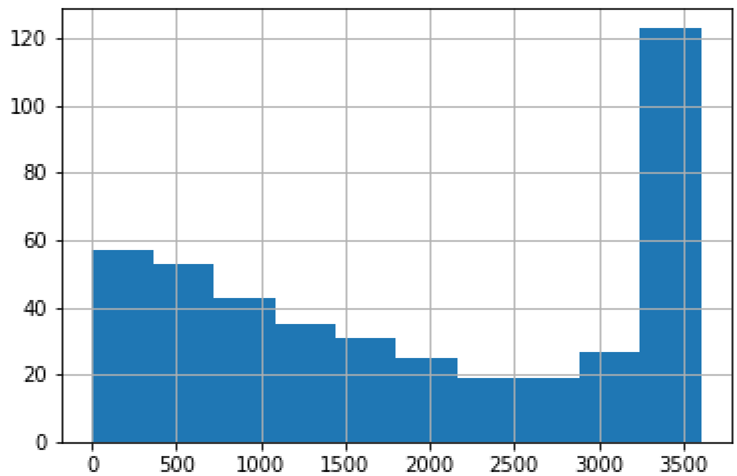
6.2 Data Visualization of Wind direction:



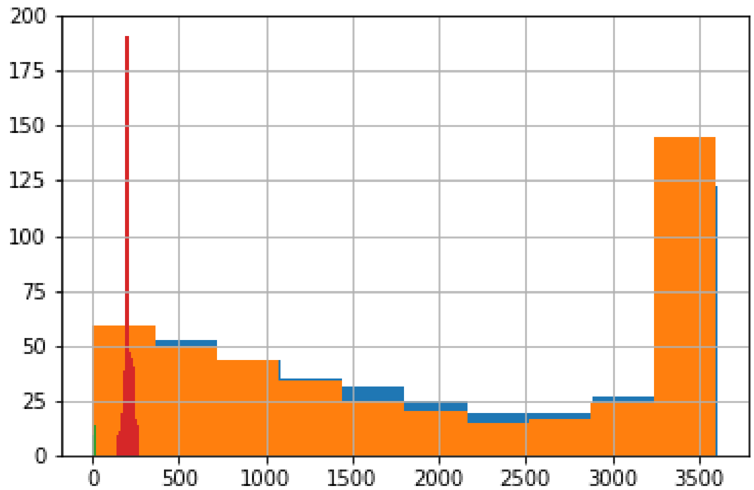
6.3 Data Visualization of Theoretical power:



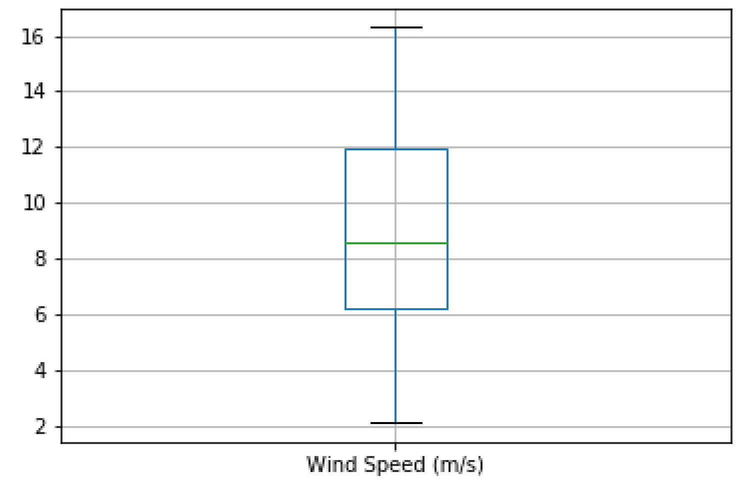
6.4 Data Visualization of LV Active power:



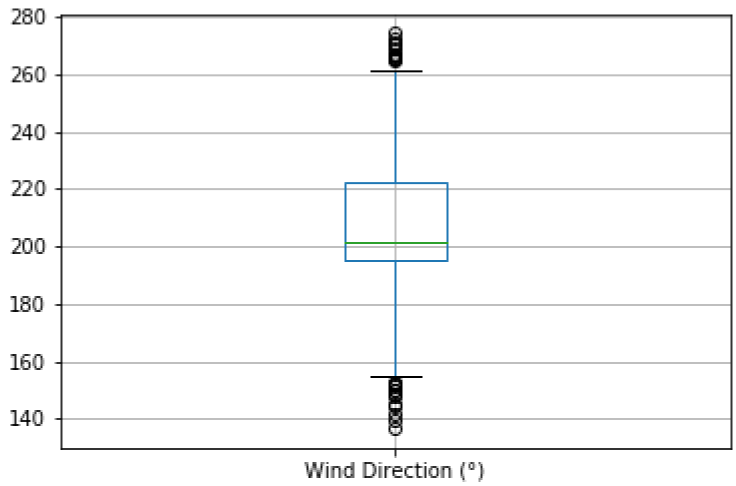
6.5 Overall Data Visualization:



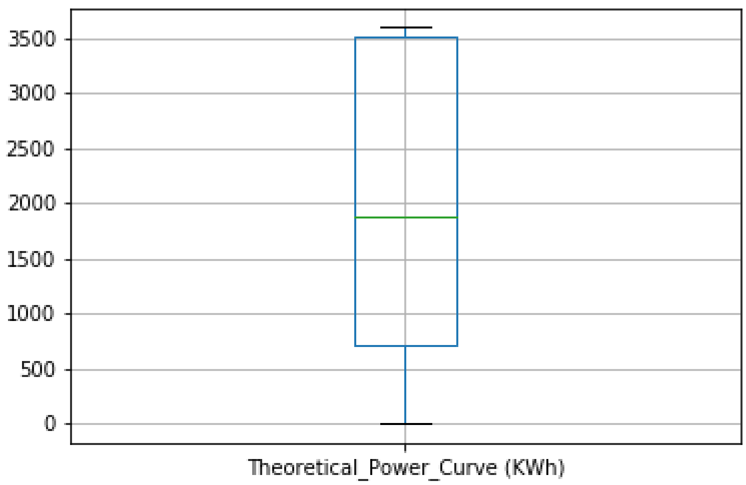
6.6 Box plot of Wind speed :



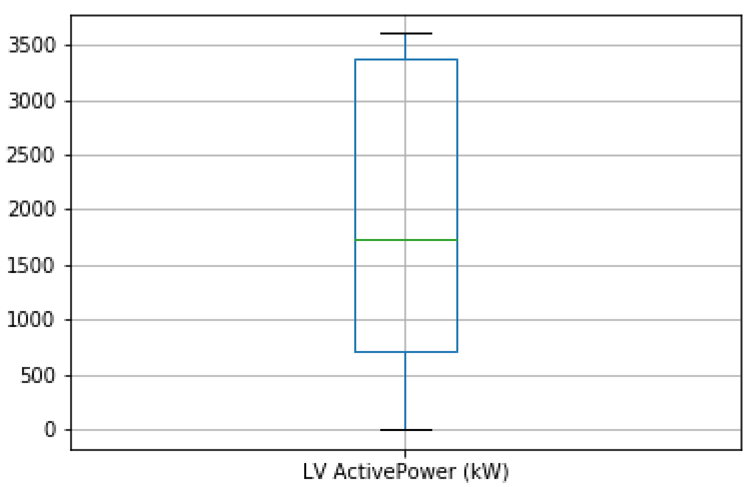
6.7 Box plot of Wind direction:



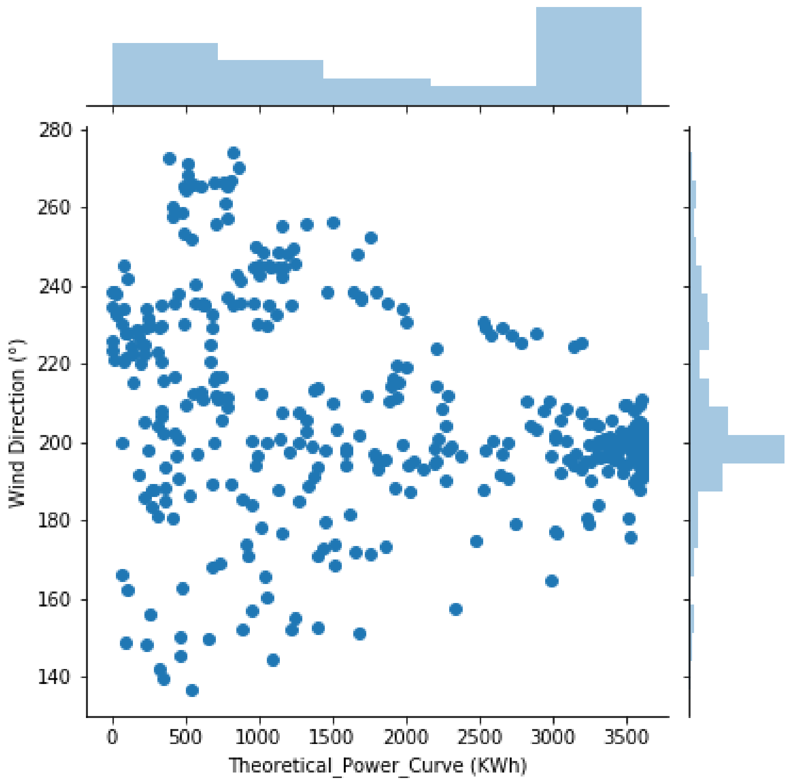
6.8 Box plot of Theoretical power:



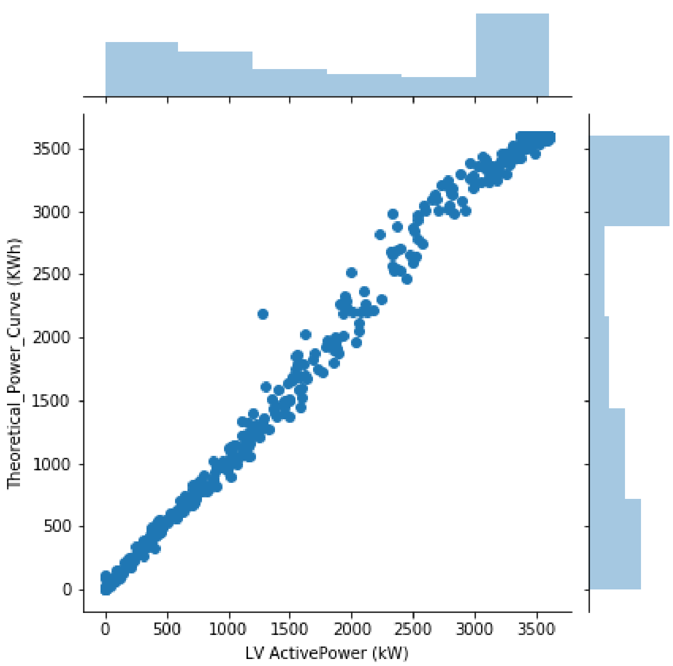
6.9 Box plot of LV Active power:



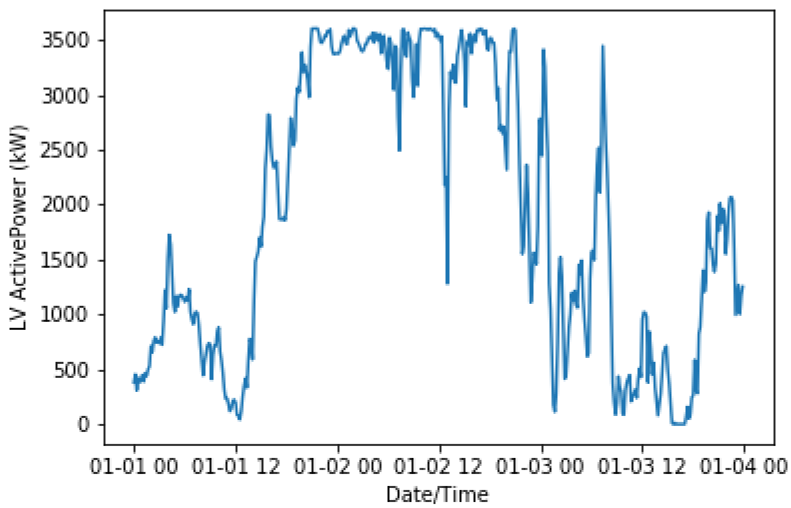
6.10 Joint plot of Theoretical power and Wind direction:



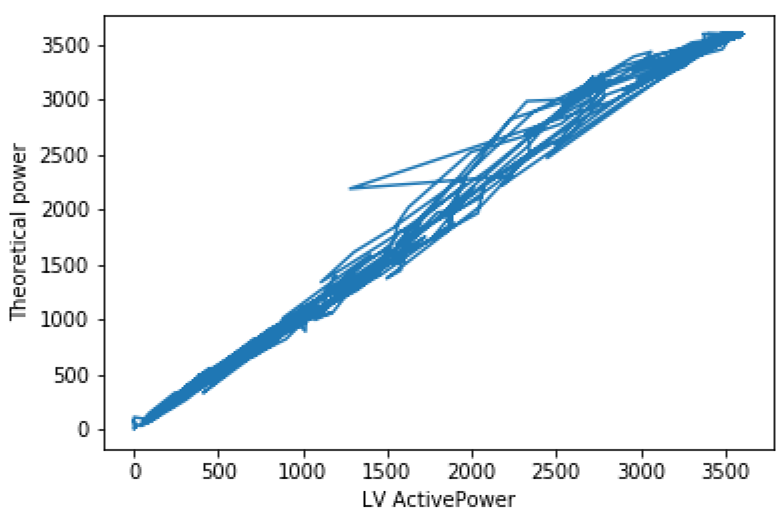
6.11 Joint plot of LV Active power and Theoretical power:



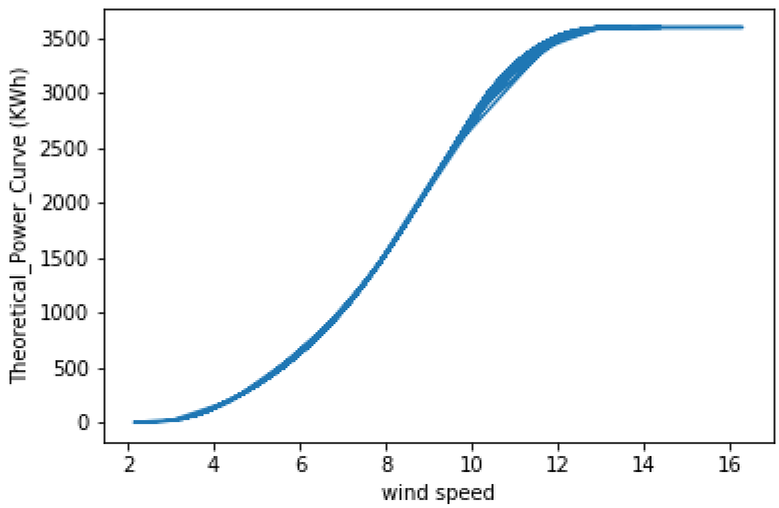
6.12 Line plot Date/Time and LV Active power:



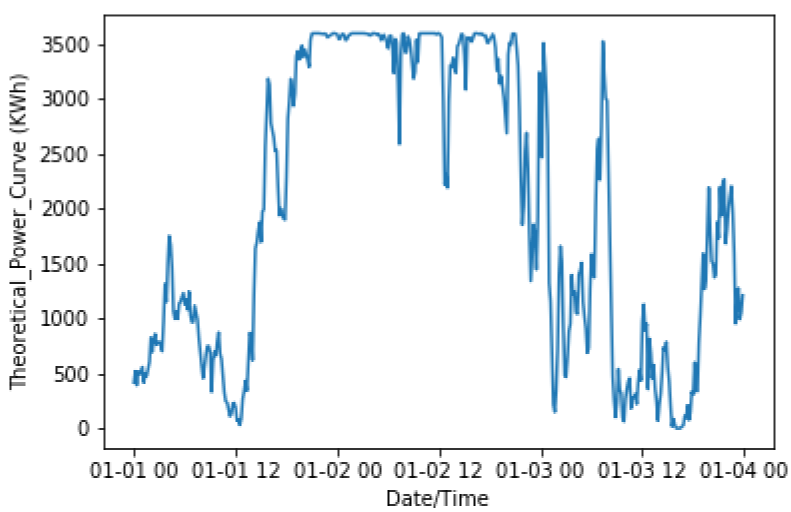
6.13 Line plot LV Active power and Theoretical power:



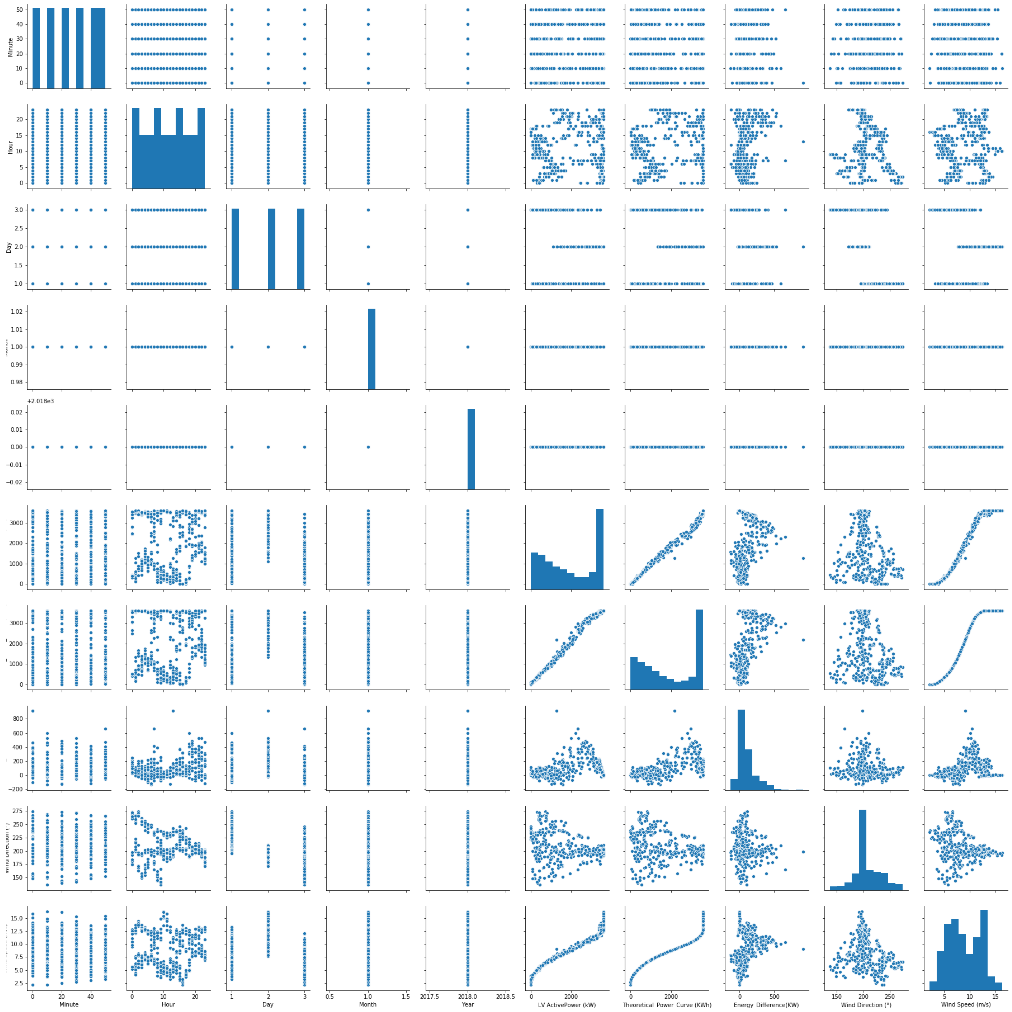
6.14 Line plot of Theoretical power and Wind speed:



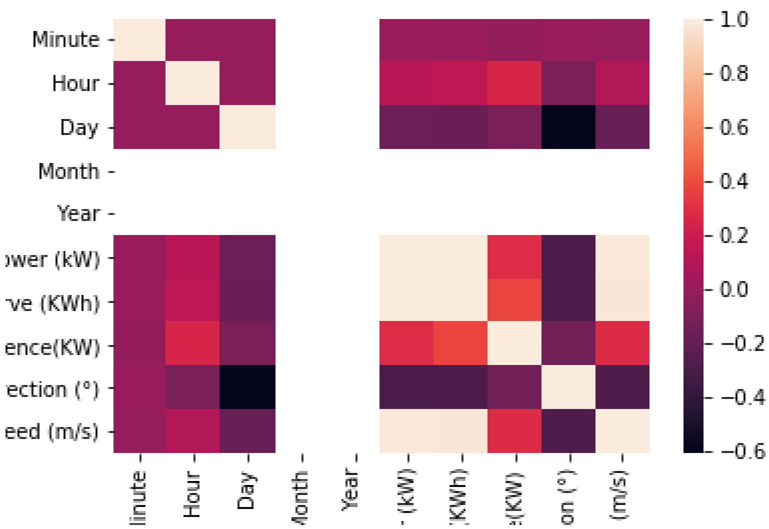
6.15 Line plot of Date/time and Theoretical power:



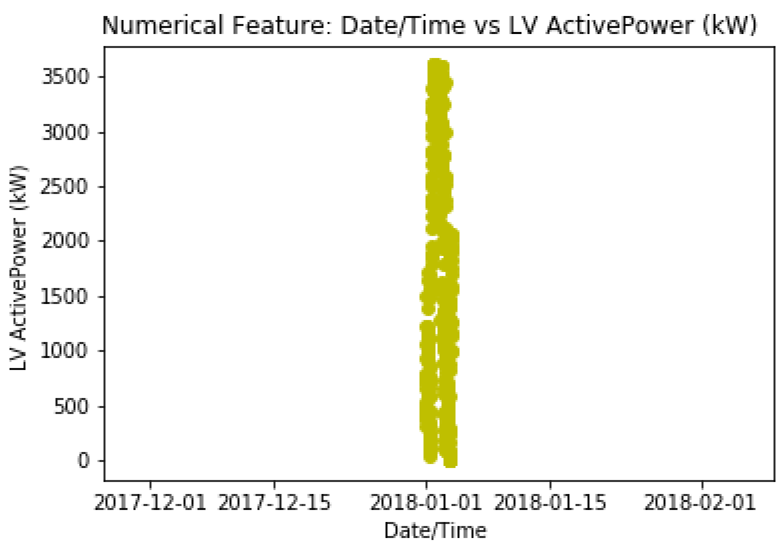
6.16 Pair plot:



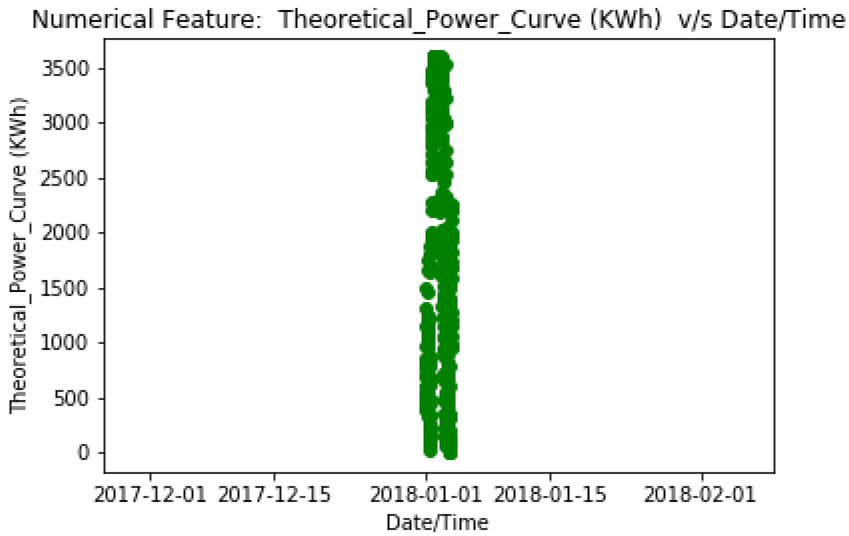
6.17 Heat map:



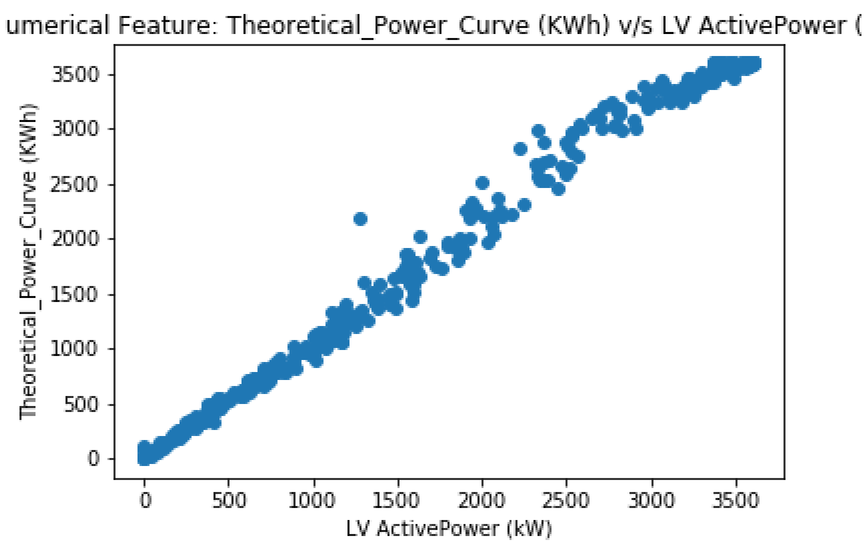
6.18 Scatter plot of Date/time and LV Active power:



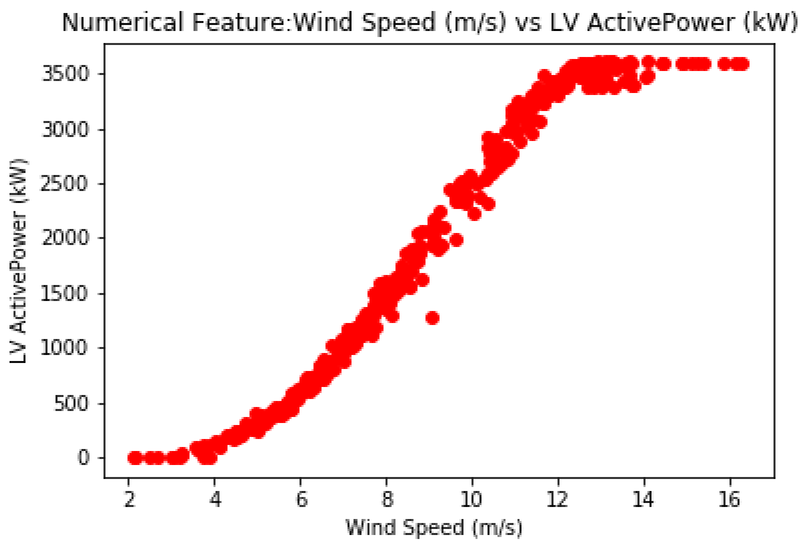
6.19 Scatter plot of Date/time and Theoretical power:



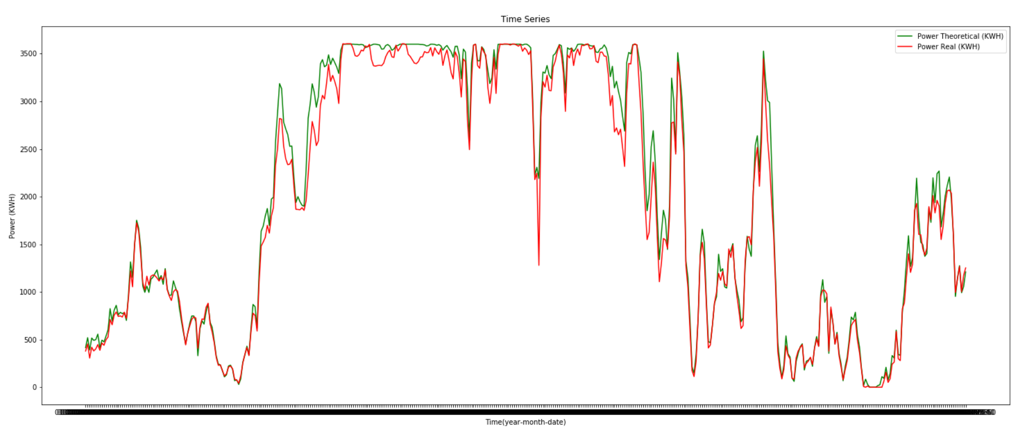
6.20 Scatter plot of LV Active power and Theoretical power:



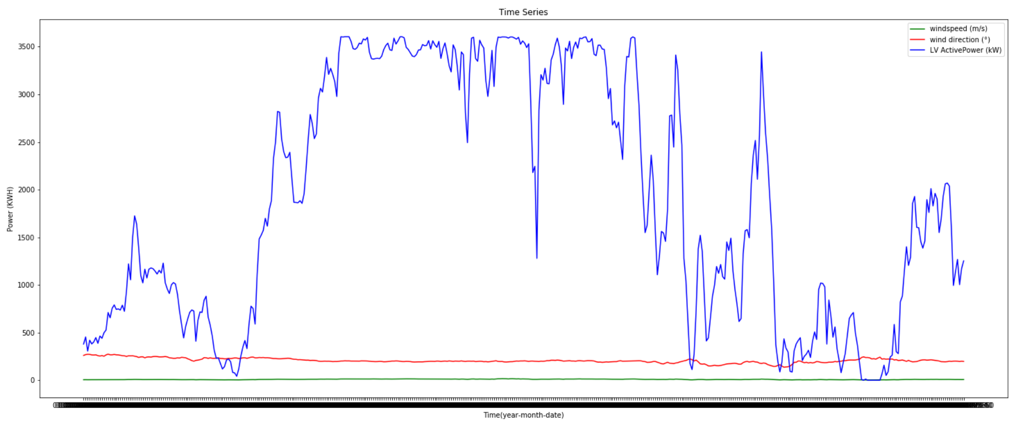
6.21 Scatter plot of LV Active power and Wind speed:



6.22 Time series (main)



6.23 Time series 2



7 CONCLUSION

The goal of this thesis was to conclude upon the following question:can the methods from Machine learning improve our ability to predict energy generation from wind parks over older methods.

8 BIBILOGRAPHY

1.https://hpi.de/friedrich/docs/paper/RE1.pdf

2. https://www.kaggle.com/berkerisen/wind-turbine-scada-dataset

APPENDIX

A. Source code:

#importing libraries:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pickle
from sklearn import preprocessing, svm
import statsmodels.tsa.api as smt
import statsmodels.api as sm
```

#importing Dataset:

```
data = pd.read_csv('T2.csv')
```

#Data visualization:

```
 data['LV ActivePower (kW)'].hist(bins=10)
plt.savefig("C:/Users/SKT/Desktop/IBM 2020/graphs/datavizLVactive.png")
data['Theoretical_Power_Curve (KWh)'].hist(bins=10)
plt.savefig("C:/Users/SKT/Desktop/IBM 2020/graphs/datavizTHEO.png")
data['Wind Speed (m/s)'].hist(bins=10)
plt.savefig("C:/Users/SKT/Desktop/IBM 2020/graphs/datavizwindspeed.png")
data['Wind Direction (°)'].hist(bins=10)
plt.savefig("C:/Users/SKT/Desktop/IBM 2020/graphs/datavizwinddirec.png")

data['LV ActivePower (kW)'].hist(bins=10)
```

```
data['Theoretical_Power_Curve (KWh)'].hist(bins=10)
data['Wind Speed (m/s)'].hist(bins=10)
data['Wind Direction (°)'].hist(bins=10)
plt.savefig("C:/Users/SKT/Desktop/IBM 2020/graphs/dataviz of all.png")
```

#Box plot:

```
x=data.iloc[0:,2:3].values
y=data.iloc[0:,1:2].values

data.boxplot(column="Theoretical_Power_Curve (KWh)")
plt.savefig("C:/Users/SKT/Desktop/IBM 2020/graphs/boxplotTHEOR.png")
data.boxplot(column="LV ActivePower (kW)")
plt.savefig("C:/Users/SKT/Desktop/IBM 2020/graphs/boxplotLVPower.png")
data.boxplot(column="Wind Speed (m/s)")
plt.savefig("C:/Users/SKT/Desktop/IBM 2020/graphs/boxplotWindSpeed.png")
data.boxplot(column="Wind Direction (°)")
plt.savefig("C:/Users/SKT/Desktop/IBM 2020/graphs/boxplotWindDirection.png")
```

#Time series graph(main):

```
x=data.iloc[0:,0:1].values
y=data.iloc[0:,1:3].values

data.index = data['Date/Time'] # indexing the Datetime to get the time period on the x-axis.
ts_theoretical = data['Theoretical_Power_Curve (KWh)']
ts_real = data['LV ActivePower (kW)']
plt.figure(figsize=(25,10))
```

```
plt.plot(ts_theoretical, label='Power Theoretical (KWH)',color='green')
plt.plot(ts_real,label='Power Real (KWH)',color='red')
plt.title('Time Series')
plt.xlabel("Time(year-month-date)")
plt.ylabel("Power (KWH)")
plt.legend(loc='best')
plt.savefig("C:/Users/SKT/Desktop/IBM 2020/graphs/timeseries1.png")
```

#Time series graph2:

```
x=data.iloc[0:,0:1].values
y=data.iloc[0:,2:].values

data.index = data['Date/Time'] # indexing the Datetime to get the time period on the x-axis.
ts_windspeed = data['Wind Speed (m/s)']
ts_winddirection = data['Wind Direction (°)']
ts_real = data['LV ActivePower (kW)']
plt.figure(figsize=(25,10))
plt.plot(ts_windspeed, label='windspeed (m/s)',color='green')
plt.plot(ts_winddirection,label='wind direction (°)',color='red')
plt.plot(ts_real,label='LV ActivePower (kW)',color='blue')
plt.title('Time Series')
plt.xlabel("Time(year-month-date)")
plt.ylabel("Power (KWH)")
plt.legend(loc='best')
plt.savefig("C:/Users/SKT/Desktop/IBM 2020/graphs/timeseries2.png")
```

#Pairplot :

```
data['Date/Time'] = pd.to_datetime(data['Date/Time'],format='%d %m %Y %H:%M')
data['Hour'] = data['Date/Time'].dt.hour
data['Minute'] = data['Date/Time'].dt.minute
data['Day'] = data['Date/Time'].dt.day
data['Month'] = data['Date/Time'].dt.month
data['Year'] = data['Date/Time'].dt.year
data.head()
data['Energy_Difference(KW)'] = data['Theoretical_Power_Curve (KWh)']-data['LV ActivePower (kW)']
data['Energy_Difference(KW)'].head(5)
data = data.reindex(columns=['Minute','Hour','Day', 'Month','Year','Date/Time', 'LV ActivePower (kW)',
        'Theoretical_Power_Curve (KWh)','Energy_Difference(KW)', 'Wind Direction (°)','Wind Speed (m/s)'])
data.head()
sns.pairplot(data)
plt.savefig("C:/Users/SKT/Desktop/IBM 2020/graphs/pairplot.png")
```

#Heatmap:

```
sns.heatmap(data.corr())
plt.savefig("C:/Users/SKT/Desktop/IBM 2020/graphs/heatmap.png")
```

#Scatterplot:

```
plt.scatter(data['LV ActivePower (kW)'],data['Theoretical_Power_Curve (KWh)'])
plt.title("Numerical Feature: Theoretical_Power_Curve (KWh)
```

```
v/s LV ActivePower (kW)")
plt.xlabel("LV ActivePower (kW)")
plt.ylabel("Theoretical_Power_Curve (KWh)")
plt.savefig("C:/Users/SKT/Desktop/IBM 2020/graphs/Scatterplot LV vs TH.png")

plt.scatter(data['Date/Time'],data['Theoretical_Power_Curve (KWh)'],color='g')
plt.title("Numerical Feature:  Theoretical_Power_Curve (KWh)  v/s Date/Time")
plt.xlabel("Date/Time")
plt.ylabel("Theoretical_Power_Curve (KWh)")
plt.savefig("C:/Users/SKT/Desktop/IBM 2020/graphs/Scatterplot dt vs thpower.png")

plt.scatter(data['Date/Time'],data['LV ActivePower (kW)'],color='y')
plt.title("Numerical Feature: Date/Time vs LV ActivePower (kW)  ")
plt.xlabel("Date/Time ")
plt.ylabel("LV ActivePower (kW)  ")
plt.savefig("C:/Users/SKT/Desktop/IBM 2020/graphs/Scatterplot dt vs LV.png")

plt.scatter(data['Wind Speed (m/s)'],data['LV ActivePower (kW)'],color='r')
plt.title("Numerical Feature:Wind Speed (m/s) vs LV ActivePower (kW)  ")
plt.xlabel("Wind Speed (m/s)  ")
plt.ylabel("LV ActivePower (kW)")
plt.savefig("C:/Users/SKT/Desktop/IBM 2020/graphs/Scatterplot windspeed vs LV.png")
```

#Jointplot:

```
sns.jointplot(data['LV ActivePower (kW)'],data['Theoretical_Power_Curve (KWh)'])
plt.savefig("C:/Users/SKT/Desktop/IBM 2020/graphs/joint LV vs THEO.png")
sns.jointplot(data['Theoretical_Power_Curve (KWh)'],data['Wind Direction (°)'])
plt.savefig("C:/Users/SKT/Desktop/IBM 2020/graphs/joint THEO vs WindDirection.png")
```

#Lineplot:

```
plt.plot(data['Date/Time'],data['Theoretical_Power_Curve (KWh)'])
plt.xlabel("Date/Time")
plt.ylabel("Theoretical_Power_Curve (KWh)")
plt.savefig("C:/Users/SKT/Desktop/IBM 2020/graphs/lineplot dt vs theo.png")
plt.plot(data['Date/Time'],data['LV ActivePower (kW)'])
plt.xlabel("Date/Time")
plt.ylabel("LV ActivePower (kW)")
plt.savefig("C:/Users/SKT/Desktop/IBM 2020/graphs/lineplot dt vs LV.png")
plt.plot(data['Wind Speed (m/s)'],data['Theoretical_Power_Curve (KWh)'])
plt.xlabel("wind speed")
plt.ylabel("Theoretical_Power_Curve (KWh)")
plt.savefig("C:/Users/SKT/Desktop/IBM 2020/graphs/lineplot wind vs theo.png")
```

```
plt.plot(data['LV ActivePower (kW)'],data['Theoretical_Power_Curve (KWh)'])
plt.xlabel("LV ActivePower")
plt.ylabel("Theoretical power")
plt.savefig("C:/Users/SKT/Desktop/IBM 2020/graphs/lineplot LV vs theo.png")
```

 # Model Building:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
x_train
y_train
x_test
y_test
plt.scatter(x[:,1],y)
plt.scatter(x[:,2],y)
plt.scatter(x[:,0],y)

from sklearn.linear_model import LinearRegression
mr=LinearRegression()
mr.fit(x_train,y_train)
pickle.dump(mr,open('decision2.pkl','wb'))
y_pred=mr.predict(x_test)
y_pred
y_test
```

#Model Evaluation:

```
from sklearn.metrics import r2_score
r2_score(y_pred,y_test)
data.head()
```

```
data.describe()
data.info()
```