ARIMA (Autoregression Integrated Moving Average)

An ARIMA model is a class of statistical models for the analysing and forecasting time series data. It is a generation of an autoregression moving averages model.

AR(Autoregression):A model that uses the dependent relation between an observation and some number of lagged observations. P is a parameter of how many lagged observations to be taken in.

I(Integrated): A model that uses the differencing of raw observations.(e.g. subtracting an observations from the previous time steps). Differencing in statistics is a transformation applied to time series data in order to make it stationary. This allows the properties do not depend on the time of observation, eliminating trend and seasonality and stabilizing the mean of the time series.

MA(Moving Average): A model that uses the dependency between an observation and a residual error from a moving average model applied to lagged observations. Q is a parameter of how many lagged observations to be taken in.

Parameters of the ARIMA model:

p (lag order): number of lag observations include in the model.

d (degree of differencing): number of times that the raw observations are differenced.

q (order of moving average): size of the moving average window.

ARIMA model in words:

Prediction Y(t) = Constant + Linear combination Lags of Y[upto p lags] + Linear combination of lagged forecast errors[upto q lags]

PROGRAM:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings('ignore')

%matplotlib inline

df = pd.read\_csv('march11.csv')

df.head()

chunksize = 10

for chunk in pd.read\_csv("C:\\Users\\HP\\Desktop\\march11.csv", chunksize=chunksize):

print(chunk)

df.date = pd.to\_datetime(df.date)

df.set\_index('date', inplace=True)

df.head()

OUTPUT:

date Met\_Mast\_Wind\_Speed Actual\_Wind\_Speed

0 03-01-2019 00:00 1.7518 2.2950

1 03-01-2019 00:10 1.8007 2.1500

2 03-01-2019 00:20 2.2892 1.7000

3 03-01-2019 00:30 3.4139 2.6890

4 03-01-2019 00:40 3.6930 3.0646

5 03-01-2019 00:50 6.0398 3.5361

6 03-01-2019 01:00 5.2690 6.2444

7 03-01-2019 01:10 6.0725 5.2823

8 03-01-2019 01:20 6.2293 5.4350

9 03-01-2019 01:30 6.7723 5.0163

date Met\_Mast\_Wind\_Speed Actual\_Wind\_Speed

10 03-01-2019 01:40 6.4859 5.1737

11 03-01-2019 01:50 6.2517 5.2684

12 03-01-2019 02:00 7.1599 5.7576

13 03-01-2019 02:10 7.2434 5.7897

14 03-01-2019 02:20 7.4380 5.8737

15 03-01-2019 02:30 7.6773 7.1673

16 03-01-2019 02:40 8.9421 7.9959

17 03-01-2019 02:50 9.5115 8.0820

18 03-01-2019 03:00 9.8026 6.8583

19 03-01-2019 03:10 9.8356 5.2074

date Met\_Mast\_Wind\_Speed Actual\_Wind\_Speed

20 03-01-2019 03:20 7.9385 4.8388

21 03-01-2019 03:30 5.8399 4.1152

22 03-01-2019 03:40 6.7535 3.8566

23 03-01-2019 03:50 4.8953 3.1458

24 03-01-2019 04:00 3.2893 3.6400

25 03-01-2019 04:10 4.2610 1.8102

26 03-01-2019 04:20 3.9302 1.8625

27 03-01-2019 04:30 3.2920 1.6065

28 03-01-2019 04:40 4.0896 2.2447

29 03-01-2019 04:50 3.6914 3.0860

date Met\_Mast\_Wind\_Speed Actual\_Wind\_Speed

30 03-01-2019 05:00 3.2087 3.2939

31 03-01-2019 05:10 3.1789 2.9928

32 03-01-2019 05:20 4.3093 2.7850

33 03-01-2019 05:30 4.5171 3.1263

34 03-01-2019 05:40 4.4674 3.7760

35 03-01-2019 05:50 4.7930 3.6310

36 03-01-2019 06:00 5.1959 3.6816

37 03-01-2019 06:10 5.1704 4.2525

38 03-01-2019 06:20 5.2531 5.0785

39 03-01-2019 06:30 5.2359 5.3429

date Met\_Mast\_Wind\_Speed Actual\_Wind\_Speed

40 03-01-2019 06:40 5.6384 5.1928

41 03-01-2019 06:50 5.8050 4.9865

42 03-01-2019 07:00 5.7427 4.5847

43 03-01-2019 07:10 5.4728 4.7847

44 03-01-2019 07:20 5.6208 5.0600

45 03-01-2019 07:30 5.3105 5.3124

46 03-01-2019 07:40 4.9459 5.4758

47 03-01-2019 07:50 5.1282 5.5031

48 03-01-2019 08:00 4.9242 4.9835

49 03-01-2019 08:10 4.9018 4.6396

date Met\_Mast\_Wind\_Speed Actual\_Wind\_Speed

50 03-01-2019 08:20 4.6355 2.3612

51 03-01-2019 08:30 4.3181 2.6144

52 03-01-2019 08:40 3.6476 2.1206

53 03-01-2019 08:50 3.9267 2.4596

54 03-01-2019 09:00 4.2205 3.6536

55 03-01-2019 09:10 4.5004 3.7396

56 03-01-2019 09:20 4.2845 3.0245

57 03-01-2019 09:30 4.8283 2.5735

58 03-01-2019 09:40 4.3326 2.1979

59 03-01-2019 09:50 3.5433 1.6458

date Met\_Mast\_Wind\_Speed Actual\_Wind\_Speed

60 03-01-2019 10:00 3.5061 2.3379

61 03-01-2019 10:10 3.4784 2.7072

62 03-01-2019 10:20 2.7847 1.5968

63 03-01-2019 10:30 2.4529 1.6163

64 03-01-2019 10:40 2.0301 1.5021

65 03-01-2019 10:50 2.5118 1.8798

66 03-01-2019 11:00 2.3350 1.9352

67 03-01-2019 11:10 2.0494 2.0278

68 03-01-2019 11:20 2.6770 1.1720

69 03-01-2019 11:30 2.7720 2.7608

date Met\_Mast\_Wind\_Speed Actual\_Wind\_Speed

70 03-01-2019 11:40 2.6562 2.6806

71 03-01-2019 11:50 2.4647 2.3990

72 03-01-2019 12:00 1.8634 3.0590

73 03-01-2019 12:10 2.8007 2.5052

74 03-01-2019 12:20 2.1652 1.6105

75 03-01-2019 12:30 1.8446 1.5500

76 03-01-2019 12:40 2.5642 3.3949

77 03-01-2019 12:50 3.1453 3.8091

78 03-01-2019 13:00 2.8058 3.6592

79 03-01-2019 13:10 1.5033 3.7219

date Met\_Mast\_Wind\_Speed Actual\_Wind\_Speed

80 03-01-2019 13:20 3.0335 3.8165

81 03-01-2019 13:30 2.6260 3.0330

82 03-01-2019 13:40 1.2790 2.5485

83 03-01-2019 13:50 2.1175 3.0165

84 03-01-2019 14:00 3.1289 4.1240

85 03-01-2019 14:10 1.7430 2.9040

86 03-01-2019 14:20 2.1393 3.3894

87 03-01-2019 14:30 2.6249 3.1480

88 03-01-2019 14:40 3.3854 2.5897

89 03-01-2019 14:50 2.7829 1.2571

date Met\_Mast\_Wind\_Speed Actual\_Wind\_Speed

90 03-01-2019 15:00 2.5950 1.8167

91 03-01-2019 15:10 3.0600 3.6000

92 03-01-2019 15:20 2.8188 2.7939

93 03-01-2019 15:30 3.1748 3.3432

94 03-01-2019 15:40 3.5744 2.9566

95 03-01-2019 15:50 1.2211 4.4061

96 03-01-2019 16:00 3.8888 2.6694

97 03-01-2019 16:10 3.4709 3.0949

98 03-01-2019 16:20 2.6012 3.1118

99 03-01-2019 16:30 2.0091 3.7351

date Met\_Mast\_Wind\_Speed Actual\_Wind\_Speed

100 03-01-2019 16:40 2.8235 2.2717

101 03-01-2019 16:50 3.7327 1.7889

102 03-01-2019 17:00 4.2723 1.4903

103 03-01-2019 17:10 2.8419 2.9347

104 03-01-2019 17:20 2.0367 2.4143

105 03-01-2019 17:30 3.0457 2.6052

106 03-01-2019 17:40 3.4905 3.7174

107 03-01-2019 17:50 3.9426 3.5892

108 03-01-2019 18:00 4.7584 3.0911

109 03-01-2019 18:10 4.8125 3.3011

date Met\_Mast\_Wind\_Speed Actual\_Wind\_Speed

110 03-01-2019 18:20 4.6346 3.4380

111 03-01-2019 18:30 4.6631 3.6896

112 03-01-2019 18:40 4.5485 4.1381

113 03-01-2019 18:50 4.7262 6.2421

114 03-01-2019 19:00 5.9285 6.2253

115 03-01-2019 19:10 7.1344 5.8639

116 03-01-2019 19:20 6.7474 5.8724

117 03-01-2019 19:30 4.4114 5.7214

118 03-01-2019 19:40 3.4274 5.0691

119 03-01-2019 19:50 3.4436 4.0697

date Met\_Mast\_Wind\_Speed Actual\_Wind\_Speed

120 03-01-2019 20:00 4.7638 3.8000

121 03-01-2019 20:10 4.7375 3.8940

122 03-01-2019 20:20 4.1400 3.3867

123 03-01-2019 20:30 4.2151 2.2889

124 03-01-2019 20:40 4.6226 2.1979

125 03-01-2019 20:50 4.0932 2.2072

126 03-01-2019 21:00 3.5163 3.1000

127 03-01-2019 21:10 3.2643 3.3309

128 03-01-2019 21:20 3.3559 3.4500

129 03-01-2019 21:30 4.1681 3.6636

date Met\_Mast\_Wind\_Speed Actual\_Wind\_Speed

130 03-01-2019 21:40 5.0796 3.6695

131 03-01-2019 21:50 4.7555 5.1792

132 03-01-2019 22:00 4.6653 5.4604

133 03-01-2019 22:10 5.5994 5.0062

134 03-01-2019 22:20 6.3592 4.8896

135 03-01-2019 22:30 5.7581 8.9172

136 03-01-2019 22:40 8.0323 8.5021

137 03-01-2019 22:50 5.7741 7.2464

138 03-01-2019 23:00 3.8623 7.3697

139 03-01-2019 23:10 3.8318 6.9847

date Met\_Mast\_Wind\_Speed Actual\_Wind\_Speed

140 03-01-2019 23:20 3.7164 6.8788

141 03-01-2019 23:30 4.0368 6.9101

142 03-01-2019 23:40 3.8042 6.6316

143 03-01-2019 23:50 3.8798 6.6296

Out[1]:

|  | **Met\_Mast\_Wind\_Speed** | **Actual\_Wind\_Speed** |
| --- | --- | --- |
| **Date** |  |  |
| **2019-03-01 00:00:00** | 1.7518 | 2.2950 |
| **2019-03-01 00:10:00** | 1.8007 | 2.1500 |
| **2019-03-01 00:20:00** | 2.2892 | 1.7000 |
| **2019-03-01 00:30:00** | 3.4139 | 2.6890 |
| **2019-03-01 00:40:00** | 3.6930 | 3.0646 |

1

**from** statsmodels.tsa.arima\_model **import** ARIMA

*#(1,1,2 )ARIMA model*

model **=** ARIMA(df.Met\_Mast\_Wind\_Speed, order**=**(1,1,2))

model\_fit **=** model.fit(disp**=**0)

print(model\_fit)

print(model\_fit.summary())

*OUTPUT:*

<statsmodels.tsa.arima\_model.ARIMAResultsWrapper object at 0x0E5D8070>

ARIMA Model Results

=================================================================================

Dep. Variable: D.Met\_Mast\_Wind\_Speed No. Observations: 143

Model: ARIMA(1, 1, 2 ) Log Likelihood -174.762

Method: css-mle S.D. of innovations 0.821

Date: Mon, 15 Jul 2019 AIC 359.524

Time: 09:45:08 BIC 374.338

Sample: 03-01-2019 HQIC 365.544

- 03-01-2019

===============================================================================================

coef std err z P>|z| [0.025 0.975]

-----------------------------------------------------------------------------------------------

const 0.0145 0.056 0.259 0.796 -0.096 0.125

ar.L1.D.Met\_Mast\_Wind\_Speed -0.2265 0.345 -0.656 0.513 -0.903 0.450

ma.L1.D.Met\_Mast\_Wind\_Speed 0.1713 0.338 0.507 0.613 -0.491 0.833

ma.L2.D.Met\_Mast\_Wind\_Speed -0.1696 0.077 -2.216 0.028 -0.320 -0.020

Roots

=============================================================================

Real Imaginary Modulus Frequency

-----------------------------------------------------------------------------

AR.1 -4.4145 +0.0000j 4.4145 0.5000

MA.1 -1.9754 +0.0000j 1.9754 0.5000

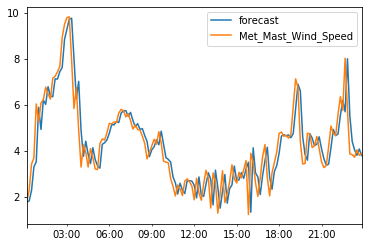
MA.2 2.9855 +0.0000j 2.9855 0.0000

model\_fit.plot\_predict(dynamic**=False**)

plt.rcParams["figure.figsize"] **=** (10.0,20.0)

*#plt.title('met mast forecast')*

plt.show()



**from** statsmodels.tsa.stattools **import** acf

*# Create Training and Test*

train **=** df.Met\_Mast\_Wind\_Speed[:85]

test **=**df.Met\_Mast\_Wind\_Speed[85:]

*# Build Model*

*#model = ARIMA(train, order=(3,2,1))*

model **=** ARIMA(train, order**=**(3, 1, 1))

fitted **=** model.fit(disp**=-**1)

*# Forecast*

fc, se, conf **=** fitted.forecast(59, alpha**=**0.05)

​

*# Make as pandas series*

fc\_series **=** pd.Series(fc, index**=**test.index)

lower\_series **=** pd.Series(conf[:, 0], index**=**test.index)

upper\_series **=** pd.Series(conf[:, 1], index**=**test.index)

*# Plot*

plt.figure(figsize**=**(12,5), dpi**=**100)

plt.plot(train, label**=**'training')

plt.plot(test, label**=**'met mast wind speed')

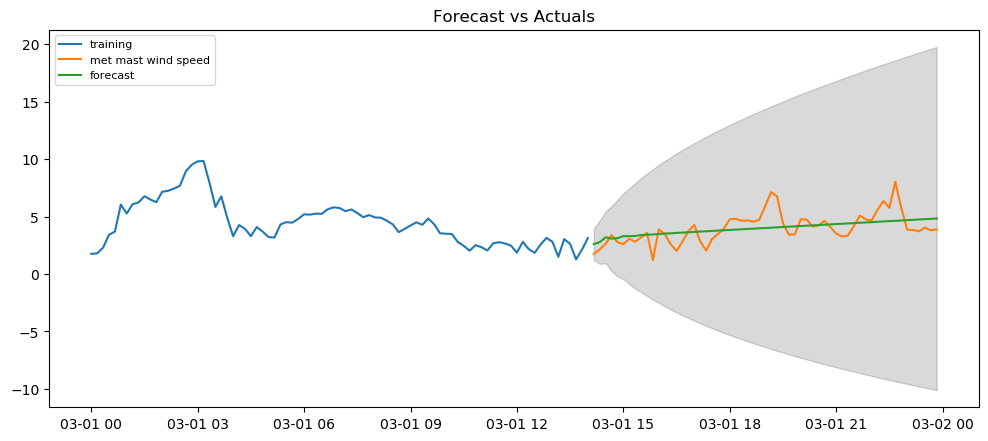
plt.plot(fc\_series, label**=**'forecast')

plt.fill\_between(lower\_series.index, lower\_series, upper\_series, color**=**'k', alpha**=**.15)

plt.title('Forecast vs Actuals')

plt.legend(loc**=**'upper left', fontsize**=**8)

plt.show()

**

*# Accuracy metrics*

**def** forecast\_accuracy(forecast, actual):

mape **=** np.mean(np.abs(forecast **-** actual)**/**np.abs(actual))

me **=** np.mean(forecast **-** actual)

mae **=** np.mean(np.abs(forecast **-** actual))

mpe **=** np.mean((forecast **-** actual)**/**actual)

rmse **=** np.mean((forecast **-** actual)**\*\***2)**\*\***.5

corr **=** np.corrcoef(forecast, actual)[0,1] *# corr*

mins **=** np.amin(np.hstack([forecast[:,**None],**actual[:,**None**]]), axis**=**1)

maxs **=** np.amax(np.hstack([forecast[:,**None**], actual[:,**None**]]), axis**=**1)

minmax **=** 1 **-** np.mean(mins**/**maxs)

acf1 **=** acf(fc**-**test)[1]

**return**({'mape':mape, 'me':me,'mpe': mpe, 'rmse':rmse, 'acf1':acf1,'corr':corr, 'minmax':minmax})

forecast\_accuracy(fc, test.values)

{'mape': 0.23283007071038483,

'me': -0.03163500589819844,

'mae': 0.8303964220590987,

'mpe': 0.08122255823546266,

'rmse': 1.0833519549346429,

'acf1': 0.6094760373679385,

'corr': 0.5788654228175466,

'minmax': 0.18041326036563932}

*# Build Model*

model **=** ARIMA(train, order**=**(3, 1, 0))

fitted **=** model.fit(disp**=-**1)

print(fitted.summary())

*# Forecast*

fc, se, conf **=** fitted.forecast(59, alpha**=**0.05)

*# Make as pandas series*

fc\_series **=** pd.Series(fc, index**=**test.index)

lower\_series **=** pd.Series(conf[:, 0], index**=**test.index)

upper\_series **=** pd.Series(conf[:, 1], index**=**test.index)

*# Plot*

plt.figure(figsize**=**(12,5), dpi**=**100)

plt.plot(train, label**=**'training')

plt.plot(test, label**=**'met mast wind speed')

plt.plot(fc\_series, label**=**'forecast')

plt.fill\_between(lower\_series.index, lower\_series, upper\_series, color**=**'k', alpha**=**.15)

plt.title('Forecast vs Actuals')

plt.legend(loc**=**'upper left', fontsize**=**8)

x**=**(fc\_series)

print(x)

*#print(lower\_series)*

plt.show()

ARIMA Model Results

=================================================================================

Dep. Variable: D.Met\_Mast\_Wind\_Speed No. Observations: 84

Model: ARIMA(3, 1, 0) Log Likelihood -88.637

Method: css-mle S.D. of innovations 0.694

Date: Mon, 15 Jul 2019 AIC 187.275

Time: 09:46:09 BIC 199.429

Sample: 03-01-2019 HQIC 192.160

- 03-01-2019

===============================================================================================

coef std err z P>|z| [0.025 0.975]

-----------------------------------------------------------------------------------------------

const 0.0247 0.101 0.244 0.808 -0.174 0.223

ar.L1.D.Met\_Mast\_Wind\_Speed -0.0065 0.104 -0.063 0.950 -0.209 0.196

ar.L2.D.Met\_Mast\_Wind\_Speed -0.0615 0.104 -0.592 0.556 -0.265 0.142

ar.L3.D.Met\_Mast\_Wind\_Speed 0.3290 0.107 3.083 0.003 0.120 0.538

Roots

=============================================================================

Real Imaginary Modulus Frequency

-----------------------------------------------------------------------------

AR.1 -0.6658 -1.2484j 1.4148 -0.3280

AR.2 -0.6658 +1.2484j 1.4148 0.3280

AR.3 1.5183 -0.0000j 1.5183 -0.0000

-----------------------------------------------------------------------------

date

2019-03-01 14:10:00 2.645829

2019-03-01 14:20:00 2.880984

2019-03-01 14:30:00 3.260177

2019-03-01 14:40:00 3.102567

2019-03-01 14:50:00 3.175929

2019-03-01 15:00:00 3.328163

2019-03-01 15:10:00 3.289064

2019-03-01 15:20:00 3.322364

2019-03-01 15:30:00 3.392900

2019-03-01 15:40:00 3.395790

2019-03-01 15:50:00 3.420654

2019-03-01 16:00:00 3.461784

2019-03-01 16:10:00 3.479200

2019-03-01 16:20:00 3.503001

2019-03-01 16:30:00 3.533570

2019-03-01 16:40:00 3.555899

2019-03-01 16:50:00 3.579967

2019-03-01 17:00:00 3.606758

2019-03-01 17:10:00 3.630712

2019-03-01 17:20:00 3.655090

2019-03-01 17:30:00 3.680535

2019-03-01 17:40:00 3.705013

2019-03-01 17:50:00 3.729572

2019-03-01 18:00:00 3.754541

2019-03-01 18:10:00 3.779185

2019-03-01 18:20:00 3.803831

2019-03-01 18:30:00 3.828633

2019-03-01 18:40:00 3.853326

2019-03-01 18:50:00 3.878011

2019-03-01 19:00:00 3.902755

2019-03-01 19:10:00 3.927462

2019-03-01 19:20:00 3.952164

2019-03-01 19:30:00 3.976887

2019-03-01 19:40:00 4.001598

2019-03-01 19:50:00 4.026307

2019-03-01 20:00:00 4.051023

2019-03-01 20:10:00 4.075735

2019-03-01 20:20:00 4.100446

2019-03-01 20:30:00 4.125160

2019-03-01 20:40:00 4.149872

2019-03-01 20:50:00 4.174584

2019-03-01 21:00:00 4.199297

2019-03-01 21:10:00 4.224010

2019-03-01 21:20:00 4.248722

2019-03-01 21:30:00 4.273435

2019-03-01 21:40:00 4.298147

2019-03-01 21:50:00 4.322860

2019-03-01 22:00:00 4.347572

2019-03-01 22:10:00 4.372285

2019-03-01 22:20:00 4.396997

2019-03-01 22:30:00 4.421710

2019-03-01 22:40:00 4.446422

2019-03-01 22:50:00 4.471135

2019-03-01 23:00:00 4.495847

2019-03-01 23:10:00 4.520560

2019-03-01 23:20:00 4.545272

2019-03-01 23:30:00 4.569985

2019-03-01 23:40:00 4.594697

2019-03-01 23:50:00 4.619410

dtype: float64

**import** matplotlib.pyplot **as** plt

**import** pandas **as** pd

**import** math

data**=**pd.read\_csv("lab.csv",index\_col**=**0)

**def** CalculatePower(actualWindSpeed):

y1 **=** math.floor(actualWindSpeed)

y2 **=** math.ceil(actualWindSpeed)

x1**=**data.loc[y1]

x2**=**data.loc[y2]

p**=**x1[0]

q**=**x2[0]

print("power")

**if** y2 **==** y1 **or** p**==**q :

**return**(y1)

**else**:

m**=**(y2**-**y1)**/**(q**-**p)

ans**=**((actualWindSpeed**-**y1)**/**m)**+**x1

**return**(ans[0])

s**=** fc\_series

**for** i **in** s:

print("\n")

print("Forecast wind speed:")

print(i)

print(CalculatePower(i))

df**=**pd.DataFrame(fc\_series)

df.to\_csv("march111.csv",index**=False**)

OUTPUT:

Forecast wind speed:

2.645828552165926

power

23.249827877973342

Forecast wind speed:

2.880984306245734

power

31.71543502484642

Forecast wind speed:

3.260177148515883

power

70.60356075261242

Forecast wind speed:

3.1025673689089346

power

49.6414600648883

Forecast wind speed:

3.1759293008393836

power

59.398597011638024

Forecast wind speed:

3.328162942584585

power

79.64567136374978

Forecast wind speed:

3.2890638705733766

power

74.44549478625908

Forecast wind speed:

3.3223638308392998

power

78.87438950162687

Forecast wind speed:

3.39289989981891

power

88.25568667591503

Forecast wind speed:

3.395789611867632

power

88.64001837839504

Forecast wind speed:

3.420654222907235

power

91.94701164666228

Forecast wind speed:

3.461784114630983

power

97.41728724592076

Forecast wind speed:

3.4791997358357887

power

99.7335648661599

Forecast wind speed:

3.50300106243563

power

102.89914130393876

Forecast wind speed:

3.5335697357608105

power

106.9647748561878

Forecast wind speed:

3.55589910646748

power

109.93458116017483

Forecast wind speed:

3.5799674967954696

power

113.13567707379745

Forecast wind speed:

3.6067575094719175

power

116.69874875976504

Forecast wind speed:

3.6307119092365974

power

119.88468392846745

Forecast wind speed:

3.6550897647013723

power

123.12693870528251

Forecast wind speed:

3.6805346031200794

power

126.51110221497056

Forecast wind speed:

3.705013451386911

power

129.76678903445918

Forecast wind speed:

3.729572368596873

power

133.03312502338412

Forecast wind speed:

3.754541193866148

power

136.3539787841977

Forecast wind speed:

3.779184582707866

power

139.63154950014615

Forecast wind speed:

3.803831251972835

power

142.90955651238704

Forecast wind speed:

3.82863277038959

power

146.20815846181551

Forecast wind speed:

3.853325997846304

power

149.49235771355848

Forecast wind speed:

3.878011495991729

power

152.77552896689997

Forecast wind speed:

3.9027546492492777

power

156.06636835015394

Forecast wind speed:

3.9274622700741073

power

159.35248191985627

Forecast wind speed:

3.952164036726436

power

162.63781688461597

Forecast wind speed:

3.976886995380223

power

165.92597038556968

Forecast wind speed:

4.001598484180642

power

169.35486348810247

Forecast wind speed:

4.026306819390826

power

174.84011390476343

Forecast wind speed:

4.05102285283378

power

180.3270733290992

Forecast wind speed:

4.075735255881166

power

185.81322680561883

Forecast wind speed:

4.100446171938297

power

191.29905017030197

Forecast wind speed:

4.125159853746501

power

196.78548753172325

Forecast wind speed:

4.149872414365541

power

202.27167598915014

Forecast wind speed:

4.174584323081099

power

207.757719724004

Forecast wind speed:

4.199297214966695

power

213.24398172260624

Forecast wind speed:

4.224009771588636

power

218.73016929267723

Forecast wind speed:

4.248722055486518

power

224.2162963180071

Forecast wind speed:

4.273434685259891

power

229.7025001276957

Forecast wind speed:

4.2981472192225025

power

235.18868266739557

Forecast wind speed:

4.322859642821726

power

240.67484070642314

Forecast wind speed:

4.347572186832636

power

246.1610254768452

Forecast wind speed:

4.372284705314744

power

251.64720457987318

Forecast wind speed:

4.396997180251194

power

257.1333740157651

Forecast wind speed:

4.421709696659795

power

262.6195526584744

Forecast wind speed:

4.446422207073821

power

268.1057299703883

Forecast wind speed:

4.471134700650636

power

273.59190354444127

Forecast wind speed:

4.495847208351339

power

279.0780802539973

Forecast wind speed:

4.520559715022102

power

284.56425673490673

Forecast wind speed:

4.5452722152917095

power

290.0504317947595

Forecast wind speed:

4.569984720313584

power

295.53660790961555

Forecast wind speed:

4.594697225358909

power

301.02278402967784

Forecast wind speed:

4.619409728005876

power

306.5089596173045

**from** pandas **import** DataFrame

predicted **=** {'Forecast\_Wind\_Speed': fc\_series, 'Power': a}

df **=** DataFrame(predicted, columns**=** ['Forecast\_Wind\_Speed','Power'])

export\_csv **=** df.to\_csv (r'march111.csv', index **=** **None**, header**=True**) *#Don't forget to add '.csv' at the end of the path*

print (df)

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

dataframe**=** pd.read\_csv("march111.csv")

y **=** dataframe.Power

dataframe.sort\_values(['Forecast\_Wind\_Speed'],axis**=**0,ascending**=True**,inplace**=True**)

x **=** dataframe.Forecast\_Wind\_Speed

plt.scatter(x, y)

plt.show()

Forecast\_Wind\_Speed Power

date

2019-03-01 14:10:00 2.645829 23.249828

2019-03-01 14:20:00 2.880984 31.715435

2019-03-01 14:30:00 3.260177 70.603561

2019-03-01 14:40:00 3.102567 49.641460

2019-03-01 14:50:00 3.175929 59.398597SS

2019-03-01 15:00:00 3.328163 79.645671

2019-03-01 15:10:00 3.289064 74.445495

2019-03-01 15:20:00 3.322364 78.874390

2019-03-01 15:30:00 3.392900 88.255687

2019-03-01 15:40:00 3.395790 88.640018

2019-03-01 15:50:00 3.420654 91.947012

2019-03-01 16:00:00 3.461784 97.417287

2019-03-01 16:10:00 3.479200 99.733565

2019-03-01 16:20:00 3.503001 102.899141

2019-03-01 16:30:00 3.533570 106.964775

2019-03-01 16:40:00 3.555899 109.934581

2019-03-01 16:50:00 3.579967 113.135677

2019-03-01 17:00:00 3.606758 116.698749

2019-03-01 17:10:00 3.630712 119.884684

2019-03-01 17:20:00 3.655090 123.126939

2019-03-01 17:30:00 3.680535 126.511102

2019-03-01 17:40:00 3.705013 129.766789

2019-03-01 17:50:00 3.729572 133.033125

2019-03-01 18:00:00 3.754541 136.353979

2019-03-01 18:10:00 3.779185 139.631550

2019-03-01 18:20:00 3.803831 142.909557

2019-03-01 18:30:00 3.828633 146.208158

2019-03-01 18:40:00 3.853326 149.492358

2019-03-01 18:50:00 3.878011 152.775529

2019-03-01 19:00:00 3.902755 156.066368

2019-03-01 19:10:00 3.927462 159.352482

2019-03-01 19:20:00 3.952164 162.637817

2019-03-01 19:30:00 3.976887 165.925970

2019-03-01 19:40:00 4.001598 169.354863

2019-03-01 19:50:00 4.026307 174.840114

2019-03-01 20:00:00 4.051023 180.327073

2019-03-01 20:10:00 4.075735 185.813227

2019-03-01 20:20:00 4.100446 191.299050

2019-03-01 20:30:00 4.125160 196.785488

2019-03-01 20:40:00 4.149872 202.271676

2019-03-01 20:50:00 4.174584 207.757720

2019-03-01 21:00:00 4.199297 213.243982

2019-03-01 21:10:00 4.224010 218.730169

2019-03-01 21:20:00 4.248722 224.216296

2019-03-01 21:30:00 4.273435 229.702500

2019-03-01 21:40:00 4.298147 235.188683

2019-03-01 21:50:00 4.322860 240.674841

2019-03-01 22:00:00 4.347572 246.161025

2019-03-01 22:10:00 4.372285 251.647205

2019-03-01 22:20:00 4.396997 257.133374

2019-03-01 22:30:00 4.421710 262.619553

2019-03-01 22:40:00 4.446422 268.105730

2019-03-01 22:50:00 4.471135 273.591904

2019-03-01 23:00:00 4.495847 279.078080

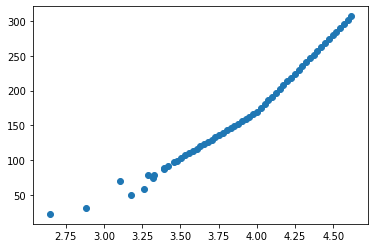
2019-03-01 23:10:00 4.520560 284.564257

2019-03-01 23:20:00 4.545272 290.050432

2019-03-01 23:30:00 4.569985 295.536608

2019-03-01 23:40:00 4.594697 301.022784

2019-03-01 23:50:00 4.619410 306.508960



bins **=** pd.cut(dataframe['Power'], [20,30,40,50,60,70,80,90,100,110,120,130,140,150,160,170,180,190,200,210,220,230,240,250,260,270,280,290,300,310])

bins1**=**dataframe.groupby(bins)['Power'].agg(['count', 'sum'])

print(bins1)

count sum

Power

(20, 30] 1 23.249828

(30, 40] 1 31.715435

(40, 50] 1 49.641460

(50, 60] 1 59.398597

(60, 70] 0 0.000000

(70, 80] 4 303.569116

(80, 90] 2 176.895705

(90, 100] 3 289.097864

(100, 110] 3 319.798497

(110, 120] 3 349.719110

(120, 130] 3 379.404830

(130, 140] 3 409.018653

(140, 150] 3 438.610073

(150, 160] 3 468.194379

(160, 170] 3 497.918651

(170, 180] 1 174.840114

(180, 190] 2 366.140300

(190, 200] 2 388.084538

(200, 210] 2 410.029396

(210, 220] 2 431.974151

(220, 230] 2 453.918796

(230, 240] 1 235.188683

(240, 250] 2 486.835866

(250, 260] 2 508.780579

(260, 270] 2 530.725283

(270, 280] 2 552.669984

(280, 290] 1 284.564257

(290, 300] 2 585.587040

(300, 310] 2 607.531744

#CALCULATION OF DEVIATION

import matplotlib.pyplot as plt

dataframe= pd.read\_csv("power1.csv")

y = dataframe.actual

z=dataframe.availability

x = dataframe.forecast

#CALCULATION OF PENALTY

dataframe= pd.read\_csv("power.csv")

y = dataframe.actual

z=dataframe.availability

x = dataframe.forecast

d=dataframe.deviationinp

deviationinp=100\*((dataframe.forecast-dataframe.actual)/dataframe.availability)

deviation=deviation\*250

for d in y:

if dataframe.deviationinp < 15:

dataframe.TB\_15=dataframe.deviation

dataframe.TB\_15\_25=0

dataframe.TB\_25\_35=0

dataframe.TB\_ABOVE35=0

if 15<=d and d<25:

dataframe.TB\_15=(dataframe.availability\*250\*0.15)

dataframe.TB\_15\_25=(dataframe.availability-(dataframe.availability\*250\*0.15))

dataframe.TB\_25\_35=0

dataframe.TB\_ABOVE35=0

if 25<=d and d<35:

dataframe.TB\_15=(dataframe.availability\*250\*0.15)

dataframe.TB\_15\_25=((dataframe.availability\*250\*0.10)-dataframe.availability- (dataframe.availability\*250\*0.15))

dataframe.TB\_25\_35=(dataframe.availability-(dataframe.availability\*250\*0.15)-(dataframe.availability\*250\*0.10))

dataframe.TB\_ABOVE35=0

else:

dataframe.TB\_15=(dataframe.availability\*250\*0.15)

dataframe.TB\_15\_25=((dataframe.availability\*250\*0.10)-dataframe.availability-(dataframe.availability\*250\*0.15))

dataframe.TB\_25\_35=((dataframe.availability\*250\*0.10)-dataframe.availability-(dataframe.availability\*250\*0.15)-(dataframe.availability\*250\*0.10))

dataframe.TB\_ABOVE35=(dataframe.availability-(dataframe.availability\*250\*0.15)-(dataframe.availability\*250\*0.10)-(dataframe.availability\*250\*0.10))

predicted = {'forecast':x,'actual':y,'availability':z,'deviationinp': deviation,'deviation':devr,’TB\_15%’:TB\_15,’TB\_15%-25%’:TB\_15\_25,’TB25%35%’:TB\_25\_35,’TB>35%’:TB\_ABOVE35}

df = DataFrame(predicted, columns= ['forecast','actual','availability','deviationinp','deviation',’TB\_15%’,’TB15%-25%’,’TB25%-35%’,’TB>35%’])

export\_csv = df.to\_csv (r'power.csV’, index = None, header=True)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Actual | Available | Forecast | Deviation  (%) | Deviation | Penalty  15% | Penalty  15%-25% | Penalty  25%-35% | Penalty  >35% |
| 46 | 100 | 70 | 24% | 6000 | 3750 | 2250 | 2250 | 6750 |
| 37 | 100 | 70 | 33% | 8250 | 3750 | 2500 | 2000 | 3250 |
| 0 | 100 | 70 | 70% | 17500 | 3750 | 2500 | 3750 | 16250 |

CONCLUSION:

Wind energy is probably the solution for our energy demands.it has great potential and is easy to manage. All you have to do is build the turbine and everything else is going to be free. With only one turbine, you can power over 200 homes. Every wind turbine lasts for about 20-25 years. As long as the wind blows, wind turbines can harness the wind to create power. Wind power only makes up a tiny percent of electricity that is produced. Unlike coal, wind turbine don’t create greenhouse gases and are completely renewable sources. Many people believe that the wind energy could soon be our main source of energy. Though wind turbine can cause complaints and fatalities of wildlife, it could be the energy solution we have been looking for.

Theoretical concepts studied being implemented in practical concepts can help me to correlate the two and as consequence can bring about out of box innovations. Overall internship was really a good program. It helps to enhance and develop my skills, abilities, and knowledge. The treatment by the company was just, equitable and professional. I have learned from different units and people. Hence, it was necessary to gain technical knowledge in engineering disciplines. Overall, it was very fruitful.

We got the accuracy of the forecasted power above the 90% which meets expected result using machine learning algorithm.

-

`