Wind Turbine Power Prediction

Submitted By:

Dhrubajyoti Chakraborty

PGPDS

Praxis Business School

Problem Understanding

- As the global warming situation worsens and the burning of fossil fuels causes air pollution and climate change, renewable energy development and deployment are becoming increasingly important.
- Changes in the climate affect wind power forecasting under different weather conditions.
- At all times in the power system, a balance between electricity demand and generation must be maintained.
- Fluctuations of wind generation require power substitution from other sources that might not be available on a short notice.
- When wind power accounts for more than a modest percentage of total electricity provided to the grid, the problem becomes more complicated.

Tools and Techniques Used

- Spark
- Python
- Google Collaboratory
- Pandas
- Seaborn
- Matplotlib

Setting up Spark Environment

- Installed all the dependencies in Colab environment such as Apache Spark 3.1.2, openjdk 8.
- Set the environment path that enables us to run PySpark in Google Colab environment.
- Run a local spark session to test our installation.

Understanding the Dataset

- **Date/Time** (for 10 minutes intervals)
- **LV Active Power (kW):** The power generated by the turbine for that moment.
- **Wind Speed (m/s):** The wind speed at the hub height of the turbine (the wind speed that turbine use for electricity generation).
- **Theoretical Power Curve (KWh):** The theoretical power values that the turbine generates with that wind speed which is given by the turbine manufacturer.
- **Wind Direction (°):** The wind direction at the hub height of the turbine (wind turbines turn to this direction automatically).

Glimpse of Dataset

date/time	lv activepower (kw)	wind speed (m/s)	theoretical_power_curve (kwh)	wind direction (°)
01 01 2018 00:00	380.047790527343	5.31133604049682	416.328907824861	259.994903564453
01 01 2018 00:10	453.76919555664	5.67216682434082	519.917511061494	268.64111328125
01 01 2018 00:20	306.376586914062	5.21603679656982	390.900015810951	272.564788818359
01 01 2018 00:30	419.645904541015	5.65967416763305	516.127568975674	271.258087158203
01 01 2018 00:40	380.650695800781	5.57794094085693	491.702971953588	265.674285888671

only showing top 5 rows

- No of Rows : 50530
- No of Columns: 5

Creating Hour and Month columns

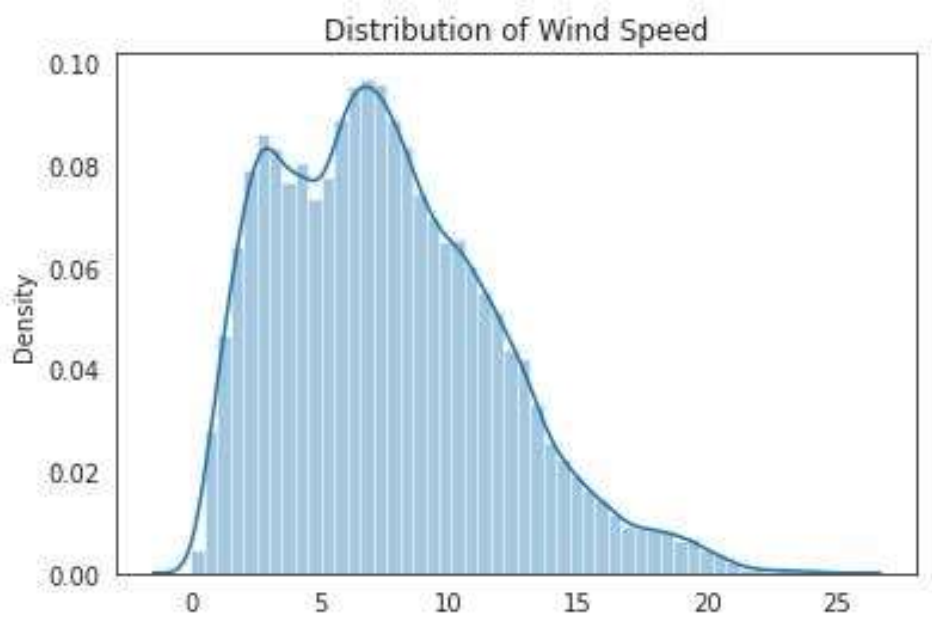
date/time	lv activepower (kw)	wind speed (m/s)	theoretical_power_curve (kwh)	wind direction (°)
01 01 2018 00:00	380.047790527343	5.31133604049682	416.328907824861	259.994903564453
01 01 2018 00:10	453.76919555664	5.67216682434082	519.917511061494	268.64111328125
01 01 2018 00:20	306.376586914062	5.21603679656982	390.900015810951	272.564788818359
01 01 2018 00:30	419.645904541015	5.65967416763305	516.127568975674	271.258087158203
01 01 2018 00:40	380.650695800781	5.57794094085693	491.702971953588	265.674285888671

only showing top 5 rows

date/time	lv activepower (kw)	wind speed (m/s)	theoretical_power_curve (kwh)	wind direction (°)	month	hour
01 01 2018 00:00	380.047790527343	5.31133604049682	416.328907824861	259.994903564453	1	0
01 01 2018 00:10	453.76919555664	5.67216682434082	519.917511061494	268.64111328125	1	0
01 01 2018 00:20	306.376586914062	5.21603679656982	390.900015810951	272.564788818359	1	0
01 01 2018 00:30	419.645904541015	5.65967416763305	516.127568975674	271.258087158203	1	0
01 01 2018 00:40	380.650695800781	5.57794094085693	491.702971953588	265.674285888671	1	0

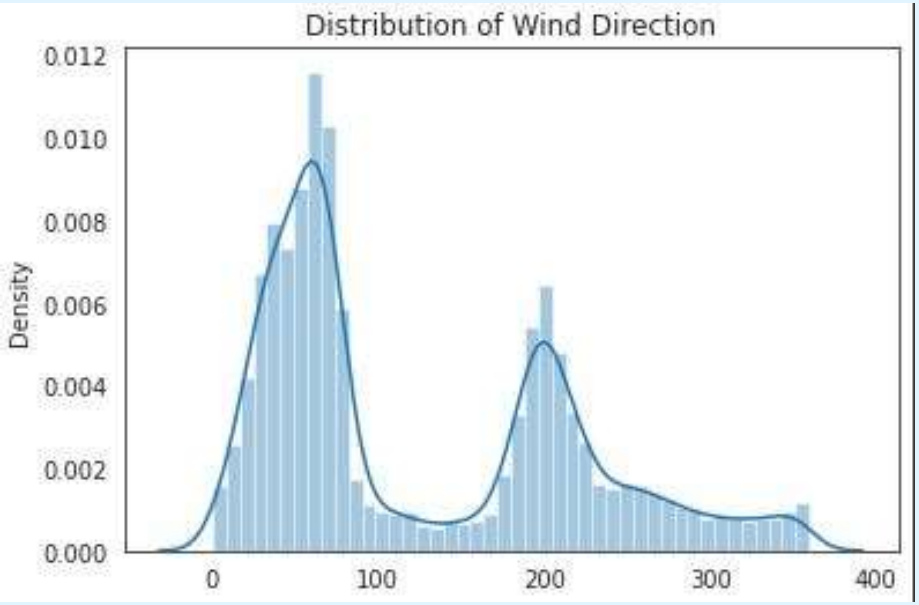
only showing top 5 rows

Exploratory Data Analysis



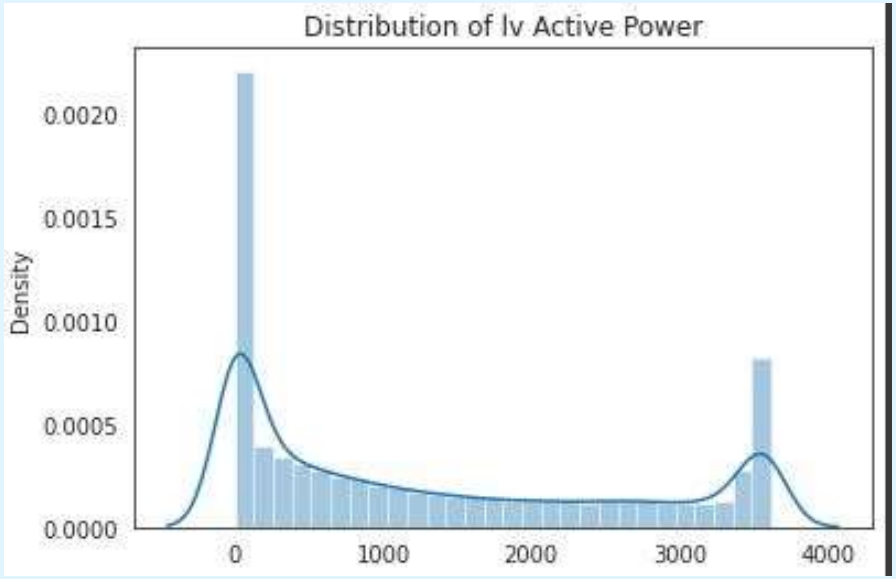
- Distribution seems to be mostly normal.
- Most wind speed falls between 2 – 7.

Exploratory Data Analysis



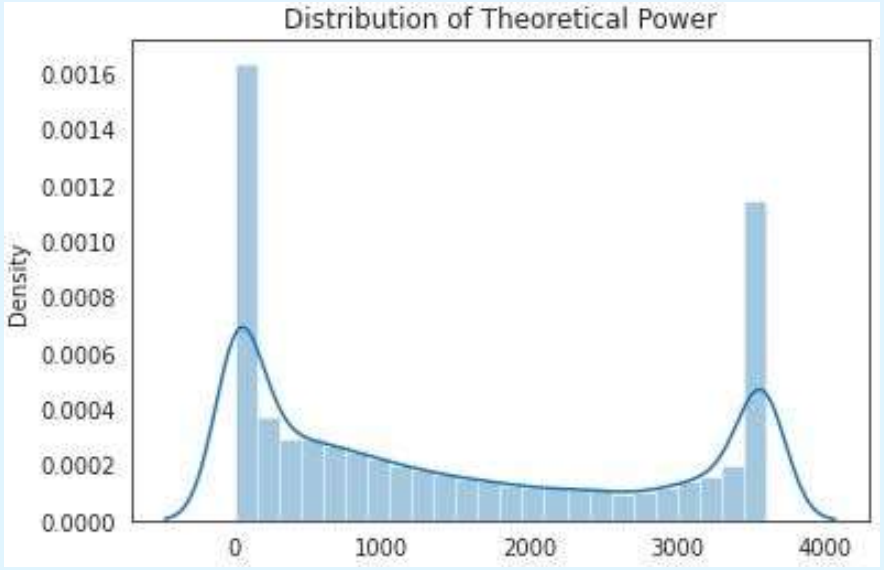
- 2 peaks indicates that it is a bimodal distribution.
- Most of the observations fall in these 2 peaks.

Exploratory Data Analysis



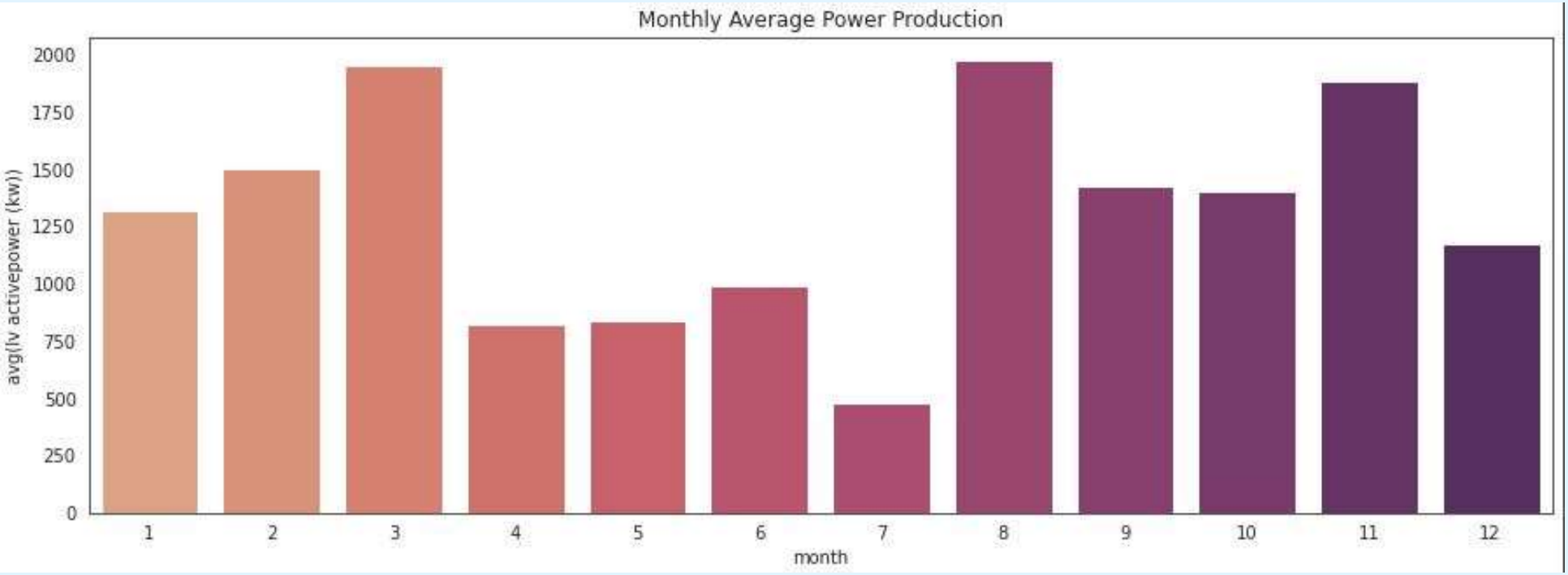
- Distribution seems to be right skewed.
- At 0 and 3500 most of the data fall in this range.

Exploratory Data Analysis



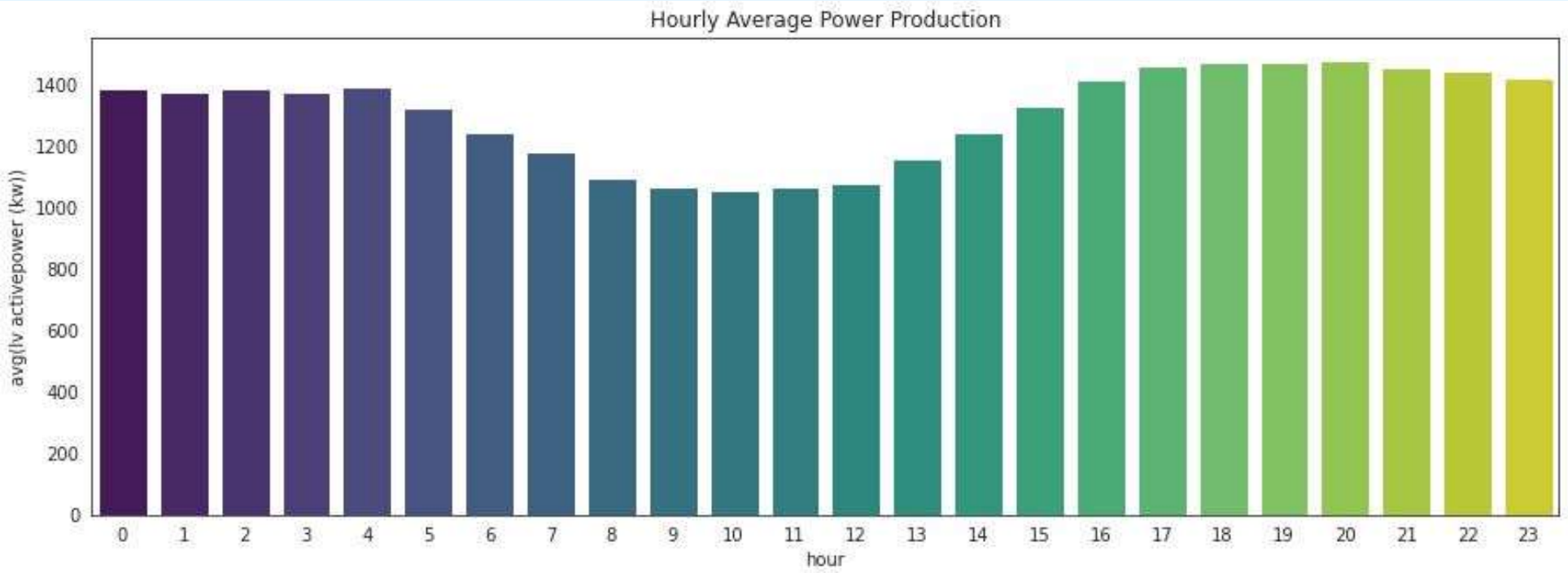
- Distribution seems to be right skewed.
- At 0 and 3500 most of the data fall in this range.

Average power production by month



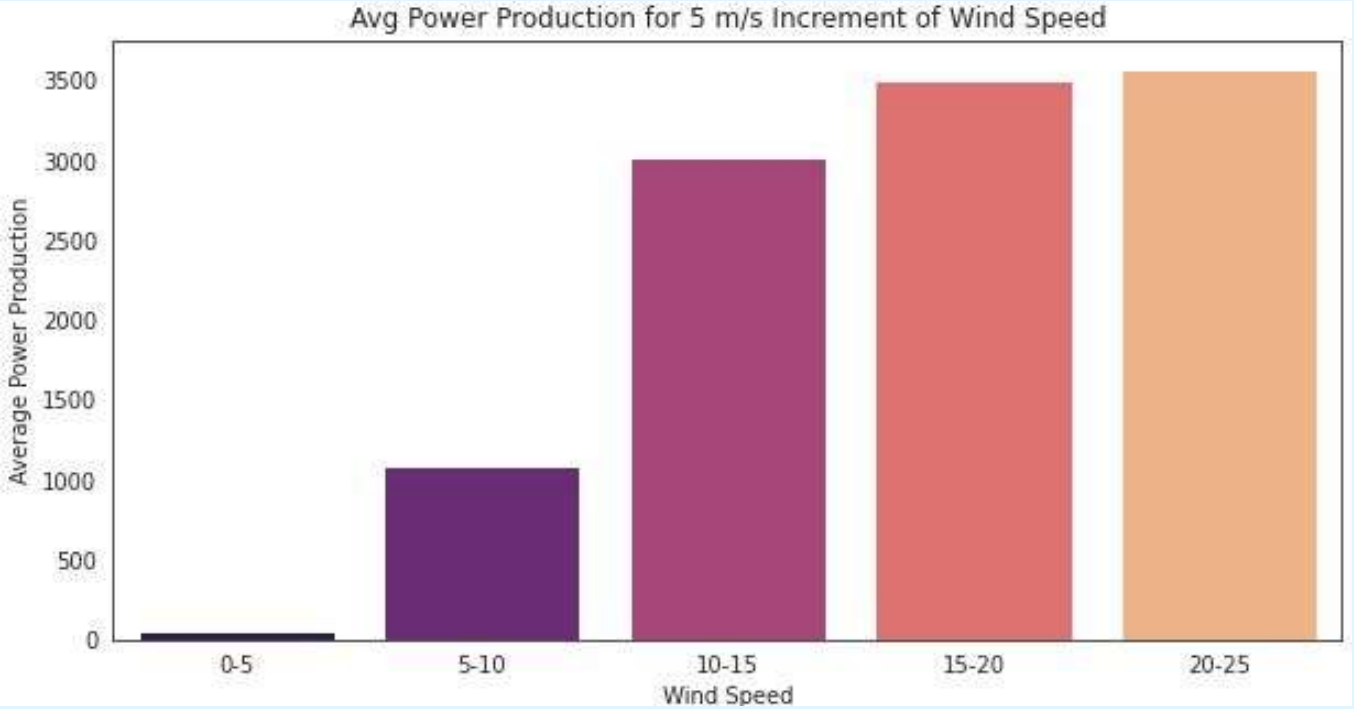
- In March, August and November, the average power production is higher.
- In April, June, July and August is lower.

Average power production by hour



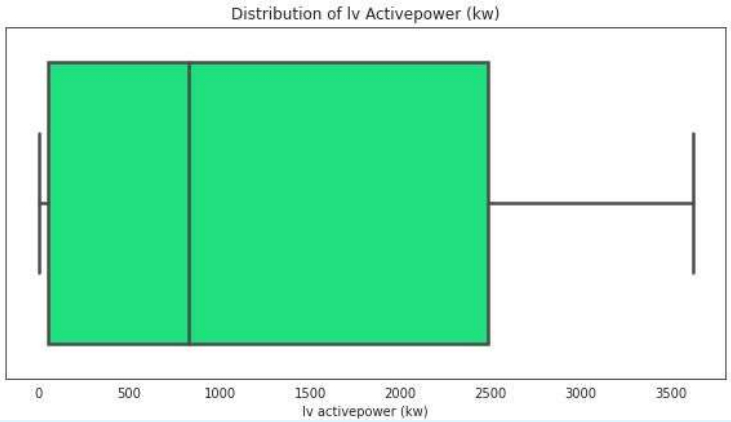
The average power production daily decays after 04:00 and start to increase after 12:00 and is higher between 16:00 and 24:00.

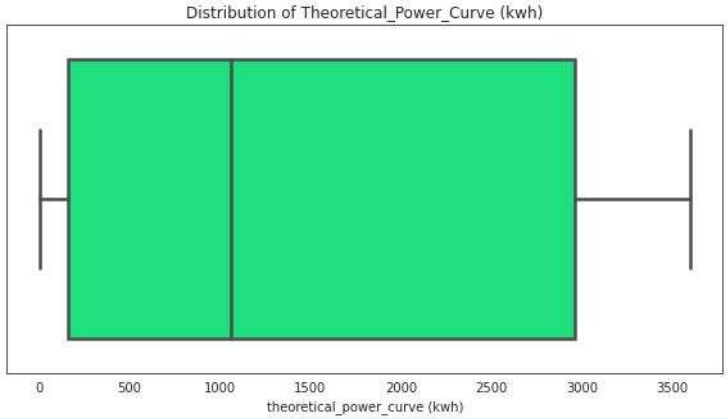
Relation between the wind speed and power production by increasing wind speed by 5m/s

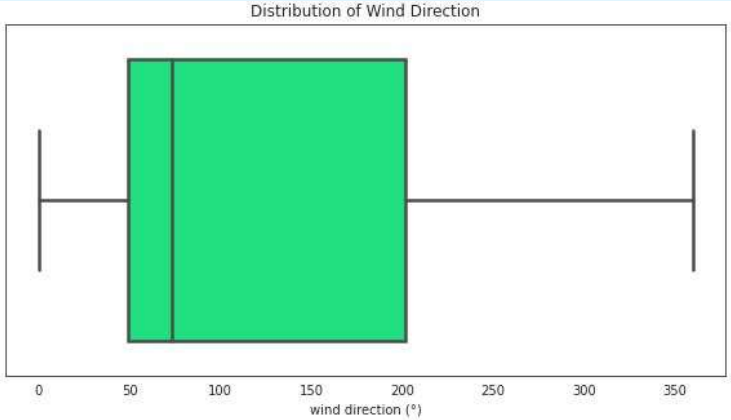


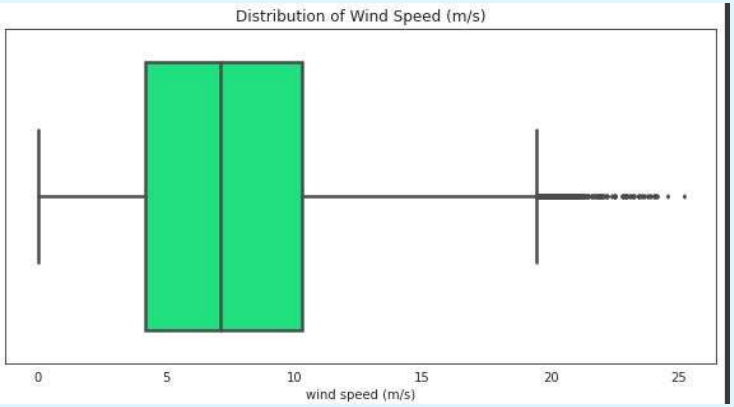
The power production reaches near a maximum level after the wind speed reaches 15 m/s.

Box Plot









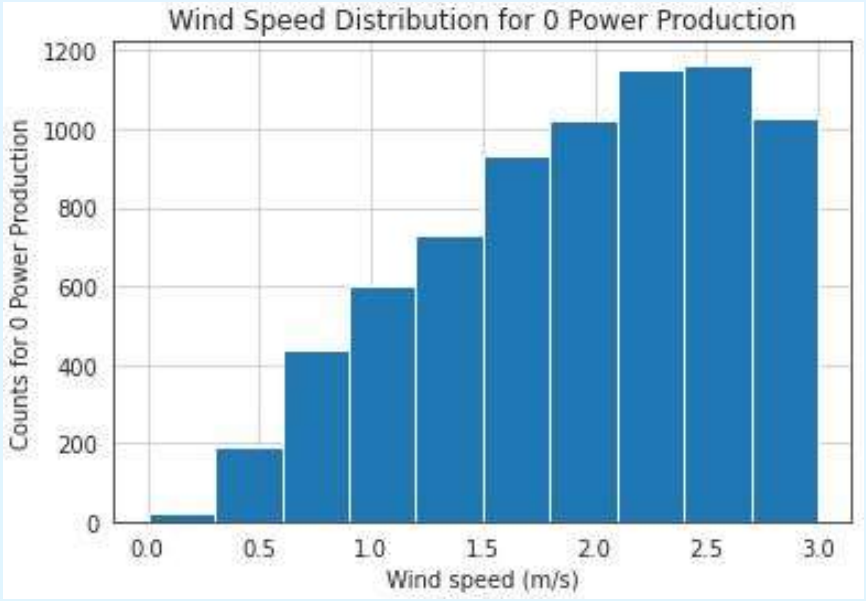
- There are some outliers for Wind Speed.
- Others don’t have any outliers.

Correlation among the variables

	lv activepower (kw)	wind speed (m/s)	theoretical_power_curve (kwh)	wind direction (°)	month	hour
lv activepower (kw)	1.00	0.91	0.95	-0.06	0.03	0.04
wind speed (m/s)	0.91	1.00	0.94	-0.08	-0.01	0.02
theoretical_power_curve (kwh)	0.95	0.94	1.00	-0.10	0.00	0.03
wind direction (°)	-0.06	-0.08	-0.10	1.00	-0.18	-0.02
month	0.03	-0.01	0.00	-0.18	1.00	0.00
hour	0.04	0.02	0.03	-0.02	0.00	1.00

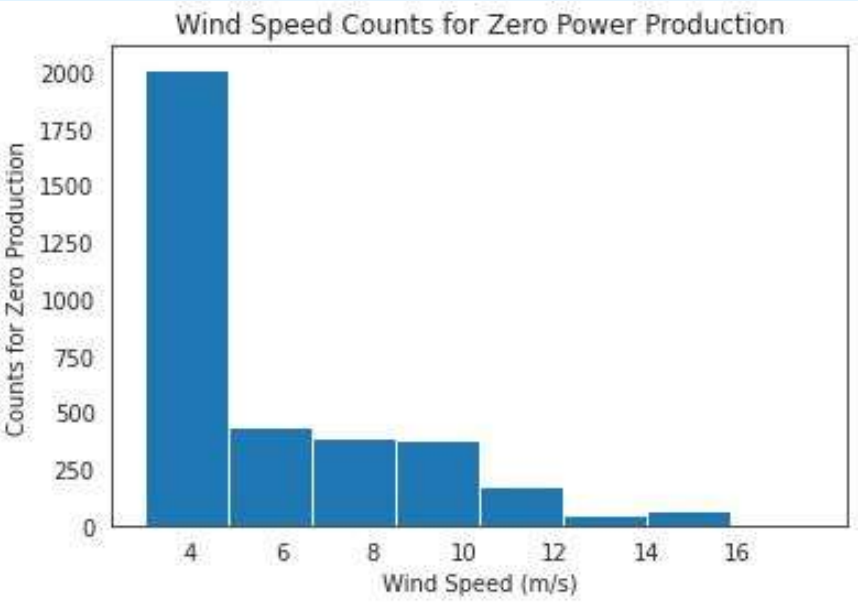
Wind speed and power production is correlated.

Wind Speed distribution for zero Power



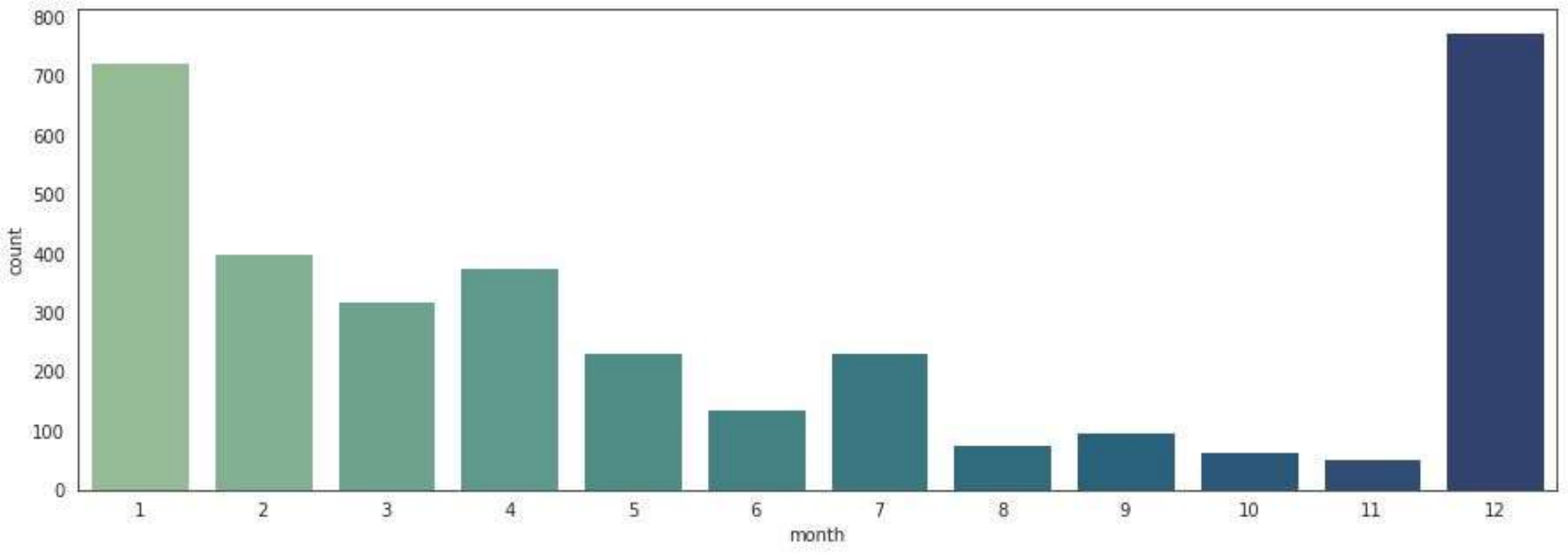
- Theoretical power curve has a limit of 3 m/s wind speed.
- There is no power output if the wind speed is less than 3 m/s.

Wind Speed counts for zero power production



- Theoretically, the wind speed barrier should be 4 m/s. However, there have been instances where no power is produced while the wind speed is higher.

Monthly distribution for zero power production



The wind turbine normally does not produce in the months of December and January.

Preprocessing

- Removing 3497 observations where windspeed is > 3 m/s but power production is zero as the reason for this observation is totally unknown.

```
df = df.filter(~((df['lv activepower (kw)'] == 0)
                 & (df['theoretical_power_curve (kwh)'] != 0)
                 & (df['wind speed (m/s)'] > 3)))
```

Preprocessing

```
# Create a pandas df for visualization
wind_speed = df.select('wind speed (m/s)').toPandas()

# Defining the quantiles and interquantile range
Q1 = wind_speed['wind speed (m/s)'].quantile(0.25)
Q3 = wind_speed['wind speed (m/s)'].quantile(0.75)
IQR = Q3-Q1
# Defining the lower and upper threshold values
lower = Q1 - 1.5*IQR
upper = Q3 + 1.5*IQR

print('Lower whisker: ', lower, ' Upper whisker: ', upper)
```

```
Lower whisker:  -4.576168060302599  Upper whisker:  19.50920486450172
```

- Calculating Upper and Lower whisker for Wind Speed.
- Finding total no of outliers for Wind Speed.

```
outlier_tf = (wind_speed['wind speed (m/s)'] < lower) | (wind_speed['wind speed (m/s)'] > upper)

print('Total Number of Outliers: ', len(wind_speed['wind speed (m/s)'][outlier_tf]))
```

```
Total Number of Outliers:  407
```

Preprocessing

```
from pyspark.sql import functions as F
df = df.withColumn('wind speed (m/s)',
                   F.when(F.col('wind speed (m/s)') > 19.50, 19)
                   .otherwise(F.col('wind speed (m/s)')))
```

- Treating outliers.
- Replace all values higher than 19.50 with 19

Preprocessing

```
# Preparing the independent variables (Features)
from pyspark.ml.feature import VectorAssembler

# Converting lv activepower (kw) variable as label
df = df.withColumn('label', df['lv activepower (kw)'])

# Defining the variables to be used
variables = ['month', 'hour', 'wind speed (m/s)', 'wind direction (°)']
vectorAssembler = VectorAssembler(inputCols = variables, outputCol = 'features')
va_df = vectorAssembler.transform(df)

# Combining features and label column
final_df = va_df.select('features', 'label')
final_df.show(10)
```

```
+--------------------+----------------+
|            features|           label|
+--------------------+----------------+
|[1.0,0.0,5.311336...|380.047790527343|
|[1.0,0.0,5.672166...| 453.76919555664|
|[1.0,0.0,5.216036...|306.376586914062|
|[1.0,0.0,5.659674...|419.645904541015|
|[1.0,0.0,5.577940...|380.650695800781|
|[1.0,0.0,5.604052...|402.391998291015|
|[1.0,1.0,5.793007...|447.605712890625|
|[1.0,1.0,5.306049...|     387.2421875|
|[1.0,1.0,5.584629...|463.651214599609|
|[1.0,1.0,5.523228...|439.725708007812|
+--------------------+----------------+
only showing top 10 rows
```

- Vector assembler is used to combine numerous columns into a vector column, which was then produced as a single feature column.
- All of the necessary columns have inputs.

Model Preparation

```
splits = final_df.randomSplit([0.8, 0.2])
train_df = splits[0]
test_df = splits[1]

print('Train dataset: ', train_df.count())
print('Test dataset : ', test_df.count())
```

```
Train dataset:  37840
Test dataset :  9193
```

- For machine learning model creation and verification, divide the data into training and test datasets.
- Train dataset has 37840 rows.
- Test dataset has 9193 rows of data.

Model Building

```
from pyspark.ml.regression import LinearRegression
lr= LinearRegression(featuresCol = 'features', labelCol='label')
lr_model=lr.fit(train_df)
y_pred=lr_model.transform(test_df)
y_pred.select('label','prediction').show(10)

#evaluation metrics
from pyspark.ml.evaluation import RegressionEvaluator

evaluator = RegressionEvaluator(predictionCol='prediction', labelCol='label')

print('R2 SCORE : ', evaluator.evaluate(y_pred, {evaluator.metricName: 'r2'}))
print('RMSE     : ', evaluator.evaluate(y_pred, {evaluator.metricName: 'rmse'}))
```

```
R2 SCORE :  0.8885649287821278
RMSE     :  435.2652768367119
```

- For implementing the model from pyspark ml regression model imported the Linear Regression.
- Created an instance of that with mentioning the feature columns , label columns and model type.
- Fit the model into the train data and performed prediction with test data.
- R2 Score and RMSE obtained as 0.8885 and 435.26 respectively.

Model Building

```
from pyspark.ml.regression import RandomForestRegressor
rf= RandomForestRegressor(featuresCol = 'features', labelCol='label')
rf_model=rf.fit(train_df)
y_pred=rf_model.transform(test_df)
y_pred.select('label','prediction').show(10)

#evaluation metrics
from pyspark.ml.evaluation import RegressionEvaluator

evaluator = RegressionEvaluator(predictionCol='prediction', labelCol='label')

print('R2 SCORE : ', evaluator.evaluate(y_pred, {evaluator.metricName: 'r2'}))
print('RMSE     : ', evaluator.evaluate(y_pred, {evaluator.metricName: 'rmse'}))
```

```
R2 SCORE :  0.9690603065956375
RMSE     :  229.35123245330666
```

- In this case another model was built which is Random Forest Regressor.
- In this model r2 score and RMSE obtained as 0.9690 and 229.35 respectively.
- This model is performing better than the Linear Regression model.

Conclusions

- From the Exploratory Data Analysis, we could generate insight from the data - how each of the features are related to the target.
- From the evaluation of two models that Random Forest Regressor performed better than Linear Regression.
- R2 Score and RMSE scores are 0.96 and 229.35 respectively for Random Forest Regressor.
- Though this model is performing better but Deep Learning models can be more useful to obtain the correlations between meteorological features and power generation.

THANK YOU