Wind Turbine Power Prediction

In this study I am going to predict a wind turbine power production by using the wind speed, wind direction, month and hour data.

The dataset consists of 50530 observations. In order to demonstrate my data science skills with big data, I am going to use Pyspark library.

The dataset contains:

Date/Time (for 10 minutes intervals) V ActivePower (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) TheoreticalPowerCurve (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 automaticly)

Aim of the Study:-

My aim is to predict wind turbine power production from the wind speed, wind direction, month of the year and the hour of the day.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('white')

from warnings import filterwarnings
filterwarnings('ignore')

# Importing pyspark libraries
import pyspark
from pyspark.sql import SparkSession
from pyspark.conf import SparkConf
from pyspark import SparkContext

# Configuration of Spark Session
spark = SparkSession.builder.master("local").appName("wind_turbine_project").getOrCreate()
sc = spark.sparkContext
sc
```

```
24/04/26 23:04:58 WARN Utils: Your hostname, vivek-VirtualBox resolves to a loopback adc
24/04/26 23:04:58 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLev
24/04/26 23:04:59 WARN NativeCodeLoader: Unable to load native-hadoop library for your p
```

SparkContext

[Spark UI]

Version
    v3.5.1
Master
    local
AppName
    wind_turbine_project

Reading the Dataset

```
# Reading the dataset as Spark DataFrame
spark_df = spark.read.csv('/home/hduser/Downloads/T1.csv', header=True, inferSchema=True)

# Caching the dataset
spark_df.cache()

# Converting all the column names to lower case
spark_df = spark_df.toDF(*[c.lower() for c in spark_df.columns])

print('Show the first 5 rows')
print(spark_df.show(5))
print()
print('What are the variable data types?')
print(spark_df.printSchema())
print()
print('How many observations do we have?')
print(spark_df.count())
```

```
Show the first 5 rows
+----------------+------------------+----------------+-----------------------------+------------------+
|       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

None

What are the variable data types?
root
 |-- date/time: string (nullable = true)
 |-- lv activepower (kw): double (nullable = true)
 |-- wind speed (m/s): double (nullable = true)
 |-- theoretical_power_curve (kwh): double (nullable = true)
 |-- wind direction (°): double (nullable = true)

None

How many observations do we have?
50530
```

Exploratory Data Analysis

```
# Extracting a substring from columns to create month and hour variables

from pyspark.sql.functions import substring
spark_df = spark_df.withColumn("month", substring("date/time", 4,2))
spark_df = spark_df.withColumn("hour", substring("date/time", 12,2))

# Converting string month and hour variables to integer
from pyspark.sql.types import IntegerType
spark_df = spark_df.withColumn('month', spark_df.month.cast(IntegerType()))
spark_df = spark_df.withColumn('hour', spark_df.hour.cast(IntegerType()))

print(spark_df.show(5))
```

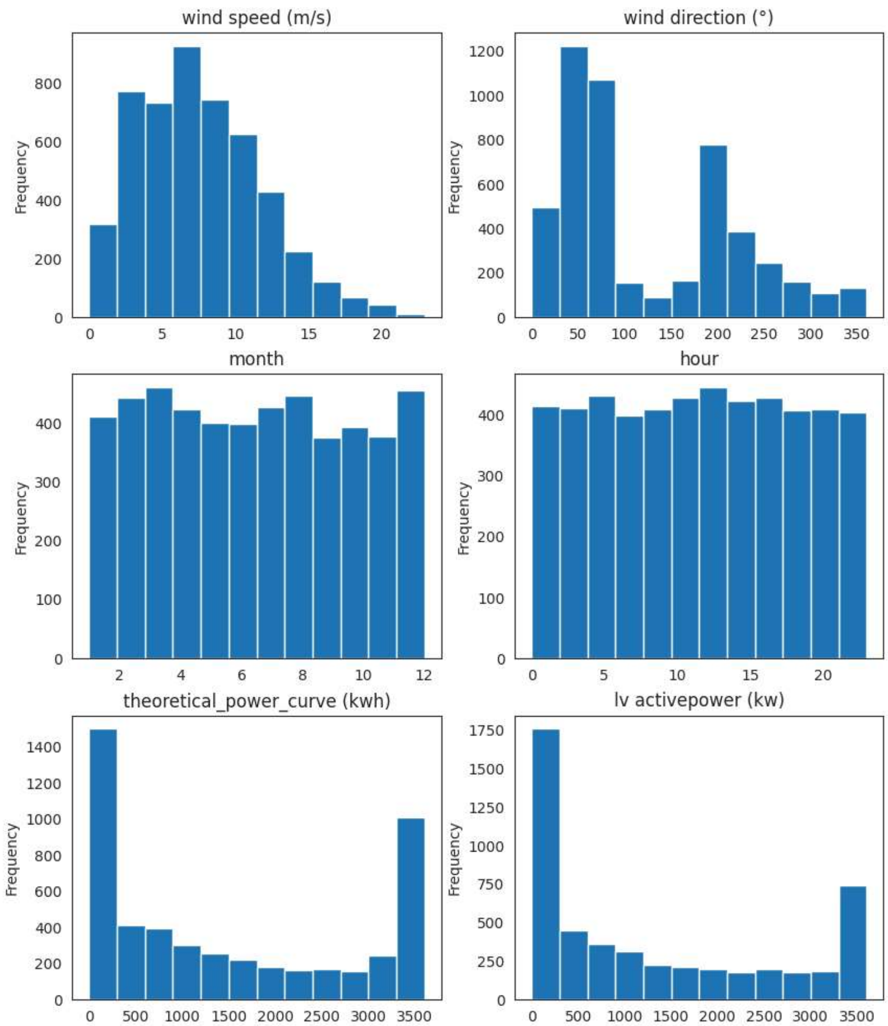
```
+----------------+------------------+----------------+-----------------------------+------------------+-----+----+
|       date/time|lv activepower (kw)|wind speed (m/s)|theoretical_power_curve (kwh)|wind direction (°)|month|hour|
+----------------+------------------+----------------+-----------------------------+------------------+-----+----+
|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

None
```

```
pd.options.display.float_format = '{:.2f}'.format
spark_df.select('wind speed (m/s)', 'theoretical_power_curve (kwh)', 'lv activepower (kw)').toPandas().describe()
```

	wind speed (m/s)	theoretical_power_curve (kwh)	lv activepower (kw)
count	50530.00	50530.00	50530.00
mean	7.56	1492.18	1307.68
std	4.23	1368.02	1312.46
min	0.00	0.00	-2.47
25%	4.20	161.33	50.68
50%	7.10	1063.78	825.84
75%	10.30	2964.97	2482.51
max	25.21	3600.00	3618.73

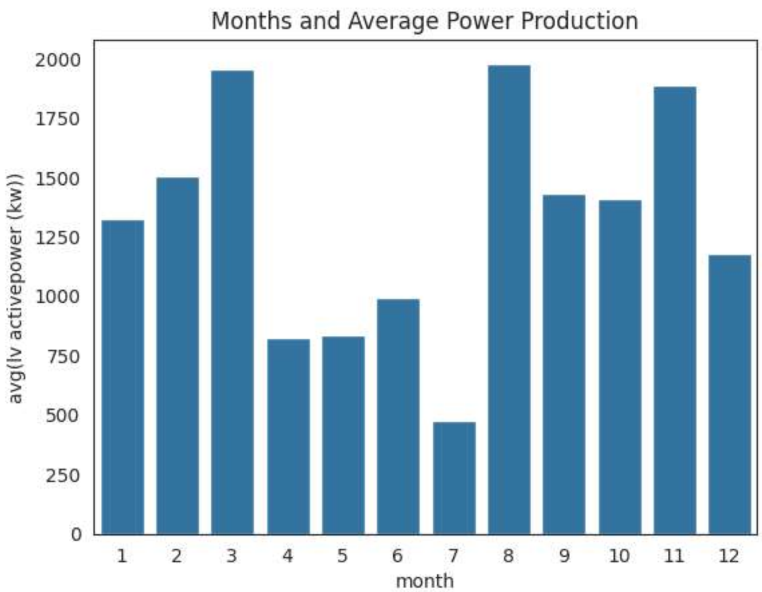
random sample from my big data.

```
# Taking a random sample from the big data
sample_df = spark_df.sample(withReplacement=False, fraction=0.1, seed=42).toPandas()

# Visualizing the distributions with the sample data
columns = ['wind speed (m/s)', 'wind direction (°)', 'month', 'hour', 'theoretical_power_curve (kwh)', 'lv activepower (kw)']
i=1
plt.figure(figsize=(10,12))
for each in columns:
    plt.subplot(3,2,i)
    sample_df[each].plot.hist(bins=12)
    plt.title(each)
    i += 1
```



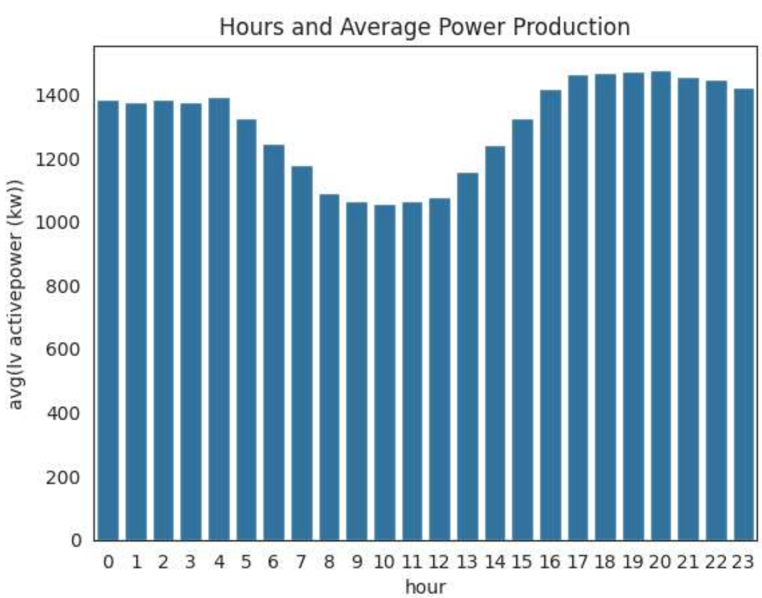
Difference between the months for average power production

```
# Average power production by month
monthly = spark_df.groupby('month').mean('lv activepower (kw)').sort('avg(lv activepower (kw))').toPandas()
sns.barplot(x='month', y='avg(lv activepower (kw))', data=monthly)
plt.title('Months and Average Power Production');
```



Difference between the hours for average power production

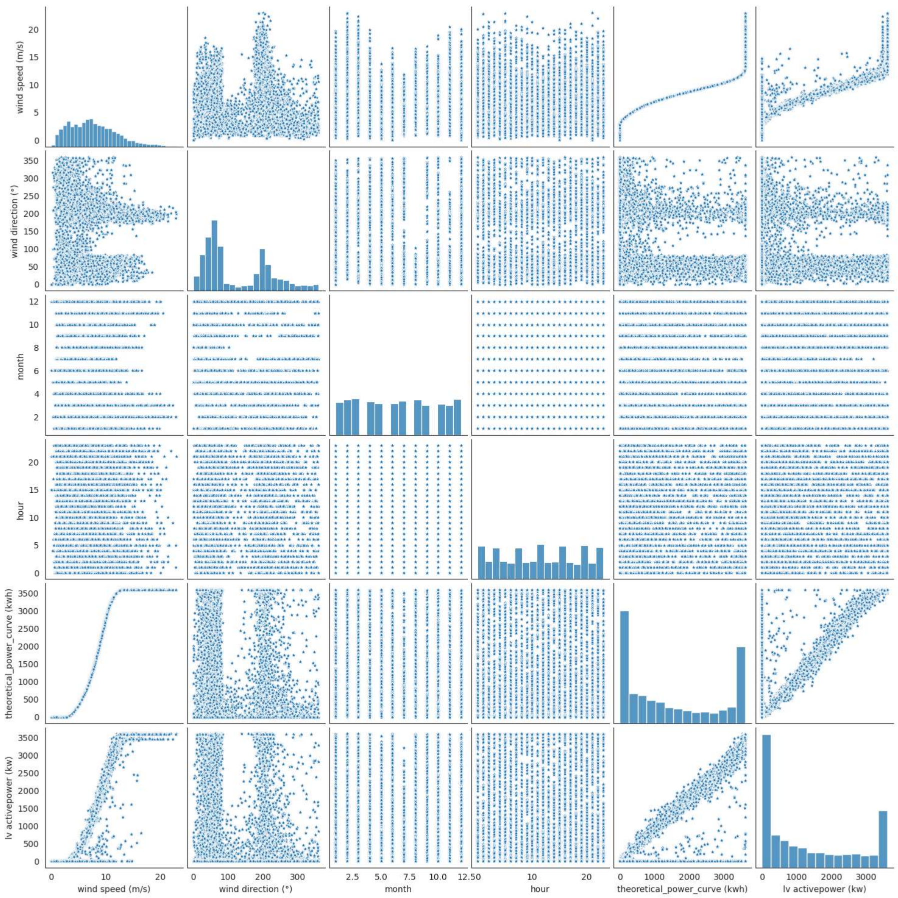
```
# Average power production by hour
hourly = spark_df.groupby('hour').mean('lv activepower (kw)').sort('avg(lv activepower (kw))').toPandas()
sns.barplot(x='hour', y='avg(lv activepower (kw))', data=hourly)
plt.title('Hours and Average Power Production');
```



correlation between the wind speed, wind direction and power production

```
display(sample_df[columns].corr())
sns.pairplot(sample_df[columns], markers='*');
```

	wind speed (m/s)	wind direction (°)	month	hour	theoretical_power_curve (kwh)	activepo (
wind speed (m/s)	1.00	-0.08	-0.01	0.03	0.95	(
wind direction (°)	-0.08	1.00	-0.18	0.00	-0.11	-(
month	-0.01	-0.18	1.00	-0.01	-0.00	(
hour	0.03	0.00	-0.01	1.00	0.03	(
theoretical_power_curve (kwh)	0.95	-0.11	-0.00	0.03	1.00	(
lv activepower (kw)	0.91	-0.06	0.04	0.03	0.94	1



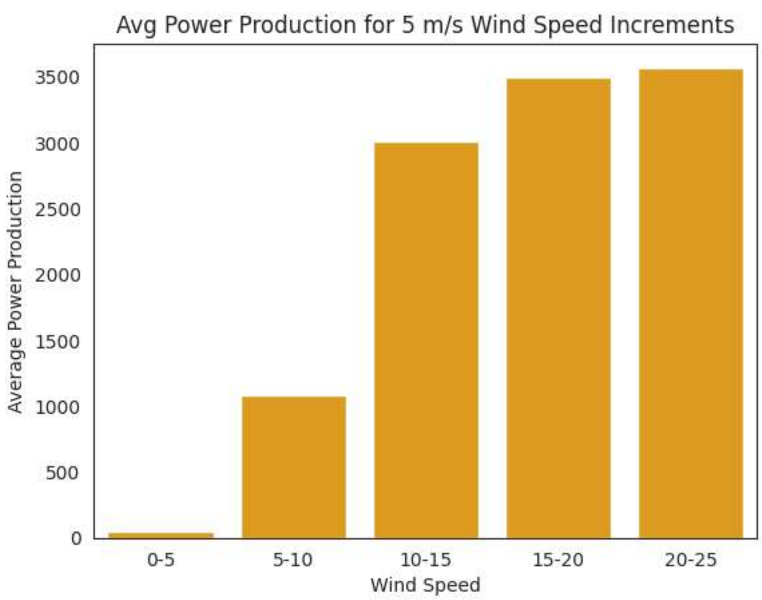
observation from above Graph:-

Wind speed and power production is highly correlated as one would expect.

We can see there are lower level power production for some wind directions.

average power production level for different wind speeds

```
# Finding average power production for 5 m/s wind speed increments
wind_speed = []
avg_power = []
for i in [0,5,10,15,20]:
    avg_value = spark_df.filter((spark_df['wind speed (m/s)'] > i)
                                & (spark_df['wind speed (m/s)'] <= i+5))\
                                .agg({'lv activepower (kw)':'mean'}).collect()[0][0]
    avg_power.append(avg_value)
    wind_speed.append(str(i) + '-' + str(i+5))

sns.barplot(x=wind_speed, y=avg_power, color='orange')
plt.title('Avg Power Production for 5 m/s Wind Speed Increments')
plt.xlabel('Wind Speed')
plt.ylabel('Average Power Production');
```

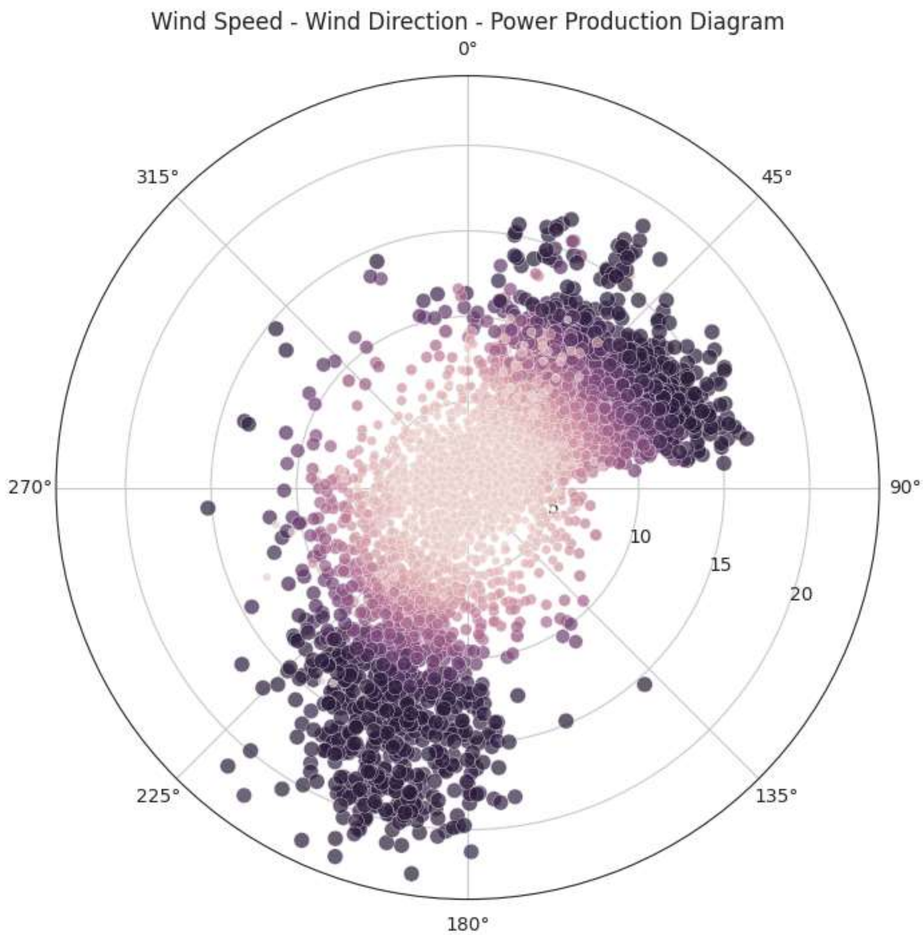


From the graph above we can see the power production reaches near a maximum level after the wind speed reaches 15 m/s.

power production for different wind directions and speeds

create a polar diagram with wind speed, wind direction and power production from the sample data.

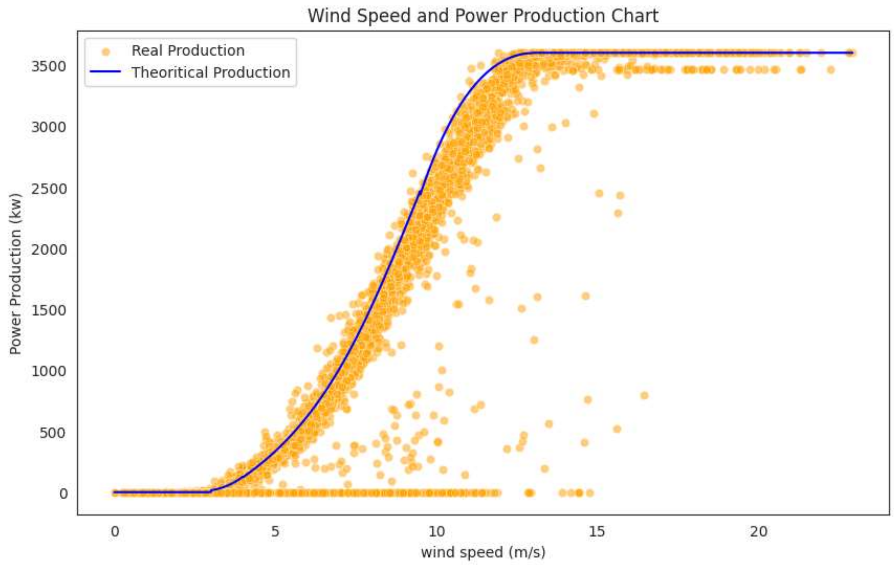
```
# Creating the polar diagram
from math import radians

plt.figure(figsize=(8,8))
ax = plt.subplot(111, polar=True)
# Inside circles are the wind speed and marker color and size represents the amount of power production
sns.scatterplot(x=[radians(x) for x in sample_df['wind direction (°)']],
                y=sample_df['wind speed (m/s)'],
                size=sample_df['lv activepower (kw)'],
                hue=sample_df['lv activepower (kw)'],
                alpha=0.7, legend=None)
# Setting the polar diagram's top represents the North
ax.set_theta_zero_location('N')
# Setting -1 to start the wind direction clockwise
ax.set_theta_direction(-1)
# Setting wind speed labels in a better position to see
ax.set_rlabel_position(110)
plt.title('Wind Speed - Wind Direction - Power Production Diagram')
plt.ylabel(None);
```



We can see that the wind turbine produces more power if the wind blows from the directions between 000-090 and 180-225 degrees.

manufacturer's theoritical power production curve fit well with the real production

```
plt.figure(figsize=(10,6))
sns.scatterplot(x='wind speed (m/s)', y='lv activepower (kw)', color='orange', label='Real Production', alpha=0.5, data=sample_df)
sns.lineplot(x='wind speed (m/s)', y='theoretical_power_curve (kwh)', color='blue', label='Theoritical Production', data=sample_df)
plt.title('Wind Speed and Power Production Chart')
plt.ylabel('Power Production (kw)');
```



From the graph above, we can see the theoritical power production curve generally fits well with the real production.

We can see the power production reaches a maximum level and continues in a straight line if the wind speed reaches to 15 m/s.

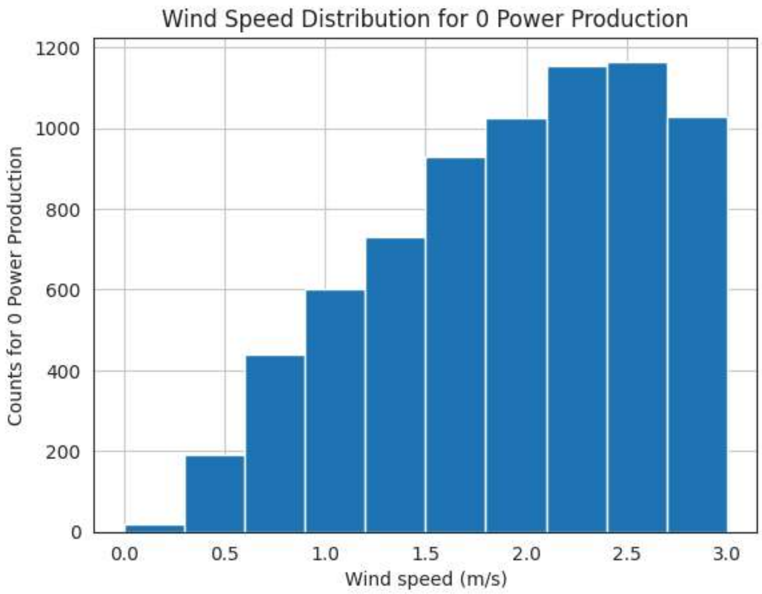
Also we can see there are some 0 power production, even the wind speed is higher than 5 m/s. I want to investigate the reason.

But before what is the minimum wind speed for theoritical power production curve

wind speed threshold value for zero theorical power

```
# Filter the big data where the real and theoritical power productions are equal to 0
zero_theo_power = spark_df.filter((spark_df['lv activepower (kw)'] == 0)
                                  & (spark_df['theoretical_power_curve (kwh)'] == 0)).toPandas()

display(zero_theo_power[['wind speed (m/s)', 'theoretical_power_curve (kwh)', 'lv activepower (kw)']].sample(5))

# Let's see the wind speed distribution for 0 power production
zero_theo_power['wind speed (m/s)'].hist()
plt.title('Wind Speed Distribution for 0 Power Production')
plt.xlabel('Wind speed (m/s)')
plt.ylabel('Counts for 0 Power Production');
```

	wind speed (m/s)	theoretical_power_curve (kwh)	lv activepower (kw)
921	0.69	0.00	0.00
4598	3.00	0.00	0.00
1962	2.38	0.00	0.00
1986	0.69	0.00	0.00
2397	2.59	0.00	0.00



We can see from above, limit for the theoritical power curve is 3 m/s wind speed. If the wind speed is below 3 m/s model doesn't expect any power production.

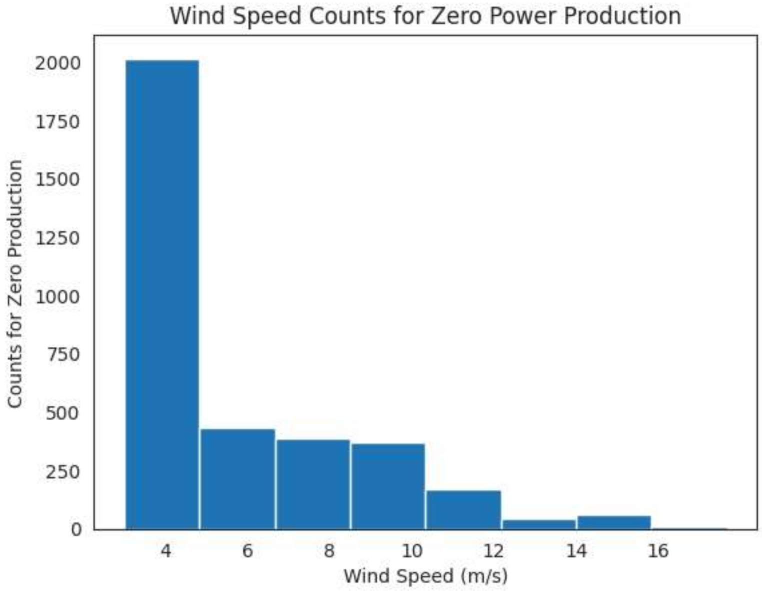
But there are some observations for 0 power production even the wind speed is more than 3

power production in some observations while the wind speed is higher than 3 m/s

```
# Observations for the wind speed > 3m/s and power production = 0,
# While theoritically there should be power production
zero_power = spark_df.filter((spark_df['lv activepower (kw)'] == 0)
                            & (spark_df['theoretical_power_curve (kwh)'] != 0)
                            & (spark_df['wind speed (m/s)'] > 3)).toPandas()
display(zero_power.head())
print('No of Observations (while Wind Speed > 3 m/s and Power Production = 0): ', len(zero_power))
```

	date/time	lv activepower (kw)	wind speed (m/s)	theoretical_power_curve (kwh)	wind direction (°)	month	hour
0	03 01 2018 15:40	0.00	3.74	83.99	245.07	1	15
1	03 01 2018 16:40	0.00	3.03	17.18	221.09	1	16
2	03 01 2018 16:50	0.00	3.20	25.43	232.68	1	16

```
zero_power['wind speed (m/s)'].plot.hist(bins=8)
plt.xlabel('Wind Speed (m/s)')
plt.ylabel('Counts for Zero Production')
plt.title('Wind Speed Counts for Zero Power Production')
plt.xticks(ticks=np.arange(4,18,2));
```



It looks like theoritically wind speed threshold should be 4 m/s. But there are also other observations with zero power production while the wind speed is higher.

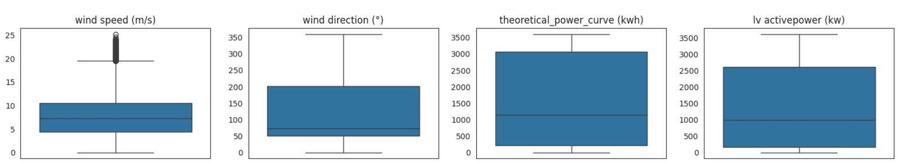
It is usually in December and January when the wind turbine doesn't produce production.

Because I cannot decide if these zero power productions are caused by maintenance periods or something else, I am going to accept those 3497 observations as outliers and remove them from the dataset.

```
# Excluding the observations meeting the filter criterias
spark_df = spark_df.filter(~((spark_df['lv activepower (kw)'] == 0)
                            & (spark_df['theoretical_power_curve (kwh)'] != 0)
                            & (spark_df['wind speed (m/s)'] > 3)))
```

outliers

```
columns = ['wind speed (m/s)', 'wind direction (°)', 'theoretical_power_curve (kwh)', 'lv activepower (kw)']
i=1
plt.figure(figsize=(20,3))
for each in columns:
    df = spark_df.select(each).toPandas()
    plt.subplot(1,4,i)
    sns.boxplot(df)
    plt.title(each)
    i += 1
```



From the graphs above I can see there are some outliers in the wind speed data.

I will find the upper and lower threshold values for the wind speed data, and I will analyze the outliers.

```
# Create a pandas df for visualization
wind_speed = spark_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('Quantile (0.25): ', Q1, '  Quantile (0.75): ', Q3)
print('Lower threshold: ', lower, ' Upper threshold: ', upper)
```

```
Quantile (0.25):  4.45584678649902   Quantile (0.75):  10.4771900177001
Lower threshold:  -4.576168060302599  Upper threshold:  19.50920486450172
```

```
# Fancy indexing for outliers
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]))
print('--'*15)
print('Some Examples of Outliers:')
print(wind_speed['wind speed (m/s)'][outlier_tf].sample(10))
```

```
Total Number of Outliers:  407
------------------------------
Some Examples of Outliers:
2101    20.80
46347   20.46
3504    21.10
3480    20.16
10895   21.33
3512    22.98
3494    21.19
46355   20.02
7456    21.15
3436    20.14
Name: wind speed (m/s), dtype: float64
```

It is a rare event for wind speed to be over 19 m/s in our dataset.

Out of 47033, there is only 407 observations while the wind speed is over 19 m/s.

average power production for these high wind speed

```
spark_df.select('wind speed (m/s)', 'lv activepower (kw)')\
.filter(spark_df['wind speed (m/s)'] >= 19)\
.agg({'lv activepower (kw)':'mean'}).show()
```

```
+------------------------+
|avg(lv activepower (kw))|
+------------------------+
|      3566.4634427974706|
+------------------------+
```

So instead of erasing the outliers, I am going to set the wind speed as 19 m/s for those observations.

```
from pyspark.sql import functions as F
spark_df = spark_df.withColumn('wind speed (m/s)',
                               F.when(F.col('wind speed (m/s)') > 19.447, 19)
                               .otherwise(F.col('wind speed (m/s)')))
spark_df.count()
```

```
47033
```

generalized criterias for power production

It is important to understand the pattern in the data. We should learn the data before the machine.

1. We saw from the graph that in March, August and November, the average power production is higher.
2. The average power production is higher daily between 16:00 and 24:00.
3. The power production is higher when the wind blows from the directions between 000-090 and 180-225 degrees.

.

predict a high and low level of power production from the criterias above before ML algorithm

```
# High level power production
spark_df.filter(((spark_df['month'] == 3) | (spark_df['month'] == 8) | (spark_df['month'] == 11))
                & ((spark_df['hour'] >= 16) | (spark_df['hour'] <= 24))
                & ((spark_df['wind direction (°)'] > 0) | (spark_df['wind direction (°)'] < 90))
                & ((spark_df['wind direction (°)'] > 180) | (spark_df['wind direction (°)'] < 225))
               ).agg({'lv activepower (kw)':'mean'}).show()
```

```
+------------------------+
|avg(lv activepower (kw))|
+------------------------+
|      2013.4446757880403|
+------------------------+
```

Data Preparation for ML Algorithms¶

After analysing and understanding the dataset, we can build a ML regression model to predict wind turbine power production by using the wind speed, wind direction, month of the year and hour of the day.

Using ML algorithms with Spark is a bit different from well known Sckitlearn library.

We need to feed the model with a dataframe made of variables compressed in vectors called as 'features', and target variable as 'label'. For these convertions I am going to use VectorAssembler method from Pyspark.

```
# Low level power production
spark_df.filter((spark_df['month'] == 7)
                & ((spark_df['hour'] >= 9) | (spark_df['hour'] <= 11))
                & ((spark_df['wind direction (°)'] > 90) | (spark_df['wind direction (°)'] < 160))
               ).agg({'lv activepower (kw)':'mean'}).show()
```

```
+------------------------+
|avg(lv activepower (kw))|
+------------------------+
|       503.1644205414878|
+------------------------+
```

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

# Converting lv activepower (kw) variable as label
spark_df = spark_df.withColumn('label', spark_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(spark_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
```

Train Test Split

Now we can split our dataset into train and test datasets.

```
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:  37591
Test dataset :  9442
```

Creating the Initial Model

I am going to use GBT regressor for this study.

```
from pyspark.ml.regression import GBTRegressor

# Creating the gbm regressor object
gbm = GBTRegressor(featuresCol='features', labelCol='label')

# Training the model with train data
gbm_model = gbm.fit(train_df)

# Predicting using the test data
y_pred = gbm_model.transform(test_df)

# Initial look at the target and predicted values
y_pred.select('label', 'prediction').show(20)
```

```
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.spark.util.SizeEstimator$ (file:/home/hduser/.local/lib/python3.10/site-packages/pyspar
WARNING: Please consider reporting this to the maintainers of org.apache.spark.util.SizeEstimator$
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
+----------------+------------------+
```