IBM-Build-A-Thon

Predicting the Energy Output of Wind Turbine Based on Weather Conditions

About

Wind energy plays an increasing role in the supply of energy worldwide. The energy output of a wind farm is highly dependent on the weather 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. In this paper, we predict energy prediction based on weather data and analyze the important parameters as well as their correlation on the energy output.

Dataset

	Date/Time	LV ActivePower (kW)	Wind Speed (m/s)	Theoretical_Power_Curve (KWh)	Wind Direction (°)
0	01 01 2018 00:00	380.047791	5.311336	416.328908	259.994904
1	01 01 2018 00:10	453.769196	5.672167	519.917511	268.641113
2	01 01 2018 00:20	306.376587	5.216037	390.900016	272.564789
3	01 01 2018 00:30	419.645905	5.659674	516.127569	271.258087
4	01 01 2018 00:40	380.650696	5.577941	491.702972	265.674286

The data's in the file are:

- Date/Time (for 10 minutes intervals)
- LV 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)
- 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 automaticly)

Dependent variable - LV Active Power

EDA and Data Cleaning

Data Summary

	LV ActivePower (kW)	Wind Speed (m/s)	Theoretical_Power_Curve (KWh)	Wind Direction (°)
count	50530.000000	50530.000000	50530.000000	50530.000000
mean	1307.684332	7.557952	1492.175463	123.687559
std	1312.459242	4.227166	1368.018238	93.443736
min	-2.471405	0.000000	0.000000	0.000000
25%	50.677890	4.201395	161.328167	49.315437
50%	825.838074	7.104594	1063.776283	73.712978
75%	2482.507568	10.300020	2964.972462	201.696720
max	3618.732910	25.206011	3600.000000	359.997589

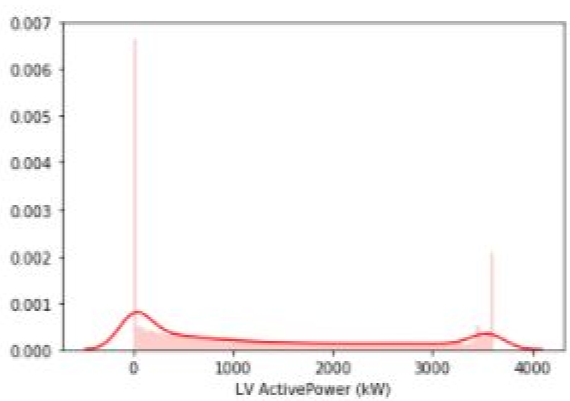
Data Info

```
#Analyzing the data
df_data_1.info()
df_data_1.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50530 entries, 0 to 50529
Data columns (total 5 columns):
Date/Time                        50530 non-null object
LV ActivePower (kW)              50530 non-null float64
Wind Speed (m/s)                 50530 non-null float64
Theoretical_Power_Curve (KWh)    50530 non-null float64
Wind Direction (°)               50530 non-null float64
dtypes: float64(4), object(1)
memory usage: 1.9+ MB
```

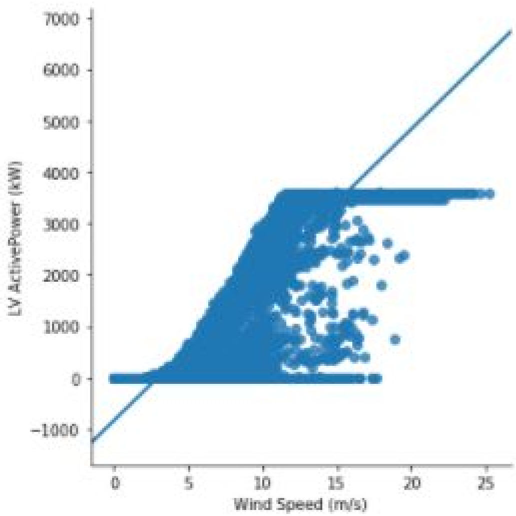
Data Visualization

```
In [39]: sns.distplot(df_data_1['LV ActivePower (kW)'], color='r', bins=100,hist_kws={'alpha':0.2})
Out[39]: <matplotlib.axes._subplots.AxesSubplot at 0x7fdaea999eb8>
```



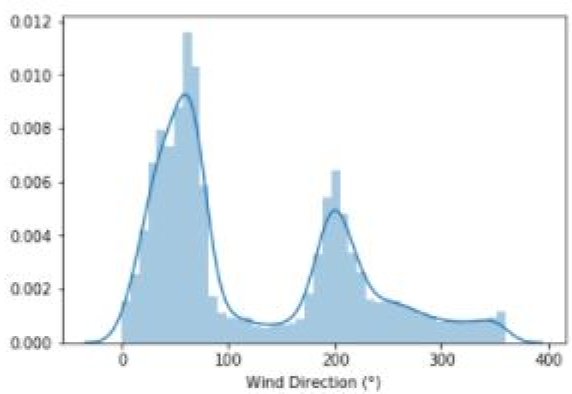
In [40]:
```
sns.lmplot(x='Wind Speed (m/s)',y='LV ActivePower (kW)',data=df_data_1)
```

Out[40]: <seaborn.axisgrid.FacetGrid at 0x7fdaebe3eef0>

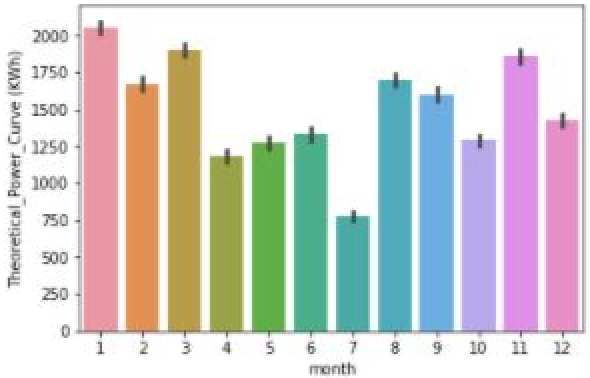


In [41]:
```
sns.distplot(a=df_data_1['Wind Direction (°)'])
```

Out[41]: <matplotlib.axes._subplots.AxesSubplot at 0x7fdaea2729b0>



```
In [49]: sns.barplot(x=df_data_1['month'],y=df_data_1['Theoretical_Power_Curve (KWh)'])
Out[49]: <matplotlib.axes._subplots.AxesSubplot at 0x7fdacd4e2fd0>
```



Data Cleaning

Removing rows in which wind speed is greater than 3.5(Threshold Wind Speed after which windmill should start giving energy output) but still L

```
In [52]: df_data_1['LV ActivePower (kW)'][(df_data_1['LV ActivePower (kW)']==0) & (df_data_1['Wind Speed (m/s)']>3.5) ].count()
Out[52]: 2217
```

```
In [53]: df_data_1.drop(df_data_1[(df_data_1['LV ActivePower (kW)']==0) & (df_data_1['Wind Speed (m/s)']>3.5) ].index,inplace=True)
```

Feature Engineering

Extracting Months from Date and clustering the Direction columns

In [42]:
```
from datetime import datetime
```

In [43]:
```
df_data_1['month'] = pd.DatetimeIndex(df_data_1['Date/Time']).month
```

In [44]:
```
def mean_direction(x):
    list=[]
    i=15
    while i<=375:
        list.append(i)
        i+=30

    for i in list:
        if x < i:
            x=i-15
            if x==360:
                return 0
            else:
                return x
```

In [45]:
```
df_data_1["mean_Direction"]=df_data_1["Wind Direction (°)"].apply(mean_direction)
df_data_1.head()
```

Out[45]:

	Date/Time	LV ActivePower (kW)	Wind Speed (m/s)	Theoretical_Power_Curve (KWh)	Wind Direction (°)	month	mean_Direction
0	01 01 2018 00:00	380.047791	5.311336	416.328908	259.994904	1	270
1	01 01 2018 00:10	453.769196	5.672167	519.917511	268.641113	1	270
2	01 01 2018 00:20	306.376587	5.216037	390.900016	272.564789	1	270
3	01 01 2018 00:30	419.645905	5.659674	516.127569	271.258087	1	270
4	01 01 2018 00:40	380.650696	5.577941	491.702972	265.674286	1	270

In [46]:
```
def find_direction(x):
    if x==0:
        return 1
    if x==30:
        return 2
    if x==60:
        return 3
    if x==90:
        return 4
    if x==120:
        return 5
    if x==150:
        return 6
    if x==180:
        return 7
    if x==210:
        return 8
    if x==240:
        return 9
    if x==270:
        return 10
    if x==300:
        return 11
    if x==330:
        return 12
```

```
In [47]: df_data_1["Direction"]=df_data_1["mean_Direction"].apply(find_direction)
df_data_1.head()
```

Out[47]:

	Date/Time	LV ActivePower (kW)	Wind Speed (m/s)	Theoretical_Power_Curve (KWh)	Wind Direction (°)	month	mean_Direction	Direction
0	01 01 2018 00:00	380.047791	5.311336	416.328908	259.994904	1	270	10
1	01 01 2018 00:10	453.769196	5.672167	519.917511	268.641113	1	270	10
2	01 01 2018 00:20	306.376587	5.216037	390.900016	272.564789	1	270	10
3	01 01 2018 00:30	419.645905	5.659674	516.127569	271.258087	1	270	10
4	01 01 2018 00:40	380.650696	5.577941	491.702972	265.674286	1	270	10

Adding Inertial factor into the table

```
df_data_1['diff']=df_data_1['Wind Speed (m/s)']-df_data_1['Wind Speed (m/s)'].shift(1)[:]
df_data_1['diff']
```

Scaling

```
In [35]: from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train_last = sc.fit_transform(X_train)
X_test_last = sc.transform(X_test)
```

Correlations

```
In [59]: #corelation of numerical data
num_corr=df_data_1.corr()['LV ActivePower (kW)'][:]
print(num_corr)
#print coerr
```

```
LV ActivePower (kW)              1.000000
Wind Speed (m/s)                 0.938366
Theoretical_Power_Curve (KWh)    0.980337
Wind Direction (°)              -0.072508
month                           -0.038940
mean_Direction                  -0.036772
Direction                       -0.036772
diff                             0.066683
Name: LV ActivePower (kW), dtype: float64
```

Output and User Interface