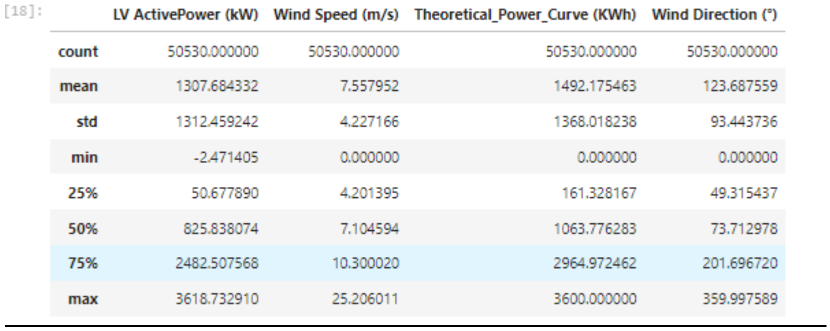
Data Collection and Preprocessing Phase

Date	12 July 2024
Team ID	SWTID1720108739
Project Title	**Predicting The Energy Output Of Wind Turbine Based On Weather Condition**
Maximum Marks	6 Marks

Data Exploration and Preprocessing Template

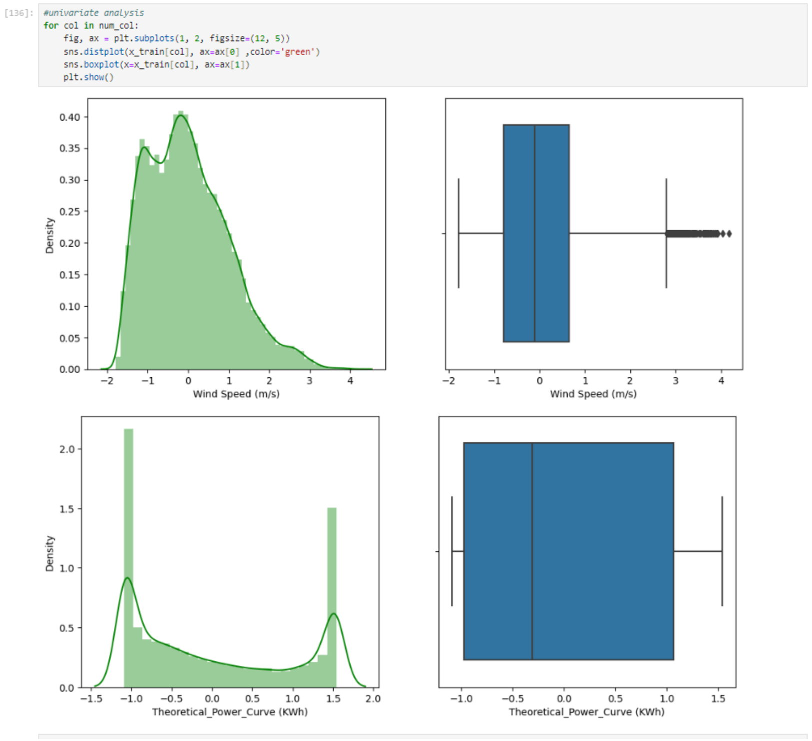
Identifies data sources, assesses quality issues like missing values and duplicates, and implements resolution plans to ensure accurate and reliable analysis.

Section	**Description**
Data Overview	Dimension: 50530 rows × 5columns Descriptive statistics: 

[18]:

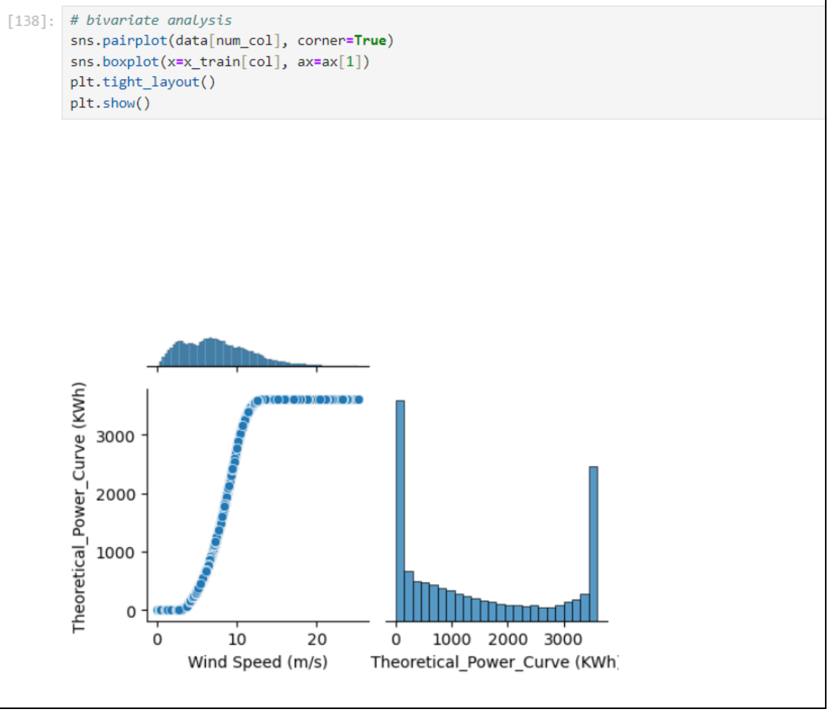
	LV ActivePower (kW)	Wind Speed (m/s)	Theoretical_Power_Curve (KWh)	Wind Direction (°)
count	50530.000000	50530.000000	50530.000000	50530.000000
mean	1307.684332	7.557952	1492.175463	123.687559
std	1312.459242	4.227166	1368.018238	93.443736
min	-2.471405	0.000000	0.000000	0.000000
25%	50.677890	4.201395	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

Univariate Analysis



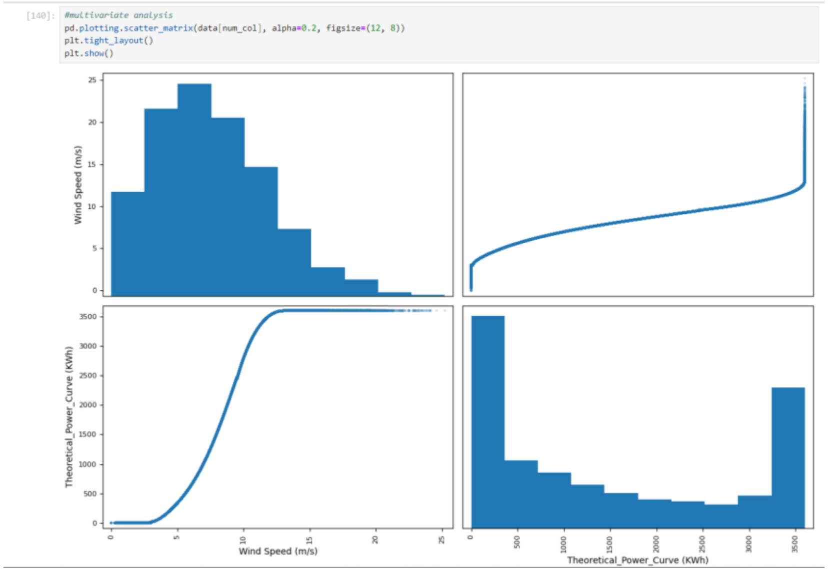
```
[136]: #univariate analysis
       for col in num_col:
           fig, ax = plt.subplots(1, 2, figsize=(12, 5))
           sns.distplot(x_train[col], ax=ax[0] ,color='green')
           sns.boxplot(x=x_train[col], ax=ax[1])
           plt.show()
```

Bivariate Analysis



```
[138]: # bivariate analysis
       sns.pairplot(data[num_col], corner=True)
       sns.boxplot(x=x_train[col], ax=ax[1])
       plt.tight_layout()
       plt.show()
```

Multivariate Analysis



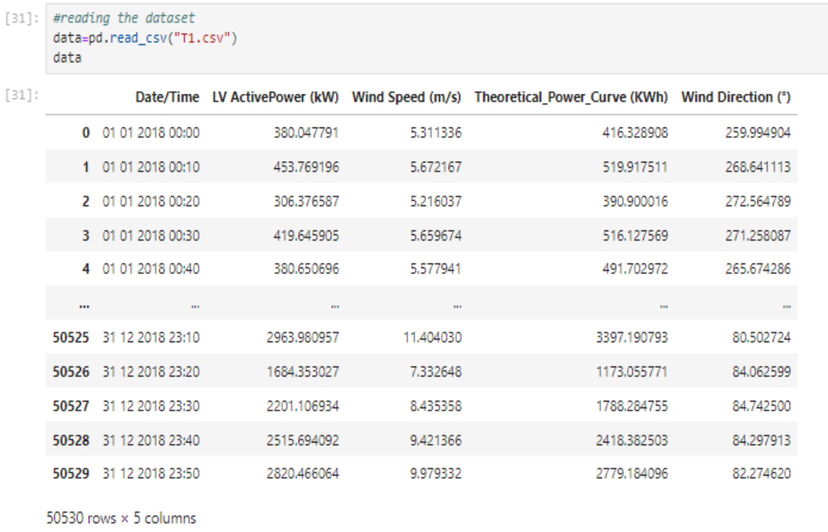
```
[140]: #multivariate analysis
       pd.plotting.scatter_matrix(data[num_col], alpha=0.2, figsize=(12, 8))
       plt.tight_layout()
       plt.show()
```

Outliers and Anomalies

-

Data Preprocessing Code Screenshots

Loading Data



```
[31]: #reading the dataset
      data=pd.read_csv("T1.csv")
      data
```

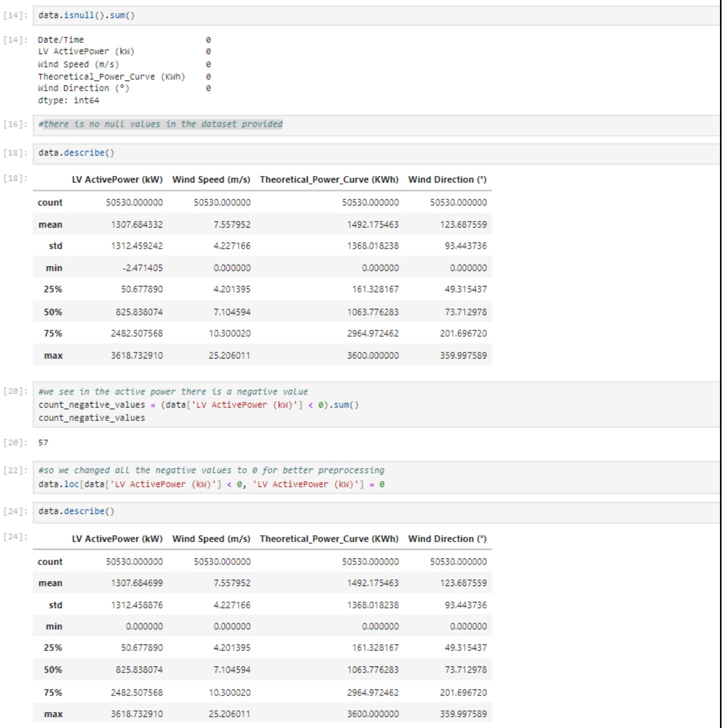
[31]:

	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
...	...	...	...	...	...
50525	31 12 2018 23:10	2963.980957	11.404030	3397.190793	80.502724
50526	31 12 2018 23:20	1684.353027	7.332648	1173.055771	84.062599
50527	31 12 2018 23:30	2201.106934	8.435358	1788.284755	84.742500
50528	31 12 2018 23:40	2515.694092	9.421366	2418.382503	84.297913
50529	31 12 2018 23:50	2820.466064	9.979332	2779.184096	82.274620

50530 rows × 5 columns

Handling Missing Data

- There is no null values in the dataset provided



```
[14]: data.isnull().sum()

[14]: Date/Time                       0
      LV ActivePower (kW)             0
      Wind Speed (m/s)                0
      Theoretical_Power_Curve (KWh)   0
      Wind Direction (°)              0
      dtype: int64

[16]: #there is no null values in the dataset provided

[18]: data.describe()
```

[18]:

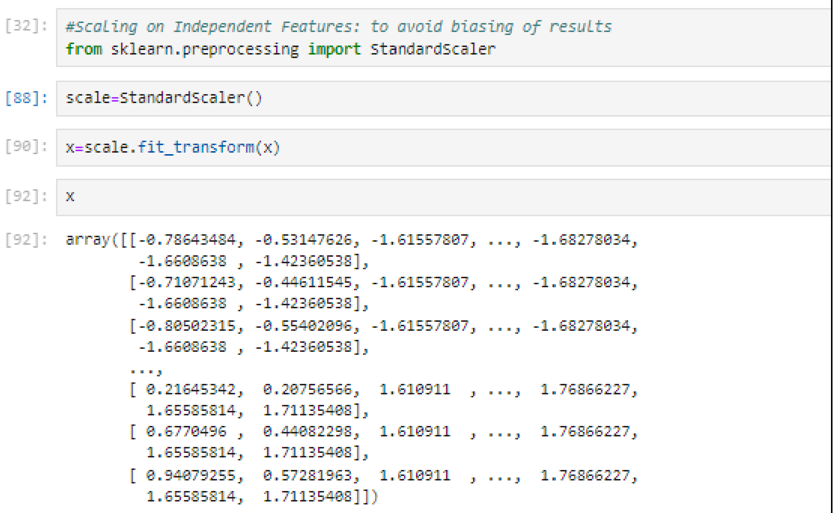
	LV ActivePower (kW)	Wind Speed (m/s)	Theoretical_Power_Curve (KWh)	Wind Direction (°)
count	50530.000000	50530.000000	50530.000000	50530.000000
mean	1307.684332	7.557952	1492.175463	123.687559
std	1312.459242	4.227166	1368.018238	93.443736
min	-2.471405	0.000000	0.000000	0.000000
25%	50.677890	4.201395	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

```
[20]: #we see in the active power there is a negative value
      count_negative_values = (data['LV ActivePower (kW)'] < 0).sum()
      count_negative_values

[20]: 57

[22]: #so we changed all the negative values to 0 for better preprocessing
      data.loc[data['LV ActivePower (kW)'] < 0, 'LV ActivePower (kW)'] = 0

[24]: data.describe()
```

[24]:

	LV ActivePower (kW)	Wind Speed (m/s)	Theoretical_Power_Curve (KWh)	Wind Direction (°)
count	50530.000000	50530.000000	50530.000000	50530.000000
mean	1307.684699	7.557952	1492.175463	123.687559
std	1312.458876	4.227166	1368.018238	93.443736
min	0.000000	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 Transformation



```
[32]: #Scaling on Independent Features: to avoid biasing of results
      from sklearn.preprocessing import StandardScaler

[88]: scale=StandardScaler()

[90]: x=scale.fit_transform(x)

[92]: x

[92]: array([[-0.78643484, -0.53147626, -1.61557807, ..., -1.68278034,
        -1.6608638 , -1.42360538],
       [-0.71071243, -0.44611545, -1.61557807, ..., -1.68278034,
        -1.6608638 , -1.42360538],
       [-0.80502315, -0.55402096, -1.61557807, ..., -1.68278034,
        -1.6608638 , -1.42360538],
       ...,
       [ 0.21645342,  0.20756566,  1.610911  , ...,  1.76866227,
         1.65585814,  1.71135408],
       [ 0.6770496 ,  0.44082298,  1.610911  , ...,  1.76866227,
         1.65585814,  1.71135408],
       [ 0.94079255,  0.57281963,  1.610911  , ...,  1.76866227,
         1.65585814,  1.71135408]])
```

Feature Engineering

- Attached the codes in final submission.

Save Processed Data

-