IMPORT REQUIRED PACKAGES(LIBRARIES)

In [1]:
```
# !pip install pandas
import pandas as pd
# !pip install numpy
import numpy as np
# !pip install seaborn
import seaborn as sns
# !pip install matplotlib
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
# !pip install ipywidgets
import ipywidgets as widgets
from matplotlib.cm import get_cmap
```

IMPORT DATASET

Import dataset from google drive and assigned the dataset as "df" ---- It will take few minutes based on internet because its file size is large.

In [2]:
```
URL = 'https://drive.google.com/file/d/1haTtjbR90YnbAGls8QVdREmHNmx1a5mQ/view?usp=sharing'
path = 'https://drive.google.com/uc?export=download&id='+URL.split('/')[-2]
```

In [3]:
```
%%time
print('*' * 127)
print('WIND TURBINE GENERATOR PARAMETER DATASET')
Master = pd.read_csv(path,low_memory = False, encoding = 'cp1252', parse_dates=['Date'], skipinitialspace=True)
print('*' * 127)
```

```
***************************************************************************************************************************
********
WIND TURBINE GENERATOR PARAMETER DATASET
***************************************************************************************************************************
********
Wall time: 15.5 s
```

In [4]:
```
df = Master
```

EXPLORATORY DATA ANALYSIS

CONFIRM THE DATA TRANSFER FROM GOOGLE DRIVE TO NOTEBOOK

In [5]:
```
print('First Five Records of the Dataset')
df.head()  # As default shows the top 5 Rows.
```

```
First Five Records of the Dataset
```

Out[5]:

	Date	Avg_Active_Power	Avg_Ambient_Temp	Avg_Generator_Speed	Avg_Nacelle_Pos	Avg_Pitch_Angle	Avg_Rotor_Speed	Avg_Wind_Speed	Bea
0	2016-08-01	1487.3	25.1	1654.9	138.0	0.03	15.5	9.3	
1	2016-08-01	1647.6	25.1	1677.8	138.0	-0.31	15.7	9.3	
2	2016-08-01	1506.4	25.2	1661.4	138.0	-0.48	15.6	9.0	
3	2016-08-01	1240.9	25.1	1601.4	138.0	-0.56	15.0	8.3	
4	2016-08-01	1202.4	25.1	1607.1	138.0	-0.73	15.1	8.5	

5 rows × 21 columns

In [6]:
```
print('Last Five Records of the Dataset')
df.tail()  # As default shows the Last 5 Rows.
```

```
Last Five Records of the Dataset
```

Out[6]:

	Date	Avg_Active_Power	Avg_Ambient_Temp	Avg_Generator_Speed	Avg_Nacelle_Pos	Avg_Pitch_Angle	Avg_Rotor_Speed	Avg_Wind_Speed
245194	2021-04-30	205.1	32.2	1051.2	1.0	1.24	9.9	4.7
245195	2021-04-30	208.4	32.1	1063.5	1.0	1.33	10.0	4.7
245196	2021-04-30	228.0	31.8	1074.9	355.7	1.32	10.1	5.2
245197	2021-04-30	355.1	31.7	1106.6	354.0	0.17	10.4	5.7
245198	2021-04-30	374.3	31.6	1119.2	0.0	0.12	10.5	5.9

5 rows × 21 columns

Check for the Shape

In [7]:
```
print("The Data Frame having the Rows of '{}' and Columns of '{}'".format (df.shape[0],df.shape[1]))
```

```
The Data Frame having the Rows of '245199' and Columns of '21'
```

Check for the Detailed Information of the Dataset

In [8]:
```
print('Total_Columns: ', len(df.columns),'\n')
print(df.columns,'\n')
print('Shape :',df.shape)
```

```
Total_Columns:  21

Index(['Date', 'Avg_Active_Power', 'Avg_Ambient_Temp', 'Avg_Generator_Speed',
       'Avg_Nacelle_Pos', 'Avg_Pitch_Angle', 'Avg_Rotor_Speed',
       'Avg_Wind_Speed', 'Bearing_DE_Temp', 'Bearing_NDE_Temp',
       'Gearbox_bearing_Temp', 'Gearbox_oil_Temp', 'Generator_wind_Temp_1',
       'Generator_wind_Temp_2', 'Generator_wind_Temp_3',
       'Generator’s_sliprings_Temp', 'Hidraulic_group_pressure',
       'Nacelle_Misalignment_Avg_Wind_Dir', 'Trafo_1_wind_Temp',
       'Trafo_2_wind_Temp', 'Trafo_3_wind_Temp'],
      dtype='object')

Shape : (245199, 21)
```

In [9]:
```
df.rename({"Generator�s_sliprings_Temp":"Generators_sliprings_Temp", "Generator’s_sliprings_Temp":"Generators_sliprings_
```

In [10]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 245199 entries, 0 to 245198
Data columns (total 21 columns):
 #   Column                            Non-Null Count   Dtype
---  ------                            --------------   -----
 0   Date                              245199 non-null  datetime64[ns]
 1   Avg_Active_Power                  236405 non-null  float64
 2   Avg_Ambient_Temp                  236405 non-null  float64
 3   Avg_Generator_Speed               236405 non-null  float64
 4   Avg_Nacelle_Pos                   236395 non-null  float64
 5   Avg_Pitch_Angle                   236405 non-null  float64
 6   Avg_Rotor_Speed                   236405 non-null  float64
 7   Avg_Wind_Speed                    236405 non-null  float64
 8   Bearing_DE_Temp                   236405 non-null  float64
 9   Bearing_NDE_Temp                  236404 non-null  float64
 10  Gearbox_bearing_Temp              236405 non-null  float64
 11  Gearbox_oil_Temp                  236405 non-null  float64
 12  Generator_wind_Temp_1             236402 non-null  float64
 13  Generator_wind_Temp_2             236402 non-null  float64
 14  Generator_wind_Temp_3             236402 non-null  float64
 15  Generators_sliprings_Temp         236402 non-null  float64
 16  Hidraulic_group_pressure          236405 non-null  float64
 17  Nacelle_Misalignment_Avg_Wind_Dir 236405 non-null  float64
 18  Trafo_1_wind_Temp                 236402 non-null  float64
 19  Trafo_2_wind_Temp                 236401 non-null  float64
 20  Trafo_3_wind_Temp                 236401 non-null  float64
dtypes: datetime64[ns](1), float64(20)
memory usage: 39.3 MB
```

CHANGE THE DATA TYPE

In [11]:
```
df['Date'] = pd.to_datetime(df['Date'], errors='coerce')
```

Displays memory consumed by each column

In [12]:
```
print(df.memory_usage(),'\n')
print('Dataset uses {0} MB'.format(df.memory_usage().sum()/1024**2))
```

```
Index                                128
Date                             1961592
Avg_Active_Power                 1961592
Avg_Ambient_Temp                 1961592
Avg_Generator_Speed              1961592
Avg_Nacelle_Pos                  1961592
Avg_Pitch_Angle                  1961592
Avg_Rotor_Speed                  1961592
Avg_Wind_Speed                   1961592
Bearing_DE_Temp                  1961592
Bearing_NDE_Temp                 1961592
Gearbox_bearing_Temp             1961592
Gearbox_oil_Temp                 1961592
Generator_wind_Temp_1            1961592
Generator_wind_Temp_2            1961592
Generator_wind_Temp_3            1961592
Generators_sliprings_Temp        1961592
Hidraulic_group_pressure         1961592
Nacelle_Misalignment_Avg_Wind_Dir 1961592
Trafo_1_wind_Temp                1961592
Trafo_2_wind_Temp                1961592
Trafo_3_wind_Temp                1961592
dtype: int64

Dataset uses 39.285240173339844 MB
```

In [13]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 245199 entries, 0 to 245198
Data columns (total 21 columns):
 #   Column                          Non-Null Count   Dtype         
---  ------                          --------------   -----         
 0   Date                            245199 non-null  datetime64[ns]
 1   Avg_Active_Power                236405 non-null  float64       
 2   Avg_Ambient_Temp                236405 non-null  float64       
 3   Avg_Generator_Speed             236405 non-null  float64       
 4   Avg_Nacelle_Pos                 236395 non-null  float64       
 5   Avg_Pitch_Angle                 236405 non-null  float64       
 6   Avg_Rotor_Speed                 236405 non-null  float64       
 7   Avg_Wind_Speed                  236405 non-null  float64       
 8   Bearing_DE_Temp                 236405 non-null  float64       
 9   Bearing_NDE_Temp                236404 non-null  float64       
 10  Gearbox_bearing_Temp            236405 non-null  float64       
 11  Gearbox_oil_Temp                236405 non-null  float64       
 12  Generator_wind_Temp_1           236402 non-null  float64       
 13  Generator_wind_Temp_2           236402 non-null  float64       
 14  Generator_wind_Temp_3           236402 non-null  float64       
 15  Generators_sliprings_Temp       236402 non-null  float64       
 16  Hidraulic_group_pressure        236405 non-null  float64       
 17  Nacelle_Misalignment_Avg_Wind_Dir 236405 non-null  float64       
 18  Trafo_1_wind_Temp               236402 non-null  float64       
 19  Trafo_2_wind_Temp               236401 non-null  float64       
 20  Trafo_3_wind_Temp               236401 non-null  float64       
dtypes: datetime64[ns](1), float64(20)
memory usage: 39.3 MB
```

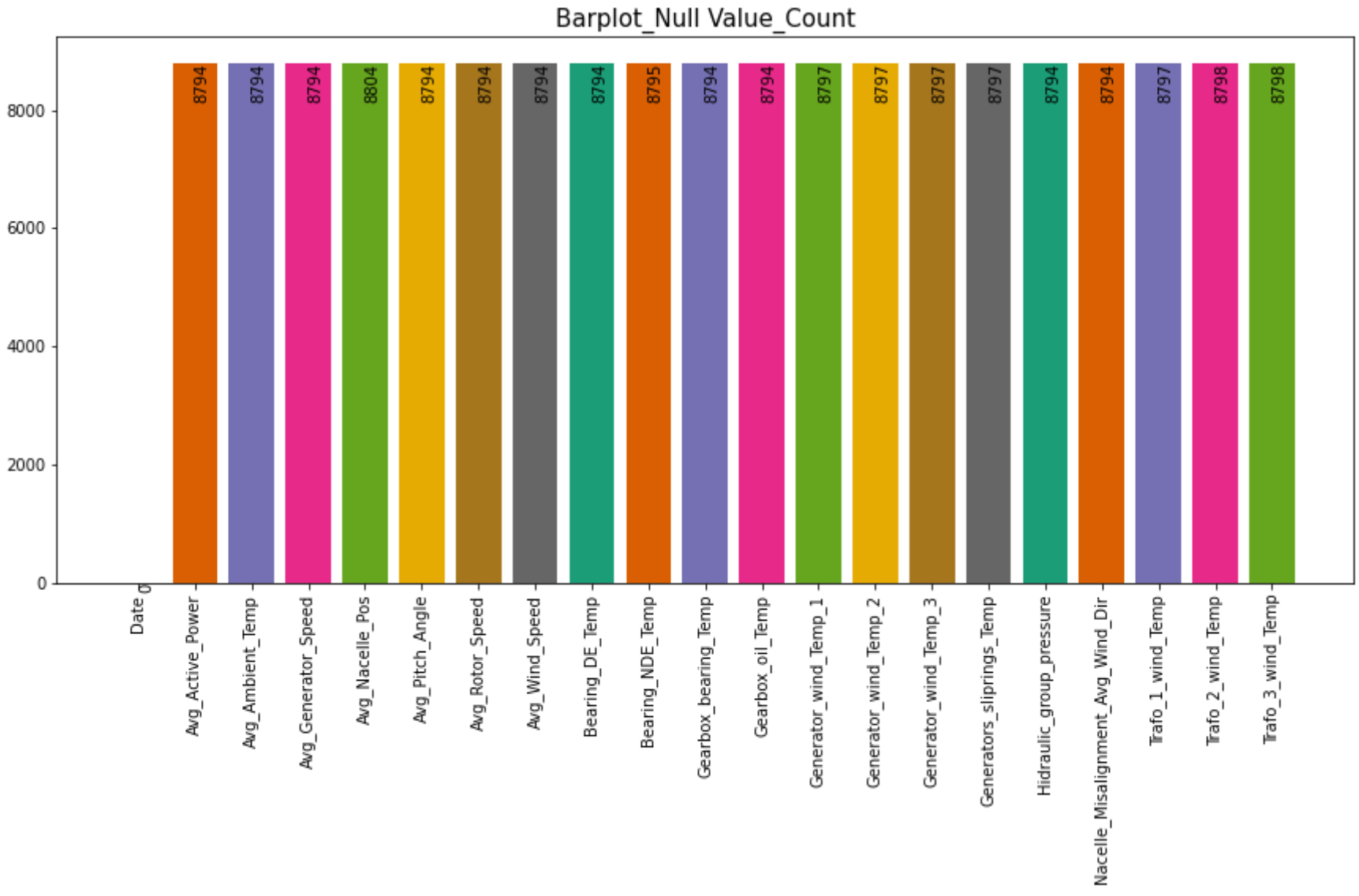
CHECK FOR NULL VALUES

In [14]:
```
df.isnull().sum()
```

Out[14]:
```
Date                                0
Avg_Active_Power                 8794
Avg_Ambient_Temp                 8794
Avg_Generator_Speed              8794
Avg_Nacelle_Pos                  8804
Avg_Pitch_Angle                  8794
Avg_Rotor_Speed                  8794
Avg_Wind_Speed                   8794
Bearing_DE_Temp                  8794
Bearing_NDE_Temp                 8795
Gearbox_bearing_Temp             8794
Gearbox_oil_Temp                 8794
Generator_wind_Temp_1            8797
Generator_wind_Temp_2            8797
Generator_wind_Temp_3            8797
Generators_sliprings_Temp        8797
Hidraulic_group_pressure         8794
Nacelle_Misalignment_Avg_Wind_Dir    8794
Trafo_1_wind_Temp                8797
Trafo_2_wind_Temp                8798
Trafo_3_wind_Temp                8798
dtype: int64
```

NULL VALUE COUNTS PLOT BEFORE TREATMENT

In [15]:
```
name = "Dark2"
cmap = get_cmap(name)
colors = cmap.colors
x = df.columns
y = df.isnull().sum()
plt.bar(x,y,color = colors, align = 'center')
plt.xticks(rotation=90)
plt.title('Barplot_Null Value_Count', fontsize = 15)
plt.subplots_adjust(left=0.4, bottom=0.1, right=2.3, top=1.3)
for i in range(len(x)):
    pos = y[i]
    string = '{:}'.format(pos)
    plt.text(i,pos,string,ha='left',color='black',rotation = 'vertical', va = 'top')
```



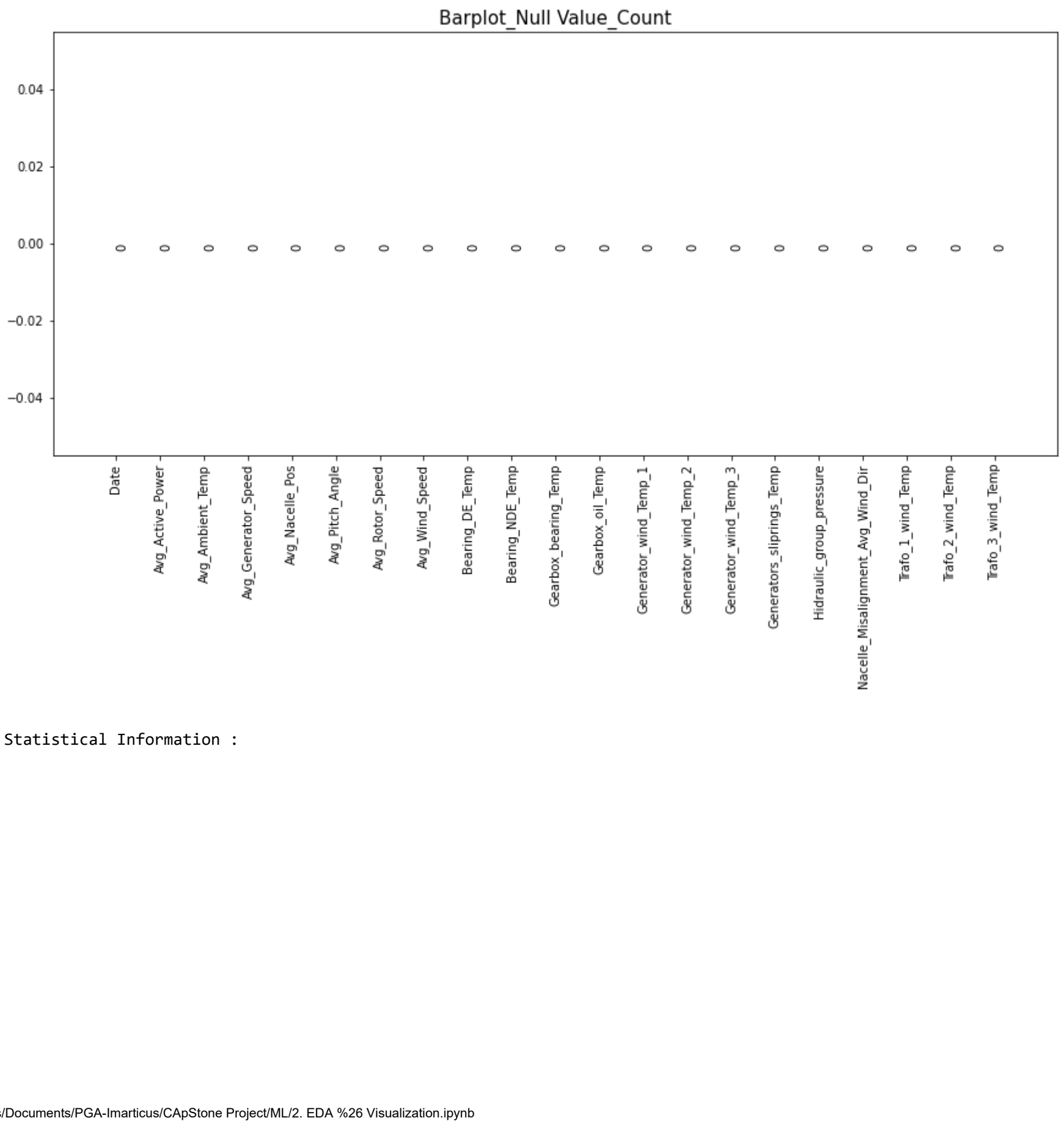
NULL VALUE TREATMENT

Every parameter of the Wind Turbine is a Numerical Variable. So, to avoid any Outliers effect on variables. I am doing Null values treatment with its Median.

In [16]:
```
constant_values = {'Avg_Active_Power': df.iloc[:,1].median(), 'Avg_Ambient_Temp': df.iloc[:,2].median(),
       'Avg_Generator_Speed': df.iloc[:,3].median(), 'Avg_Nacelle_Pos': df.iloc[:,4].median(), 'Avg_Pitch_Angle': df.iloc
       'Avg_Rotor_Speed': df.iloc[:,6].median(), 'Avg_Wind_Speed': df.iloc[:,7].median(), 'Bearing_DE_Temp': df.iloc[:,8
       'Bearing_NDE_Temp': df.iloc[:,9].median(), 'Gearbox_bearing_Temp': df.iloc[:,10].median(), 'Gearbox_oil_Temp': df
       'Generator_wind_Temp_1': df.iloc[:,12].median(), 'Generator_wind_Temp_2': df.iloc[:,13].median(),
       'Generator_wind_Temp_3': df.iloc[:,14].median(), "Generators_sliprings_Temp": df.iloc[:,15].median(),
       'Hidraulic_group_pressure': df.iloc[:,16].median(), 'Nacelle_Misalignment_Avg_Wind_Dir': df.iloc[:,17].median(),
       'Trafo_1_wind_Temp': df.iloc[:,18].median(), 'Trafo_2_wind_Temp': df.iloc[:,19].median(), 'Trafo_3_wind_Temp': df
df[['Avg_Active_Power', 'Avg_Ambient_Temp', 'Avg_Generator_Speed', 'Avg_Nacelle_Pos', 'Avg_Pitch_Angle', 'Avg_Rotor_Speed
    'Bearing_NDE_Temp', 'Gearbox_bearing_Temp', 'Gearbox_oil_Temp', 'Generator_wind_Temp_1', 'Generator_wind_Temp_2', 'G
    'Hidraulic_group_pressure', 'Nacelle_Misalignment_Avg_Wind_Dir',
    'Trafo_1_wind_Temp', 'Trafo_2_wind_Temp', 'Trafo_3_wind_Temp']] = df[['Avg_Active_Power', 'Avg_Ambient_Temp', 'Avg_G
                                                                          'Avg_Rotor_Speed', 'Avg_Wind_Speed', 'Bearing_
                                                                          'Gearbox_oil_Temp', 'Generator_wind_Temp_1', '
                                                                          "Generators_sliprings_Temp", 'Hidraulic_group_
                                                                          'Trafo_1_wind_Temp', 'Trafo_2_wind_Temp', 'Tra
```

NULL VALUE COUNTS PLOT AFTER TREATMENT

In [17]:
```
name = "Dark2"
cmap = get_cmap(name)
colors = cmap.colors
x = df.columns
y = df.isnull().sum()
plt.bar(x,y,color = colors, align = 'center')
plt.xticks(rotation=90)
plt.title('Barplot_Null Value_Count', fontsize = 15)
plt.subplots_adjust(left=0.4, bottom=0.1, right=2.3, top=1.3)
for i in range(len(x)):
    pos = y[i]
    string = '{:}'.format(pos)
    plt.text(i,pos,string,ha='left',color='black',rotation = 'vertical', va = 'top')
```



Statistical Information :

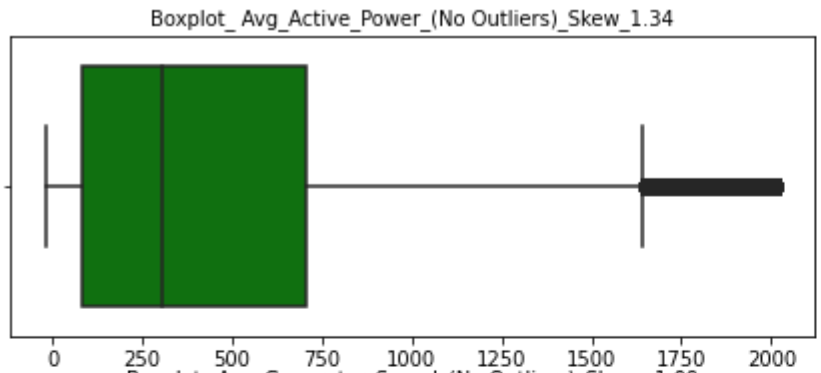
In [18]:
```
df.describe()
```

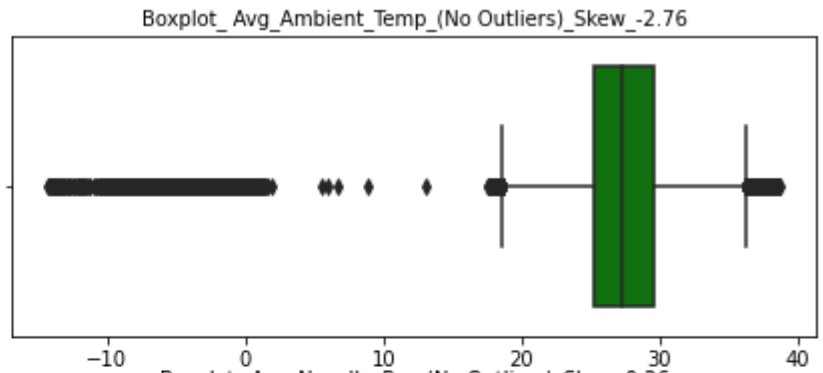
Out[18]:

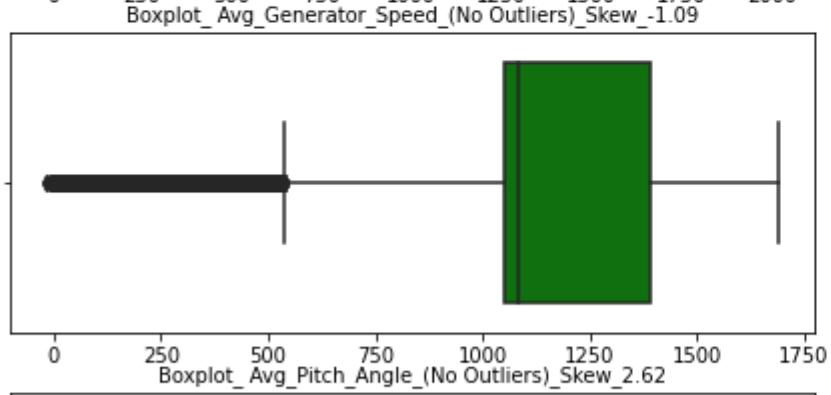
	Avg_Active_Power	Avg_Ambient_Temp	Avg_Generator_Speed	Avg_Nacelle_Pos	Avg_Pitch_Angle	Avg_Rotor_Speed	Avg_Wind_Speed	Bearir
count	245199.000000	245199.000000	245199.00000	245199.000000	245199.000000	245199.000000	245199.000000	24
mean	499.378702	27.357988	1090.59523	160.795747	9.678610	10.239833	5.853972	
std	551.593183	4.459187	439.39702	97.220176	23.826089	4.121497	2.518097	
min	-17.300000	-14.300000	-18.10000	0.000000	-1.010000	0.000000	0.100000	
25%	84.600000	25.200000	1049.90000	83.000000	0.000000	9.800000	4.100000	
50%	305.900000	27.200000	1081.70000	165.400000	0.710000	10.200000	5.600000	
75%	707.950000	29.600000	1392.60000	203.000000	2.840000	13.100000	7.300000	
max	2019.900000	38.700000	1689.70000	360.000000	87.100000	15.900000	24.700000	

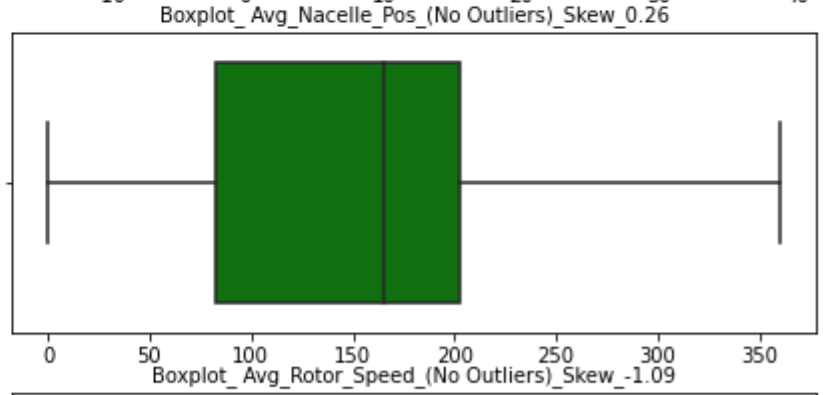
CHECK FOR OUTLIER VALUES USING BOX PLOTS

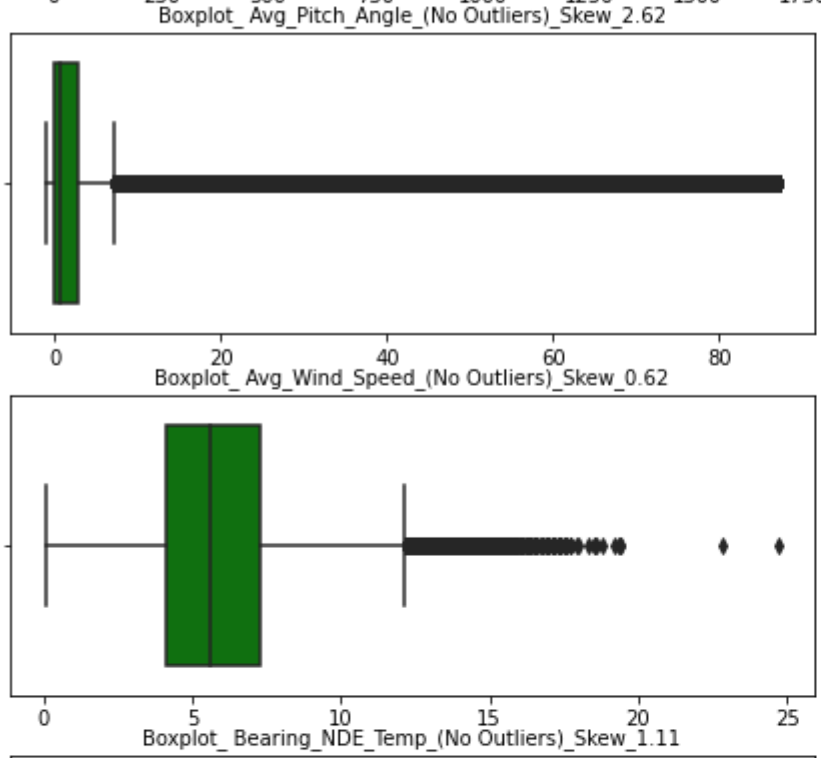
In [19]:
```
plt.subplot(8,2,1)
sns.boxplot(df.iloc[:,1], color = 'green')
plt.xlabel("")
plt.title('Boxplot_ '+ df.iloc[:,1].name +'_(No Outliers)_Skew_' + str(round(df.iloc[:,1].skew(),2)), fontsize = 10)
plt.subplot(8,2,2)
sns.boxplot(df.iloc[:,2], color = 'green')
plt.xlabel("")
plt.title('Boxplot_ '+ df.iloc[:,2].name +'_(No Outliers)_Skew_' + str(round(df.iloc[:,2].skew(),2)), fontsize = 10)
plt.subplot(8,2,3)
sns.boxplot(df.iloc[:,3], color = 'green')
plt.xlabel("")
plt.title('Boxplot_ '+ df.iloc[:,3].name +'_(No Outliers)_Skew_' + str(round(df.iloc[:,3].skew(),2)), fontsize = 10)
plt.subplot(8,2,4)
sns.boxplot(df.iloc[:,4], color = 'green')
plt.xlabel("")
plt.title('Boxplot_ '+ df.iloc[:,4].name +'_(No Outliers)_Skew_' + str(round(df.iloc[:,4].skew(),2)), fontsize = 10)
plt.subplot(8,2,5)
sns.boxplot(df.iloc[:,5], color = 'green')
plt.xlabel("")
plt.title('Boxplot_ '+ df.iloc[:,5].name +'_(No Outliers)_Skew_' + str(round(df.iloc[:,5].skew(),2)), fontsize = 10)
plt.subplot(8,2,6)
sns.boxplot(df.iloc[:,6], color = 'green')
plt.xlabel("")
plt.title('Boxplot_ '+ df.iloc[:,6].name +'_(No Outliers)_Skew_' + str(round(df.iloc[:,6].skew(),2)), fontsize = 10)
plt.subplot(8,2,7)
sns.boxplot(df.iloc[:,7], color = 'green')
plt.xlabel("")
plt.title('Boxplot_ '+ df.iloc[:,7].name +'_(No Outliers)_Skew_' + str(round(df.iloc[:,7].skew(),2)), fontsize = 10)
plt.subplot(8,2,8)
sns.boxplot(df.iloc[:,8], color = 'green')
plt.xlabel("")
plt.title('Boxplot_ '+ df.iloc[:,8].name +'_(No Outliers)_Skew_' + str(round(df.iloc[:,8].skew(),2)), fontsize = 10)
plt.subplot(8,2,9)
sns.boxplot(df.iloc[:,9], color = 'green')
plt.xlabel("")
plt.title('Boxplot_ '+ df.iloc[:,9].name +'_(No Outliers)_Skew_' + str(round(df.iloc[:,9].skew(),2)), fontsize = 10)
plt.subplot(8,2,10)
sns.boxplot(df.iloc[:,10], color = 'green')
plt.xlabel("")
plt.title('Boxplot_ '+ df.iloc[:,10].name +'_(No Outliers)_Skew_' + str(round(df.iloc[:,10].skew(),2)), fontsize = 10)

plt.subplots_adjust(left=0.45, bottom=0, right=2.5, top=4.9)
```

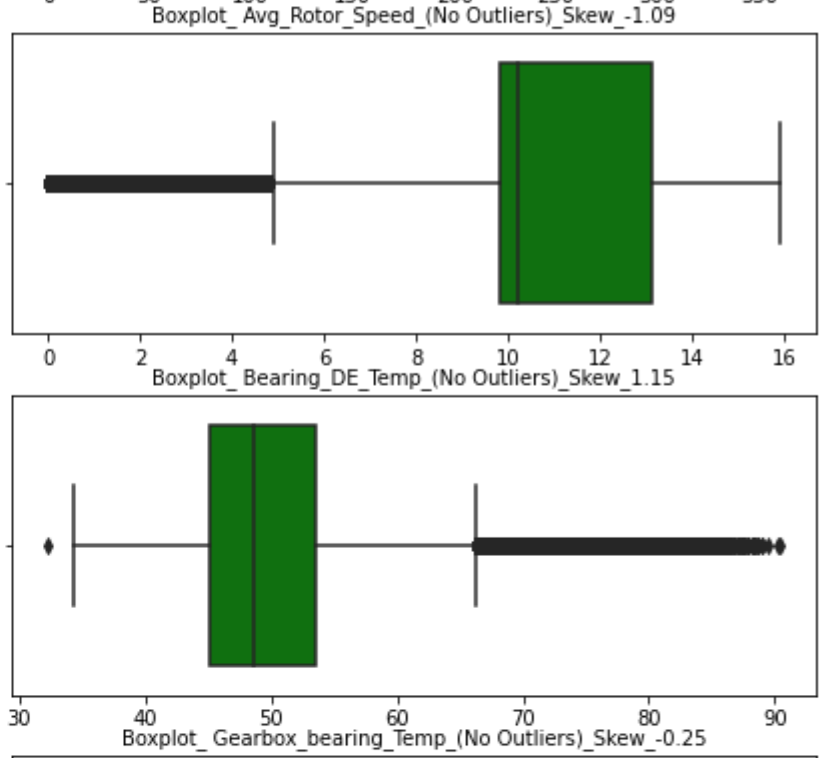

Boxplot_ Avg_Active_Power_(No Outliers)_Skew_1.34
0
250
500
750
1000
1250
1500
1750
2000

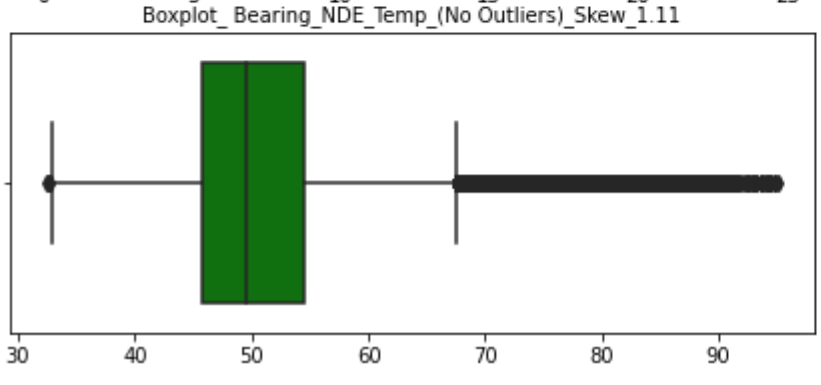

Boxplot_ Avg_Ambient_Temp_(No Outliers)_Skew_-2.76
−10
0
10
20
30
40

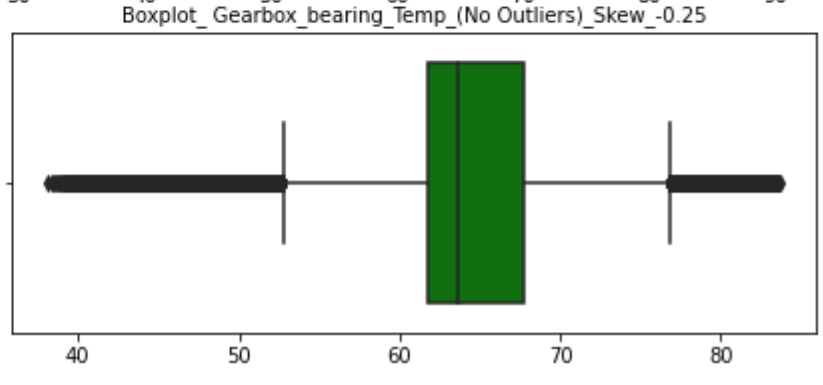

Boxplot_ Avg_Generator_Speed_(No Outliers)_Skew_-1.09
0
250
500
750
1000
1250
1500
1750


Boxplot_ Avg_Nacelle_Pos_(No Outliers)_Skew_0.26
0
50
100
150
200
250
300
350

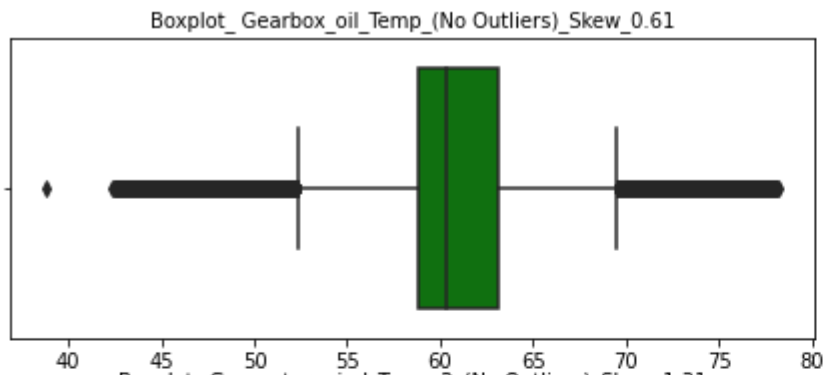

Boxplot_ Avg_Pitch_Angle_(No Outliers)_Skew_2.62
0
20
40
60
80
Boxplot_ Avg_Wind_Speed_(No Outliers)_Skew_0.62
0
5
10
15
20
25

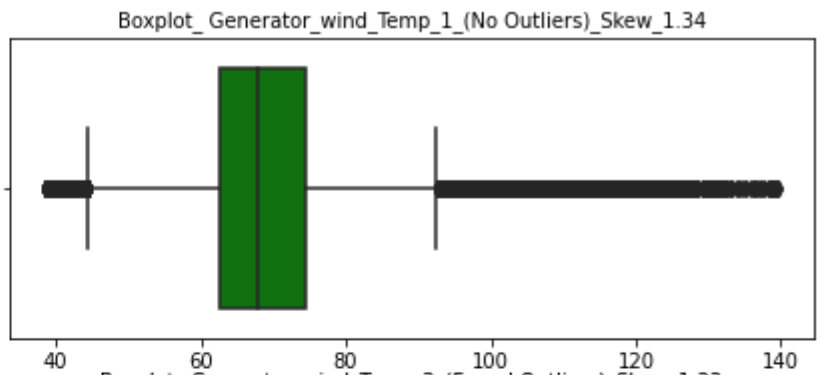

Boxplot_ Avg_Rotor_Speed_(No Outliers)_Skew_-1.09
0
2
4
6
8
10
12
14
16
Boxplot_ Bearing_DE_Temp_(No Outliers)_Skew_1.15
30
40
50
60
70
80
90

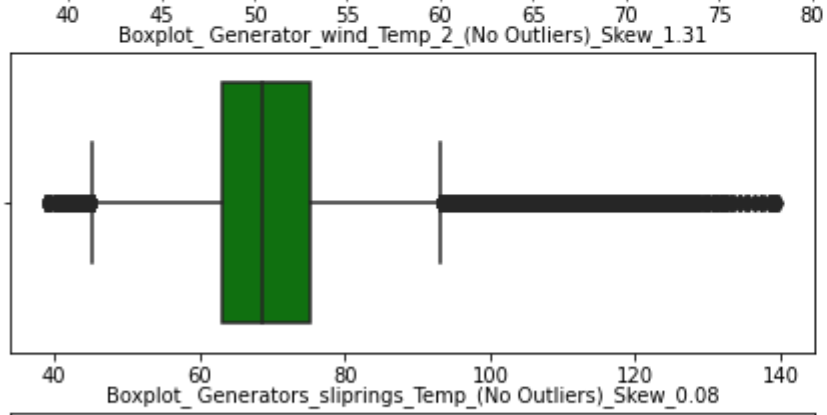

Boxplot_ Bearing_NDE_Temp_(No Outliers)_Skew_1.11
30
40
50
60
70
80
90

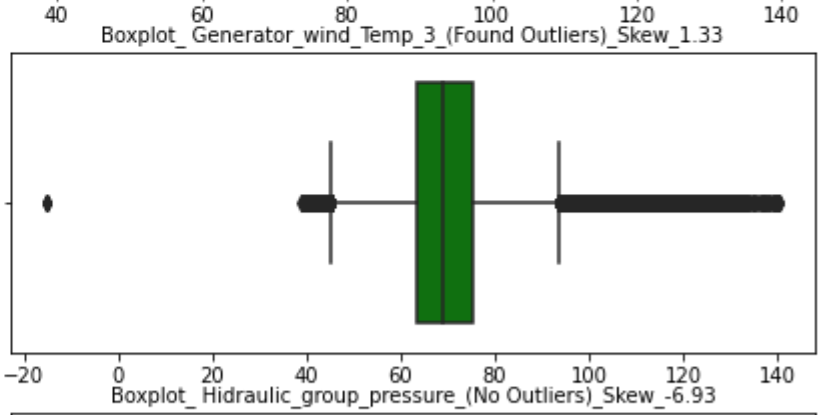

Boxplot_ Gearbox_bearing_Temp_(No Outliers)_Skew_-0.25
40
50
60
70
80

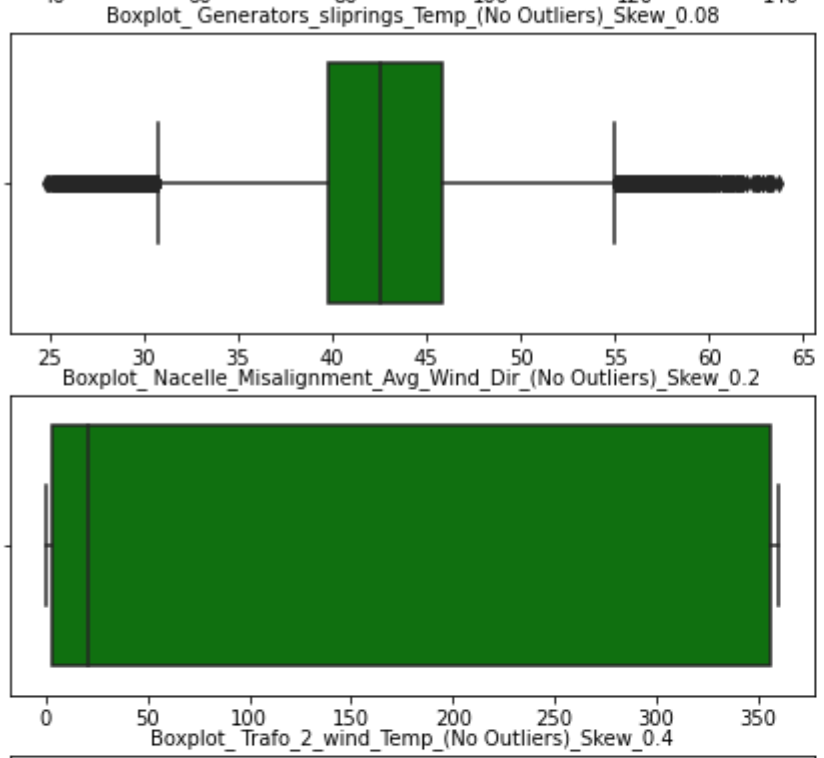
In [20]:
```
plt.subplot(8,2,1)
sns.boxplot(df.iloc[:,11], color = 'green')
plt.xlabel("")
plt.title('Boxplot_ '+ df.iloc[:,11].name +'_(No Outliers)_Skew_' + str(round(df.iloc[:,11].skew(),2)), fontsize = 10)
plt.subplot(8,2,2)
sns.boxplot(df.iloc[:,12], color = 'green')
plt.xlabel("")
plt.title('Boxplot_ '+ df.iloc[:,12].name +'_(No Outliers)_Skew_' + str(round(df.iloc[:,12].skew(),2)), fontsize = 10)
plt.subplot(8,2,3)
sns.boxplot(df.iloc[:,13], color = 'green')
plt.xlabel("")
plt.title('Boxplot_ '+ df.iloc[:,13].name +'_(No Outliers)_Skew_' + str(round(df.iloc[:,13].skew(),2)), fontsize = 10)
plt.subplot(8,2,4)
sns.boxplot(df.iloc[:,14], color = 'green')
plt.xlabel("")
plt.title('Boxplot_ '+ df.iloc[:,14].name +'_(Found Outliers)_Skew_' + str(round(df.iloc[:,14].skew(),2)), fontsize = 10
plt.subplot(8,2,5)
sns.boxplot(df.iloc[:,15], color = 'green')
plt.xlabel("")
plt.title('Boxplot_ '+ df.iloc[:,15].name +'_(No Outliers)_Skew_' + str(round(df.iloc[:,15].skew(),2)), fontsize = 10)
plt.subplot(8,2,6)
sns.boxplot(df.iloc[:,16], color = 'green')
plt.xlabel("")
plt.title('Boxplot_ '+ df.iloc[:,16].name +'_(No Outliers)_Skew_' + str(round(df.iloc[:,16].skew(),2)), fontsize = 10)
plt.subplot(8,2,7)
sns.boxplot(df.iloc[:,17], color = 'green')
plt.xlabel("")
plt.title('Boxplot_ '+ df.iloc[:,17].name +'_(No Outliers)_Skew_' + str(round(df.iloc[:,17].skew(),2)), fontsize = 10)
plt.subplot(8,2,8)
sns.boxplot(df.iloc[:,18], color = 'green')
plt.xlabel("")
plt.title('Boxplot_ '+ df.iloc[:,18].name +'_(No Outliers)_Skew_' + str(round(df.iloc[:,18].skew(),2)), fontsize = 10)
plt.subplot(8,2,9)
sns.boxplot(df.iloc[:,19], color = 'green')
plt.xlabel("")
plt.title('Boxplot_ '+ df.iloc[:,19].name +'_(No Outliers)_Skew_' + str(round(df.iloc[:,19].skew(),2)), fontsize = 10)
plt.subplot(8,2,10)
sns.boxplot(df.iloc[:,20], color = 'green')
plt.xlabel("")
plt.title('Boxplot_ '+ df.iloc[:,20].name +'_(No Outliers)_Skew_' + str(round(df.iloc[:,20].skew(),2)), fontsize = 10)

plt.subplots_adjust(left=0.45, bottom=0, right=2.5, top=4.9)
```

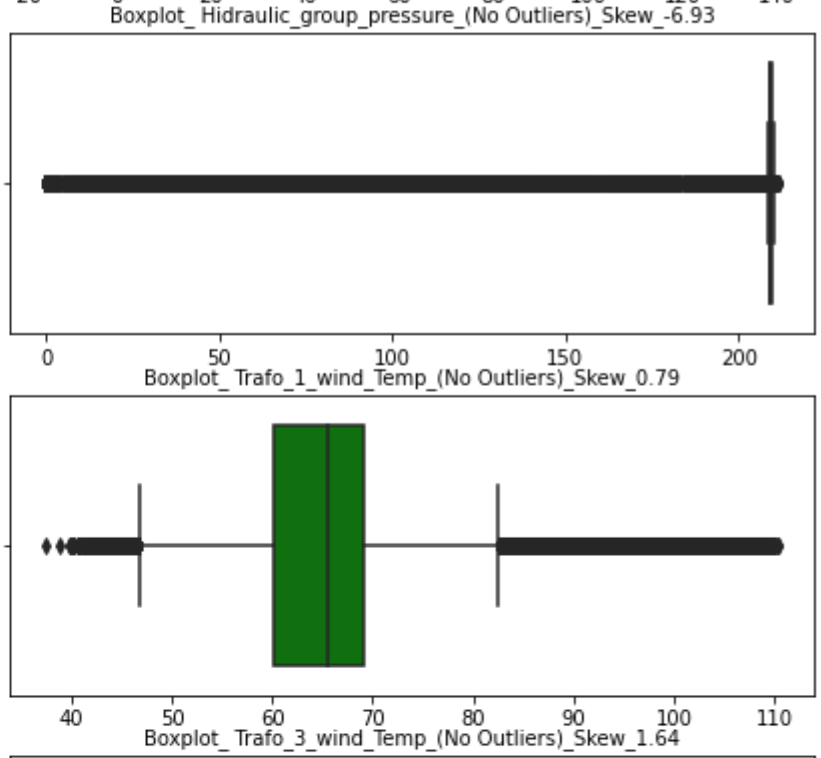












From above box plots i found that the data is almost not having the outliers except one variable that is Generat or Winding Temperature-3. So, we need Outliers Treatment as using Quantiles.

Check for the Quartile Ranges

In [21]:
```
print('Lower Limit - 5% :', df.iloc[:,14].quantile(0.05), '\n Upper Limit - 95% :', df.iloc[:,14].quantile(0.95))
```

```
Lower Limit - 5% : 51.8
 Upper Limit - 95% : 105.3
```

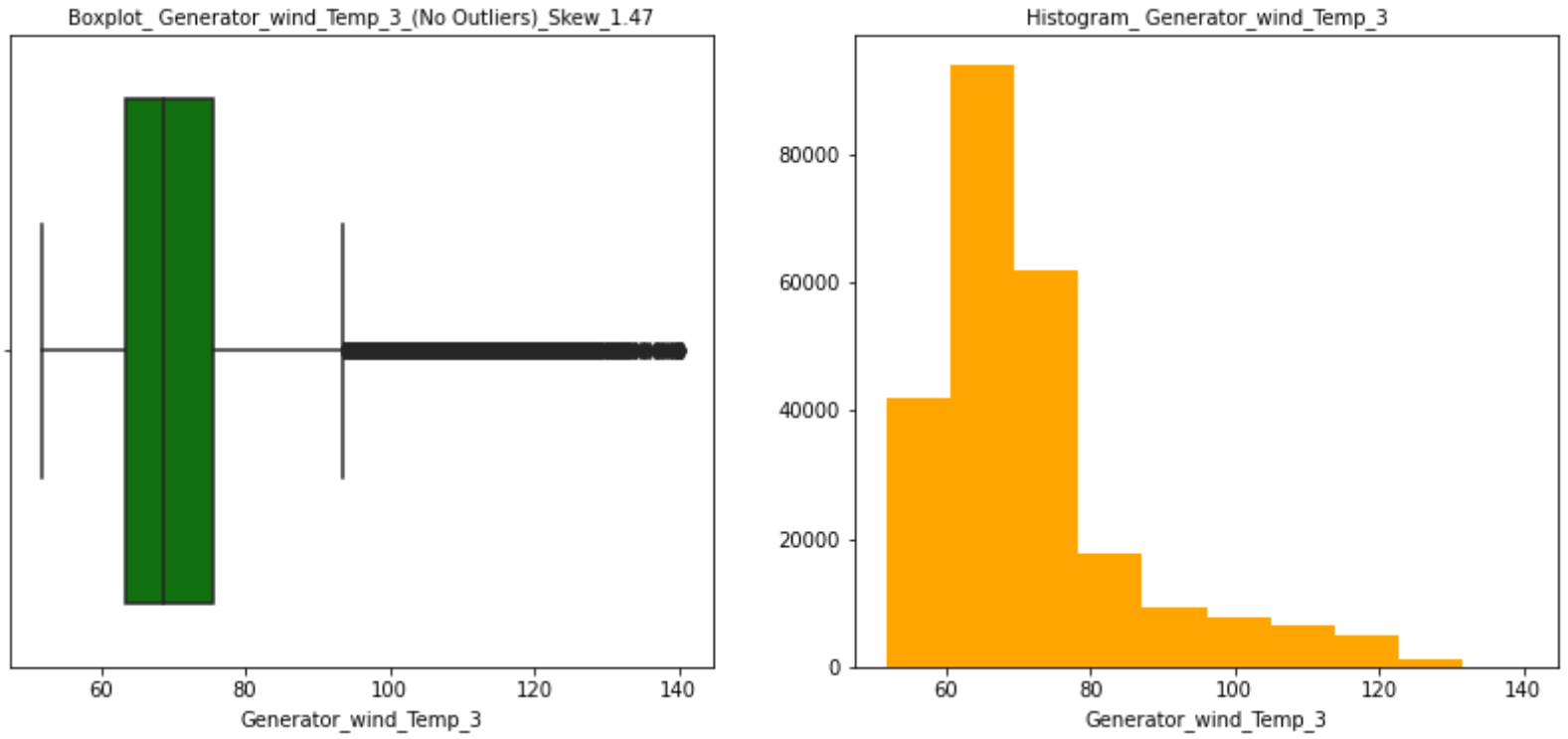
Replace the Outliers with its Quartile ranges

In [22]:
```
df.iloc[:,14] = np.where(df.iloc[:,14] < df.iloc[:,14].quantile(0.05), df.iloc[:,14].quantile(0.05), df.iloc[:,14])
df.iloc[:,14].describe()
```

Out[22]:
```
count    245199.000000
mean         71.662278
std          14.424620
min          51.800000
25%          63.400000
50%          68.800000
75%          75.500000
max         140.400000
Name: Generator_wind_Temp_3, dtype: float64
```

Box Plot and Histogram plot for re-checking the Outliers

In [23]:
```
plt.subplot(1,2,1)
sns.boxplot(df.iloc[:,14], color = 'green')
plt.xlabel(df.iloc[:,14].name, fontsize = 10)
plt.title('Boxplot_ '+ df.iloc[:,14].name +'_(No Outliers)_Skew_' + str(round(df.iloc[:,14].skew(),2)), fontsize = 10)
plt.subplot(1,2,2)
plt.hist(df.iloc[:,14], color = 'orange')
plt.xlabel(df.iloc[:,14].name, fontsize = 10)
plt.title('Histogram_ '+ df.iloc[:,14].name, fontsize = 10)
plt.subplots_adjust(left=0.4, bottom=0.1, right=2.2, top=1.2)
```



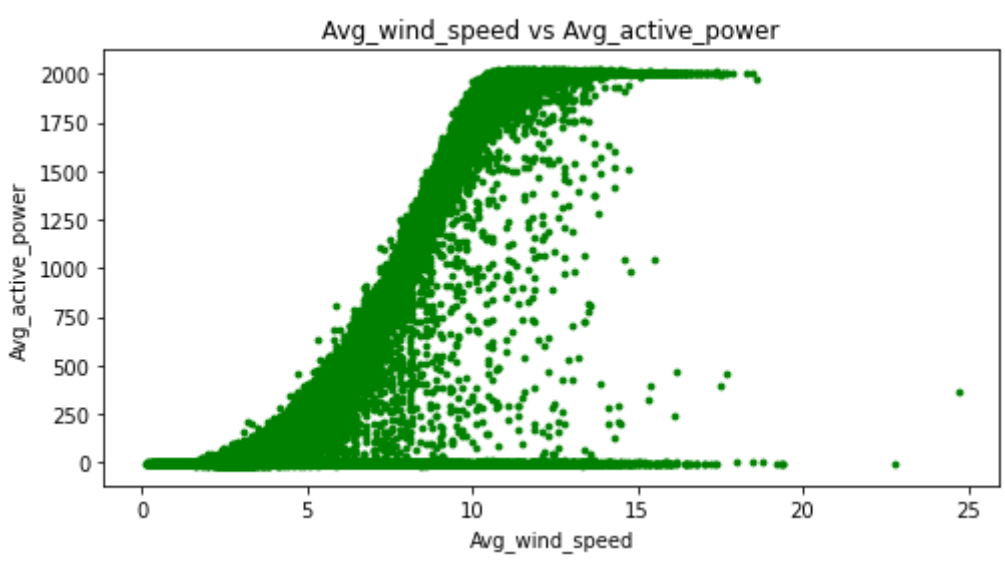
In [24]:
```
df.to_csv('Cleaned_df.csv')
```

VISUALIZATION

In [25]:
```
def Scatter_Plot(X,Y,Size,Color):
        fig = plt.figure(figsize=(8,4))

        plt.scatter(x = X,
                    y = Y, s = float(Size), color = Color)

        plt.xlabel(X.name.capitalize())
        plt.ylabel(Y.name.capitalize())

        plt.title("%s vs %s"%(X.name.capitalize(), Y.name.capitalize()))
```

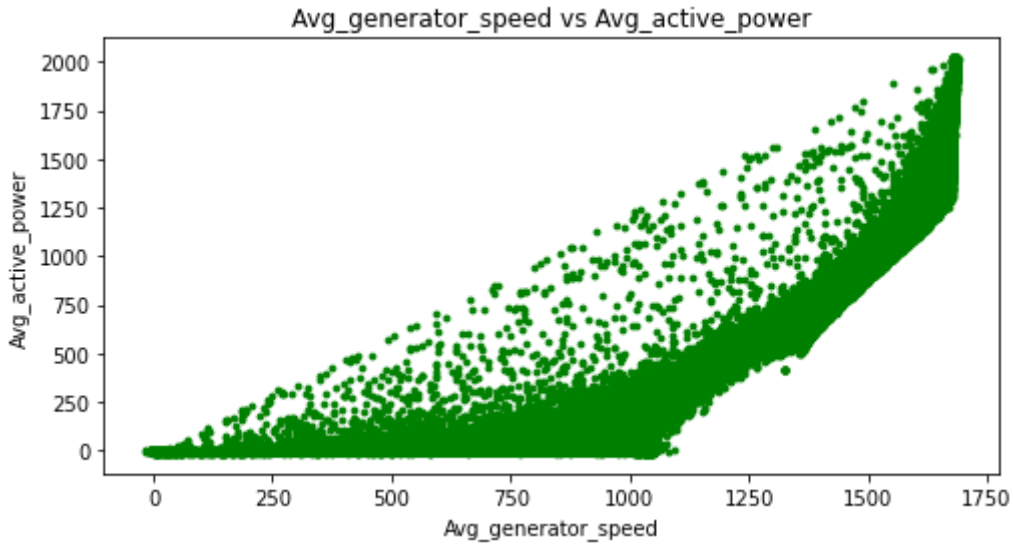
POWER CURVE - Average Wind Speed Vs Average Actvie Power

In [26]:
```
Scatter_Plot(df.iloc[:,7],df.iloc[:,1], 8, 'green')
```



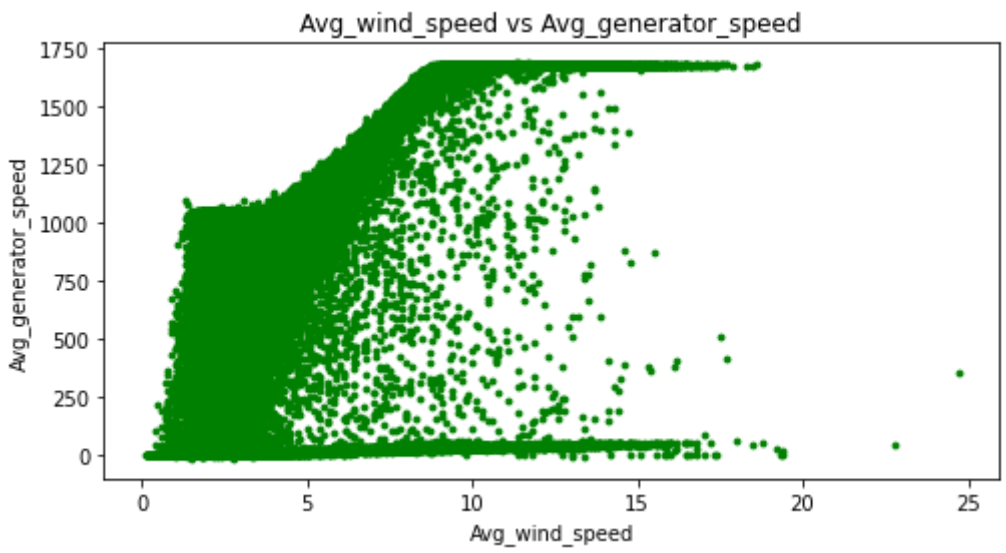
Average Generator Speed Vs Average Actvie Power

In [27]:
```
Scatter_Plot(df.iloc[:,3],df.iloc[:,1], 8, 'green')
```



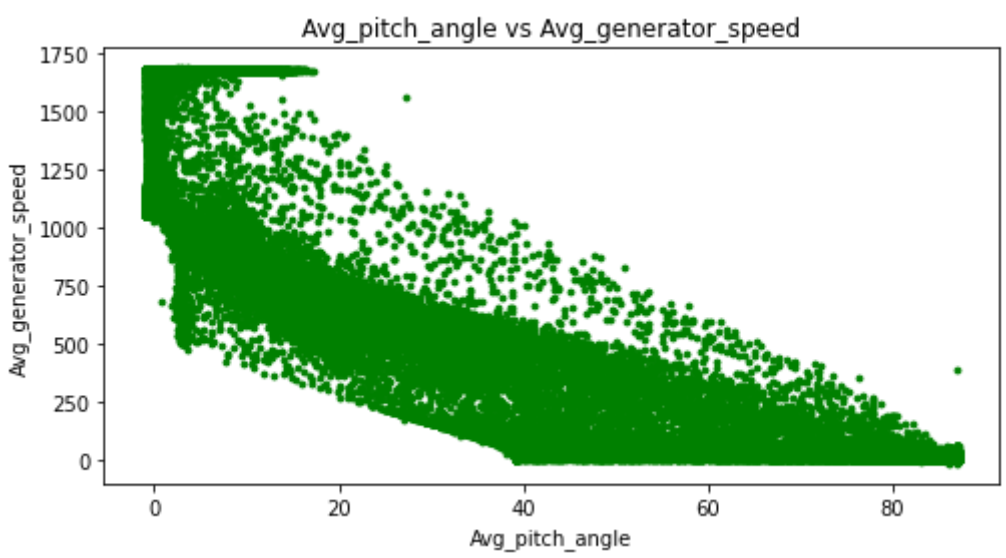
Average Generator Speed Vs Average Wind Speed

In [28]:
```
Scatter_Plot(df.iloc[:,7],df.iloc[:,3], 8, 'green')
```



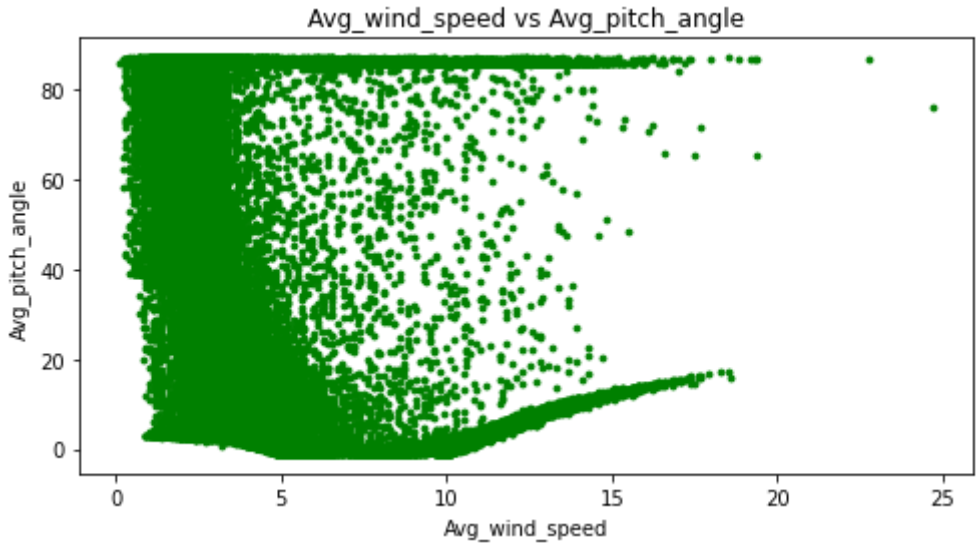
Average Pitch Angle Vs Average Generator Speed

In [29]:
```
Scatter_Plot(df.iloc[:,5],df.iloc[:,3], 8, 'green')
```



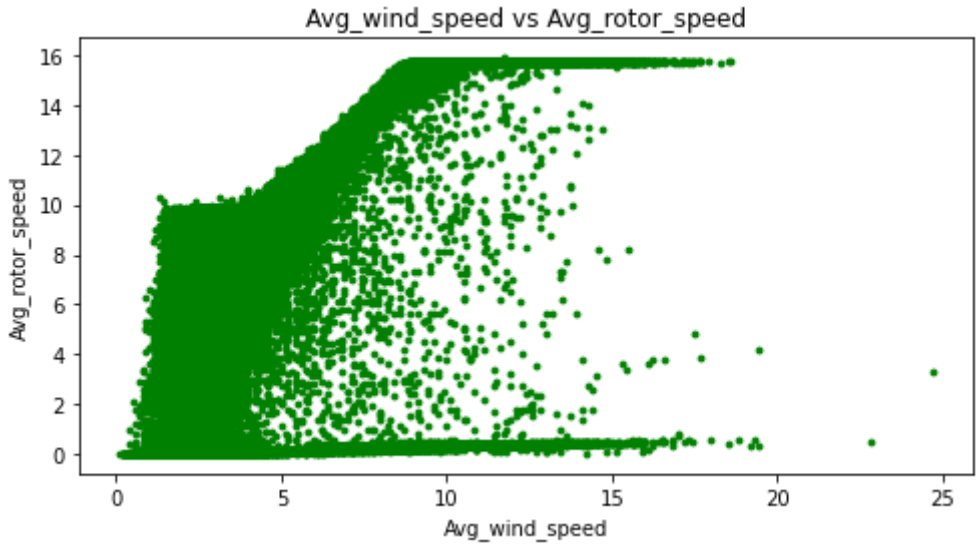
Average Pitch Angle Vs Average Wind Speed

In [30]:
```
Scatter_Plot(df.iloc[:,7],df.iloc[:,5], 8, 'green')
```



Average Wind Speed Vs Average Rotor Speed

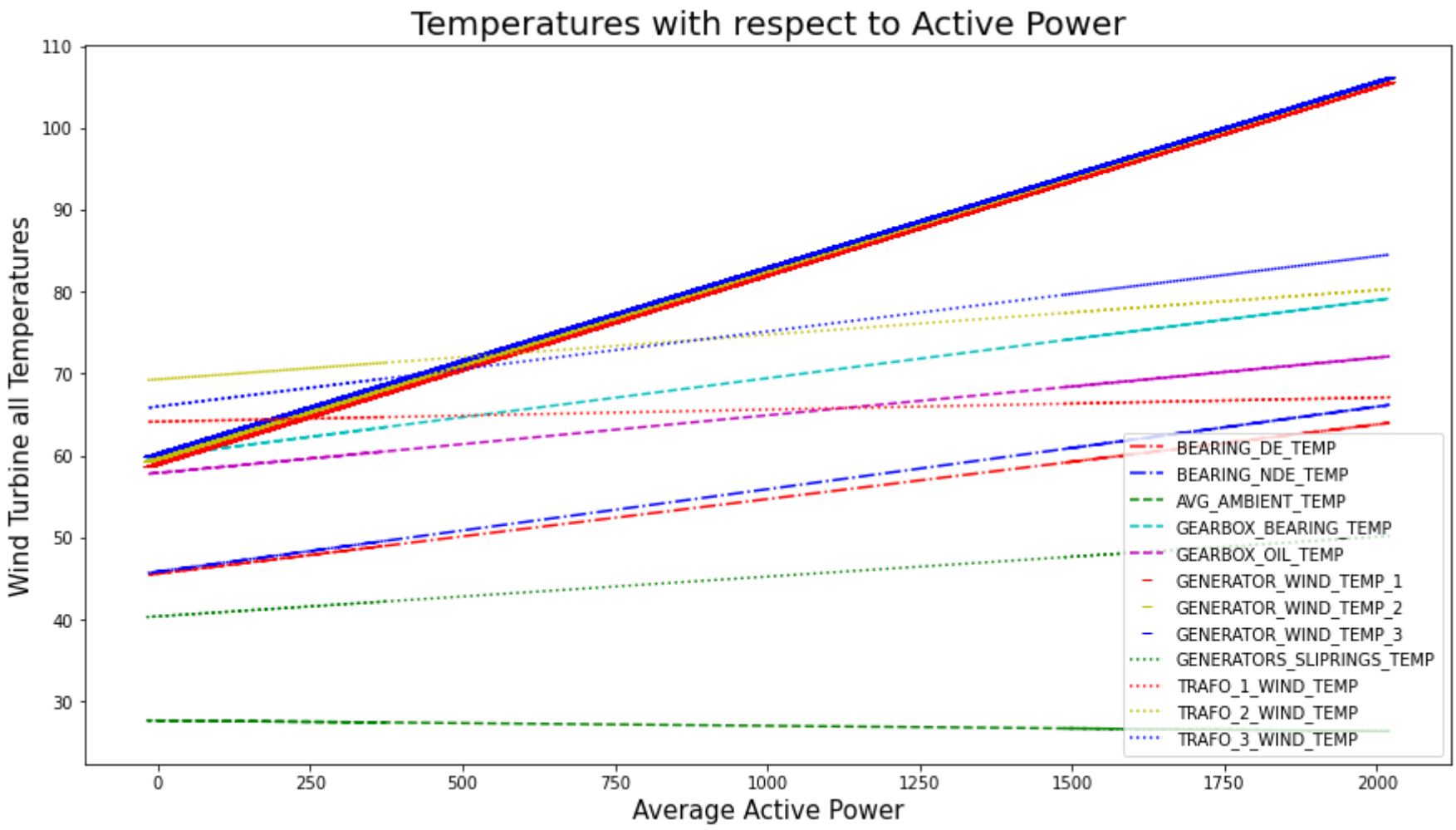
In [31]:
```
Scatter_Plot(df.iloc[:,7],df.iloc[:,6], 8, 'green')
```



All Temperature Parameters Vs Active Power

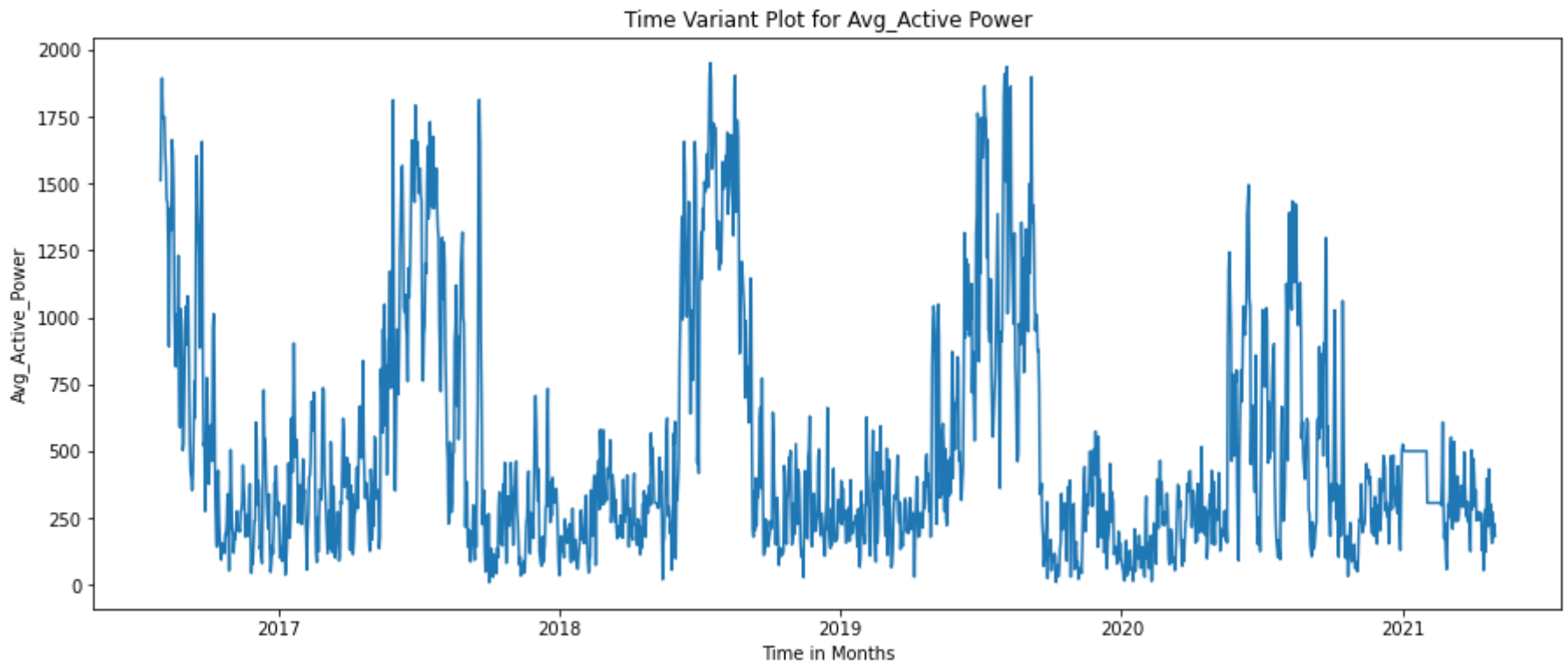
In [32]:
```
# line 1 points

x1 = df.iloc[:,1]
y1 = df.iloc[:,8]

z1 = np.polyfit(x1, y1, 1)
p1 = np.poly1d(z1)
plt.plot(x1,p1(x1),"r-.", label = df.iloc[:,8].name.upper())

# line 2 points
x2 = df.iloc[:,1]
y2 = df.iloc[:,9]

z2 = np.polyfit(x2, y2, 1)
p2 = np.poly1d(z2)
plt.plot(x2,p2(x2),"b-.",label = df.iloc[:,9].name.upper())

# line 3 points
x3 = df.iloc[:,1]
y3 = df.iloc[:,2]

z3 = np.polyfit(x3, y3, 1)
p3 = np.poly1d(z3)
plt.plot(x3,p3(x3),"g--", label = df.iloc[:,2].name.upper())

# line 4 points
x4 = df.iloc[:,1]
y4 = df.iloc[:,10]

z4 = np.polyfit(x4, y4, 1)
p4 = np.poly1d(z4)
plt.plot(x4,p4(x4),"c--",label = df.iloc[:,10].name.upper())

# line 5 points
x5 = df.iloc[:,1]
y5 = df.iloc[:,11]

z5 = np.polyfit(x5, y5, 1)
p5 = np.poly1d(z5)
plt.plot(x5,p5(x5),"m--", label = df.iloc[:,11].name.upper())

# line 6 points
x6 = df.iloc[:,1]
y6 = df.iloc[:,12]

z6 = np.polyfit(x6, y6, 1)
p6 = np.poly1d(z6)
plt.plot(x6,p6(x6),"r_",label = df.iloc[:,12].name.upper())

# line 7 points
x7 = df.iloc[:,1]
y7 = df.iloc[:,13]

z7 = np.polyfit(x7, y7, 1)
p7 = np.poly1d(z7)
plt.plot(x7,p7(x7),"y_",label = df.iloc[:,13].name.upper())

# line 8 points
x8 = df.iloc[:,1]
y8 = df.iloc[:,14]

z8 = np.polyfit(x8, y8, 1)
p8 = np.poly1d(z8)
plt.plot(x8,p8(x8),"b_",label = df.iloc[:,14].name.upper())

# line 9 points
x9 = df.iloc[:,1]
y9 = df.iloc[:,15]

z9 = np.polyfit(x9, y9, 1)
p9 = np.poly1d(z9)
plt.plot(x9,p9(x9),"g:",label = df.iloc[:,15].name.upper())

# line 10 points
x10 = df.iloc[:,1]
y10 = df.iloc[:,18]

z10 = np.polyfit(x10, y10, 1)
p10 = np.poly1d(z10)
plt.plot(x10,p10(x10),"r:",label = df.iloc[:,18].name.upper())

# line 11 points
x11 = df.iloc[:,1]
y11 = df.iloc[:,19]

z11 = np.polyfit(x11, y11, 1)
p11 = np.poly1d(z11)
plt.plot(x11,p11(x11),"y:",label = df.iloc[:,19].name.upper())
```

```
# line 12 points
x12 = df.iloc[:,1]
y12 = df.iloc[:,20]

z12 = np.polyfit(x12, y12, 1)
p12 = np.poly1d(z12)
plt.plot(x12,p12(x12),"b:",label = df.iloc[:,20].name.upper())


# Set the y & x axis label of the current axis.
plt.ylabel('Wind Turbine all Temperatures', fontsize = 15)
plt.xlabel('Average Active Power', fontsize = 15)
# Set a title of the current axes.
plt.title('Temperatures with respect to Active Power', fontsize = 20)
# show a legend on the plot
plt.legend()
# Display a figure.
plt.subplots_adjust(left=0.5, bottom=0, right=2.4, top=1.5)
plt.show()
```



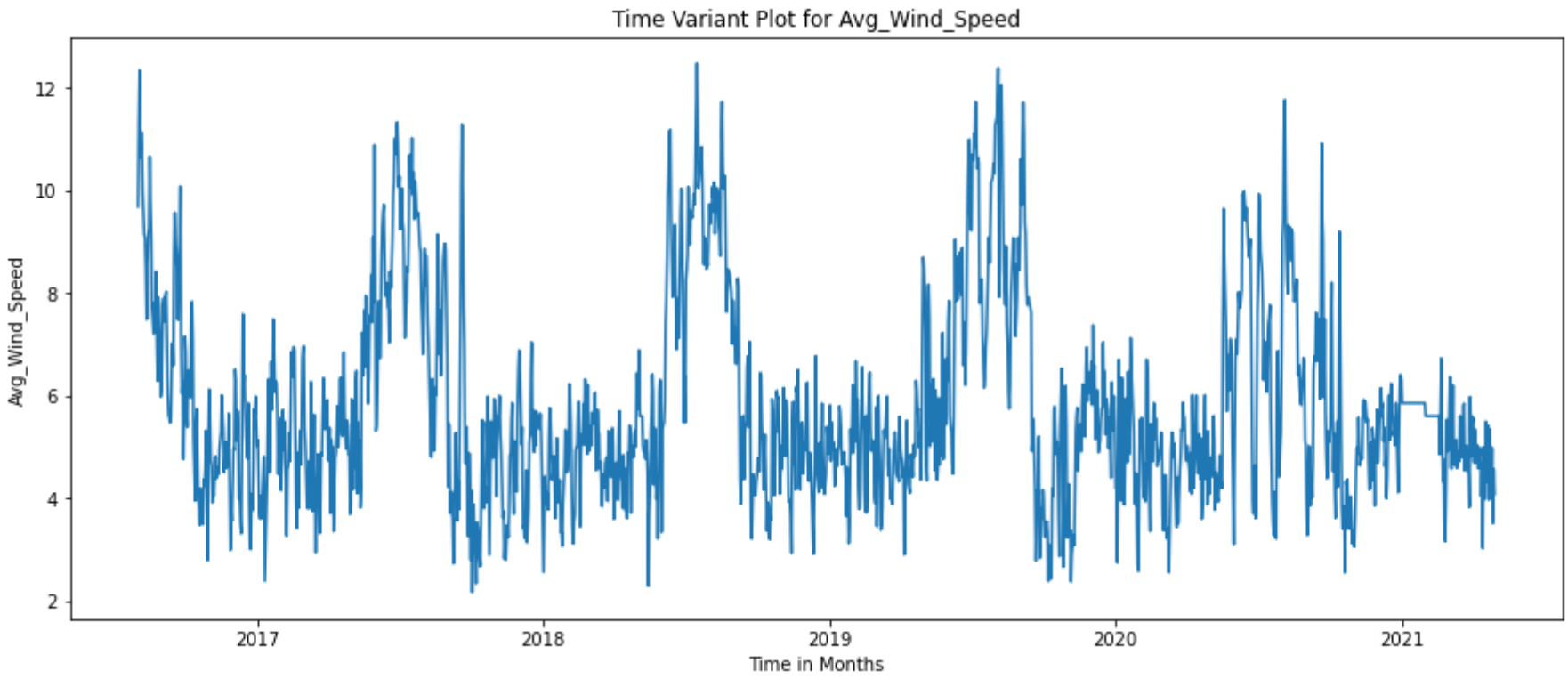
ACTIVE POWER TREND DURING A TIME PERIOD

In [33]:
```
pf1 = df[['Date','Avg_Active_Power']]
pf1 = pf1.set_index('Date')
y1 = pf1['Avg_Active_Power'].resample('D').mean()
y1 = y1.fillna(y1.mean())
plt.figure(figsize=(15,6))
plt.plot(y1)
plt.title('Time Variant Plot for Avg_Active Power')
plt.xlabel("Time in Days")
plt.ylabel("Avg_Active_Power")
plt.show()
```



WIND SPEED TREND OVER A TIME PERIOD

In [34]:
```
pf2 = df[['Date','Avg_Wind_Speed']]
pf2 = pf2.set_index('Date')
y2 = pf2['Avg_Wind_Speed'].resample('D').mean()
y2 = y2.fillna(y2.mean())
plt.figure(figsize=(15,6))
plt.plot(y2)
plt.title('Time Variant Plot for Avg_Wind_Speed')
plt.xlabel("Time in Days")
plt.ylabel("Avg_Wind_Speed")
plt.show()
```



WIND ROSE PLOTS

In [35]:
```
pf3 = df[['Avg_Wind_Speed', 'Avg_Nacelle_Pos','Nacelle_Misalignment_Avg_Wind_Dir', 'Avg_Active_Power']]
pf3
```

Out[35]:

	Avg_Wind_Speed	Avg_Nacelle_Pos	Nacelle_Misalignment_Avg_Wind_Dir	Avg_Active_Power
0	9.3	138.0	357.0	1487.3
1	9.3	138.0	356.9	1647.6
2	9.0	138.0	356.5	1506.4
3	8.3	138.0	356.5	1240.9
4	8.5	138.0	356.2	1202.4
...	...	...	...	...
245194	4.7	1.0	0.6	205.1
245195	4.7	1.0	3.9	208.4
245196	5.2	355.7	355.1	228.0
245197	5.7	354.0	3.8	355.1
245198	5.9	0.0	2.6	374.3

245199 rows × 4 columns

In [36]:
```
pf3['Direction'] = np.where((pf3['Avg_Nacelle_Pos']>337.5) & (pf3['Avg_Nacelle_Pos']<=22.5),"0N","0N")

pf3['Direction'] = np.where((pf3['Avg_Nacelle_Pos']>22.5) & (pf3['Avg_Nacelle_Pos']<=67.5),"1NE",pf3['Direction'])

pf3['Direction'] = np.where((pf3['Avg_Nacelle_Pos']>67.5) & (pf3['Avg_Nacelle_Pos']<=112.5),"2E",pf3['Direction'])

pf3['Direction'] = np.where((pf3['Avg_Nacelle_Pos']>112.5) & (pf3['Avg_Nacelle_Pos']<=157.7),"3SE",pf3['Direction'])

pf3['Direction'] = np.where((pf3['Avg_Nacelle_Pos']>157.7) & (pf3['Avg_Nacelle_Pos']<=202.5),"4S",pf3['Direction'])

pf3['Direction'] = np.where((pf3['Avg_Nacelle_Pos']>202.5) & (pf3['Avg_Nacelle_Pos']<=247.5),"5SW",pf3['Direction'])

pf3['Direction'] = np.where((pf3['Avg_Nacelle_Pos']>247.5) & (pf3['Avg_Nacelle_Pos']<=292.5),"6W",pf3['Direction'])

pf3['Direction'] = np.where((pf3['Avg_Nacelle_Pos']>292.5) & (pf3['Avg_Nacelle_Pos']<=337.5),"7NW",pf3['Direction'])
```

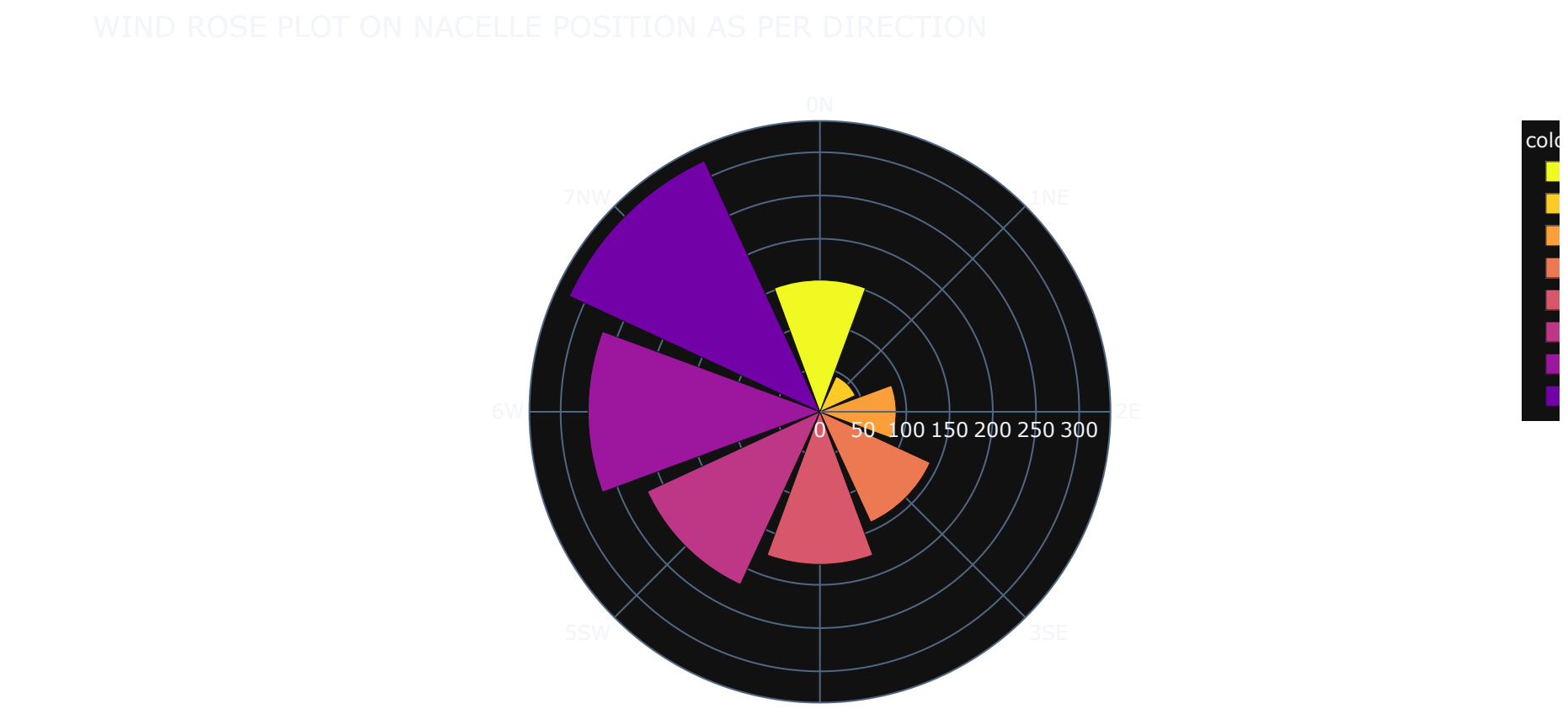
In [37]:
```
pf4 = pf3.groupby('Direction').mean()
pf4
```

Out[37]:

Direction	Avg_Wind_Speed	Avg_Nacelle_Pos	Nacelle_Misalignment_Avg_Wind_Dir	Avg_Active_Power
0N	5.329365	152.206170	165.683403	302.621934
1NE	4.756077	44.900122	163.068175	255.076733
2E	4.908596	88.145161	164.629859	291.300842
3SE	6.365296	140.367936	150.401662	653.891086
4S	7.220002	176.387014	161.040035	798.247221
5SW	5.285135	219.814881	187.714309	439.715250
6W	3.880825	268.179053	169.902159	156.240819
7NW	4.361006	319.378438	156.757856	198.279020

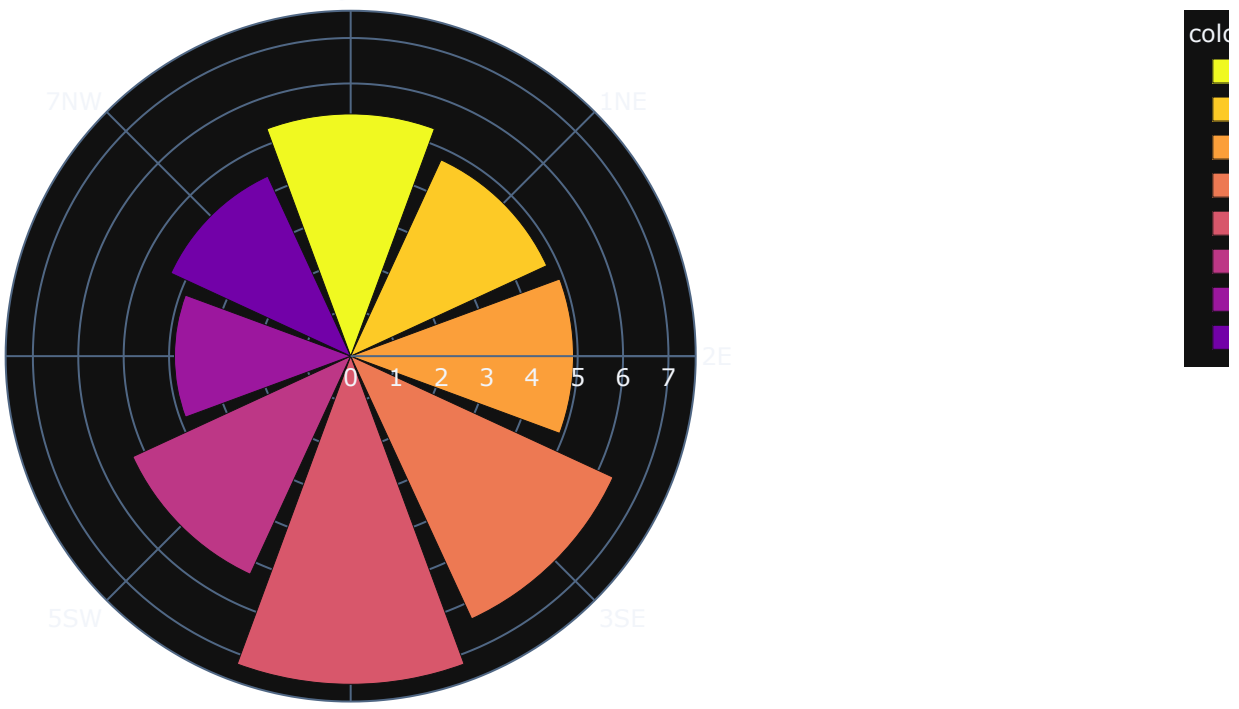
NACELLE POSITION AS PER WIND DIRECTION

In [38]:
```
# !pip install plotly
import plotly.express as px
fig = px.bar_polar(pf4, r="Avg_Nacelle_Pos", theta=pf4.index.values,
                   color=pf4.index.values, template="plotly_dark",
                   color_discrete_sequence= px.colors.sequential.Plasma_r, title = 'WIND ROSE PLOT ON NACELLE POSITION AS
fig.show()
```



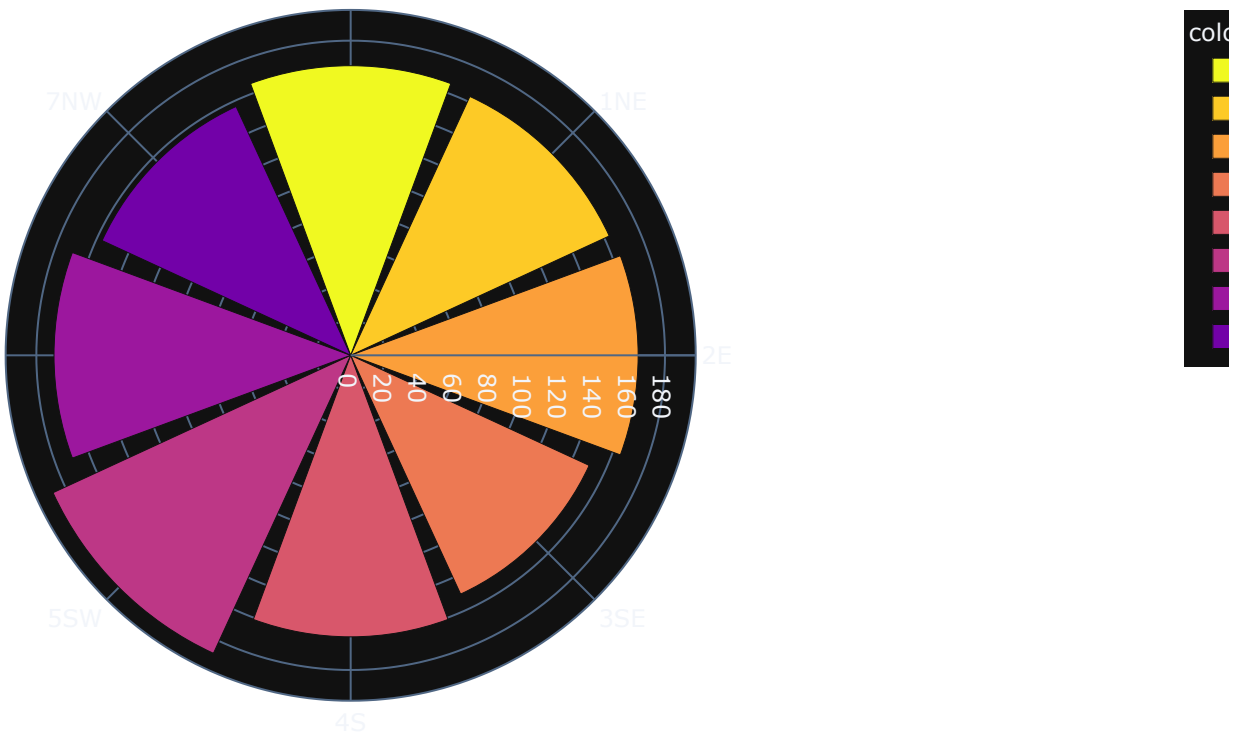
AVERAGE WIND DIRECTION

In [39]:
```
fig = px.bar_polar(pf4, r="Avg_Wind_Speed", theta=pf4.index.values,
                   color=pf4.index.values, template="plotly_dark",
                   color_discrete_sequence= px.colors.sequential.Plasma_r, title = 'WIND ROSE PLOT ON AVERAGE WIND DIREC
fig.show()
```



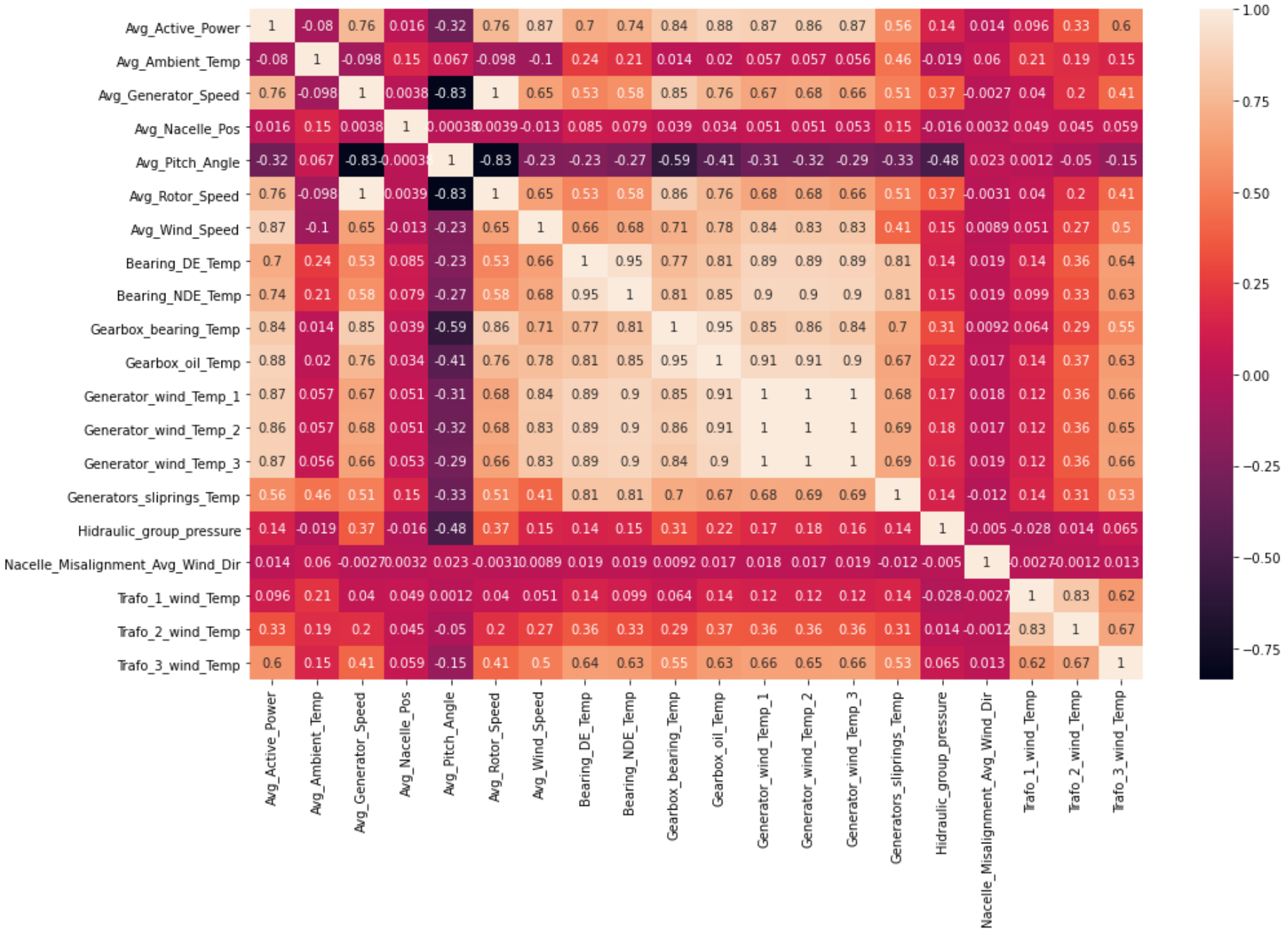
AVERAGE NACELLE MISALIGNMENT WITH RESPECT TO WIND DIRECTION

In [40]:
```
fig = px.bar_polar(pf4, r="Nacelle_Misalignment_Avg_Wind_Dir", theta=pf4.index.values,
                   color=pf4.index.values, template="plotly_dark",
                   color_discrete_sequence= px.colors.sequential.Plasma_r, title = 'AVERAGE NACELLE MISALIGNMENT WITH RES
fig.show()
```



CORRELATION PLOT

In [41]:
```
sns.heatmap(df.corr(), annot = True)
plt.subplots_adjust(left=0.8, bottom=0, right=2.8, top=1.8)
plt.show()
```



Thank You.....