```
1 !pip install -r rec.txt
```

```
Collecting numpy==1.19.5 (from -r rec.txt (line 1))
  Using cached numpy-1.19.5.zip (7.3 MB)
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (fro
Collecting matplotlib==3.3.4 (from -r rec.txt (line 3))
  Using cached matplotlib-3.3.4.tar.gz (37.9 MB)
  Preparing metadata (setup.py) ... done
Collecting seaborn==0.11.1 (from -r rec.txt (line 4))
  Using cached seaborn-0.11.1-py3-none-any.whl.metadata (2.3 kB)
Collecting scikit-learn==0.24.1 (from -r rec.txt (line 5))
  Using cached scikit-learn-0.24.1.tar.gz (7.4 MB)
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Collecting scipy==1.5.4 (from -r rec.txt (line 6))
  Using cached scipy-1.5.4.tar.gz (25.2 MB)
  error: subprocess-exited-with-error

  × pip subprocess to install build dependencies did not run successfully.
  │ exit code: 1
  ╰─> See above for output.

  note: This error originates from a subprocess, and is likely not a problem with pip
  Installing build dependencies ... error
error: subprocess-exited-with-error

× pip subprocess to install build dependencies did not run successfully.
│ exit code: 1
╰─> See above for output.

note: This error originates from a subprocess, and is likely not a problem with pip.
```

```
1 !pip install catboost
```

```
Requirement already satisfied: catboost in /usr/local/lib/python3.10/dist-packages (1
Requirement already satisfied: graphviz in /usr/local/lib/python3.10/dist-packages (f
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: numpy<2.0,>=1.16.0 in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: pandas>=0.24 in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: plotly in /usr/local/lib/python3.10/dist-packages (fro
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from c
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/di
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-pac
```

```
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.10/dist-pack
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import preprocessing
import scipy.stats as stats
from sklearn.model_selection import train_test_split
from collections import Counter
from imblearn.over_sampling import SMOTE
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
from sklearn import metrics
from sklearn.ensemble import RandomForestClassifier
from catboost import CatBoostClassifier
from xgboost import XGBClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
import joblib
```

```
df = pd.read_csv("processed_weather_data.csv")
pd.set_option("display.max_columns", None)
df
```

	Date	Location	Temp	WindSpeed	Humidity	RainToday	RainTomorrow
0	2008-12-01	Albury	18.15	22.0	46.5	No	No
1	2008-12-02	Albury	16.25	13.0	34.5	No	No
2	2008-12-03	Albury	19.30	22.5	34.0	No	No
3	2008-12-04	Albury	18.60	10.0	30.5	No	No
4	2008-12-05	Albury	24.90	13.5	57.5	No	No
...	...	...	...	...	...	...	...
145455	2017-06-21	Uluru	13.10	12.0	37.5	No	No
145456	2017-06-22	Uluru	14.45	11.0	38.5	No	No
145457	2017-06-23	Uluru	16.15	9.0	38.5	No	No
145458	2017-06-24	Uluru	17.40	10.0	37.5	No	No
145459	2017-06-25	Uluru	14.90	17.0	49.0	No	NaN

145460 rows × 7 columns

```
numerical_feature = [feature for feature in df.columns if df[feature].dtypes != 'O']
discrete_feature=[feature for feature in numerical_feature if len(df[feature].unique()
continuous_feature = [feature for feature in numerical_feature if feature not in discr
categorical_feature = [feature for feature in df.columns if feature not in numerical_f
```

```
print("Numerical Features Count {}".format(len(numerical_feature)))
print("Discrete feature Count {}".format(len(discrete_feature)))
print("Continuous feature Count {}".format(len(continuous_feature)))
print("Categorical feature Count {}".format(len(categorical_feature)))
```

```
Numerical Features Count 3
Discrete feature Count 0
Continuous feature Count 3
Categorical feature Count 4
```

```
# Handle Missing Values
df.isnull().sum()*100/len(df)
```

	0
Date	0.000000
Location	0.000000
Temp	0.594665
WindSpeed	0.833906
Humidity	1.297264
RainToday	2.241853
RainTomorrow	2.245978

dtype: float64

```
print(numerical_feature)
```

```
['Temp', 'WindSpeed', 'Humidity']
```

```
def randomsampleimputation(df, variable):
    df[variable]=df[variable]
    random_sample=df[variable].dropna().sample(df[variable].isnull().sum(),random_stat
    random_sample.index=df[df[variable].isnull()].index
    df.loc[df[variable].isnull(),variable]=random_sample
```

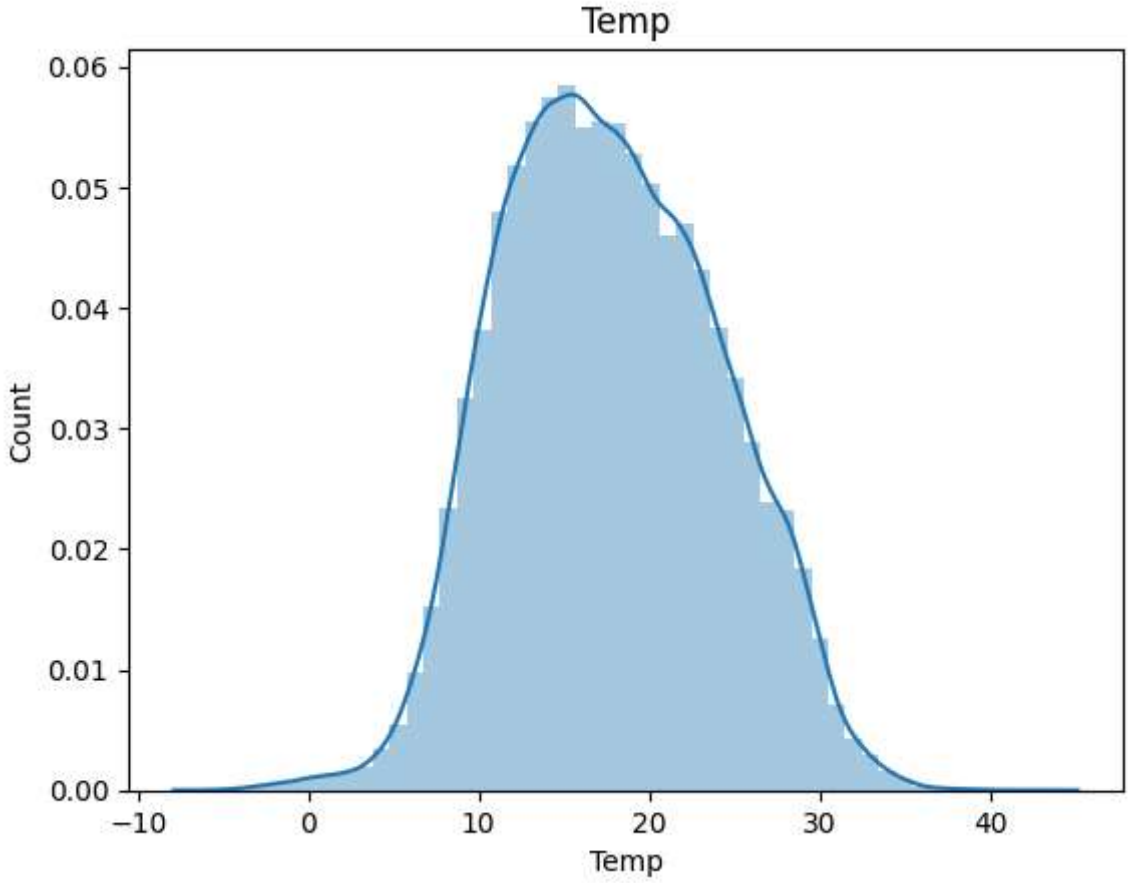
```
df
```

	Date	Location	Temp	WindSpeed	Humidity	RainToday	RainTomorrow
0	2008-12-01	Albury	18.15	22.0	46.5	No	No
1	2008-12-02	Albury	16.25	13.0	34.5	No	No
2	2008-12-03	Albury	19.30	22.5	34.0	No	No
3	2008-12-04	Albury	18.60	10.0	30.5	No	No
4	2008-12-05	Albury	24.90	13.5	57.5	No	No
...	...	...	...	...	...	...	...
145455	2017-06-21	Uluru	13.10	12.0	37.5	No	No
145456	2017-06-22	Uluru	14.45	11.0	38.5	No	No
145457	2017-06-23	Uluru	16.15	9.0	38.5	No	No
145458	2017-06-24	Uluru	17.40	10.0	37.5	No	No
145459	2017-06-25	Uluru	14.90	17.0	49.0	No	NaN

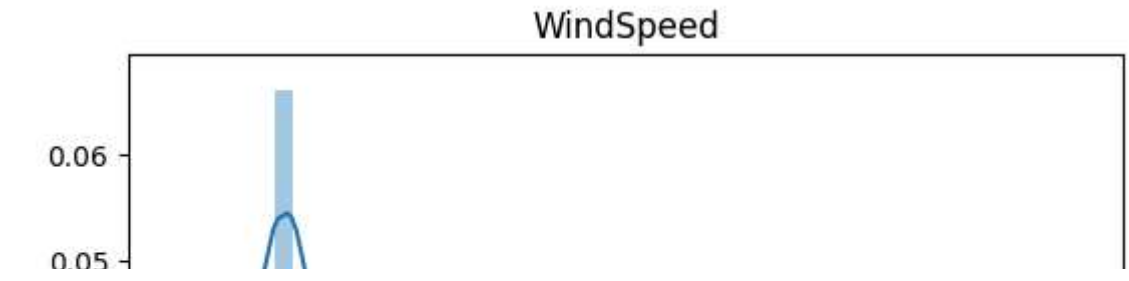
145460 rows × 7 columns

```
for feature in continuous_feature:
    data=df.copy()
    sns.distplot(df[feature])
    plt.xlabel(feature)
    plt.ylabel("Count")
    plt.title(feature)
    plt.figure(figsize=(15,15))
    plt.show()
```

```
<ipython-input-10-f3f36df57cc9>:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df[feature])
```

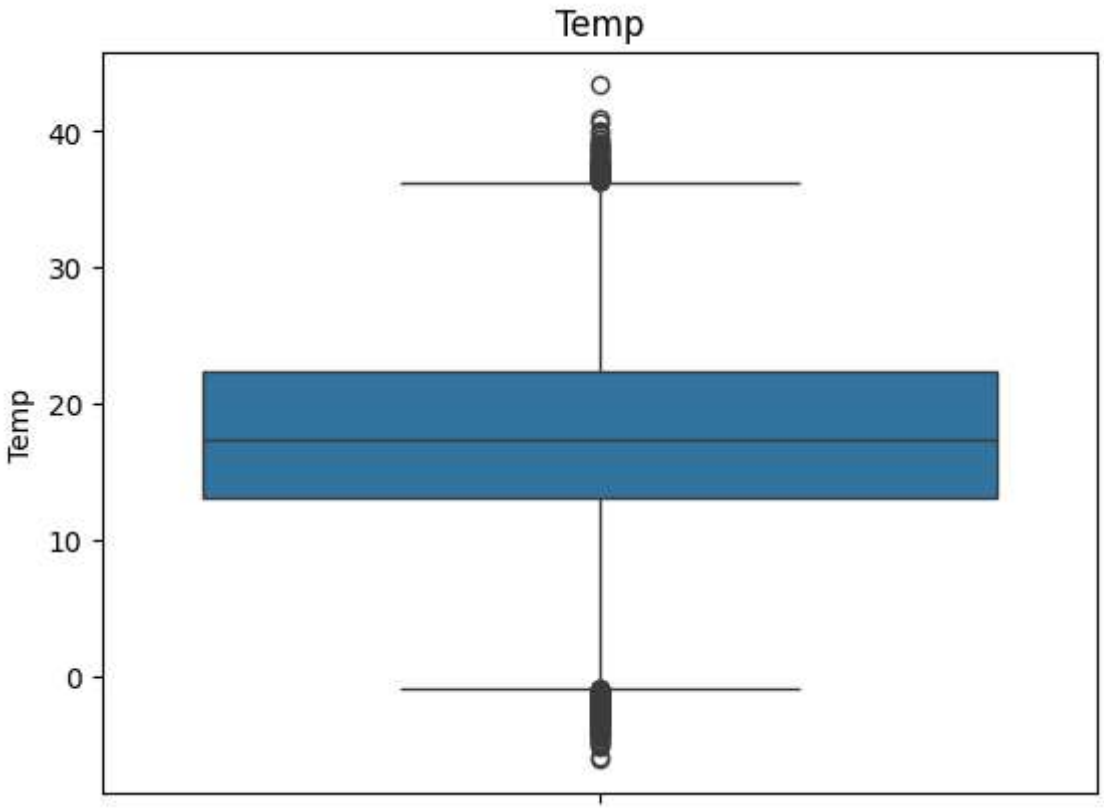


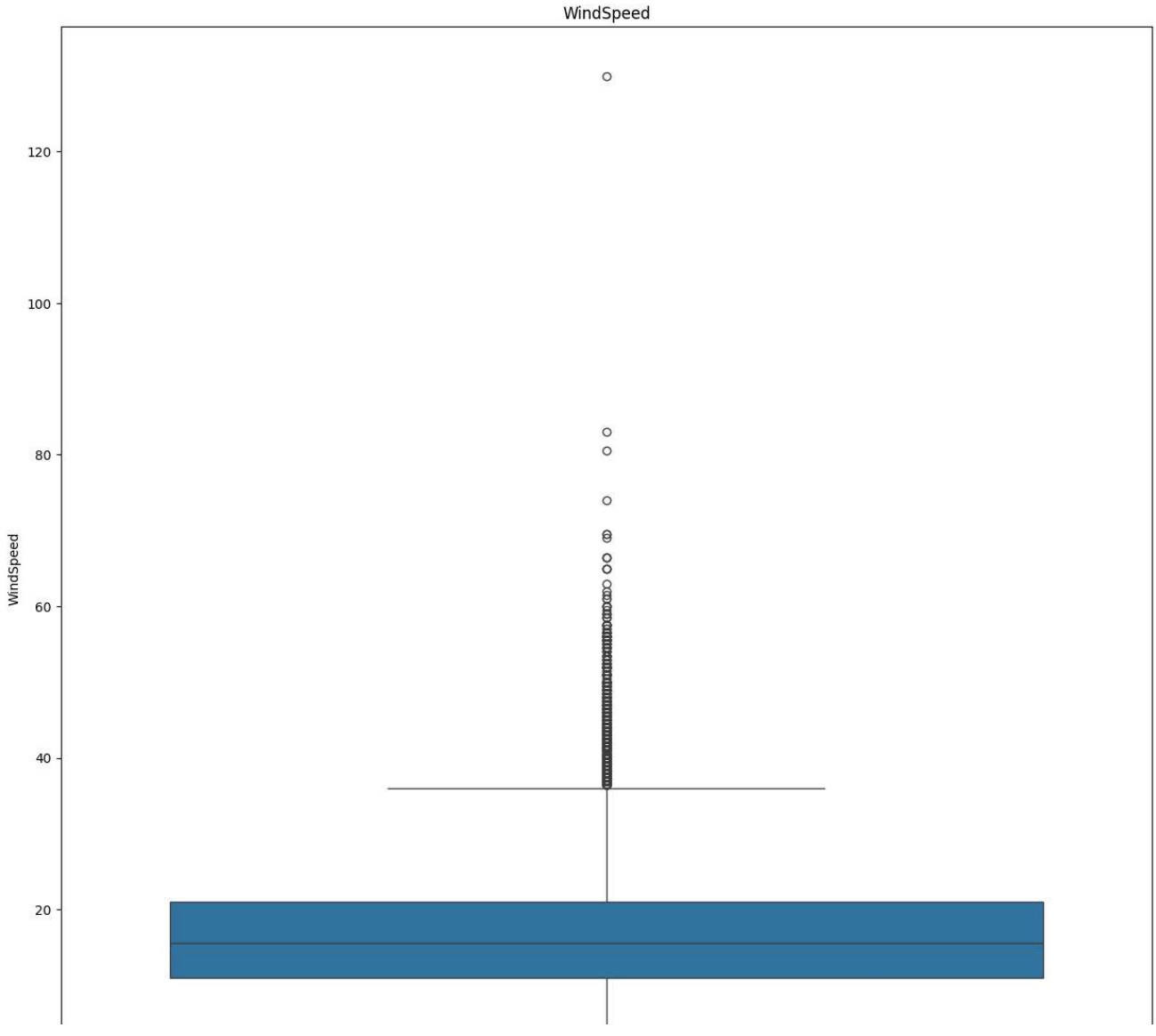

```
<Figure size 1500x1500 with 0 Axes>
<ipython-input-10-f3f36df57cc9>:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df[feature])
```




```
1 #A for loop is used to plot a boxplot for all the continuous features to see the outli
2 for feature in continuous_feature:
```

```
    data=df.copy()
    sns.boxplot(data[feature])
    plt.title(feature)
    plt.figure(figsize=(15,15))
```






```
1 for feature in continuous_feature:
2     if(df[feature].isnull().sum()*100/len(df))>0:
```

```
3         df[feature] = df[feature].fillna(df[feature].median())
```

```
1 df.isnull().sum()*100/len(df)
```

	0
Date	0.000000
Location	0.000000
Temp	0.000000
WindSpeed	0.000000
Humidity	0.000000
RainToday	2.241853
RainTomorrow	2.245978

dtype: float64

```
1 discrete_feature
```

```
[]
```

```
1 def mode_nan(df,variable):
2     mode=df[variable].value_counts().index[0]
3     df[variable].fillna(mode,inplace=True)
4
```

```
1 df["RainToday"] = pd.get_dummies(df["RainToday"], drop_first = True)
2 df["RainTomorrow"] = pd.get_dummies(df["RainTomorrow"], drop_first = True)
3 df
```

	Date	Location	Temp	WindSpeed	Humidity	RainToday	RainTomorrow
0	2008-12-01	Albury	18.15	22.0	46.5	False	False
1	2008-12-02	Albury	16.25	13.0	34.5	False	False
2	2008-12-03	Albury	19.30	22.5	34.0	False	False
3	2008-12-04	Albury	18.60	10.0	30.5	False	False
4	2008-12-05	Albury	24.90	13.5	57.5	False	False
...	...	...	...	...	...	...	...
145455	2017-06-21	Uluru	13.10	12.0	37.5	False	False
145456	2017-06-22	Uluru	14.45	11.0	38.5	False	False
145457	2017-06-23	Uluru	16.15	9.0	38.5	False	False
145458	2017-06-24	Uluru	17.40	10.0	37.5	False	False
145459	2017-06-25	Uluru	14.90	17.0	49.0	False	False

145460 rows × 7 columns

```
1 for feature in categorical_feature:
2     print(feature, (df.groupby([feature])["RainTomorrow"].mean().sort_values(ascending
```

```
Date Index(['2007-11-01', '2007-12-15', '2008-02-03', '2008-01-31', '2008-01-30',
       '2008-01-19', '2008-01-18', '2008-01-16', '2008-01-12', '2007-11-30',
       ...
       '2008-05-08', '2008-05-09', '2008-01-03', '2008-01-02', '2008-05-10',
       '2008-05-11', '2008-05-12', '2008-05-13', '2008-05-14', '2007-12-25'],
      dtype='object', name='Date', length=3436)
Location Index(['Portland', 'Walpole', 'Cairns', 'Dartmoor', 'NorfolkIsland',
       'MountGambier', 'Albany', 'Witchcliffe', 'CoffsHarbour', 'MountGinini',
       'NorahHead', 'Darwin', 'Sydney', 'SydneyAirport', 'Ballarat',
       'GoldCoast', 'Watsonia', 'Newcastle', 'Hobart', 'Wollongong',
       'Williamtown', 'Launceston', 'Brisbane', 'MelbourneAirport', 'Adelaide',
       'Sale', 'Albury', 'Perth', 'Melbourne', 'Nuriootpa', 'Penrith',
       'BadgerysCreek', 'PerthAirport', 'Tuggeranong', 'Richmond', 'Bendigo',
       'Canberra', 'WaggaWagga', 'Townsville', 'Katherine', 'PearceRAAF',
       'SalmonGums', 'Nhil', 'Moree', 'Cobar', 'Mildura', 'AliceSprings',
       'Uluru', 'Woomera'],
      dtype='object', name='Location')
RainToday Index([True, False], dtype='bool', name='RainToday')
RainTomorrow Index([True, False], dtype='bool', name='RainTomorrow')
```

```
1 del(df["Location"])
2 df
```

	Date	Temp	WindSpeed	Humidity	RainToday	RainTomorrow
0	2008-12-01	18.15	22.0	46.5	False	False
1	2008-12-02	16.25	13.0	34.5	False	False
2	2008-12-03	19.30	22.5	34.0	False	False
3	2008-12-04	18.60	10.0	30.5	False	False
4	2008-12-05	24.90	13.5	57.5	False	False
...	...	...	...	...	...	...
145455	2017-06-21	13.10	12.0	37.5	False	False
145456	2017-06-22	14.45	11.0	38.5	False	False
145457	2017-06-23	16.15	9.0	38.5	False	False
145458	2017-06-24	17.40	10.0	37.5	False	False
145459	2017-06-25	14.90	17.0	49.0	False	False

145460 rows × 6 columns

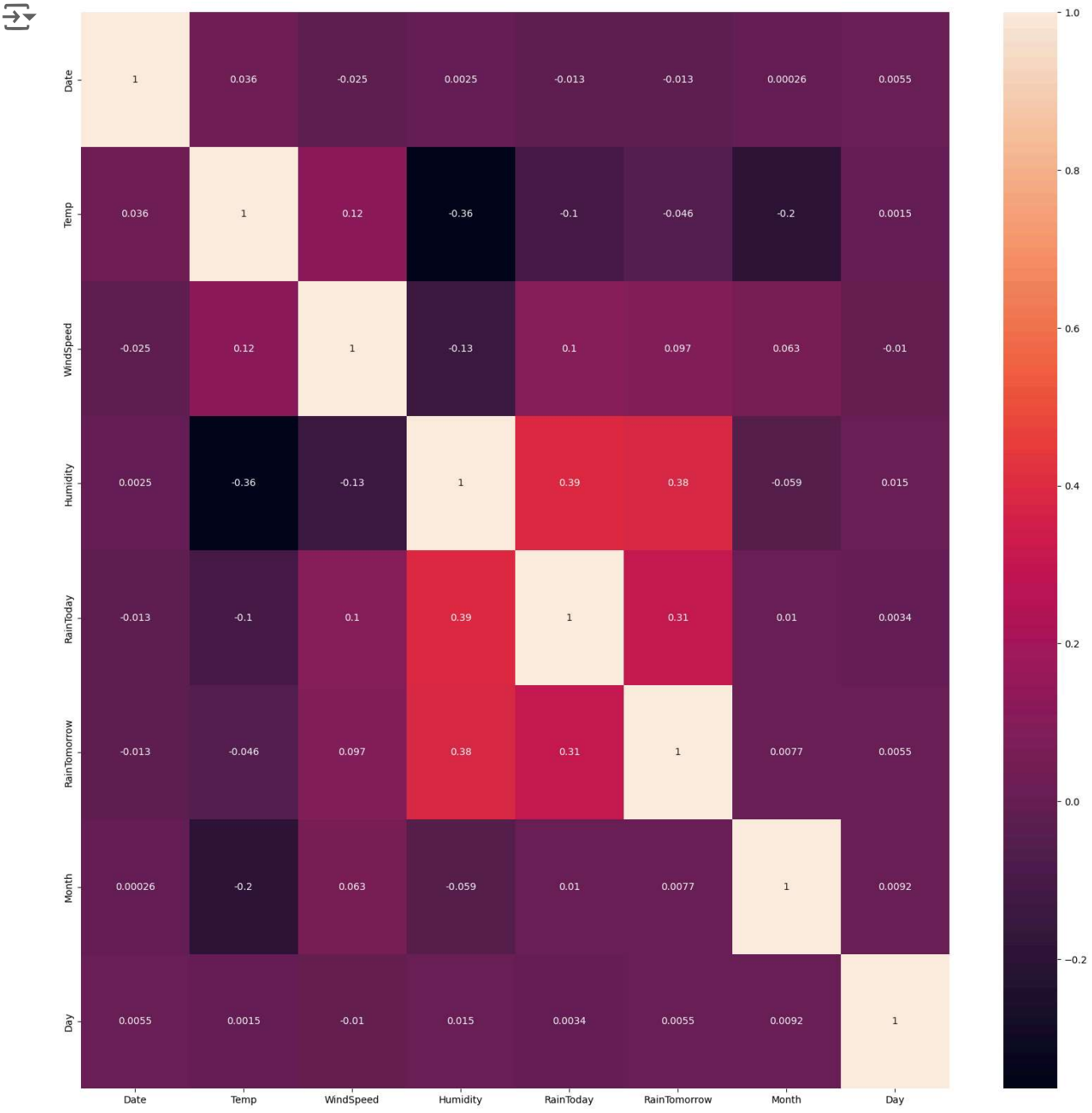
```
df['Date'] = pd.to_datetime(df['Date'])
df['Month'] = df['Date'].dt.month
df['Day'] = df['Date'].dt.day
df
```

	Date	Temp	WindSpeed	Humidity	RainToday	RainTomorrow	Month	Day
0	2008-12-01	18.15	22.0	46.5	False	False	12	1
1	2008-12-02	16.25	13.0	34.5	False	False	12	2
2	2008-12-03	19.30	22.5	34.0	False	False	12	3
3	2008-12-04	18.60	10.0	30.5	False	False	12	4
4	2008-12-05	24.90	13.5	57.5	False	False	12	5
...	...	...	...	...	...	...	...	...
145455	2017-06-21	13.10	12.0	37.5	False	False	6	21
145456	2017-06-22	14.45	11.0	38.5	False	False	6	22
145457	2017-06-23	16.15	9.0	38.5	False	False	6	23
145458	2017-06-24	17.40	10.0	37.5	False	False	6	24
145459	2017-06-25	14.90	17.0	49.0	False	False	6	25

145460 rows × 8 columns

```
corrmat = df.corr()
plt.figure(figsize=(20,20))
```

```
#plot heat map
g=sns.heatmap(corrmat,annot=True)
```

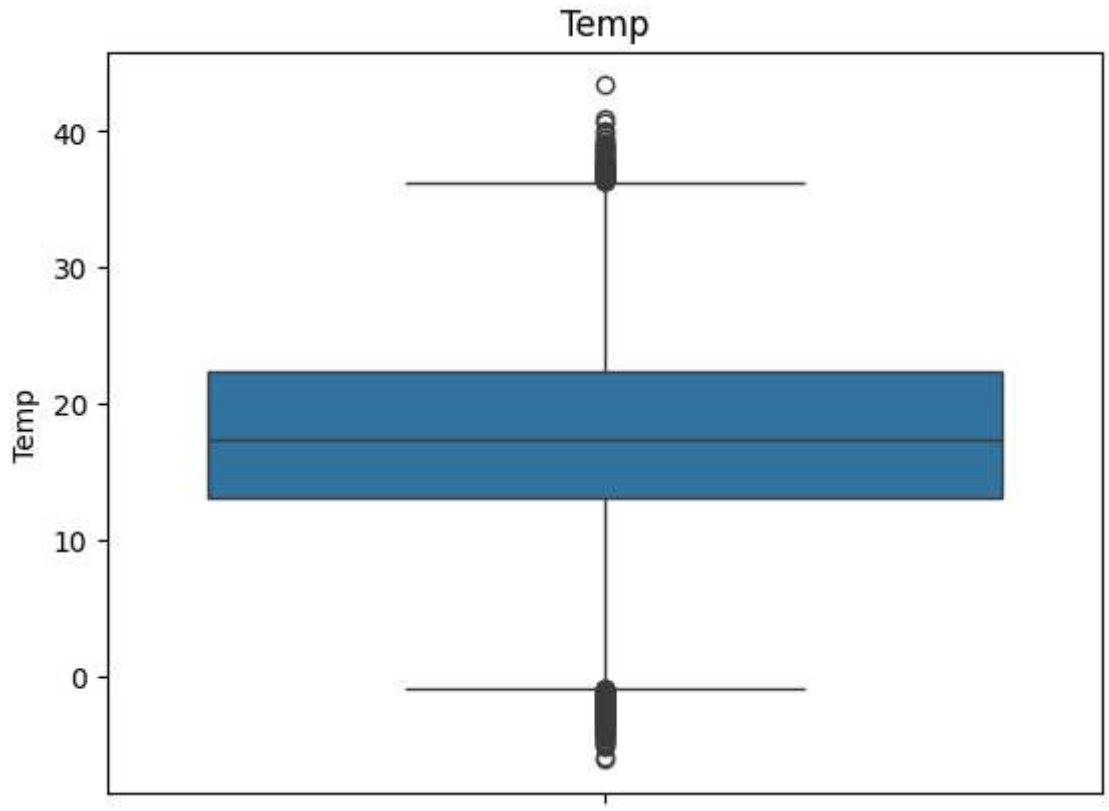

Date 1 0.036 -0.025 0.0025 -0.013 -0.013 0.00026 0.0055
Temp 0.036 1 0.12 -0.36 -0.1 -0.046 -0.2 0.0015
WindSpeed -0.025 0.12 1 -0.13 0.1 0.097 0.063 -0.01
Humidity 0.0025 -0.36 -0.13 1 0.39 0.38 -0.059 0.015
RainToday -0.013 -0.1 0.1 0.39 1 0.31 0.01 0.0034
RainTomorrow -0.013 -0.046 0.097 0.38 0.31 1 0.0077 0.0055
Month 0.00026 -0.2 0.063 -0.059 0.01 0.0077 1 0.0092
Day 0.0055 0.0015 -0.01 0.015 0.0034 0.0055 0.0092 1
Date Temp WindSpeed Humidity RainToday RainTomorrow Month Day
1.0 0.8 0.6 0.4 0.2 0.0 −0.2

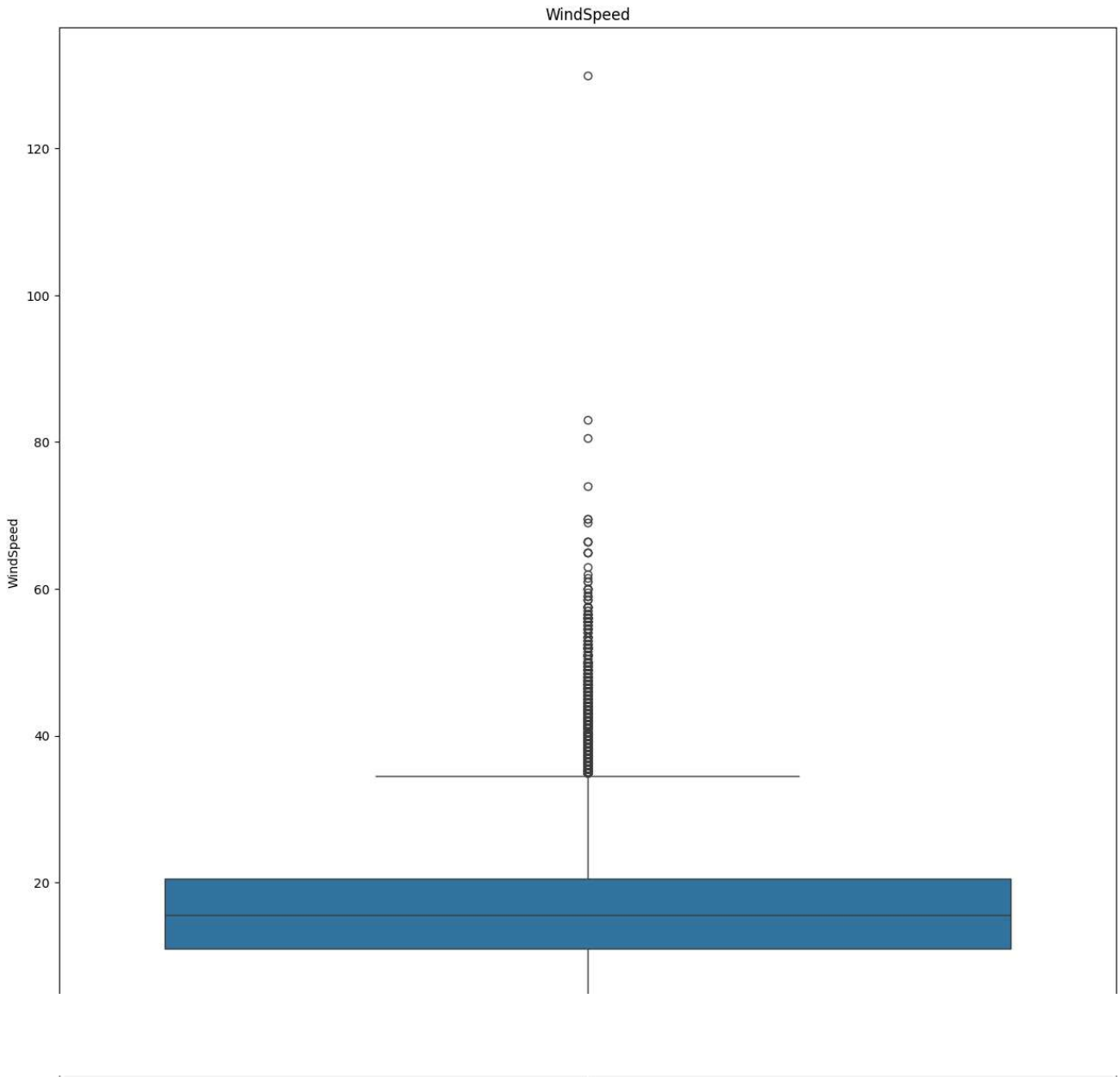
```
1 df
```

	Date	Temp	WindSpeed	Humidity	RainToday	RainTomorrow	Month	Day
0	2008-12-01	18.15	22.0	46.5	False	False	12	1
1	2008-12-02	16.25	13.0	34.5	False	False	12	2
2	2008-12-03	19.30	22.5	34.0	False	False	12	3
3	2008-12-04	18.60	10.0	30.5	False	False	12	4
4	2008-12-05	24.90	13.5	57.5	False	False	12	5
...	...	...	...	...	...	...	...	...
145455	2017-06-21	13.10	12.0	37.5	False	False	6	21
145456	2017-06-22	14.45	11.0	38.5	False	False	6	22
145457	2017-06-23	16.15	9.0	38.5	False	False	6	23
145458	2017-06-24	17.40	10.0	37.5	False	False	6	24
145459	2017-06-25	14.90	17.0	49.0	False	False	6	25

145460 rows × 8 columns

```
for feature in continuous_feature:
    data=df.copy()
    sns.boxplot(data[feature])
    plt.title(feature)
    plt.figure(figsize=(15,15))
```


Temp
Temp
40
30
20
10
0


WindSpeed
WindSpeed
120
100
80
60
40
20

```
for feature in continuous_feature:
    print(feature)
```

```
Temp
WindSpeed
Humidity
```

```
IQR=df.Temp.quantile(0.75)-df.Temp.quantile(0.25)
lower_bridge=df.Temp.quantile(0.25)-(IQR*1.5)
upper_bridge=df.Temp.quantile(0.75)+(IQR*1.5)
print(lower_bridge, upper_bridge)
```

```
-0.9500000000000011 36.25
```

```
df.loc[df['Temp']>=36.975,'Temp']=36.975
df.loc[df['Temp']<=2.7,'Temp']=2.7
```

```
IQR=df.WindSpeed.quantile(0.75)-df.WindSpeed.quantile(0.25)
lower_bridge=df.WindSpeed.quantile(0.25)-(IQR*1.5)
upper_bridge=df.WindSpeed.quantile(0.75)+(IQR*1.5)
print(lower_bridge, upper_bridge)
```

```
-3.25 34.75
```

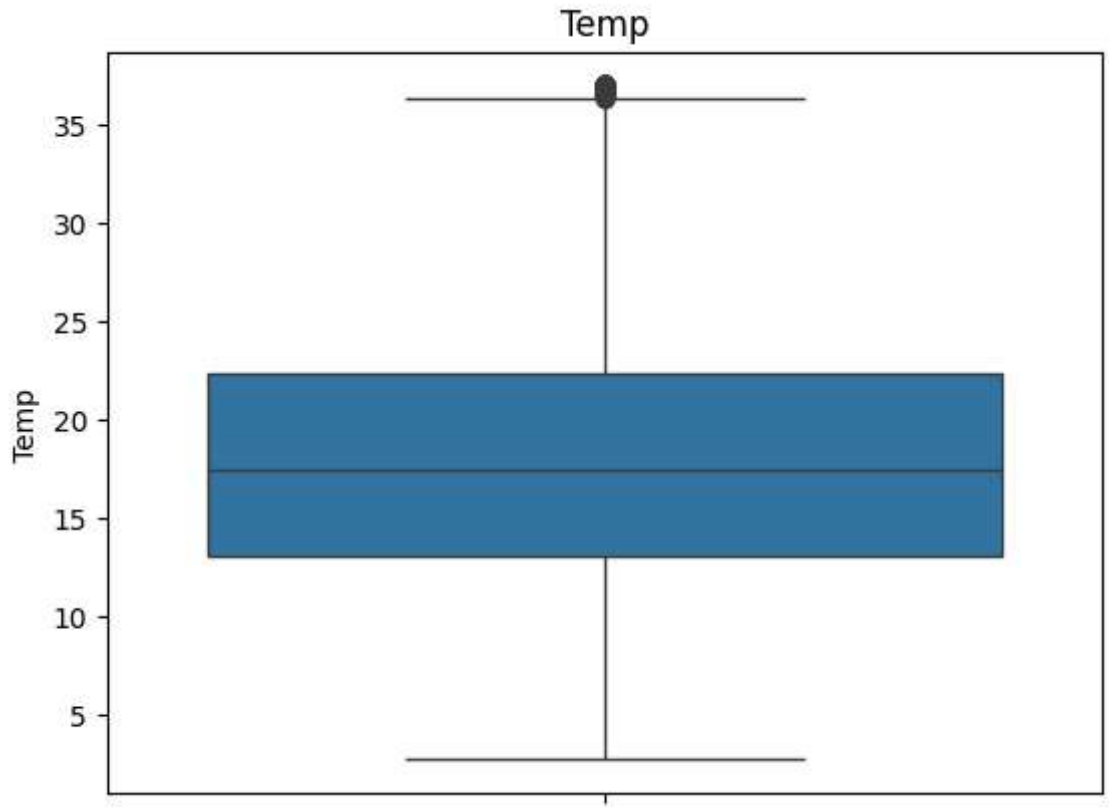
```
df.loc[df['WindSpeed']>=38.75,'WindSpeed']=38.75
df.loc[df['WindSpeed']<=-7.25,'WindSpeed']=-7.25
```

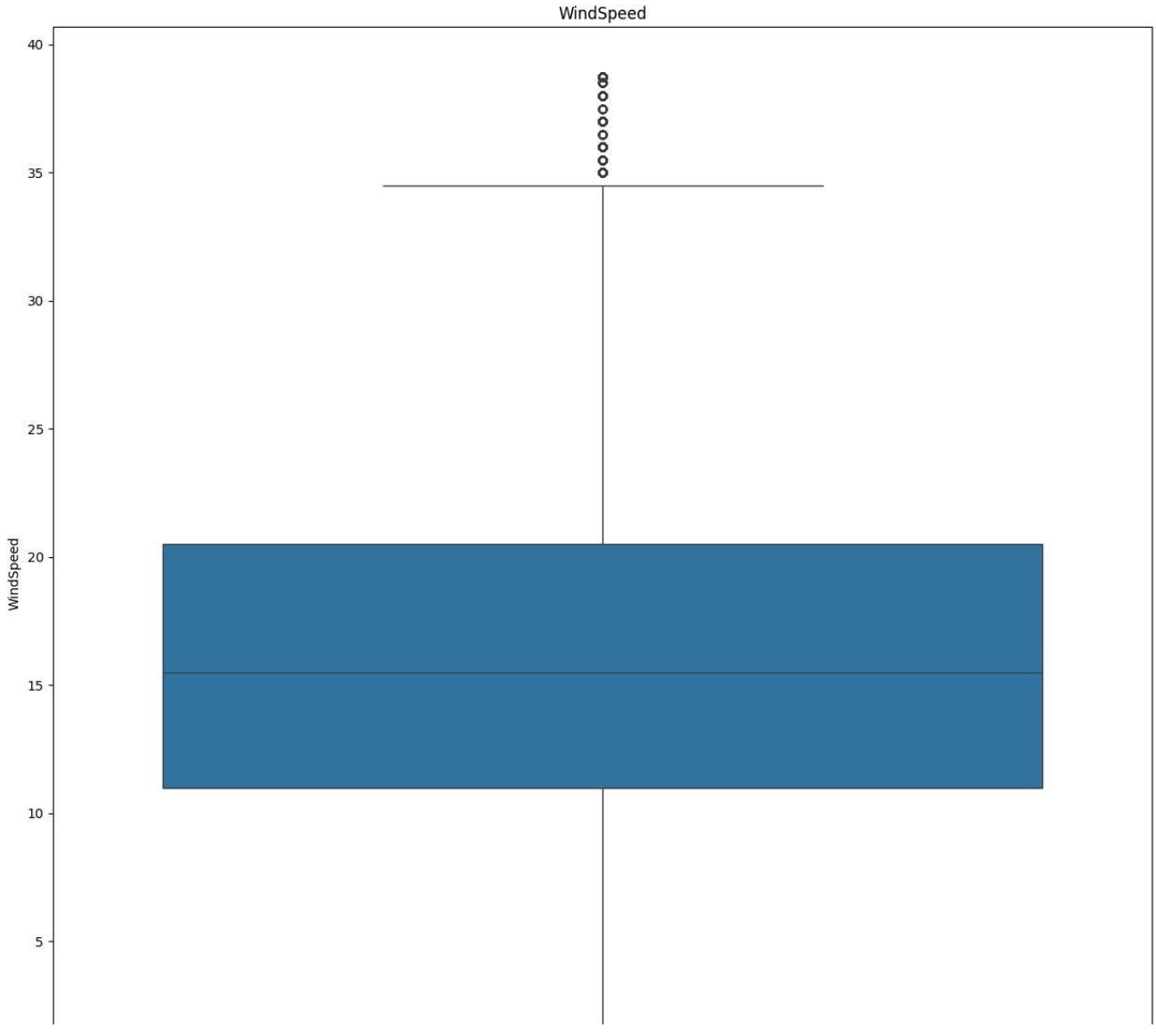
```
IQR=df.Humidity.quantile(0.75)-df.Humidity.quantile(0.25)
lower_bridge=df.Humidity.quantile(0.25)-(IQR*1.5)
upper_bridge=df.Humidity.quantile(0.75)+(IQR*1.5)
print(lower_bridge, upper_bridge)
```

```
13.0 109.0
```

```
df.loc[df['Humidity']>=122,'Humidity']=122
df.loc[df['Humidity']<=18,'Humidity']=18
```

```
for feature in continuous_feature:
    data=df.copy()
    sns.boxplot(data[feature])
    plt.title(feature)
    plt.figure(figsize=(15,15))
```



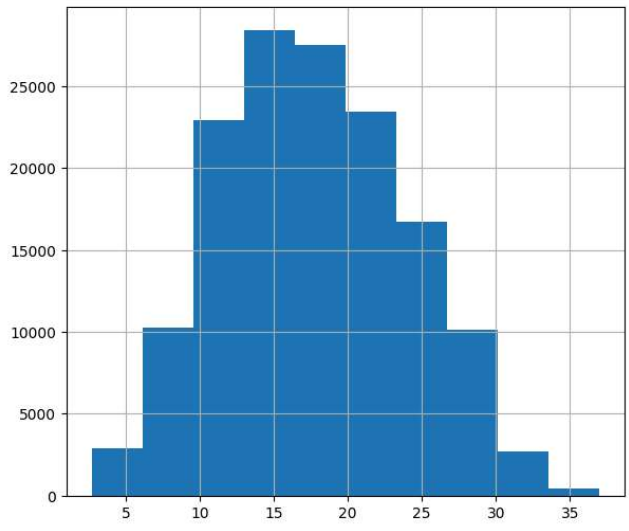


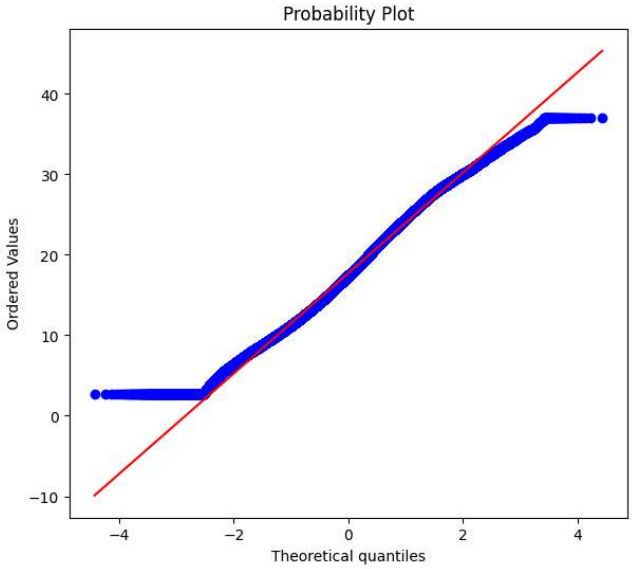

```
1 def qq_plots(df, variable):
2     plt.figure(figsize=(15,6))
```

```
    plt.subplot(1, 2, 1)
    df[variable].hist()
    plt.subplot(1, 2, 2)
    stats.probplot(df[variable], dist="norm", plot=plt)
    plt.show()
```

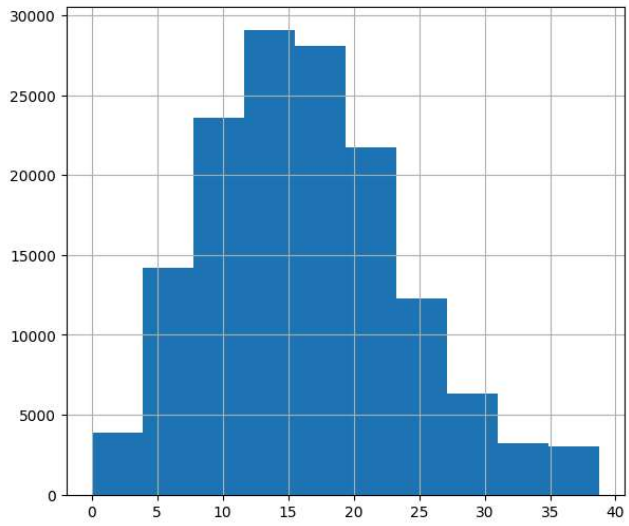
```
for feature in continuous_feature:
    print(feature)
    plt.figure(figsize=(15,6))
    plt.subplot(1, 2, 1)
    df[feature].hist()
    plt.subplot(1, 2, 2)
    stats.probplot(df[feature], dist="norm", plot=plt)
    plt.show()
```

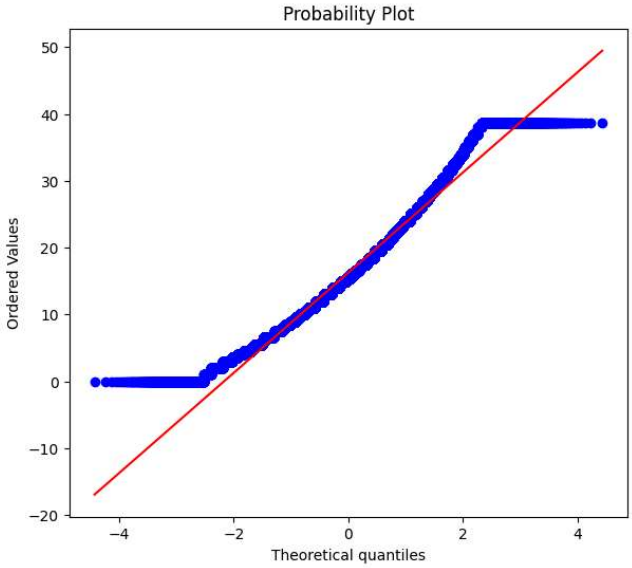
Temp



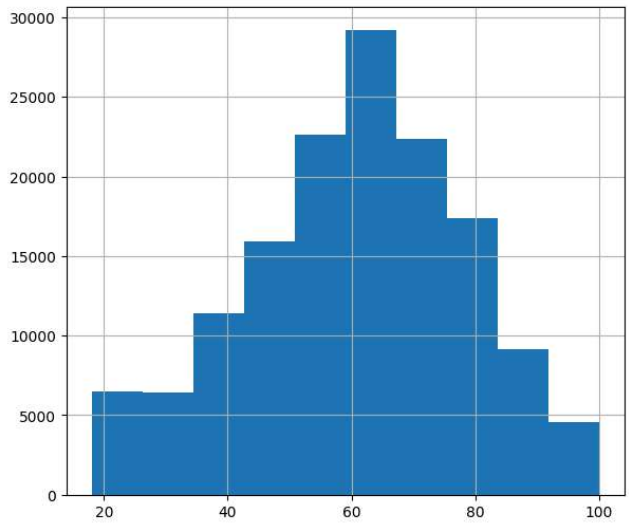


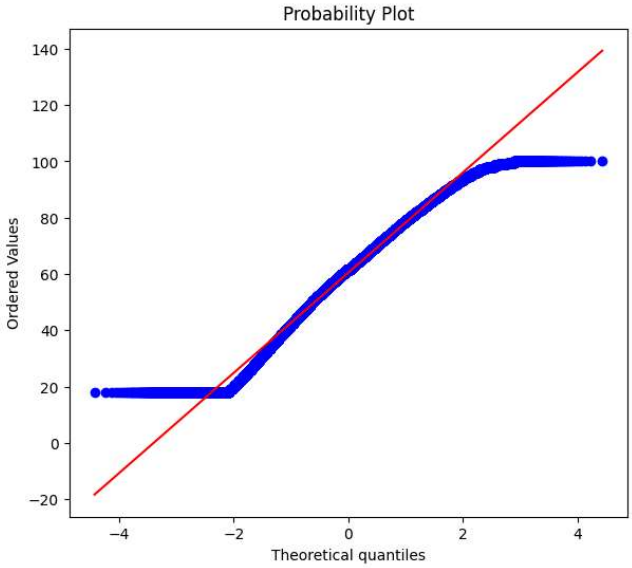
WindSpeed





Humidity






```
1 df.to_csv("preprocessed_1.csv", index=False)
```

```
1 X = df.drop(["RainTomorrow", "Date"], axis=1)
2 Y = df["RainTomorrow"]
```

```
1 # scaler = RobustScaler()
2 # X_scaled = scaler.fit_transform(X)
```

```
1 X_train, X_test, y_train, y_test = train_test_split(X,Y, test_size =0.2, stratify = Y,
```

```
1 X_train
```

	Temp	WindSpeed	Humidity	RainToday	Month	Day
43447	14.50	16.0	48.0	False	6	19
140904	27.15	24.0	51.0	False	8	26
54615	20.80	19.5	39.5	False	2	25
109793	15.90	17.5	68.5	True	5	6
74462	21.95	26.5	51.0	False	1	25
...	...	...	...	...	...	...
106567	20.85	18.5	32.5	False	11	2
52791	7.25	25.0	99.0	True	11	30
56339	5.15	27.0	76.5	False	5	21
82299	9.00	21.0	87.5	True	8	24
115175	12.50	15.5	78.0	False	7	10

116368 rows × 6 columns

```
1 y_train
```

	RainTomorrow
43447	False
140904	False
54615	False
109793	False
74462	False
...	...
106567	False
52791	True
56339	False
82299	True
115175	False

116368 rows × 1 columns

dtype: bool

```
sm=SMOTE(random_state=0)
X_train_res, y_train_res = sm.fit_resample(X_train, y_train)
print("The number of classes before fit {}".format(Counter(y_train)))
print("The number of classes after fit {}".format(Counter(y_train_res)))
```

```
The number of classes before fit Counter({False: 90866, True: 25502})
The number of classes after fit Counter({False: 90866, True: 90866})
```

```
cat = CatBoostClassifier(iterations=2000, eval_metric = "AUC")
cat.fit(X_train_res, y_train_res)
```

```
Learning rate set to 0.050311
0:      total: 108ms    remaining: 3m 36s
1:      total: 164ms    remaining: 2m 44s
2:      total: 217ms    remaining: 2m 24s
3:      total: 257ms    remaining: 2m 8s
4:      total: 291ms    remaining: 1m 56s
5:      total: 338ms    remaining: 1m 52s
6:      total: 390ms    remaining: 1m 51s
7:      total: 435ms    remaining: 1m 48s
8:      total: 471ms    remaining: 1m 44s
9:      total: 509ms    remaining: 1m 41s
10:     total: 550ms    remaining: 1m 39s
11:     total: 589ms    remaining: 1m 37s
12:     total: 624ms    remaining: 1m 35s
13:     total: 676ms    remaining: 1m 35s
14:     total: 714ms    remaining: 1m 34s
15:     total: 755ms    remaining: 1m 33s
16:     total: 795ms    remaining: 1m 32s
17:     total: 832ms    remaining: 1m 31s
18:     total: 867ms    remaining: 1m 30s
```

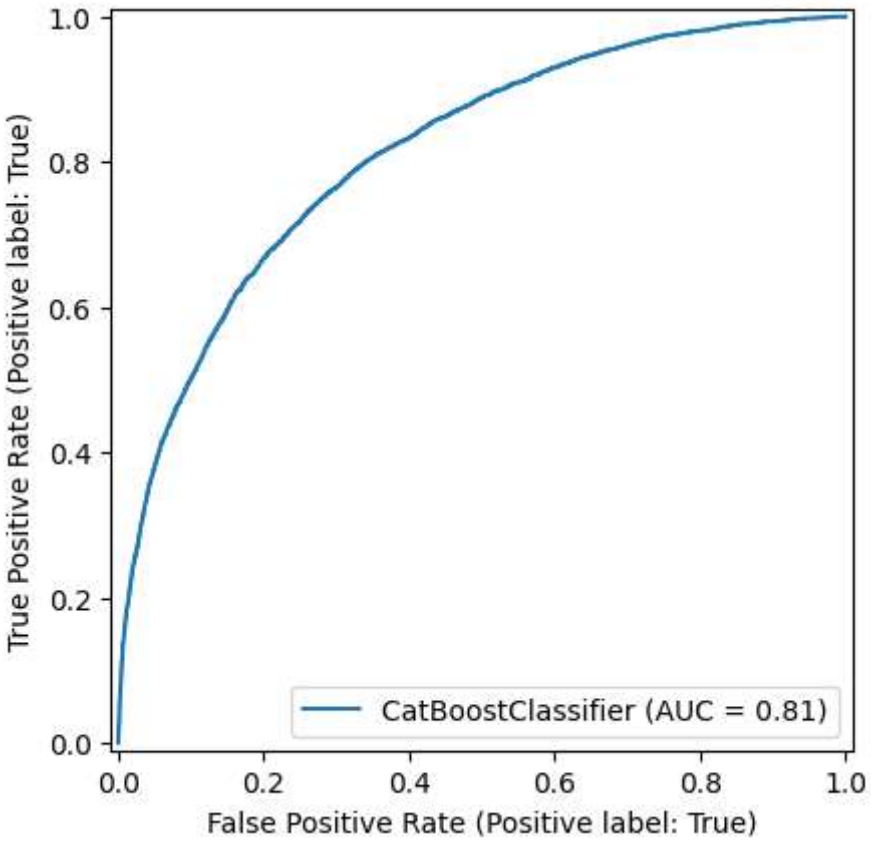
```
19:     total: 917ms    remaining: 1m 30s
20:     total: 972ms    remaining: 1m 31s
21:     total: 1.01s    remaining: 1m 30s
22:     total: 1.05s    remaining: 1m 30s
23:     total: 1.09s    remaining: 1m 29s
24:     total: 1.13s    remaining: 1m 29s
25:     total: 1.18s    remaining: 1m 29s
26:     total: 1.23s    remaining: 1m 30s
27:     total: 1.31s    remaining: 1m 32s
28:     total: 1.39s    remaining: 1m 34s
29:     total: 1.49s    remaining: 1m 37s
30:     total: 1.54s    remaining: 1m 37s
31:     total: 1.6s     remaining: 1m 38s
32:     total: 1.67s    remaining: 1m 39s
33:     total: 1.76s    remaining: 1m 41s
34:     total: 1.86s    remaining: 1m 44s
35:     total: 1.98s    remaining: 1m 47s
36:     total: 2.05s    remaining: 1m 49s
37:     total: 2.15s    remaining: 1m 50s
38:     total: 2.23s    remaining: 1m 52s
39:     total: 2.32s    remaining: 1m 53s
40:     total: 2.42s    remaining: 1m 55s
41:     total: 2.52s    remaining: 1m 57s
42:     total: 2.61s    remaining: 1m 58s
43:     total: 2.71s    remaining: 2m
44:     total: 2.81s    remaining: 2m 1s
45:     total: 2.88s    remaining: 2m 2s
46:     total: 2.94s    remaining: 2m 2s
47:     total: 3.03s    remaining: 2m 3s
48:     total: 3.12s    remaining: 2m 4s
49:     total: 3.21s    remaining: 2m 5s
50:     total: 3.29s    remaining: 2m 5s
51:     total: 3.38s    remaining: 2m 6s
52:     total: 3.47s    remaining: 2m 7s
53:     total: 3.55s    remaining: 2m 8s
54:     total: 3.63s    remaining: 2m 8s
55:     total: 3.71s    remaining: 2m 8s
56:     total: 3.81s    remaining: 2m 9s
```

```
1 y_pred = cat.predict(X_test)
2 print(confusion_matrix(y_test,y_pred))
3 print(accuracy_score(y_test,y_pred))
4 print(classification_report(y_test,y_pred))
```

```
[[21166  1551]
 [ 3608  2767]]
0.8226660250240616
              precision    recall  f1-score   support

       False       0.85      0.93      0.89     22717
        True       0.64      0.43      0.52      6375

    accuracy                           0.82     29092
   macro avg       0.75      0.68      0.70     29092
weighted avg       0.81      0.82      0.81     29092
```

```
1 metrics.RocCurveDisplay.from_estimator(cat, X_test, y_test)
2
```

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x7b7f17417e50>
```




```
1 metrics.roc_auc_score(y_test, y_pred, average=None)
```

```
0.6828821777247237
```

```
1 rf=RandomForestClassifier()
2 rf.fit(X_train_res,y_train_res)
```

```
RandomForestClassifier()
```

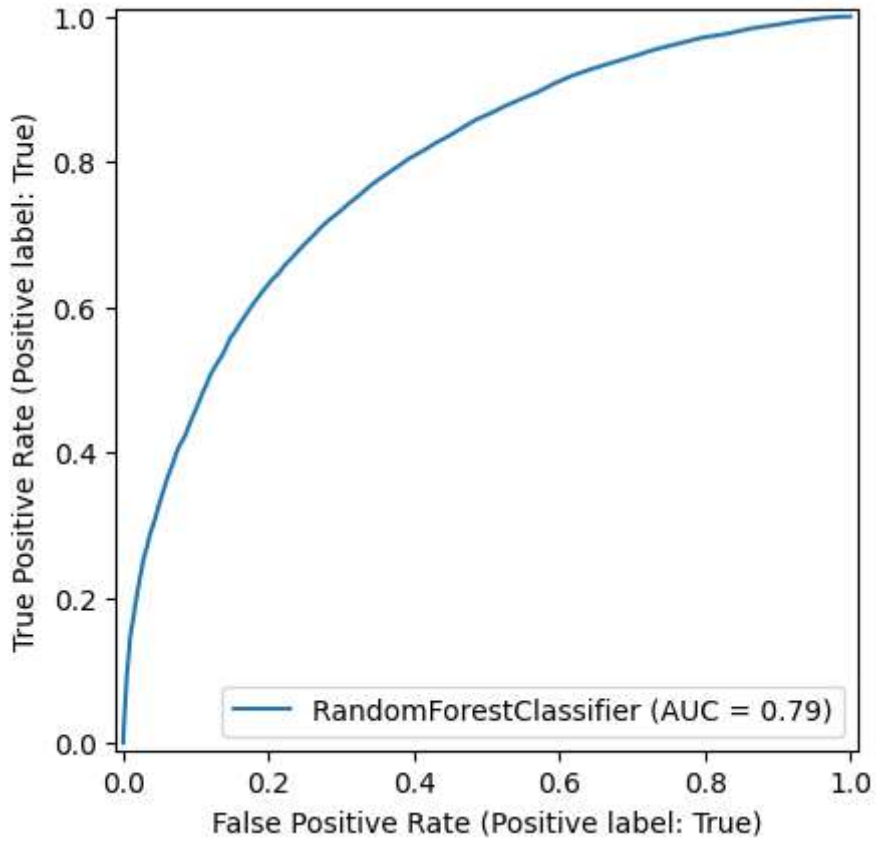
```
1 y_pred1 = rf.predict(X_test)
2 print(confusion_matrix(y_test,y_pred1))
3 print(accuracy_score(y_test,y_pred1))
4 print(classification_report(y_test,y_pred1))
```

```
[[19757  2960]
 [ 3029  3346]]
0.794135844905816
              precision    recall  f1-score   support

       False       0.87      0.87      0.87     22717
        True       0.53      0.52      0.53      6375

    accuracy                           0.79     29092
   macro avg       0.70      0.70      0.70     29092
```

```
weighted avg       0.79      0.79      0.79     29092
```

```
1 metrics.RocCurveDisplay.from_estimator(rf, X_test, y_test)
2 metrics.roc_auc_score(y_test, y_pred1, average=None)
```

```
0.6972819249987269
```




```
1 logreg = LogisticRegression()
2 logreg.fit(X_train_res, y_train_res)
```

```
LogisticRegression()
```

```
1 y_pred2 = logreg.predict(X_test)
2 print(confusion_matrix(y_test,y_pred2))
3 print(accuracy_score(y_test,y_pred2))
4 print(classification_report(y_test,y_pred2))
```

```
[[17443  5274]
 [ 2139  4236]]
0.7451876804619827
              precision    recall  f1-score   support

       False       0.89      0.77      0.82     22717
```