Breakout-3

Amna

Samreen

Ghazia

Misha

Hamid

Haris

Abdullah

In [1]:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.
mount("/content/drive", force_remount=True).
```

In [2]:

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
```

In [3]:

```
df = pd.read_csv('/content/drive/MyDrive/weatherAUS.csv')
df
```

Out[3]:

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustI
0	2008-12-01	Albury	13.4	22.9	0.6	NaN	NaN	
1	2008-12-02	Albury	7.4	25.1	0.0	NaN	NaN	WN
2	2008-12-03	Albury	12.9	25.7	0.0	NaN	NaN	WS
3	2008-12-04	Albury	9.2	28.0	0.0	NaN	NaN	
4	2008-12-05	Albury	17.5	32.3	1.0	NaN	NaN	
...	...	...	...	...	...	...	...	
145455	2017-06-21	Uluru	2.8	23.4	0.0	NaN	NaN	
145456	2017-06-22	Uluru	3.6	25.3	0.0	NaN	NaN	NN
145457	2017-06-23	Uluru	5.4	26.9	0.0	NaN	NaN	
145458	2017-06-24	Uluru	7.8	27.0	0.0	NaN	NaN	
145459	2017-06-25	Uluru	14.9	NaN	0.0	NaN	NaN	Na

145460 rows × 23 columns

In [4]:
```
df.head()
```

Out[4]:

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	W
0	2008-12-01	Albury	13.4	22.9	0.6	NaN	NaN	W	
1	2008-12-02	Albury	7.4	25.1	0.0	NaN	NaN	WNW	
2	2008-12-03	Albury	12.9	25.7	0.0	NaN	NaN	WSW	
3	2008-12-04	Albury	9.2	28.0	0.0	NaN	NaN	NE	
4	2008-12-05	Albury	17.5	32.3	1.0	NaN	NaN	W	

5 rows × 23 columns

In [5]:
```
df.shape
```

Out[5]:
```
(145460, 23)
```

In [6]:
```
df.describe()
```

Out[6]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGust
count	143975.000000	144199.000000	142199.000000	82670.000000	75625.000000	135197.
mean	12.194034	23.221348	2.360918	5.468232	7.611178	40.
std	6.398495	7.119049	8.478060	4.193704	3.785483	13.
min	-8.500000	-4.800000	0.000000	0.000000	0.000000	6.
25%	7.600000	17.900000	0.000000	2.600000	4.800000	31.
50%	12.000000	22.600000	0.000000	4.800000	8.400000	39.
75%	16.900000	28.200000	0.800000	7.400000	10.600000	48.
max	33.900000	48.100000	371.000000	145.000000	14.500000	135.

In [7]:
```
df.describe(include = ['object'])
```

Out[7]:

	Date	Location	WindGustDir	WindDir9am	WindDir3pm	RainToday	RainTomor
count	145460	145460	135134	134894	141232	142199	142
unique	3436	49	16	16	16	2	
top	2013-11-12	Canberra	W	N	SE	No	
freq	49	3436	9915	11758	10838	110319	110

In [8]:

```
df.isnull().sum()
```

Out[8]:

```
Date                 0
Location             0
MinTemp           1485
MaxTemp           1261
Rainfall          3261
Evaporation      62790
Sunshine         69835
WindGustDir      10326
WindGustSpeed    10263
WindDir9am       10566
WindDir3pm        4228
WindSpeed9am      1767
WindSpeed3pm      3062
Humidity9am       2654
Humidity3pm       4507
Pressure9am      15065
Pressure3pm      15028
Cloud9am         55888
Cloud3pm         59358
Temp9am           1767
Temp3pm           3609
RainToday         3261
RainTomorrow      3267
dtype: int64
```

In [9]:

```
df.duplicated().sum()
```

Out[9]:

```
0
```

In [10]:

```
missing_percentages = df.isnull().mean()*100
missing_percentages
```

```
Out[10]: Date              0.000000
         Location          0.000000
         MinTemp           1.020899
         MaxTemp           0.866905
         Rainfall          2.241853
         Evaporation      43.166506
         Sunshine         48.009762
         WindGustDir       7.098859
         WindGustSpeed     7.055548
         WindDir9am        7.263853
         WindDir3pm        2.906641
         WindSpeed9am      1.214767
         WindSpeed3pm      2.105046
         Humidity9am       1.824557
         Humidity3pm       3.098446
         Pressure9am      10.356799
         Pressure3pm      10.331363
         Cloud9am         38.421559
         Cloud3pm         40.807095
         Temp9am           1.214767
         Temp3pm           2.481094
         RainToday         2.241853
         RainTomorrow      2.245978
         dtype: float64
```

```
In [11]: df.columns
```

```
Out[11]: Index(['Date', 'Location', 'MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation',
                'Sunshine', 'WindGustDir', 'WindGustSpeed', 'WindDir9am', 'WindDir3pm',
                'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm',
                'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am',
                'Temp3pm', 'RainToday', 'RainTomorrow'],
               dtype='object')
```

Imputing Missing values in RainTomorrow column using mode.

```
In [12]: df['RainTomorrow'].value_counts()
```

```
Out[12]: RainTomorrow
         No     110316
         Yes     31877
         Name: count, dtype: int64
```

```
In [13]: df['RainTomorrow'].unique()
```

```
Out[13]: array(['No', 'Yes', nan], dtype=object)
```

```
In [14]: mode_value = df['RainTomorrow'].mode()[0]
         print(mode_value)
```

```
No
```

```
In [15]: df['RainTomorrow'].fillna(value=mode_value, inplace=True)
```

```
In [16]: df[['RainTomorrow']]
```

Out[16]:

	RainTomorrow
0	No
1	No
2	No
3	No
4	No
...	...
145455	No
145456	No
145457	No
145458	No
145459	No

145460 rows × 1 columns

In [17]:
```
df.isnull().sum()
```

Out[17]:
```
Date                 0
Location             0
MinTemp           1485
MaxTemp           1261
Rainfall          3261
Evaporation      62790
Sunshine         69835
WindGustDir      10326
WindGustSpeed    10263
WindDir9am       10566
WindDir3pm        4228
WindSpeed9am      1767
WindSpeed3pm      3062
Humidity9am       2654
Humidity3pm       4507
Pressure9am      15065
Pressure3pm      15028
Cloud9am         55888
Cloud3pm         59358
Temp9am           1767
Temp3pm           3609
RainToday         3261
RainTomorrow         0
dtype: int64
```

Imputing missing values in Rain Fall column using Median

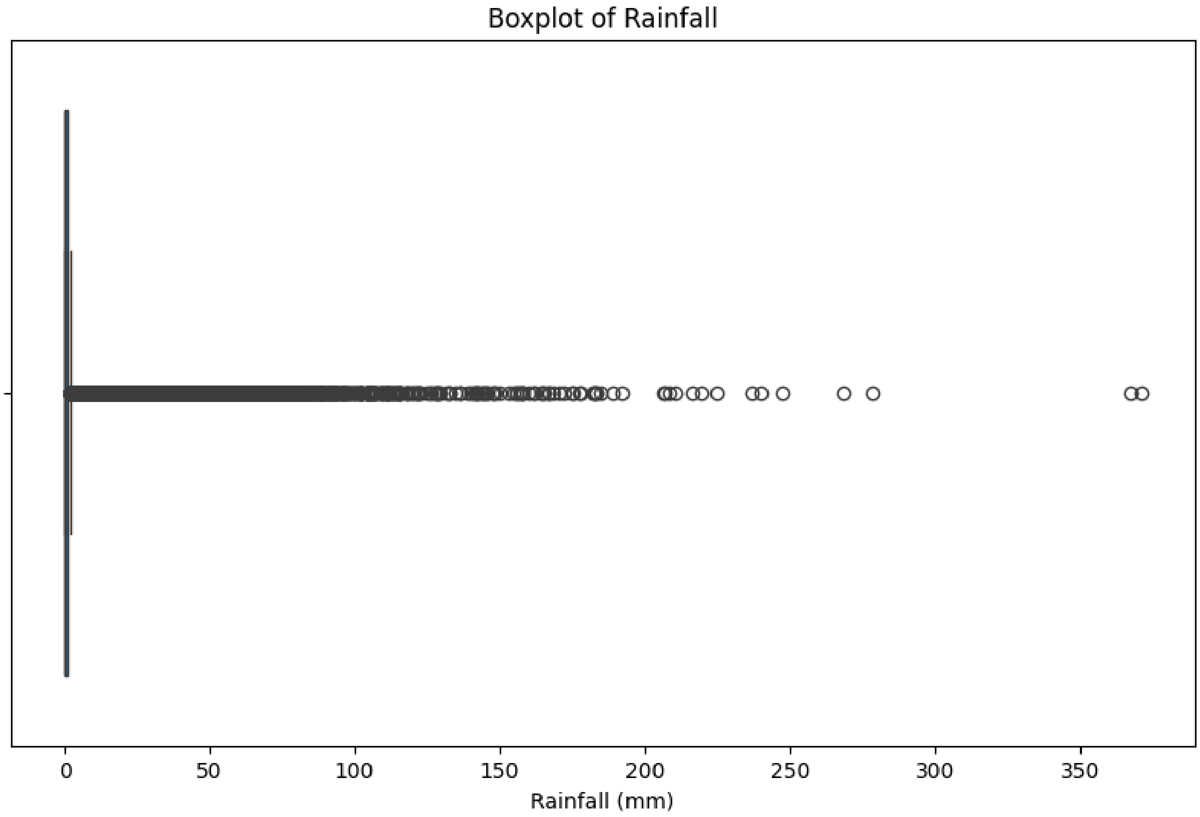
In [18]:
```
df['Rainfall'].describe()
```

```
Out[18]: count    142199.000000
         mean          2.360918
         std           8.478060
         min           0.000000
         25%           0.000000
         50%           0.000000
         75%           0.800000
         max         371.000000
         Name: Rainfall, dtype: float64
```

```
In [19]: df['Rainfall'].isnull().sum()
```

```
Out[19]: 3261
```

```
In [20]: plt.figure(figsize=(10, 6))
         sns.boxplot(x=df['Rainfall'])
         plt.title('Boxplot of Rainfall')
         plt.xlabel('Rainfall (mm)')
         plt.show()
```



```
In [21]: Q1 = df['Rainfall'].quantile(0.25)
         Q3 = df['Rainfall'].quantile(0.75)
         IQR = Q3 - Q1
         lower_bound = Q1 - 1.5 * IQR
         upper_bound = Q3 + 1.5 * IQR

         outliers = df[(df['Rainfall'] < lower_bound) | (df['Rainfall'] > upper_bound)]
         percentage_outliers = len(outliers) / len(df) * 100
         print(f"Percentage of outliers in 'Rainfall': {percentage_outliers:.2f}%")
```

```
Percentage of outliers in 'Rainfall': 17.58%
```

In [22]:
```
median_value = df['Rainfall'].median()
```

In [23]:
```
print(median_value)
```

```
0.0
```

In [24]:
```
df['Rainfall'].fillna(value = median_value, inplace = True)
```

In [25]:
```
df['Rainfall'].isnull().sum()
```

Out[25]:
```
0
```

In [26]:
```
df.isnull().sum()
```

Out[26]:
```
Date                 0
Location             0
MinTemp           1485
MaxTemp           1261
Rainfall             0
Evaporation      62790
Sunshine         69835
WindGustDir      10326
WindGustSpeed    10263
WindDir9am       10566
WindDir3pm        4228
WindSpeed9am      1767
WindSpeed3pm      3062
Humidity9am       2654
Humidity3pm       4507
Pressure9am      15065
Pressure3pm      15028
Cloud9am         55888
Cloud3pm         59358
Temp9am           1767
Temp3pm           3609
RainToday         3261
RainTomorrow         0
dtype: int64
```

Imputing missing values in Humidity9am column using Mean

In [27]:
```
df[['Humidity9am']]
```

Out[27]:

	Humidity9am
0	71.0
1	44.0
2	38.0
3	45.0
4	82.0
...	...
145455	51.0
145456	56.0
145457	53.0
145458	51.0
145459	62.0

145460 rows × 1 columns

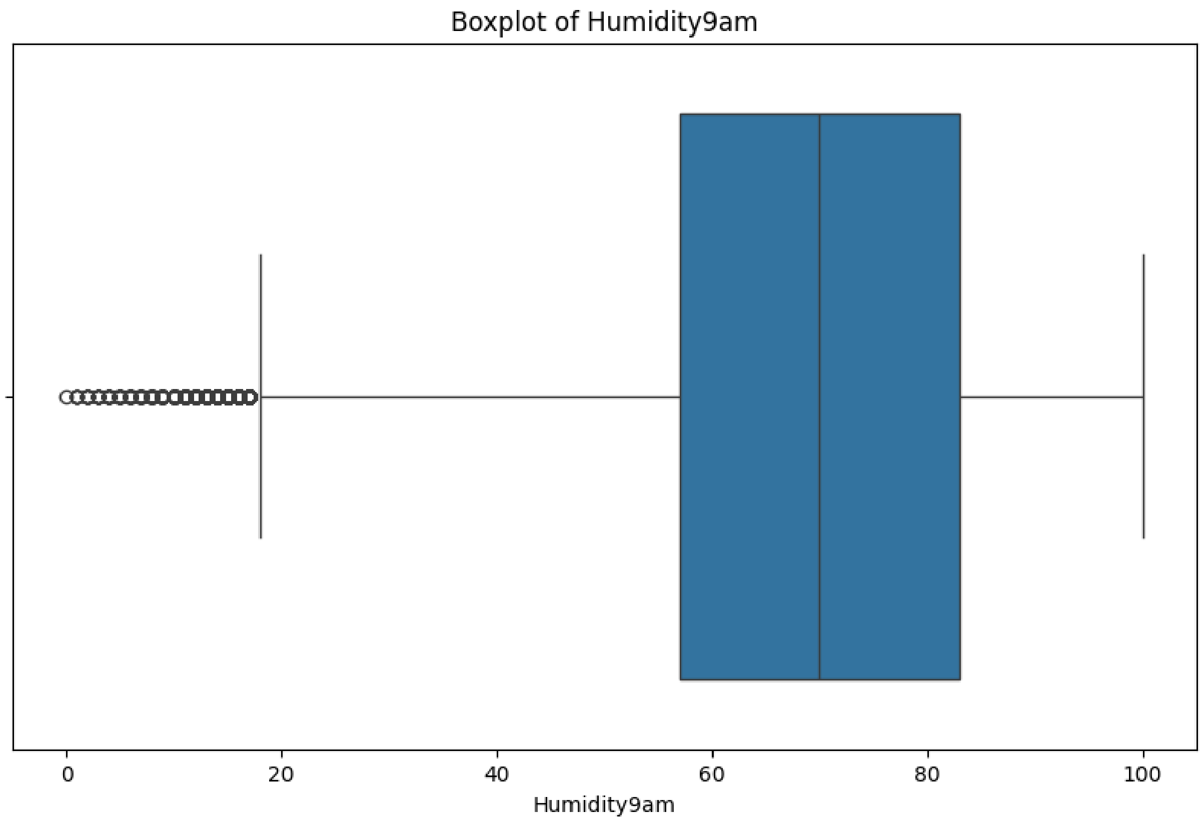
In [28]:
```
df['Humidity9am'].describe()
```

Out[28]:
```
count    142806.000000
mean         68.880831
std          19.029164
min           0.000000
25%          57.000000
50%          70.000000
75%          83.000000
max         100.000000
Name: Humidity9am, dtype: float64
```

In [29]:
```
df['Humidity9am'].isnull().sum()
```

Out[29]:
```
2654
```

In [30]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(x=df['Humidity9am'])
plt.title('Boxplot of Humidity9am')
plt.xlabel('Humidity9am')
plt.show()
```



In [31]:
```
Q1 = df['Humidity9am'].quantile(0.25)
Q3 = df['Humidity9am'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

outliers = df[(df['Humidity9am'] < lower_bound) | (df['Humidity9am'] > upper_bound)
percentage_outliers = len(outliers) / len(df) * 100
print(f"Percentage of outliers in 'Humidity9am': {percentage_outliers:.2f}%")
```

```
Percentage of outliers in 'Humidity9am': 0.98%
```

In [32]:
```
mean_Humidity9am = df['Humidity9am'].mean()
```

In [33]:
```
print(mean_Humidity9am)
```

```
68.88083133761887
```

In [34]:
```
df['Humidity9am'].fillna( value = mean_Humidity9am, inplace = True)
```

In [35]:
```
df.isnull().sum()
```

```
Out[35]: Date                 0
         Location             0
         MinTemp           1485
         MaxTemp           1261
         Rainfall             0
         Evaporation      62790
         Sunshine         69835
         WindGustDir      10326
         WindGustSpeed    10263
         WindDir9am       10566
         WindDir3pm        4228
         WindSpeed9am      1767
         WindSpeed3pm      3062
         Humidity9am          0
         Humidity3pm       4507
         Pressure9am      15065
         Pressure3pm      15028
         Cloud9am         55888
         Cloud3pm         59358
         Temp9am           1767
         Temp3pm           3609
         RainToday         3261
         RainTomorrow         0
         dtype: int64
```

Imputing missing values WindGustSpeed column using median

```
In [36]: df[['WindGustSpeed']]
```

Out[36]:

	WindGustSpeed
0	44.0
1	44.0
2	46.0
3	24.0
4	41.0
...	...
145455	31.0
145456	22.0
145457	37.0
145458	28.0
145459	NaN

145460 rows × 1 columns

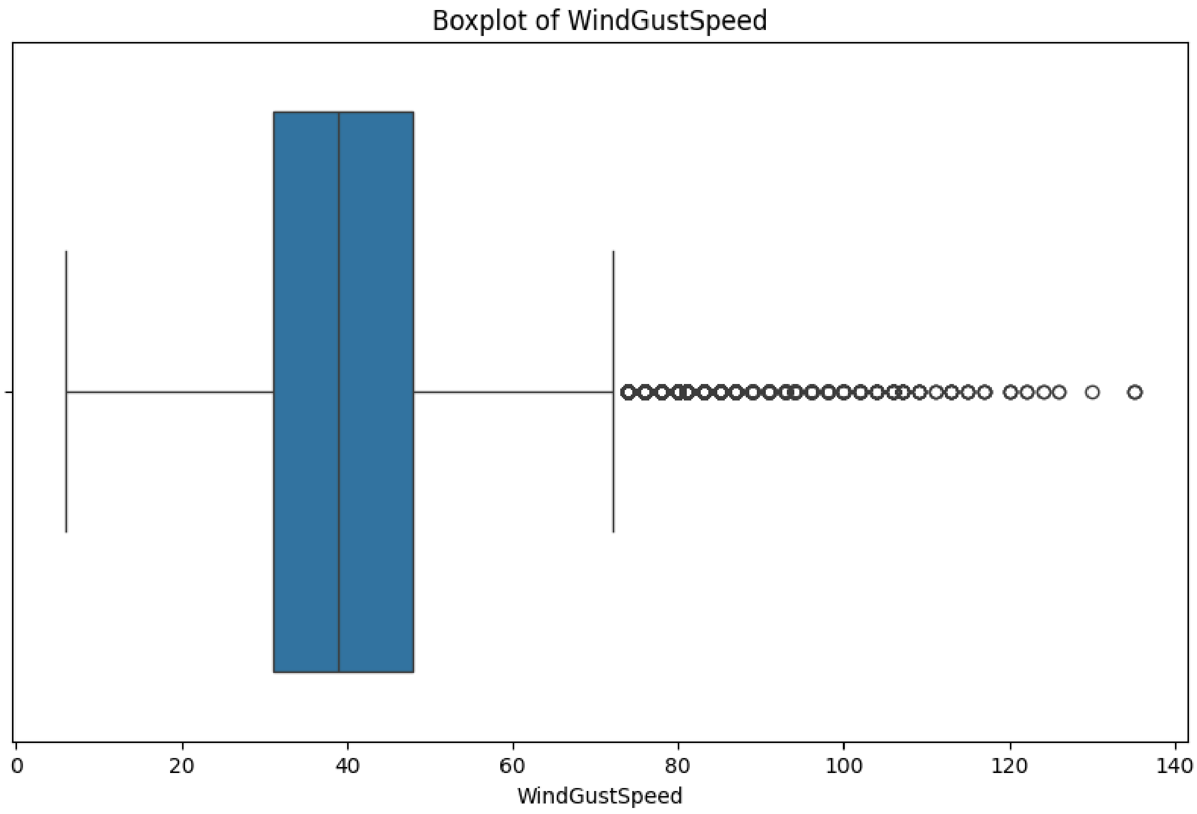
In [37]:
```
df['WindGustSpeed'].describe()
```

Out[37]:
```
count    135197.000000
mean         40.035230
std          13.607062
min           6.000000
25%          31.000000
50%          39.000000
75%          48.000000
max         135.000000
Name: WindGustSpeed, dtype: float64
```

In [38]:
```
df['WindGustSpeed'].isnull().sum()
```

Out[38]:
```
10263
```

In [39]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(x=df['WindGustSpeed'])
plt.title('Boxplot of WindGustSpeed')
plt.xlabel('WindGustSpeed')
plt.show()
```



In [40]:
```
Q1 = df['WindGustSpeed'].quantile(0.25)
Q3 = df['WindGustSpeed'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

outliers = df[(df['WindGustSpeed'] < lower_bound) | (df['WindGustSpeed'] > upper_bo
percentage_outliers = len(outliers) / len(df) * 100
print(f"Percentage of outliers in 'WindGustSpeed': {percentage_outliers:.2f}%")
```

```
Percentage of outliers in 'WindGustSpeed': 2.13%
```

In [41]:

```
median_wgs = df['WindGustSpeed'].median()
df['WindGustSpeed'].fillna(value = median_wgs, inplace = True)
```

In [42]:

```
df.isnull().sum()
```

Out[42]:

```
Date                 0
Location             0
MinTemp           1485
MaxTemp           1261
Rainfall             0
Evaporation      62790
Sunshine         69835
WindGustDir      10326
WindGustSpeed        0
WindDir9am       10566
WindDir3pm        4228
WindSpeed9am      1767
WindSpeed3pm      3062
Humidity9am          0
Humidity3pm       4507
Pressure9am      15065
Pressure3pm      15028
Cloud9am         55888
Cloud3pm         59358
Temp9am           1767
Temp3pm           3609
RainToday         3261
RainTomorrow         0
dtype: int64
```

Imputing missing values in Pressure9am column using median

In [43]:

```
df[['Pressure9am']]
```

Out[43]:

	Pressure9am
0	1007.7
1	1010.6
2	1007.6
3	1017.6
4	1010.8
...	...
145455	1024.6
145456	1023.5
145457	1021.0
145458	1019.4
145459	1020.2

145460 rows × 1 columns

In [44]:

```
df['Pressure9am'].describe()
```

Out[44]:

```
count    130395.00000
mean       1017.64994
std           7.10653
min         980.50000
25%        1012.90000
50%        1017.60000
75%        1022.40000
max        1041.00000
Name: Pressure9am, dtype: float64
```

In [45]:

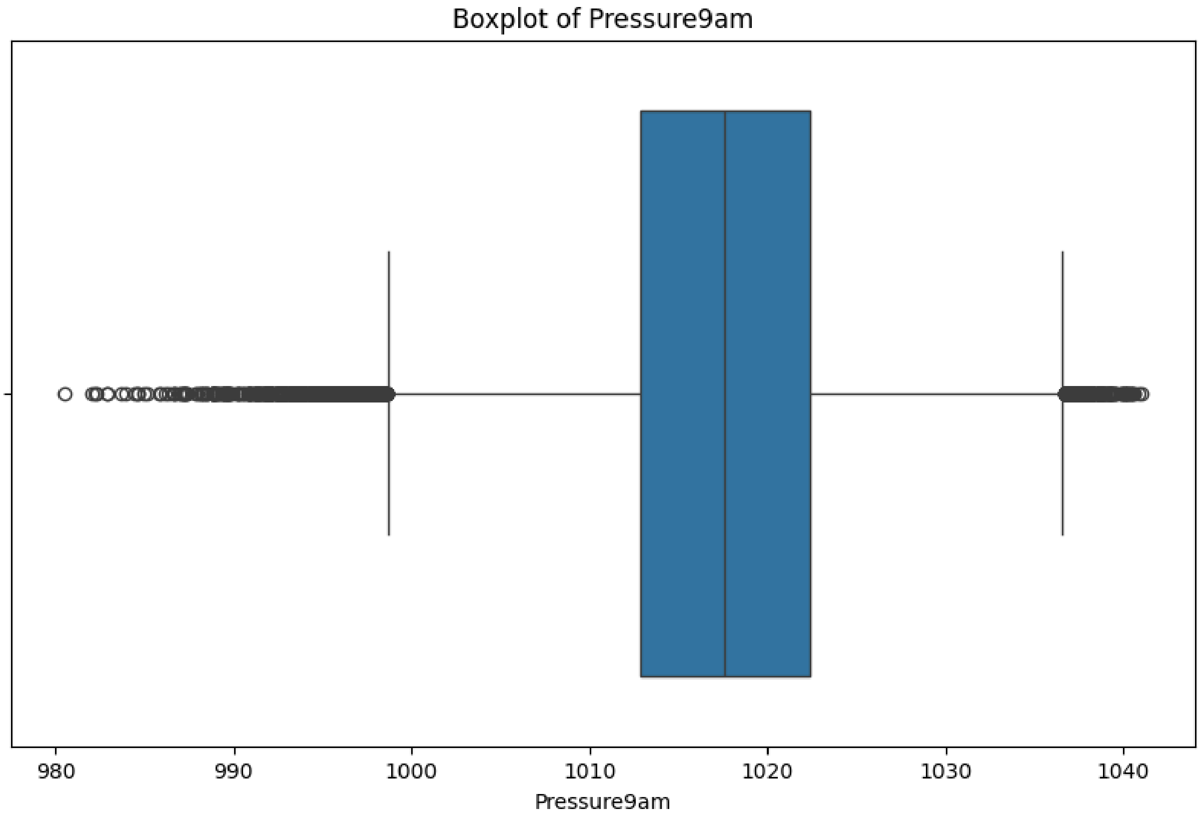
```
df['Pressure9am'].isnull().sum()
```

Out[45]:

```
15065
```

In [46]:

```
plt.figure(figsize=(10, 6))
sns.boxplot(x=df['Pressure9am'])
plt.title('Boxplot of Pressure9am')
plt.xlabel('Pressure9am')
plt.show()
```



In [47]:

```
def outlier_per (df, column_name):
    """
    Calculate and return the percentage of outliers in a specified column of a Data

    Parameters:
        df (pd.DataFrame): The DataFrame containing the data.
        column_name (str): The name of the column to analyze for outliers.

    Returns:
        float: The percentage of values in the column that are considered outliers.
    """
    # Calculate the interquartile range (IQR)
    Q1 = df[column_name].quantile(0.25)
    Q3 = df[column_name].quantile(0.75)
    IQR = Q3 - Q1

    # Determine the bounds for outliers
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Identify outliers
    outliers = df[(df[column_name] < lower_bound) | (df[column_name] > upper_bound)

    # Calculate the percentage of outliers
    percentage_outliers = len(outliers) / len(df) * 100

    return percentage_outliers
outlier_per(df,'Pressure9am')
```

Out[47]:

```
0.818781795682662
```

In [48]:

```
Q1 = df['Pressure9am'].quantile(0.25)
Q3 = df['Pressure9am'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

outliers = df[(df['Pressure9am'] < lower_bound) | (df['Pressure9am'] > upper_bound)
percentage_outliers = len(outliers) / len(df) * 100
print(f"Percentage of outliers in 'Pressure9am': {percentage_outliers:.2f}%")
```

```
Percentage of outliers in 'Pressure9am': 0.82%
```

In [49]:

```
median_pressure = df['Pressure9am'].median()
print(median_pressure)
```

```
1017.6
```

In [50]:

```
df['Pressure9am'].fillna(value = median_pressure, inplace = True)
```

In [51]:

```
df.isnull().sum()
```

Out[51]:

```
Date                 0
Location             0
MinTemp           1485
MaxTemp           1261
Rainfall             0
Evaporation      62790
Sunshine         69835
WindGustDir      10326
WindGustSpeed        0
WindDir9am       10566
WindDir3pm        4228
WindSpeed9am      1767
WindSpeed3pm      3062
Humidity9am          0
Humidity3pm       4507
Pressure9am          0
Pressure3pm      15028
Cloud9am         55888
Cloud3pm         59358
Temp9am           1767
Temp3pm           3609
RainToday         3261
RainTomorrow         0
dtype: int64
```

Covert RainToday and RainTomorrow columns into numerical column using label encoding method these were catergoical columns

In [52]:

```
df.describe(include = ['object'])
```

Out[52]:

	Date	Location	WindGustDir	WindDir9am	WindDir3pm	RainToday	RainTomor
count	145460	145460	135134	134894	141232	142199	145
unique	3436	49	16	16	16	2	
top	2013-11-12	Canberra	W	N	SE	No	
freq	49	3436	9915	11758	10838	110319	113

```
In [53]: from sklearn.preprocessing import LabelEncoder
```

```
In [54]: le = LabelEncoder()
```

```
In [55]: df['RainToday']=le.fit_transform(df['RainToday'])
```

```
In [56]: df['RainTomorrow']=le.fit_transform(df['RainTomorrow'])
```

```
In [57]: df[['RainToday','RainTomorrow']]
```

Out[57]:

	RainToday	RainTomorrow
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
...	...	...
145455	0	0
145456	0	0
145457	0	0
145458	0	0
145459	0	0

145460 rows × 2 columns

Imputing Missing Values in Cloud9am USING mean

```
In [58]: df[['Cloud9am']]
```

Out[58]:

	Cloud9am
0	8.0
1	NaN
2	NaN
3	NaN
4	7.0
...	...
145455	NaN
145456	NaN
145457	NaN
145458	3.0
145459	8.0

145460 rows × 1 columns

In [59]:

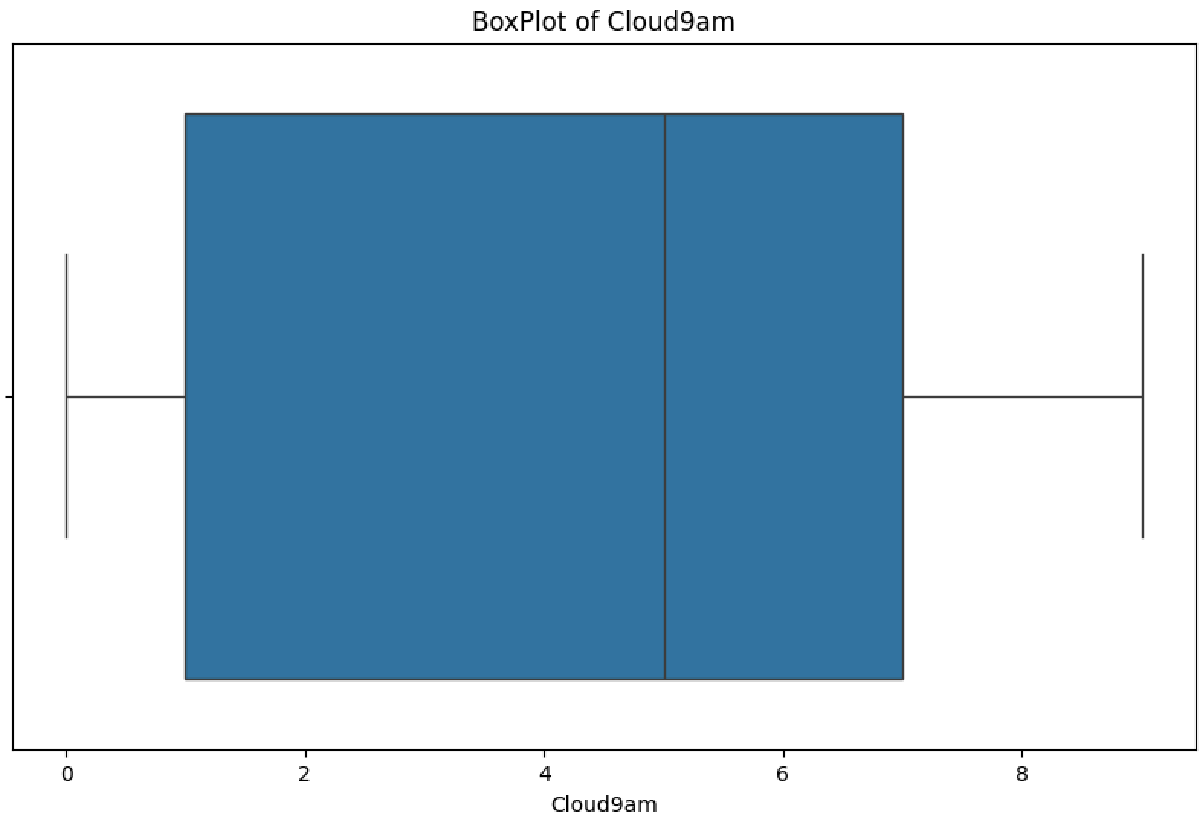
```
df['Cloud9am'].isnull().sum()
```

Out[59]:

```
55888
```

In [60]:

```
plt.figure(figsize=(10,6))
sns.boxplot(x=df['Cloud9am'])
plt.title('BoxPlot of Cloud9am')
plt.xlabel('Cloud9am')
plt.show()
```



In [61]:
```
outlier_per(df,'Cloud9am')
```

Out[61]:
```
0.0
```

In [62]:
```
mean_cloud9am = df['Cloud9am'].mean()
print(mean_cloud9am)
```

```
4.4474612602152455
```

In [63]:
```
df['Cloud9am'].fillna(value = mean_cloud9am, inplace = True)
```

In [64]:
```
df.isnull().sum()
```

Out[64]:
```
Date                 0
Location             0
MinTemp           1485
MaxTemp           1261
Rainfall             0
Evaporation      62790
Sunshine         69835
WindGustDir      10326
WindGustSpeed        0
WindDir9am       10566
WindDir3pm        4228
WindSpeed9am      1767
WindSpeed3pm      3062
Humidity9am          0
Humidity3pm       4507
Pressure9am          0
Pressure3pm      15028
Cloud9am             0
Cloud3pm         59358
Temp9am           1767
Temp3pm           3609
RainToday            0
RainTomorrow         0
dtype: int64
```

Imputing missing values in Temp9am column using Median

In [65]:
```
df[['Temp9am']]
```

Out[65]:

	Temp9am
0	16.9
1	17.2
2	21.0
3	18.1
4	17.8
...	...
145455	10.1
145456	10.9
145457	12.5
145458	15.1
145459	15.0

145460 rows × 1 columns

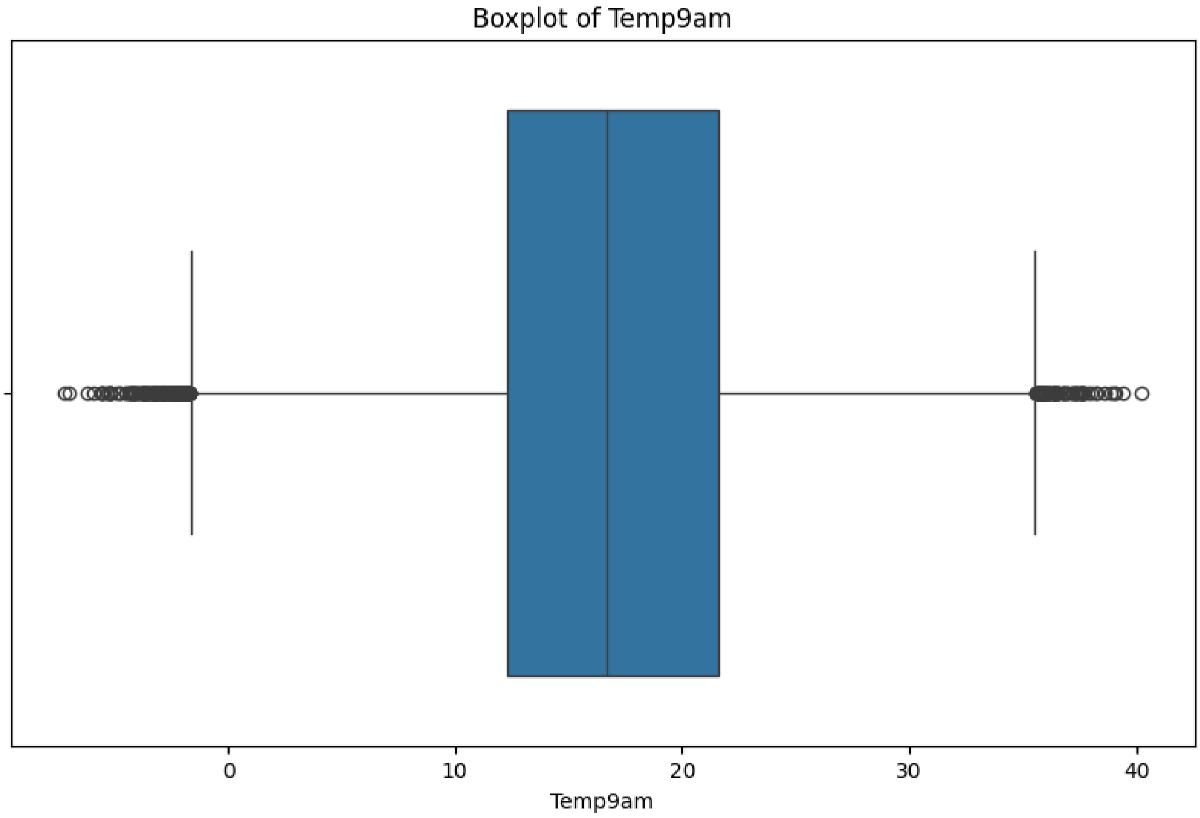
In [66]:
```
df['Temp9am'].isnull().sum()
```

Out[66]:

```
1767
```

In [67]:

```
plt.figure(figsize=(10, 6))
sns.boxplot(x=df['Temp9am'])
plt.title('Boxplot of Temp9am')
plt.xlabel('Temp9am')
plt.show()
```



In [68]:

```
outlier_per(df,'Temp9am')
```

Out[68]:

```
0.18011824556579129
```

In [69]:

```
median_temp = df['Temp9am'].median()
print(median_temp)
```

```
16.7
```

In [70]:

```
df['Temp9am'].fillna(value = median_temp , inplace = True)
```

In [71]:

```
df.isnull().sum()
```

Out[71]:

```
Date                 0
Location             0
MinTemp           1485
MaxTemp           1261
Rainfall             0
Evaporation      62790
Sunshine         69835
WindGustDir      10326
WindGustSpeed        0
WindDir9am       10566
WindDir3pm        4228
WindSpeed9am      1767
WindSpeed3pm      3062
Humidity9am          0
Humidity3pm       4507
Pressure9am          0
Pressure3pm      15028
Cloud9am             0
Cloud3pm         59358
Temp9am              0
Temp3pm           3609
RainToday            0
RainTomorrow         0
dtype: int64
```

Imputing missing values in MinTemp column using Median

In [72]:

```
df[['MinTemp']]
```

Out[72]:

	MinTemp
0	13.4
1	7.4
2	12.9
3	9.2
4	17.5
...	...
145455	2.8
145456	3.6
145457	5.4
145458	7.8
145459	14.9

145460 rows × 1 columns

In [73]:

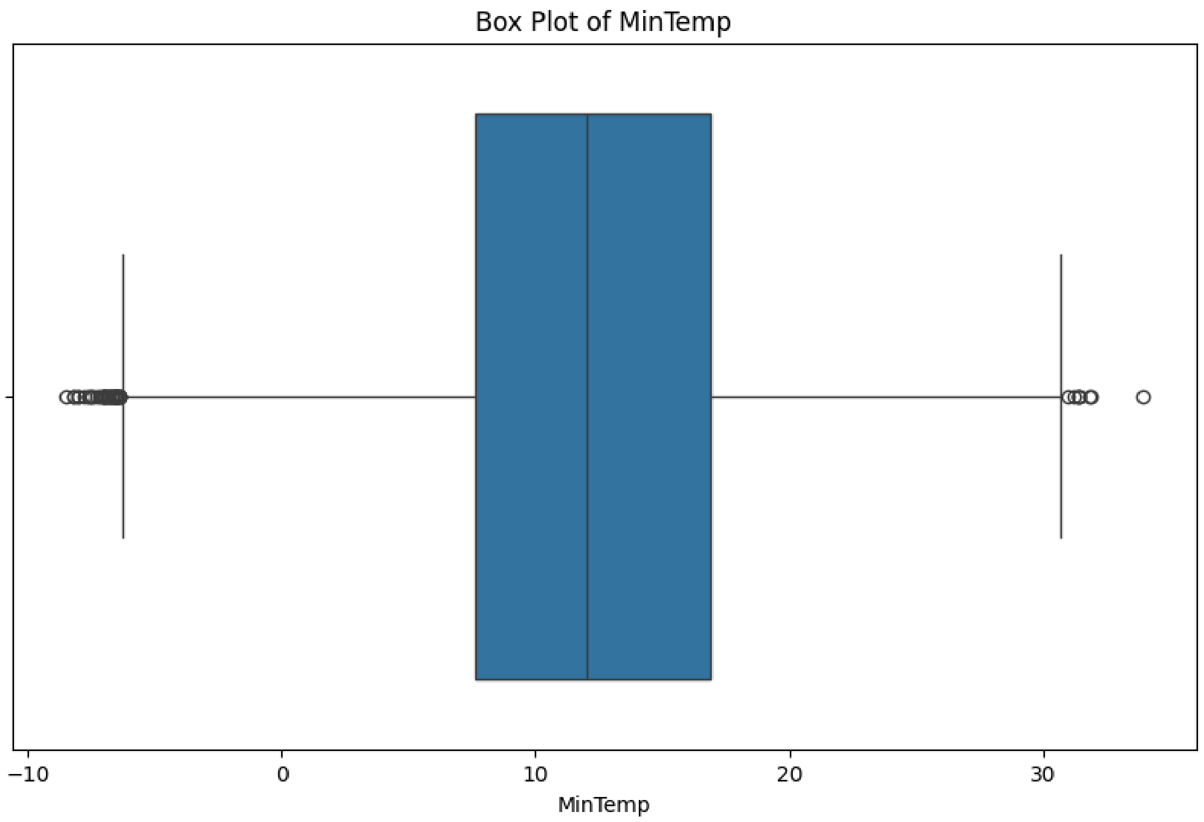
```
df['MinTemp'].isnull().sum()
```

Out[73]:

```
1485
```

In [74]:

```
plt.figure(figsize=(10,6))
sns.boxplot(x=df['MinTemp'])
plt.title('Box Plot of MinTemp')
plt.xlabel('MinTemp')
plt.show()
```



In [75]:
```
outlier_per(df,'MinTemp')
```

Out[75]:
```
0.037123607864705074
```

In [76]:
```
median_mintemp = df['MinTemp'].median()
print(median_mintemp)
```

```
12.0
```

In [77]:
```
df['MinTemp'].fillna(value=median_mintemp, inplace= True)
```

In [78]:
```
df.isnull().sum()
```

Out[78]:

```
Date                 0
Location             0
MinTemp              0
MaxTemp           1261
Rainfall             0
Evaporation      62790
Sunshine         69835
WindGustDir      10326
WindGustSpeed        0
WindDir9am       10566
WindDir3pm        4228
WindSpeed9am      1767
WindSpeed3pm      3062
Humidity9am          0
Humidity3pm       4507
Pressure9am          0
Pressure3pm      15028
Cloud9am             0
Cloud3pm         59358
Temp9am              0
Temp3pm           3609
RainToday            0
RainTomorrow         0
dtype: int64
```

Imputing values for Maxtemp using Median

In [79]:

```
df['MaxTemp']
```

Out[79]:

```
0         22.9
1         25.1
2         25.7
3         28.0
4         32.3
          ... 
145455    23.4
145456    25.3
145457    26.9
145458    27.0
145459     NaN
Name: MaxTemp, Length: 145460, dtype: float64
```

In [80]:

```
df['MaxTemp'].dtypes
```

Out[80]:

```
dtype('float64')
```

In [81]:

```
df['MaxTemp'].describe()
```

Out[81]:

```
count    144199.000000
mean         23.221348
std           7.119049
min          -4.800000
25%          17.900000
50%          22.600000
75%          28.200000
max          48.100000
Name: MaxTemp, dtype: float64
```

In [82]:

```
df['MaxTemp'].isnull().sum()
```

Out[82]:

```
1261
```

In [83]:

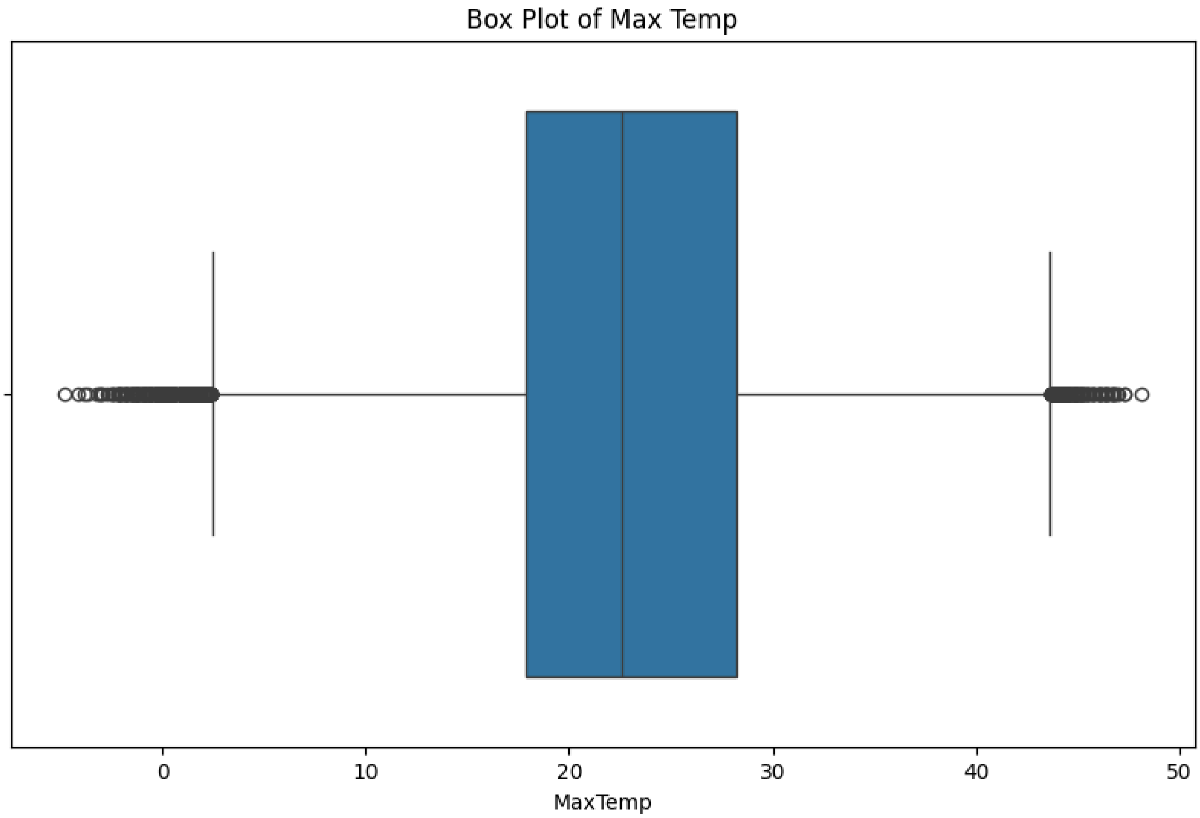
```
outlier_per(df,'MaxTemp')
```

Out[83]:

```
0.3361748934414959
```

In [84]:

```
plt.figure(figsize=(10,6))
sns.boxplot(x=df['MaxTemp'])
plt.title('Box Plot of Max Temp')
plt.xlabel('MaxTemp')
plt.show()
```



In [85]:

```
median_maxtemp = df['MaxTemp'].median()
print('median_maxtemp')
df['MaxTemp'].fillna(value = median_maxtemp, inplace = True)
```

```
median_maxtemp
```

```
In [86]: df.isnull().sum()
```

```
Out[86]: Date                 0
         Location             0
         MinTemp              0
         MaxTemp              0
         Rainfall             0
         Evaporation      62790
         Sunshine         69835
         WindGustDir      10326
         WindGustSpeed        0
         WindDir9am       10566
         WindDir3pm        4228
         WindSpeed9am      1767
         WindSpeed3pm      3062
         Humidity9am          0
         Humidity3pm       4507
         Pressure9am          0
         Pressure3pm      15028
         Cloud9am             0
         Cloud3pm         59358
         Temp9am              0
         Temp3pm           3609
         RainToday            0
         RainTomorrow         0
         dtype: int64
```

IMPUTING VALUES FOR Evaporation using Median

```
In [87]: df['Evaporation']
```

```
Out[87]: 0        NaN
         1        NaN
         2        NaN
         3        NaN
         4        NaN
                   ..
         145455   NaN
         145456   NaN
         145457   NaN
         145458   NaN
         145459   NaN
         Name: Evaporation, Length: 145460, dtype: float64
```

```
In [88]: df['Evaporation'].dtypes
```

```
Out[88]: dtype('float64')
```

```
In [89]: df['Evaporation'].describe()
```

Out[89]:

```
count    82670.000000
mean         5.468232
std          4.193704
min          0.000000
25%          2.600000
50%          4.800000
75%          7.400000
max        145.000000
Name: Evaporation, dtype: float64
```

In [90]:

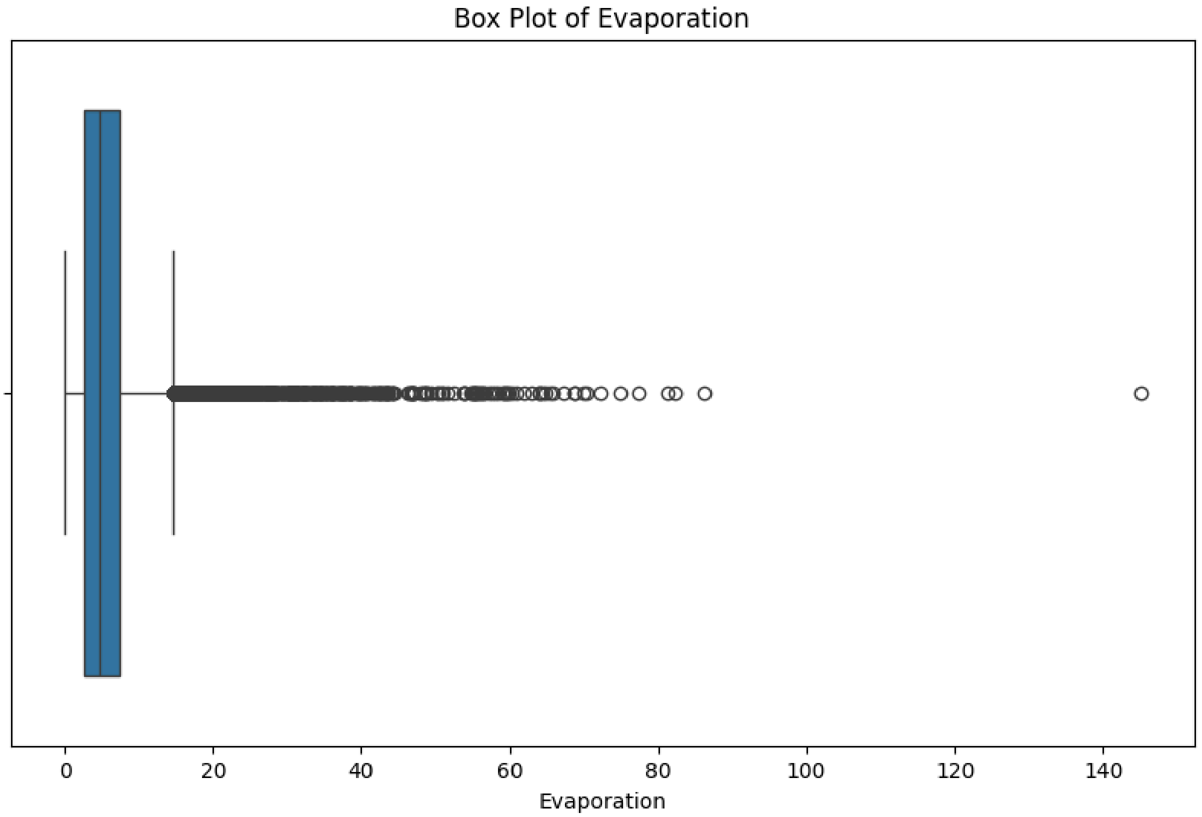
```
outlier_per(df,'Evaporation')
```

Out[90]:

```
1.3715110683349374
```

In [91]:

```
plt.figure(figsize=(10,6))
sns.boxplot(x=df['Evaporation'])
plt.title('Box Plot of Evaporation')
plt.xlabel('Evaporation')
plt.show()
```



In [92]:

```
median_evaporation = df['Evaporation'].median()
```

In [93]:

```
print(median_evaporation)
```

```
4.8
```

In [94]:

```
df['Evaporation'].fillna(value = median_evaporation, inplace = True)
```

In [95]:

```
df.isnull().sum()
```

Out[95]:

```
Date                 0
Location             0
MinTemp              0
MaxTemp              0
Rainfall             0
Evaporation          0
Sunshine         69835
WindGustDir      10326
WindGustSpeed        0
WindDir9am       10566
WindDir3pm        4228
WindSpeed9am      1767
WindSpeed3pm      3062
Humidity9am          0
Humidity3pm       4507
Pressure9am          0
Pressure3pm      15028
Cloud9am             0
Cloud3pm         59358
Temp9am              0
Temp3pm           3609
RainToday            0
RainTomorrow         0
dtype: int64
```

In [96]:

```
df[['Sunshine']]
```

Out[96]:

```
        Sunshine
     0       NaN
     1       NaN
     2       NaN
     3       NaN
     4       NaN
   ...       ...
145455       NaN
145456       NaN
145457       NaN
145458       NaN
145459       NaN

145460 rows × 1 columns
```

In [97]:

```
df['Sunshine'].dtypes
```

Out[97]:

```
dtype('float64')
```

In [98]:

```
df['Sunshine'].describe()
```

Out[98]:

```
count    75625.000000
mean         7.611178
std          3.785483
min          0.000000
25%          4.800000
50%          8.400000
75%         10.600000
max         14.500000
Name: Sunshine, dtype: float64
```

In [99]:

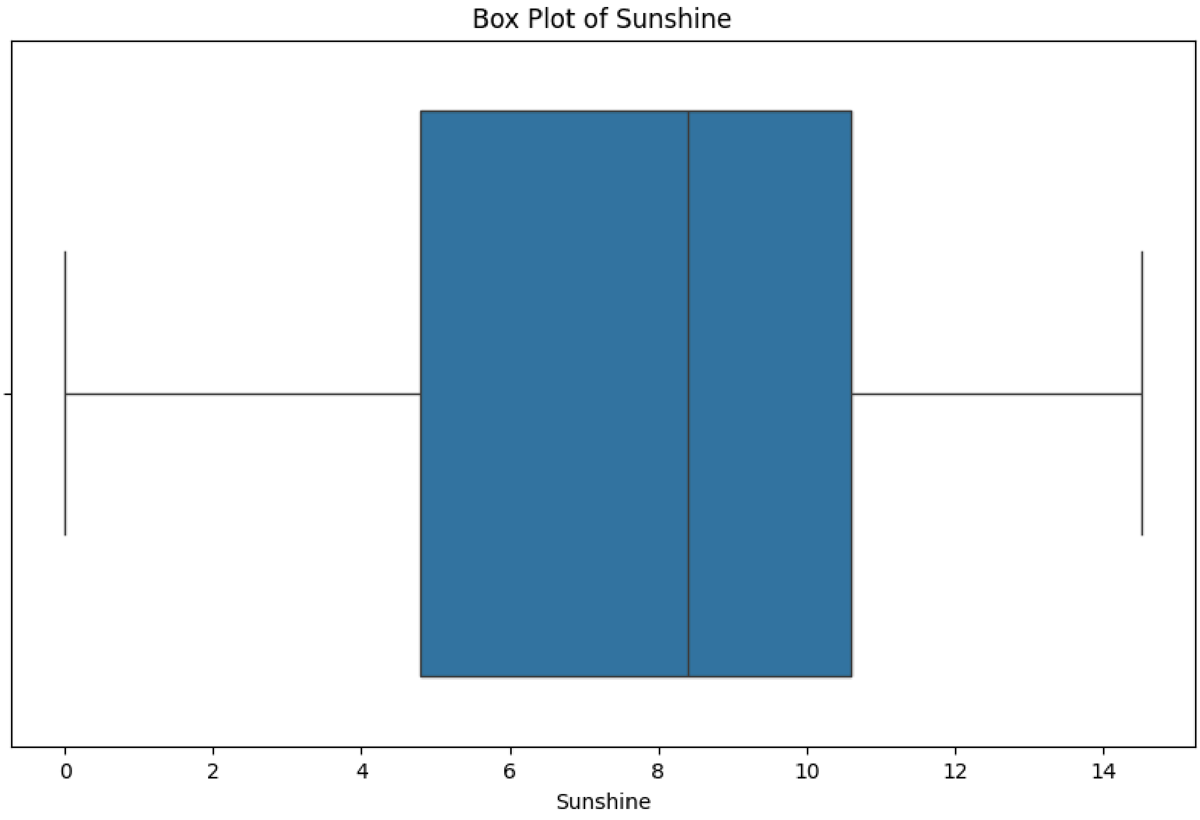
```
outlier_per(df,'Sunshine')
```

Out[99]:

```
0.0
```

In [100…

```
plt.figure(figsize=(10,6))
sns.boxplot(x=df['Sunshine'])
plt.title('Box Plot of Sunshine')
plt.xlabel('Sunshine')
plt.show()
```



In [101…

```
mean_sunshine = df['Sunshine'].mean()
```

In [102…

```
df['Sunshine'].fillna(value = mean_sunshine, inplace = True)
```

In [103…

```
df.isnull().sum()
```

Out[103…

```
Date                0
Location            0
MinTemp             0
MaxTemp             0
Rainfall            0
Evaporation         0
Sunshine            0
WindGustDir     10326
WindGustSpeed       0
WindDir9am      10566
WindDir3pm       4228
WindSpeed9am     1767
WindSpeed3pm     3062
Humidity9am         0
Humidity3pm      4507
Pressure9am         0
Pressure3pm     15028
Cloud9am            0
Cloud3pm        59358
Temp9am             0
Temp3pm          3609
RainToday           0
RainTomorrow        0
dtype: int64
```

In [104…

```
df.describe(include = ['object'])
```

Out[104…

	Date	Location	WindGustDir	WindDir9am	WindDir3pm
count	145460	145460	135134	134894	141232
unique	3436	49	16	16	16
top	2013-11-12	Canberra	W	N	SE
freq	49	3436	9915	11758	10838

Imputing values for WindSpeed using median

In [105…

```
df[['WindSpeed9am']]
```

Out[105...

	WindSpeed9am
0	20.0
1	4.0
2	19.0
3	11.0
4	7.0
...	...
145455	13.0
145456	13.0
145457	9.0
145458	13.0
145459	17.0

145460 rows × 1 columns

In [106...

```
df[['WindSpeed9am']].dtypes
```

Out[106...

```
WindSpeed9am    float64
dtype: object
```

In [107...

```
outlier_per(df,'WindSpeed9am')
```

Out[107...

```
1.249140657225354
```

In [108...

```
median_windspeed9am = df['WindSpeed9am'].median()
print(median_windspeed9am)
```

```
13.0
```

In [109...

```
df['WindSpeed9am'].fillna(value = median_windspeed9am, inplace = True)
```

In [110...

```
df.isnull().sum()
```

Out[110…

```
Date                 0
Location             0
MinTemp              0
MaxTemp              0
Rainfall             0
Evaporation          0
Sunshine             0
WindGustDir      10326
WindGustSpeed        0
WindDir9am       10566
WindDir3pm        4228
WindSpeed9am         0
WindSpeed3pm      3062
Humidity9am          0
Humidity3pm       4507
Pressure9am          0
Pressure3pm      15028
Cloud9am             0
Cloud3pm         59358
Temp9am              0
Temp3pm           3609
RainToday            0
RainTomorrow         0
dtype: int64
```

Imputing values for WindSpeed3pm using mean

In [111…

```
df[['WindSpeed3pm']]
```

Out[111…

	WindSpeed3pm
0	24.0
1	22.0
2	26.0
3	9.0
4	20.0
...	...
145455	11.0
145456	9.0
145457	9.0
145458	7.0
145459	17.0

145460 rows × 1 columns

In [112…

```
df[['WindSpeed3pm']].describe()
```

Out[112...

	WindSpeed3pm
count	142398.000000
mean	18.662657
std	8.809800
min	0.000000
25%	13.000000
50%	19.000000
75%	24.000000
max	87.000000

In [113...

```
df[['WindSpeed3pm']].dtypes
```

Out[113...

```
WindSpeed3pm    float64
dtype: object
```

In [114...

```
outlier_per(df,'WindSpeed3pm')
```

Out[114...

```
1.734497456345387
```

In [115...

```
mean_windspeed3pm = df['WindSpeed3pm'].mean()
```

In [116...

```
df['WindSpeed3pm'].fillna(value = mean_windspeed3pm, inplace = True)
```

In [117...

```
df.isnull().sum()
```

Out[117…

```
Date                 0
Location             0
MinTemp              0
MaxTemp              0
Rainfall             0
Evaporation          0
Sunshine             0
WindGustDir      10326
WindGustSpeed        0
WindDir9am       10566
WindDir3pm        4228
WindSpeed9am         0
WindSpeed3pm         0
Humidity9am          0
Humidity3pm       4507
Pressure9am          0
Pressure3pm      15028
Cloud9am             0
Cloud3pm         59358
Temp9am              0
Temp3pm           3609
RainToday            0
RainTomorrow         0
dtype: int64
```

Imputing Values for Humidity3pm using Mean

In [118…

```
df[['Humidity3pm']]
```

Out[118…

	Humidity3pm
0	22.0
1	25.0
2	30.0
3	16.0
4	33.0
...	...
145455	24.0
145456	21.0
145457	24.0
145458	24.0
145459	36.0

145460 rows × 1 columns

In [119…

```
df['Humidity3pm'].isnull().sum()
```

```
Out[119…  4507
```

```
In [120…  df['Humidity3pm'].describe()
```

```
Out[120…  count    140953.000000
          mean         51.539116
          std          20.795902
          min           0.000000
          25%          37.000000
          50%          52.000000
          75%          66.000000
          max         100.000000
          Name: Humidity3pm, dtype: float64
```

```
In [121…  df['Humidity3pm'].dtypes
```

```
Out[121…  dtype('float64')
```

```
In [122…  outlier_per(df,'Humidity3pm')
```

```
Out[122…  0.0
```

```
In [123…  mean_humidity3pm = df['Humidity3pm'].mean()
```

```
In [124…  df['Humidity3pm'].fillna(value = mean_humidity3pm, inplace = True)
```

```
In [125…  df.isnull().sum()
```

```
Out[125…  Date                 0
          Location             0
          MinTemp              0
          MaxTemp              0
          Rainfall             0
          Evaporation          0
          Sunshine             0
          WindGustDir      10326
          WindGustSpeed        0
          WindDir9am       10566
          WindDir3pm        4228
          WindSpeed9am         0
          WindSpeed3pm         0
          Humidity9am          0
          Humidity3pm          0
          Pressure9am          0
          Pressure3pm      15028
          Cloud9am             0
          Cloud3pm         59358
          Temp9am              0
          Temp3pm           3609
          RainToday            0
          RainTomorrow         0
          dtype: int64
```

Imputing values for Pressure3pm using median

In [126…

```
df['Pressure3pm']
```

Out[126…

```
0         1007.1
1         1007.8
2         1008.7
3         1012.8
4         1006.0
           ...  
145455    1020.3
145456    1019.1
145457    1016.8
145458    1016.5
145459    1017.9
Name: Pressure3pm, Length: 145460, dtype: float64
```

In [127…

```
df['Pressure3pm'].describe()
```

Out[127…

```
count    130432.000000
mean       1015.255889
std           7.037414
min         977.100000
25%        1010.400000
50%        1015.200000
75%        1020.000000
max        1039.600000
Name: Pressure3pm, dtype: float64
```

In [128…

```
df['Pressure3pm'].dtypes
```

Out[128…

```
dtype('float64')
```

In [129…

```
outlier_per(df,'Pressure3pm')
```

Out[129…

```
0.6317888079197029
```

In [130…

```
median_pressure3pm = df['Pressure3pm'].median()
```

In [131…

```
print(median_pressure3pm)
```

```
1015.2
```

In [132…

```
df['Pressure3pm'].fillna(value = median_pressure3pm, inplace = True)
```

In [133…

```
df.isnull().sum()
```

Out[133…

```
Date                0
Location            0
MinTemp             0
MaxTemp             0
Rainfall            0
Evaporation         0
Sunshine            0
WindGustDir     10326
WindGustSpeed       0
WindDir9am      10566
WindDir3pm       4228
WindSpeed9am        0
WindSpeed3pm        0
Humidity9am         0
Humidity3pm         0
Pressure9am         0
Pressure3pm         0
Cloud9am            0
Cloud3pm        59358
Temp9am             0
Temp3pm          3609
RainToday           0
RainTomorrow        0
dtype: int64
```

Imputing values for Cloud3pm using Mean

In [134…

```
df[['Cloud3pm']]
```

Out[134…

	Cloud3pm
0	NaN
1	NaN
2	2.0
3	NaN
4	8.0
...	...
145455	NaN
145456	NaN
145457	NaN
145458	2.0
145459	8.0

145460 rows × 1 columns

In [135…

```
df['Cloud3pm'].describe()
```

Out[135…

```
count    86102.000000
mean         4.509930
std          2.720357
min          0.000000
25%          2.000000
50%          5.000000
75%          7.000000
max          9.000000
Name: Cloud3pm, dtype: float64
```

In [136…

```
df['Cloud3pm'].dtypes
```

Out[136…

```
dtype('float64')
```

In [137…

```
outlier_per(df,'Cloud3pm')
```

Out[137…

```
0.0
```

In [138…

```
mean_cloud3pm = df['Cloud3pm'].mean()
```

In [139…

```
df['Cloud3pm'].fillna(value = mean_cloud3pm, inplace = True)
```

In [140…

```
df.isnull().sum()
```

Out[140…

```
Date                 0
Location             0
MinTemp              0
MaxTemp              0
Rainfall             0
Evaporation          0
Sunshine             0
WindGustDir      10326
WindGustSpeed        0
WindDir9am       10566
WindDir3pm        4228
WindSpeed9am         0
WindSpeed3pm         0
Humidity9am          0
Humidity3pm          0
Pressure9am          0
Pressure3pm          0
Cloud9am             0
Cloud3pm             0
Temp9am              0
Temp3pm           3609
RainToday            0
RainTomorrow         0
dtype: int64
```

Imputing values for Temp3pm using Mean

In [141…

```
df[['Temp3pm']]
```

Out[141…

	Temp3pm
0	21.8
1	24.3
2	23.2
3	26.5
4	29.7
...	...
145455	22.4
145456	24.5
145457	26.1
145458	26.0
145459	20.9

145460 rows × 1 columns

In [142…

```
df['Temp3pm'].describe()
```

Out[142…

```
count    141851.00000
mean         21.68339
std           6.93665
min          -5.40000
25%          16.60000
50%          21.10000
75%          26.40000
max          46.70000
Name: Temp3pm, dtype: float64
```

In [143…

```
df['Temp3pm'].dtypes
```

Out[143…

```
dtype('float64')
```

In [144…

```
outlier_per(df,'Temp3pm')
```

Out[144…

```
0.525230303863605
```

In [145…

```
mean_temp3pm = df['Temp3pm'].mean()
```

In [146…

```
print(mean_temp3pm)
```

```
21.68339031800974
```

In [147…

```
df['Temp3pm'].fillna(value = mean_temp3pm, inplace = True)
```

In [148…

```
df.isnull().sum()
```

Out[148…

```
Date                 0
Location             0
MinTemp              0
MaxTemp              0
Rainfall             0
Evaporation          0
Sunshine             0
WindGustDir      10326
WindGustSpeed        0
WindDir9am       10566
WindDir3pm        4228
WindSpeed9am         0
WindSpeed3pm         0
Humidity9am          0
Humidity3pm          0
Pressure9am          0
Pressure3pm          0
Cloud9am             0
Cloud3pm             0
Temp9am              0
Temp3pm              0
RainToday            0
RainTomorrow         0
dtype: int64
```

Imputing values for WindGustDir using MOde

In [149…

```
df[['WindGustDir']]
```

Out[149…

	WindGustDir
0	W
1	WNW
2	WSW
3	NE
4	W
...	...
145455	E
145456	NNW
145457	N
145458	SE
145459	NaN

145460 rows × 1 columns

In [150…

```
df['WindGustDir'].value_counts()
```

Out[150…

```
WindGustDir
W      9915
SE     9418
N      9313
SSE    9216
E      9181
S      9168
WSW    9069
SW     8967
SSW    8736
WNW    8252
NW     8122
ENE    8104
ESE    7372
NE     7133
NNW    6620
NNE    6548
Name: count, dtype: int64
```

In [151…

```
df['WindGustDir'].describe(include=['object'])
```

Out[151…

```
count     135134
unique        16
top            W
freq        9915
Name: WindGustDir, dtype: object
```

In [152…

```
df['WindGustDir'].dtypes
```

Out[152…

```
dtype('O')
```

In [153…

```
df['WindGustDir'].isnull().sum()
```

Out[153…

```
10326
```

In [154…

```
mode_windgustdir = df['WindGustDir'].mode()[0]
```

In [155…

```
print(mode_windgustdir)
```

```
W
```

In [156…

```
df['WindGustDir'].fillna(value = mode_windgustdir, inplace = True)
```

In [157…

```
df.isnull().sum()
```

Out[157…

```
Date                 0
Location             0
MinTemp              0
MaxTemp              0
Rainfall             0
Evaporation          0
Sunshine             0
WindGustDir          0
WindGustSpeed        0
WindDir9am       10566
WindDir3pm        4228
WindSpeed9am         0
WindSpeed3pm         0
Humidity9am          0
Humidity3pm          0
Pressure9am          0
Pressure3pm          0
Cloud9am             0
Cloud3pm             0
Temp9am              0
Temp3pm              0
RainToday            0
RainTomorrow         0
dtype: int64
```

Imputing Values For WindDir9am using mode

In [158…

```
df[['WindDir9am']]
```

Out[158…

	WindDir9am
0	W
1	NNW
2	W
3	SE
4	ENE
...	...
145455	SE
145456	SE
145457	SE
145458	SSE
145459	ESE

145460 rows × 1 columns

In [159…

```
df[['WindDir9am']].value_counts()
```

Out[159…

```
WindDir9am
N              11758
SE              9287
E               9176
SSE             9112
NW              8749
S               8659
W               8459
SW              8423
NNE             8129
NNW             7980
ENE             7836
NE              7671
ESE             7630
SSW             7587
WNW             7414
WSW             7024
Name: count, dtype: int64
```

In [160…

```
df[['WindDir9am']].dtypes
```

Out[160…

```
WindDir9am    object
dtype: object
```

In [161…

```
mode_WindDir9am = df['WindDir9am'].mode()[0]
```

In [162…

```
print(mode_WindDir9am)
```

```
N
```

In [163…

```
df['WindDir9am'].fillna(value = mode_WindDir9am, inplace = True)
```

In [164…

```
df.isnull().sum()
```

Out[164…

```
Date                0
Location            0
MinTemp             0
MaxTemp             0
Rainfall            0
Evaporation         0
Sunshine            0
WindGustDir         0
WindGustSpeed       0
WindDir9am          0
WindDir3pm       4228
WindSpeed9am        0
WindSpeed3pm        0
Humidity9am         0
Humidity3pm         0
Pressure9am         0
Pressure3pm         0
Cloud9am            0
Cloud3pm            0
Temp9am             0
Temp3pm             0
RainToday           0
RainTomorrow        0
dtype: int64
```

Imputing Values for WindDir3pm using mode

In [165…

```
df[['WindDir3pm']]
```

Out[165…

	WindDir3pm
0	WNW
1	WSW
2	WSW
3	E
4	NW
...	...
145455	ENE
145456	N
145457	WNW
145458	N
145459	ESE

145460 rows × 1 columns

In [166…

```
df[['WindDir3pm']].value_counts()
```

Out[166…

```
WindDir3pm
SE             10838
W              10110
S               9926
WSW             9518
SSE             9399
SW              9354
N               8890
WNW             8874
NW              8610
ESE             8505
E               8472
NE              8263
SSW             8156
NNW             7870
ENE             7857
NNE             6590
Name: count, dtype: int64
```

In [167…

```
df['WindDir3pm'].dtypes
```

Out[167…

```
dtype('O')
```

In [168…

```
mode_WindDir3pm = df['WindDir3pm'].mode()[0]
```

In [169…

```
print(mode_WindDir3pm)
```

```
SE
```

In [170…

```
df['WindDir3pm'].fillna(value = mode_WindDir3pm, inplace = True)
```

In [171…

```
df.isnull().sum()
```

Out[171…

```
Date             0
Location         0
MinTemp          0
MaxTemp          0
Rainfall         0
Evaporation      0
Sunshine         0
WindGustDir      0
WindGustSpeed    0
WindDir9am       0
WindDir3pm       0
WindSpeed9am     0
WindSpeed3pm     0
Humidity9am      0
Humidity3pm      0
Pressure9am      0
Pressure3pm      0
Cloud9am         0
Cloud3pm         0
Temp9am          0
Temp3pm          0
RainToday        0
RainTomorrow     0
dtype: int64
```

Data Type Conversion: Convert date and categorical variables to appropriate formats. The date might be segmented into year, month, and day components for more detailed analysis

In [172…

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

In [173…

```
df[['Date']]
```

Out[173…

	Date
0	2008-12-01
1	2008-12-02
2	2008-12-03
3	2008-12-04
4	2008-12-05
...	...
145455	2017-06-21
145456	2017-06-22
145457	2017-06-23
145458	2017-06-24
145459	2017-06-25

145460 rows × 1 columns

In [174…

```
df
```

Out[174…

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustI
0	2008-12-01	Albury	13.4	22.9	0.6	4.8	7.611178	
1	2008-12-02	Albury	7.4	25.1	0.0	4.8	7.611178	WN
2	2008-12-03	Albury	12.9	25.7	0.0	4.8	7.611178	WS
3	2008-12-04	Albury	9.2	28.0	0.0	4.8	7.611178	I
4	2008-12-05	Albury	17.5	32.3	1.0	4.8	7.611178	
...	...	...	...	...	...	...	...	
145455	2017-06-21	Uluru	2.8	23.4	0.0	4.8	7.611178	
145456	2017-06-22	Uluru	3.6	25.3	0.0	4.8	7.611178	NN
145457	2017-06-23	Uluru	5.4	26.9	0.0	4.8	7.611178	
145458	2017-06-24	Uluru	7.8	27.0	0.0	4.8	7.611178	
145459	2017-06-25	Uluru	14.9	22.6	0.0	4.8	7.611178	

145460 rows × 26 columns

In [175…

```
df['Location'].unique()
```

Out[175…

```
array(['Albury', 'BadgerysCreek', 'Cobar', 'CoffsHarbour', 'Moree',
       'Newcastle', 'NorahHead', 'NorfolkIsland', 'Penrith', 'Richmond',
       'Sydney', 'SydneyAirport', 'WaggaWagga', 'Williamtown',
       'Wollongong', 'Canberra', 'Tuggeranong', 'MountGinini', 'Ballarat',
       'Bendigo', 'Sale', 'MelbourneAirport', 'Melbourne', 'Mildura',
       'Nhil', 'Portland', 'Watsonia', 'Dartmoor', 'Brisbane', 'Cairns',
       'GoldCoast', 'Townsville', 'Adelaide', 'MountGambier', 'Nuriootpa',
       'Woomera', 'Albany', 'Witchcliffe', 'PearceRAAF', 'PerthAirport',
       'Perth', 'SalmonGums', 'Walpole', 'Hobart', 'Launceston',
       'AliceSprings', 'Darwin', 'Katherine', 'Uluru'], dtype=object)
```

In [176…

```
locations = ['Albury', 'BadgerysCreek', 'Cobar', 'CoffsHarbour', 'Moree',
             'Newcastle', 'NorahHead', 'NorfolkIsland', 'Penrith', 'Richmond',
             'Sydney', 'SydneyAirport', 'WaggaWagga', 'Williamtown',
             'Wollongong', 'Canberra', 'Tuggeranong', 'MountGinini', 'Ballarat',
             'Bendigo', 'Sale', 'MelbourneAirport', 'Melbourne', 'Mildura',
             'Nhil', 'Portland', 'Watsonia', 'Dartmoor', 'Brisbane', 'Cairns',
             'GoldCoast', 'Townsville', 'Adelaide', 'MountGambier', 'Nuriootpa',
             'Woomera', 'Albany', 'Witchcliffe', 'PearceRAAF', 'PerthAirport',
             'Perth', 'SalmonGums', 'Walpole', 'Hobart', 'Launceston',
             'AliceSprings', 'Darwin', 'Katherine', 'Uluru']

location_dict = {location: index + 1 for index, location in enumerate(locations)}

df['Location'] = df['Location'].map(location_dict)

print(df)
```

```
              Date  Location  MinTemp  MaxTemp  Rainfall  Evaporation  \
0       2008-12-01         1     13.4     22.9       0.6          4.8   
1       2008-12-02         1      7.4     25.1       0.0          4.8   
2       2008-12-03         1     12.9     25.7       0.0          4.8   
3       2008-12-04         1      9.2     28.0       0.0          4.8   
4       2008-12-05         1     17.5     32.3       1.0          4.8   
...            ...       ...      ...      ...       ...          ...   
145455  2017-06-21        49      2.8     23.4       0.0          4.8   
145456  2017-06-22        49      3.6     25.3       0.0          4.8   
145457  2017-06-23        49      5.4     26.9       0.0          4.8   
145458  2017-06-24        49      7.8     27.0       0.0          4.8   
145459  2017-06-25        49     14.9     22.6       0.0          4.8   

        Sunshine WindGustDir  WindGustSpeed WindDir9am  ... Pressure3pm  \
0       7.611178           W           44.0          W  ...      1007.1   
1       7.611178         WNW           44.0        NNW  ...      1007.8   
2       7.611178         WSW           46.0          W  ...      1008.7   
3       7.611178          NE           24.0         SE  ...      1012.8   
4       7.611178           W           41.0        ENE  ...      1006.0   
...          ...         ...            ...        ...  ...         ...   
145455  7.611178           E           31.0         SE  ...      1020.3   
145456  7.611178         NNW           22.0         SE  ...      1019.1   
145457  7.611178           N           37.0         SE  ...      1016.8   
145458  7.611178          SE           28.0        SSE  ...      1016.5   
145459  7.611178           W           39.0        ESE  ...      1017.9   

        Cloud9am  Cloud3pm  Temp9am  Temp3pm  RainToday  RainTomorrow  Year  \
0       8.000000   4.50993     16.9     21.8          0             0  2008   
1       4.447461   4.50993     17.2     24.3          0             0  2008   
2       4.447461   2.00000     21.0     23.2          0             0  2008   
3       4.447461   4.50993     18.1     26.5          0             0  2008   
4       7.000000   8.00000     17.8     29.7          0             0  2008   
...          ...       ...      ...      ...        ...           ...   ...   
145455  4.447461   4.50993     10.1     22.4          0             0  2017   
145456  4.447461   4.50993     10.9     24.5          0             0  2017   
145457  4.447461   4.50993     12.5     26.1          0             0  2017   
145458  3.000000   2.00000     15.1     26.0          0             0  2017   
145459  8.000000   8.00000     15.0     20.9          0             0  2017   

        Month  Day  
0          12    1  
1          12    2  
2          12    3  
3          12    4  
4          12    5  
...       ...  ...  
145455      6   21  
145456      6   22  
145457      6   23  
145458      6   24  
145459      6   25  

[145460 rows x 26 columns]
```

In [177…

```
df['WindGustDir'].unique()
```

Out[177…

```
array(['W', 'WNW', 'WSW', 'NE', 'NNW', 'N', 'NNE', 'SW', 'ENE', 'SSE',
       'S', 'NW', 'SE', 'ESE', 'E', 'SSW'], dtype=object)
```

In [178…

```
WGD = ['W', 'WNW', 'WSW', 'NE', 'NNW', 'N', 'NNE', 'SW', 'ENE', 'SSE',
       'S', 'NW', 'SE', 'ESE', 'E', 'SSW']

wgd_dict = {WGD: index + 1 for index, WGD in enumerate(WGD )}

df['WindGustDir'] = df['WindGustDir'].map(wgd_dict)

print(df)
```

```
                 Date  Location  MinTemp  MaxTemp  Rainfall  Evaporation  \
0      2008-12-01         1     13.4     22.9       0.6          4.8   
1      2008-12-02         1      7.4     25.1       0.0          4.8   
2      2008-12-03         1     12.9     25.7       0.0          4.8   
3      2008-12-04         1      9.2     28.0       0.0          4.8   
4      2008-12-05         1     17.5     32.3       1.0          4.8   
...           ...       ...      ...      ...       ...          ...   
145455 2017-06-21        49      2.8     23.4       0.0          4.8   
145456 2017-06-22        49      3.6     25.3       0.0          4.8   
145457 2017-06-23        49      5.4     26.9       0.0          4.8   
145458 2017-06-24        49      7.8     27.0       0.0          4.8   
145459 2017-06-25        49     14.9     22.6       0.0          4.8   

        Sunshine  WindGustDir  WindGustSpeed WindDir9am  ... Pressure3pm  \
0       7.611178            1           44.0          W  ...      1007.1   
1       7.611178            2           44.0        NNW  ...      1007.8   
2       7.611178            3           46.0          W  ...      1008.7   
3       7.611178            4           24.0         SE  ...      1012.8   
4       7.611178            1           41.0        ENE  ...      1006.0   
...          ...          ...            ...        ...  ...         ...   
145455  7.611178           15           31.0         SE  ...      1020.3   
145456  7.611178            5           22.0         SE  ...      1019.1   
145457  7.611178            6           37.0         SE  ...      1016.8   
145458  7.611178           13           28.0        SSE  ...      1016.5   
145459  7.611178            1           39.0        ESE  ...      1017.9   

        Cloud9am  Cloud3pm  Temp9am  Temp3pm  RainToday  RainTomorrow  Year  \
0       8.000000   4.50993     16.9     21.8          0             0  2008   
1       4.447461   4.50993     17.2     24.3          0             0  2008   
2       4.447461   2.00000     21.0     23.2          0             0  2008   
3       4.447461   4.50993     18.1     26.5          0             0  2008   
4       7.000000   8.00000     17.8     29.7          0             0  2008   
...          ...       ...      ...      ...        ...           ...   ...   
145455  4.447461   4.50993     10.1     22.4          0             0  2017   
145456  4.447461   4.50993     10.9     24.5          0             0  2017   
145457  4.447461   4.50993     12.5     26.1          0             0  2017   
145458  3.000000   2.00000     15.1     26.0          0             0  2017   
145459  8.000000   8.00000     15.0     20.9          0             0  2017   

        Month  Day  
0          12    1  
1          12    2  
2          12    3  
3          12    4  
4          12    5  
...       ...  ...  
145455      6   21  
145456      6   22  
145457      6   23  
145458      6   24  
145459      6   25  

[145460 rows x 26 columns]
```

In [179…

```
df['WindDir9am'].unique()
```

Out[179…

```
array(['W', 'NNW', 'SE', 'ENE', 'SW', 'SSE', 'S', 'NE', 'N', 'SSW', 'WSW',
       'ESE', 'E', 'NW', 'WNW', 'NNE'], dtype=object)
```

In [180…

```
WGD9am = ['W', 'NNW', 'SE', 'ENE', 'SW', 'SSE', 'S', 'NE', 'N', 'SSW', 'WSW',
       'ESE', 'E', 'NW', 'WNW', 'NNE']

wgd9am_dict = {WGD9am: index + 1 for index, WGD9am in enumerate(WGD9am )}

df['WindDir9am'] = df['WindDir9am'].map(wgd9am_dict)

print(df)
```

```
              Date  Location  MinTemp  MaxTemp  Rainfall  Evaporation  \
0       2008-12-01         1     13.4     22.9       0.6          4.8   
1       2008-12-02         1      7.4     25.1       0.0          4.8   
2       2008-12-03         1     12.9     25.7       0.0          4.8   
3       2008-12-04         1      9.2     28.0       0.0          4.8   
4       2008-12-05         1     17.5     32.3       1.0          4.8   
...            ...       ...      ...      ...       ...          ...   
145455  2017-06-21        49      2.8     23.4       0.0          4.8   
145456  2017-06-22        49      3.6     25.3       0.0          4.8   
145457  2017-06-23        49      5.4     26.9       0.0          4.8   
145458  2017-06-24        49      7.8     27.0       0.0          4.8   
145459  2017-06-25        49     14.9     22.6       0.0          4.8   

        Sunshine  WindGustDir  WindGustSpeed  WindDir9am  ... Pressure3pm  \
0       7.611178            1           44.0           1  ...      1007.1   
1       7.611178            2           44.0           2  ...      1007.8   
2       7.611178            3           46.0           1  ...      1008.7   
3       7.611178            4           24.0           3  ...      1012.8   
4       7.611178            1           41.0           4  ...      1006.0   
...          ...          ...            ...         ...  ...         ...   
145455  7.611178           15           31.0           3  ...      1020.3   
145456  7.611178            5           22.0           3  ...      1019.1   
145457  7.611178            6           37.0           3  ...      1016.8   
145458  7.611178           13           28.0           6  ...      1016.5   
145459  7.611178            1           39.0          12  ...      1017.9   

        Cloud9am  Cloud3pm  Temp9am  Temp3pm  RainToday  RainTomorrow  Year  \
0       8.000000   4.50993     16.9     21.8          0             0  2008   
1       4.447461   4.50993     17.2     24.3          0             0  2008   
2       4.447461   2.00000     21.0     23.2          0             0  2008   
3       4.447461   4.50993     18.1     26.5          0             0  2008   
4       7.000000   8.00000     17.8     29.7          0             0  2008   
...          ...       ...      ...      ...        ...           ...   ...   
145455  4.447461   4.50993     10.1     22.4          0             0  2017   
145456  4.447461   4.50993     10.9     24.5          0             0  2017   
145457  4.447461   4.50993     12.5     26.1          0             0  2017   
145458  3.000000   2.00000     15.1     26.0          0             0  2017   
145459  8.000000   8.00000     15.0     20.9          0             0  2017   

        Month  Day  
0          12    1  
1          12    2  
2          12    3  
3          12    4  
4          12    5  
...       ...  ...  
145455      6   21  
145456      6   22  
145457      6   23  
145458      6   24  
145459      6   25  

[145460 rows x 26 columns]
```

In [181…

```
df['WindDir3pm'].unique()
```

Out[181…

```
array(['WNW', 'WSW', 'E', 'NW', 'W', 'SSE', 'ESE', 'ENE', 'NNW', 'SSW',
       'SW', 'SE', 'N', 'S', 'NNE', 'NE'], dtype=object)
```

In [182…

```
WGD3pm = ['WNW', 'WSW', 'E', 'NW', 'W', 'SSE', 'ESE', 'ENE', 'NNW', 'SSW',
       'SW', 'SE', 'N', 'S', 'NNE', 'NE']

wgd3pm_dict = {WGD3pm: index + 1 for index, WGD3pm in enumerate(WGD3pm )}

df['WindDir3pm'] = df['WindDir3pm'].map(wgd3pm_dict)

print(df)
```

```
             Date  Location  MinTemp  MaxTemp  Rainfall  Evaporation  \
0      2008-12-01         1     13.4     22.9       0.6          4.8   
1      2008-12-02         1      7.4     25.1       0.0          4.8   
2      2008-12-03         1     12.9     25.7       0.0          4.8   
3      2008-12-04         1      9.2     28.0       0.0          4.8   
4      2008-12-05         1     17.5     32.3       1.0          4.8   
...           ...       ...      ...      ...       ...          ...   
145455 2017-06-21        49      2.8     23.4       0.0          4.8   
145456 2017-06-22        49      3.6     25.3       0.0          4.8   
145457 2017-06-23        49      5.4     26.9       0.0          4.8   
145458 2017-06-24        49      7.8     27.0       0.0          4.8   
145459 2017-06-25        49     14.9     22.6       0.0          4.8   

        Sunshine  WindGustDir  WindGustSpeed  WindDir9am  ...  Pressure3pm  \
0       7.611178            1           44.0           1  ...       1007.1   
1       7.611178            2           44.0           2  ...       1007.8   
2       7.611178            3           46.0           1  ...       1008.7   
3       7.611178            4           24.0           3  ...       1012.8   
4       7.611178            1           41.0           4  ...       1006.0   
...          ...          ...            ...         ...  ...          ...   
145455  7.611178           15           31.0           3  ...       1020.3   
145456  7.611178            5           22.0           3  ...       1019.1   
145457  7.611178            6           37.0           3  ...       1016.8   
145458  7.611178           13           28.0           6  ...       1016.5   
145459  7.611178            1           39.0          12  ...       1017.9   

        Cloud9am  Cloud3pm  Temp9am  Temp3pm  RainToday  RainTomorrow  Year  \
0       8.000000   4.50993     16.9     21.8          0             0  2008   
1       4.447461   4.50993     17.2     24.3          0             0  2008   
2       4.447461   2.00000     21.0     23.2          0             0  2008   
3       4.447461   4.50993     18.1     26.5          0             0  2008   
4       7.000000   8.00000     17.8     29.7          0             0  2008   
...          ...       ...      ...      ...        ...           ...   ...   
145455  4.447461   4.50993     10.1     22.4          0             0  2017   
145456  4.447461   4.50993     10.9     24.5          0             0  2017   
145457  4.447461   4.50993     12.5     26.1          0             0  2017   
145458  3.000000   2.00000     15.1     26.0          0             0  2017   
145459  8.000000   8.00000     15.0     20.9          0             0  2017   

        Month  Day  
0          12    1  
1          12    2  
2          12    3  
3          12    4  
4          12    5  
...       ...  ...  
145455      6   21  
145456      6   22  
145457      6   23  
145458      6   24  
145459      6   25  

[145460 rows x 26 columns]
```

In [183…

```
df.dtypes
```

Out[183…

```
Date             datetime64[ns]
Location                  int64
MinTemp                 float64
MaxTemp                 float64
Rainfall                float64
Evaporation             float64
Sunshine                float64
WindGustDir               int64
WindGustSpeed           float64
WindDir9am                int64
WindDir3pm                int64
WindSpeed9am            float64
WindSpeed3pm            float64
Humidity9am             float64
Humidity3pm             float64
Pressure9am             float64
Pressure3pm             float64
Cloud9am                float64
Cloud3pm                float64
Temp9am                 float64
Temp3pm                 float64
RainToday                 int64
RainTomorrow              int64
Year                      int32
Month                     int32
Day                       int32
dtype: object
```

In [184…

```python
df.shape
```

Out[184…

```
(145460, 26)
```

In [185…

```python
df.columns
```

Out[185…

```
Index(['Date', 'Location', 'MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation',
       'Sunshine', 'WindGustDir', 'WindGustSpeed', 'WindDir9am', 'WindDir3pm',
       'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm',
       'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am',
       'Temp3pm', 'RainToday', 'RainTomorrow', 'Year', 'Month', 'Day'],
      dtype='object')
```

Outlier Detection

In [186…

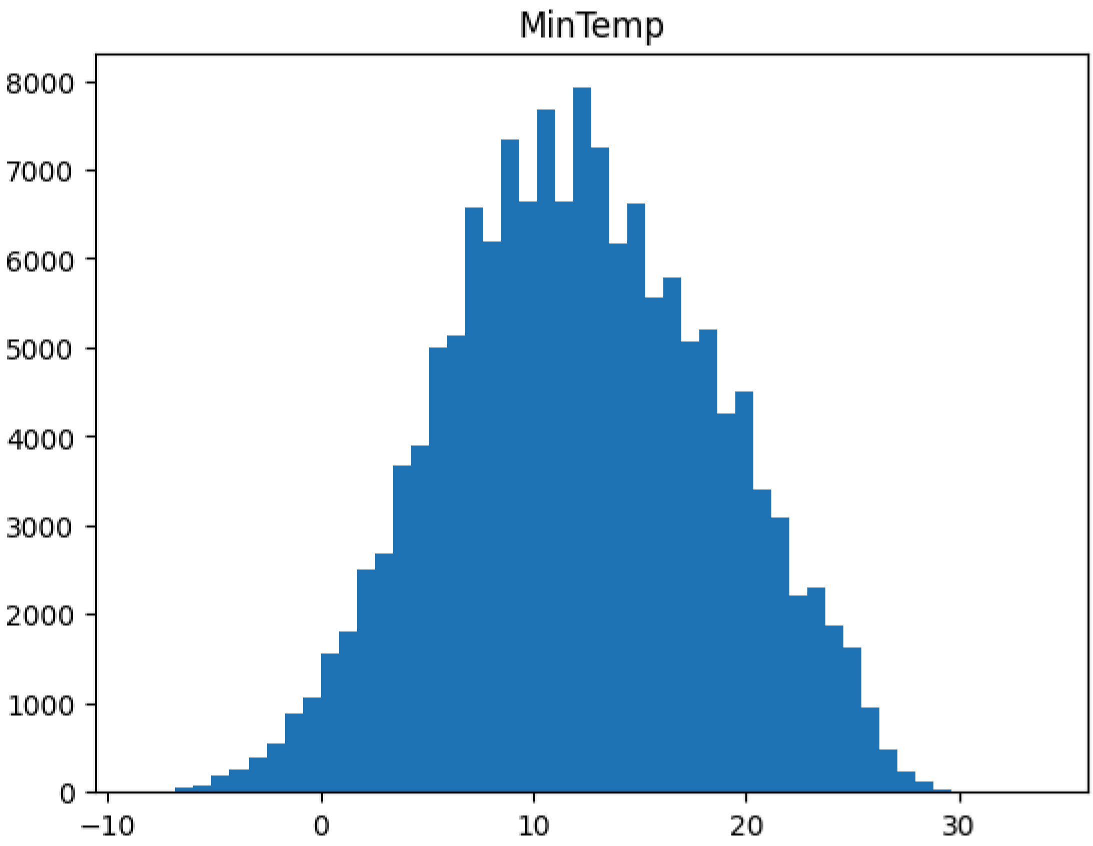
```python
# Assuming df is your DataFrame
# Replace the below line with your actual DataFrame loading or creation code
# df = pd.read_csv('your_data.csv')

# List of numeric columns for which outlier detection is suitable
numeric_columns = ['MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation',
                   'Sunshine', 'WindGustSpeed', 'WindSpeed9am', 'WindSpeed3pm',
                   'Humidity9am', 'Humidity3pm', 'Pressure9am', 'Pressure3pm',
                   'Cloud9am', 'Cloud3pm', 'Temp9am', 'Temp3pm']

# Dictionary to store the percentage of outliers for each column
```

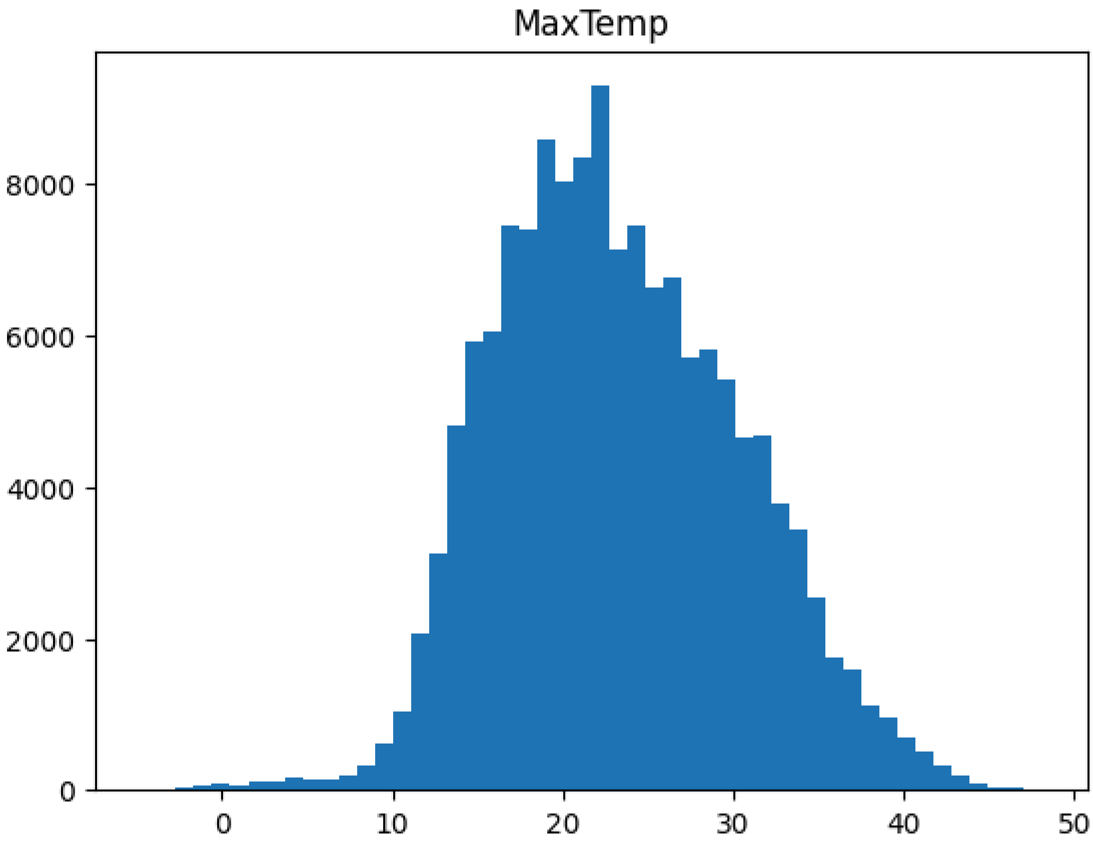
```python
outlier_percentages = {}

# Loop through the numeric columns and calculate the percentage of outliers
for column in numeric_columns:
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Counting outliers
    outliers = df[(df[column] < lower_bound) | (df[column] > upper_bound)]
    total_count = df[column].count()
    outlier_percentage = (len(outliers) / total_count) * 100
    outlier_percentages[column] = outlier_percentage

# Print the percentage of outliers for each column
for column, percentage in outlier_percentages.items():
    print(f'Percentage of outliers in {column}: {percentage:.2f}%')
```

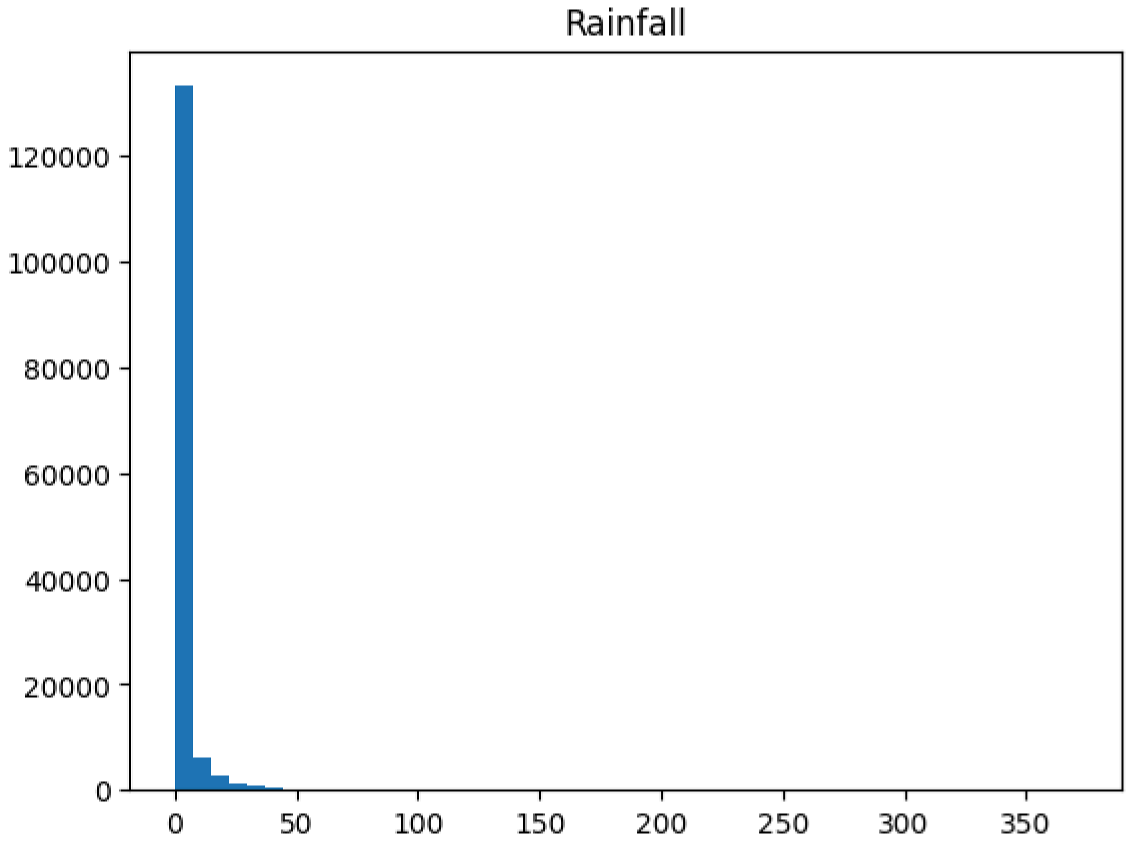
```
Percentage of outliers in MinTemp: 0.06%
Percentage of outliers in MaxTemp: 0.37%
Percentage of outliers in Rainfall: 19.89%
Percentage of outliers in Evaporation: 25.76%
Percentage of outliers in Sunshine: 31.33%
Percentage of outliers in WindGustSpeed: 3.80%
Percentage of outliers in WindSpeed9am: 1.25%
Percentage of outliers in WindSpeed3pm: 1.73%
Percentage of outliers in Humidity9am: 0.98%
Percentage of outliers in Humidity3pm: 0.00%
Percentage of outliers in Pressure9am: 1.90%
Percentage of outliers in Pressure3pm: 1.74%
Percentage of outliers in Cloud9am: 0.00%
Percentage of outliers in Cloud3pm: 3.42%
Percentage of outliers in Temp9am: 0.21%
Percentage of outliers in Temp3pm: 0.68%
```

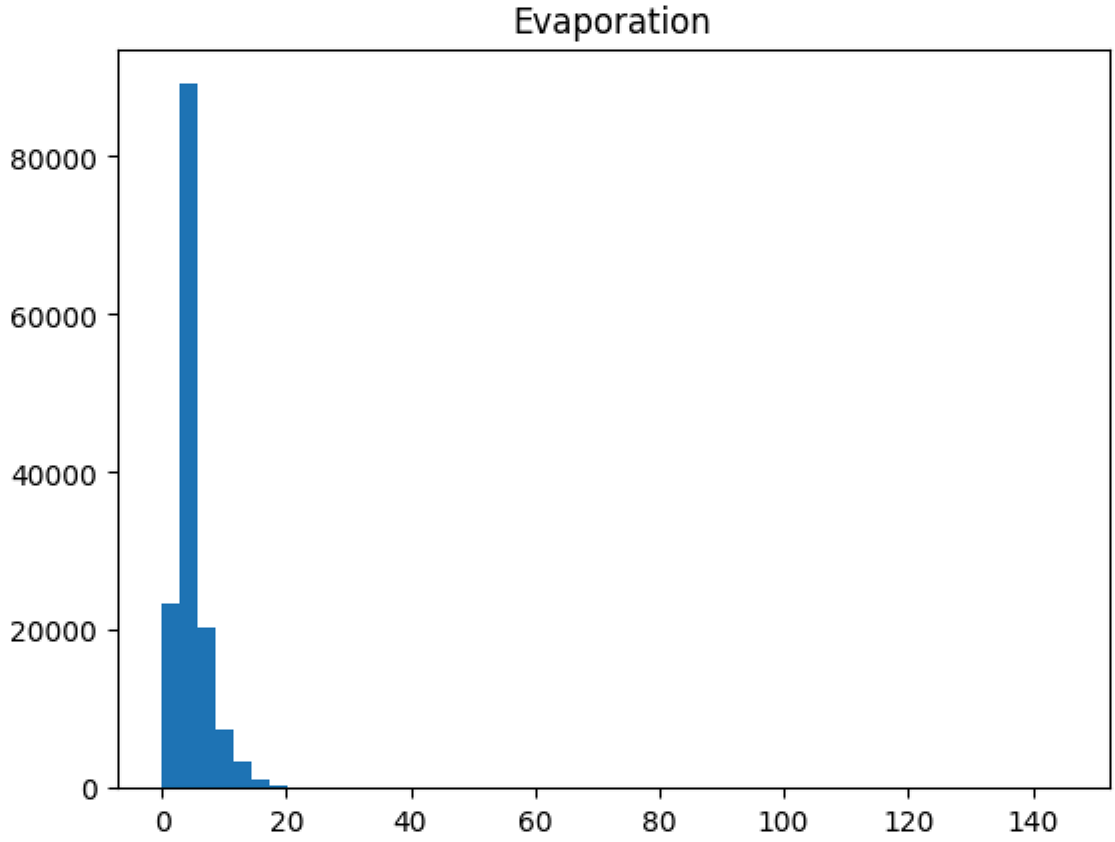
In [187…

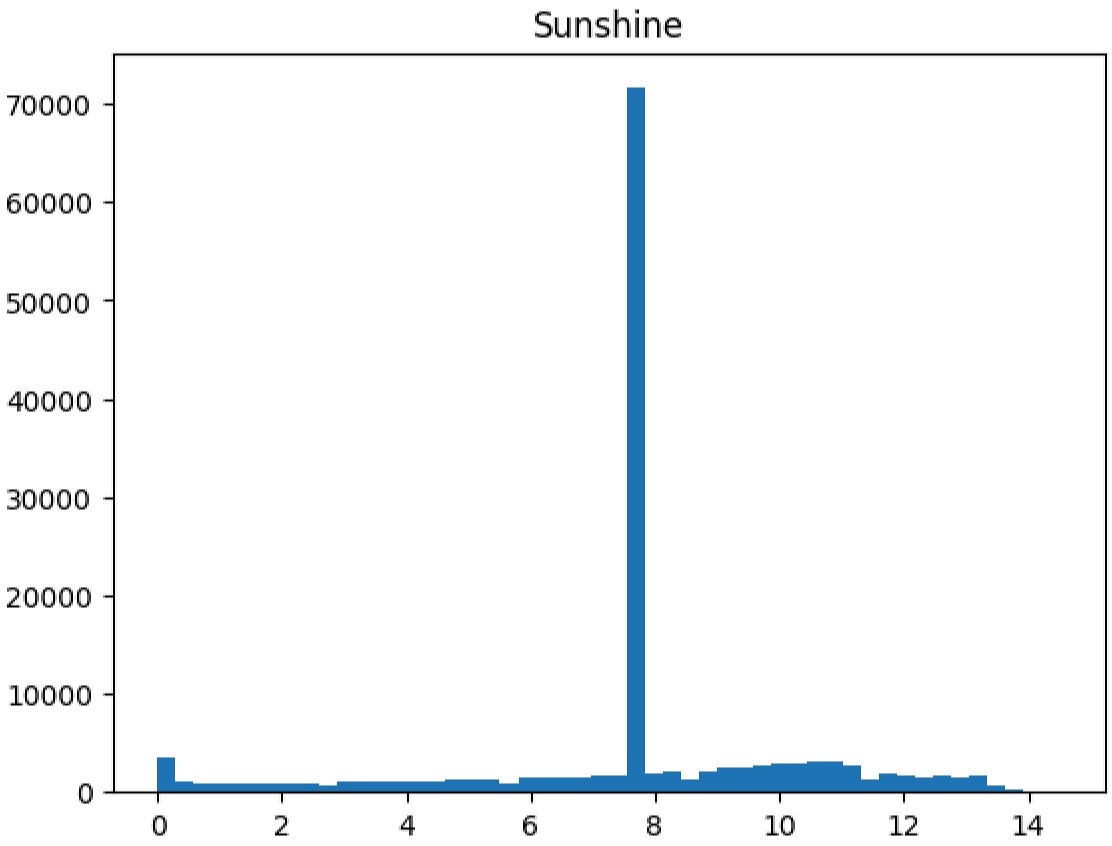
```python
import matplotlib.pyplot as plt
numeric_columns = ['MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation',
                   'Sunshine', 'WindGustSpeed', 'WindSpeed9am', 'WindSpeed3pm',
                   'Humidity9am', 'Humidity3pm', 'Pressure9am', 'Pressure3pm',
                   'Cloud9am', 'Cloud3pm', 'Temp9am', 'Temp3pm']

for col in numeric_columns:
    plt.hist(df[col], bins=50)
    plt.title(col)
    plt.show()
```

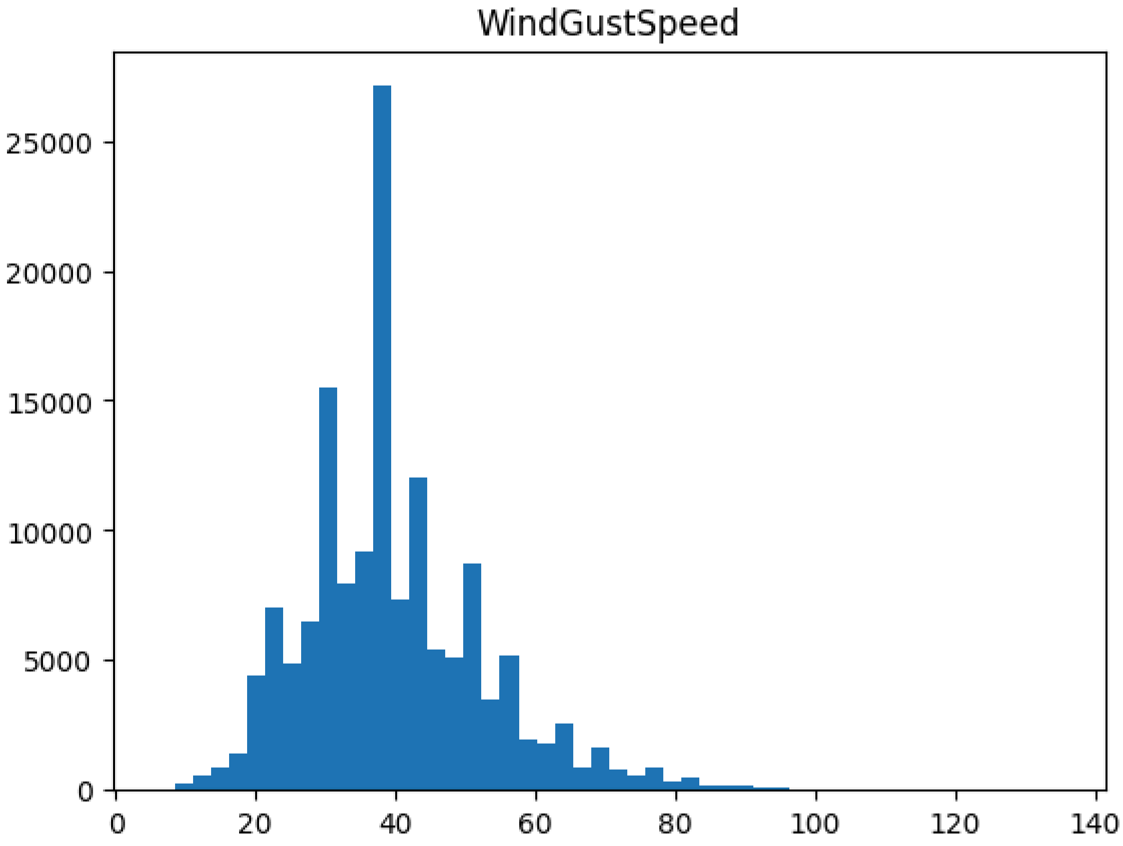

MinTemp
0
1000
2000
3000
4000
5000
6000
7000
8000
−10
0
10
20
30

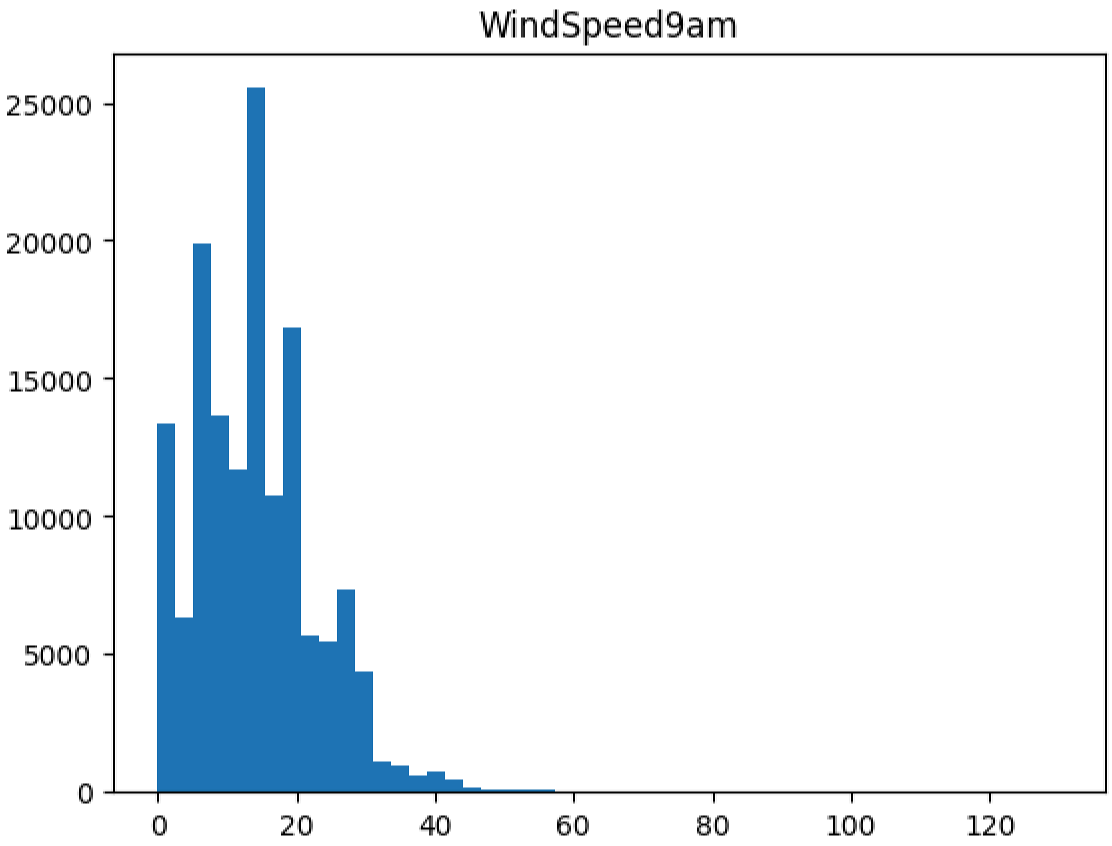

MaxTemp
0
2000
4000
6000
8000
0
10
20
30
40
50

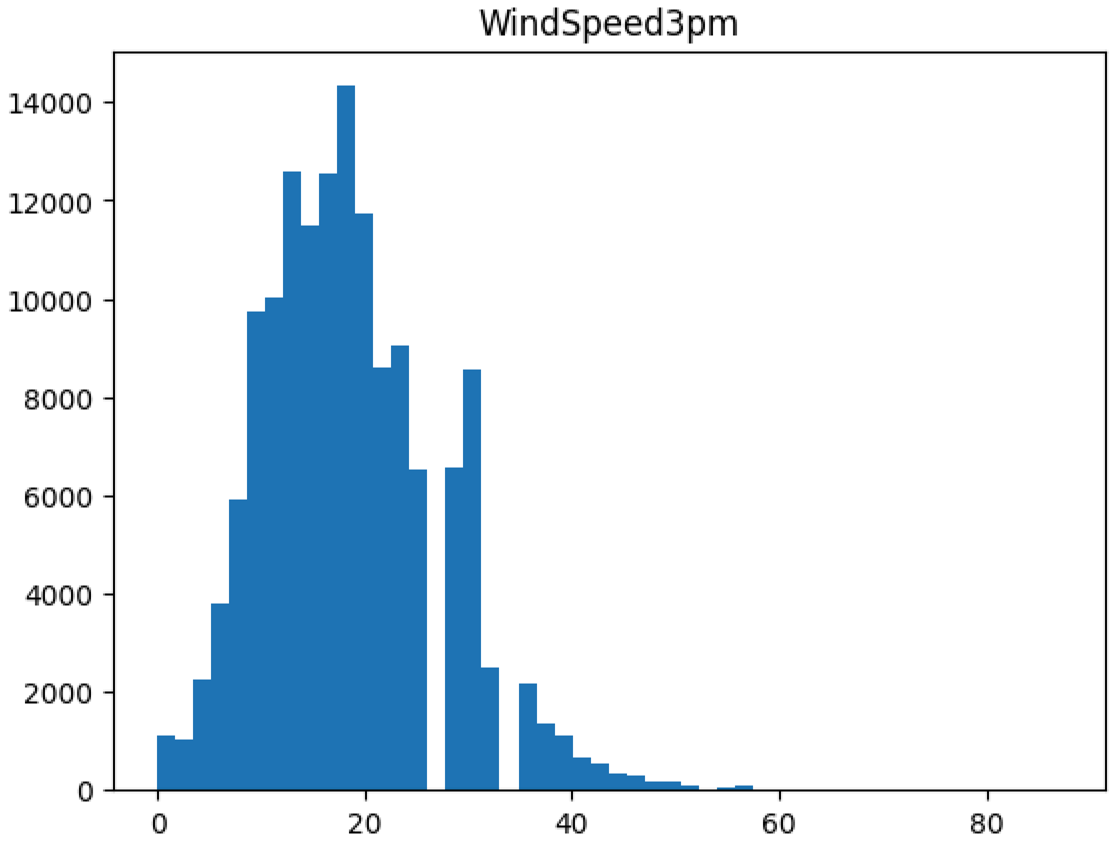

Rainfall
0
20000
40000
60000
80000
100000
120000
0
50
100
150
200
250
300
350

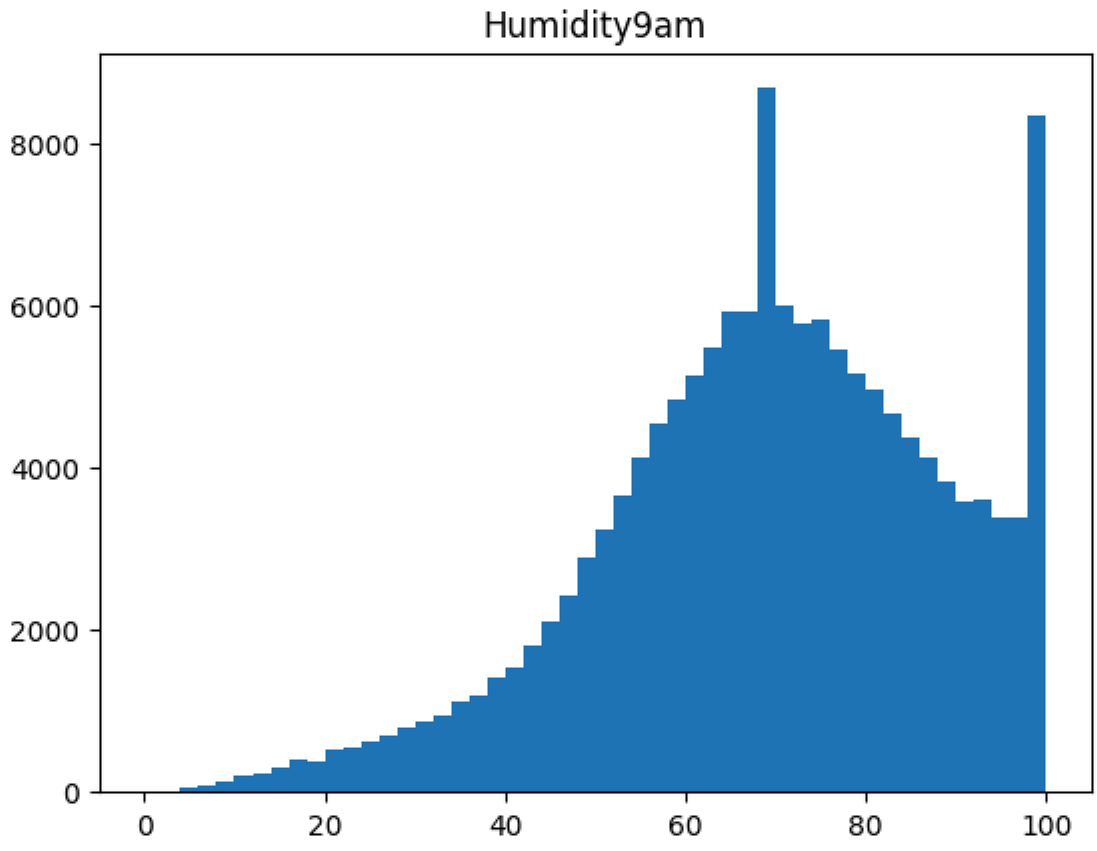

Evaporation
0
20000
40000
60000
80000
0
20
40
60
80
100
120
140

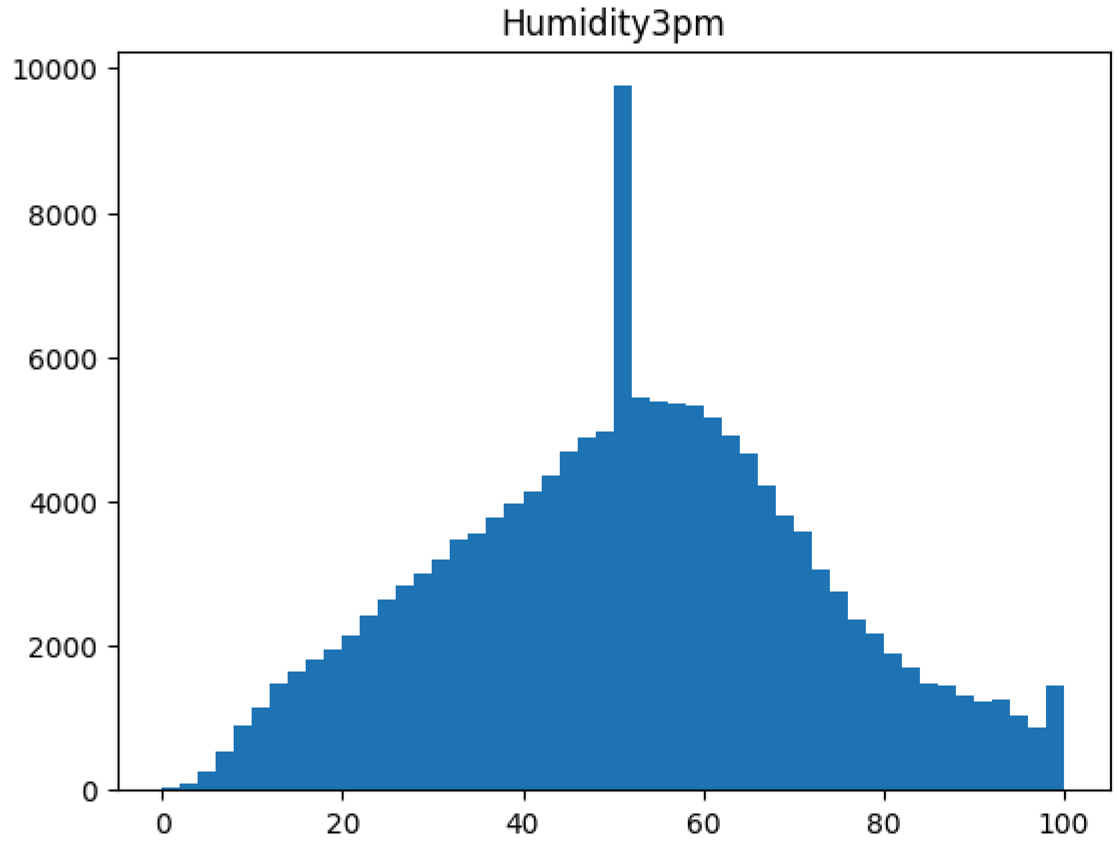

Sunshine
0
10000
20000
30000
40000
50000
60000
70000
0
2
4
6
8
10
12
14

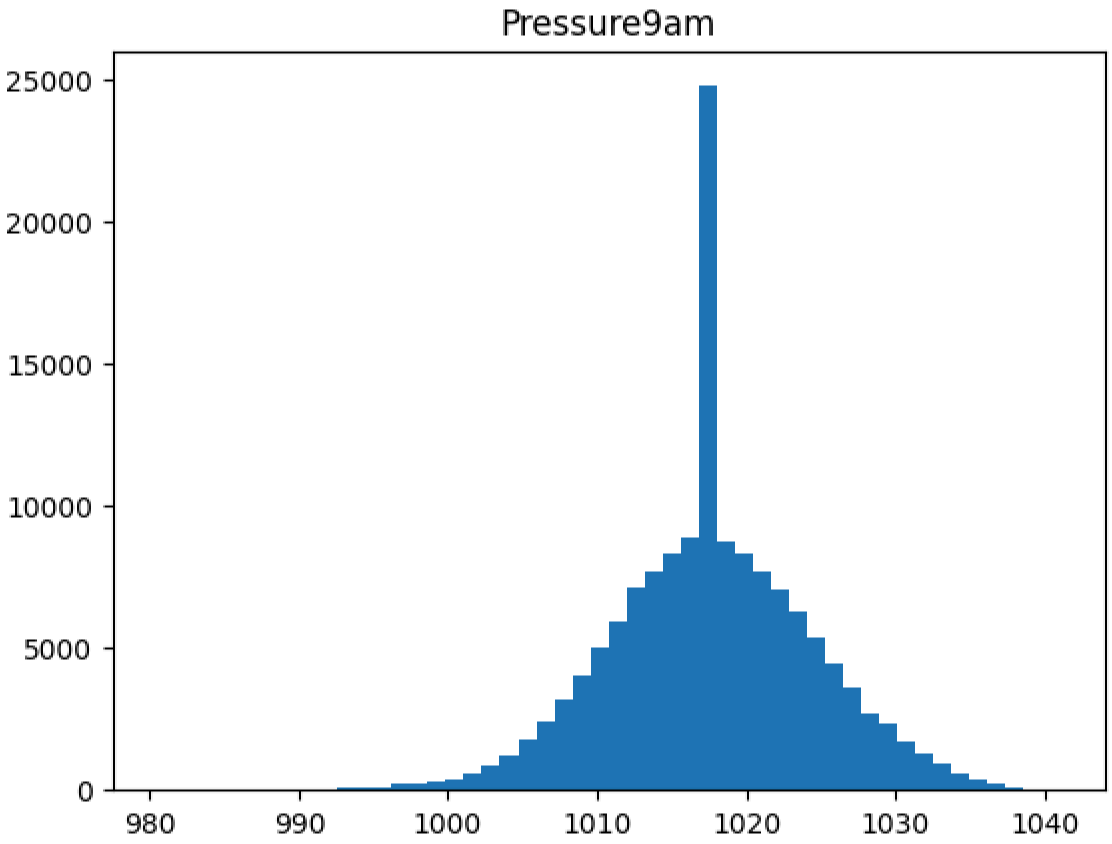

WindGustSpeed
0
5000
10000
15000
20000
25000
0
20
40
60
80
100
120
140

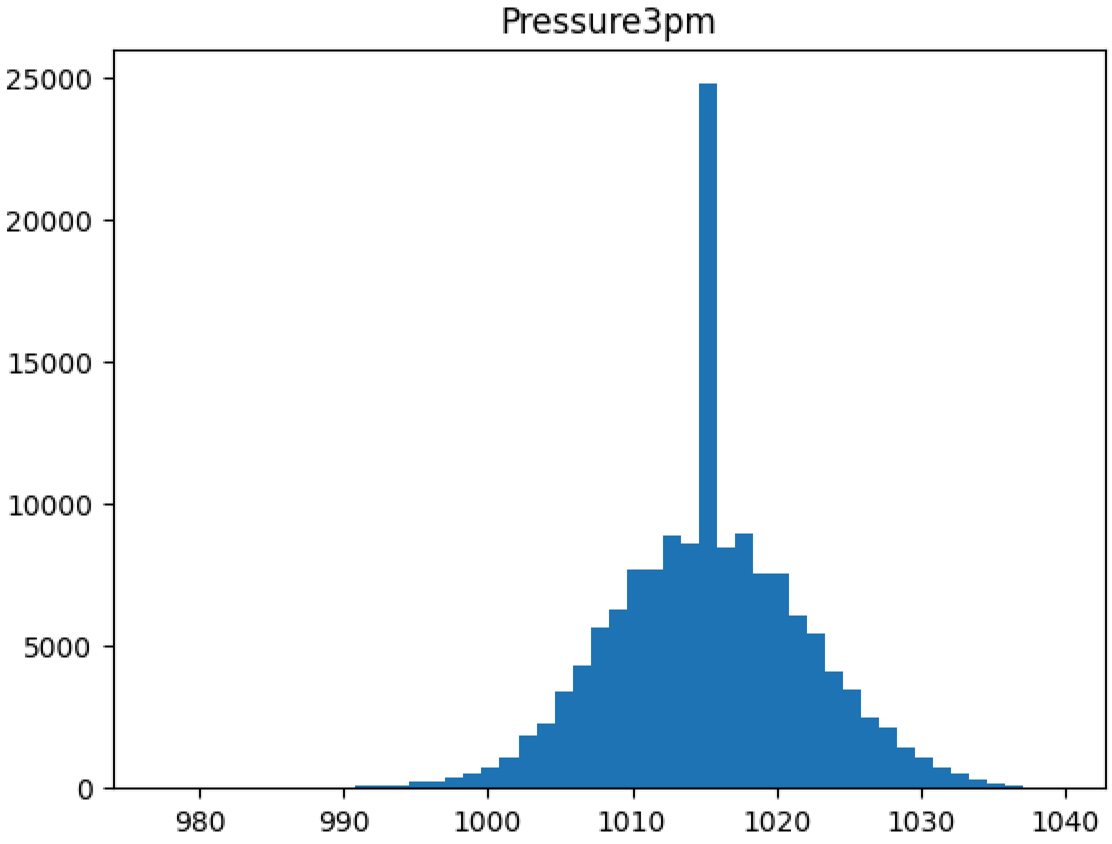

WindSpeed9am
25000
20000
15000
10000
5000
0
0
20
40
60
80
100
120

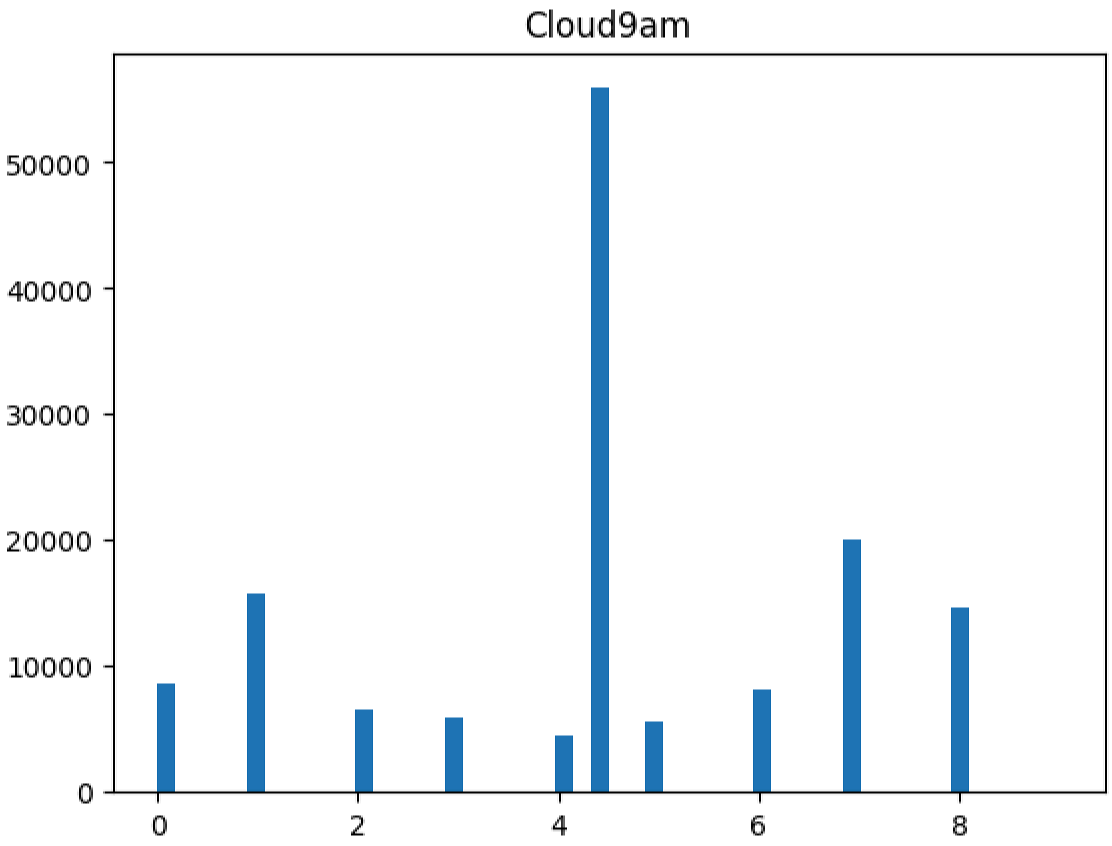

WindSpeed3pm
14000
12000
10000
8000
6000
4000
2000
0
0
20
40
60
80

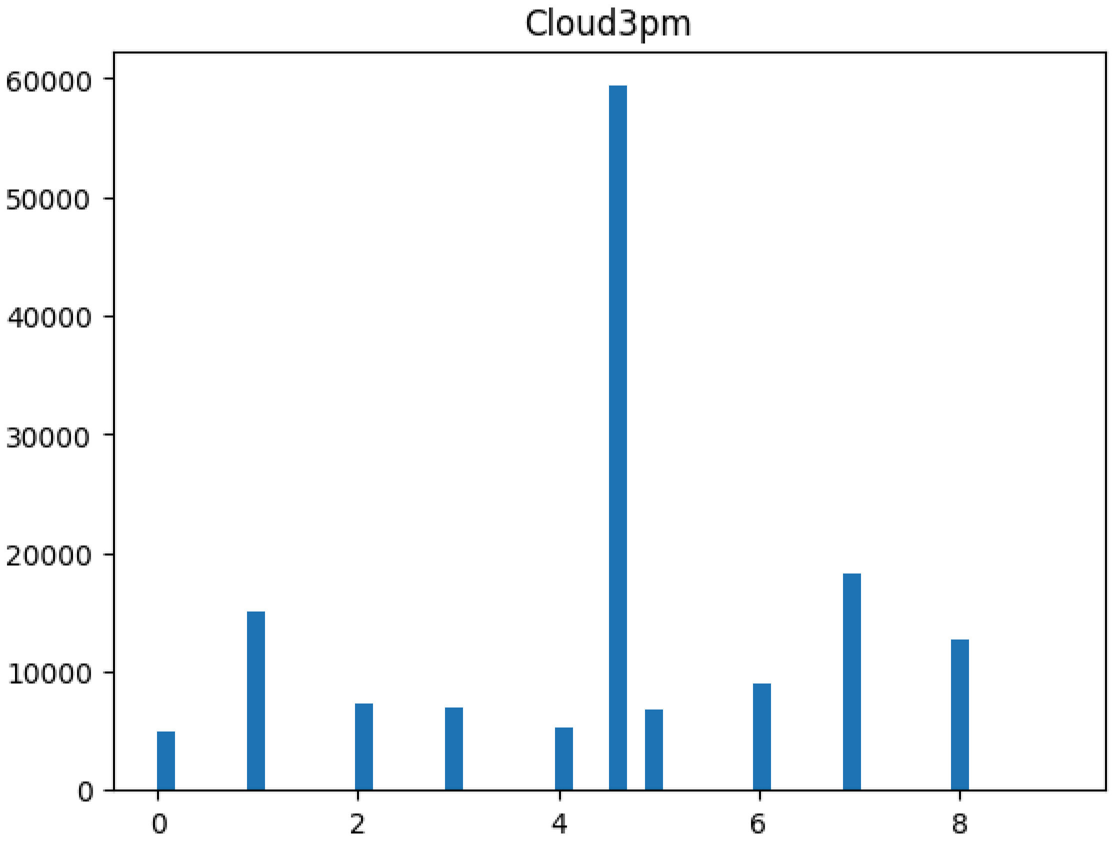

Humidity9am
0
2000
4000
6000
8000
0
20
40
60
80
100

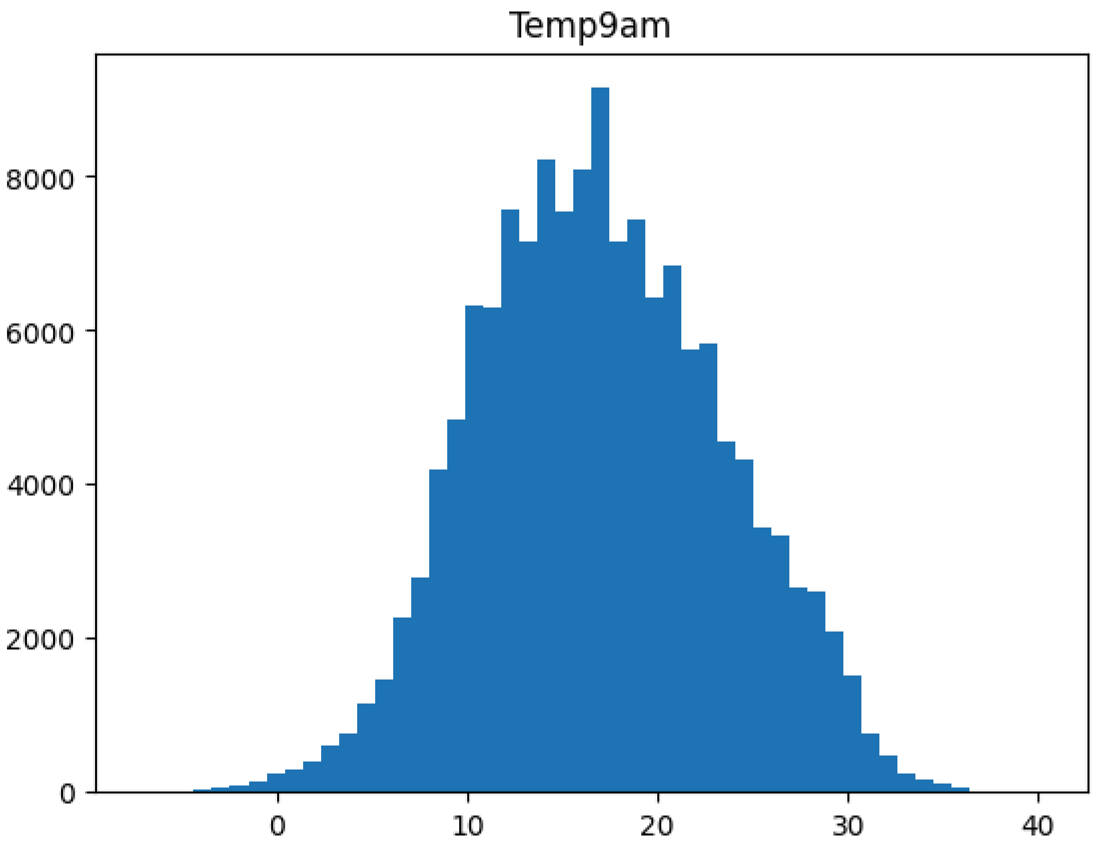

Humidity3pm
0
2000
4000
6000
8000
10000
0
20
40
60
80
100

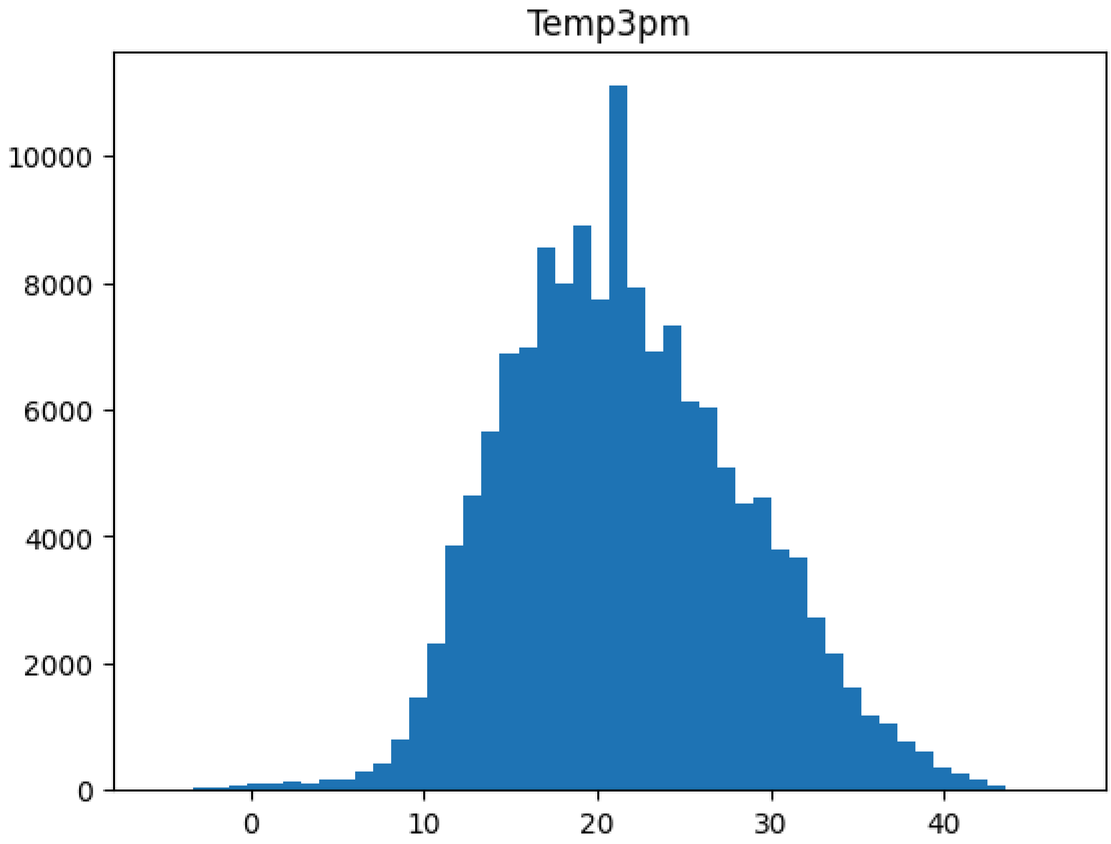

Pressure9am
25000
20000
15000
10000
5000
0
980
990
1000
1010
1020
1030
1040


Pressure3pm
25000
20000
15000
10000
5000
0
980
990
1000
1010
1020
1030
1040


Cloud9am
0
10000
20000
30000
40000
50000
0
2
4
6
8


Cloud3pm
0
10000
20000
30000
40000
50000
60000
0
2
4
6
8





Handling Outliers

In [188…

```
import pandas as pd

# Function to cap outliers
def cap_outliers(series, iqr_factor=1.5):
    Q1 = series.quantile(0.25)
    Q3 = series.quantile(0.75)
    IQR = Q3 - Q1
    lower_cap = Q1 - iqr_factor * IQR
    upper_cap = Q3 + iqr_factor * IQR
    return series.clip(lower=lower_cap, upper=upper_cap)

# Applying the function to each numeric column in the DataFrame
numeric_cols = df.select_dtypes(include=[np.number]).columns  # Adjust as necessary
for col in numeric_cols:
    df[col] = cap_outliers(df[col])

# Optional: Check results for one of the columns or summary stats
print(df.describe())  # Provides summary statistics to check the max and min, ensur
```

```
                                Date       Location        MinTemp  \
count                         145460  145460.000000  145460.000000   
mean   2013-04-04 21:08:51.907053568      24.560126      12.192336   
min              2007-11-01 00:00:00       1.000000      -5.950000   
25%              2011-01-11 00:00:00      12.000000       7.700000   
50%              2013-06-02 00:00:00      24.000000      12.000000   
75%              2015-06-14 00:00:00      37.000000      16.800000   
max              2017-06-25 00:00:00      49.000000      30.450000   
std                              NaN      13.941805       6.364499   

             MaxTemp       Rainfall    Evaporation       Sunshine  \
count  145460.000000  145460.000000  145460.000000  145460.000000   
mean       23.219758       0.381674       4.750932       7.922535   
min         2.700000       0.000000       2.200000       5.977944   
25%        18.000000       0.000000       4.000000       7.611178   
50%        22.600000       0.000000       4.800000       7.611178   
75%        28.200000       0.600000       5.200000       8.700000   
max        43.500000       1.500000       7.000000      10.333234   
std         7.067804       0.608638       1.454089       1.386787   

         WindGustDir  WindGustSpeed     WindDir9am  ...    Pressure3pm  \
count  145460.000000   145460.00000  145460.000000  ...  145460.000000   
mean        8.013028       39.64328       8.463151  ...    1015.268537   
min         1.000000        8.50000       1.000000  ...     998.650000   
25%         3.000000       31.00000       5.000000  ...    1011.100000   
50%         8.000000       39.00000       9.000000  ...    1015.200000   
75%        12.000000       46.00000      12.000000  ...    1019.400000   
max        16.000000       68.50000      16.000000  ...    1031.850000   
std         4.905515       12.17591       4.399079  ...       6.528909   

            Cloud9am       Cloud3pm        Temp9am        Temp3pm  RainToday  \
count  145460.000000  145460.000000  145460.000000  145460.000000   145460.0   
mean        4.447461       4.544125      16.988207      21.685669        0.0   
min         0.000000       1.000000      -1.500000       2.450000        0.0   
25%         3.000000       4.000000      12.300000      16.700000        0.0   
50%         4.447461       4.509930      16.700000      21.400000        0.0   
75%         6.000000       6.000000      21.500000      26.200000        0.0   
max         9.000000       9.000000      35.300000      40.450000        0.0   
std         2.265604       2.026092       6.440883       6.812734        0.0   

       RainTomorrow           Year          Month            Day  
count      145460.0  145460.000000  145460.000000  145460.000000  
mean            0.0    2012.769751       6.399615      15.712258  
min             0.0    2007.000000       1.000000       1.000000  
25%             0.0    2011.000000       3.000000       8.000000  
50%             0.0    2013.000000       6.000000      16.000000  
75%             0.0    2015.000000       9.000000      23.000000  
max             0.0    2017.000000      12.000000      31.000000  
std             0.0       2.537684       3.427262       8.794789  

[8 rows x 26 columns]
```

Feature Engineering

In [189…

```
df['TempChange9amTo3pm'] = df['Temp3pm'] - df['Temp9am']
df['HumidityChange9amTo3pm'] = df['Humidity3pm'] - df['Humidity9am']
df['PressureChange9amTo3pm'] = df['Pressure3pm'] - df['Pressure9am']
print(df[['TempChange9amTo3pm', 'HumidityChange9amTo3pm', 'PressureChange9amTo3pm']
```

```
   TempChange9amTo3pm  HumidityChange9amTo3pm  PressureChange9amTo3pm
0                 4.9                   -49.0                    -0.6
1                 7.1                   -19.0                    -2.8
2                 2.2                    -8.0                     1.1
3                 8.4                   -29.0                    -4.8
4                11.9                   -49.0                    -4.8
```

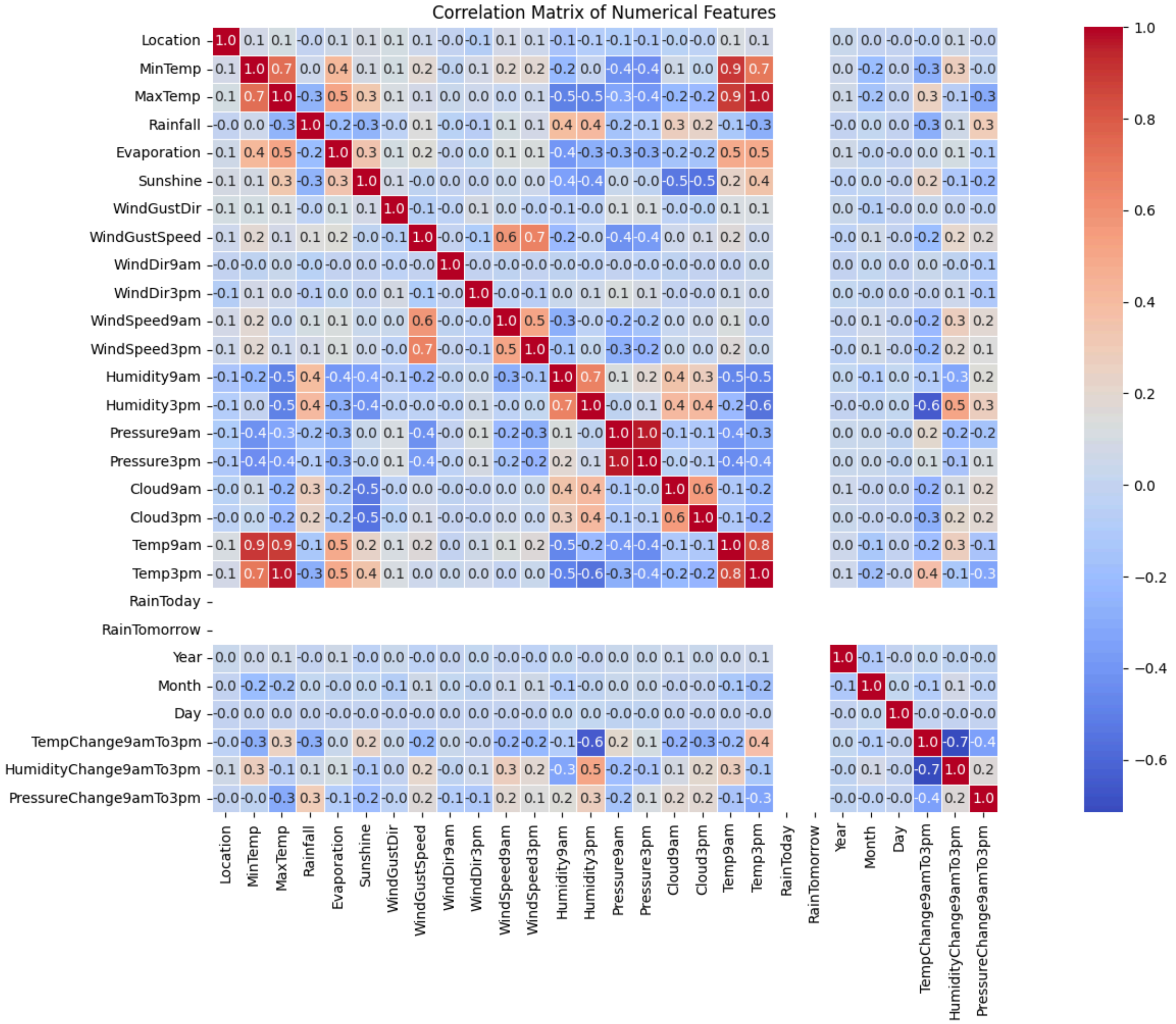
Model Development

In [190…

```
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
```

In [191…

```
numeric_df = df.select_dtypes(include=['number'])

# Calculate the correlation matrix
corr = numeric_df.corr()

# Set up the matplotlib figure
plt.figure(figsize=(22, 10))  # Adjust the size as needed

# Generate a heatmap
sns.heatmap(corr, annot=True, fmt=".1f", cmap='coolwarm',
            cbar=True, square=True, linewidths=.5)

# Add a title to the heatmap
plt.title('Correlation Matrix of Numerical Features')

# Show the plot
plt.show()
```



In [192…

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.ensemble import HistGradientBoostingClassifier
from sklearn.metrics import accuracy_score

# 2. Drop rows with missing values in RainTomorrow (target variable)
df.dropna(subset=["RainTomorrow"], inplace=True)

# 3. Separate features and target
X = df.drop(["RainTomorrow", "Date"], axis=1)  # Drop "Date" column
y = df["RainTomorrow"]

# 4. Encode categorical features
X = pd.get_dummies(X)

# 5. Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta

# 6. Impute missing values using SimpleImputer
imputer = SimpleImputer(strategy="mean")  # Replace missing values with the mean
X_train_imputed = imputer.fit_transform(X_train)
X_test_imputed = imputer.transform(X_test)
```

```
# 7. Train a model that supports missing values (e.g., HistGradientBoostingClassifi
model = HistGradientBoostingClassifier()
model.fit(X_train_imputed, y_train)

# 8. Make predictions and evaluate
y_pred = model.predict(X_test_imputed)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy*100)
```

```
Accuracy: 100.0
```

Parameter Tuning

In [193…

```
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
param_grid = {
    'max_depth': [3, 5, 7, 10],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
model = DecisionTreeClassifier()
grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5, scoring='a
grid_search.fit(X_train, y_train)
print("Best Parameters:", grid_search.best_params_)
print("Best Score:", grid_search.best_score_)
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Best Parameters: {'max_depth': 3, 'min_samples_leaf': 1, 'min_samples_split': 2}
Best Score: 1.0
Accuracy: 1.0
```

Model Evaluation

Performance Metrics

In [194…

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# ... (Load data, preprocess, train model as before) ...

# Train the model
model = DecisionTreeClassifier()
model.fit(X_train, y_train)

# Predict probabilities for ROC-AUC calculation
y_pred_proba = model.predict_proba(X_test_imputed)[:, 1]
```

```
# Calculate metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, pos_label='Yes')
recall = recall_score(y_test, y_pred, pos_label='Yes')
f1 = f1_score(y_test, y_pred, pos_label='Yes')
roc_auc = roc_auc_score(y_test, y_pred_proba)

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)
print("ROC-AUC:", roc_auc)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not
have valid feature names, but DecisionTreeClassifier was fitted with feature names
  warnings.warn(
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-194-d97cb17528a3> in <cell line: 13>()
     11
     12 # Predict probabilities for ROC-AUC calculation
---> 13 y_pred_proba = model.predict_proba(X_test_imputed)[:, 1]
     14
     15 # Calculate metrics

IndexError: index 1 is out of bounds for axis 1 with size 1
```

Validation Strategy

In [ ]:

```
import pandas as pd
from sklearn.model_selection import train_test_split, KFold, cross_val_score
from sklearn.impute import SimpleImputer
from sklearn.ensemble import HistGradientBoostingClassifier

# ... (Load data, preprocess, define model as before) ...

# 1. Create KFold object
kfold = KFold(n_splits=10, shuffle=True, random_state=42)  # 10 folds

# 2. Perform cross-validation
cv_scores = cross_val_score(model, X, y, cv=kfold, scoring='accuracy')

# 3. Print results
print("Cross-Validation Scores:", cv_scores)
print("Average Accuracy:", cv_scores.mean())
```

Insights and Recommendation

In [ ]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.ensemble import HistGradientBoostingClassifier
```

```
# ... (Load data, preprocess, train model as before) ...

# Get feature importances
importances = model.feature_importances_

# Get feature names
feature_names = X.columns

# Create a DataFrame with feature names and importances
feature_importance_df = pd.DataFrame({'Feature': feature_names, 'Importance': impor

# Sort by importance in descending order
feature_importance_df = feature_importance_df.sort_values(by='Importance', ascendin

# Print the DataFrame
print(feature_importance_df)
```

In [ ]: