main

May 10, 2023

1 Assignment

1.1 Problem

The objective of this project is develop a predictive classifier to predict the next-day rain on the target variable RainTomorrow

1.2 Group Members

S.No.	Name	Student ID
1	Hemang Sharma	24695785
2	Jyoti Khurana	14075648
3	Mahjabeen Mohiuddin	24610507
4	Manal Ydkw	24980447
5	Suyash Santosh Tapase	24678207

1.3 Library used

pandas

numpy

matplotlib

seaborn

plotly

sklearn

1.3.1 Link for DataSet & Source & Acknowledgements

Observations were drawn from numerous weather stations

The daily observations are available from http://www.bom.gov.au/climate/data

```
<li>Definitions adapted from <a href="http://www.bom.gov.au/climate/dwo/IDCJDW0000.shtml">http
```

Data source

http://www.bom.gov.au/climate/data

```
<li><a href="https://www.kaggle.com/datasets/arunavakrchakraborty/australia-weather-data">https
```

1.4 Importing packages

We will import all the required packages and define our dataset

[ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer, make_column_selector
from sklearn.preprocessing import LabelEncoder
from sklearn.utils import resample
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.metrics import confusion_matrix
```

1.5 Dataset

In this step we will describe the data in the dataset

This dataset contains about 10 years of daily weather observations from many locations across Australia.

1.5.1 Data Description

Location - Name of the city from Australia. . MinTemp - The Minimum temperature during a particular day. (degree Celsius) MaxTemp - The maximum temperature during a particular day. (degree Celsius) MeanTemp - The mean temperature during a particular day. (degree Celsius) Rainfall - Rainfall during a particular day. (millimeters) Evaporation - Evaporation during a particular day. (millimeters) Sunshine - Bright sunshine during a particular day. (hours) WindGusDir - The direction of the strongest gust during a particular day. (16 compass points) WindGuSpeed - Speed of strongest gust during a particular day. (kilometers per hour) WindDir9am - The direction of the wind for 10 min prior to 9 am. (compass points) WindDir3pm - The direction of the wind for 10 min prior to 3 pm. (compass points) WindSpeed9am - Speed of the wind for 10 min prior to 9 am. (kilometers per hour) WindSpeed3pm - Speed of the wind for 10 min prior to 3 pm. (kilometers per hour) Humidity9am - The humidity of the wind at 9 am. (percent) Humidity3pm - The humidity of the wind at 3 pm. (percent) AvgHumidity - The average of humidity of the wind. (percent) Pressure9am - Atmospheric pressure at 9 am. (hectopascals) Pressure3pm - Atmospheric pressure at 3 pm. (hectopascals) AvgPressure - The average Atmospheric pressure. (hectopascals) Cloud9am - Cloud-obscured portions of the sky at 9 am. (eighths) Cloud3pm - Cloud-obscured portions of the sky at 3 pm. (eighths) Temp9am - The temperature at 9 am. (degree Celsius) Temp3pm - The temperature at 3 pm. (degree Celsius) RainToday - If today is rainy then 'Yes'. If today is not rainy then 'No'. RainTomorrow - This is will be the variable containing value of "if tomorrow is rainy then 1 (Yes) or if tomorrow is not rainy then 0 (No)"

[ ]:
```
df_train = pd.read_csv('WeatherTrainingData.csv')
df_test = pd.read_csv('WeatherTestData.csv')
```

[ ]:
```
print(df_train.shape)
print(df_test.shape)
```

```
(99516, 26)
(42677, 25)
```

[ ]:
```
df_train
```

[ ]:

	row ID	Location	MinTemp	MaxTemp	MeanTemp	Rainfall	Evaporation	\
0	0	Albury	13.4	22.9	18.20	0.6	NaN	
1	1	Albury	7.4	25.1	16.30	0.0	NaN	
2	2	Albury	17.5	32.3	24.90	1.0	NaN	
3	3	Albury	14.6	29.7	22.20	0.2	NaN	
4	4	Albury	7.7	26.7	17.20	0.0	NaN	
…	…	…	…	…	…	…	…	
99511	99511	Uluru	8.0	20.7	14.35	0.0	NaN	
99512	99512	Uluru	3.5	21.8	12.70	0.0	NaN	
99513	99513	Uluru	2.8	23.4	13.10	0.0	NaN	
99514	99514	Uluru	3.6	25.3	14.50	0.0	NaN	
99515	99515	Uluru	5.4	26.9	16.20	0.0	NaN	

	Sunshine	WindGustDir	WindGustSpeed	…	AvgHumidity	Pressure9am	\
0	NaN	W	44.0	…	46.5	1007.7	
1	NaN	WNW	44.0	…	34.5	1010.6	
2	NaN	W	41.0	…	57.5	1010.8	
3	NaN	WNW	56.0	…	39.0	1009.2	
4	NaN	W	35.0	…	33.5	1013.4	
…	…	…	…	…	…	…	
99511	NaN	ESE	41.0	…	44.0	1028.1	
99512	NaN	E	31.0	…	43.0	1024.7	
99513	NaN	E	31.0	…	37.5	1024.6	
99514	NaN	NNW	22.0	…	38.5	1023.5	
99515	NaN	N	37.0	…	38.5	1021.0	

	Pressure3pm	AvgPressure	Cloud9am	Cloud3pm	Temp9am	Temp3pm	\
0	1007.1	1007.4	8.0	NaN	16.9	21.8	
1	1007.8	1009.2	NaN	NaN	17.2	24.3	
2	1006.0	1008.4	7.0	8.0	17.8	29.7	
3	1005.4	1007.3	NaN	NaN	20.6	28.9	
4	1010.1	1011.8	NaN	NaN	16.3	25.5	
…	…	…	…	…	…	…	
99511	1024.3	1026.2	NaN	7.0	11.6	20.0	
99512	1021.2	1023.0	NaN	NaN	9.4	20.9	
99513	1020.3	1022.5	NaN	NaN	10.1	22.4	
99514	1019.1	1021.3	NaN	NaN	10.9	24.5	
99515	1016.8	1018.9	NaN	NaN	12.5	26.1	

	RainToday	RainTomorrow

```
0              No              0
1              No              0
2              No              0
3              No              0
4              No              0
…           …              …
99511          No              0
99512          No              0
99513          No              0
99514          No              0
99515          No              0

[99516 rows x 26 columns]
```

1.6 Data Cleaning

Now in order to use this data, we need to clean the data and remove all the empty cells from the dataset. So we will use dropna( )

[ ]:
```
data_test=df_test
data_train=df_train
data_test['RainToday'] = data_test['RainToday'].map({'Yes': 1, 'No': 0})
```

[ ]:
```
data_test
```

[ ]:
```
       row ID Location  MinTemp  MaxTemp  MeanTemp  Rainfall  Evaporation  \
0           0   Albury     12.9     25.7     19.30       0.0          NaN
1           1   Albury      9.2     28.0     18.60       0.0          NaN
2           2   Albury     14.3     25.0     19.65       0.0          NaN
3           3   Albury      9.7     31.9     20.80       0.0          NaN
4           4   Albury     15.9     18.6     17.30      15.6          NaN
…       …       …       …       …        …        …          …
42672   42672    Uluru      2.4     19.1     10.80       0.0          NaN
42673   42673    Uluru      2.3     21.4     11.90       0.0          NaN
42674   42674    Uluru      2.6     22.5     12.60       0.0          NaN
42675   42675    Uluru      7.4     20.6     14.00       0.0          NaN
42676   42676    Uluru      7.8     27.0     17.40       0.0          NaN

       Sunshine WindGustDir  WindGustSpeed  … Humidity3pm AvgHumidity  \
0           NaN         WSW           46.0  …        30.0        34.0
1           NaN          NE           24.0  …        16.0        30.5
2           NaN           W           50.0  …        19.0        34.0
3           NaN         NNW           80.0  …         9.0        25.5
4           NaN           W           61.0  …        93.0        84.5
…         …          …             …  …          …         …
42672       NaN           E           33.0  …        24.0        41.5
42673       NaN          SE           22.0  …        28.0        44.0
42674       NaN           S           19.0  …        24.0        41.5
```

```
42675       NaN           E           35.0  …        33.0        48.0
42676       NaN          SE           28.0  …        24.0        37.5

       Pressure9am  Pressure3pm  AvgPressure  Cloud9am  Cloud3pm  Temp9am  \
0           1007.6       1008.7      1008.20       NaN       2.0     21.0
1           1017.6       1012.8      1015.20       NaN       NaN     18.1
2           1009.6       1008.2      1008.90       1.0       NaN     18.1
3           1008.9       1003.6      1006.30       NaN       NaN     18.3
4            994.3        993.0       993.65       8.0       8.0     17.4
…              …            …           …          …         …       …
42672       1030.0       1026.2      1028.10       NaN       NaN      8.0
42673       1026.9       1022.8      1024.90       NaN       NaN      8.9
42674       1025.0       1021.4      1023.20       NaN       NaN      8.8
42675       1027.2       1023.3      1025.30       NaN       NaN     11.0
42676       1019.4       1016.5      1018.00       3.0       2.0     15.1

       Temp3pm  RainToday
0         23.2        0.0
1         26.5        0.0
2         24.6        0.0
3         30.2        0.0
4         15.8        1.0
…          …          …
42672     18.8        0.0
42673     20.3        0.0
42674     22.1        0.0
42675     20.3        0.0
42676     26.0        0.0

[42677 rows x 25 columns]
```

[ ]:
```
data_test.drop(columns=['Sunshine', 'Evaporation'], inplace=True)
categorical = data_test.select_dtypes(include = "object").columns
cleaner = ColumnTransformer([
    ('categorical_transformer', SimpleImputer(strategy='most_frequent'),␣
 ↪categorical)
])
data_test[categorical] = cleaner.fit_transform(data_test[categorical])
null_columns=data_test.columns[data_test.isnull().any()]
data_test[null_columns].isnull().sum()
```

[ ]:
```
MinTemp           194
MaxTemp            92
Rainfall          427
WindGustSpeed    2790
WindSpeed9am      413
WindSpeed3pm      795
```

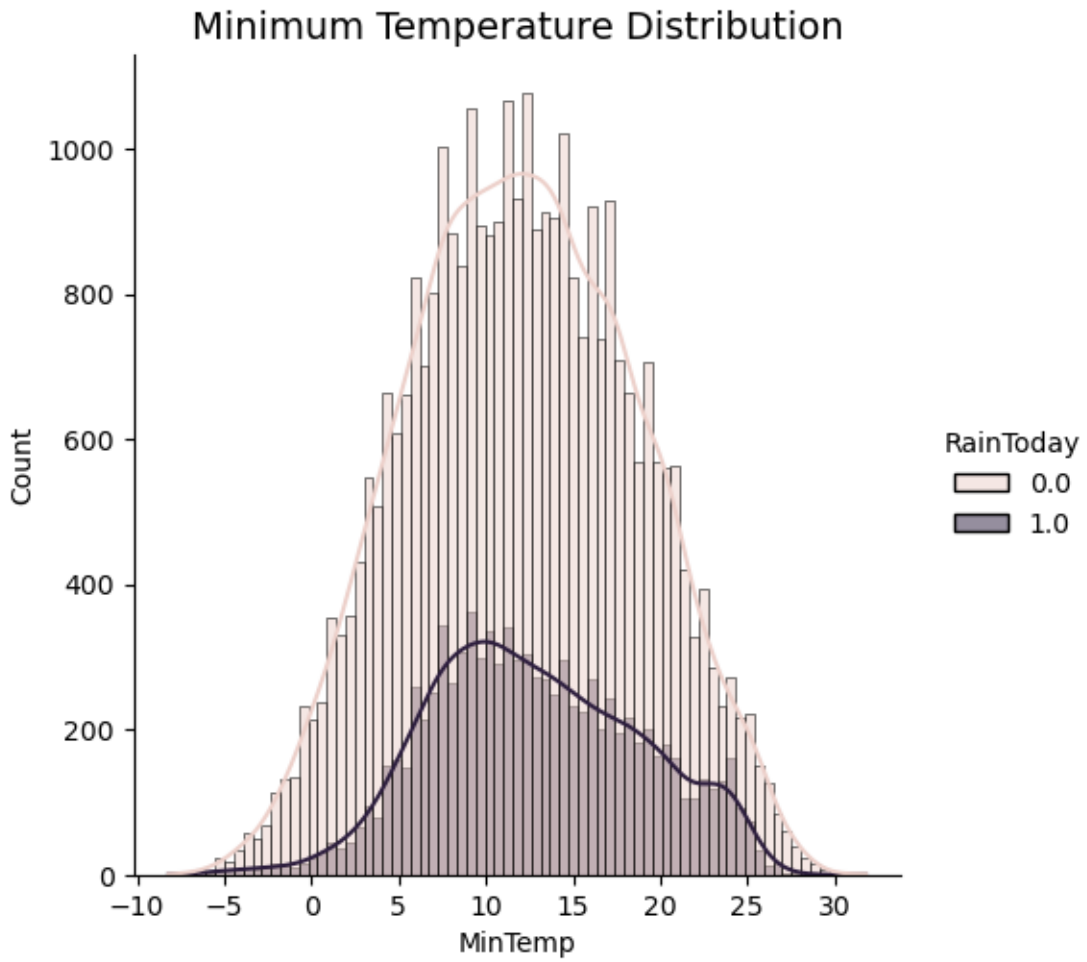
```
Humidity9am        541
Humidity3pm       1104
Pressure9am       4266
Pressure3pm       4245
Cloud9am         16085
Cloud3pm         17092
Temp9am            290
Temp3pm            822
RainToday          427
dtype: int64
```

2 Data Analysis

Now we will plot graphs comparing diffrent characteristics of our dataset

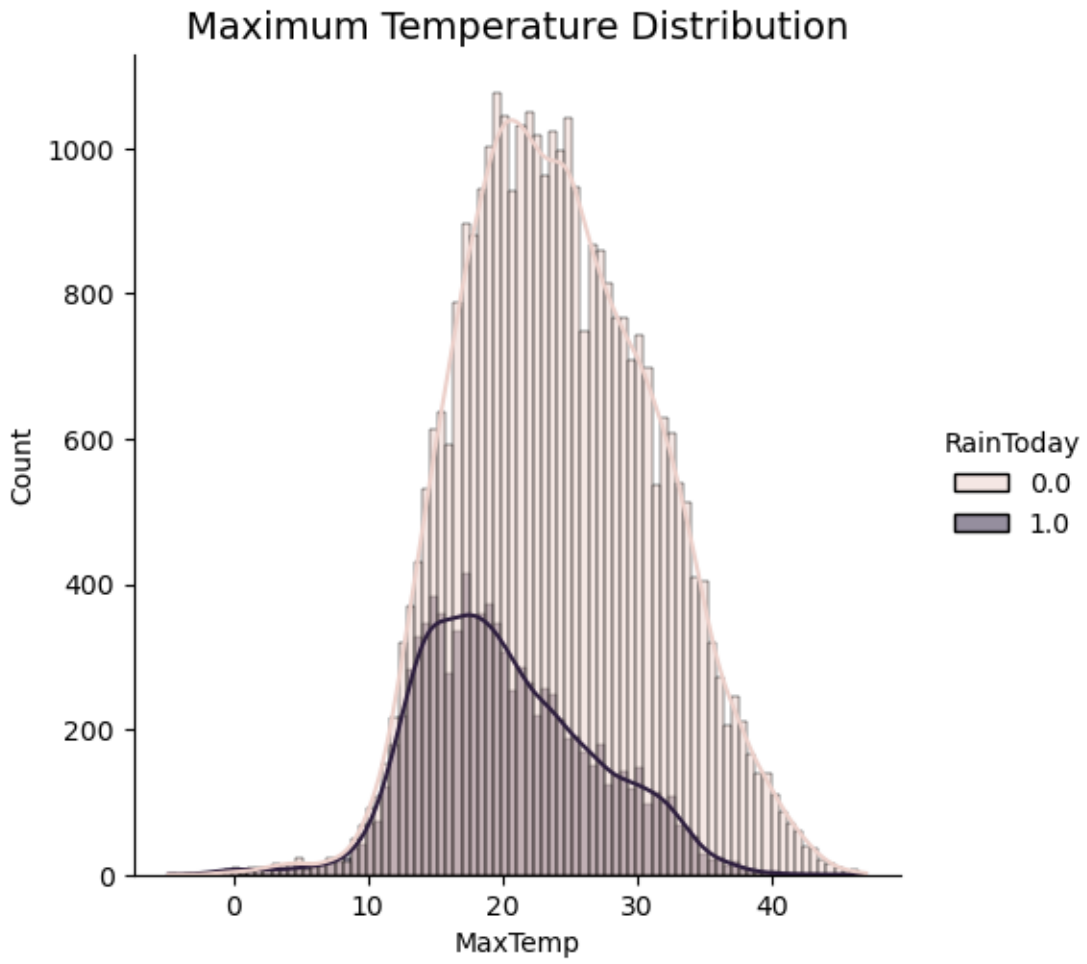
2.1 1. Feature Distribution

[ ]:
```
sns.displot(data_test, x="MinTemp", hue='RainToday', kde=True)
plt.title("Minimum Temperature Distribution", fontsize = 14)
plt.show()
```



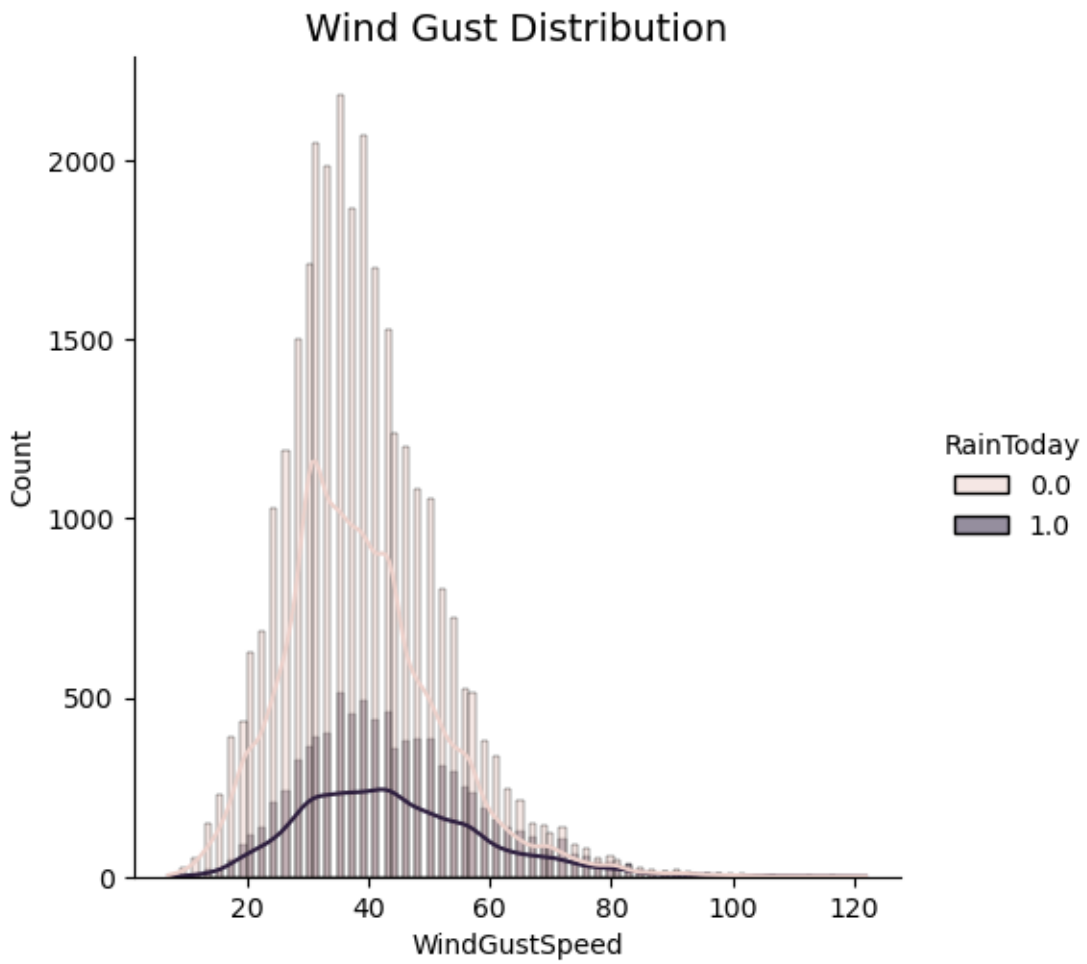
The analysis revealed that the minimum temperature range from -8.5 ℃ to 33.9 ℃ and the minimum temperature of 11 ℃ had the highest frequency in the data set.

[ ]:
```
sns.displot(data_test, x="MaxTemp", hue='RainToday', kde=True)
plt.title("Maximum Temperature Distribution", fontsize = 14)
plt.show()
```



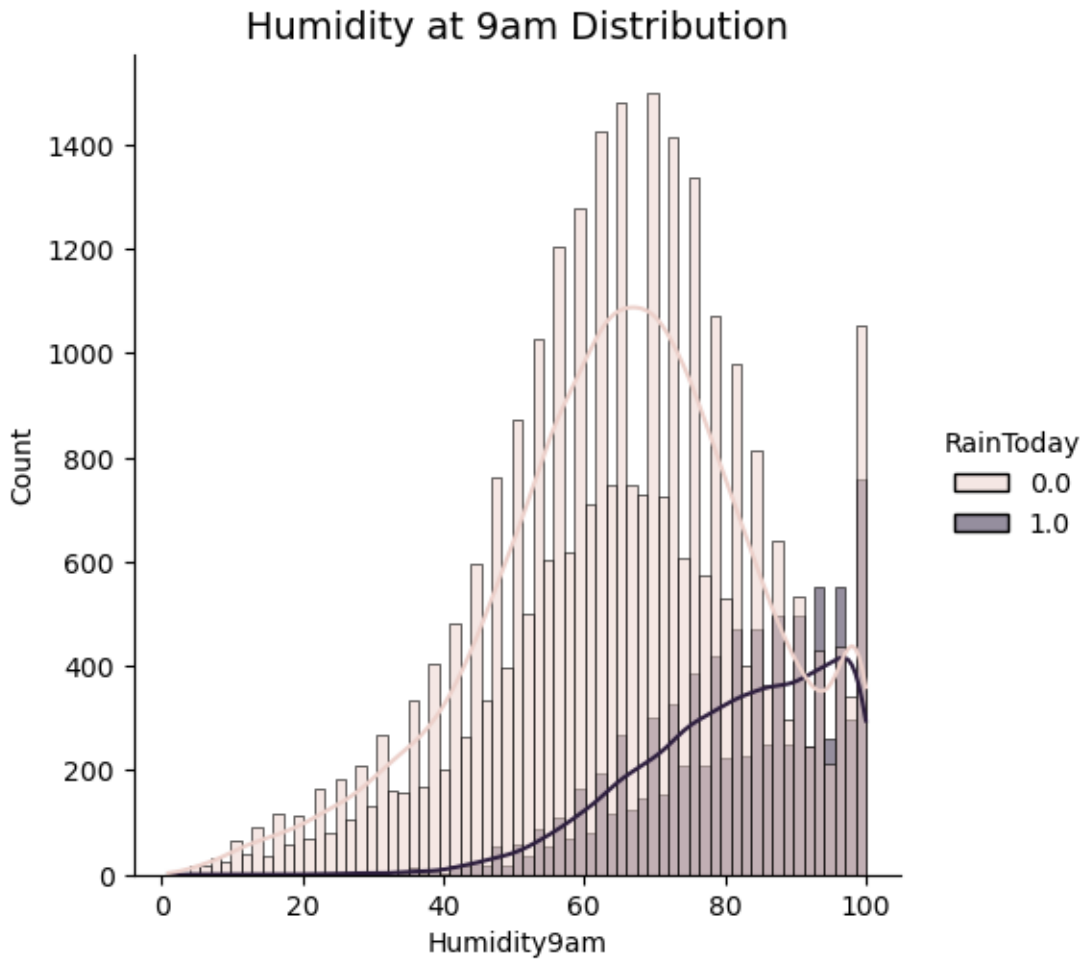
On the other hand, the maximum temperature range from -4.1 ℃ to 48.1 ℃ and the maximum temperature of 20 ℃ has the highest frequency in the data set.

[ ]:
```
sns.displot(data_test, x="WindGustSpeed", hue='RainToday', kde=True)
plt.title("Wind Gust Distribution", fontsize = 14)
plt.show()
```



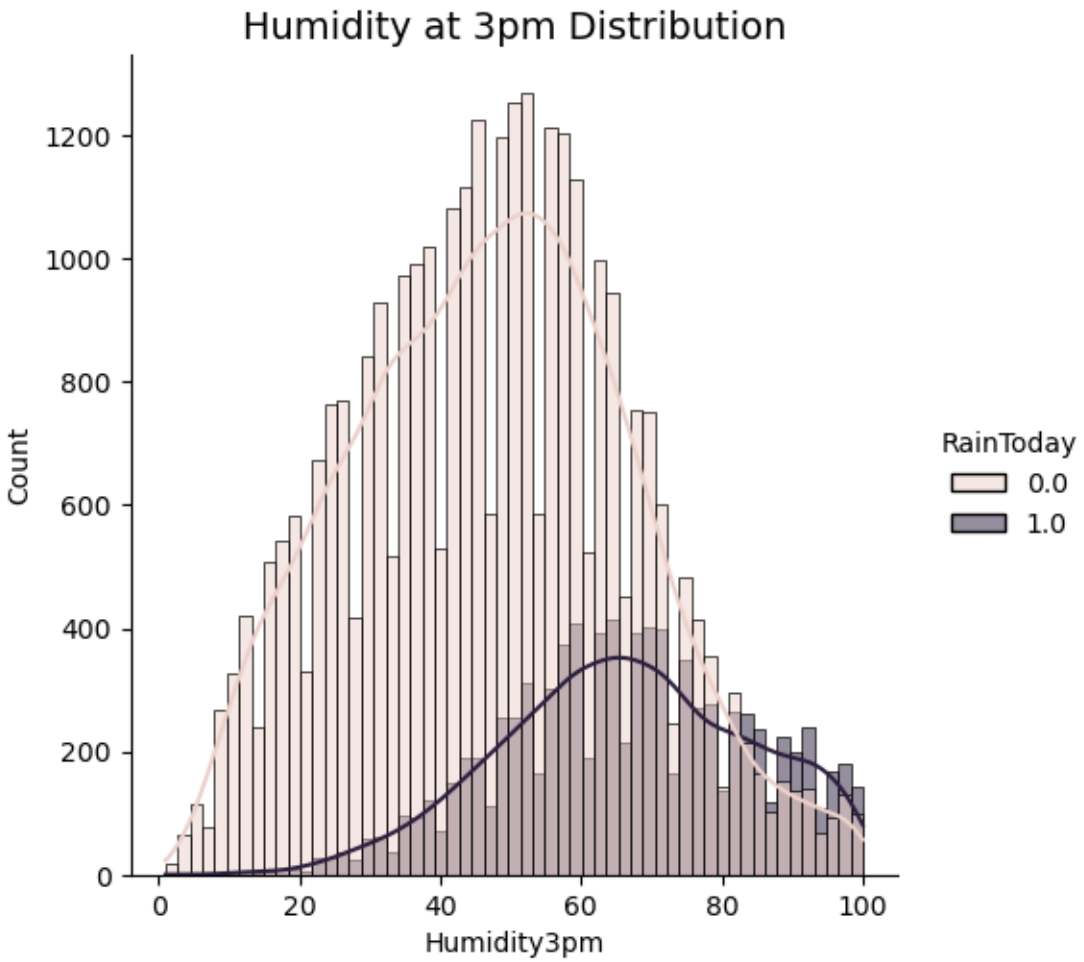
During the analysis, it was found that the range of gusts was from 6 main points to 135 main points and 39.98 main points of gusts had the highest frequency in the data set.

[ ]:
```
sns.displot(data_test, x="Humidity9am", hue='RainToday', kde=True)
plt.title("Humidity at 9am Distribution", fontsize = 14)
plt.show()
```



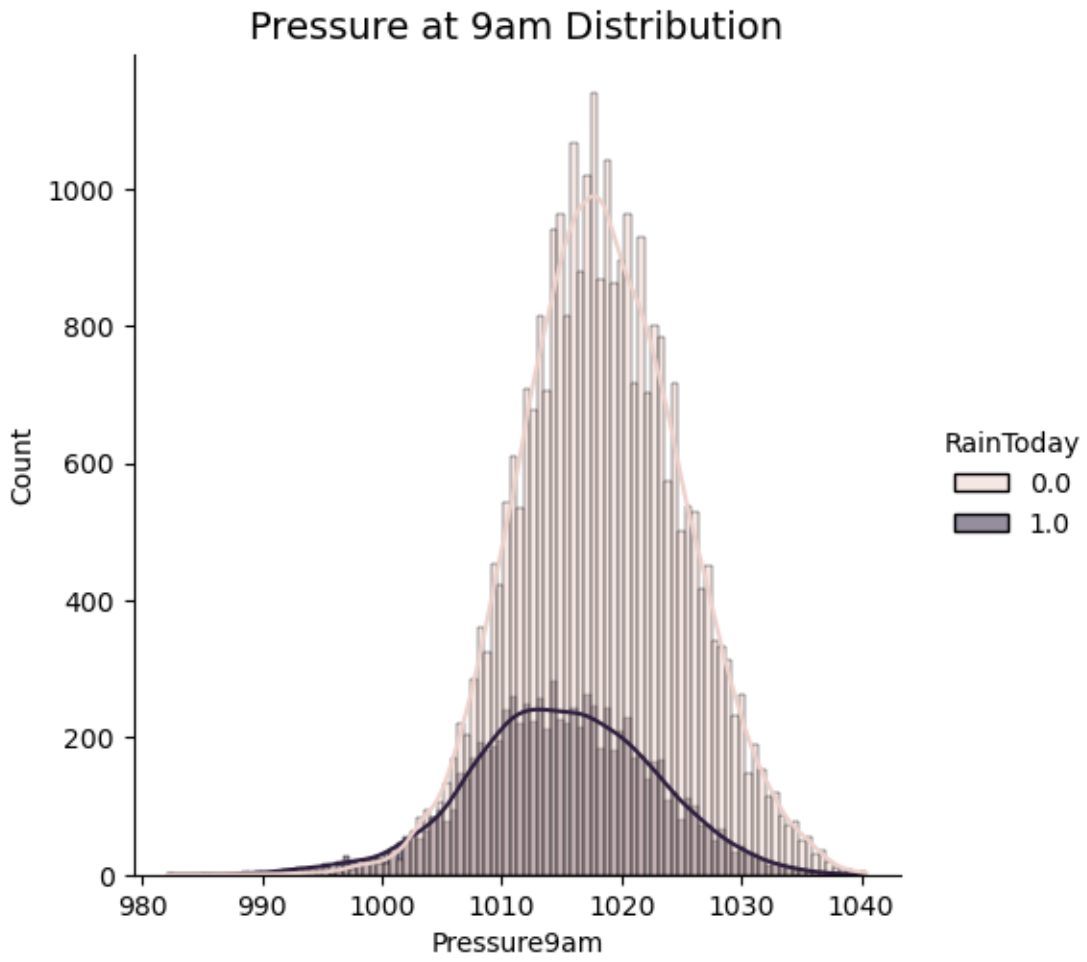
During the analysis, it was found that the range of air humidity at 9 o'clock in the morning. and at 3:00 p.m. from 0% to 100% and 99% humidity at 9:00 am. has the highest frequency in the data set.

[ ]:
```
sns.displot(data_test, x="Humidity3pm", hue='RainToday', kde=True)
plt.title("Humidity at 3pm Distribution", fontsize = 14)
plt.show()
```



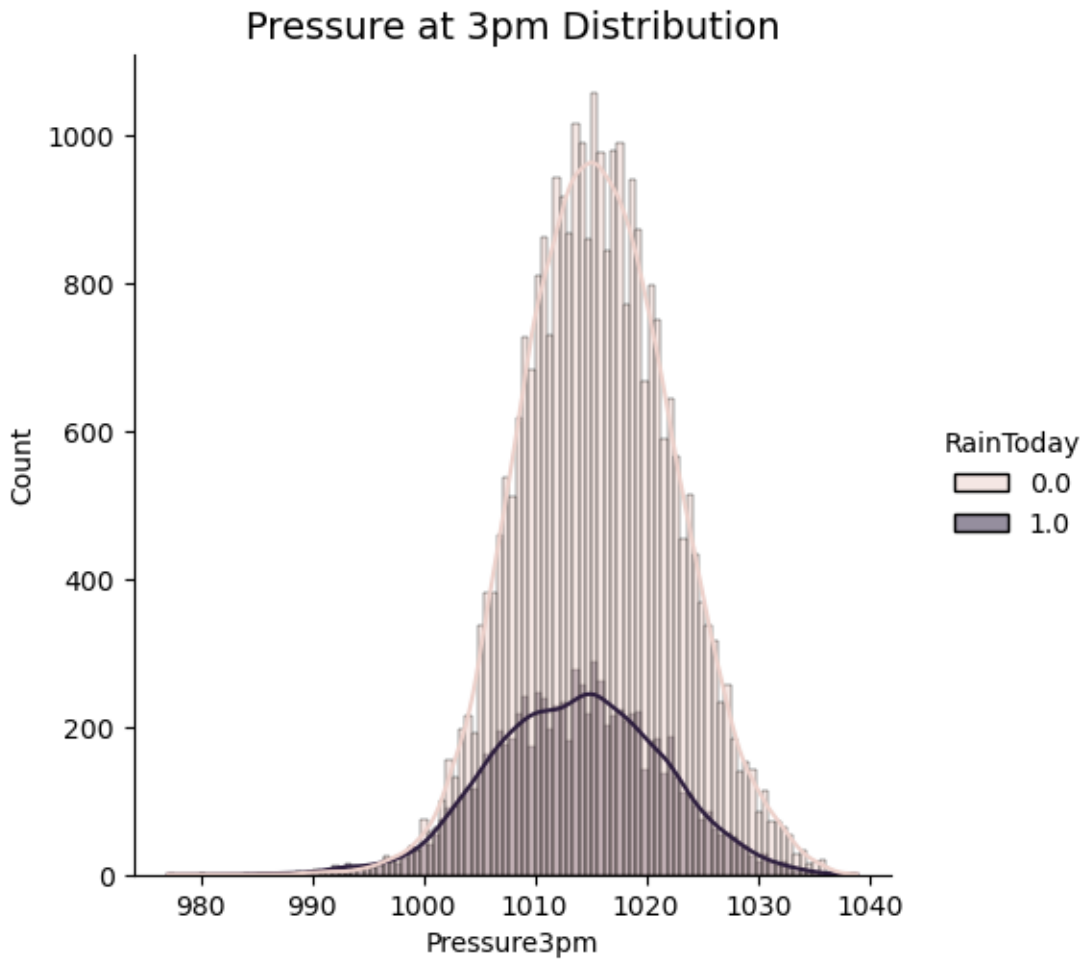
On the other hand, 54.43% of humidity at 3 pm has the highest frequency in the dataset.

[ ]:
```
sns.displot(data_test, x="Pressure9am", hue='RainToday', kde=True)
plt.title("Pressure at 9am Distribution", fontsize = 14)
plt.show()
```



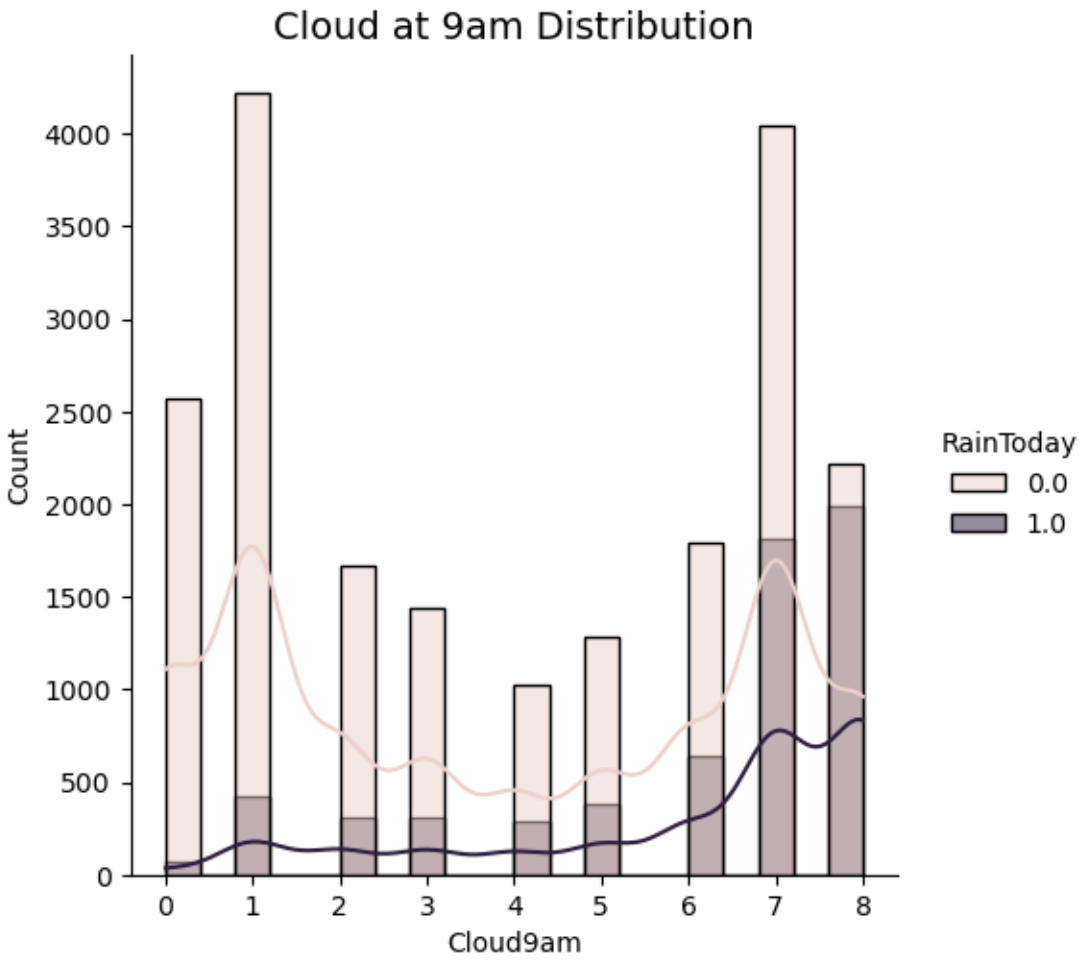
During the analysis, it was found that the range of wind pressure at 9 am. ranges from 980.5 hPa to 1042 hPa, and the pressure of 1017.68 hPa has the highest frequency in the data set.

[ ]:
```
sns.displot(data_test, x="Pressure3pm", hue='RainToday', kde=True)
plt.title("Pressure at 3pm Distribution", fontsize = 14)
plt.show()
```



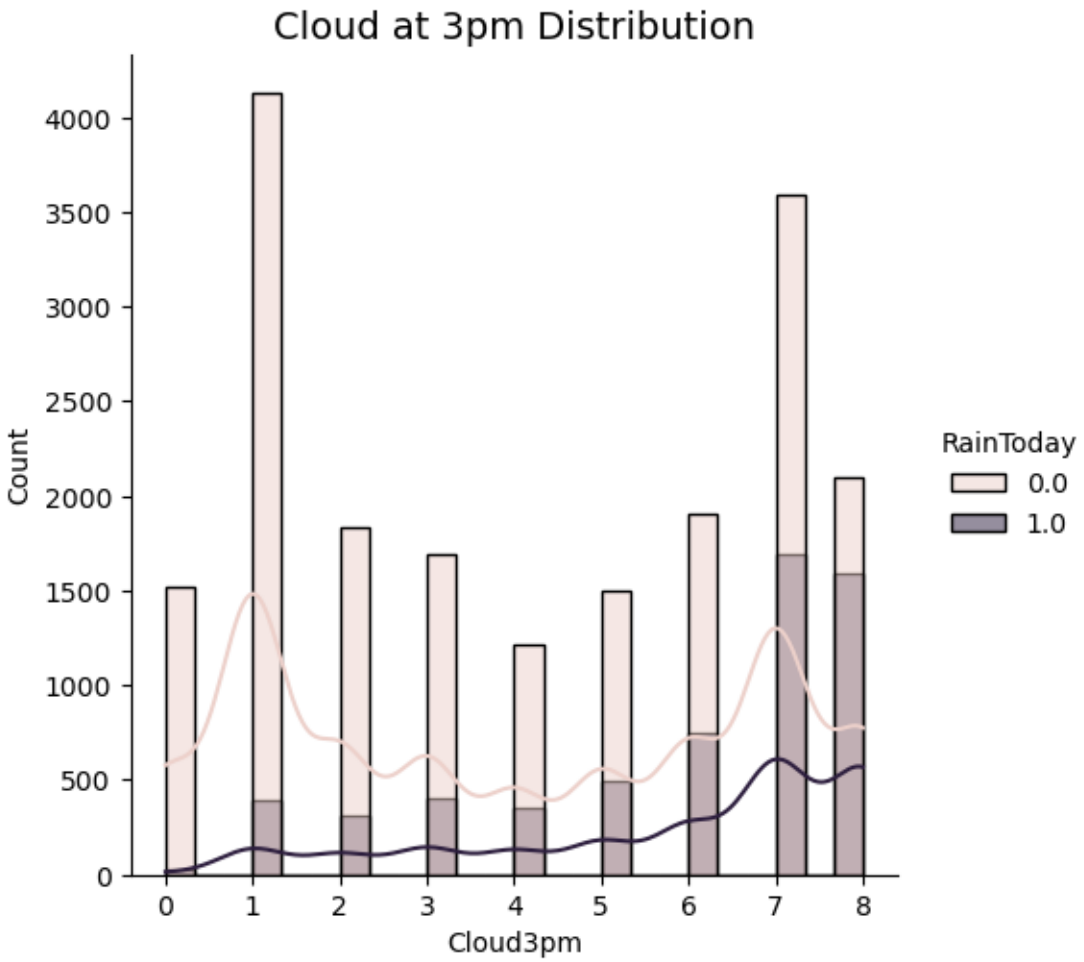
On the opposite hand, the variety of strain at three pm is from 978.2 hPa to 1039.6 hPa and 1015.28 hPa of strain has the very best frequency withinside the dataset.

[ ]:
```
sns.displot(data_test, x="Cloud9am", hue='RainToday', kde=True)
plt.title("Cloud at 9am Distribution", fontsize = 14)
plt.show()
```



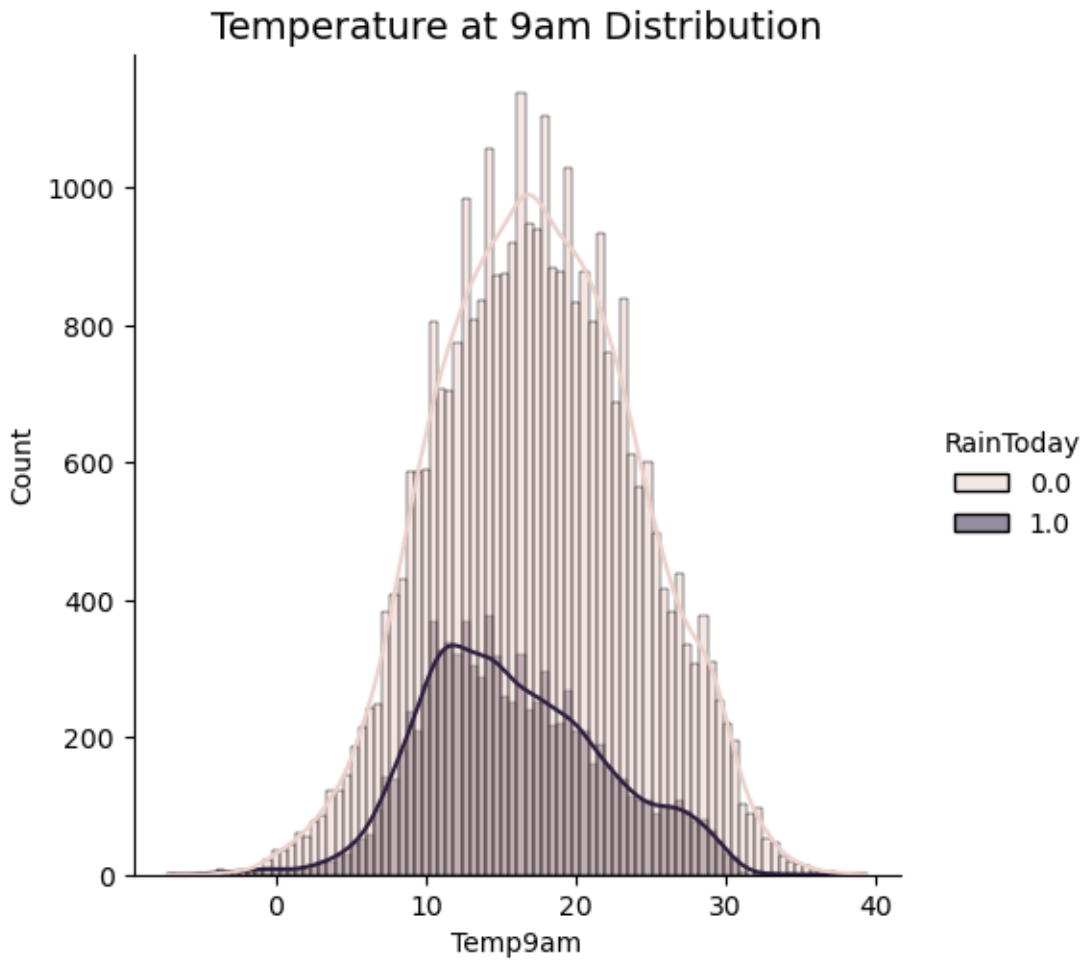
During the analysis, it's been determined that the variety of cloud at 9 am and 3 pm is from zero eighths to nine eighths and 4.44 eighths of cloud at nine am has the best frequency withinside the dataset.

[ ]:
```
sns.displot(data_test, x="Cloud3pm", hue='RainToday', kde=True)
plt.title("Cloud at 3pm Distribution", fontsize = 14)
plt.show()
```



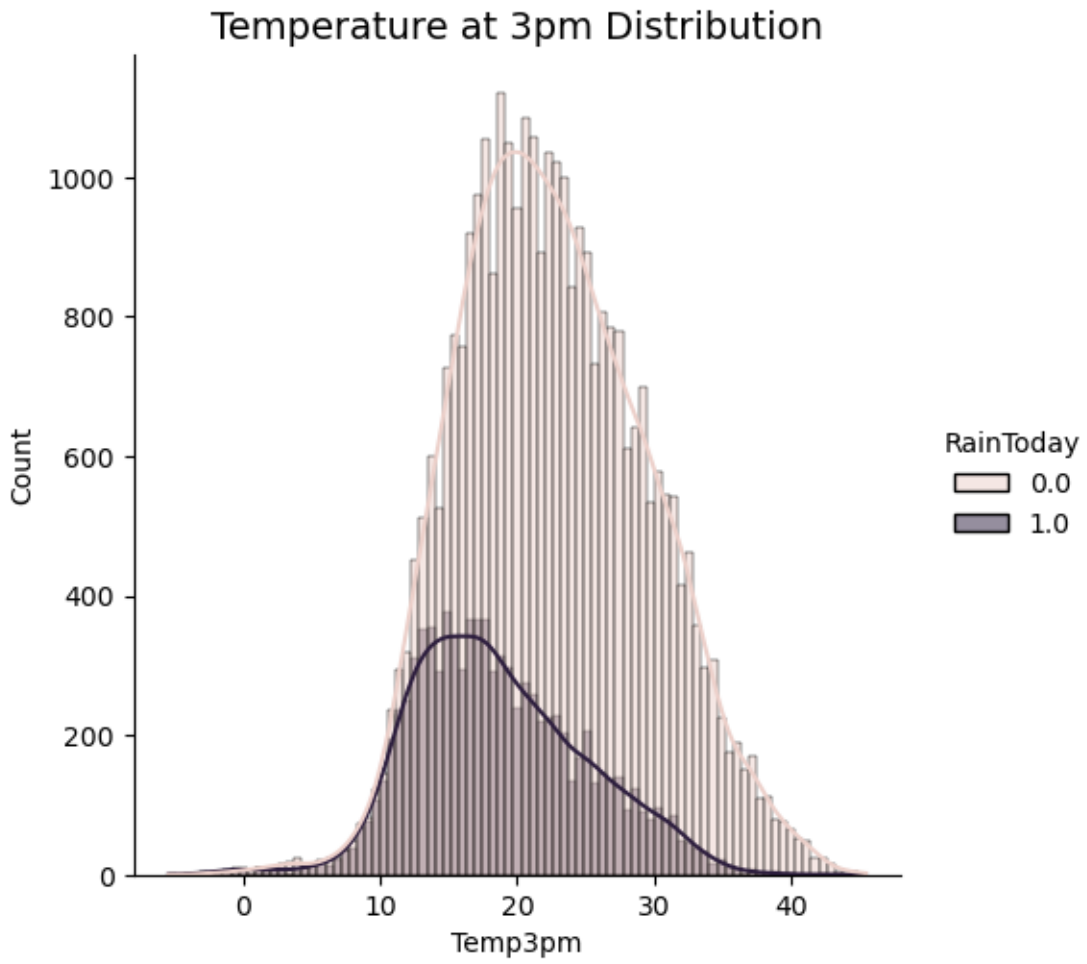
On the other hand, 4.52 eighths of cloud at 3 pm has the highest frequency in the dataset.

[ ]:
```
sns.displot(data_test, x="Temp9am", hue='RainToday', kde=True)
plt.title("Temperature at 9am Distribution", fontsize = 14)
plt.show()
```



During the analysis, it has been found that the range of wind temperature at 9 am is from -7 ℃ to 40.2 ℃ and 17 ℃ of temperature has the highest frequency in the dataset.

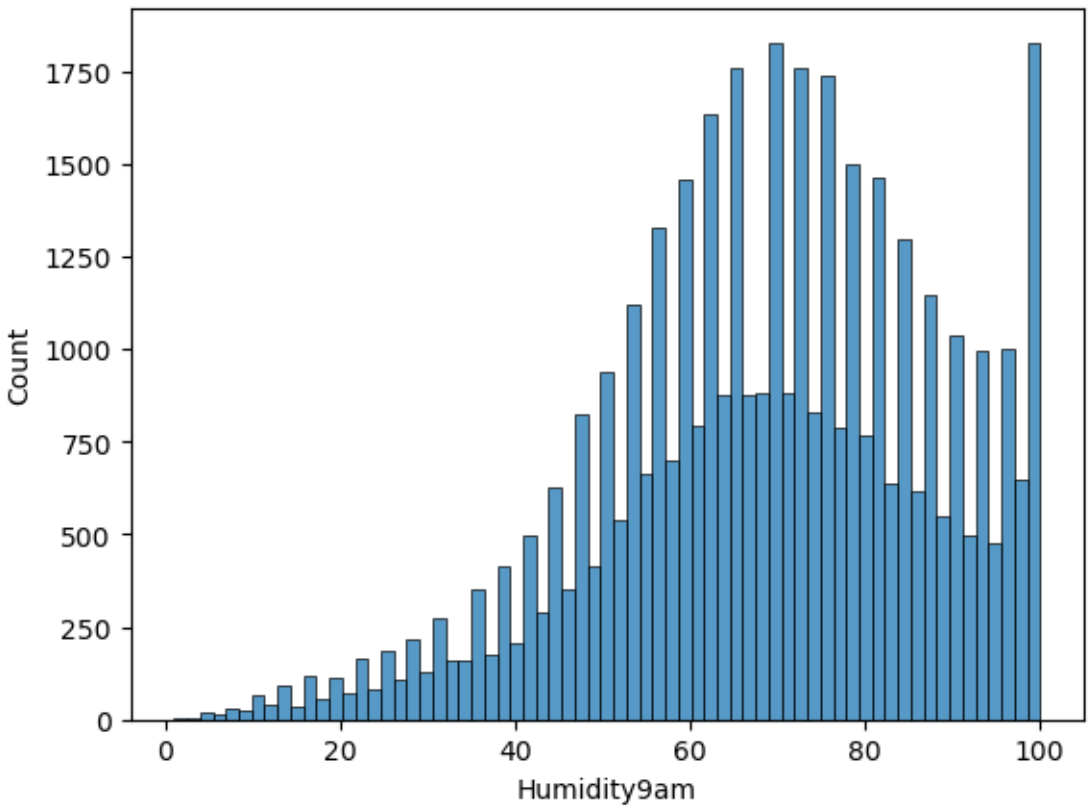
[ ]:
```
sns.displot(data_test, x="Temp3pm", hue='RainToday', kde=True)
plt.title("Temperature at 3pm Distribution", fontsize = 14)
plt.show()
```



On the other hand, the range of pressure at 3 pm is from -5.1 ℃ to 46.7 ℃ and 27.68 ℃ of temperature has the highest frequency in the dataset.

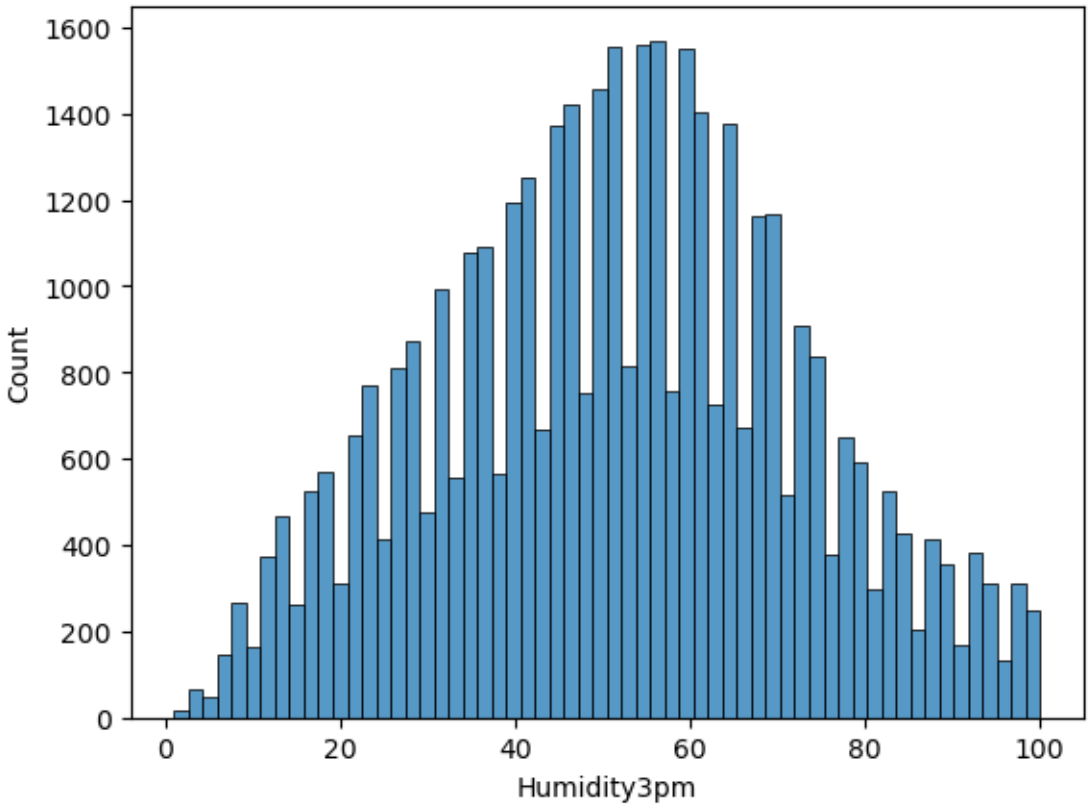
[ ]:
```
df=data_test
sns.histplot(df['Humidity9am'])
```

[ ]:
```
<Axes: xlabel='Humidity9am', ylabel='Count'>
```



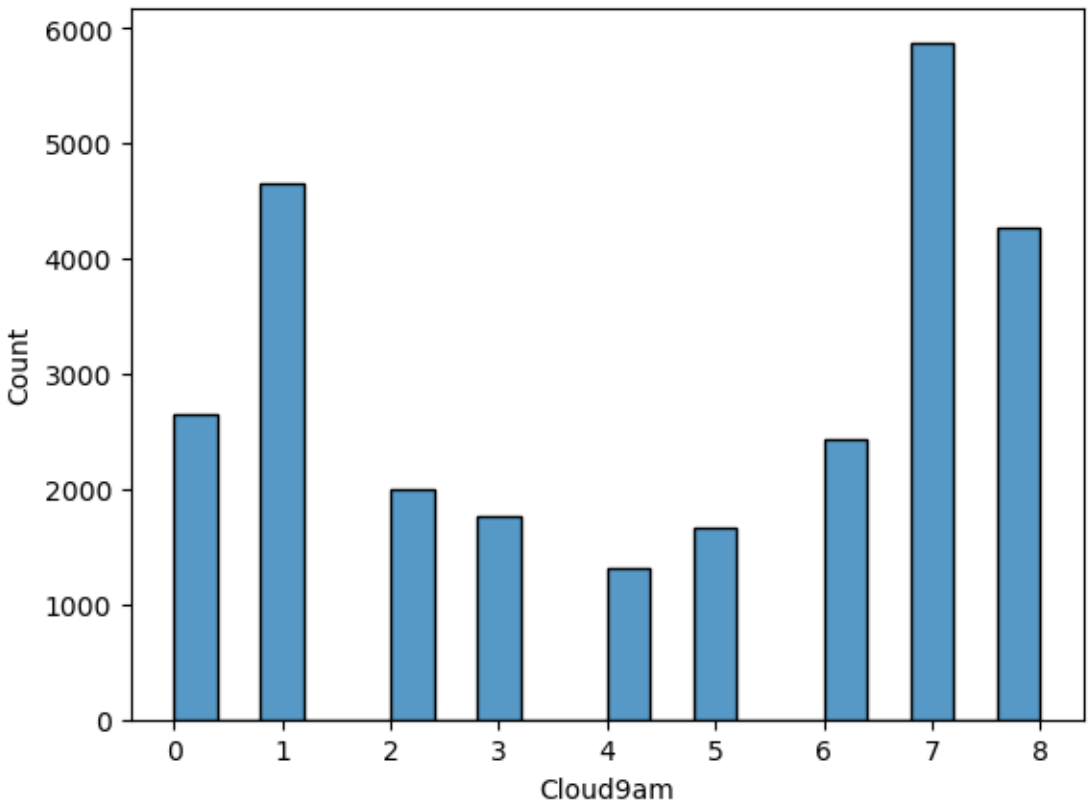
[ ]:
```
sns.histplot(df['Humidity3pm'])
```

[ ]:
```
<Axes: xlabel='Humidity3pm', ylabel='Count'>
```



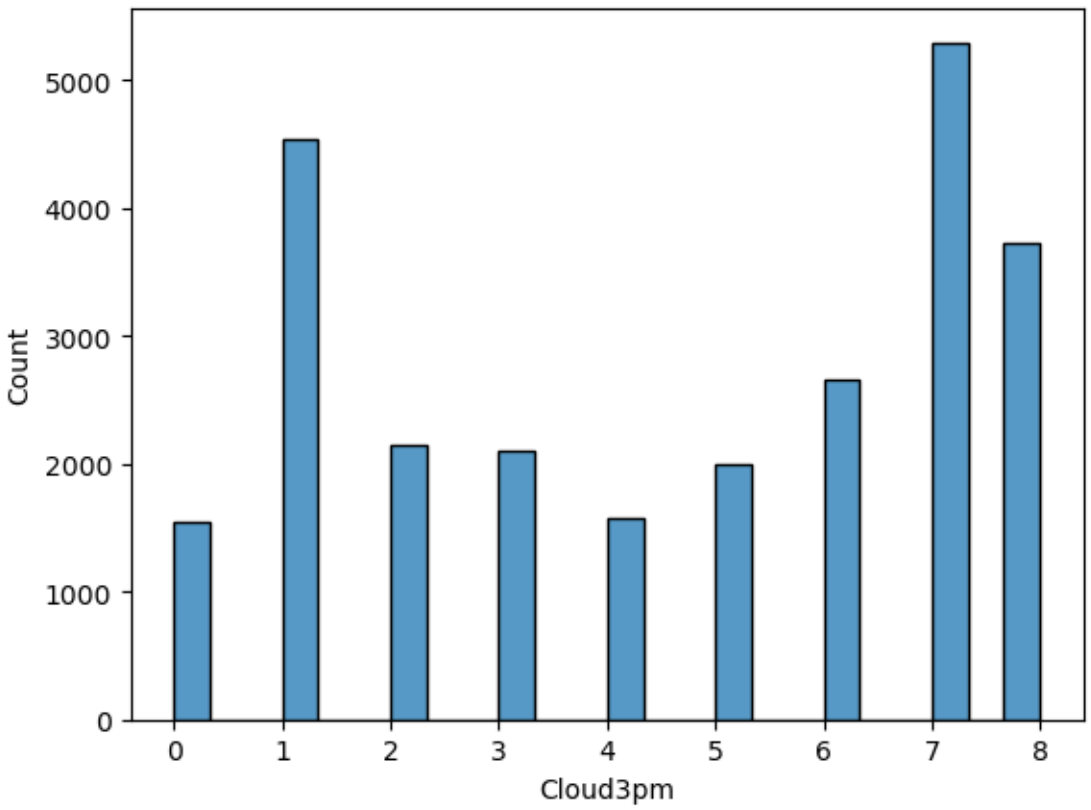
[ ]:
```
sns.histplot(df['Cloud9am'])
```

[ ]:
```
<Axes: xlabel='Cloud9am', ylabel='Count'>
```



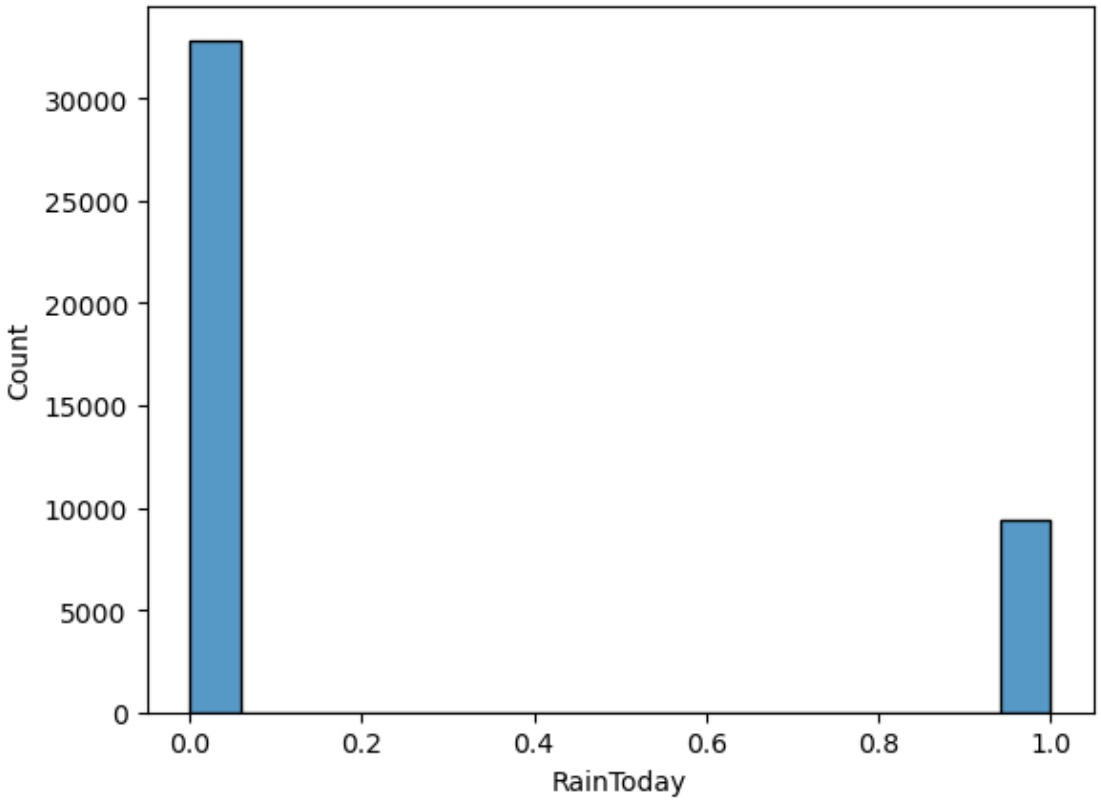
[ ]:
```
sns.histplot(df['Cloud3pm'])
```

[ ]:
```
<Axes: xlabel='Cloud3pm', ylabel='Count'>
```

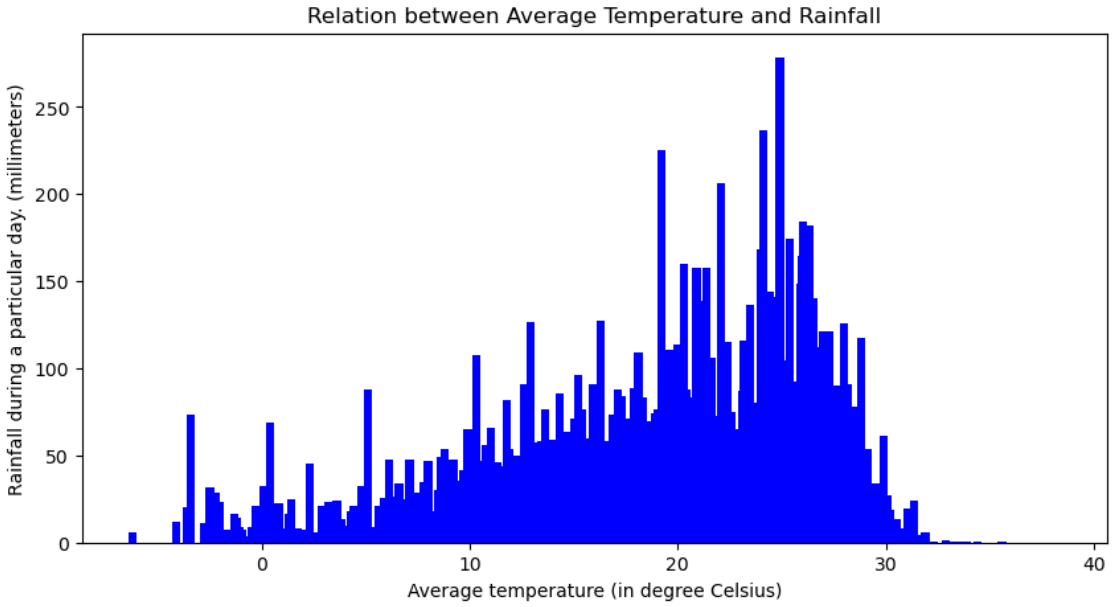


[ ]:
```
sns.histplot(df['RainToday'])
```

[ ]:
```
<Axes: xlabel='RainToday', ylabel='Count'>
```



[ ]:
```
x = list(data_test.MeanTemp)
y = list(data_test.Rainfall)

fig = plt.figure(figsize = (10, 5))

# creating the bar plot
plt.bar(x, y, color ='blue',
        width = 0.4)

plt.xlabel("Average temperature (in degree Celsius)")
plt.ylabel("Rainfall during a particular day. (millimeters)")
plt.title("Relation between Average Temperature and Rainfall")
plt.show()
```



[ ]:
```
import seaborn as sns
import plotly.express as px

figure = px.scatter(data_frame = data_test, x="AvgHumidity",
                    y="MeanTemp", size="AvgHumidity",
                    trendline="ols",
                    labels={
                     "AvgHumidity": "Humidity (in percent)",
                     "MeanTemp": "Mean Temperature (in degree Celsius)"
                },
                    title = "Relationship Between Temperature and Humidity")
figure.show()
```

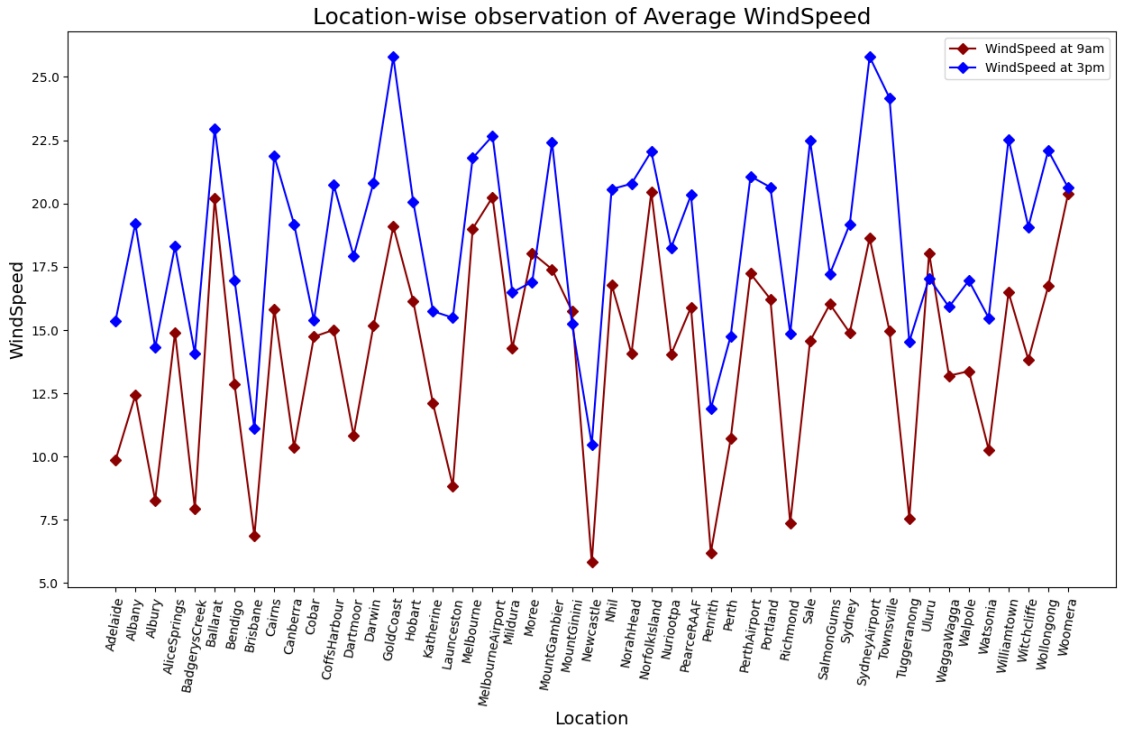
2.2 2. Average WindSpeed Analysis

[ ]:
```
windspeed_weather_df = data_test.groupby(['Location'])[['WindSpeed9am',␣
↪'WindSpeed3pm']].mean()
windspeed_weather_df = windspeed_weather_df.reset_index()
windspeed_weather_df.head()
```

[ ]:

	Location	WindSpeed9am	WindSpeed3pm
0	Adelaide	9.849616	15.354555
1	Albany	12.418605	19.203297
2	Albury	8.274194	14.317552
3	AliceSprings	14.890231	18.300768
4	BadgerysCreek	7.952273	14.075964

```
[ ]: x = windspeed_weather_df.loc[:, 'Location']
     y1 = windspeed_weather_df['WindSpeed9am']
     y2 = windspeed_weather_df['WindSpeed3pm']

     plt.figure(figsize = (15, 8))

     plt.plot(x, y1, marker='D', color = 'darkred', label = 'WindSpeed at 9am')
     plt.plot(x, y2, marker='D', color = 'blue', label = 'WindSpeed at 3pm')

     plt.xlabel('Location', fontsize = 14)
     plt.ylabel('WindSpeed', fontsize = 14)
     plt.title('Location-wise observation of Average WindSpeed', fontsize = 18)
     plt.legend(fontsize = 10, loc = 'best')
     plt.xticks(rotation=80)
     plt.show()
```



From this analysis, the wind speed at Melbourne Airport was determined to be the highest at 9:00 AM. with a speed of 20.29 km/h. On the other hand, at 3 o'clock in the afternoon. The highest wind speed is on the Gold Coast of Australia with 25.77 km/h. It can be concluded that the wind speed at 15:00. it is much higher than the wind speed at 9 o'clock in the morning.

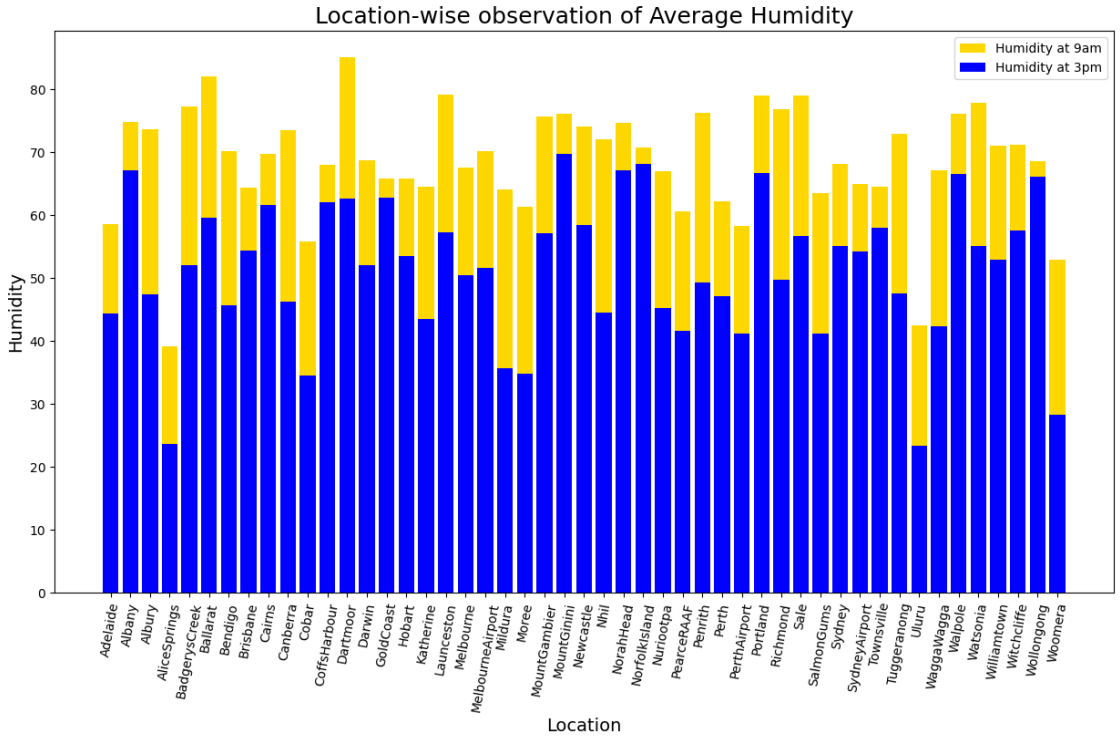
2.3 3. Average Humidity Analysis

[ ]:
```
humidity_weather_df = data_test.groupby(['Location'])[['Humidity9am',␣
↪'Humidity3pm']].mean()
humidity_weather_df = humidity_weather_df.reset_index()
humidity_weather_df.head()
```

[ ]:

	Location	Humidity9am	Humidity3pm
0	Adelaide	58.539560	44.398463
1	Albany	74.787592	67.116848
2	Albury	73.603926	47.346774
3	AliceSprings	39.140351	23.670692
4	BadgerysCreek	77.174603	52.029545

[ ]:
```
x = humidity_weather_df.loc[:, 'Location']
y1 = humidity_weather_df['Humidity9am']
y2 = humidity_weather_df['Humidity3pm']

plt.figure(figsize = (15, 8))

plt.bar(x, y1, color = 'gold', label = 'Humidity at 9am')
plt.bar(x, y2, color = 'blue',label = 'Humidity at 3pm')

plt.xlabel('Location', fontsize = 14)
plt.ylabel('Humidity', fontsize = 14)
plt.title('Location-wise observation of Average Humidity', fontsize = 18)
plt.legend(fontsize = 10, loc = 'best')
plt.xticks(rotation=80)
plt.show()
```



From this analysis it was found that the humidity of Dartmoor was highest at 9 am. 84.38%. On the other hand, at 3:00 p.m., Australia's Mount Ginnie has the highest humidity at 68.24%. In conclusion, it can be concluded that the humidity at 9 o'clock is much higher than the wind speed at 3 o'clock.

2.4 4. Average Pressure Analysis

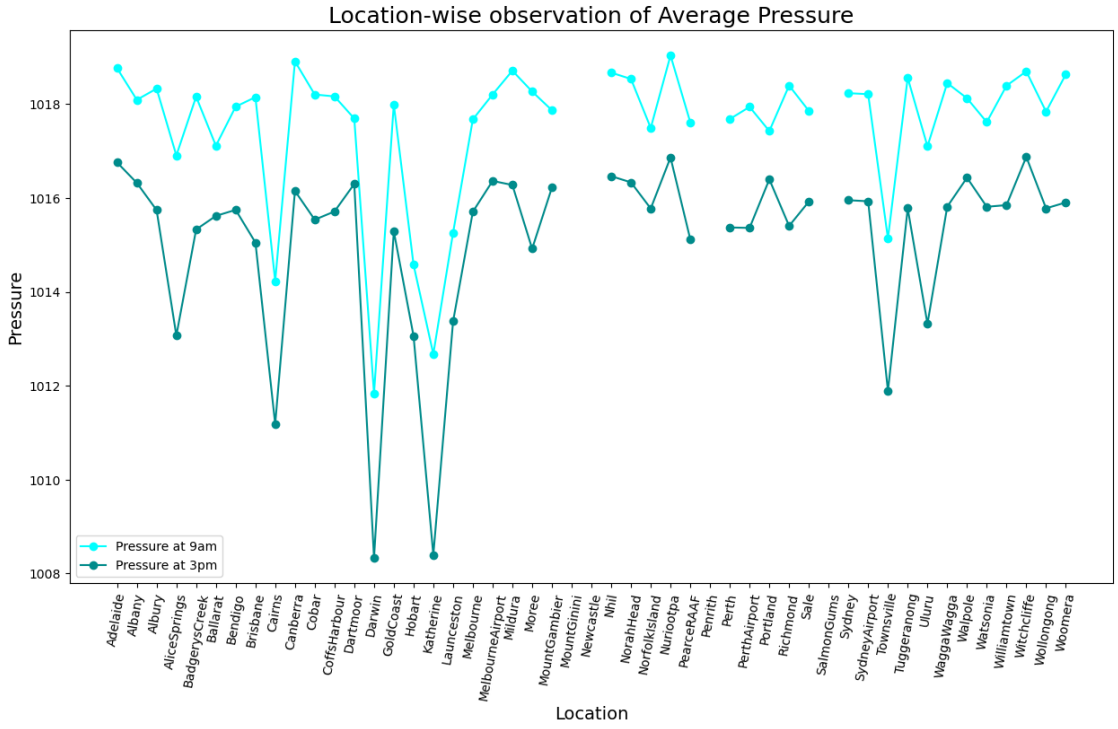
[ ]:
```
pressure_weather_df = data_test.groupby(['Location'])[['Pressure9am',␣
↪'Pressure3pm']].mean()
pressure_weather_df = pressure_weather_df.reset_index()
pressure_weather_df.head()
```

[ ]:

	Location	Pressure9am	Pressure3pm
0	Adelaide	1018.765897	1016.758901
1	Albany	1018.092685	1016.322547
2	Albury	1018.330725	1015.743894
3	AliceSprings	1016.907675	1013.070362
4	BadgerysCreek	1018.159763	1015.329858

[ ]:
```
x = pressure_weather_df.loc[:, 'Location']
y1 = pressure_weather_df['Pressure9am']
y2 = pressure_weather_df['Pressure3pm']

plt.figure(figsize = (15, 8))
```

```
plt.plot(x, y1, marker='o', color = 'cyan', label = 'Pressure at 9am')
plt.plot(x, y2, marker='o', color = 'darkcyan', label = 'Pressure at 3pm')

plt.xlabel('Location', fontsize = 14)
plt.ylabel('Pressure', fontsize = 14)
plt.title('Location-wise observation of Average Pressure', fontsize = 18)
plt.legend(fontsize = 10, loc = 'best')
plt.xticks(rotation=80)
plt.show()
```



During this analysis, it was found that the pressure in Canberra is the highest at 9 o'clock in the morning. at 1018.93 hPa. On the other hand, Adelaide, Australia has the highest pressure at 15:00 at 1016.79 hPa. In short, it can be concluded that the pressure at 9 o'clock is much higher than the wind speed at 3 o'clock.

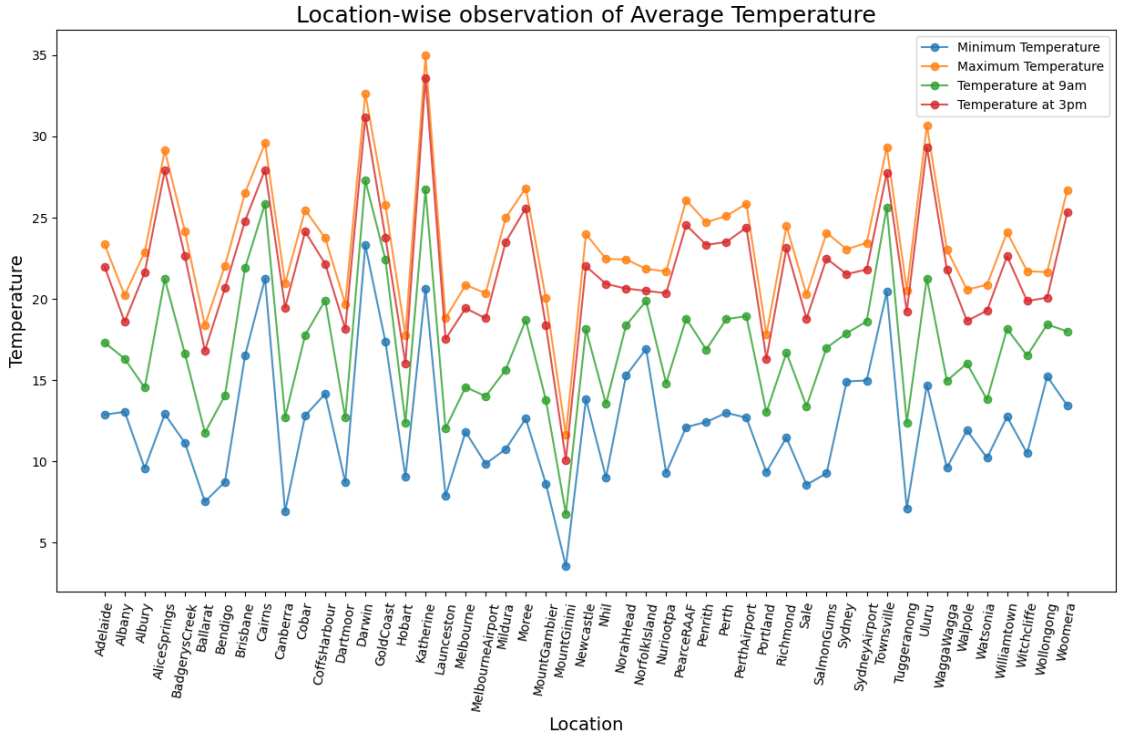
2.5 5. Average Temperature Analysis

[ ]:
```
location_weather_df = data_test.groupby(['Location'])[['MinTemp', 'MaxTemp',␣
↪'Temp9am', 'Temp3pm']].mean()
location_weather_df = location_weather_df.reset_index()
location_weather_df.head()
```

[ ]:

	Location	MinTemp	MaxTemp	Temp9am	Temp3pm
0	Adelaide	12.874643	23.337500	17.311868	21.972887
1	Albany	13.048097	20.219078	16.311321	18.584125
2	Albury	9.579700	22.881473	14.530370	21.622465
3	AliceSprings	12.905811	29.124808	21.212390	27.892645
4	BadgerysCreek	11.146833	24.163318	16.643552	22.636281

[ ]:
```
x = location_weather_df.loc[:, 'Location']
y1 = location_weather_df['MinTemp']
y2 = location_weather_df['MaxTemp']
y3 = location_weather_df['Temp9am']
y4 = location_weather_df['Temp3pm']

plt.figure(figsize = (15, 8))

plt.plot(x, y1, label = 'Minimum Temperature', marker='o', alpha = 0.8)
plt.plot(x, y2, label = 'Maximum Temperature', marker='o', alpha = 0.8)
plt.plot(x, y3, label = 'Temperature at 9am', marker='o', alpha = 0.8)
plt.plot(x, y4, label = 'Temperature at 3pm', marker='o', alpha = 0.8)

plt.xlabel('Location', fontsize = 14)
plt.ylabel('Temperature', fontsize = 14)
plt.title('Location-wise observation of Average Temperature', fontsize = 18)
plt.legend(fontsize = 10, loc = 'best')
plt.xticks(rotation=80)
plt.show()
```



2.6 Model

[ ]:
```
X = df_train.drop(columns=['RainTomorrow'])
y = df_train['RainTomorrow']
df = pd.concat([X, df_test], axis=0)
```

[ ]:
```
df.describe().T
```

[ ]:

	count	mean	std	min	25%	50%	\
row ID	142193.0	41227.832566	28156.744372	0.0	17774.0	35548.00	
MinTemp	141556.0	12.186400	6.403283	-8.5	7.6	12.00	
MaxTemp	141871.0	23.226784	7.117618	-4.8	17.9	22.60	
MeanTemp	142193.0	17.672565	6.328795	-6.2	12.9	17.35	
Rainfall	140787.0	2.349974	8.465173	0.0	0.0	0.00	
Evaporation	56985.0	5.461320	4.162490	0.0	2.6	4.80	
Sunshine	52199.0	7.615090	3.783008	0.0	4.8	8.40	
WindGustSpeed	132923.0	39.984292	13.588801	6.0	31.0	39.00	
WindSpeed9am	140845.0	14.001988	8.893337	0.0	7.0	13.00	
WindSpeed3pm	139563.0	18.637576	8.803345	0.0	13.0	19.00	
Humidity9am	140419.0	68.843810	19.051293	0.0	57.0	70.00	
Humidity3pm	138583.0	51.482606	20.797772	0.0	37.0	52.00	
AvgHumidity	142193.0	59.080240	19.084638	0.0	47.0	60.50	
Pressure9am	128179.0	1017.653758	7.105476	980.5	1012.9	1017.60	
Pressure3pm	128212.0	1015.258204	7.036677	977.1	1010.4	1015.20	
AvgPressure	142193.0	916.413905	301.650595	0.0	1009.8	1015.50	
Cloud9am	88536.0	4.437189	2.887016	0.0	1.0	5.00	
Cloud3pm	85099.0	4.503167	2.720633	0.0	2.0	5.00	
Temp9am	141289.0	16.987509	6.492838	-7.2	12.3	16.70	
Temp3pm	139467.0	21.687235	6.937594	-5.4	16.6	21.10	

	75%	max
row ID	63967.0	99515.0
MinTemp	16.8	33.9
MaxTemp	28.2	48.1
MeanTemp	22.3	38.8
Rainfall	0.8	371.0
Evaporation	7.4	86.2
Sunshine	10.6	14.5
WindGustSpeed	48.0	135.0
WindSpeed9am	19.0	130.0
WindSpeed3pm	24.0	87.0
Humidity9am	83.0	100.0
Humidity3pm	66.0	100.0
AvgHumidity	72.5	100.0
Pressure9am	1022.4	1041.0

```
Pressure3pm     1020.0   1039.6
AvgPressure     1020.6   1040.1
Cloud9am           7.0      9.0
Cloud3pm           7.0      9.0
Temp9am           21.6     40.2
Temp3pm           26.4     46.7
```

[ ]:
```
df.drop(columns='row ID', inplace=True)
total = df.isnull().sum().sort_values(ascending=False)
percent = (df.isnull().sum() / df.isnull().count()).sort_values(ascending=False)
missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
missing_data
```

[ ]:
```
               Total   Percent
Sunshine       89994  0.632900
Evaporation    85208  0.599242
Cloud3pm       57094  0.401525
Cloud9am       53657  0.377353
Pressure9am    14014  0.098556
Pressure3pm    13981  0.098324
WindGustSpeed   9270  0.065193
WindDir9am      7006  0.049271
WindGustDir     6521  0.045860
Humidity3pm     3610  0.025388
Temp3pm         2726  0.019171
WindDir3pm      2648  0.018623
WindSpeed3pm    2630  0.018496
Humidity9am     1774  0.012476
RainToday       1406  0.009888
Rainfall        1406  0.009888
WindSpeed9am    1348  0.009480
Temp9am          904  0.006358
MinTemp          637  0.004480
MaxTemp          322  0.002265
AvgHumidity        0  0.000000
AvgPressure        0  0.000000
MeanTemp           0  0.000000
Location           0  0.000000
```

[ ]:
```
df.drop(columns=['Sunshine', 'Evaporation'], inplace=True)
df.dtypes
```

[ ]:
```
Location          object
MinTemp          float64
MaxTemp          float64
MeanTemp         float64
Rainfall         float64
```

```
WindGustDir       object
WindGustSpeed    float64
WindDir9am        object
WindDir3pm        object
WindSpeed9am     float64
WindSpeed3pm     float64
Humidity9am      float64
Humidity3pm      float64
AvgHumidity      float64
Pressure9am      float64
Pressure3pm      float64
AvgPressure      float64
Cloud9am         float64
Cloud3pm         float64
Temp9am          float64
Temp3pm          float64
RainToday         object
dtype: object
```

[ ]:
```
categorical = df.select_dtypes(include = "object").columns
cleaner = ColumnTransformer([
    ('categorical_transformer', SimpleImputer(strategy='most_frequent'),␣
 ↪categorical)
])
df[categorical] = cleaner.fit_transform(df[categorical])

null_columns=df.columns[df.isnull().any()]
df[null_columns].isnull().sum()
```

[ ]:
```
MinTemp            637
MaxTemp            322
Rainfall          1406
WindGustSpeed     9270
WindSpeed9am      1348
WindSpeed3pm      2630
Humidity9am       1774
Humidity3pm       3610
Pressure9am      14014
Pressure3pm      13981
Cloud9am         53657
Cloud3pm         57094
Temp9am            904
Temp3pm           2726
dtype: int64
```

[ ]:
```
df = df.fillna(df.median())
df.isnull().sum()
```

```
/var/folders/df/npmhf4fs0qb8cnwm2kmptxk00000gn/T/ipykernel_14585/1273592041.py:1
: FutureWarning:

The default value of numeric_only in DataFrame.median is deprecated. In a future
version, it will default to False. In addition, specifying 'numeric_only=None'
is deprecated. Select only valid columns or specify the value of numeric_only to
silence this warning.
```

[ ]:
```
Location         0
MinTemp          0
MaxTemp          0
MeanTemp         0
Rainfall         0
WindGustDir      0
WindGustSpeed    0
WindDir9am       0
WindDir3pm       0
WindSpeed9am     0
WindSpeed3pm     0
Humidity9am      0
Humidity3pm      0
AvgHumidity      0
Pressure9am      0
Pressure3pm      0
AvgPressure      0
Cloud9am         0
Cloud3pm         0
Temp9am          0
Temp3pm          0
RainToday        0
dtype: int64
```

[ ]:
```
categorical = df.select_dtypes(include = "object").columns
for i in range(len(categorical)):
    print(df[categorical[i]].value_counts())
    print('**********************************\n')
```

```
Canberra            3418
Sydney              3337
Perth               3193
Darwin              3192
Hobart              3188
Brisbane            3161
Adelaide            3090
Bendigo             3034
Townsville          3033
AliceSprings        3031
```

```
MountGambier        3030
Launceston          3028
Ballarat            3028
Albany              3016
Albury              3011
PerthAirport        3009
MelbourneAirport    3009
Mildura             3007
SydneyAirport       3005
Nuriootpa           3002
Sale                3000
Watsonia            2999
Tuggeranong         2998
Portland            2996
Woomera             2990
Cairns              2988
Cobar               2988
Wollongong          2983
GoldCoast           2980
WaggaWagga          2976
Penrith             2964
NorfolkIsland       2964
SalmonGums          2955
Newcastle           2955
CoffsHarbour        2953
Witchcliffe         2952
Richmond            2951
Dartmoor            2943
NorahHead           2929
BadgerysCreek       2928
MountGinini         2907
Moree               2854
Walpole             2819
PearceRAAF          2762
Williamtown         2553
Melbourne           2435
Nhil                1569
Katherine           1559
Uluru               1521
Name: Location, dtype: int64
**********************************

W      19110
SE      9309
E       9071
N       9033
SSE     8993
S       8949
```

```
WSW     8901
SW      8797
SSW     8610
WNW     8066
NW      8003
ENE     7992
ESE     7305
NE      7060
NNW     6561
NNE     6433
Name: WindGustDir, dtype: int64
**********************************

N      21406
SE      9162
E       9024
SSE     8966
NW      8552
S       8493
W       8260
SW      8237
NNE     7948
NNW     7840
ENE     7735
ESE     7558
NE      7527
SSW     7448
WNW     7194
WSW     6843
Name: WindDir9am, dtype: int64
**********************************

SE     14441
W       9911
S       9598
WSW     9329
SW      9182
SSE     9142
N       8667
WNW     8656
NW      8468
ESE     8382
E       8342
NE      8164
SSW     8010
NNW     7733
ENE     7724
NNE     6444
```

```
Name: WindDir3pm, dtype: int64
**********************************

No     77887
0.0    32851
Yes    22056
1.0     9399
Name: RainToday, dtype: int64
**********************************
```

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

for col in df.columns:
    if df[col].dtype == 'object':
        df[col] = df[col].astype(str)
        df[col] = LabelEncoder().fit_transform(df[col])

df
```

[ ]:

	Location	MinTemp	MaxTemp	MeanTemp	Rainfall	WindGustDir	\
0	2	13.4	22.9	18.2	0.6	13	
1	2	7.4	25.1	16.3	0.0	14	
2	2	17.5	32.3	24.9	1.0	13	
3	2	14.6	29.7	22.2	0.2	14	
4	2	7.7	26.7	17.2	0.0	13	
…	…	…	…	…	…	…	
42672	41	2.4	19.1	10.8	0.0	0	
42673	41	2.3	21.4	11.9	0.0	9	
42674	41	2.6	22.5	12.6	0.0	8	
42675	41	7.4	20.6	14.0	0.0	0	
42676	41	7.8	27.0	17.4	0.0	9	

	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	…	Humidity3pm	\
0	44.0	13	14	20.0	…	22.0	
1	44.0	6	15	4.0	…	25.0	
2	41.0	1	7	7.0	…	33.0	
3	56.0	13	13	19.0	…	23.0	
4	35.0	10	13	6.0	…	19.0	
…	…	…	…	…	…	…	
42672	33.0	9	0	17.0	…	24.0	
42673	22.0	9	10	11.0	…	28.0	
42674	19.0	8	0	9.0	…	24.0	
42675	35.0	2	0	15.0	…	33.0	
42676	28.0	10	3	13.0	…	24.0	

	AvgHumidity	Pressure9am	Pressure3pm	AvgPressure	Cloud9am	Cloud3pm	\

```
0             46.5       1007.7       1007.1       1007.4       8.0       5.0
1             34.5       1010.6       1007.8       1009.2       5.0       5.0
2             57.5       1010.8       1006.0       1008.4       7.0       8.0
3             39.0       1009.2       1005.4       1007.3       5.0       5.0
4             33.5       1013.4       1010.1       1011.8       5.0       5.0
…              …          …            …            …           …         …
42672         41.5       1030.0       1026.2       1028.1       5.0       5.0
42673         44.0       1026.9       1022.8       1024.9       5.0       5.0
42674         41.5       1025.0       1021.4       1023.2       5.0       5.0
42675         48.0       1027.2       1023.3       1025.3       5.0       5.0
42676         37.5       1019.4       1016.5       1018.0       3.0       2.0

       Temp9am  Temp3pm  RainToday
0         16.9     21.8          2
1         17.2     24.3          2
2         17.8     29.7          2
3         20.6     28.9          2
4         16.3     25.5          2
…          …        …            …
42672      8.0     18.8          0
42673      8.9     20.3          0
42674      8.8     22.1          0
42675     11.0     20.3          0
42676     15.1     26.0          0

[142193 rows x 22 columns]
```

[ ]:
```
'''objects = df.select_dtypes(include = "object").columns
for i in range(len(objects)):
    df[objects[i]] = LabelEncoder().fit_transform(df[objects[i]])

df'''
```

[ ]:
```
'objects = df.select_dtypes(include = "object").columns\nfor i in
range(len(objects)):\n    df[objects[i]] =
LabelEncoder().fit_transform(df[objects[i]])\n\ndf'
```

[ ]:
```
train = df.iloc[:99516,:]
new_train = pd.concat([train, y], axis=1)
test = df.iloc[99516:, :]
new_train
```

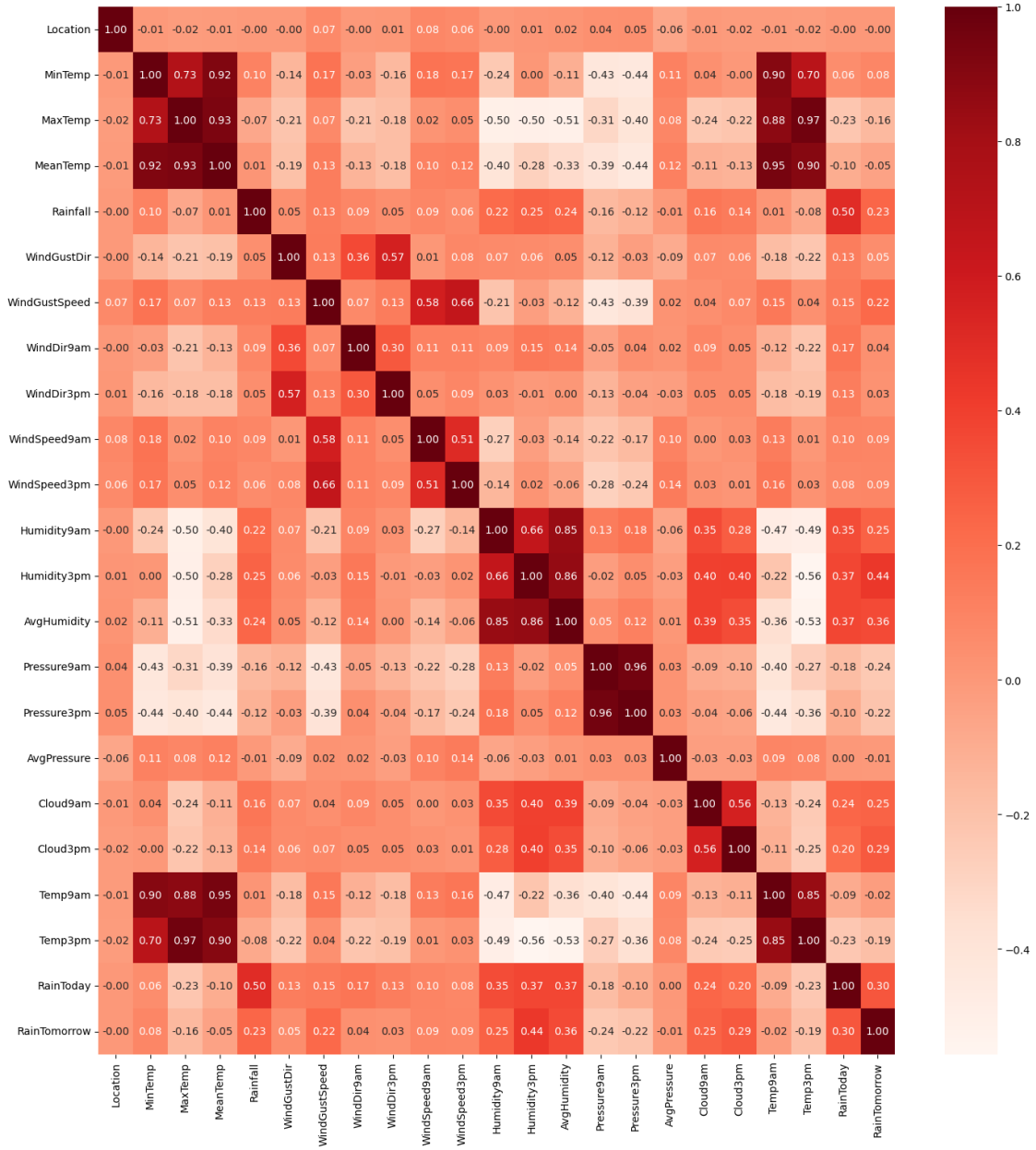
[ ]:
```
       Location  MinTemp  MaxTemp  MeanTemp  Rainfall  WindGustDir  \
0             2     13.4     22.9     18.20       0.6           13
1             2      7.4     25.1     16.30       0.0           14
2             2     17.5     32.3     24.90       1.0           13
3             2     14.6     29.7     22.20       0.2           14
```

```
4             2      7.7     26.7     17.20       0.0           13
…         …      …       …        …        …          …
99511        41      8.0     20.7     14.35       0.0            2
99512        41      3.5     21.8     12.70       0.0            0
99513        41      2.8     23.4     13.10       0.0            0
99514        41      3.6     25.3     14.50       0.0            6
99515        41      5.4     26.9     16.20       0.0            3

       WindGustSpeed  WindDir9am  WindDir3pm  WindSpeed9am  …  AvgHumidity  \
0               44.0          13          14          20.0  …         46.5
1               44.0           6          15           4.0  …         34.5
2               41.0           1           7           7.0  …         57.5
3               56.0          13          13          19.0  …         39.0
4               35.0          10          13           6.0  …         33.5
…              …          …         …           …  …          …
99511           41.0           9           0          19.0  …         44.0
99512           31.0           2           0          15.0  …         43.0
99513           31.0           9           1          13.0  …         37.5
99514           22.0           9           3          13.0  …         38.5
99515           37.0           9          14           9.0  …         38.5

       Pressure9am  Pressure3pm  AvgPressure  Cloud9am  Cloud3pm  Temp9am  \
0           1007.7       1007.1       1007.4       8.0       5.0     16.9
1           1010.6       1007.8       1009.2       5.0       5.0     17.2
2           1010.8       1006.0       1008.4       7.0       8.0     17.8
3           1009.2       1005.4       1007.3       5.0       5.0     20.6
4           1013.4       1010.1       1011.8       5.0       5.0     16.3
…            …          …          …       …       …       …
99511       1028.1       1024.3       1026.2       5.0       7.0     11.6
99512       1024.7       1021.2       1023.0       5.0       5.0      9.4
99513       1024.6       1020.3       1022.5       5.0       5.0     10.1
99514       1023.5       1019.1       1021.3       5.0       5.0     10.9
99515       1021.0       1016.8       1018.9       5.0       5.0     12.5

       Temp3pm  RainToday  RainTomorrow
0         21.8          2             0
1         24.3          2             0
2         29.7          2             0
3         28.9          2             0
4         25.5          2             0
…        …        …           …
99511     20.0          2             0
99512     20.9          2             0
99513     22.4          2             0
99514     24.5          2             0
99515     26.1          2             0
```

[99516 rows x 23 columns]

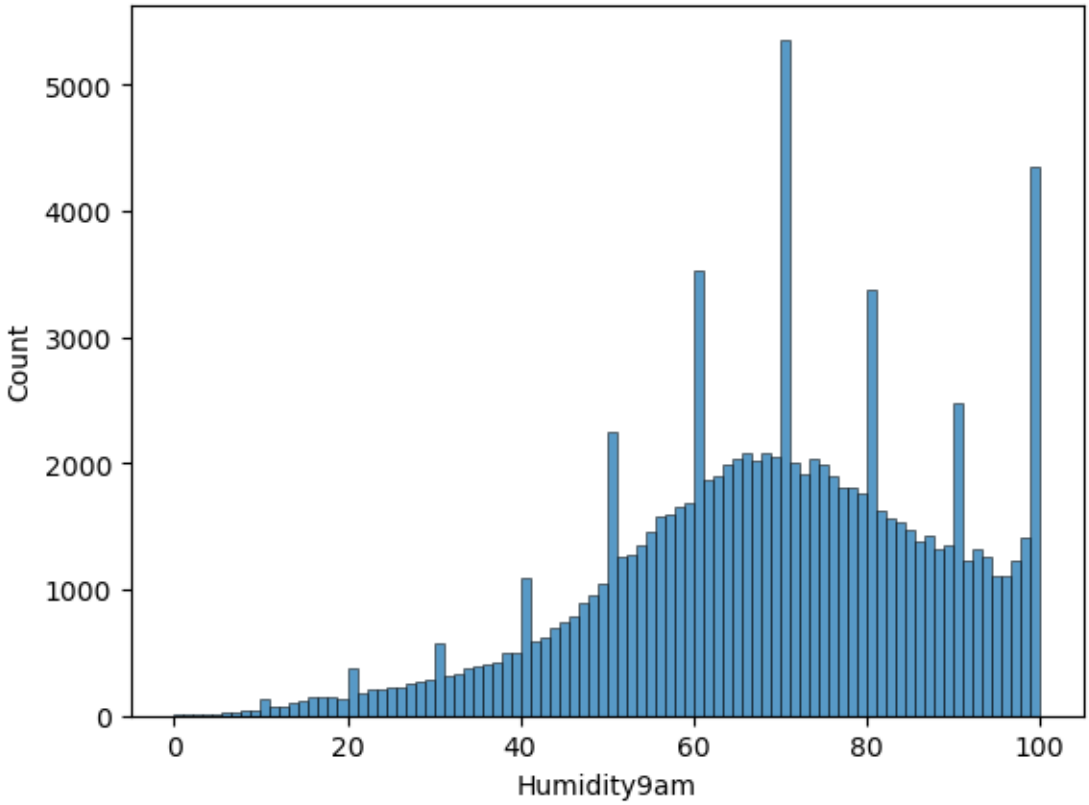
[ ]:
```
plt.figure(figsize=(17,18))
cor = new_train.corr()
sns.heatmap(cor, annot=True, cmap=plt.cm.Reds,fmt='.2f')
```

[ ]: <Axes: >



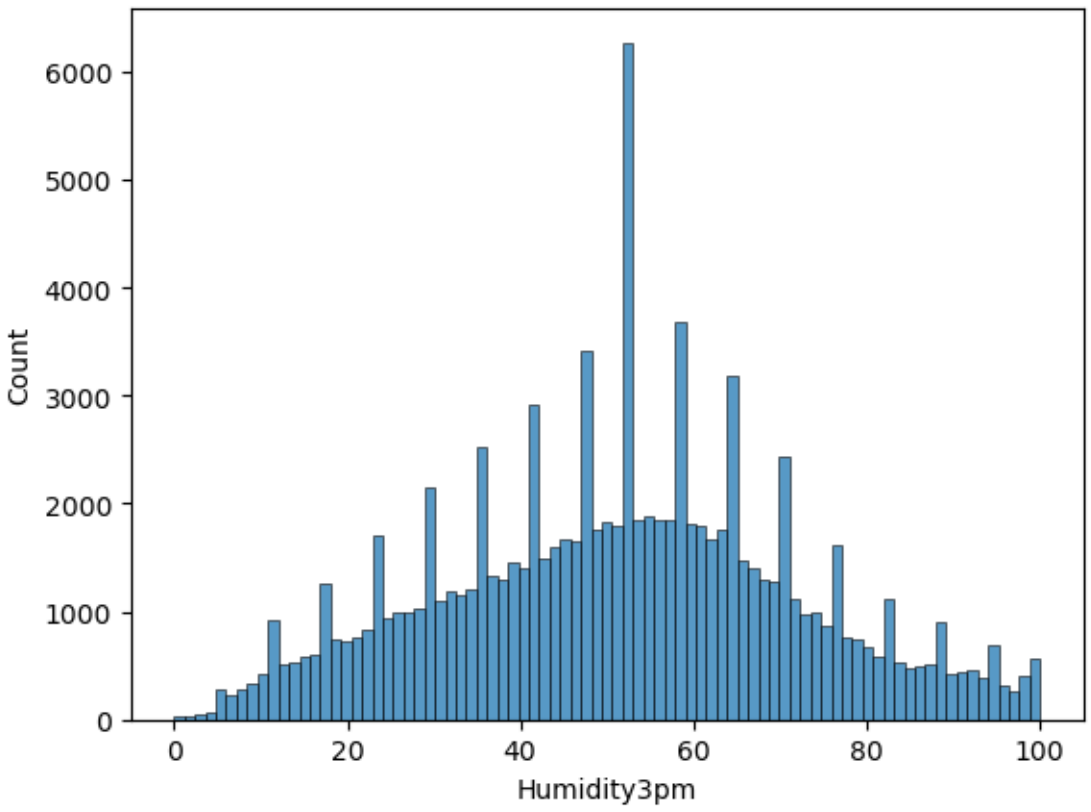
[ ]:
```
sns.histplot(new_train['Humidity9am'])
```

[ ]: <Axes: xlabel='Humidity9am', ylabel='Count'>



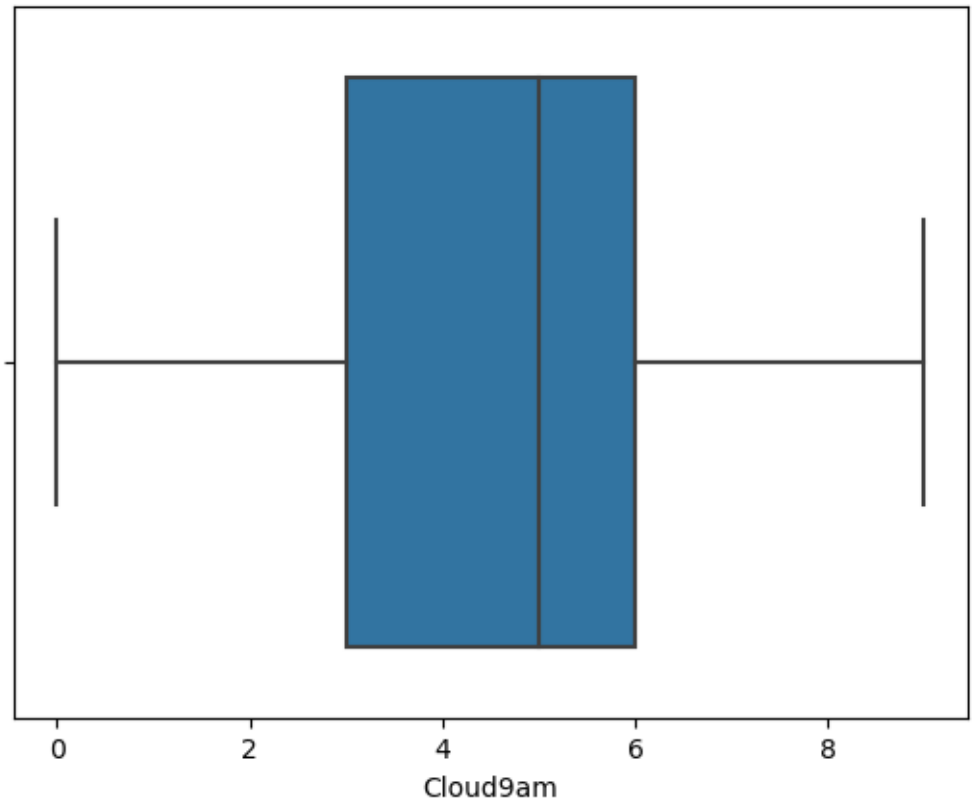
[ ]:
```
sns.histplot(new_train['Humidity3pm'])
```

[ ]: <Axes: xlabel='Humidity3pm', ylabel='Count'>



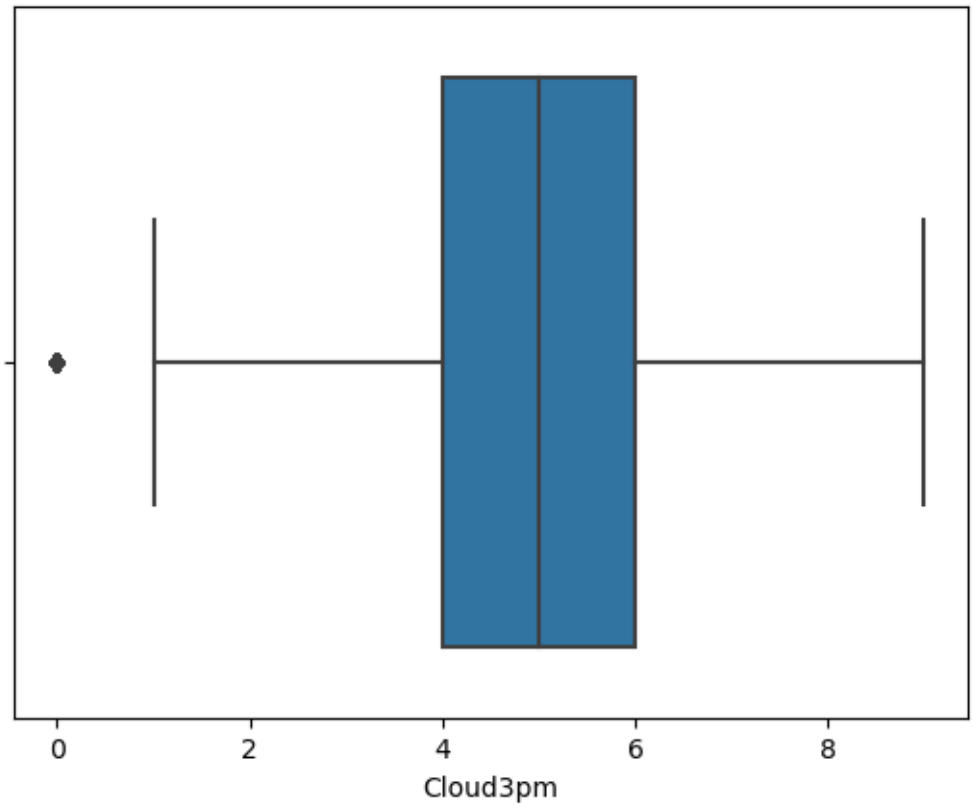
[ ]:
```
sns.boxplot(x=new_train['Cloud9am'])
```

[ ]:
```
<Axes: xlabel='Cloud9am'>
```



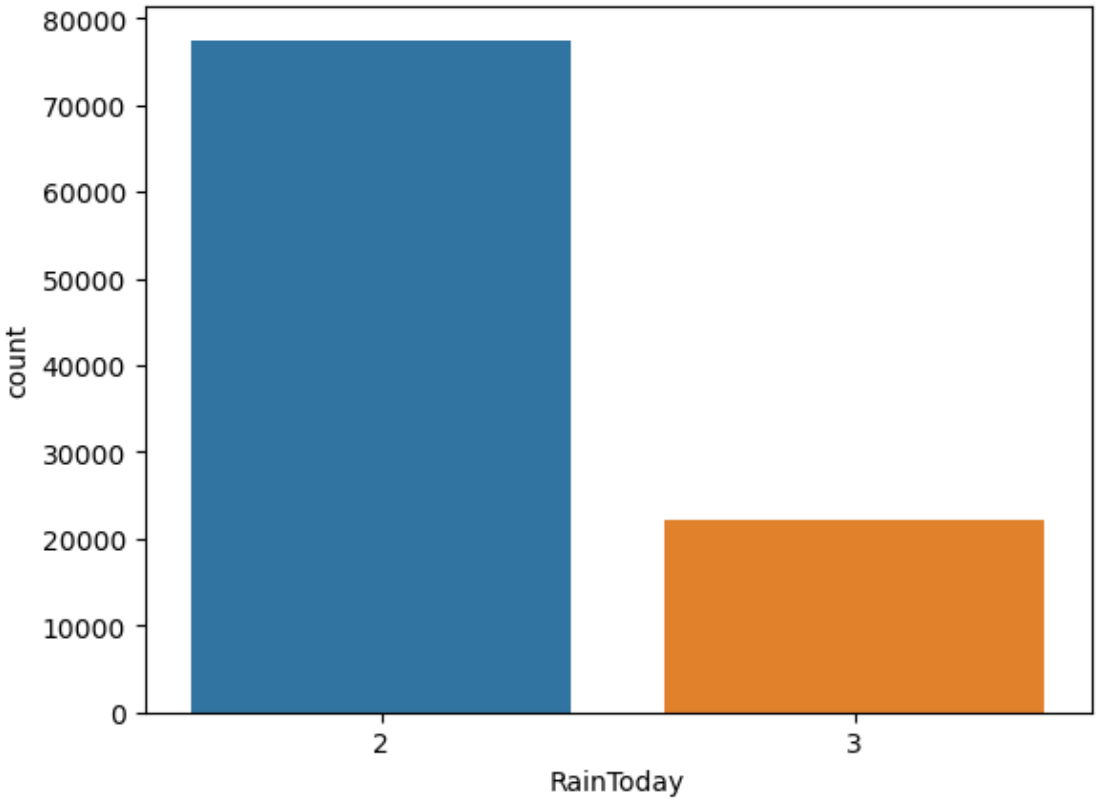
[ ]:
```
sns.boxplot(x=new_train['Cloud3pm'])
```

[ ]:
```
<Axes: xlabel='Cloud3pm'>
```



[ ]:
```
sns.countplot(x=new_train['RainToday'])
```

[ ]:
```
<Axes: xlabel='RainToday', ylabel='count'>
```

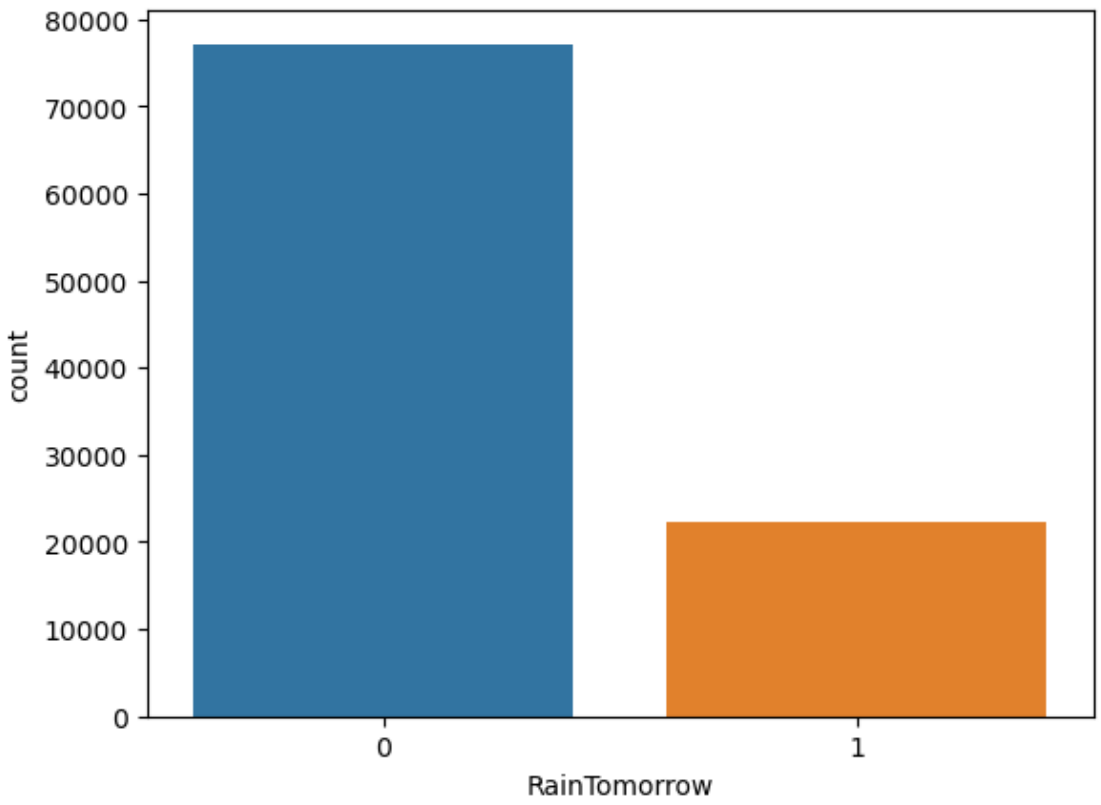


[ ]:
```
new_train['RainTomorrow'].value_counts()
```

[ ]:
```
0    77157
1    22359
Name: RainTomorrow, dtype: int64
```

[ ]:
```
sns.countplot(x=new_train['RainTomorrow'])
```

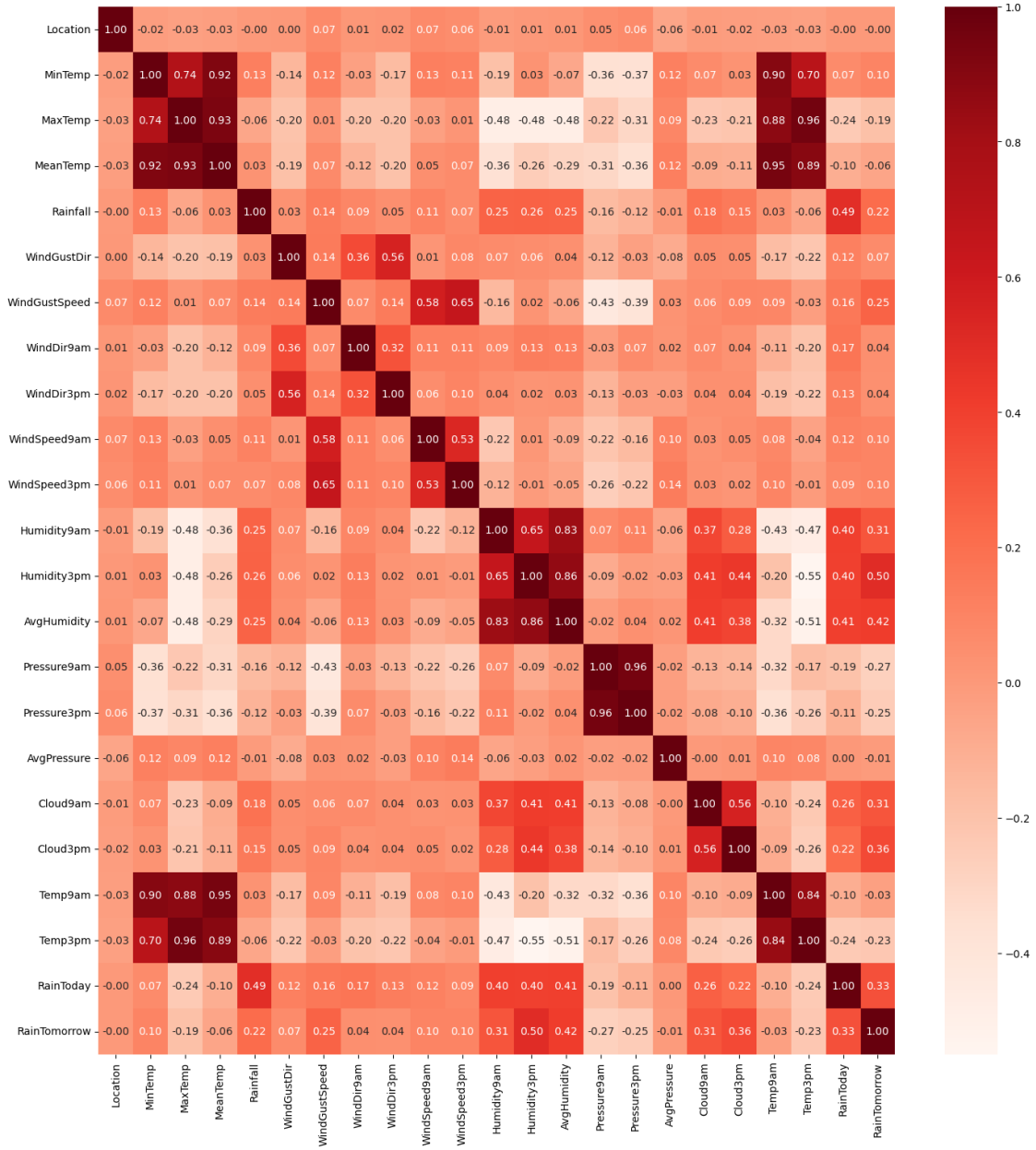
[ ]:
```
<Axes: xlabel='RainTomorrow', ylabel='count'>
```



[ ]:
```
df_majority_0 = new_train[(new_train['RainTomorrow']==0)]
df_minority_1 = new_train[(new_train['RainTomorrow']==1)]

df_minority_upsampled = resample(df_minority_1,
                                 replace=True,
                                 n_samples= 77157,
                                 random_state=42)

df_upsampled = pd.concat([df_minority_upsampled, df_majority_0])
```

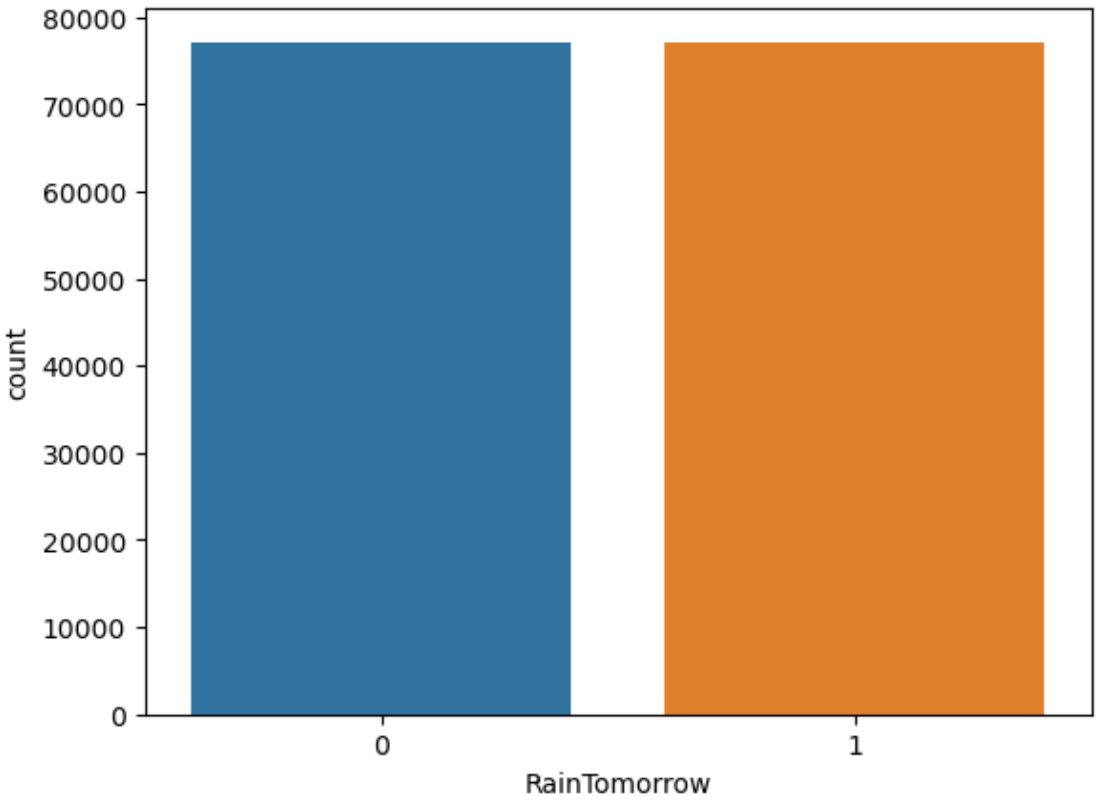
[ ]:
```
plt.figure(figsize=(17,18))
cor = df_upsampled.corr()
sns.heatmap(cor, annot=True, cmap=plt.cm.Reds,fmt='.2f')
```

[ ]: <Axes: >



[ ]:
```
sns.countplot(x=df_upsampled['RainTomorrow'])
```

[ ]: <Axes: xlabel='RainTomorrow', ylabel='count'>



[ ]:
```
X = df_upsampled.drop(columns='RainTomorrow')
y = df_upsampled['RainTomorrow']
```

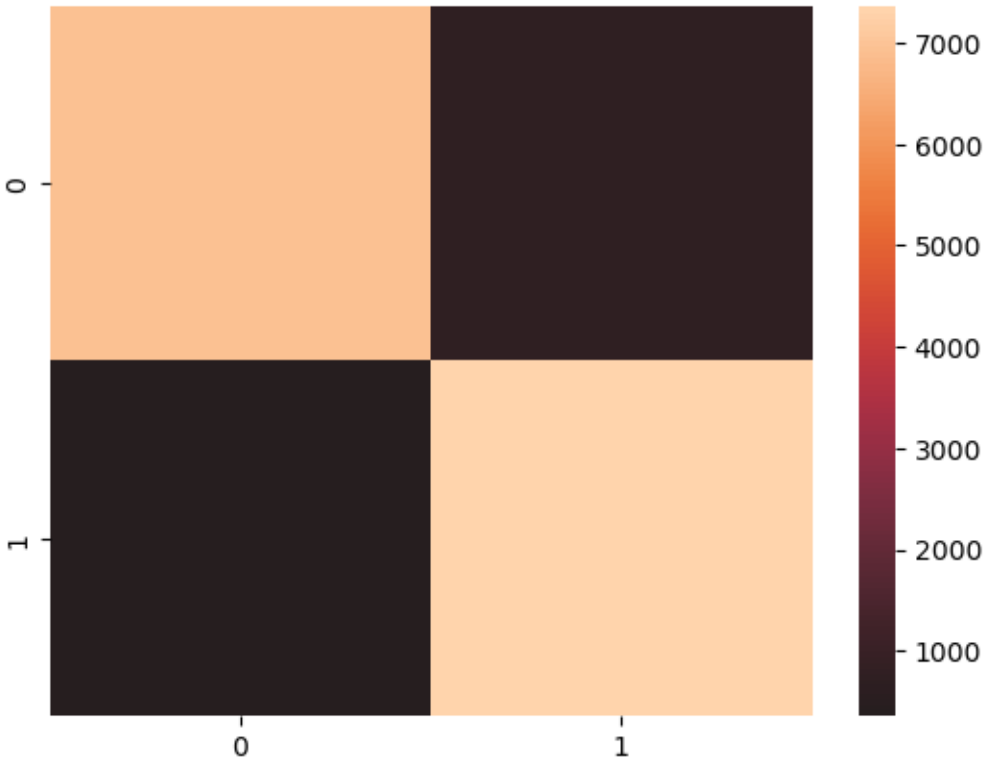
[ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.9,␣
 ↪shuffle=True, random_state=44)
```

[ ]:
```
RandomForestClassifierModel = RandomForestClassifier(criterion = 'gini',␣
 ↪max_depth=17, n_estimators=100, random_state=44)
RandomForestClassifierModel.fit(X_train, y_train)

print('RandomForestClassifierModel Train Score is : ' ,␣
 ↪RandomForestClassifierModel.score(X_train, y_train))
print('RandomForestClassifierModel Test Score is : ' ,␣
 ↪RandomForestClassifierModel.score(X_test, y_test))
```

```
RandomForestClassifierModel Train Score is :  0.9804006278711425
RandomForestClassifierModel Test Score is :  0.9274235355106273
```

[ ]:
```
y_pred_RF = RandomForestClassifierModel.predict(X_test)
CM_RF = confusion_matrix(y_test, y_pred_RF)
```

```
sns.heatmap(CM_RF, center=True)
plt.show()

print('Confusion Matrix is\n', CM_RF)
```



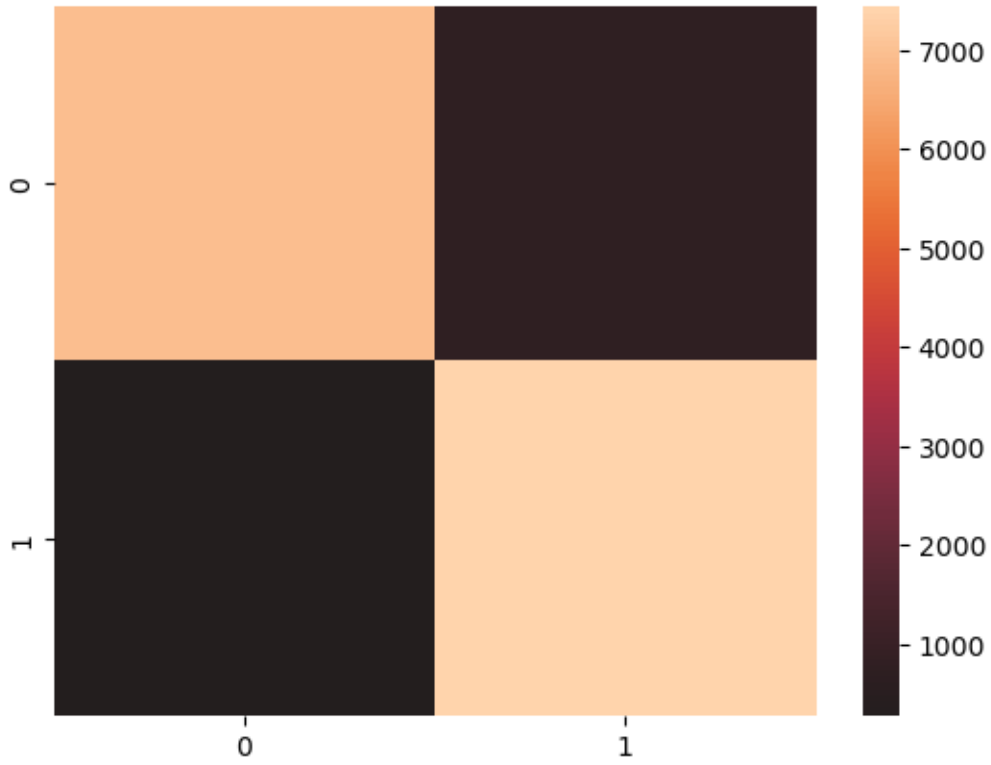

```
Confusion Matrix is
 [[6946  763]
 [ 357 7366]]
```

[ ]:
```
GBCModel = GradientBoostingClassifier(n_estimators=200, max_depth=11,␣
 ↪learning_rate=0.07, random_state=44)
GBCModel.fit(X_train, y_train)
print('GBCModel Train Score is : ' , GBCModel.score(X_train, y_train))
print('GBCModel Test Score is : ' , GBCModel.score(X_test, y_test))
```

```
GBCModel Train Score is :  0.9900923085785055
GBCModel Test Score is :  0.9331907724209435
```

[ ]:
```
y_pred_GB = GBCModel.predict(X_test)
CM_GB = confusion_matrix(y_test, y_pred_GB)

sns.heatmap(CM_GB, center=True)
plt.show()
```

```
print('Confusion Matrix is\n', CM_GB)
```

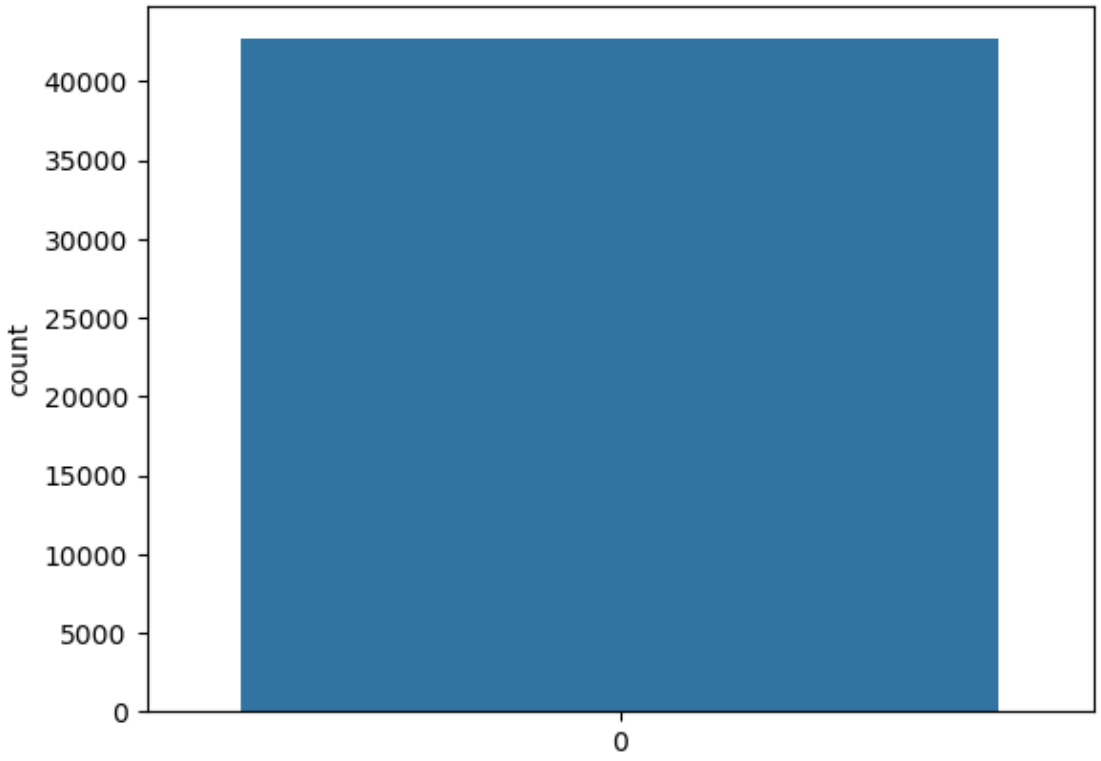



```
Confusion Matrix is
 [[6951  758]
 [ 273 7450]]
```

[ ]:
```
y_pred = GBCModel.predict(test)
```

[ ]:
```
sns.countplot(y_pred)
```

[ ]:
```
<Axes: ylabel='count'>
```



[ ]:
```
test = pd.read_csv('WeatherTestData.csv')
submission = test[["row ID"]]
submission["RainTomorrow"] = y_pred
```

```
/var/folders/df/npmhf4fs0qb8cnwm2kmptxk00000gn/T/ipykernel_14585/1624392679.py:3
: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

[ ]:
```
submission.to_csv('predict_weather.csv', index=False)
```

Two different testing algorithms that we use:

1. Randomized Search Cross Validation for Hyperparameter Tuning: This algorithm randomly selects a set of hyperparameters and uses cross-validation to evaluate the model's performance. It then repeats this process multiple times and selects the best set of hyperparameters that give the highest accuracy score.

[ ]:
```python
from sklearn.model_selection import RandomizedSearchCV

# define the hyperparameter grid
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_features': ['auto', 'sqrt'],
    'max_depth': [5, 10, 15, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# create a Random Forest Classifier object
rfc = RandomForestClassifier(random_state=42)

# create a RandomizedSearchCV object
rscv = RandomizedSearchCV(
    estimator=rfc, param_distributions=param_grid,
    n_iter=10, cv=5, verbose=2, random_state=42, n_jobs=-1
)

# fit the RandomizedSearchCV object on the training data
rscv.fit(X_train, y_train)

# print the best hyperparameters and the corresponding accuracy score
print("Best Hyperparameters:", rscv.best_params_)
print("Best Accuracy Score:", rscv.best_score_)

# evaluate the model on the test data
rfc_best = rscv.best_estimator_
print("Test Accuracy Score:", rfc_best.score(X_test, y_test))
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=4,
min_samples_split=10, n_estimators=100; total time=  20.6s
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=4,
min_samples_split=10, n_estimators=100; total time=  20.7s
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=4,
min_samples_split=10, n_estimators=100; total time=  20.8s
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=2,
min_samples_split=5, n_estimators=100; total time=  21.6s
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=2,
min_samples_split=5, n_estimators=100; total time=  21.8s
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=2,
min_samples_split=5, n_estimators=100; total time=  21.8s
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=2,
min_samples_split=5, n_estimators=100; total time=  22.0s
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=2,
min_samples_split=5, n_estimators=100; total time=  22.0s
```

```
/Users/hemang/miniconda3/lib/python3.10/site-
packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_features='auto'`
has been deprecated in 1.1 and will be removed in 1.3. To keep the past
behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it
is also the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(

[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1,
min_samples_split=5, n_estimators=100; total time=  17.7s

/Users/hemang/miniconda3/lib/python3.10/site-
packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_features='auto'`
has been deprecated in 1.1 and will be removed in 1.3. To keep the past
behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it
is also the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(

[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1,
min_samples_split=5, n_estimators=100; total time=  17.7s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1,
min_samples_split=5, n_estimators=100; total time=  17.7s

/Users/hemang/miniconda3/lib/python3.10/site-
packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_features='auto'`
has been deprecated in 1.1 and will be removed in 1.3. To keep the past
behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it
is also the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
/Users/hemang/miniconda3/lib/python3.10/site-
packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_features='auto'`
has been deprecated in 1.1 and will be removed in 1.3. To keep the past
behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it
is also the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(

[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1,
min_samples_split=5, n_estimators=100; total time=  17.6s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1,
min_samples_split=5, n_estimators=100; total time=  17.9s

/Users/hemang/miniconda3/lib/python3.10/site-
packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_features='auto'`
has been deprecated in 1.1 and will be removed in 1.3. To keep the past
behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it
is also the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
/Users/hemang/miniconda3/lib/python3.10/site-
packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_features='auto'`
has been deprecated in 1.1 and will be removed in 1.3. To keep the past
behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it
is also the default value for RandomForestClassifiers and ExtraTreesClassifiers.
```

```
  warn(

[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=4,
min_samples_split=10, n_estimators=100; total time=  19.3s
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=4,
min_samples_split=10, n_estimators=100; total time=  19.4s

/Users/hemang/miniconda3/lib/python3.10/site-
packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_features='auto'`
has been deprecated in 1.1 and will be removed in 1.3. To keep the past
behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it
is also the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
/Users/hemang/miniconda3/lib/python3.10/site-
packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_features='auto'`
has been deprecated in 1.1 and will be removed in 1.3. To keep the past
behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it
is also the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(

[CV] END max_depth=None, max_features=auto, min_samples_leaf=2,
min_samples_split=10, n_estimators=100; total time=  20.0s

/Users/hemang/miniconda3/lib/python3.10/site-
packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_features='auto'`
has been deprecated in 1.1 and will be removed in 1.3. To keep the past
behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it
is also the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(

[CV] END max_depth=5, max_features=auto, min_samples_leaf=2,
min_samples_split=10, n_estimators=100; total time=   6.8s

/Users/hemang/miniconda3/lib/python3.10/site-
packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_features='auto'`
has been deprecated in 1.1 and will be removed in 1.3. To keep the past
behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it
is also the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(

[CV] END max_depth=5, max_features=auto, min_samples_leaf=2,
min_samples_split=10, n_estimators=100; total time=   6.7s

/Users/hemang/miniconda3/lib/python3.10/site-
packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_features='auto'`
has been deprecated in 1.1 and will be removed in 1.3. To keep the past
behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it
is also the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(

[CV] END max_depth=5, max_features=auto, min_samples_leaf=2,
min_samples_split=10, n_estimators=100; total time=   6.9s
```

```
/Users/hemang/miniconda3/lib/python3.10/site-
packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_features='auto'`
has been deprecated in 1.1 and will be removed in 1.3. To keep the past
behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it
is also the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(

[CV] END max_depth=5, max_features=auto, min_samples_leaf=2,
min_samples_split=10, n_estimators=100; total time=   6.8s

/Users/hemang/miniconda3/lib/python3.10/site-
packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_features='auto'`
has been deprecated in 1.1 and will be removed in 1.3. To keep the past
behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it
is also the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(

[CV] END max_depth=5, max_features=auto, min_samples_leaf=2,
min_samples_split=10, n_estimators=100; total time=   6.6s

/Users/hemang/miniconda3/lib/python3.10/site-
packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_features='auto'`
has been deprecated in 1.1 and will be removed in 1.3. To keep the past
behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it
is also the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(

[CV] END max_depth=None, max_features=auto, min_samples_leaf=2,
min_samples_split=10, n_estimators=100; total time=  19.2s

/Users/hemang/miniconda3/lib/python3.10/site-
packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_features='auto'`
has been deprecated in 1.1 and will be removed in 1.3. To keep the past
behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it
is also the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(

[CV] END max_depth=None, max_features=auto, min_samples_leaf=2,
min_samples_split=10, n_estimators=100; total time=  19.1s
[CV] END max_depth=None, max_features=auto, min_samples_leaf=2,
min_samples_split=10, n_estimators=100; total time=  19.2s

/Users/hemang/miniconda3/lib/python3.10/site-
packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_features='auto'`
has been deprecated in 1.1 and will be removed in 1.3. To keep the past
behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it
is also the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
/Users/hemang/miniconda3/lib/python3.10/site-
packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_features='auto'`
has been deprecated in 1.1 and will be removed in 1.3. To keep the past
behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it
```

```
is also the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
[CV] END max_depth=None, max_features=auto, min_samples_leaf=2,
min_samples_split=10, n_estimators=100; total time=  19.2s
/Users/hemang/miniconda3/lib/python3.10/site-
packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_features='auto'`
has been deprecated in 1.1 and will be removed in 1.3. To keep the past
behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it
is also the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
[CV] END max_depth=15, max_features=auto, min_samples_leaf=1,
min_samples_split=5, n_estimators=100; total time=  17.2s
/Users/hemang/miniconda3/lib/python3.10/site-
packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_features='auto'`
has been deprecated in 1.1 and will be removed in 1.3. To keep the past
behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it
is also the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
[CV] END max_depth=15, max_features=auto, min_samples_leaf=1,
min_samples_split=5, n_estimators=100; total time=  17.2s
/Users/hemang/miniconda3/lib/python3.10/site-
packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_features='auto'`
has been deprecated in 1.1 and will be removed in 1.3. To keep the past
behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it
is also the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
[CV] END max_depth=15, max_features=auto, min_samples_leaf=1,
min_samples_split=5, n_estimators=100; total time=  17.2s
/Users/hemang/miniconda3/lib/python3.10/site-
packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_features='auto'`
has been deprecated in 1.1 and will be removed in 1.3. To keep the past
behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it
is also the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
[CV] END max_depth=15, max_features=auto, min_samples_leaf=1,
min_samples_split=5, n_estimators=100; total time=  17.2s
/Users/hemang/miniconda3/lib/python3.10/site-
packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_features='auto'`
has been deprecated in 1.1 and will be removed in 1.3. To keep the past
behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it
is also the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
```

```
[CV] END max_depth=15, max_features=auto, min_samples_leaf=1,
min_samples_split=5, n_estimators=100; total time=  17.1s

/Users/hemang/miniconda3/lib/python3.10/site-
packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_features='auto'`
has been deprecated in 1.1 and will be removed in 1.3. To keep the past
behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it
is also the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(

[CV] END max_depth=10, max_features=auto, min_samples_leaf=4,
min_samples_split=2, n_estimators=200; total time=  24.3s

/Users/hemang/miniconda3/lib/python3.10/site-
packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_features='auto'`
has been deprecated in 1.1 and will be removed in 1.3. To keep the past
behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it
is also the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(

[CV] END max_depth=10, max_features=auto, min_samples_leaf=4,
min_samples_split=2, n_estimators=200; total time=  24.1s

/Users/hemang/miniconda3/lib/python3.10/site-
packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_features='auto'`
has been deprecated in 1.1 and will be removed in 1.3. To keep the past
behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it
is also the default value for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(

[CV] END max_depth=10, max_features=auto, min_samples_leaf=4,
min_samples_split=2, n_estimators=200; total time=  24.1s
[CV] END max_depth=None, max_features=auto, min_samples_leaf=4,
min_samples_split=2, n_estimators=300; total time=  56.3s
[CV] END max_depth=None, max_features=auto, min_samples_leaf=4,
min_samples_split=2, n_estimators=300; total time=  56.2s
[CV] END max_depth=10, max_features=auto, min_samples_leaf=4,
min_samples_split=2, n_estimators=200; total time=  24.5s
[CV] END max_depth=None, max_features=auto, min_samples_leaf=4,
min_samples_split=2, n_estimators=300; total time=  56.6s
[CV] END max_depth=10, max_features=auto, min_samples_leaf=4,
min_samples_split=2, n_estimators=200; total time=  24.5s
[CV] END max_depth=None, max_features=auto, min_samples_leaf=4,
min_samples_split=2, n_estimators=300; total time=  55.7s
[CV] END max_depth=None, max_features=auto, min_samples_leaf=4,
min_samples_split=2, n_estimators=300; total time=  56.4s
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=2,
min_samples_split=10, n_estimators=200; total time=  39.0s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1,
min_samples_split=5, n_estimators=200; total time=  34.6s
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=2,
```

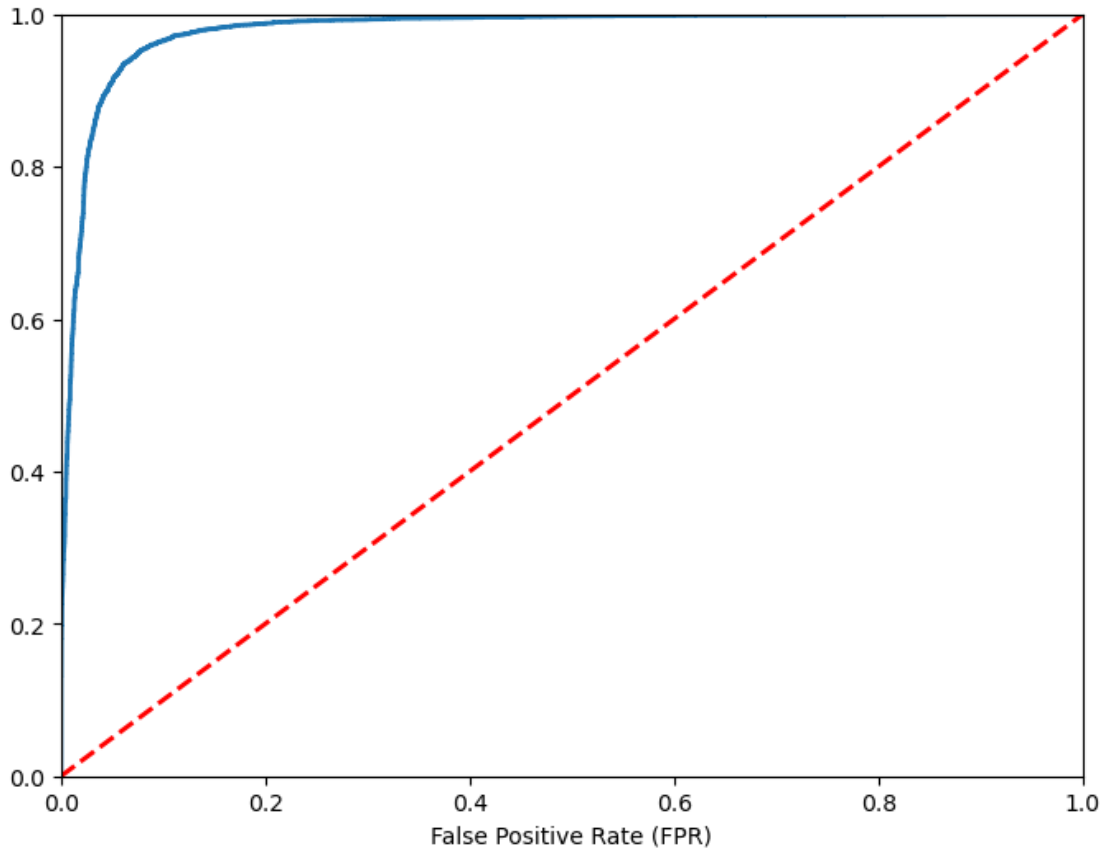
```
min_samples_split=10, n_estimators=200; total time=  38.7s
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=2,
min_samples_split=10, n_estimators=200; total time=  38.9s
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=2,
min_samples_split=10, n_estimators=200; total time=  39.1s
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=2,
min_samples_split=10, n_estimators=200; total time=  38.8s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1,
min_samples_split=5, n_estimators=200; total time=  34.4s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1,
min_samples_split=5, n_estimators=200; total time=  34.2s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1,
min_samples_split=5, n_estimators=200; total time=  29.9s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1,
min_samples_split=5, n_estimators=200; total time=  27.0s
Best Hyperparameters: {'n_estimators': 100, 'min_samples_split': 5,
'min_samples_leaf': 2, 'max_features': 'sqrt', 'max_depth': None}
Best Accuracy Score: 0.9268227680017898
Test Accuracy Score: 0.9444660445826853
```

2. Receiver Operating Characteristic (ROC) Curve: This algorithm is used to evaluate the performance of a binary classifier at different classification thresholds. It plots the True Positive Rate (TPR) against the False Positive Rate (FPR) for different threshold values. The area under the ROC curve (AUC-ROC) is a performance metric that ranges from 0.5 to 1. A higher AUC-ROC indicates better model performance.

[ ]:
```python
from sklearn.metrics import roc_curve, auc

# fit the Gradient Boosting Classifier on the training data
gbc = GradientBoostingClassifier(n_estimators=200, max_depth=11,␣
 ↪learning_rate=0.07, random_state=44)
gbc.fit(X_train, y_train)

# predict the probabilities of the positive class for the test data
y_proba = gbc.predict_proba(X_test)[:, 1]

# calculate the False Positive Rate (FPR), True Positive Rate (TPR), and␣
 ↪threshold values
fpr, tpr, thresholds = roc_curve(y_test, y_proba)

# calculate the Area Under the Curve (AUC-ROC)
auc_roc = auc(fpr, tpr)

# plot the ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, lw=2, label=f'AUC = {auc_roc:.2f}')
plt.plot([0, 1], [0, 1], linestyle='--', lw=2, color='r', label='Random Guess')
plt.xlim([0.0, 1.0])
```

```
plt.ylim([0.0, 1.0])
plt.xlabel('False Positive Rate (FPR)')
```

[ ]:
```
Text(0.5, 0, 'False Positive Rate (FPR)')
```



[ ]:
```
print(fpr, tpr)
```

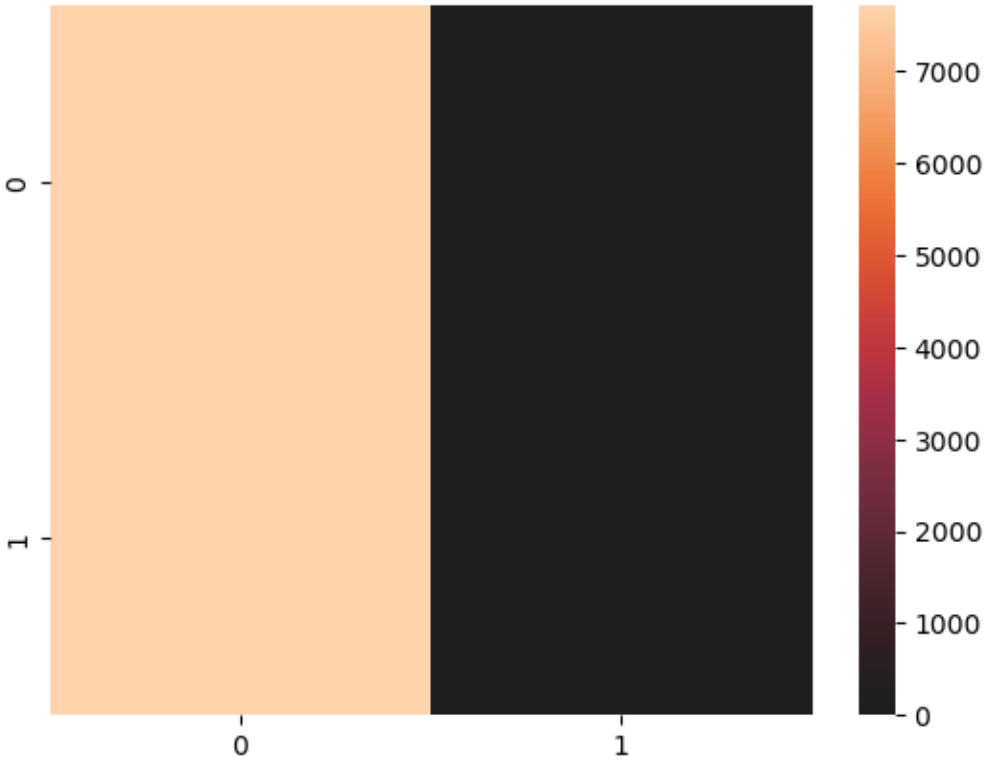
```
[0.         0.         0.         … 0.87624854 0.87624854 1.        ]
[0.00000000e+00 1.29483361e-04 6.47416807e-04 … 9.99870517e-01
 1.00000000e+00 1.00000000e+00]
```

DummyClassifier from scikit-learn, which provides a simple strategy for generating predictions.

[ ]:
```
# import DummyClassifier
from sklearn.dummy import DummyClassifier

# create a new instance of the classifier
dummy = DummyClassifier(strategy='most_frequent')

# fit the model on the training data
dummy.fit(X_train, y_train)
```

```
# predict on the test data
y_pred_dummy = dummy.predict(X_test)

# evaluate the model
print('DummyClassifier Test Score is : ' , dummy.score(X_test, y_test))

# calculate and print the confusion matrix
CM_dummy = confusion_matrix(y_test, y_pred_dummy)
sns.heatmap(CM_dummy, center=True)
plt.show()
print('Confusion Matrix is\n', CM_dummy)
```

```
DummyClassifier Test Score is :  0.49954639709694143
```




```
Confusion Matrix is
 [[7709    0]
 [7723    0]]
```