A PROJECT REPORT

ON

Rainfall Prediction - Weather Forecasting

Submitted by

HIMAJA. I

ACKNOWLEDGMENT

I would like to express my sincere gratitude to my mentors from DataTrained for helping me work on this project. Their timely and valuable inputs have helped me in the completion of this project successfully.

Finally, I would like to thank my family who have always been a great support to me and have been very helpful in various stages of project completion. Without their support it would not be possible to pursue my PGcourse and internship.

References:

- https://www.kaggle.com/datasets/jsphyg/weather-dataset-rattle-package
- Definitions adapted from http://www.bom.gov.au/climate/dwo/IDCJDW0000.shtml
- Datasource:
- http://www.bom.gov.au/climate/dwo/ and http://www.bom.gov.au/climate/data.
- https://goweatherforecast.com/news/importance-of-weather-forecasting-235

TABLE OF CONTENTS:

INTRODUCTION

Predicting rainfall is an important step in generating data for Weather forecasting. Rainfall predictions are a key process for providing Weather forecasting assessments with inputs to provide information people and organizations can use to reduce weather-related losses and enhance societal benefits, including protection of life and property, public health and safety, and support of economic prosperity and quality of life. Machine Learning (ML) can be helpful in overcoming such issues; for example, ML can be used to predict rainfall and apply it to foresee crop health and yield. Predictive analysis is a subset of data mining that forecasts future probabilities and patterns. Forecasting could be applied in air traffic, severe weather alerts, marine, agriculture, utility companies, private sector and military application.

Business Problem Framing:

Weather forecasting is the application of science and technology to predict the conditions of the atmosphere for a given location and time. Weather forecasts are made by collecting quantitative data about the current state of the atmosphere at a given place and using meteorology to project how the atmosphere will change.

Problem Definition: Design a predictive model with the use of machine learning algorithms to forecast whether or not it will rain tomorrow in Australia.

Objective: To predict two things

1) Design a predictive model with the use of machine learning algorithms to forecast whether or not it will rain tomorrow.

2) Design a predictive model with the use of machine learning algorithms to predict how much rainfall could be there.

Conceptual Background of the Domain Problem:

The weather has a great impact on various aspects of human life. So, efforts have been made for years to improve the accuracy of weather forecasting to ensure a better life.

What is weather forecasting?

Weather forecasting refers to the process in which science and technology are applied to predict the conditions of the atmosphere for a given location and time. Humans have been making an unofficial weather forecasting attempt millennia ago, and official weather forecasting dates back to the nineteenth century. Weather forecasting is done by collecting data on the current state of the atmosphere and applying a scientific understanding of atmospheric processes to predict atmospheric progression.

Rainfall forecasting has been around for years using traditional methods that employ statistical techniques to assess the correlation between rainfall, geographic coordinates (such as latitude and longitude), and other atmospheric factors (like pressure, temperature, wind speed, and humidity).

Key Importance of weather forecasting

The ultimate goal of weather forecasting is to protect human lives and property, improve health, safety, and economic prosperity. Weather Forecasting is crucial since it helps to determine future climate changes. With the use of latitude, we can determine the probability of snow and hail reaching the surface. We are able to identify the thermal energy from the sun that is exposed to a region.

On an everyday basis, many use weather forecasts to determine what to wear on a given day. Since outdoor activities are severely curtailed by heavy rain, snow and wind chill, forecasts can be used to plan activities around these events, and to plan ahead and survive them.

Review of literature:.

Rainfall remains one of the most influential meteorological parameters in many aspects of our daily lives. With effects ranging from damage to infrastructure in the event of a flood to disruptions in the transport network, the socio-economic impacts of rainfall are noteworthy. Floods and similar extreme events are consequences of climate change that are expected to occur more frequently and have catastrophic effects in years to come. More interestingly, recent studies have highlighted that weather conditions can potentially increase air pollution (another major topic of discourse alongside climate change in recent times) in winter and summer periods. It is pertinent to reiterate that increased air pollution results in health conditions such as asthma and similar problems related to the lungs. Therefore, as a mitigation approach, many studies have investigated and proposed rainfall forecasting techniques in preparation for any eventuality. However, in order to enhance human mobility activities and enhance agriculture and industrial development, these approaches must provide efficient and timely predictions.

Rainfall Prediction is the application area of data science and machine learning to predict the state of the atmosphere. It is important to predict the rainfall intensity for effective use of water resources and crop production to reduce mortality due to flood and any disease caused by rain.

Rainfall forecasting has gained utmost research relevance in recent times due to its complexities and persistent applications such as flood forecasting and monitoring of pollutant concentration levels, among others. Existing models use complex statistical models that are often too costly, both computationally and budgetary, or are not applied to downstream applications. Therefore, approaches that use Machine Learning algorithms in conjunction with time-series data are being explored as an alternative to overcome these drawbacks..

Motivation for the Problem Undertaken:

Artificial intelligence helps in understanding past weather models and this can make decision-making faster. The predictions of extreme weather phenomenon can be of tremendous value in minimizing the damage caused, and plan ahead to prevent casualties. The AI system can make more accurate short-term predictions, including for critical storms and floods. Climate change is making it harder to anticipate adverse weather conditions, as the frequency and severity of heavy rain increases, which researchers believe will lead to both significant material damage and death. To this end, this study presents a comparative analysis using simplified rainfall estimation models based on conventional Machine Learning algorithms to forecast whether or not it will rain tomorrow and to predict how much rainfall could be there.

ANALYTICAL PROBLEM FRAMING

Mathematical/ Analytical Modelling of the Problem:

The main thing that I found in problem statement is, the data given for us is supervised data. The problem statement contains both continuous and categorical data. I have performed both univariate and bivariate analysis to analyze these values using different plots like pie plot, count plot, distribution plot, factor plot etc. These plots give better pattern for analyzing the data. In this project I have done various mathematical and statistical analysis such as describing the statistical summary of the columns in which I found that the count is not same for all the columns which means null values present. Since the dataset contains object data type, I used label encoding method to convert the object data into numerical data. Checked for correlation between the features and visualized it using heat map for both the target variables separately and created models individually for each of the target variables.

Data Sources and their formats:

- The Dataset contains about 10 years of daily weather observations of different locations in Australia. The dataset is provided by Datatrained in the project portal which is in the CSV format.
- I have used this CSV for the processing.

- The dataset contains 8425 rows and 23 columns which is comprised of categorical columns. The datset has most of the features in float data type and some in the object data type.

- While describing the data I found skewness and outliers present in some of the columns. Since some of these columns are categorical they need not be removed.

Hardware & Software Requirements & Tools Used:

Hardware required:

- Processor: core i5 or above
- RAM: 8 GB or above
- ROM/SSD: 250 GB or above

Software required:

- Anaconda 3- language used
- Python 3

Data Analysis

- Firstly, I have uploaded the dataset in the Jupyter notebook.
- Then I have imported the necessary libraries and dataset.

Libraries:
The important libraries that I have used for this project are below.

```
# Importing the necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os
%matplotlib inline
import warnings
warnings.simplefilter("ignore")
warnings.filterwarnings("ignore")
```

- **import numpy as np:**

It is defined as a Python package used for performing the various numerical computations and processing of the multidimensional and

single dimensional array elements. The calculations using Numpy arrays are faster than the normal Python array.

- **import pandas as pd:**

Pandas is a Python library that is used for faster data analysis, data cleaning and data pre-processing. The data-frame term is comingfrom Pandas only.

- **import matplotlib.pyplot as plt and import seaborn as sns:**

Matplotlib and Seaborn acts as the backbone of data visualization through Python.

Matplotlib: It is a Python library used for plotting graphs with the help of other libraries like Numpy and Pandas. It is a powerful tool for visualizing data in Python. It is used for creating statistical interferences and plotting 2D graphs of arrays.

Seaborn: It is also a Python library used for plotting graphs with the help of Matplotlib, Pandas, and Numpy. It is built on the roof of Matplotlib and is considered as a superset of the Matplotlib library.It helps in visualizing univariate and bivariate data.

- **import warnings**

The warnings module was introduced in PEP 230 as a way to warn programmers about changes in language or library features in anticipation of backwards incompatible changes coming with Python 3.0. Since warnings are not fatal, a program may encounter the same warn-able situation many times in the course of running. Using “ignore” never displays warnings which match

➢ I have imported the given dataset from jupyter notebook as follows

```
#Importing the dataset
df = pd.read_csv('weatherAUS.csv')
df
```

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	...	Humidity9am	Humidity3pm
0	2008-12-01	Albury	13.4	22.9	0.6	NaN	NaN	W	44.0	W	...	71.0	22.0
1	2008-12-02	Albury	7.4	25.1	0.0	NaN	NaN	WNW	44.0	NNW	...	44.0	25.0
2	2008-12-03	Albury	12.9	25.7	0.0	NaN	NaN	WSW	46.0	W	...	38.0	30.0
3	2008-12-04	Albury	9.2	28.0	0.0	NaN	NaN	NE	24.0	SE	...	45.0	16.0
4	2008-12-05	Albury	17.5	32.3	1.0	NaN	NaN	W	41.0	ENE	...	82.0	33.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
8420	2017-06-21	Uluru	2.8	23.4	0.0	NaN	NaN	E	31.0	SE	...	51.0	24.0

- Checked the dimension of the dataset.

```
#Checking the shape of the dataset
df.shape
```

```
(8425, 23)
```

The dataset has 8425 rows and 23 columns

- Checked the columns in the dataset

```
#Displaying the columns
df.columns
```

```
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')
```

Dataset Description:

`Date` - The date of observation

`Location` -The common name of the location of the weather station

`MinTemp` -The minimum temperature in degrees celsius

`MaxTemp` -The maximum temperature in degrees celsius

`Rainfall` -The amount of rainfall recorded for the day in mm

`Evaporation` -The so-called Class A pan evaporation (mm) in the 24 hours to 9am

`Sunshine` -The number of hours of bright sunshine in the day.

`WindGustDir`- The direction of the strongest wind gust in the 24 hours to midnight

`WindGustSpeed` -The speed (km/h) of the strongest wind gust in the 24 hours to midnight

`WindDir9am` -Direction of the wind at 9am

`WindDir3pm` -Direction of the wind at 3pm

`WindSpeed9am` -Wind speed (km/hr) averaged over 10 minutes prior to 9am

`WindSpeed3pm` -Wind speed (km/hr) averaged over 10 minutes prior to 3pm

`Humidity9am` -Humidity (percent) at 9am

`Humidity3pm` -Humidity (percent) at 3pm

`Pressure9am` -Atmospheric pressure (hpa) reduced to mean sea level at 9am

`Pressure3pm` -Atmospheric pressure (hpa) reduced to mean sea level at 3pm

`Cloud9am` - Fraction of sky obscured by cloud at 9am.

`Cloud3pm` -Fraction of sky obscured by cloud

`Temp9am` -Temperature (degrees C) at 9am

`Temp3pm` -Temperature (degrees C) at 3pm

`RainToday` -Boolean: 1 if precipitation (mm) in the 24 hours to 9am exceeds 1mm, otherwise 0

`RainTomorrow` -The amount of next day rain in mm. Used to create response variable . A kind of measure of the "risk".

- Checked the information of the data frame using info()

```
#Getting an overall overview of the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8425 entries, 0 to 8424
Data columns (total 23 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Date           8425 non-null   object
 1   Location       8425 non-null   object
 2   MinTemp        8350 non-null   float64
 3   MaxTemp        8365 non-null   float64
 4   Rainfall       8185 non-null   float64
 5   Evaporation    4913 non-null   float64
 6   Sunshine       4431 non-null   float64
 7   WindGustDir    7434 non-null   object
 8   WindGustSpeed  7434 non-null   float64
 9   WindDir9am     7596 non-null   object
 10  WindDir3pm     8117 non-null   object
 11  WindSpeed9am   8349 non-null   float64
 12  WindSpeed3pm   8318 non-null   float64
 13  Humidity9am    8366 non-null   float64
 14  Humidity3pm    8323 non-null   float64
 15  Pressure9am    7116 non-null   float64
 16  Pressure3pm    7113 non-null   float64
 17  Cloud9am       6004 non-null   float64
 18  Cloud3pm       5970 non-null   float64
 19  Temp9am        8369 non-null   float64
 20  Temp3pm        8329 non-null   float64
 21  RainToday      8185 non-null   object
 22  RainTomorrow   8186 non-null   object
dtypes: float64(16), object(7)
memory usage: 1.5+ MB
```

- Checked the data types of the features.

```
#Checking the data types of the dataset
df.dtypes
```

```
Date             object
Location         object
MinTemp         float64
MaxTemp         float64
Rainfall        float64
Evaporation     float64
Sunshine        float64
WindGustDir      object
WindGustSpeed   float64
WindDir9am       object
WindDir3pm       object
WindSpeed9am    float64
WindSpeed3pm    float64
Humidity9am     float64
Humidity3pm     float64
Pressure9am     float64
Pressure3pm     float64
Cloud9am        float64
Cloud3pm        float64
Temp9am         float64
Temp3pm         float64
RainToday        object
RainTomorrow     object
dtype: object
```

- Checked the number of unique values present in the dataset.

```
#checking the unique numbers of the dataset
df.nunique()
```

```
Date            3004
Location          12
MinTemp          285
MaxTemp          331
Rainfall         250
Evaporation      116
Sunshine         140
WindGustDir       16
WindGustSpeed     52
WindDir9am        16
WindDir3pm        16
WindSpeed9am      34
WindSpeed3pm      35
Humidity9am       90
Humidity3pm       94
Pressure9am      384
Pressure3pm      374
Cloud9am           9
Cloud3pm           9
Temp9am          304
Temp3pm          328
RainToday          2
RainTomorrow       2
dtype: int64
```

- I have checked for null values and found null values present in the dataset

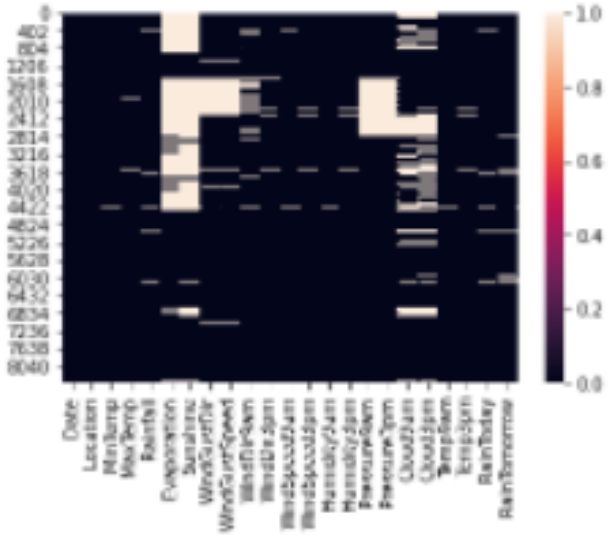
```
#Checkig for null values in the dataset
df.isnull().sum()
```

```
Date              0
Location          0
MinTemp          75
MaxTemp          60
Rainfall        240
Evaporation    3512
Sunshine       3994
WindGustDir     991
WindGustSpeed   991
WindDir9am      829
WindDir3pm      308
WindSpeed9am     76
WindSpeed3pm    107
Humidity9am      59
Humidity3pm     102
Pressure9am    1309
Pressure3pm    1312
Cloud9am       2421
Cloud3pm       2455
Temp9am          56
Temp3pm          96
RainToday       240
RainTomorrow    239
dtype: int64
```

- I have visualized it using heat map as below

```
#Visualizing the null values using a heatmap
sns.heatmap(df.isnull())
```

```
<AxesSubplot:>
```



EDA Concluding Remark

Observations

- We can see that our dataset comprises of 8425 rows and 23 columns.
- By observing the dataset we can say that we have two target

variables.

- The values present in our target column 'RainTomorrow' have data in categorical data type. So we can use Classification model in this case. While the other target variable 'Rainfall' is in continuous data making it a regression problem.
- EDA and preprocessing is same for prediction of both the targets.
- In the above output we can see that there are missing values in many columns and few even have approximately 50% of the data empty. Such columns need to be removed since it provides no insights.
- Though we have the following columns with nearly 50% of missing values... they are important features. Hence we cannot drop them
    - Sunshine has 3994 null values which says 47.4% of the data is missing
    - Evaporation has 3512 null values which says 41.6% of the data is missing
- There are null values in the target columns too.
- All the null values need to be treated using respective imputation methods

Pre-processing Pipeline

- I have used Imputation methods to fill those null values.

Filling null values

Filling the null values for those columns having numerical(float) type data using mean and filling the null values for those columns having categorical(object) type data using mode

```
#Filling null values in MinTemp with it's mean
df['MinTemp'] = df['MinTemp'].fillna(df['MinTemp'].mean())

#Filling null values in MaxTemp with it's mean
df['MaxTemp'] = df['MaxTemp'].fillna(df['MaxTemp'].mean())

#Fillinf null values in Rainfall with it's mode, as most of the rows have 0 rainfall.
df['Rainfall'] = df['Rainfall'].fillna(df['Rainfall'].mode()[0])

#Filling null values in Evaporation with it's mean
df['Evaporation'] = df['Evaporation'].fillna(df['Evaporation'].mean())

#Filling null values in Sunshine with it's mean
df['Sunshine'] = df['Sunshine'].fillna(df['Sunshine'].mean())

#Filling null values in WindGustDir with it's mode
df['WindGustDir'] = df['WindGustDir'].fillna(df['WindGustDir'].mode()[0])

#Filling null values in WindGustSpeed with it's mean
df['WindGustSpeed'] = df['WindGustSpeed'].fillna(df['WindGustSpeed'].mean())

#Filling null values in WindDir9am with it's mode
df['WindDir9am'] = df['WindDir9am'].fillna(df['WindDir9am'].mode()[0])

#Filling null values in WindDir3pm with it's mode
df['WindDir3pm'] = df['WindDir3pm'].fillna(df['WindDir3pm'].mode()[0])

#Filling null values in WindSpeed9am with it's mean
df['WindSpeed9am'] = df['WindSpeed9am'].fillna(df['WindSpeed9am'].mean())

#Filling null values in WindSpeed3pm with it's mean
df['WindSpeed3pm'] = df['WindSpeed3pm'].fillna(df['WindSpeed3pm'].mean())

#Filling null values in Humidity9am with it's mean
df['Humidity9am'] = df['Humidity9am'].fillna(df['Humidity9am'].mean())

#Filling null values in Humidity3pm with it's mean
df['Humidity3pm'] = df['Humidity3pm'].fillna(df['Humidity3pm'].mean())

#Filling null values in Pressure9am with it's mean
df['Pressure9am'] = df['Pressure9am'].fillna(df['Pressure9am'].mean())

#Filling null values in Pressure3pm with it's mean
df['Pressure3pm'] = df['Pressure3pm'].fillna(df['Pressure3pm'].mean())

#Filling null values in Cloud9am with it's mean
df['Cloud9am'] = df['Cloud9am'].fillna(df['Cloud9am'].mean())

#Filling null values in Cloud3pm with it's mean
df['Cloud3pm'] = df['Cloud3pm'].fillna(df['Cloud3pm'].mean())

#Filling null values in Temp9am with it's mean
df['Temp9am'] = df['Temp9am'].fillna(df['Temp9am'].mean())

#Filling null values in Temp3pm with it's mean
df['Temp3pm'] = df['Temp3pm'].fillna(df['Temp3pm'].mean())

#Filling null values in Temp3pm with it's mean
df['Temp3pm'] = df['Temp3pm'].fillna(df['Temp3pm'].mean())

#Filling null values in RainToday with it's mode
df['RainToday'] = df['RainToday'].fillna(df['RainToday'].mode()[0])

#Filling null values in RainTomorrow with it's mode
df['RainTomorrow'] = df['RainTomorrow'].fillna(df['RainTomorrow'].mode()[0])
```

- Next I have converted the date feature to datetime format.

- Extracted the date, month and year from the date feature to individual columns.

- Dropped the Date feature as it is redundant.

```
#Converting date column to datetime format
df['Date'] = pd.to_datetime(df['Date'])
```

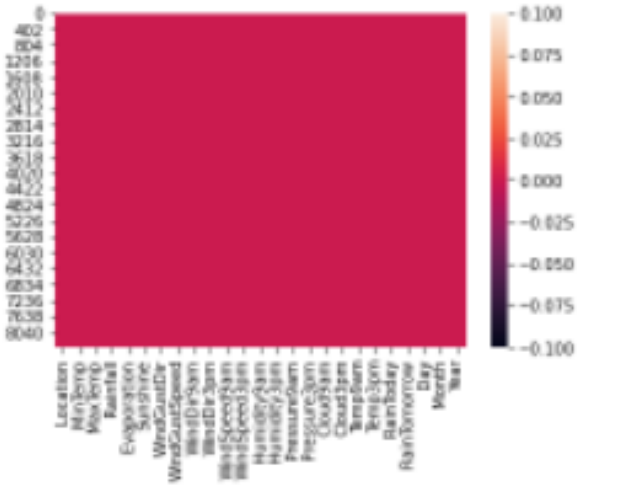
```
#Extracting Day from Date_of_journey column
df['Day'] = pd.to_datetime(df.Date,format="%d/%m/%Y").dt.day

#Extracting Month from Date_of_journey column
df['Month'] = pd.to_datetime(df.Date, format="%d/%m/%Y").dt.month

#Extracting Year from Date_of_journey column
df['Year'] = pd.to_datetime(df.Date, format="%d/%m/%Y").dt.year
```

```
#Dropping the Date column as it is redundant
df.drop(columns=['Date'],axis=1,inplace=True)
```

- Checked the null values and plotted the same using a heatmap to confirm there are no null values in the dataset.

```
#Visualizing the null values using a heatmap
sns.heatmap(df.isnull())
```

```
<AxesSubplot:>
```



This shows there are no more null values in the dataset.

- Described the data using describe() to view the statistical summary of the dataset.

Statistical summary of the Dataset

```
# Viewing the statistical summary using describe method
df.describe()
```

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressur
count	8425.000000	8425.000000	8425.000000	8425.000000	8425.000000	8425.000000	8425.000000	8425.000000	8425.000000	8425.000000	8425.0
mean	13.193305	23.859976	2.725982	5.389395	7.632205	40.174469	13.847646	18.533662	67.822496	51.249790	1017.6
std	5.379488	6.114516	10.319872	3.852004	2.825451	13.776101	10.128579	9.704759	16.774231	18.311894	6.2
min	-2.000000	8.200000	0.000000	0.000000	0.000000	7.000000	0.000000	0.000000	10.000000	6.000000	989.8
25%	9.300000	19.300000	0.000000	4.000000	7.632205	31.000000	6.000000	11.000000	56.000000	39.000000	1014.0
50%	13.200000	23.300000	0.000000	5.389395	7.632205	40.174469	13.000000	19.000000	68.000000	51.000000	1017.6
75%	17.300000	28.000000	0.800000	5.389395	8.900000	48.000000	20.000000	24.000000	80.000000	63.000000	1021.3
max	28.500000	45.500000	371.000000	145.000000	13.900000	107.000000	63.000000	83.000000	100.000000	99.000000	1039.0

Observations :

The describe method shows statistical summary of the numerical columns only, it doesnot displays the categorical data. From the above table we can say that

- The count of all the columns is uniform which says there are no null values in the dataset.

- The Rainfall columns has mean value higher than the median (50% - 2nd quantile), so the distribution is skewed to right.

- Other columns have mean almost equal to median, which says the distribution of curve is normal

- The following columns have huge difference between the 75% (3rd quantile) and the max values, which shows the chance for presence of outliers

    - MinTemp
    - MaxTemp
    - Rainfall
    - Evaporation
    - Sunshine
    - WindGustSpeed
    - WindSpeed9am
    - WindSpeed3pm
    - Humidity9am
    - Humidity3pm
    - Temp9am
    - Temp3pm

- The following columns have no much difference between the 75% (3rd quantile) and the max values, which shows there are less chance for presence of outliers

    - Pressure9am
    - Pressure3pm

- Cloud9am
- Cloud3pm

- Checked the value count of each column. By using for loop value count function.

```
for col in df:
    print(col)
    print(df[col].value_counts())
    print("-"*100)
```

```
Location
Melbourne       1622
Williamtown     1230
PerthAirport    1204
Albury           907
Newcastle        822
CoffsHarbour     611
Brisbane         579
Penrith          482
Wollongong       474
Darwin           250
Adelaide         205
Uluru             39
Name: Location, dtype: int64
----------------------------------------------------------------------------------------------------
MinTemp
 13.193305    75
 12.000000    74
 13.200000    71
```

- Checked for the features with object type data and float type data using for loop and stored them in individual lists for better and easy plotting of graphs.

Checking the categorical columns and numerical columns

```
#checking categorical columns
cat_col=[]
for i in df.dtypes.index:
    if df.dtypes[i]=='object':
        cat_col.append(i)
print(cat_col)
```

```
['Location', 'WindGustDir', 'WindDir9am', 'WindDir3pm', 'RainToday', 'RainTomorrow']
```

```
# checking for numerical columns
num_col=[]
for i in df.dtypes.index:
    if df.dtypes[i]!='object':
        num_col.append(i)
print(num_col)
```

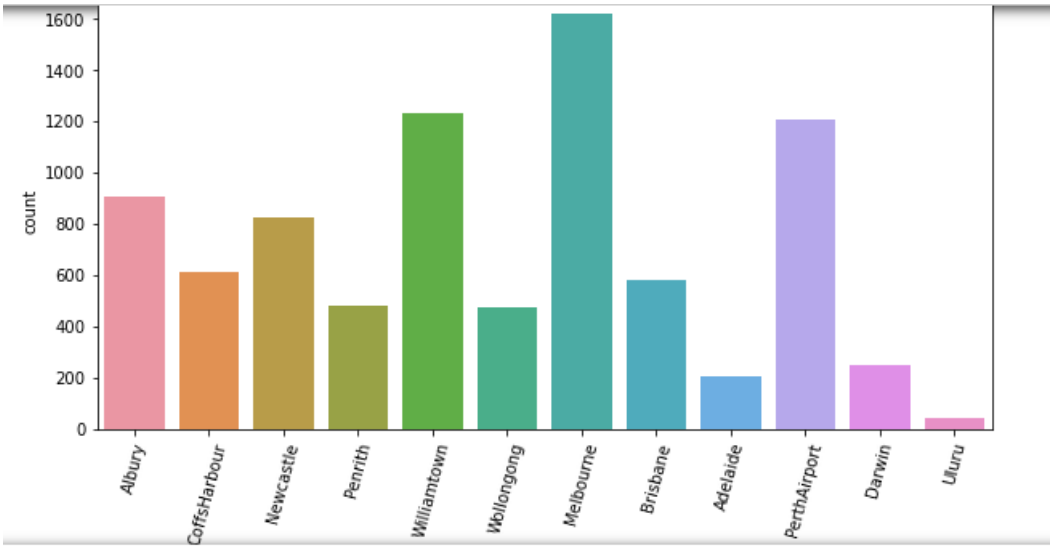
```
['MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine', 'WindGustSpeed', 'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am',
'Humidity3pm', 'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am', 'Temp3pm', 'Day', 'Month', 'Year']
```

- Visualized each feature using seaborn and matplotlib libraries by plotting count plot, distribution plot, Scatterplot, bar plot and factor plot.

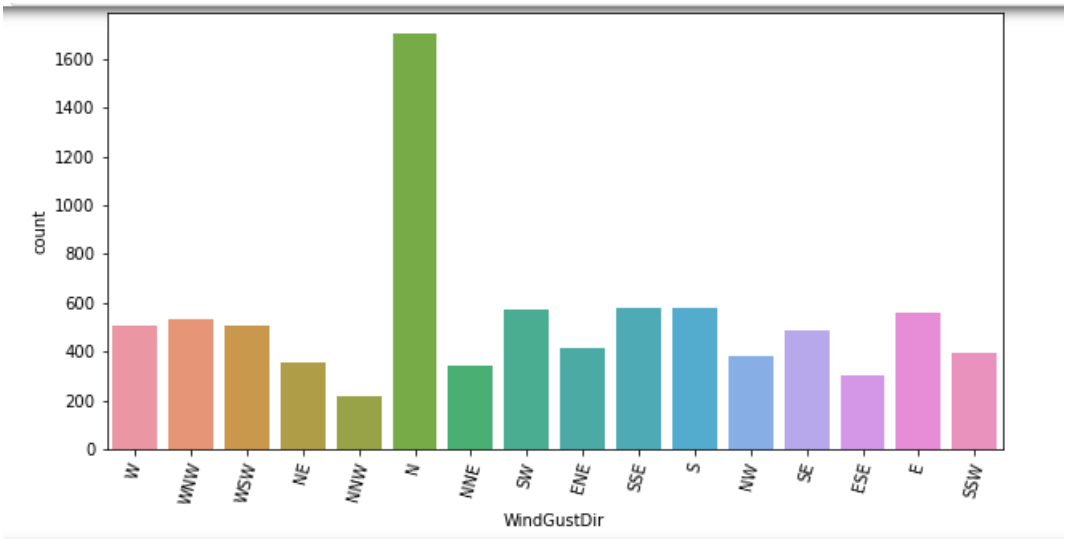
➢ Univariate Analysis

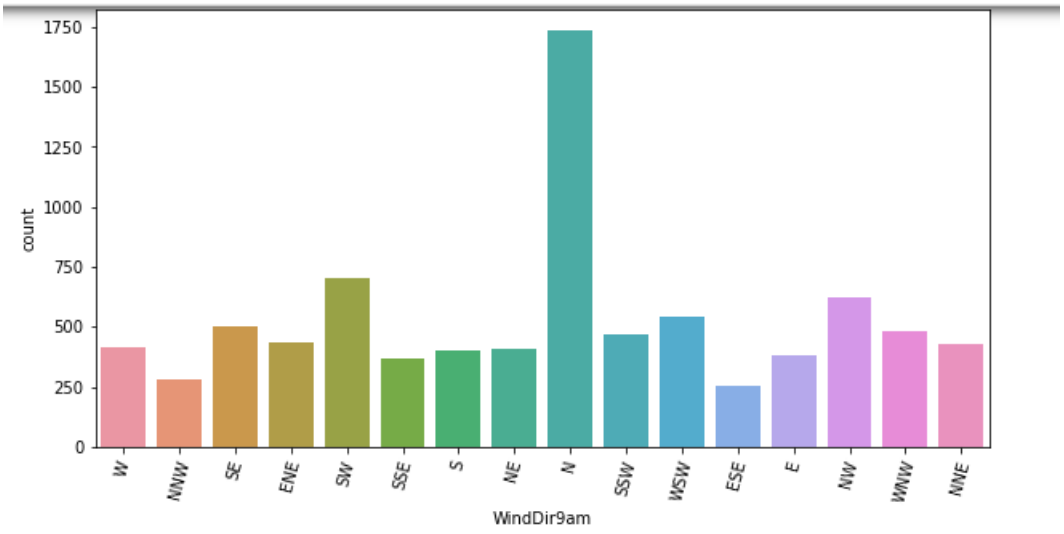
Visualizing the various categorical columns present in the dataset

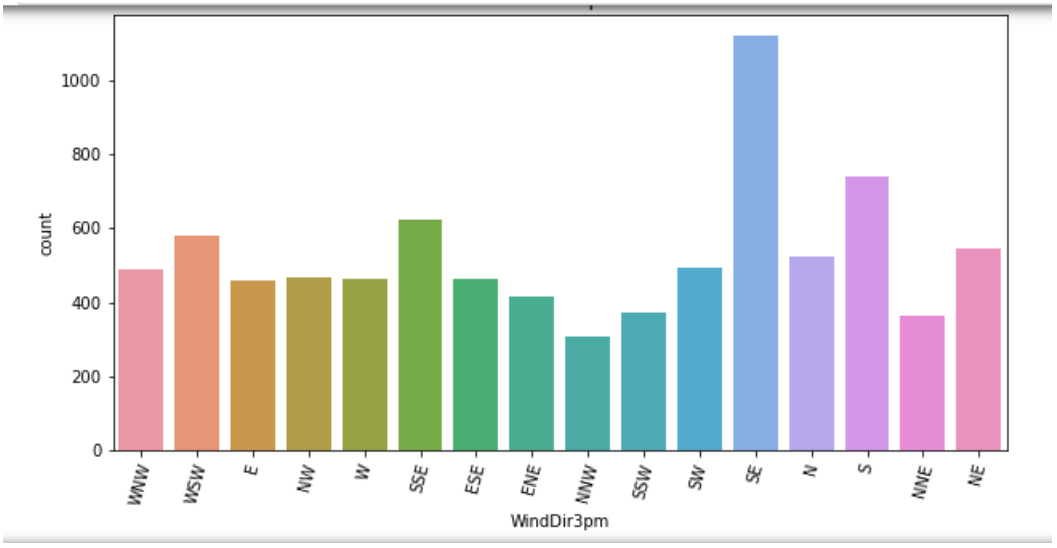
➢ **Plotting count plots of the categorical columns**

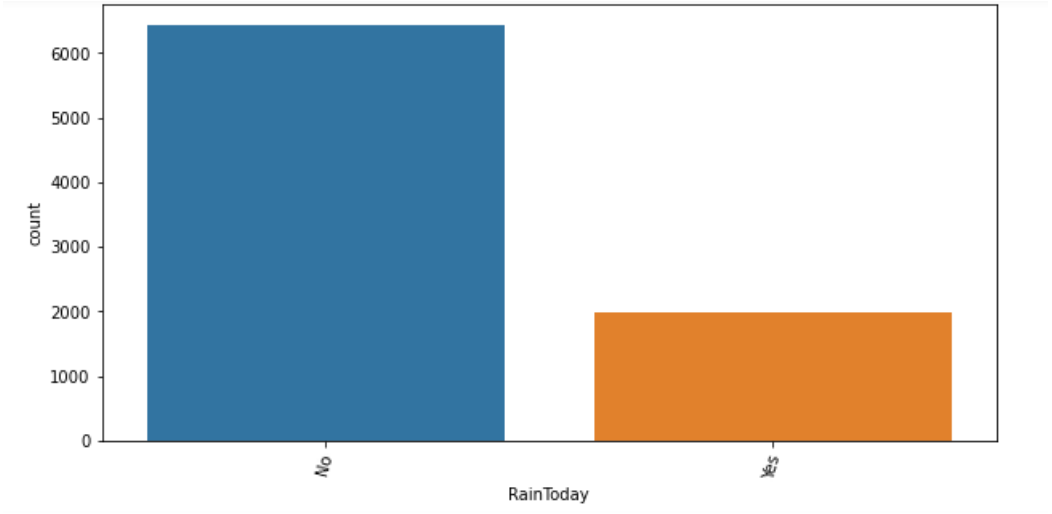


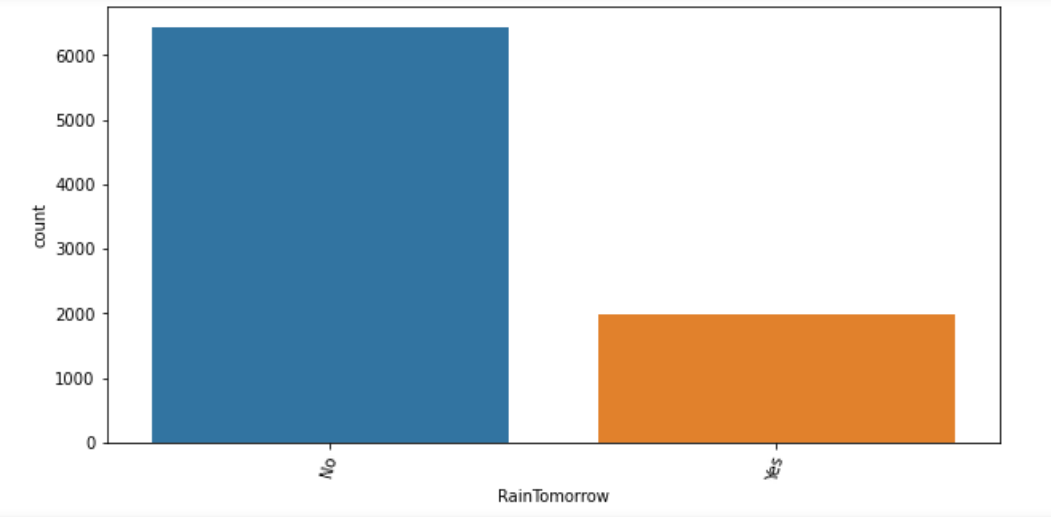
Location




1750
1500
1250
1000
750
500
250
0
count
W
NNW
SE
ENE
SW
SSE
S
NE
N
SSW
WSW
ESE
E
NW
WNW
NNE
WindDir9am


1000
800
600
400
200
0
count
WNW
WSW
E
NW
W
SSE
ESE
ENE
NNW
SSW
SW
SE
N
S
NNE
NE
WindDir3pm


6000
5000
4000
3000
2000
1000
0
count
No
Yes
RainToday

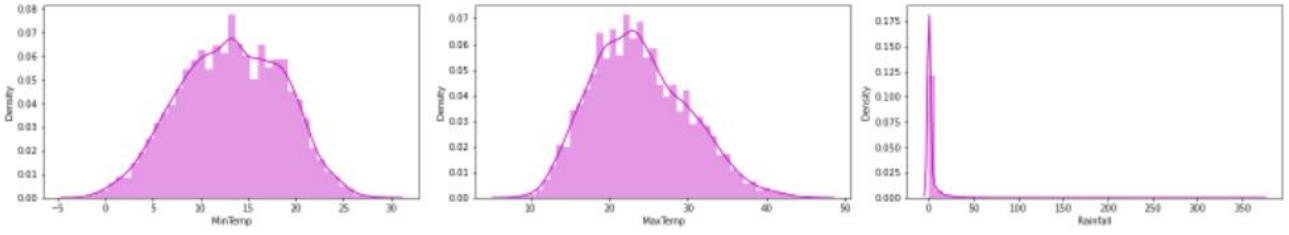


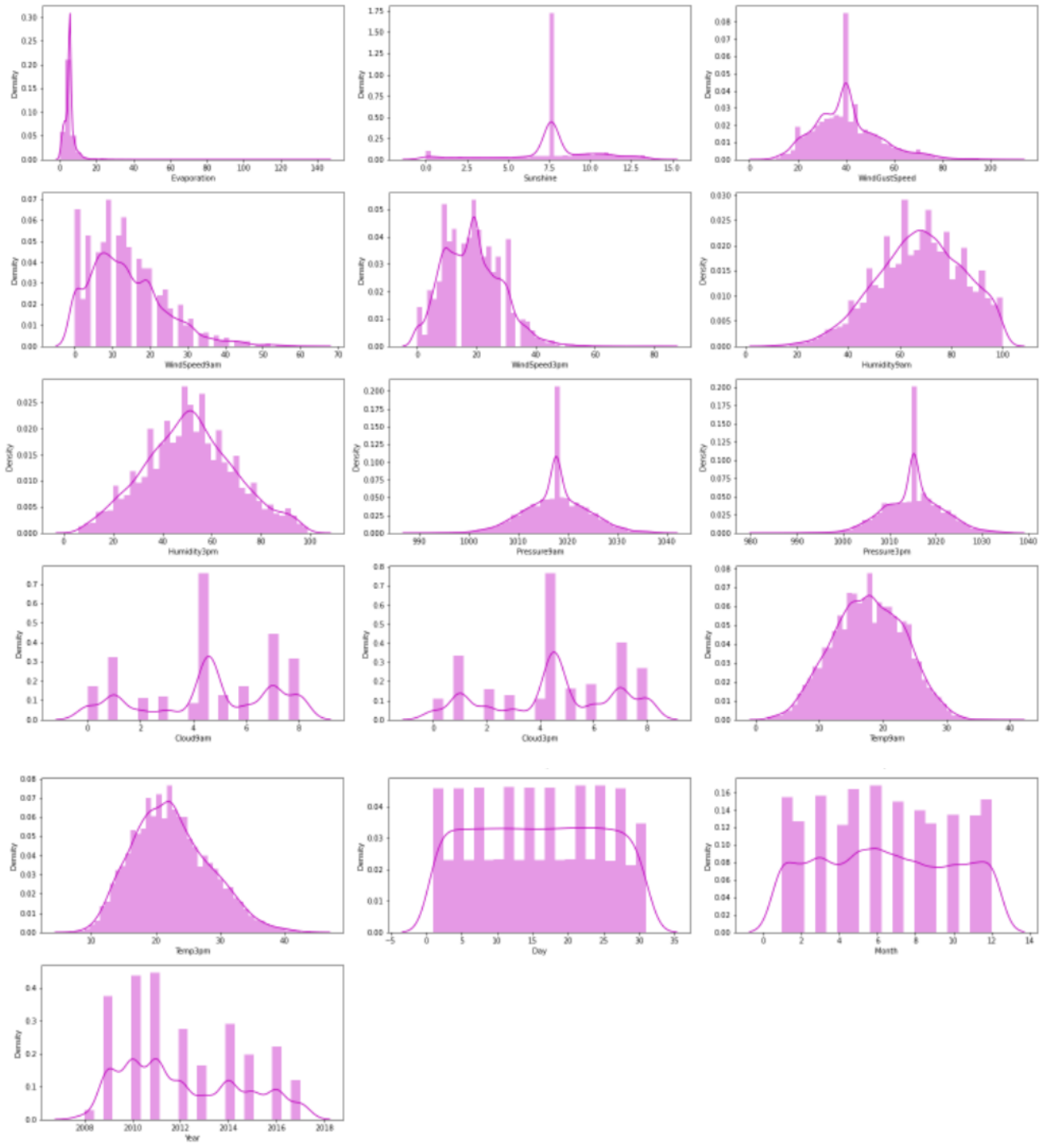
Observations

- Melbourne has highest count in the Location column, where as Uluru has the least count
- N has the highest count in the WindGustDir, where as the NNW has the least count
- N has the highest count in the WindDir9am, where as NNW has the least count
- SE has the highest count in the WindDir3pm, where as the NNW has the least count
- No RainToday has high count
- No RainTomorrow has high count

Visualizing the distribution of the numerical columns

➢ Plotting distplot for all the numerical columns





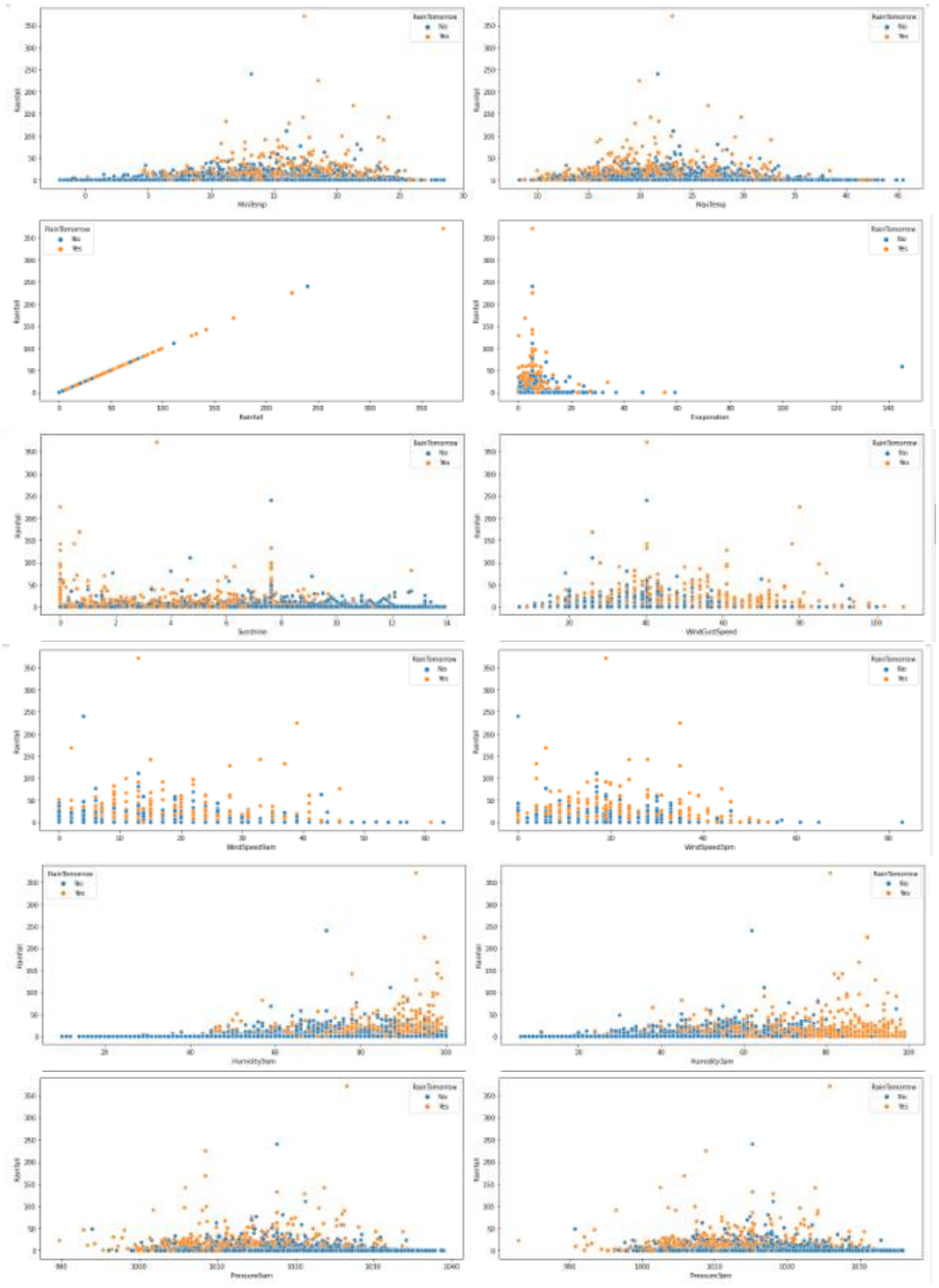
Observations

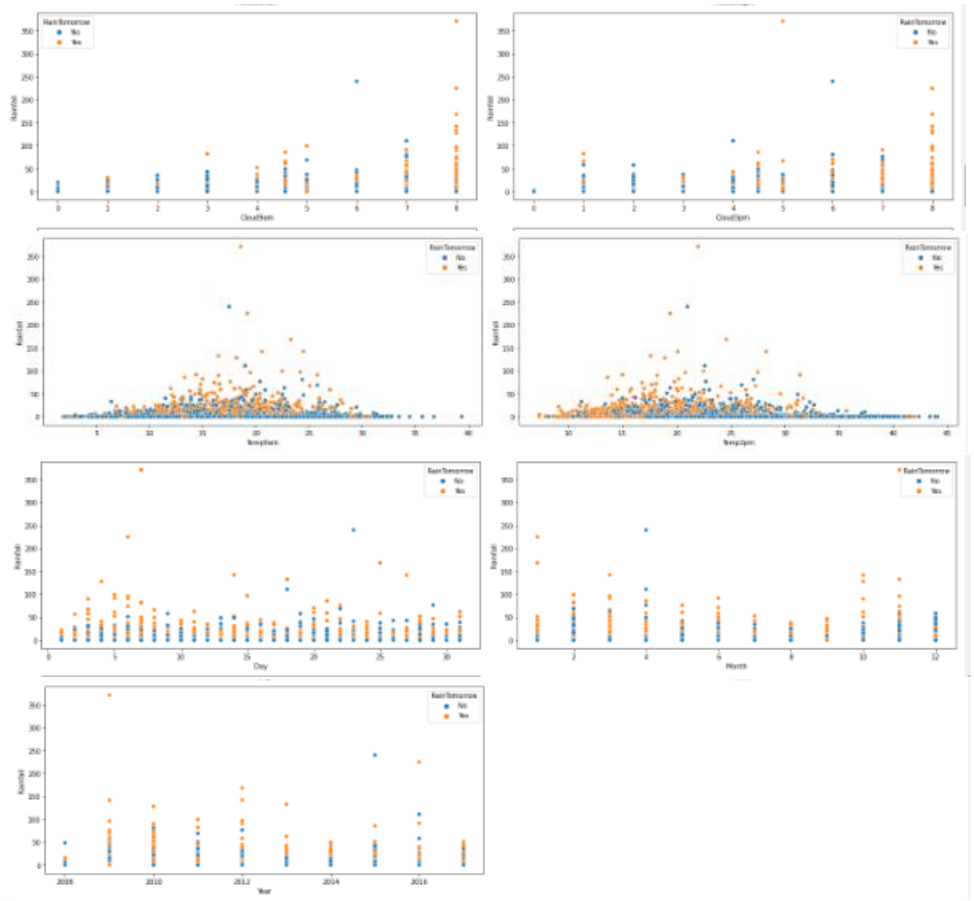
- We can see most of the features have normal distribution
- Some of the features are skewed to the right and left as expected , this confirms the presence of the skewness.

➢ Bivariate Analysis

Visualizing the relation between the features and the two target variables

➢ **Plotting Scatterplot for all the numerical columns with X as the feature , Y as ‘Rainfall’ and hue as ‘RainTomorrow’**





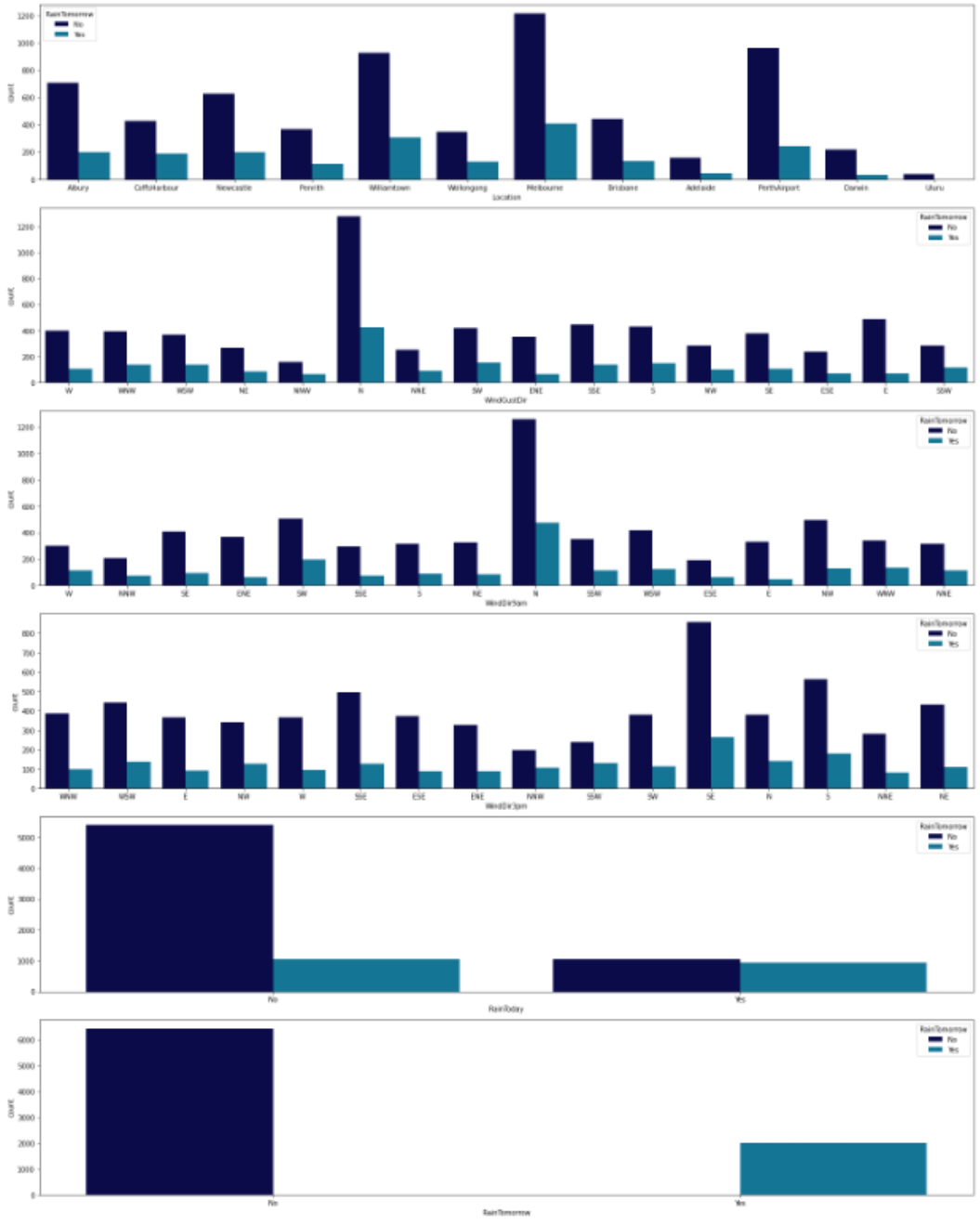
Observations

From the above plots we can say that

- As the MinTemp is between 10 and 20 it shows there is lower level of rainfall with high chance of no rainfall tomorrow
- As the MaxTemp is between 10 and 35 there is a lower level of rainfall with high chance of rainfall tomorrow
- Most of the rainfall is lower level and is densely populated at the lower level of evaporation. The more the evaporation, more the chance of rain tomorrow
- As the sunshine is lower there is more chance of rainfall and rain tomorrow
- When the wind speed is between 20 and 60 there is a chance of lower level rainfall with equal chance of rain tomorrow
- When the windspeed9am is between 5 and 30 there is a rainfall of lower level with high chance of rain tomorrow
- When the windspeed3pm is between 5 and 45 there is a rainfall of higher level with high chance of rain tomorrow
- When humidity9am is between 45 and 100 there is rainfall with higher chance of rain tomorrow increases as the humidity increases
- When humidity3pm is between 30 and 100 there is rainfall with higher chance of rain tomorrow increases as the humidity increases
- When the Pressure9am is between 1000 and 1030 there is rainfall with higher chance of no rain tomorrow as the pressure increases
- When the Pressure3pm is between 1000 and 1020 there is rainfall with higher chance of no rain tomorrow as the pressure increases
- If Cloud9am is higher there is higher rain fall and higher chance of rain tomorrow

- If Cloud3pm is higher there is higher rain fall and higher chance of rain tomorrow
- When Temp9am is between 10 and 30 there is a rainfall and higher chance of no rain tomorrow
- When Temp3pm is between 10 and 30 there is a rainfall and higher chance of no rain tomorrow
- There is more rainfall in the first 10 days of the month and there is high chance of rain tomorrow compared to other days
- There is high rain fall in the months of 2,3,4, and 10, 11 with high chance of rain tomorrow compared to other months
- The year 2012 has higher rainfall with higher chances of rain tomorrow than other years

➢ Plotting Countplot for all the categorical columns with X as the feature and hue as 'RainTomorrow'

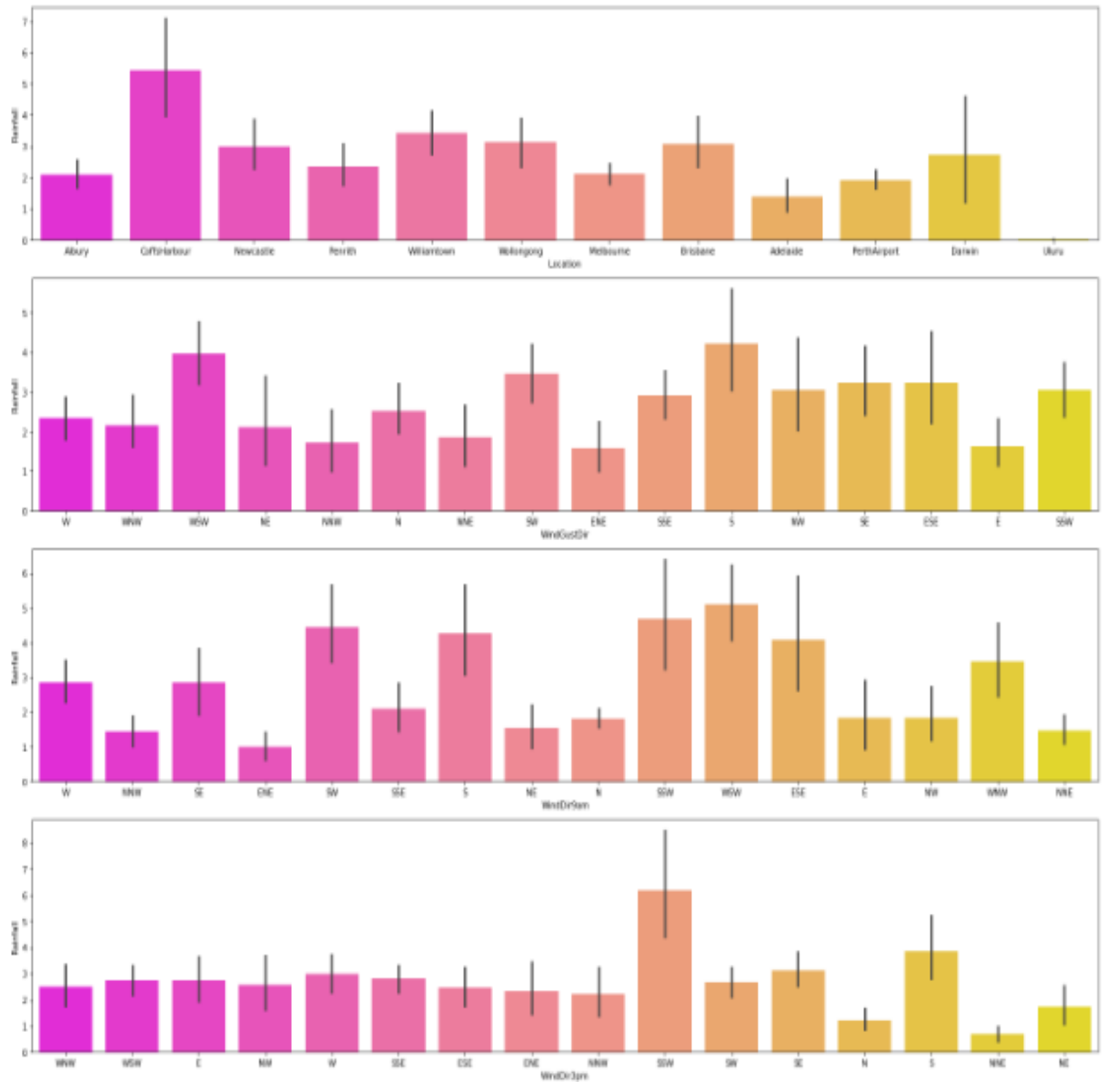


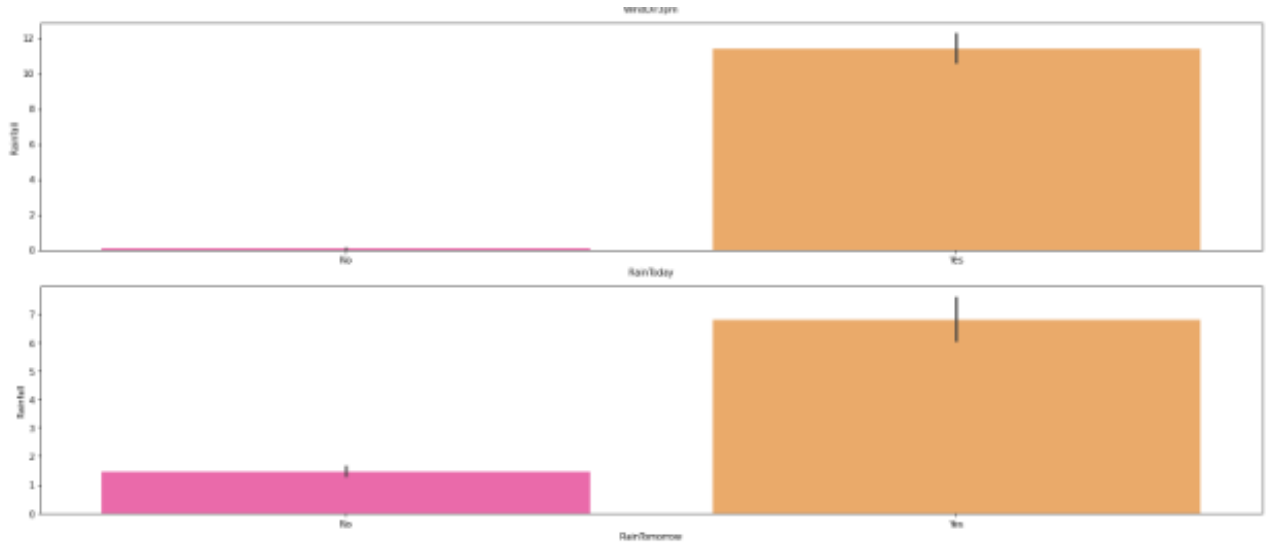
Observations

From the above plots we can say that

- Melbourne has high rainfall with lower chance of rain tomorrow
- Uluru has lower rainfall and there is no chance for rain tomorrow
- N has high rainfall in WindGustDir with higher chance of no rain tomorrow
- In WindGustDir NNW has lower rainfall with lowest chance of rain tomorrow
- N has high rain fall in WindDir9am with higher chance of no rain tomorrow
- SE has high rain fall in WindDir3pm with higher chance of no rain tomorrow
- There is high chance for no rain today with no rain today and no rain tomorrow

➢ Plotting barplot for all the categorical columns with X as the feature and Y as 'Rainfall'



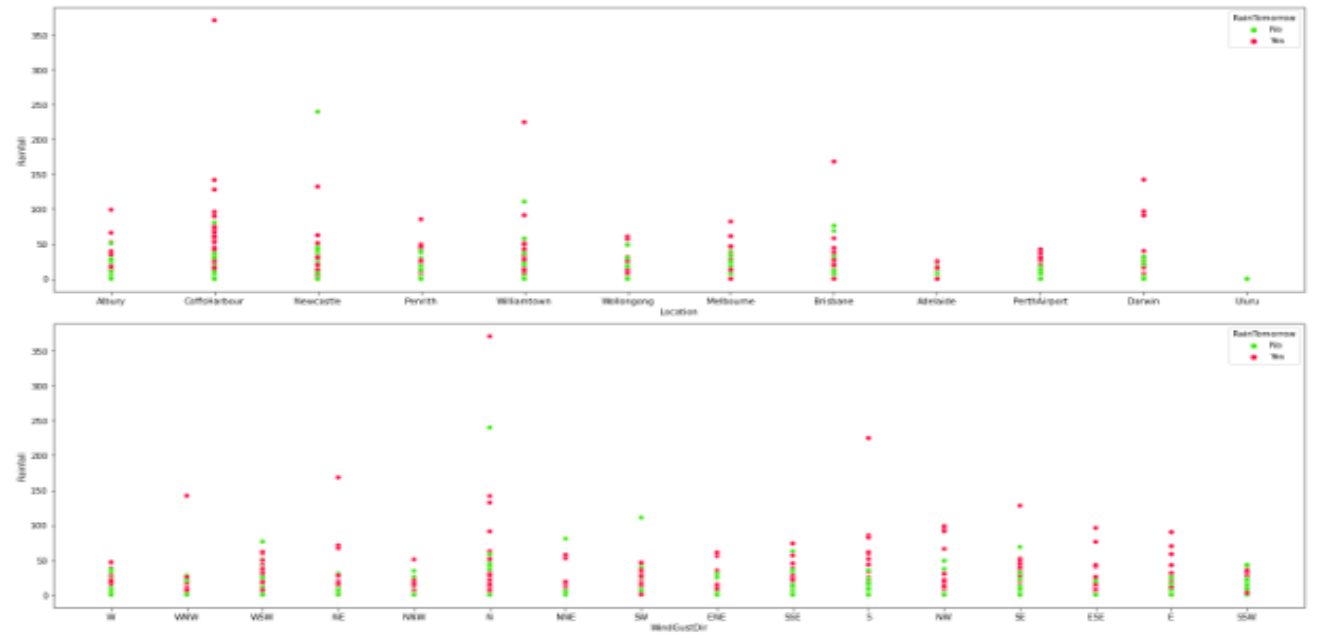


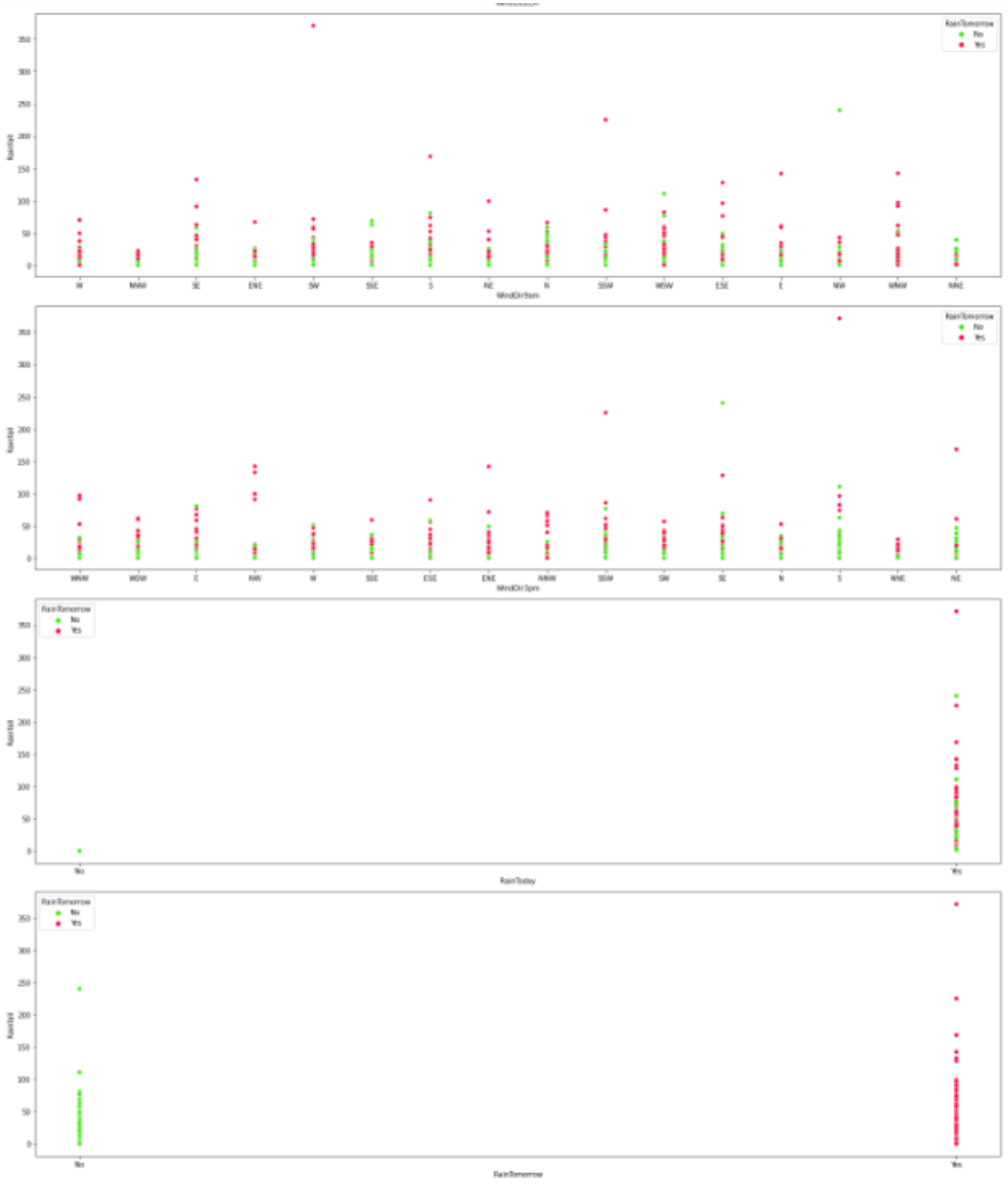
Observations

From the above plots we can say that

- CoffsHarbour has higher rainfall and uluru has lowest rainfall compared to others
- S has higher rainfall and E has lowestrainfall in WindGustDir
- Rain fall is highest in the WSW of WindDir9am and ENE has lowest rainfall
- Rain fall is highest in the SSW of WindDir3pm and NNE has lowest rainfall
- RainToday has higher chance of rainfall
- Rain tomorrow has higher chance of rainfall

➢ Plotting Scatterplot for all the categorical columns with X as the feature and Y as ‘Rainfall’ and hue as ‘RainTomorrow’





Observations

From the above plots we can say that

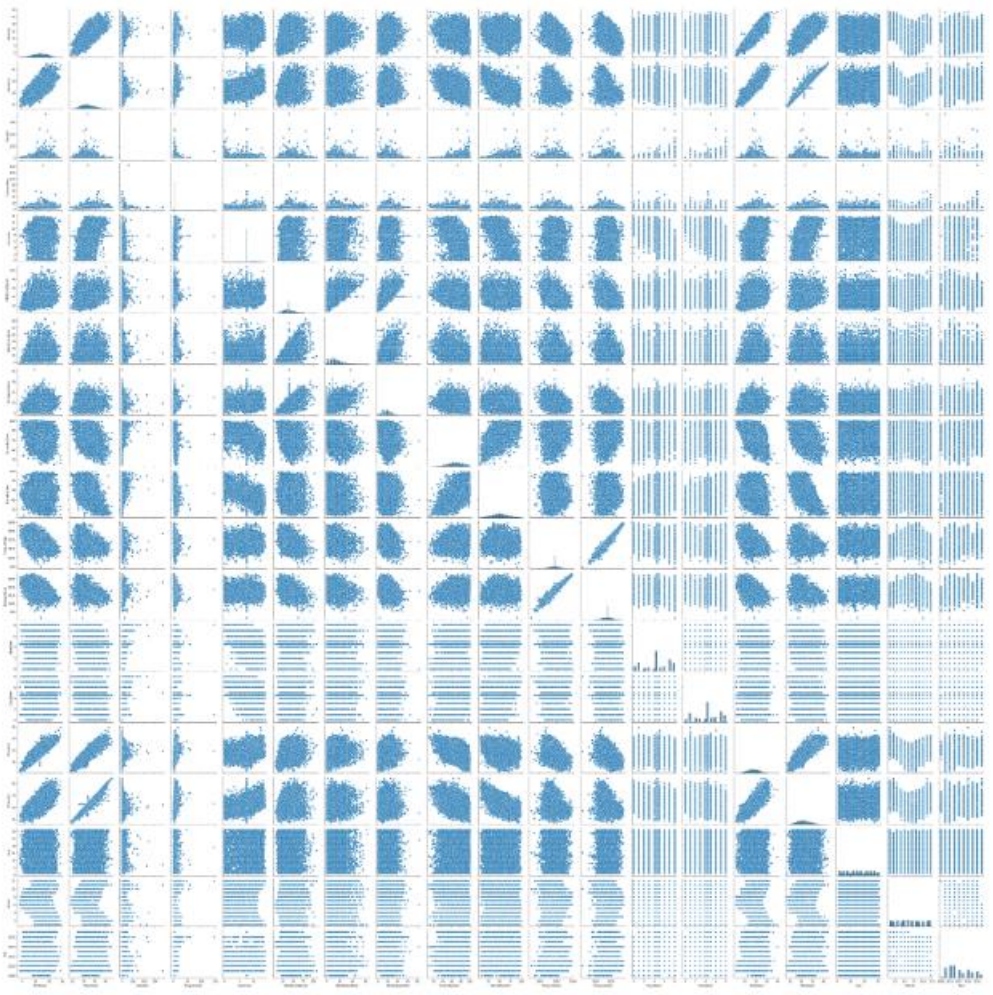
- CoffsHarbour has higher rainfall with higher chance of rain tomorrow and uluru has lowest rainfall with high chance of rain tomorrow compared to others
- N has higher rainfall with higher chance of higher chance of rain tomorrow and E has lowestrainfall in WindGustDir
- Rain fall is highest in the WSW in WindDir9am with higher chance of rain tomorrow and ENE has lowest rainfall

- Rain fall is highest in the SSW in WindDir3pm with higher chance of rain tomorrow and NNE has lowest rainfall with high chance of rain tomorrow
- RainToday has higher chance of rainfall
- Rain tomorrow has higher chance of rainfall

➢ Performed Multivariate analysis using pairplot method in seaborn.

```
#Using pair plot for multivariate analysis
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x2125018a100>
```

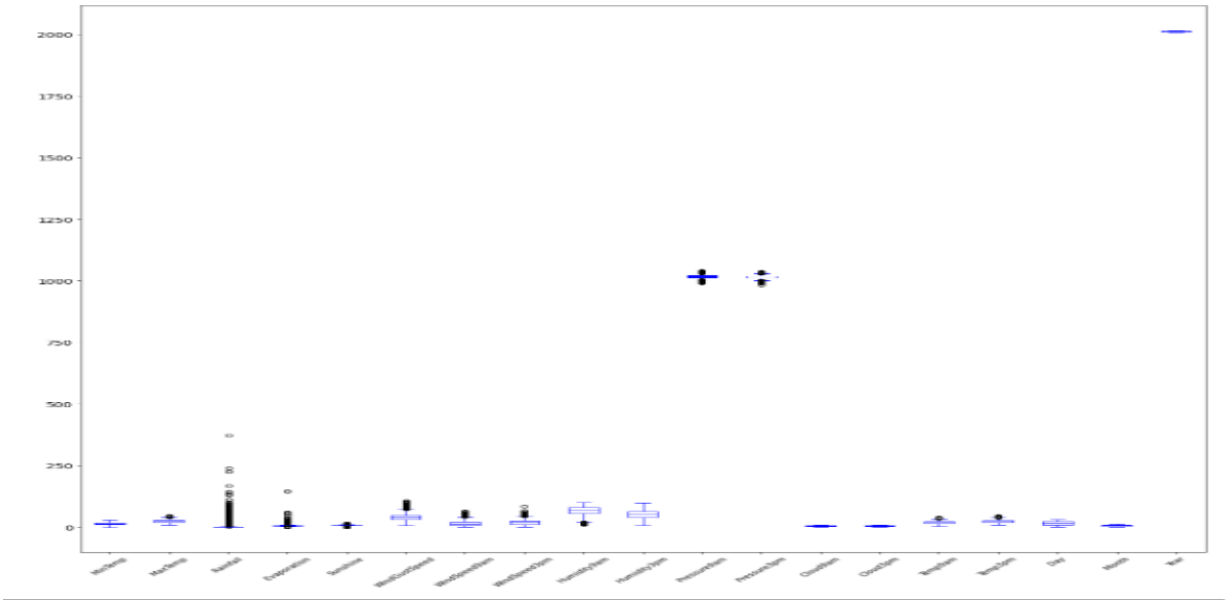


Observations

- Most of the pair plots are linear and outliers are observed in some of the features.

➢ Performed Data Cleaning for removing outliers.

```
#Checking for outliers
df.plot(kind='box',fontsize=14,figsize=(20,20),color='b')
plt.xticks(rotation=45,fontsize=10)
plt.show()
```



- Plotted Box plot of all the features to check for the outliers present in the dataset. I found out that there were outliers in all the columns except for the following columns
  - MinTemp,
  - Humidity3pm,
  - Cloud9am,
  - Cloud3pm,
  - Day,
  - Month,
  - Year

- After identifying the outliers I have added all the columns to a list and removed the outliers from them using ZScore method and IQR method.

- The ZScore method has a dataloss of 5.19% which is less than 10% and IQR method has a dataloss of 51%, which is too high. The ZScore method has showed less data loss than the IQR method so carried on with the dataframe created after removal of the outliers using ZScore method.

```
# Viewing the new dataframe
df_new
```

	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	Wind
0	Albury	13.400000	22.900000	0.6	5.389395	7.632205	W	44.000000	W	WNW	20.000000	
1	Albury	7.400000	25.100000	0.0	5.389395	7.632205	WNW	44.000000	NNW	WSW	4.000000	
2	Albury	12.900000	25.700000	0.0	5.389395	7.632205	WSW	46.000000	W	WSW	19.000000	
3	Albury	9.200000	28.000000	0.0	5.389395	7.632205	NE	24.000000	SE	E	11.000000	
4	Albury	17.500000	32.300000	1.0	5.389395	7.632205	W	41.000000	ENE	NW	7.000000	
5	Albury	14.600000	29.700000	0.2	5.389395	7.632205	WNW	56.000000	W	W	19.000000	
6	Albury	14.300000	25.000000	0.0	5.389395	7.632205	W	50.000000	SW	W	20.000000	
7	Albury	7.700000	26.700000	0.0	5.389395	7.632205	W	35.000000	SSE	W	6.000000	
8	Albury	9.700000	31.900000	0.0	5.389395	7.632205	NNW	80.000000	SE	NW	7.000000	
9	Albury	13.100000	30.100000	1.4	5.389395	7.632205	W	28.000000	S	SSE	15.000000	

This is the new dataframe after removing the outliers using ZScore

- Checked the skewness using skew() option.

Checking skewness

```
df_new.skew().sort_values()
```

```
Sunshine        -0.723970
Cloud9am        -0.356892
Cloud3pm        -0.266829
Humidity9am     -0.231656
MinTemp         -0.084549
Temp9am         -0.038035
Day              0.002731
Pressure9am      0.020735
Pressure3pm      0.044876
Month            0.054460
Humidity3pm      0.125150
WindSpeed3pm     0.300109
MaxTemp          0.314510
Temp3pm          0.334170
Year             0.430100
WindGustSpeed    0.506897
WindSpeed9am     0.711395
Evaporation      0.846181
Rainfall         3.519090
dtype: float64
```

Observations

We can see that the following features have skewness

- Sunshine
- Rainfall
- Evaporation
- Windspeed9am
- Windgustspeed .

- I have removed skewness using log transformation method as follows and Checked for the skewness again

Removing skewness using log transformation

```
#Applying Log method on all the columns with skewness
df_new["Sunshine"] = np.log1p(df_new["Sunshine"])
df_new["Rainfall"] = np.log1p(df_new["Rainfall"])
df_new["Evaporation"] = np.log1p(df_new["Evaporation"])
df_new["WindSpeed9am"] = np.log1p(df_new["WindSpeed9am"])
df_new["WindGustSpeed"] = np.log1p(df_new["WindGustSpeed"])
```

```
#checking skewness again
df_new.skew().sort_values()
```

```
Sunshine        -2.460672
WindSpeed9am    -1.186810
Evaporation     -0.993594
WindGustSpeed   -0.526267
Cloud9am        -0.356892
Cloud3pm        -0.266829
Humidity9am     -0.231656
MinTemp         -0.084549
Temp9am         -0.038035
Day              0.002731
Pressure9am      0.020735
Pressure3pm      0.044876
Month            0.054460
Humidity3pm      0.125150
WindSpeed3pm     0.300109
MaxTemp          0.314510
Temp3pm          0.334170
Year             0.430100
Rainfall         1.811941
dtype: float64
```

We can see that we have successfully removed some of the skewness but some of the columns still have skewness. This may be due to the filling of mean values in all the cells with NaN values. These columns being important in the model training we cannot drop them. Lets now continue with encoding the categorical columns into numerical data

- Some of the columns having high skewness could not be dropped due its importance in the dataset. This skewness may also be a result of a major chunk of the values in the columns being Null values and filling all of them using the same mean median values.

Label Encoding

- Performed label encoding to convert the categorical data into the numerical form for the better convenience in model training.

```
#Importing necessary library
from sklearn.preprocessing import LabelEncoder

# Encoding the categorical columns
le=LabelEncoder()
df_new[cat_col]= df_new[cat_col].apply(le.fit_transform)
df_new
```

	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	WindSp
0	1	13.400000	22.900000	0.470004	1.854640	2.155500	13	3.806662	13	14	3.044522	24
1	1	7.400000	25.100000	0.000000	1.854640	2.155500	14	3.806662	6	15	1.609438	22
2	1	12.900000	25.700000	0.000000	1.854640	2.155500	15	3.850148	13	15	2.995732	26
3	1	9.200000	28.000000	0.000000	1.854640	2.155500	4	3.218876	9	0	2.484907	9
4	1	17.500000	32.300000	0.693147	1.854640	2.155500	13	3.737670	1	7	2.079442	20
5	1	14.600000	29.700000	0.182322	1.854640	2.155500	14	4.043051	13	13	2.995732	24
6	1	14.300000	25.000000	0.000000	1.854640	2.155500	13	3.931826	12	13	3.044522	24
7	1	7.700000	26.700000	0.000000	1.854640	2.155500	13	3.583519	10	13	1.945910	17
8	1	9.700000	31.900000	0.000000	1.854640	2.155500	6	4.394449	9	7	2.079442	26
9	1	13.100000	30.100000	0.875469	1.854640	2.155500	13	3.367296	8	10	2.772589	11

Correlation

- Checked the correlation between the features and visualized it using heat map.

```
# Checking the corelation between feature and label
cor =df_new.corr()
cor
```

	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	W
Location	1.000000	0.116694	0.063798	0.013354	0.123801	0.068152	-0.037707	0.263202	-0.041030	-0.031195	0.219486	
MinTemp	0.116694	1.000000	0.718598	0.080984	0.355551	0.008673	-0.154935	0.262663	-0.045231	-0.154184	0.177233	
MaxTemp	0.063798	0.718598	1.000000	-0.206494	0.458312	0.316812	-0.237955	0.163935	-0.217160	-0.190417	0.039052	
Rainfall	0.013354	0.080984	-0.206494	1.000000	-0.151081	-0.189936	0.116389	0.086445	0.148138	0.103933	0.032173	
Evaporation	0.123801	0.355551	0.458312	-0.151081	1.000000	0.349285	-0.116103	0.154434	-0.096182	-0.039583	0.009505	
Sunshine	0.068152	0.008673	0.316812	-0.189936	0.349285	1.000000	-0.073069	0.002758	-0.054298	-0.049383	-0.063329	
WindGustDir	-0.037707	-0.154935	-0.237955	0.116389	-0.116103	-0.073069	1.000000	-0.002519	0.405681	0.507292	0.110829	
WindGustSpeed	0.263202	0.262663	0.163935	0.086445	0.154434	0.002758	-0.002519	1.000000	-0.059409	0.086693	0.453087	
WindDir9am	-0.041030	-0.045231	-0.217160	0.148138	-0.096182	-0.054298	0.405681	-0.059409	1.000000	0.243622	0.210760	
WindDir3pm	-0.031195	-0.154184	-0.190417	0.103933	-0.039583	-0.049383	0.507292	0.086693	0.243622	1.000000	0.032890	
WindSpeed9am	0.219486	0.177233	0.039052	0.032173	0.009505	-0.063329	0.110829	0.453087	0.210760	0.032890	1.000000	
WindSpeed3pm	0.231939	0.185976	0.093241	0.026797	0.068306	0.014423	0.108357	0.627023	0.058887	0.079716	0.513002	
Humidity9am	0.003725	-0.126383	-0.362687	0.374141	-0.291772	-0.276397	0.031081	-0.282759	0.009571	-0.007820	-0.353248	
Humidity3pm	0.059978	0.090206	-0.409560	0.357797	-0.214364	-0.397561	0.051061	-0.067943	0.117910	-0.009385	-0.062625	
Pressure9am	-0.024265	-0.433113	-0.333569	-0.108548	-0.253139	0.011947	-0.095047	-0.364616	-0.003943	-0.129842	-0.137451	
Pressure3pm	-0.012288	-0.427936	-0.414358	-0.034828	-0.269095	-0.035891	-0.008263	-0.322502	0.081400	-0.037280	-0.102700	
Cloud9am	0.041859	0.089354	-0.252579	0.278194	-0.101440	-0.438407	0.123415	0.001923	0.092396	0.073742	-0.009027	
Cloud3pm	0.027050	0.036649	-0.248268	0.240747	-0.137281	-0.460418	0.093527	0.053528	0.062637	0.054321	0.027404	
Temp9am	0.118100	0.888690	0.864864	-0.076736	0.424506	0.182254	-0.185080	0.237165	-0.101576	-0.172000	0.156994	
Temp3pm	0.060707	0.687570	0.974957	-0.215864	0.446336	0.332749	-0.249791	0.130016	-0.224390	-0.202000	0.021519	
RainToday	0.002934	0.081728	-0.219747	0.902769	-0.166170	-0.187463	0.121709	0.075087	0.149239	0.100983	0.035486	
RainTomorrow	0.009222	0.091021	-0.149708	0.317142	-0.087821	-0.293951	0.044378	0.161878	0.027091	0.003674	0.048750	
Day	-0.004978	0.010924	0.017068	-0.016476	0.009767	-0.000143	0.017003	-0.005867	-0.010014	-0.000714	-0.014493	
Month	-0.086011	-0.247066	-0.164351	-0.005666	-0.047647	-0.001379	0.038044	0.048639	0.018344	0.028629	0.033834	
Year	0.481143	0.040001	0.120204	0.002860	0.148225	0.115062	-0.096042	-0.014468	-0.034468	-0.010781	-0.110893	

```
#Plotting heatmap to check the correlation
plt.figure(figsize=(20,10))
sns.heatmap(df_new.corr(),linewidths=.1,annot=True)
plt.yticks(rotation=0,fontsize=12);
plt.xticks(rotation=45,fontsize=12);
```

Observations

- MaxTemp and MinTemp are higly correlated with Temp9am and Temp3pm

- MaxTemp and MinTemp are higly correlated with each other.

- MaxTemp and MinTemp are higly negatively correlated with Pressure9am, Pressure3pm

- MaxTemp is also higly negatively correlated with Humidity9am, Humidity3pm

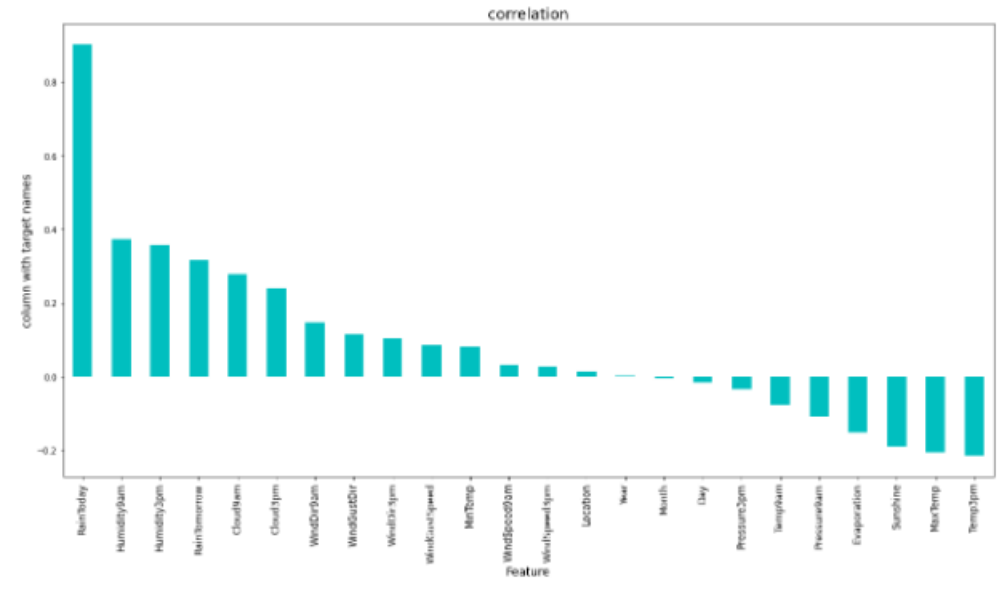
- WindDir3pm is highly negatively correlated with the Humidity9am

- WindSpeed3pm is highly negatively correlated with Temp9am and Temp3pm

This may lead to multicolinearity issue which needs to be addressed in the later part

- Plotted bar plot to get better insights on correlation of both target variables with other columns individually.

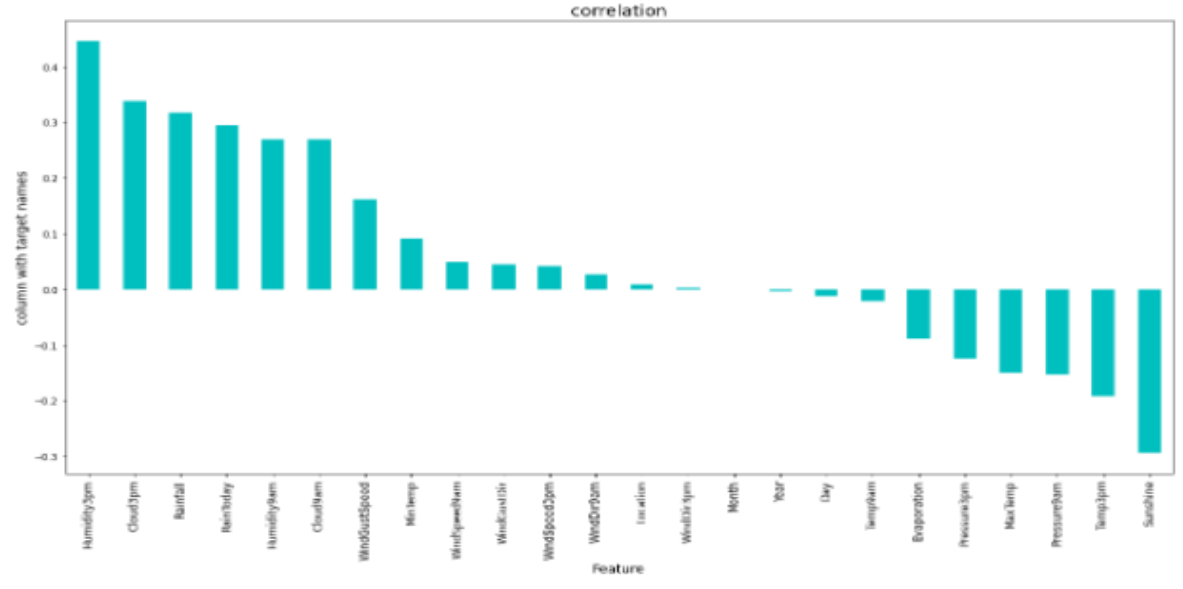
```
#Plotting bar plot to get better insights on target variable correlation with other columns .
plt.figure(figsize=(20,10))
df_new.corr()['Rainfall'].sort_values(ascending=False).drop(['Rainfall']).plot(kind='bar',color='c')
plt.xlabel('Feature',fontsize=14)
plt.ylabel('column with target names',fontsize=14)
plt.xticks(rotation=90,fontsize=12);
plt.title('correlation',fontsize=18)
plt.show()
```



Observations

- The target 'Rainfall' is highly positively correlated with the feature 'RainToday'
- The target 'Rainfall' is positively well correlated with the following features
  - Humidity9am
  - Humidity3pm
  - RainTomorrow
  - Cloud9am
  - Cloud3pm
- The target 'Rainfall' is least positively correlated with the following features
  - WindDir9am
  - WindGustDir
  - WindDir3pm
  - WindGustspeed
  - MinTemp
  - WindSpeed9am
  - WindSpeed3pm
  - Location
- The features 'Year','Month' has least correlation with the target 'Rainfall', which says 'Year' and 'Month' has least impact on 'Rainfall'. Hence they can be dropped
- The target 'Rainfall' is most negatively correlated with the feature 'Temp3pm'
  - The target 'Rainfall' is negatively correlated with the remaining features

```
#Plotting bar plot to get better insights on target variable correlation with other columns .
plt.figure(figsize=(20,10))
df_new.corr()['RainTomorrow'].sort_values(ascending=False).drop(['RainTomorrow']).plot(kind='bar',color='c')
plt.xlabel('Feature',fontsize=14)
plt.ylabel('column with target names',fontsize=14)
plt.xticks(rotation=90,fontsize=12);
plt.title('correlation',fontsize=18)
plt.show()
```



Observations

- The target 'RainTomorrow' is positively well correlated with the following features
    - Humidity3pm
    - Cloud3pm
    - Rainfall
    - RainToday
    - Humidity9am
    - Cloud9am
- The target 'RainTomorrow' is least positively correlated with the following features
    - WindGustspeed
    - MinTemp
    - WindSpeed9am
    - WindGustDir
    - WindSpeed3pm
    - WindDir9am
    - Location
    - WindDir3pm
- The feature 'Month' has least correlation with the target 'RainTomorrow', which says 'Month' has least impact on 'RainTomorrow'. Hence it can be dropped.
- The target 'RainTomorrow' is highly negatively correlated with the feature 'Sunshine
- The target 'RainTomorrow' is negatively correlated with the remaining features

Model/s Development and Evaluation

Building Machine Learning Models

1. Model training for target 'RainTomorrow'

- I have dropped the column ‘Month’ from the dataframe as it is least correlated with the target ‘RainTomorrow’.

```
#Dropping the least correlated column
df_new_rt = df_new.drop(["Month"],axis=1)
```

- Then I split the features and label into X and Y

Splitting feature and label into x and y

```
#Splitting the data for model training and Assigning variables for the further procedure
x = df_new_rt.drop("RainTomorrow", axis=1)
y = df_new_rt["RainTomorrow"]
```

- **SCALING -** I have scaled the data using standard scaler

```
#importing necessary libraries
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
```

```
sc=StandardScaler()
X = pd.DataFrame(sc.fit_transform(x), columns=x.columns)
```

- The data has been scaled using Standard Scaler

- **Checking VIF (Variance Inflation Factor)**

I have checked the VIF for all the features to know about the multicollinearity.

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif=pd.DataFrame()
vif["vif_Features"]=[variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
vif["Features"]=X.columns
vif
```

	vif_Features	Features
0	1.587438	Location
1	8.438035	MinTemp
2	26.290205	MaxTemp
3	5.618620	Rainfall
4	1.503351	Evaporation
5	1.633337	Sunshine
6	1.621182	WindGustDir
7	2.151515	WindGustSpeed
8	1.397232	WindDir9am
9	1.492213	WindDir3pm
10	1.813414	WindSpeed9am
11	1.961801	WindSpeed3pm
12	3.964858	Humidity9am
13	5.370478	Humidity3pm
14	19.946358	Pressure9am
15	19.043803	Pressure3pm
16	1.864838	Cloud9am
17	1.744280	Cloud3pm
18	17.077838	Temp9am
19	32.873458	Temp3pm
20	5.484334	RainToday
21	1.004164	Day
22	1.522891	Year

Since there can be seen high VIF for Total volume, the column shall be dropped and check for VIF again.

- I have dropped the ‘Temp3pm’ feature and checked the VIF again

```
#Droping Temp3pm column
X = X.drop(["Temp3pm"],axis=1)
```

```
# Checking VIF again
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif=pd.DataFrame()
vif["vif_Features"]=[variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
vif["Features"]=X.columns
vif
```

	vif_Features	Features
0	1.585832	Location
1	8.369011	MinTemp
2	9.570455	MaxTemp
3	5.615950	Rainfall
4	1.501891	Evaporation
5	1.632425	Sunshine
6	1.618326	WindGustDir
7	2.142374	WindGustSpeed
8	1.396564	WindDir9am
9	1.491398	WindDir3pm
10	1.813092	WindSpeed9am
11	1.959078	WindSpeed3pm
12	3.517394	Humidity9am
13	3.660464	Humidity3pm
14	19.375985	Pressure9am
15	18.579014	Pressure3pm
16	1.864553	Cloud9am
17	1.732830	Cloud3pm
18	15.845085	Temp9am
19	5.479753	RainToday
20	1.003742	Day
21	1.522642	Year

- After dropping ‘Temp3pm’ ‘Pressure9am’ can be seen having high VIF, hence dropping that

column and checked the VIF again.

```
#Droping Pressure9am column
X = X.drop(["Pressure9am"],axis=1)
```

```
# Checking VIF again
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif=pd.DataFrame()
vif["vif_Features"]=[variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
vif["Features"]=X.columns
vif
```

	vif_Features	Features
0	1.585896	Location
1	8.208098	MinTemp
2	9.164874	MaxTemp
3	5.604371	Rainfall
4	1.490243	Evaporation
5	1.632071	Sunshine
6	1.603166	WindGustDir
7	2.123890	WindGustSpeed
8	1.372974	WindDir9am
9	1.424448	WindDir3pm
10	1.803738	WindSpeed9am
11	1.942519	WindSpeed3pm
12	3.517391	Humidity9am
13	3.658497	Humidity3pm
14	1.421729	Pressure3pm
15	1.864509	Cloud9am
16	1.729548	Cloud3pm
17	15.809051	Temp9am
18	5.479406	RainToday
19	1.003089	Day
20	1.520595	Year

After dropping Pressure9am, Temp9am can be seen having high VIF,hence dropping that column.

- Dropped the column ‘Temp9am’ and check the VIF again

```
#Droping Temp9am column
X = X.drop(["Temp9am"],axis=1)
```

```
# Checking VIF again
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif=pd.DataFrame()
vif["vif_Features"]=[variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
vif["Features"]=X.columns
vif
```

	vif_Features	Features
0	1.584545	Location
1	4.383821	MinTemp
2	5.347124	MaxTemp
3	5.594375	Rainfall
4	1.485171	Evaporation
5	1.627444	Sunshine
6	1.601383	WindGustDir
7	2.086115	WindGustSpeed
8	1.372677	WindDir9am
9	1.423918	WindDir3pm
10	1.793541	WindSpeed9am
11	1.904739	WindSpeed3pm
12	2.465116	Humidity9am
13	2.886190	Humidity3pm
14	1.418191	Pressure3pm
15	1.843491	Cloud9am
16	1.724109	Cloud3pm
17	5.478711	RainToday
18	1.002888	Day
19	1.497006	Year

After dropping Temp9am, Rainfall can be seen having high VIF. But the Rainfall is well positively correlated with target 'RainTomorrow'. So we shall not drop this column

- The multicolinearity has been treated to some extent, But some features still have VIF values above 4. But dropping too many columns have a negative impact on the model. So letting them be for now and moving forward with next step.

Oversampling

- I have checked for the value counts in the target 'y' to fix the imbalance in the data using SMOTE Oversampling technique and checked for the value counts again.

```
#Checking value counts in target 'y'
y.value_counts()
```

```
0    6165
1    1822
Name: RainTomorrow, dtype: int64
```

We can see that there is a significant difference in the classes hence lets fix this using SMOTE

```
#Importing necessary library to treat imbalance
from imblearn.over_sampling import SMOTE

# Treating imbalance
SM = SMOTE()
X, y = SM.fit_resample(X,y)
```

```
#Checking value counts in target 'y'
y.value_counts()
```

```
0    6165
1    6165
Name: RainTomorrow, dtype: int64
```

- Now we can see that the imbalance issue has been fixed. Proceeding with model development
- Here the target has classes as output data. Hence this is a classification problem. We have to use the classification algorithms in machine learning for training and testing.
- Importing the necessary libraries and machine learning algorithms for model training and testing

```
#importing necessary libraries
import warnings
warnings.simplefilter("ignore")
warnings.filterwarnings("ignore")
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier as KNN
from sklearn.ensemble import GradientBoostingClassifier, AdaBoostClassifier, BaggingClassifier
from xgboost import XGBClassifier as xgb
from sklearn.metrics import classification_report, confusion_matrix, roc_curve, accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import cross_val_score
from sklearn import metrics
```

- I have used necessary steps for finding the random state and Accuracy as follows

```
maxAccu=0
maxRS=0
for i in range(1,200):
    X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=.30, random_state =i)
    rf = RandomForestClassifier()
    rf.fit(X_train, y_train)
    pred = rf.predict(X_test)
    acc=accuracy_score(y_test, pred)
    if acc>maxAccu:
        maxAccu=acc
        maxRS=i
print("Best accuracy is ",maxAccu," on Random_state ",maxRS)
```

```
Best accuracy is  0.9413354960800216  on Random_state  10
```

- I have checked for the feature importance using random forest classifier as follows

Important Features

```
# Lets check the feature importance using Random Forest Classifier

rf = RandomForestClassifier()
rf.fit(X_train, y_train)
importances = pd.DataFrame({'Features':X.columns, 'Importance':np.round(rf.feature_importances_,3)})
importances = importances.sort_values('Importance', ascending=False).set_index('Features')
importances
```

Features	Importance
Humidity3pm	0.159
Cloud3pm	0.097
Humidity9am	0.062
Rainfall	0.058
WindGustSpeed	0.058
Sunshine	0.057
Cloud9am	0.056
Pressure3pm	0.048
MinTemp	0.043
MaxTemp	0.041
WindSpeed3pm	0.041
Day	0.039
WindDir9am	0.037
WindSpeed9am	0.036
WindDir3pm	0.035
Year	0.032
WindGustDir	0.031
Evaporation	0.029
Location	0.023
RainToday	0.016

- Then I have created the train-test split as follows taking test size as 30% , which means 70% of the data is taken as the train data . The random state used here is the random state we have got the best accuracy which we previously calculated

Creating train_test split

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=.30,random_state=maxRS)
```

- Then I have checked the accuracy of the model using Decision Tree classifier

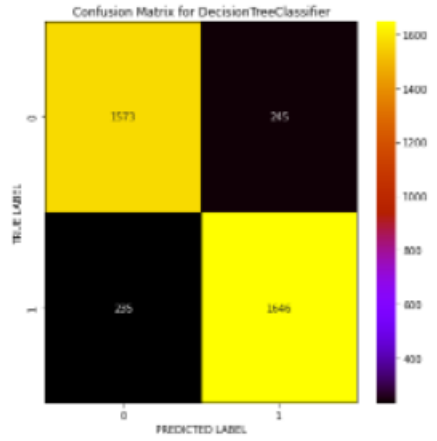
Decision Tree Classifier

```
DTC = DecisionTreeClassifier()
DTC.fit(X_train,y_train)
predDTC = DTC.predict(X_test)

print(accuracy_score(y_test, predDTC))
print(confusion_matrix(y_test, predDTC))
print(classification_report(y_test,predDTC))
```

```
0.870235198702352
[[1573  245]
 [ 235 1646]]
              precision    recall  f1-score   support

           0       0.87      0.87      0.87      1818
           1       0.87      0.88      0.87      1881

    accuracy                           0.87      3699
   macro avg       0.87      0.87      0.87      3699
weighted avg       0.87      0.87      0.87      3699
```

- We can observe that DTC has an accuracy of 87.02%
- Then plotted the confusion matrix for the same as follows

```
# Lets plot confusion matrix for DecisionTreeClassifier
cm = confusion_matrix(y_test,predDTC)

x_axis_labels = ["0","1"]
y_axis_labels = ["0","1"]

f , ax = plt.subplots(figsize=(7,7))
sns.heatmap(cm, annot = True,linewidths=.2, linecolor="black", fmt = ".0f", ax=ax, cmap="gnuplot",xticklabels=x_axis_labels,ytic

plt.xlabel("PREDICTED LABEL")
plt.ylabel("TRUE LABEL")
plt.title('Confusion Matrix for DecisionTreeClassifier')
plt.show()
```



- Then I have checked the accuracy of the model using Random forest classifier

Random Forest Classifier

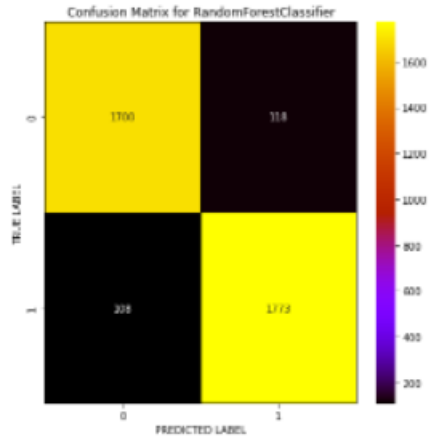
```
rf = RandomForestClassifier()
rf.fit(X_train,y_train)

# Prediction
predrf = rf.predict(X_test)

print(accuracy_score(y_test, predrf))
print(confusion_matrix(y_test, predrf))
print(classification_report(y_test,predrf))
```

```
0.9389024060556908
[[1700  118]
 [ 108 1773]]
              precision    recall  f1-score   support

           0       0.94      0.94      0.94      1818
           1       0.94      0.94      0.94      1881

    accuracy                           0.94      3699
   macro avg       0.94      0.94      0.94      3699
weighted avg       0.94      0.94      0.94      3699
```

- We can observe that RFC has an accuracy of 93.89%

- Then plotted the confusion matrix for the same as follows

```
# Lets plot confusion matrix for RandomForestClassifier
cm = confusion_matrix(y_test,predrf)

x_axis_labels = ["0","1"]
y_axis_labels = ["0","1"]

f , ax = plt.subplots(figsize=(7,7))
sns.heatmap(cm, annot = True,linewidths=.2, linecolor="black", fmt = ".0f", ax=ax, cmap="gnuplot",xticklabels=x_axis_labels,ytic

plt.xlabel("PREDICTED LABEL")
plt.ylabel("TRUE LABEL")
plt.title('Confusion Matrix for RandomForestClassifier')
plt.show()
```



- Then I have checked the accuracy of the model using Logistic Regression

Logistic Regression

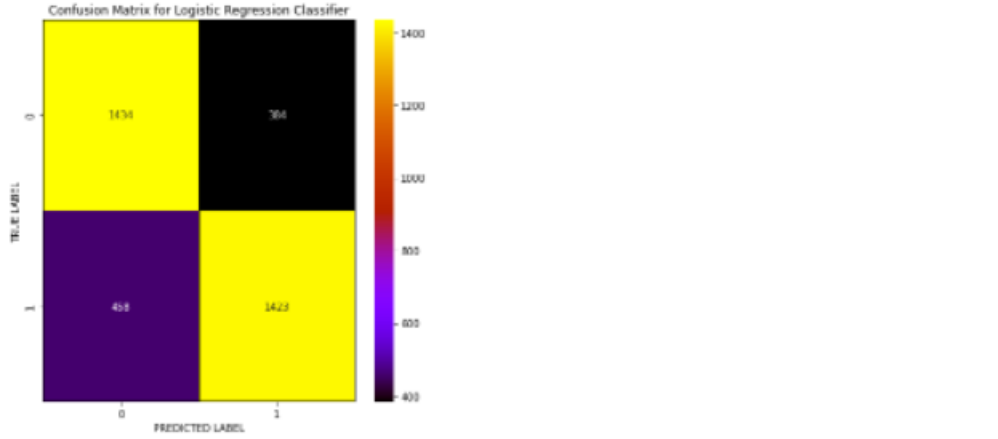
```
lr = LogisticRegression()
lr.fit(X_train,y_train)

# Prediction
predlr = lr.predict(X_test)

print(accuracy_score(y_test, predlr))
print(confusion_matrix(y_test, predlr))
print(classification_report(y_test,predlr))
```

```
0.7723709110570425
[[1434  384]
 [ 458 1423]]
              precision    recall  f1-score   support

           0       0.76      0.79      0.77      1818
           1       0.79      0.76      0.77      1881

    accuracy                           0.77      3699
   macro avg       0.77      0.77      0.77      3699
weighted avg       0.77      0.77      0.77      3699
```

- We can observe that the above model has an accuracy of 77%

- Then plotted the confusion matrix for the same as follows

```
# Lets plot confusion matrix for Logistic Regression
cm = confusion_matrix(y_test,predlr)

x_axis_labels = ["0","1"]
y_axis_labels = ["0","1"]

f , ax = plt.subplots(figsize=(7,7))
sns.heatmap(cm, annot = True,linewidths=.2, linecolor="black", fmt = ".0f", ax=ax, cmap="gnuplot",xticklabels=x_axis_labels,ytic

plt.xlabel("PREDICTED LABEL")
plt.ylabel("TRUE LABEL")
plt.title('Confusion Matrix for Logistic Regression Classifier')
plt.show()
```



- Then I have checked the accuracy of the model using Ada boost classifier

AdaBoost Classifier

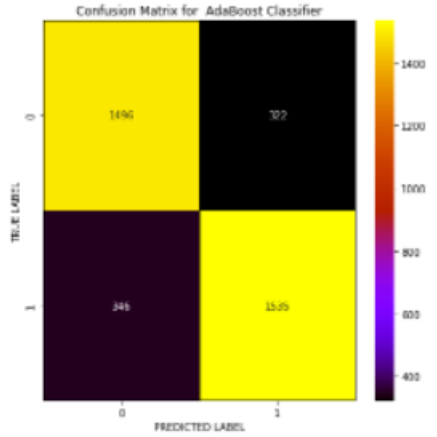
```
# Checking accuracy for AdaBoost Classifier
abc = AdaBoostClassifier()
abc.fit(X_train,y_train)

# Prediction
predabc = abc.predict(X_test)

print(accuracy_score(y_test, predabc))
print(confusion_matrix(y_test, predabc))
print(classification_report(y_test,predabc))
```

```
0.8194106515274399
[[1496  322]
 [ 346 1535]]
              precision    recall  f1-score   support

           0       0.81      0.82      0.82      1818
           1       0.83      0.82      0.82      1881

    accuracy                           0.82      3699
   macro avg       0.82      0.82      0.82      3699
weighted avg       0.82      0.82      0.82      3699
```

- We can observe that ABC has an accuracy of 81.94%

- Then plotted the confusion matrix for the same as follows

```
# Lets plot confusion matrix for  AdaBoost Classifier
cm = confusion_matrix(y_test,predabc)

x_axis_labels = ["0","1"]
y_axis_labels = ["0","1"]

f , ax = plt.subplots(figsize=(7,7))
sns.heatmap(cm, annot = True,linewidths=.2, linecolor="black", fmt = ".0f", ax=ax, cmap="gnuplot",xticklabels=x_axis_labels,ytic

plt.xlabel("PREDICTED LABEL")
plt.ylabel("TRUE LABEL")
plt.title('Confusion Matrix for  AdaBoost Classifier')
plt.show()
```



- Then I have checked the accuracy of the model using Gradient boosting classifier

Gradient Boosting Classifier

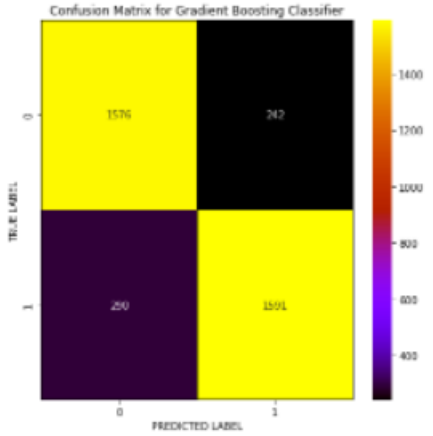
```
gb = GradientBoostingClassifier()
gb.fit(X_train,y_train)

# Prediction
predgb = gb.predict(X_test)

print(accuracy_score(y_test, predgb))
print(confusion_matrix(y_test, predgb))
print(classification_report(y_test,predgb))
```

```
0.8561773452284401
[[1576  242]
 [ 290 1591]]
              precision    recall  f1-score   support

           0       0.84      0.87      0.86      1818
           1       0.87      0.85      0.86      1881

    accuracy                           0.86      3699
   macro avg       0.86      0.86      0.86      3699
weighted avg       0.86      0.86      0.86      3699
```

- We can observe that GBC has an accuracy of 85.6%

- Then plotted the confusion matrix for the same as follows

```
# Lets plot confusion matrix for Gradient Boosting Classifier
cm = confusion_matrix(y_test,predgb)

x_axis_labels = ["0","1"]
y_axis_labels = ["0","1"]

f , ax = plt.subplots(figsize=(7,7))
sns.heatmap(cm, annot = True,linewidths=.2, linecolor="black", fmt = ".0f", ax=ax, cmap="gnuplot",xticklabels=x_axis_labels,ytic

plt.xlabel("PREDICTED LABEL")
plt.ylabel("TRUE LABEL")
plt.title('Confusion Matrix for Gradient Boosting Classifier')
plt.show()
```



- Then I have checked the accuracy of the model using XGradient boosting classifier

XGB Classifier

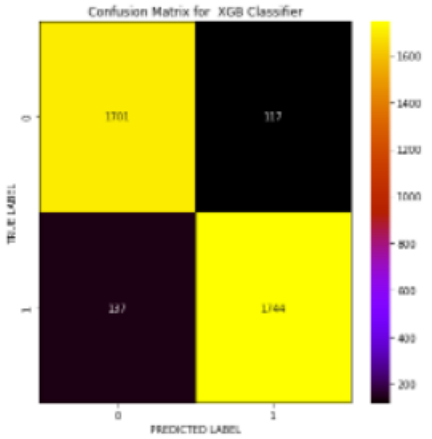
```
# Checking accuracy for XGBClassifier
xgb = xgb(verbosity=0)
xgb.fit(X_train,y_train)

# Prediction
predxgb = xgb.predict(X_test)

print(accuracy_score(y_test, predxgb))
print(confusion_matrix(y_test, predxgb))
print(classification_report(y_test,predxgb))
```

```
0.9313327926466612
[[1701  117]
 [ 137 1744]]
              precision    recall  f1-score   support

           0       0.93      0.94      0.93      1818
           1       0.94      0.93      0.93      1881

    accuracy                           0.93      3699
   macro avg       0.93      0.93      0.93      3699
weighted avg       0.93      0.93      0.93      3699
```

- We can observe that XGB has an accuracy of 93.1%

- Then plotted the confusion matrix for the same as follows

```
# Lets plot confusion matrix for  XGBClassifier
cm = confusion_matrix(y_test,predxgb)

x_axis_labels = ["0","1"]
y_axis_labels = ["0","1"]

f , ax = plt.subplots(figsize=(7,7))
sns.heatmap(cm, annot = True,linewidths=.2, linecolor="black", fmt = ".0f", ax=ax, cmap="gnuplot",xticklabels=x_axis_labels,ytic

plt.xlabel("PREDICTED LABEL")
plt.ylabel("TRUE LABEL")
plt.title('Confusion Matrix for  XGB Classifier')
plt.show()
```



- Then I have checked the accuracy of the model using Extra trees classifier

Extra Trees Classifier

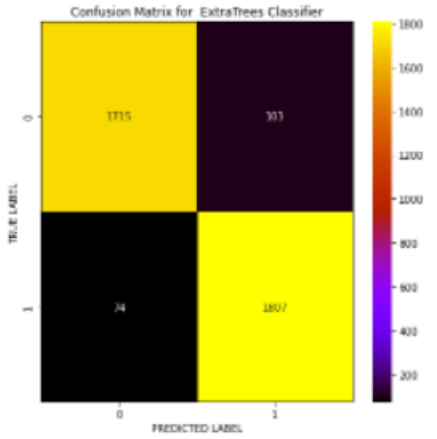
```
from sklearn.ensemble import ExtraTreesClassifier
# Checking accuracy for ExtraTreesClassifier
xtc = ExtraTreesClassifier()
xtc.fit(X_train,y_train)

# Prediction
predxtc = xtc.predict(X_test)

print(accuracy_score(y_test, predxtc))
print(confusion_matrix(y_test, predxtc))
print(classification_report(y_test,predxtc))
```

```
0.9521492295214923
[[1715  103]
 [  74 1807]]
              precision    recall  f1-score   support

           0       0.96      0.94      0.95      1818
           1       0.95      0.96      0.95      1881

    accuracy                           0.95      3699
   macro avg       0.95      0.95      0.95      3699
weighted avg       0.95      0.95      0.95      3699
```

- We can observe that ETC has an accuracy of 95.21%

- Then plotted the confusion matrix for the same as follows

```
# Lets plot confusion matrix for  ExtraTreesClassifier
cm = confusion_matrix(y_test,predxtc)

x_axis_labels = ["0","1"]
y_axis_labels = ["0","1"]

f , ax = plt.subplots(figsize=(7,7))
sns.heatmap(cm, annot = True,linewidths=.2, linecolor="black", fmt = ".0f", ax=ax, cmap="gnuplot",xticklabels=x_axis_labels,ytic

plt.xlabel("PREDICTED LABEL")
plt.ylabel("TRUE LABEL")
plt.title('Confusion Matrix for  ExtraTrees Classifier')
plt.show()
```



- Then I have checked the accuracy of the model using K Nearest Neighbors classifier

KNeighborsClassifier

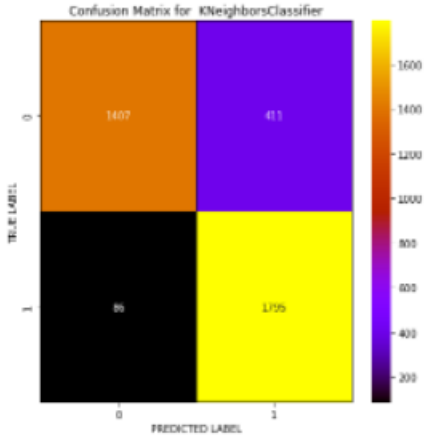
```
# Checking accuracy for KNeighborsClassifier
knn = KNN()
knn.fit(X_train,y_train)

# Prediction
predknn = knn.predict(X_test)

print(accuracy_score(y_test, predknn))
print(confusion_matrix(y_test, predknn))
print(classification_report(y_test,predknn))
```

```
0.8656393619897269
[[1407  411]
 [  86 1795]]
              precision    recall  f1-score   support

           0       0.94      0.77      0.85      1818
           1       0.81      0.95      0.88      1881

    accuracy                           0.87      3699
   macro avg       0.88      0.86      0.86      3699
weighted avg       0.88      0.87      0.86      3699
```

- We can observe that KNN has an accuracy of 86.56%

- Then plotted the confusion matrix for the same as follows

```
# Lets plot confusion matrix for  KNeighborsClassifier
cm = confusion_matrix(y_test,predknn)

x_axis_labels = ["0","1"]
y_axis_labels = ["0","1"]

f , ax = plt.subplots(figsize=(7,7))
sns.heatmap(cm, annot = True,linewidths=.2, linecolor="black", fmt = ".0f", ax=ax, cmap="gnuplot",xticklabels=x_axis_labels,ytic

plt.xlabel("PREDICTED LABEL")
plt.ylabel("TRUE LABEL")
plt.title('Confusion Matrix for  KNeighborsClassifier')
plt.show()
```



- Then I have checked the accuracy of the model using Support Vector Machine classifier

Support Vector Machine Classifier

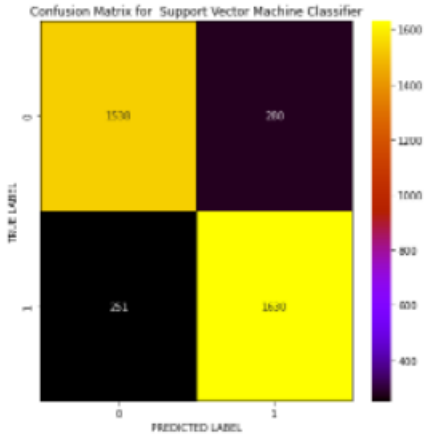
```
# Checking accuracy for Support Vector Machine Classifier
svc = SVC()
svc.fit(X_train,y_train)

#Prediction
predsvc = svc.predict(X_test)

print(accuracy_score(y_test, predsvc))
print(confusion_matrix(y_test, predsvc))
print(classification_report(y_test,predsvc))
```

```
0.8564476885644768
[[1538  280]
 [ 251 1630]]
              precision    recall  f1-score   support

           0       0.86      0.85      0.85      1818
           1       0.85      0.87      0.86      1881

    accuracy                           0.86      3699
   macro avg       0.86      0.86      0.86      3699
weighted avg       0.86      0.86      0.86      3699
```

- We can observe that SVC has an accuracy of 85.64%

- Then plotted the confusion matrix for the same as follows

```
# Lets plot confusion matrix for  Support Vector Machine Classifier
cm = confusion_matrix(y_test,predsvc)

x_axis_labels = ["0","1"]
y_axis_labels = ["0","1"]

f , ax = plt.subplots(figsize=(7,7))
sns.heatmap(cm, annot = True,linewidths=.2, linecolor="black", fmt = ".0f", ax=ax, cmap="gnuplot",xticklabels=x_axis_labels,ytic

plt.xlabel("PREDICTED LABEL")
plt.ylabel("TRUE LABEL")
plt.title('Confusion Matrix for  Support Vector Machine Classifier')
plt.show()
```



- Then I have checked for the Cross validation scores for all the above models as follows

Checking the Cross Validation Score

```
# cv score for Logistic Regression
print('Logistic Regression:',cross_val_score(lr,X,y,cv=5).mean())

# cv score for Decision Tree classifier
print('Decision Tree classifier:',cross_val_score(DTC,X,y,cv=5).mean())

# cv score for Random Forest Classifier
print('Random Forest Classifier:',cross_val_score(rf,X,y,cv=5).mean())

# cv score for AdaBoosting Classifier
print('AdaBoosting Classifier:',cross_val_score(abc,X,y,cv=5).mean())

# cv score for Gradient Boosting Classifier
print('Gradient Boosting Classifier:',cross_val_score(gb,X,y,cv=5).mean())

# cv score for XGB Classifier
print('XGB Classifier:',cross_val_score(xgb,X,y,cv=5).mean())

# cv score for Extra Trees Classifier
print('Extra Trees Classifier:',cross_val_score(xtc,X,y,cv=5).mean())

# cv score for KNeighborsClassifier
print('KNeighborsClassifier:',cross_val_score(knn,X,y,cv=5).mean())

# cv score for Support Vector Machine Classifier
print('Support Vector Machine Classifier:',cross_val_score(svc,X,y,cv=5).mean())
```

```
Logistic Regression: 0.7118410381184104
Decision Tree classifier: 0.7409570154095702
Random Forest Classifier: 0.7924574209245743
AdaBoosting Classifier: 0.6691808596918085
Gradient Boosting Classifier: 0.6611516626115166
XGB Classifier: 0.6987834549878345
Extra Trees Classifier: 0.8037307380373073
KNeighborsClassifier: 0.7334144363341444
Support Vector Machine Classifier: 0.7362530413625304
```

- From the above we can observe the difference between the accuracy score and cross validation score is least in SVC and Gradient Boosting classifier but Extra trees classifier has higher accuracy than other classifiers. Hence we can predict that ETC is the best model.

- Then I have tuned the selected best model using Hyper parameter tuning for better performance as follows

Hyper parameter tuning

```
from sklearn.model_selection import GridSearchCV
```

```
# ExtraTrees Classifier
parameters = {'criterion' : ['gini','entropy'],
              'random_state' : [10, 50, 1000],
              'max_depth' : [0, 10, 20],
              'n_jobs' : [-2, -1, 1],
              'n_estimators' : [50,100, 200, 300]}
```

```
GCV=GridSearchCV(ExtraTreesClassifier(),parameters,cv=5)
```

- I have tuned the extra trees classifier and fit the current model for better performance as follows

```
GCV.fit(X_train,y_train)
```

```
GridSearchCV(cv=5, estimator=ExtraTreesClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [0, 10, 20],
                         'n_estimators': [50, 100, 200, 300],
                         'n_jobs': [-2, -1, 1],
                         'random_state': [10, 50, 1000]})
```

```
GCV.best_params_
```

```
{'criterion': 'gini',
 'max_depth': 20,
 'n_estimators': 200,
 'n_jobs': -2,
 'random_state': 10}
```

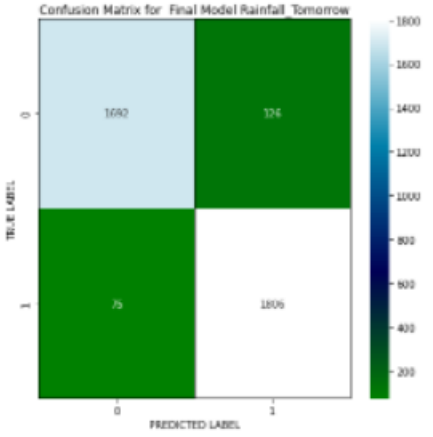
- This has resulted in the improvement of accuracy as 94.56%, which is an expected level of accuracy

```
Rainfall_Tomorrow=ExtraTreesClassifier(criterion='gini', max_depth=20, n_estimators=200, n_jobs=-2, random_state=10)
Rainfall_Tomorrow.fit(X_train, y_train)
pred = Rainfall_Tomorrow.predict(X_test)
acc=accuracy_score(y_test,pred)
print(acc*100)
```

```
94.56698945661
```

- Then plotted the confusion matrix for the same as follows

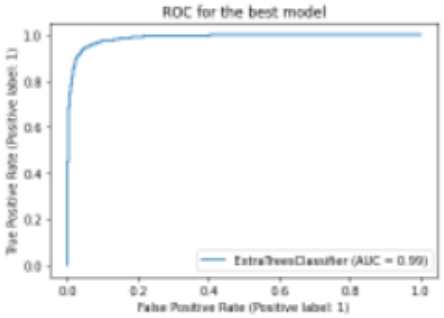
```
# Lets plot confusion matrix for  FinalModel Rainfall_Tomorrow
cm = confusion_matrix(y_test,pred)

x_axis_labels = ["0","1"]
y_axis_labels = ["0","1"]

f , ax = plt.subplots(figsize=(7,7))
sns.heatmap(cm, annot = True,linewidths=.2, linecolor="black", fmt = ".0f", ax=ax, cmap="ocean",xticklabels=x_axis_labels,ytickl

plt.xlabel("PREDICTED LABEL")
plt.ylabel("TRUE LABEL")
plt.title('Confusion Matrix for  Final Model Rainfall_Tomorrow')
plt.show()
```



- Then I have plotted the AUC ROC curve for the selected best model as follows

Plotting ROC and Compare AUC for the best model

```
from sklearn.metrics import plot_roc_curve
plot_roc_curve(Rainfall_Tomorrow, X_test, y_test)
plt.title("ROC for the best model")
plt.show()
```



- The predicted model has 99% AUC which is approximately ideal
- Then I have saved the model importing and using joblib as follows

Saving the Model

```
import joblib
joblib.dump(Rainfall_Tomorrow,"Prediction_of_Rainfall_Tomorrow.pkl")
```

```
['Prediction_of_Rainfall_Tomorrow.pkl']
```

- Predicted using the saved model as follows

Predicting from the saved model

```
# Loading the saved model
model=joblib.load("Prediction_of_Rainfall_Tomorrow.pkl")

#Prediction
prediction = model.predict(X_test)
prediction
```

```
array([1, 1, 1, ..., 0, 0, 1])
```

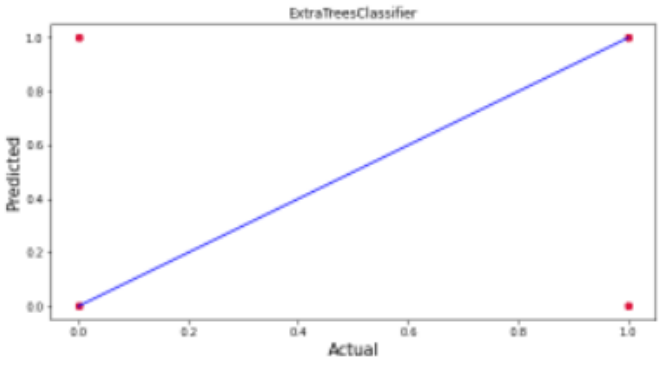
- I have created a dataframe of the actual vs predicted values for better view and comparison of outputs from the predicted model

```
pd.DataFrame([model.predict(X_test)[:],y_test[:]],index=["Predicted","Original"]).T
```

	Predicted	Original
0	1	1
1	1	1
2	1	1
3	1	1
4	0	0
5	0	0
6	1	0
7	0	1
8	1	1
9	0	0
10	0	0

- Then I have plotted Actual vs Predicted to get better insights as follows

```
# Plotting Actual vs Predicted to get better insights
plt.figure(figsize=(10,5))
plt.scatter(y_test, prediction, c='crimson')
p1 = max(max(prediction), max(y_test))
p2 = min(min(prediction), min(y_test))
plt.plot([p1, p2], [p1, p2], 'b-')
plt.xlabel('Actual', fontsize=15)
plt.ylabel('Predicted', fontsize=15)
plt.title("ExtraTreesClassifier")
plt.show()
```



- This shows that all the actual values look similar to predicted which is the expected outcome of the model training using machine learning

2. Model training for target 'Rainfall'

- I have dropped the column ‘Month’ and ‘Year’ from the dataframe as they are least correlated with the target ‘Rainfall’.

```
#Dropping the least correlated column
df_new_rf = df_new.drop(["Year","Month"],axis=1)
```

- Then I split the features and label into X and Y

Splitting feature and label into x and y

```
#Splitting the data for model training and Assigning variables for the further procedure
x = df_new_rf.drop("Rainfall", axis=1)
y = df_new_rf["Rainfall"]
```

- **SCALING -** I have scaled the data using standard scaler

Scaling the data using Standard Scaler

```
#importing necessary libraries
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split

sc=StandardScaler()
X = pd.DataFrame(sc.fit_transform(x), columns=x.columns)
```

- The data has been scaled using Standard Scaler

- **Checking VIF (Variance Inflation Factor)**

I have checked the VIF for all the features to know about the multicollinearity.

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif=pd.DataFrame()
vif["vif_Features"]=[variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
vif["Features"]=X.columns
vif
```

	vif_Features	Features
0	1.186869	Location
1	8.313227	MinTemp
2	26.275159	MaxTemp
3	1.492193	Evaporation
4	1.647178	Sunshine
5	1.611628	WindGustDir
6	2.209566	WindGustSpeed
7	1.385009	WindDir9am
8	1.485567	WindDir3pm
9	1.794045	WindSpeed9am
10	1.964071	WindSpeed3pm
11	3.830357	Humidity9am
12	5.549427	Humidity3pm
13	19.920002	Pressure9am
14	19.084502	Pressure3pm
15	1.864046	Cloud9am
16	1.755201	Cloud3pm
17	16.820691	Temp9am
18	32.852320	Temp3pm
19	1.361314	RainToday
20	1.429065	RainTomorrow
21	1.004569	Day

Since there can be seen high VIF for Total volume, the column shall be dropped and check for VIF again.

- I have dropped the ‘Temp3pm’ feature and checked the VIF again

```
#Droping Temp3pm column
X = X.drop(["Temp3pm"],axis=1)
```

```
# Checking VIF again
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif=pd.DataFrame()
vif["vif_Features"]=[variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
vif["Features"]=X.columns
vif
```

	vif_Features	Features
0	1.184574	Location
1	8.247014	MinTemp
2	9.582451	MaxTemp
3	1.490611	Evaporation
4	1.646275	Sunshine
5	1.608600	WindGustDir
6	2.200838	WindGustSpeed
7	1.384439	WindDir9am
8	1.484841	WindDir3pm
9	1.793674	WindSpeed9am
10	1.951121	WindSpeed3pm
11	3.371618	Humidity9am
12	3.833628	Humidity3pm
13	19.353071	Pressure9am
14	18.623099	Pressure3pm
15	1.863735	Cloud9am
16	1.744026	Cloud3pm
17	15.566409	Temp9am
18	1.360794	RainToday
19	1.429063	RainTomorrow
20	1.004154	Day

- After dropping ‘Temp3pm’ ‘Pressure9am’ can be seen having high VIF, hence dropping that column and checked the VIF again.

```
#Droping Pressure9am column
X = X.drop(["Pressure9am"],axis=1)
```

```
# Checking VIF again
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif=pd.DataFrame()
vif["vif_Features"]=[variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
vif["Features"]=X.columns
vif
```

	vif_Features	Features
0	1.184338	Location
1	8.080836	MinTemp
2	9.180464	MaxTemp
3	1.479472	Evaporation
4	1.645673	Sunshine
5	1.592077	WindGustDir
6	2.177684	WindGustSpeed
7	1.361817	WindDir9am
8	1.419870	WindDir3pm
9	1.785783	WindSpeed9am
10	1.934719	WindSpeed3pm
11	3.371618	Humidity9am
12	3.833430	Humidity3pm
13	1.432646	Pressure3pm
14	1.863701	Cloud9am
15	1.740308	Cloud3pm
16	15.534667	Temp9am
17	1.346883	RainToday
18	1.425929	RainTomorrow
19	1.003560	Day

After dropping Pressure9am, Temp9am can be seen having high VIF,hence dropping that column.

- Dropped the column ‘Temp9am’ and check the VIF again

```
#Droping Temp9am column
X = X.drop(["Temp9am"],axis=1)
```

```
# Checking VIF again
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif=pd.DataFrame()
vif["vif_Features"]=[variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
vif["Features"]=X.columns
vif
```

	vif_Features	Features
0	1.171479	Location
1	4.383313	MinTemp
2	5.307619	MaxTemp
3	1.476112	Evaporation
4	1.640820	Sunshine
5	1.591264	WindGustDir
6	2.140971	WindGustSpeed
7	1.360822	WindDir9am
8	1.419722	WindDir3pm
9	1.771780	WindSpeed9am
10	1.902122	WindSpeed3pm
11	2.401281	Humidity9am
12	3.093676	Humidity3pm
13	1.429485	Pressure3pm
14	1.843306	Cloud9am
15	1.736201	Cloud3pm
16	1.342847	RainToday
17	1.425878	RainTomorrow
18	1.003339	Day

After dropping Temp9am, Rainfall can be seen having high VIF. But the Rainfall is well positively correlated with target 'RainTomorrow'. So we shall not drop this column

The multicolinearity has been treated to some extent, But some features still have VIF values above 4. But dropping too many columns have a negative impact on the model. So letting them be for now and moving forward with next step.

- Here the target has continuous data. Hence this is a Regression problem. We have to use the regression algorithms in machine learning for training and testing.

- Importing the necessary libraries and machine learning algorithms for model training and testing

```
#importing necessary libraries
import warnings
warnings.simplefilter("ignore")
warnings.filterwarnings("ignore")
import joblib
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR
from sklearn.linear_model import LinearRegression,Ridge, Lasso
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import SGDRegressor
from sklearn.metrics import classification_report
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.ensemble import GradientBoostingRegressor
from xgboost import XGBRegressor
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import BaggingRegressor
from sklearn import metrics
```

- I have used necessary steps for finding the random state and Accuracy as follows

```
maxAccu=0
maxRS=0
for i in range(1,200):
    X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=.30, random_state=i)
    mod = RandomForestRegressor()
    mod.fit(X_train, y_train)
    pred = mod.predict(X_test)
    acc=r2_score(y_test, pred)
    if acc>maxAccu:
        maxAccu=acc
        maxRS=i
print("Maximum r2 score is ",maxAccu," on Random_state ",maxRS)
```

```
Maximum r2 score is  0.8865274639535751  on Random_state  195
```

- I have checked for the feature importance using random forest classifier as follows

Important Features

```
# Lets check the feature importance using Random Forest Regressor

rfr = RandomForestRegressor()
rfr.fit(X_train, y_train)
importances = pd.DataFrame({'Features':X.columns, 'Importance':np.round(rfr.feature_importances_,3)})
importances = importances.sort_values('Importance', ascending=False).set_index('Features')
importances
```

Features	Importance
RainToday	0.807
Humidity9am	0.021
MinTemp	0.019
Pressure3pm	0.015
Day	0.014
MaxTemp	0.013
Humidity3pm	0.013
WindSpeed9am	0.012
WindGustSpeed	0.012
WindDir9am	0.010
WindSpeed3pm	0.010
WindDir3pm	0.009
WindGustDir	0.008
Sunshine	0.008
Evaporation	0.008
Location	0.007
Cloud9am	0.006
Cloud3pm	0.006
RainTomorrow	0.002

- Then I have checked the accuracy of the model using Random Forest Regressor

Random Forest Regressor

```
# Checking R2 score for Random Forest Regressor
rf=RandomForestRegressor()
rf.fit(X_train,y_train)

# prediction
predrf=rf.predict(X_test)
print('R2_Score:',r2_score(y_test,predrf))

# Metric evaluation
print('MAE:',metrics.mean_absolute_error(y_test, predrf))
print('MSE:',metrics.mean_squared_error(y_test, predrf))
print("RMSE:",np.sqrt(metrics.mean_squared_error(y_test, predrf)))
```

```
R2_Score: 0.8753613714612437
MAE: 0.1582942923305902
MSE: 0.09750985056976842
RMSE: 0.3122656730570435 6
```

- We can observe that rfr has an accuracy of 87.53%

- Then I have checked the accuracy of the model using Decision Tree Regressor

Decision Tree Regressor

```
# Checking R2 score for Decision Tree Regressor
dt=DecisionTreeRegressor()
dt.fit(X_train,y_train)

# prediction
preddt=dt.predict(X_test)
print('R2_Score:',r2_score(y_test,preddt))

# Metric evaluation
print('MAE:',metrics.mean_absolute_error(y_test, preddt))
print('MSE:',metrics.mean_squared_error(y_test, preddt))
print("RMSE:",np.sqrt(metrics.mean_squared_error(y_test, preddt)))
```

```
R2_Score: 0.7639064287809314
MAE: 0.17740681301562877
MSE: 0.1847055693724667
RMSE: 0.42977385840982313
```

- We can observe that DTR has an accuracy of 76.39%

- Then I have checked the accuracy of the model using Gradient Boosting Regressor

GradientBoosting Regressor

```
# Checking R2 score for GradientBoosting Regressor
gb=GradientBoostingRegressor()
gb.fit(X_train,y_train)

# prediction
predgb=gb.predict(X_test)
print('R2_Score:',r2_score(y_test,predgb))

# Metric Evaluation
print('MAE:',metrics.mean_absolute_error(y_test, predgb))
print('MSE:',metrics.mean_squared_error(y_test, predgb))
print("RMSE:",np.sqrt(metrics.mean_squared_error(y_test, predgb)))
```

```
R2_Score: 0.8500224268650491
MAE: 0.18311508792856596
MSE: 0.1173335338863919
RMSE: 0.3425398281753406
```

- We can observe that GBR has an accuracy of 85%

- Then I have checked the accuracy of the model using Bagging Regressor

BaggingRegressor

```
br=BaggingRegressor()
br.fit(X_train,y_train)

# prediction
predbr=br.predict(X_test)
print('R2_Score:',r2_score(y_test,predbr))

# Metric Evaluation
print('MAE:',metrics.mean_absolute_error(y_test, predbr))
print('MSE:',metrics.mean_squared_error(y_test, predbr))
print("RMSE:",np.sqrt(metrics.mean_squared_error(y_test, predbr)))
```

```
R2_Score: 0.8618101321531275
MAE: 0.16070829450675272
MSE: 0.10811153429704642
RMSE: 0.32880318474285863
```

- We can observe that Bagging regressor has an accuracy of 86%

- Then I have checked the accuracy of the model using Extra trees Regressor

ExtraTrees Regressor

```
# Checking R2 score for Extra Trees Regressor

et=ExtraTreesRegressor()
et.fit(X_train,y_train)

# prediction
predet=et.predict(X_test)
print('R2_Score:',r2_score(y_test,predet))

# Metric Evaluation
print('MAE:',metrics.mean_absolute_error(y_test, predet))
print('MSE:',metrics.mean_squared_error(y_test, predet))
print("RMSE:",np.sqrt(metrics.mean_squared_error(y_test, predet)))
```

```
R2_Score: 0.8798063889026921
MAE: 0.1381437778365413
MSE: 0.09403233327374917
RMSE: 0.30664691955692164
```

- We can observe that etr has an accuracy of 88%
- Then I have checked the accuracy of the model using XGB Regressor

XGB Regressor

```
# Checking R2 score for XGB Regressor
from xgboost import XGBRegressor as xgb

xgb=xgb(verbosity=0)
xgb.fit(X_train,y_train)

# prediction
predxgb=xgb.predict(X_test)
print('R2_Score:',r2_score(y_test,predxgb))

# Metric Evaluation
print('MAE:',metrics.mean_absolute_error(y_test, predxgb))
print('MSE:',metrics.mean_squared_error(y_test, predxgb))
print("RMSE:",np.sqrt(metrics.mean_squared_error(y_test, predxgb)))
```

```
R2_Score: 0.8631352229901986
MAE: 0.16357447246401977
MSE: 0.1070748620307595
RMSE: 0.3272229546207899
```

- We can observe that XGB has an accuracy of 86%
- Then I have checked for the Cross validation scores for all the above models as follows

Cross-Validation

```
# Checking cv score for Random Forest Regressor
print('Random Forest:',cross_val_score(rf,x,y,cv=5).mean())

# Checking cv score for Decision Tree Regressor
print('Decision Tree:',cross_val_score(dt,x,y,cv=5).mean())

# Checking cv score for Gradient Boosting Regressor
print('Gradient Boosting:',cross_val_score(gb,x,y,cv=5).mean())

# Checking cv score for Bagging Regressor
print('Bagging Regressor:',cross_val_score(br,x,y,cv=5).mean())

# Checking cv score for ExtraTreesRegressor
print('ExtraTreesRegressor:',cross_val_score(et,x,y,cv=5).mean())

# Checking cv score for XGBRegressor
print('XGBRegressor:',cross_val_score(xgb,x,y,cv=5).mean())
```

```
Random Forest: 0.8527981120632908
Decision Tree: 0.7287125532957052
Gradient Boosting: 0.8302161076300514
Bagging Regressor: 0.8443872994654067
ExtraTreesRegressor: 0.8625591396271665
XGBRegressor: 0.8370112418236797
```

- From the above we can observe the difference between the accuracy score and cross validation score is least in Random forestregressor. Hence we can predict that Random forestregressor is the best model.

- Then I have tuned the selected best model using Hyper parameter tuning for better performance as follows

Hyper parameter tuning

```
#Importing necessary Libraries
from sklearn.model_selection import GridSearchCV
```

```
# Giving RFR parameters.
parameters = {'n_estimators' : [50,100,200],
              'criterion' :['mse', 'mae'],
              'max_depth' : [4, 6, 8]}
```

- I have tuned the extra trees classifier and fit the current model for better performance as follows

```
GCV=GridSearchCV(rf,parameters,cv=5)
```

```
# Tuning the model using GCV.
GCV.fit(X_train,y_train)
```

```
GridSearchCV(cv=5, estimator=RandomForestRegressor(),
             param_grid={'criterion': ['mse', 'mae'], 'max_depth': [4, 6, 8],
                         'n_estimators': [50, 100, 200]})
```

```
# Getting the best parameters for RFR.
GCV.best_params_
```

```
{'criterion': 'mse', 'max_depth': 8, 'n_estimators': 200}
```

- This has resulted in the accuracy as 85.88%, which is less than an expected level of accuracy. But due to lots of missing data and multi collinearity rising from filling the missing data using imputation has resulted in lesser accuracy.

- Then I have saved the model importing and using joblib as follows

Saving the Model

```
import joblib
joblib.dump(Rainfall,"Prediction_of_Rainfall.pkl")
```

```
['Prediction_of_Rainfall.pkl']
```

- Predicted using the saved model as follows

Predicting from the saved model

```
# Loading the saved model
model=joblib.load("Prediction_of_Rainfall.pkl")
```

```
#Prediction
prediction = model.predict(X_test)
prediction
```

```
array([0.02526946, 0.02135759, 2.28919542, ..., 0.04496693, 0.0307711 ,
       0.03075498])
```

- I have created a dataframe of the actual vs predicted values for better view and comparison of outputs from the predicted model

```
pd.DataFrame([model.predict(X_test)[:],y_test[:]],index=["Predicted","Original"]).T
```

	Predicted	Original
0	0.025269	0.000000
1	0.021358	0.000000
2	2.289195	2.721295
3	0.040323	0.000000
4	0.020629	0.000000
5	0.039355	0.000000
6	0.046579	0.000000
7	0.008912	0.000000
8	0.009836	0.000000
9	0.079405	0.182322
10	1.809326	2.151762

- We can see that the Actual Values and Predicted Values look similar with few exceptions.

Conclusions

Key Findings and Conclusions of the Study:

Findings:

- In this project we have investigated the Rainfall and developed new knowledge to understand the most important factors influencing Rainfall prediction for weather forecasting.

- This project aimed to enhance prior understanding of how Rainfall prediction is dependent on temperature, pressure, and other factors for forecasting.

- The factors like 'RainToday' ,'Huidity','Temperature' had a positive impact on the Rainfall and RainTomorrow

- Thus one needs to pay attention to these factors more specifically and seek breakthroughs that can improve its Predictability and help in weather forecasting.

CONCLUDING REMARKS

- The endeavor of this study is to identify the factors influencing rainfall tomorrow and the intensity of rainfall tomorrow in mm.

- In this project, I have done some feature engineering by using imputation methods to fill the null values in the data. Visualized the data using count plot, factor plot, Scatter plot, bar plot and distribution plot, also encoded the object data into numerical using label encoding method. Checked the statistical summary of the dataset and checked for skewness, outliers and correlation between the features.

- From the analysis it was found that rainfall prediction for tomorrow is dependent on various factors. All these factors

influence the rainfall tomorrow. Majorly the Rainfall today and the Humidity, Pressure, Temperature and the clouds had a great influence on the rainfall tomorrow.

- Also the windspeed and wind direction has an impact on the rainfall tomorrow. Whereas the Day, Month, Year and Minimum temperature has least impact on the rainfall tomorrow. Hence some of these columns were dropped.

- Though the sunshine has a great importance in the prediction of rainfall tomorrow, it is seen that nearly 50% of its data is missing, we had to fill those values using imputation and this had a negative impact on the modelling as the data is more skewed due to the same values being repeated in 50% of the rows.

- After visualizing the data, I found that CoffsHarbour has higher rainfall with higher chance of rain tomorrow and Uluru has lowest rainfall with high chance of rain tomorrow compared to others and RainToday has higher chance of impacting rainfall tomorrow.

- Also, Melbourne has high rainfall with lower chance of rain tomorrow and Uluru has lower rainfall and there is no chance for rain tomorrow.

- Most of the rainfall is lower level and is densely populated at the lower level of evaporation. The more the evaporation, more the chance of rain tomorrow. As the sunshine is lower there is more chance of rainfall and rain tomorrow.