<u>Capstone Project:</u>

<u>Interim Report:</u>

Batch details	PGDSE OFFLINE JUL-2023
Team members	Dhanapallan E Swathika V L Priyadharshini S Hariharan V Anas R
Domain of Project	Meteorology
Proposed project title	Australian Rainfall Predictions
Group Number	Group 1
Team Leader	Dhanapallan E
Mentor Name	Mr. Ankush Bansal

Date: 24-01-2023

Signature of the Mentor

Ankush Bansal

Signature of the Team Leader

Dhanapallan E

Table of Contents:

1 Industry Review:

1.1 Abstract:

The accurate prediction of rainfall in Australia is pivotal for various stakeholders, impacting agricultural planning, water resource management, and overall climate resilience. This capstone project seeks to leverage historical meteorological data and pertinent features, including geographic location, temporal variables, and atmospheric conditions, to construct a machine learning model for predicting rainfall in different regions of Australia. By training on a comprehensive dataset of historical rainfall patterns, the model aims to offer precise predictions that can aid farmers, water authorities, and environmental researchers in making informed decisions. The project's deliverable is an advanced predictive model capable of estimating rainfall, providing a valuable tool for a wide range of applications, including agricultural planning, flood prediction, and natural resource management across diverse Australian landscapes.

1.2 Current practice:

In contemporary practices related to the classification of Australian rainfall using machine learning models, a diverse set of supervised learning algorithms is commonly employed. These algorithms, including Decision Trees, Random Forest, Support Vector Machines (SVM), and Neural Networks, are trained on extensive datasets derived from historical meteorological records. The datasets encompass crucial features such as geographical location, temporal variables, and atmospheric conditions, contributing to the development of robust classification models.

Feature engineering techniques play a pivotal role in refining the input data for machine learning models. Analysts utilize these techniques to select and transform relevant features, extracting meaningful information such as temporal patterns, seasonality, and spatial characteristics. This step ensures that the models can effectively discern and classify varying rainfall events.

In addition to supervised learning, researchers explore the application of unsupervised learning techniques to enhance the accuracy of rainfall classification models. Clustering methods, such as K-Means, and anomaly detection algorithms are leveraged to identify patterns, clusters, and outliers within the data. This nuanced approach aids in improving the models' ability to categorize rainfall events accurately, particularly in the identification of atypical occurrences.

Temporal and spatial analysis constitute critical components of the classification process. Time series classification is employed to capture seasonality and long-term trends, while spatial analysis techniques are integrated to account for regional variations in rainfall patterns. These considerations ensure that the classification models provide nuanced and accurate predictions tailored to the diverse geography of Australia

Ensemble learning approaches, involving the combination of predictions from multiple classification models, are explored to enhance accuracy and robustness. Evaluation metrics such as accuracy, precision, recall, and F1 score are commonly utilized to assess the performance of these models, ensuring a comprehensive understanding of their effectiveness in classifying rainfall events.

Continuous monitoring and updating mechanisms are implemented to evaluate model performance over time and adapt to evolving climatic conditions. This iterative process ensures that the classification models remain accurate and relevant as new data becomes available. Lastly, interdisciplinary collaborations between meteorologists, data scientists, and machine learning experts are encouraged to foster a holistic and informed approach to rainfall classification in Australia.

1.3 Problem Statement:

Business Problem Understanding: The business problem for rainfall prediction in Australia arises from the need to anticipate and plan for weather conditions. Predicting rainfall accurately is crucial for various sectors such as agriculture, water resource management, and disaster preparedness. A machine learning model can provide valuable insights into future rainfall, enabling proactive decision-making.

Business Objective: The business objective of predicting rainfall in Australia using machine learning is to develop a reliable model that can forecast precipitation patterns. This model aims to assist farmers, water resource managers, and disaster response teams in making informed decisions, thereby enhancing overall preparedness and resource allocation.

Approach: The approach for predicting rainfall in Australia using machine learning involves gathering extensive data on meteorological variables, pre-processing and cleaning the dataset, selecting suitable algorithms, training and validating the model, and evaluating its performance on new data. The focus is on understanding the complex interactions among various factors influencing rainfall to create an accurate predictive model.

Conclusions: In conclusion, applying machine learning to predict rainfall in Australia holds immense practical significance. It can empower stakeholders with timely and accurate information, allowing for better planning and response to weather events. The utilization of machine learning in the realm of rainfall prediction has the potential to revolutionize how we manage resources and mitigate the impact of weather-related challenges, contributing to a more resilient and adaptive society.

1.4 Impact on Business:

- ***Increased Agricultural Productivity:*** Accurate rainfall prediction in Australia through machine learning can significantly enhance efficiency in agricultural planning. Farmers can make informed decisions about crop planting and irrigation, optimizing resource utilization and maximizing yields. This, in turn, contributes to increased overall efficiency in the agricultural sector.
- ***Improved Water Resource Management:*** Predicting rainfall patterns using machine learning allows for better management of water resources. Water authorities and conservation agencies can anticipate periods of drought or heavy rainfall, enabling strategic planning for water distribution, reservoir management, and drought response. This proactive approach contributes to improved water resource sustainability.
- ***Enhanced Disaster Preparedness:*** Machine learning-based rainfall prediction plays a crucial role in disaster preparedness. By accurately forecasting rainfall, emergency response teams can anticipate potential floods or landslides, enabling timely evacuation plans and resource allocation. This proactive approach enhances overall disaster preparedness and response effectiveness.
- ***Cost Savings in Infrastructure Planning:*** Accurate rainfall prediction assists in infrastructure planning by mitigating the risk of weather-related damage. For example, urban planners

can optimize drainage systems and construction projects based on anticipated rainfall levels, reducing the likelihood of costly repairs and enhancing the resilience of infrastructure to weather events.

2 Dataset and pre-processing :

2.1 Project statement, complexity and outcomes:

- ***Project Statement:*** The analysis of the Australian rain dataset is justified by the critical need for accurate and timely rainfall predictions in Australia. The dataset's comprehensive information, including 18 influential meteorological factors, aligns with the imperative to understand and forecast rainfall patterns, addressing challenges in agriculture, water resource management, and disaster preparedness.
- ***Complexity Involved:*** The project involves dealing with the complexity of diverse meteorological variables and their interactions, requiring advanced analytical techniques, and machine learning models. The intricacies of Australian weather conditions, varying across regions, contribute to the complexity, demanding a nuanced approach for accurate predictions.
- ***Social Value:*** The social value of this project lies in its potential to enhance public safety and well-being. Accurate rainfall predictions support disaster preparedness, helping communities anticipate and respond to weather-related challenges, such as floods and droughts. Additionally, the agricultural sector benefits from improved planning, contributing to food security and economic stability.
- ***Commercial Value:*** From a commercial perspective, accurate rainfall predictions provide value to industries dependent on weather conditions, such as agriculture and insurance. Farmers can optimize crop planning, and insurers can better assess and

manage risks associated with extreme weather events. This contributes to increased efficiency and financial stability in relevant sectors.

- ***Academic Value:*** In an academic context, the project adds value by contributing to the field of meteorological research and data science. The exploration of advanced analytical techniques and machine learning algorithms for rainfall prediction in a diverse geographic region like Australia presents opportunities for academic advancements, potentially leading to new methodologies and insights in the field.

2.2 **Basic dataset information :**

Dataset name and link	Australian rain data, Kaggle
Total number of rows present	145460
Total number of columns present	23
Total size	3345580
Total number of numeric columns	15
Total number of categorical columns	8
Dependent variable	Rain Tomorrow

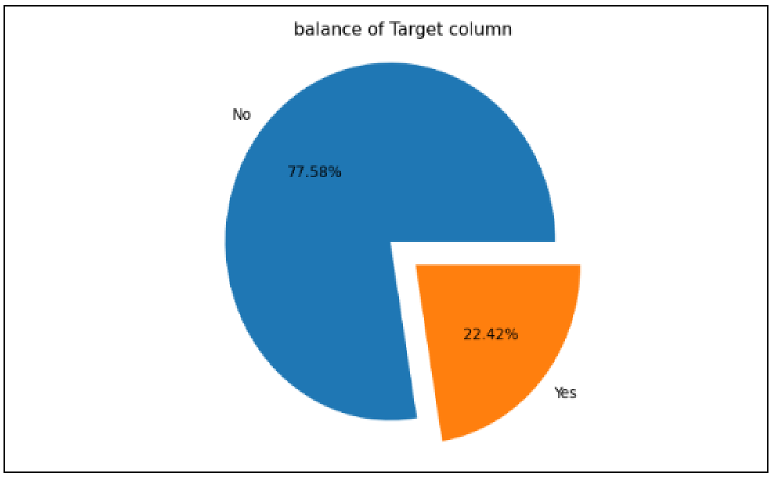
2.3 Data dictionary :

- **Date**: The date of the weather observations.
- **Location**: The location (city or region) where the weather observations were recorded.
- **Min Temp**: The minimum temperature recorded on the given day.
- **Max Temp**: The maximum temperature recorded on the given day.
- **Rainfall**: The amount of rainfall recorded in millimetres.
- **Evaporation**: The amount of water, measured in millimetres that evaporated on the given day.
- **Sunshine**: The number of hours of bright sunshine in the day.
- **Wind Gust Dir**: The direction of the strongest wind gust in compass points.
- **Wind Gust Speed**: The speed of the strongest wind gust in kilometres per hour.
- **WindDir9am**: The direction of the wind at 9 am in compass points.
- **WindDir3pm**: The direction of the wind at 3 pm in compass points.
- **WindSpeed9am**: Wind speed at 9 am in kilometres per hour.
- **WindSpeed3pm**: Wind speed at 3 pm in kilometres per hour.
- **Humidity9am**: Humidity at 9 am as a percentage.
- **Humidity3pm**: Humidity at 3 pm as a percentage.
- **Pressure9am**: Atmospheric pressure at 9 am in hPa (hectopascals).
- **Pressure3pm**: Atmospheric pressure at 3 pm in hPa.
- **Cloud9am**: cloud cover observed at 9 am.
- **Cloud3pm**: cloud cover observed at 3 pm.
- **Temp9am**: Temperature at 9 am in degrees Celsius.
- **Temp3pm**: Temperature at 3 pm in degrees Celsius.

- **Rain Today**: Variable indicating whether it rained today.
- **Rain Tomorrow**: Variable indicating whether it will rain tomorrow (the target variable).

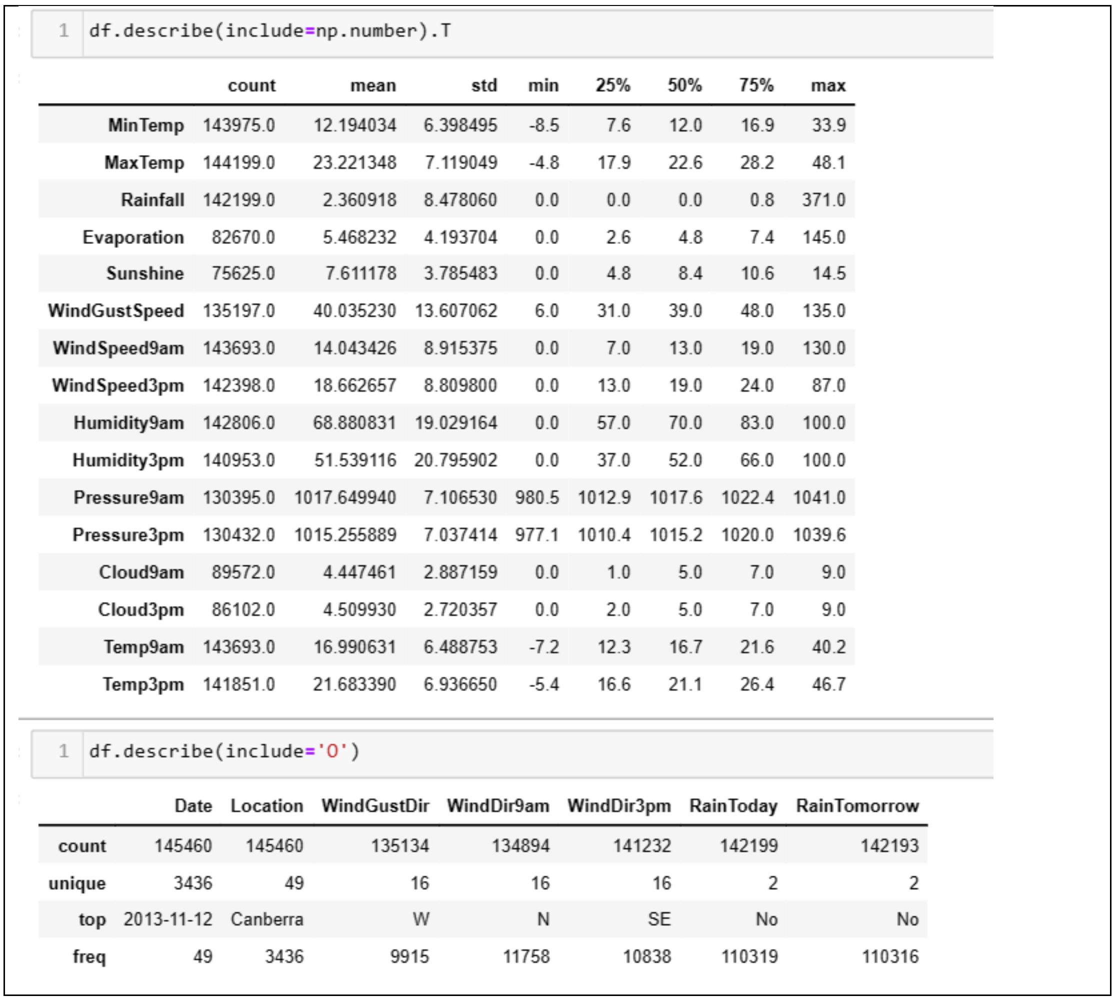
3 Exploratory data analysis :

3.1 Target variable classes:



- The distribution of our target variable "Rain tomorrow" reveals a predominance of 77% for "No" and 22% for "Yes." Despite the dataset's inherent imbalance, we are advancing with the analysis under the assumption that the dataset is effectively balanced.

3.2 Basic statistics of numerical and categorical variables :



```
df.describe(include=np.number).T
```

	count	mean	std	min	25%	50%	75%	max
MinTemp	143975.0	12.194034	6.398495	-8.5	7.6	12.0	16.9	33.9
MaxTemp	144199.0	23.221348	7.119049	-4.8	17.9	22.6	28.2	48.1
Rainfall	142199.0	2.360918	8.478060	0.0	0.0	0.0	0.8	371.0
Evaporation	82670.0	5.468232	4.193704	0.0	2.6	4.8	7.4	145.0
Sunshine	75625.0	7.611178	3.785483	0.0	4.8	8.4	10.6	14.5
WindGustSpeed	135197.0	40.035230	13.607062	6.0	31.0	39.0	48.0	135.0
WindSpeed9am	143693.0	14.043426	8.915375	0.0	7.0	13.0	19.0	130.0
WindSpeed3pm	142398.0	18.662657	8.809800	0.0	13.0	19.0	24.0	87.0
Humidity9am	142806.0	68.880831	19.029164	0.0	57.0	70.0	83.0	100.0
Humidity3pm	140953.0	51.539116	20.795902	0.0	37.0	52.0	66.0	100.0
Pressure9am	130395.0	1017.649940	7.106530	980.5	1012.9	1017.6	1022.4	1041.0
Pressure3pm	130432.0	1015.255889	7.037414	977.1	1010.4	1015.2	1020.0	1039.6
Cloud9am	89572.0	4.447461	2.887159	0.0	1.0	5.0	7.0	9.0
Cloud3pm	86102.0	4.509930	2.720357	0.0	2.0	5.0	7.0	9.0
Temp9am	143693.0	16.990631	6.488753	-7.2	12.3	16.7	21.6	40.2
Temp3pm	141851.0	21.683390	6.936650	-5.4	16.6	21.1	26.4	46.7

```
df.describe(include='O')
```

	Date	Location	WindGustDir	WindDir9am	WindDir3pm	RainToday	RainTomorrow
count	145460	145460	135134	134894	141232	142199	142193
unique	3436	49	16	16	16	2	2
top	2013-11-12	Canberra	W	N	SE	No	No
freq	49	3436	9915	11758	10838	110319	110316

Basic statistical inference from numerical data:

- MinTemp ranges from -8.5°C to a maximum of 33.9°C and MaxTemp vary from -4.8°C to a maximum of 48.1°C.
- Temp9am span from -7.2°C to 40.2°C and Temp3pm range from -5.4°C to 46.7°C.
- Rainfall varies widely, with a minimum of 0.0mm and a maximum of 371.0mm.The mean rainfall is 2.36mm, and the distribution appears positively skewed (mean > median).

- Wind speeds at 9 am range from 0.0 to 130.0 km/h and Wind speeds at 3 pm vary from 0.0 to 87.0 km/h.
- The maximum gust speed reaches up to 135.0 km/h.
- Morning humidity at 9am and Afternoon humidity at 3pm ranges from 0% to 100%.
- Morning atmospheric pressure at 9am ranges from 980.5 hPa to 1041.0 hPa and Afternoon atmospheric pressure at 3pm varies from 977.1 hPa to 1039.6 hPa.
- Cloud Cover ranges from 0.0 to 9.0.
- Evaporation ranges from 0.0 to 145.0 and Sunshine hours vary from 0.0 to 14.5 hours.

Basic statistical inference from categorical data:

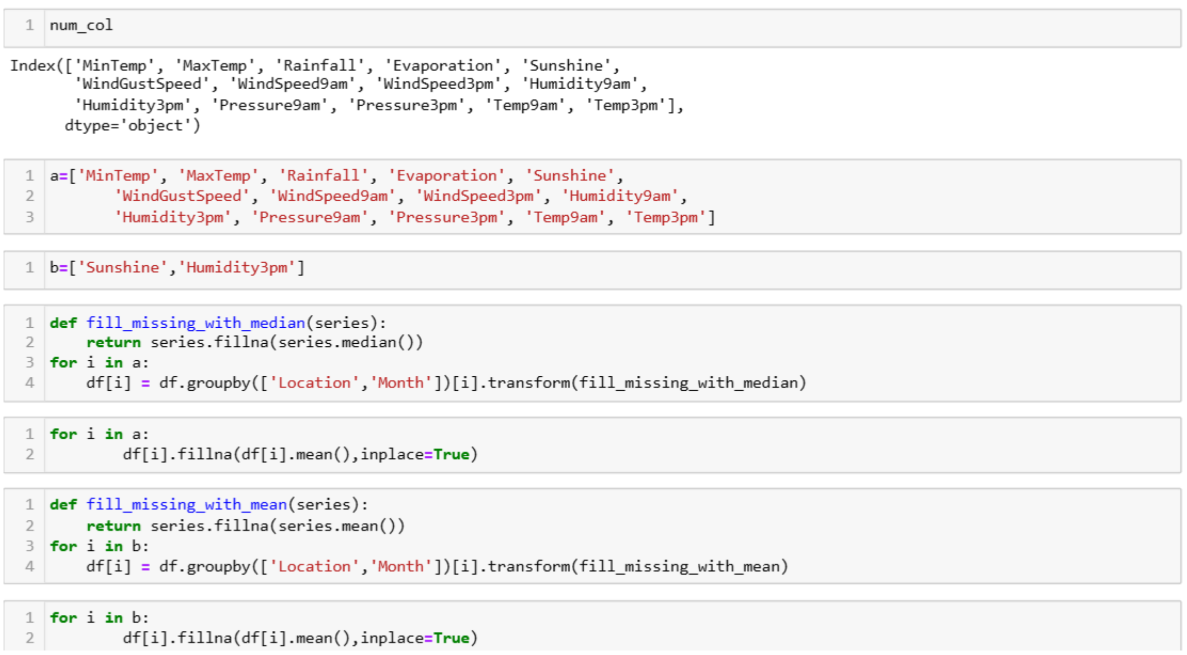
- Date-wise, November 12, 2013, stands out with 49 recorded instances, potentially indicating a data collection anomaly or significant weather event on that day.
- The dataset includes information from 49 unique locations. Canberra is the most frequently represented location, occurring 3,436 times.
- There are 16 unique wind directions recorded in each of the three wind-related columns. The most common wind direction for WindGustDir, WindDir9am, and WindDir3pm is 'W' (west).
- There are two unique values for both RainToday and RainTomorrow ('Yes' and 'No'). The most frequent occurrence for both RainToday and RainTomorrow is 'No', indicating that rain is not expected. The prevalence of 'No' in RainToday and RainTomorrow suggests a bias towards non-rainy days, highlighting potential class imbalance.

3.3 Null values :

Addressing null values is crucial in data analysis to ensure accurate insights and reliable results by either imputing missing data or employing appropriate handling techniques.

Column name	**Null value count**
Date	0
Location	0
MinTemp	1485
MaxTemp	1261
Rainfall	3261
Evaporation	62790
Sunshine	69835
WindGustDir	10326
WindGustSpeed	10263
WindDir9am	10566
WindDir3pm	4228
WindSpeed9am	1767
WindSpeed3pm	3062
Humidity9am	2654
Humidity3pm	4507
Pressure9am	15065
Pressure3pm	15028
Cloud9am	55888
Cloud3pm	59358
Temp9am	1767
Temp3pm	3609
RainToday	3261
RainTomorrow	3267
Week	0
Year	0
Month	0

• Null values in the "Rain Tomorrow" column, i.e dependent variable, were removed.
• Missing values in categorical columns were imputed with the mode by grouping locations and month.
• For numerical columns, null values were addressed by first separating the variables based on outlier and then it was grouped by location and month, finally it was imputed by the median and mean.



```
num_col
```

```
Index(['MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine',
       'WindGustSpeed', 'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am',
       'Humidity3pm', 'Pressure9am', 'Pressure3pm', 'Temp9am', 'Temp3pm'],
      dtype='object')
```

```
a=['MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine',
       'WindGustSpeed', 'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am',
       'Humidity3pm', 'Pressure9am', 'Pressure3pm', 'Temp9am', 'Temp3pm']
```

```
b=['Sunshine','Humidity3pm']
```

```
def fill_missing_with_median(series):
    return series.fillna(series.median())
for i in a:
    df[i] = df.groupby(['Location','Month'])[i].transform(fill_missing_with_median)
```

```
for i in a:
        df[i].fillna(df[i].mean(),inplace=True)
```

```
def fill_missing_with_mean(series):
    return series.fillna(series.mean())
for i in b:
    df[i] = df.groupby(['Location','Month'])[i].transform(fill_missing_with_mean)
```

```
for i in b:
        df[i].fillna(df[i].mean(),inplace=True)
```

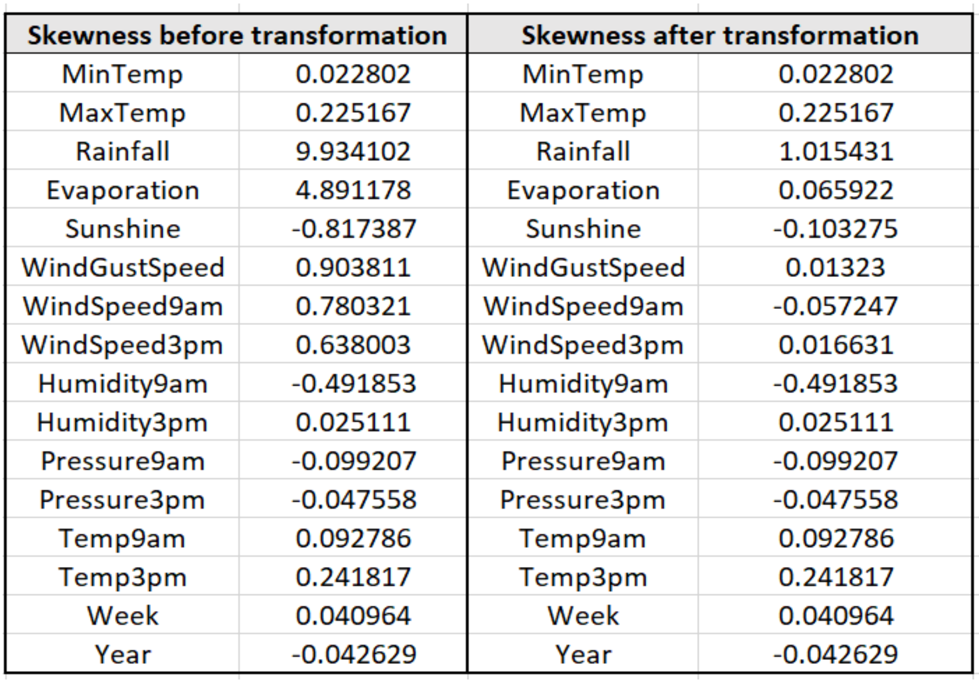
3.4 Outlier Treatment :

Outlier treatment is essential for ensuring data integrity, as it addresses extreme values that can distort statistical measures and impact the performance of machine learning models.

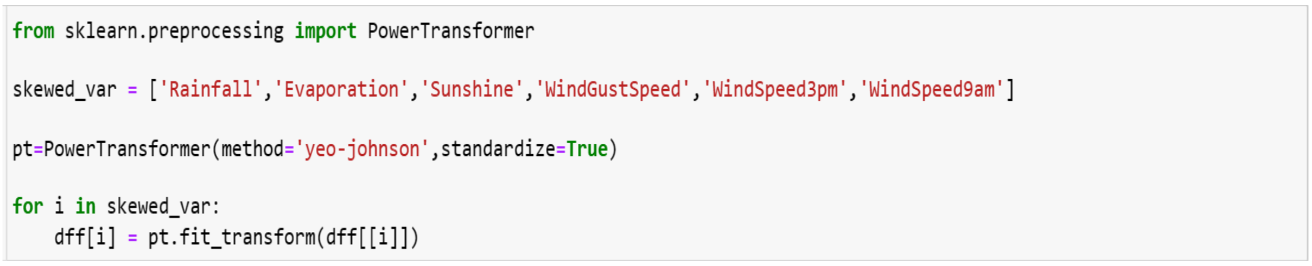
Things followed to treat outliers:

- **Capping:** Employing capping methods to establish predetermined thresholds, constraining outlier values and mitigating their undue impact on the analysis. Nonetheless, for our dataset, capping was not applied; instead, a transformation approach was utilized.
- **Transformation:** Applying data transformation methods to modify the distribution, mitigating the impact of outliers and ensuring a more balanced representation of the dataset. In this data, we have used the Yeo-Johnson power transformer for our data.

```
outliers_rows = pd.DataFrame()
for i in num_col:
  Q1 = df[i].quantile(0.25)
  Q3 = df[i].quantile(0.75)
  IQR = Q3 - Q1
  outliers = ((df[i] < (Q1 - 1.5 * IQR)) | (df[i] > (Q3 + 1.5 * IQR)))
  outliers_rows = pd.concat([outliers_rows, df[outliers]])
```



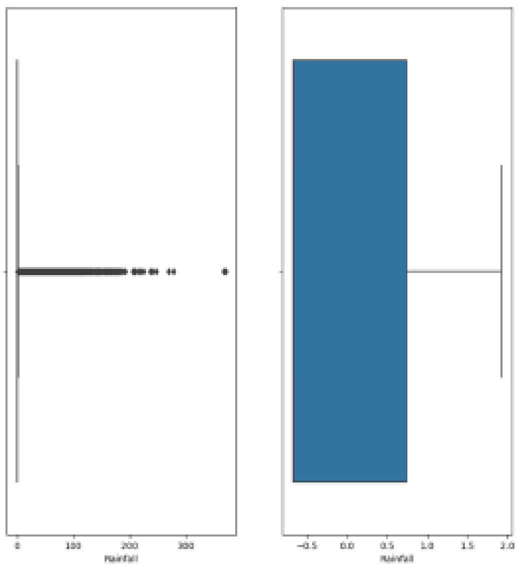
Skewness before transformation		Skewness after transformation	
MinTemp	0.022802	MinTemp	0.022802
MaxTemp	0.225167	MaxTemp	0.225167
Rainfall	9.934102	Rainfall	1.015431
Evaporation	4.891178	Evaporation	0.065922
Sunshine	-0.817387	Sunshine	-0.103275
WindGustSpeed	0.903811	WindGustSpeed	0.01323
WindSpeed9am	0.780321	WindSpeed9am	-0.057247
WindSpeed3pm	0.638003	WindSpeed3pm	0.016631
Humidity9am	-0.491853	Humidity9am	-0.491853
Humidity3pm	0.025111	Humidity3pm	0.025111
Pressure9am	-0.099207	Pressure9am	-0.099207
Pressure3pm	-0.047558	Pressure3pm	-0.047558
Temp9am	0.092786	Temp9am	0.092786
Temp3pm	0.241817	Temp3pm	0.241817
Week	0.040964	Week	0.040964
Year	-0.042629	Year	-0.042629

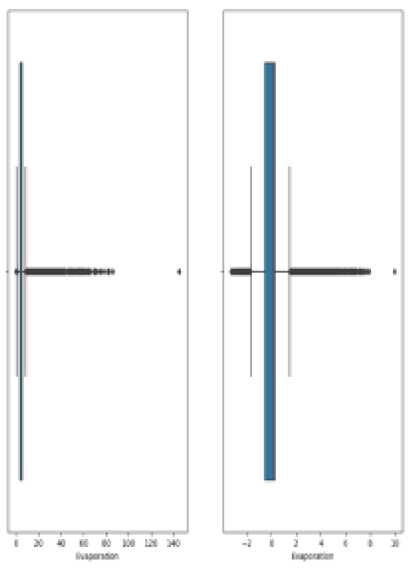


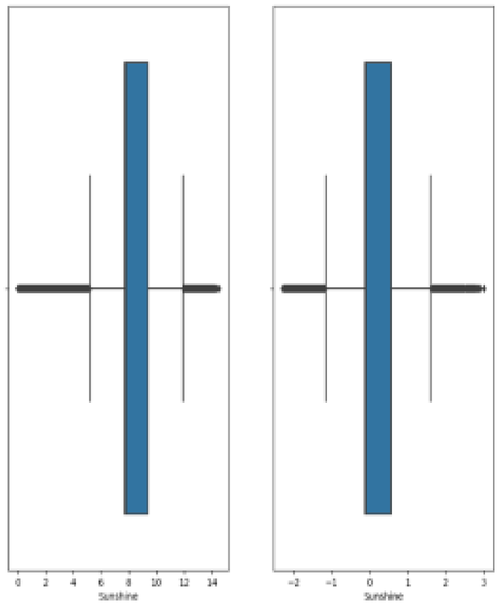
```
from sklearn.preprocessing import PowerTransformer

skewed_var = ['Rainfall','Evaporation','Sunshine','WindGustSpeed','WindSpeed3pm','WindSpeed9am']

pt=PowerTransformer(method='yeo-johnson',standardize=True)

for i in skewed_var:
    dff[i] = pt.fit_transform(dff[[i]])
```

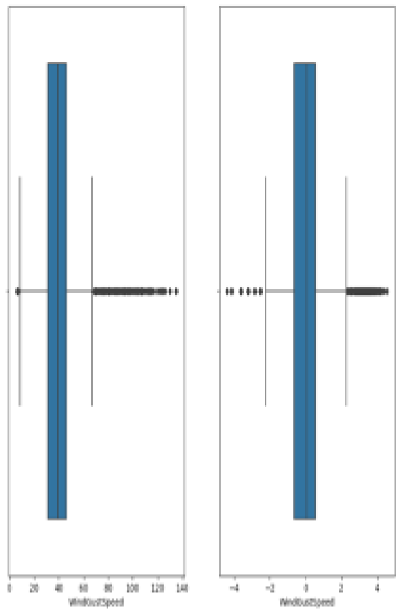
- 'Rainfall,' 'Evaporation,' 'Sunshine,' 'WindGustSpeed,' 'WindSpeed3pm,' and 'WindSpeed9am,' which exhibit skewness, will undergo a transformation using the Yeo-Johnson power transformer.
- The Yeo-Johnson method is chosen because it is effective for both positive and negative values.

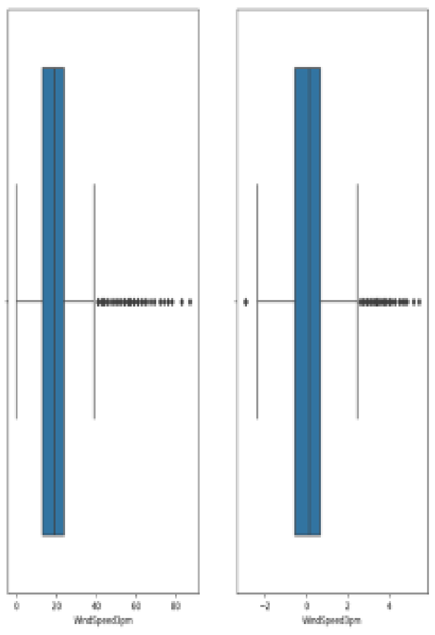
Transformation : Before & After


Rainfall

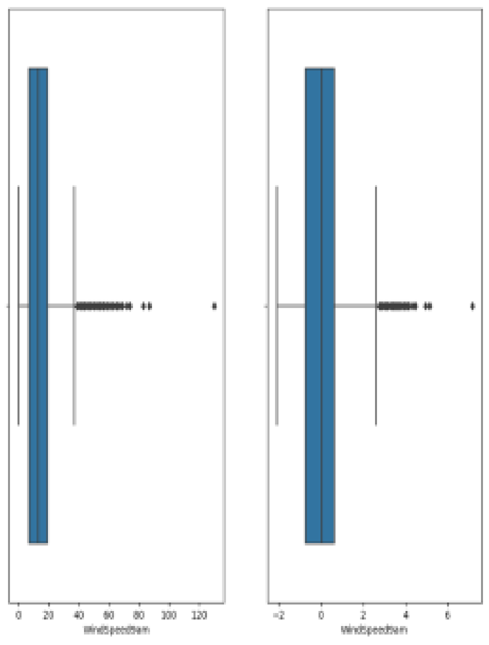

Evaporation


Sunshine


WindGustSpeed

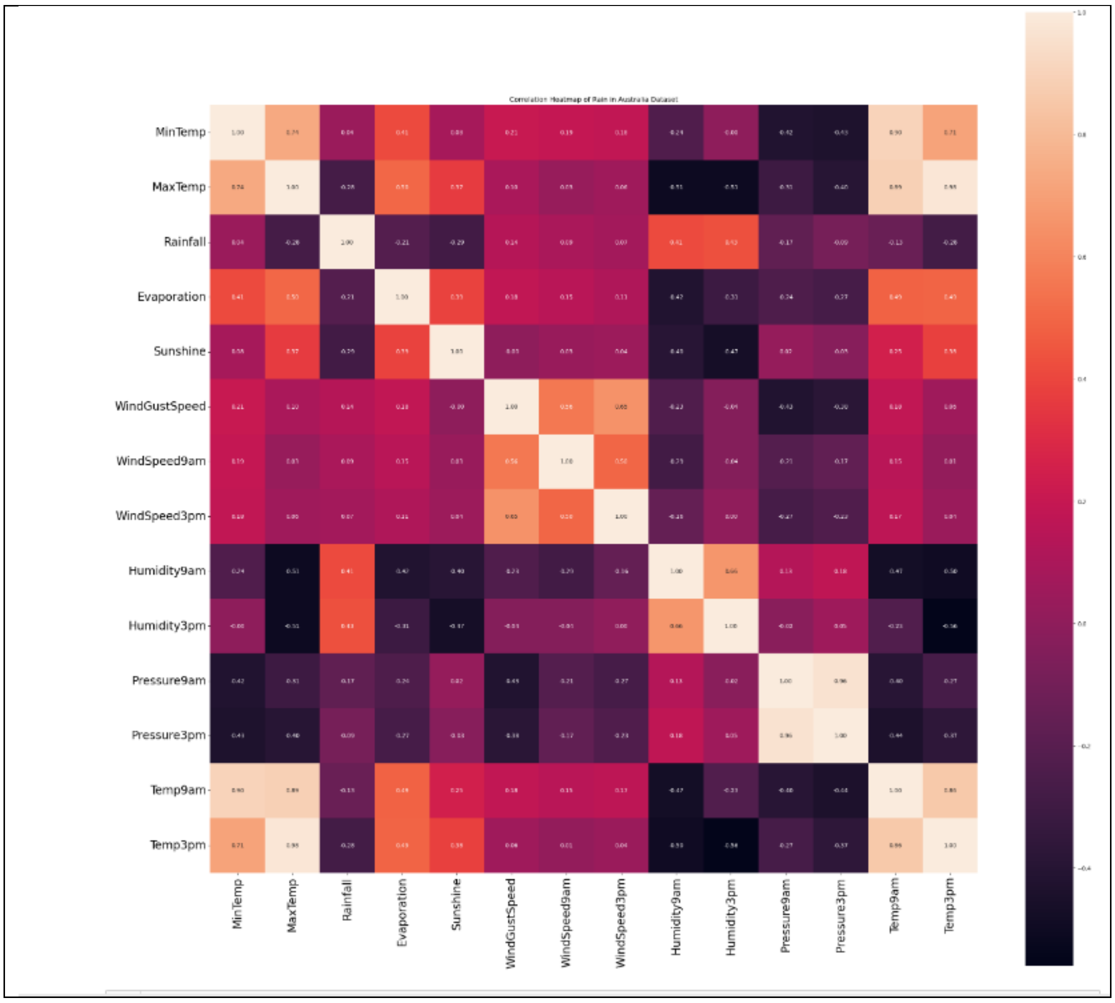


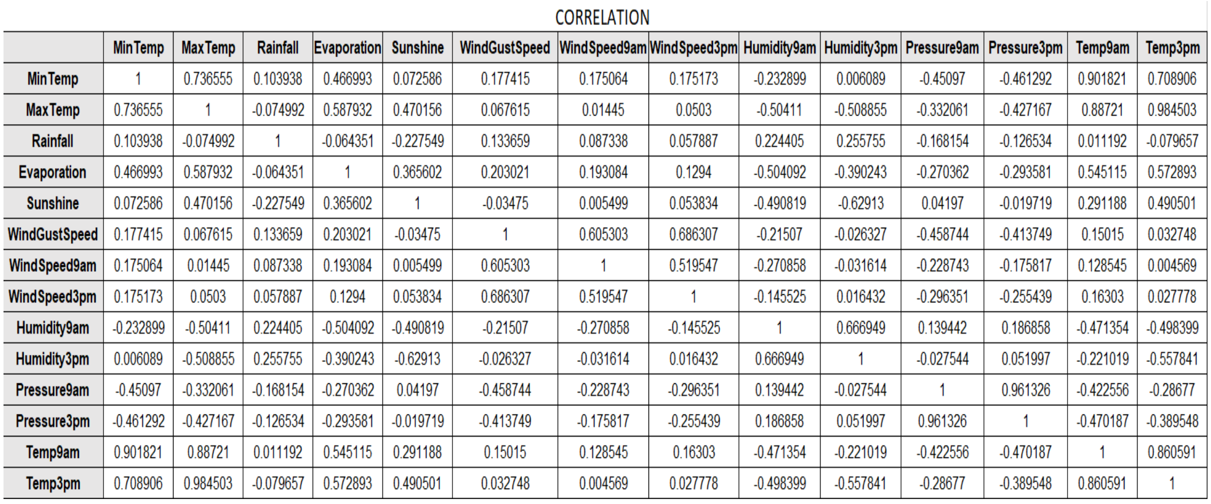
Wind3pm



Windspeed9am

3.5 Correlation between variables :





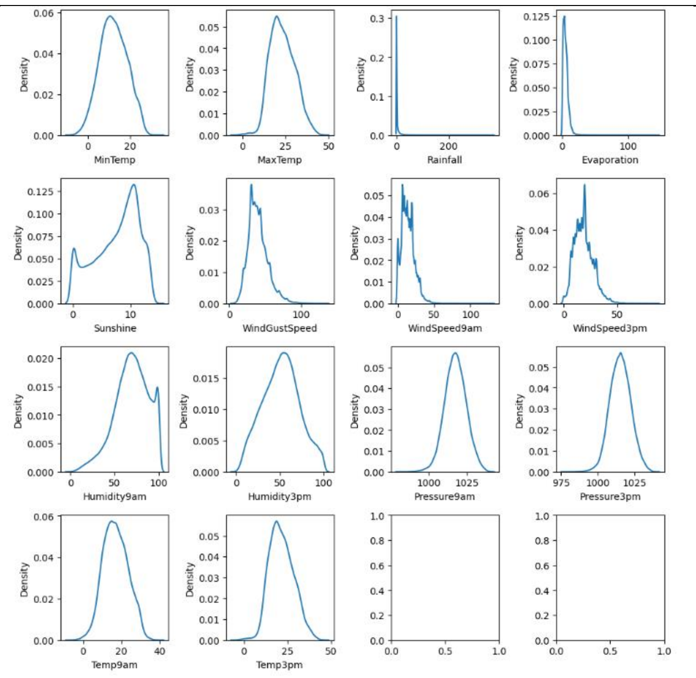
CORRELATION

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Temp9am	Temp3pm
MinTemp	1	0.736555	0.103938	0.466993	0.072586	0.177415	0.175064	0.175173	-0.232899	0.006089	-0.45097	-0.461292	0.901821	0.708906
MaxTemp	0.736555	1	-0.074992	0.587932	0.470156	0.067615	0.01445	0.0503	-0.50411	-0.508855	-0.332061	-0.427167	0.88721	0.984503
Rainfall	0.103938	-0.074992	1	-0.064351	-0.227549	0.133659	0.087338	0.057887	0.224405	0.255755	-0.168154	-0.126534	0.011192	-0.079657
Evaporation	0.466993	0.587932	-0.064351	1	0.365602	0.203021	0.193084	0.1294	-0.504092	-0.390243	-0.270362	-0.293581	0.545115	0.572893
Sunshine	0.072586	0.470156	-0.227549	0.365602	1	-0.03475	0.005499	0.053834	-0.490819	-0.62913	0.04197	-0.019719	0.291188	0.490501
WindGustSpeed	0.177415	0.067615	0.133659	0.203021	-0.03475	1	0.605303	0.686307	-0.21507	-0.026327	-0.458744	-0.413749	0.15015	0.032748
WindSpeed9am	0.175064	0.01445	0.087338	0.193084	0.005499	0.605303	1	0.519547	-0.270858	-0.031614	-0.228743	-0.175817	0.128545	0.004569
WindSpeed3pm	0.175173	0.0503	0.057887	0.1294	0.053834	0.686307	0.519547	1	-0.145525	0.016432	-0.296351	-0.255439	0.16303	0.027778
Humidity9am	-0.232899	-0.50411	0.224405	-0.504092	-0.490819	-0.21507	-0.270858	-0.145525	1	0.666949	0.139442	0.186858	-0.471354	-0.498399
Humidity3pm	0.006089	-0.508855	0.255755	-0.390243	-0.62913	-0.026327	-0.031614	0.016432	0.666949	1	-0.027544	0.051997	-0.221019	-0.557841
Pressure9am	-0.45097	-0.332061	-0.168154	-0.270362	0.04197	-0.458744	-0.228743	-0.296351	0.139442	-0.027544	1	0.961326	-0.422556	-0.28677
Pressure3pm	-0.461292	-0.427167	-0.126534	-0.293581	-0.019719	-0.413749	-0.175817	-0.255439	0.186858	0.051997	0.961326	1	-0.470187	-0.389548
Temp9am	0.901821	0.88721	0.011192	0.545115	0.291188	0.15015	0.128545	0.16303	-0.471354	-0.221019	-0.422556	-0.470187	1	0.860591
Temp3pm	0.708906	0.984503	-0.079657	0.572893	0.490501	0.032748	0.004569	0.027778	-0.498399	-0.557841	-0.28677	-0.389548	0.860591	1

Basic inference from the correlation matrix:

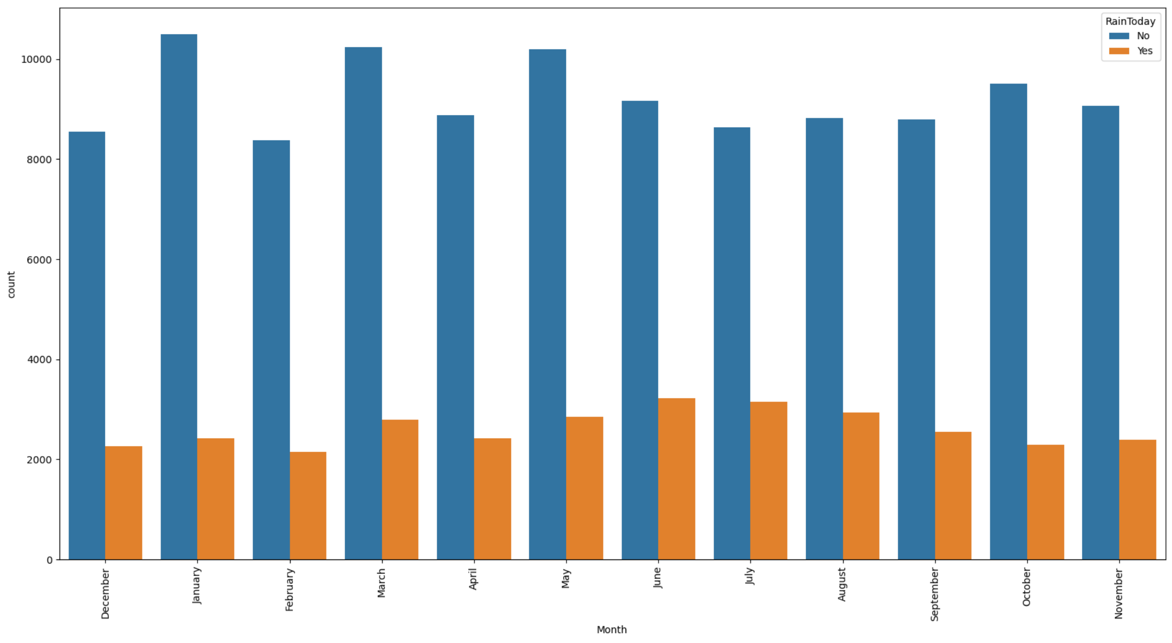
- Strong positive correlations exist between MinTemp and related variables (MaxTemp, Temp9am, Temp3pm), indicating logical dependencies.
- MaxTemp and Temp3pm exhibit a high positive correlation (0.985), reinforcing the close relationship between maximum temperature and afternoon temperature.
- Negative correlations between Humidity9am and temperature variables (MinTemp: -0.233, MaxTemp: -0.504, Temp9am: -0.471) suggest that higher humidity tends to be associated with lower temperatures.
- Negative correlations between Pressure9am and temperature variables (MinTemp: -0.451, MaxTemp: -0.332, Temp9am: -0.423) indicate a potential relationship between lower atmospheric pressure and higher temperatures.
- Positive correlation between WindSpeed9am and WindSpeed3pm (0.519) indicates consistency in wind speeds between morning and afternoon.
- Positive correlations between Evaporation and temperature variables (MinTemp: 0.467, MaxTemp: 0.588, Temp9am: 0.545, Temp3pm: 0.573) suggest a positive relationship between evaporation and temperature.
- Positive correlations between Rainfall and Humidity variables (Humidity9am: 0.224, Humidity3pm: 0.256) suggest that rainy conditions may be associated with higher humidity.
- Negative correlation between Pressure9am and Humidity9am (-0.471) indicates a potential relationship between lower atmospheric pressure and higher morning humidity.
- Negative correlations between Sunshine and Cloud variables (Cloud9am: -0.676, Cloud3pm: -0.704) indicate an inverse relationship between cloud cover and sunshine.

3.6 Distribution of variables:



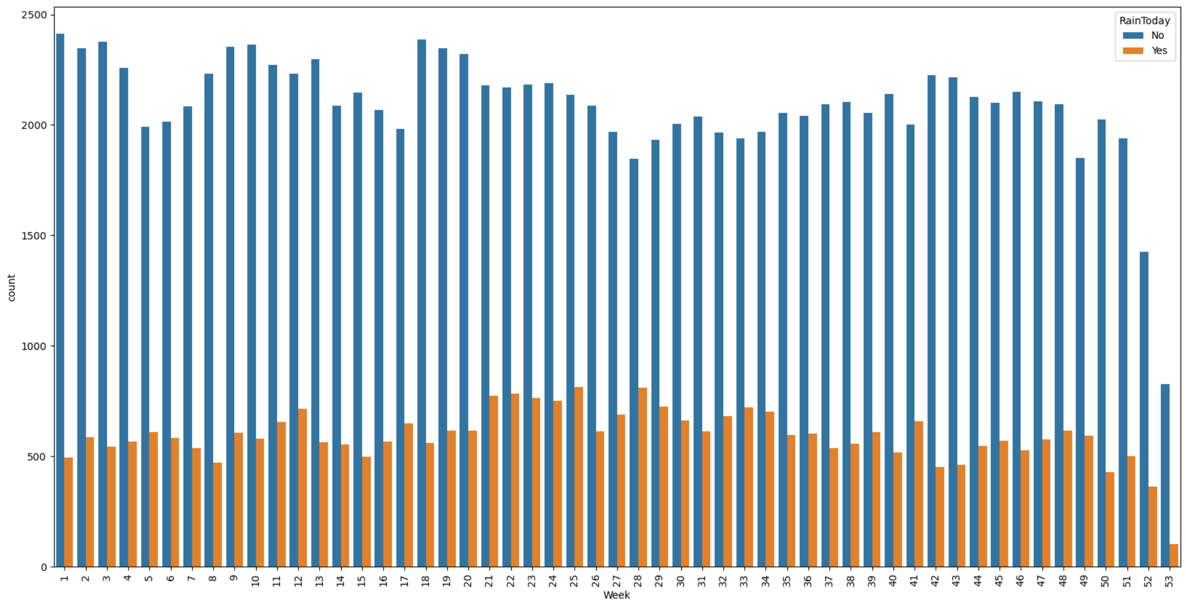
4. Analysis :

4.1 Month vs Rain Today :



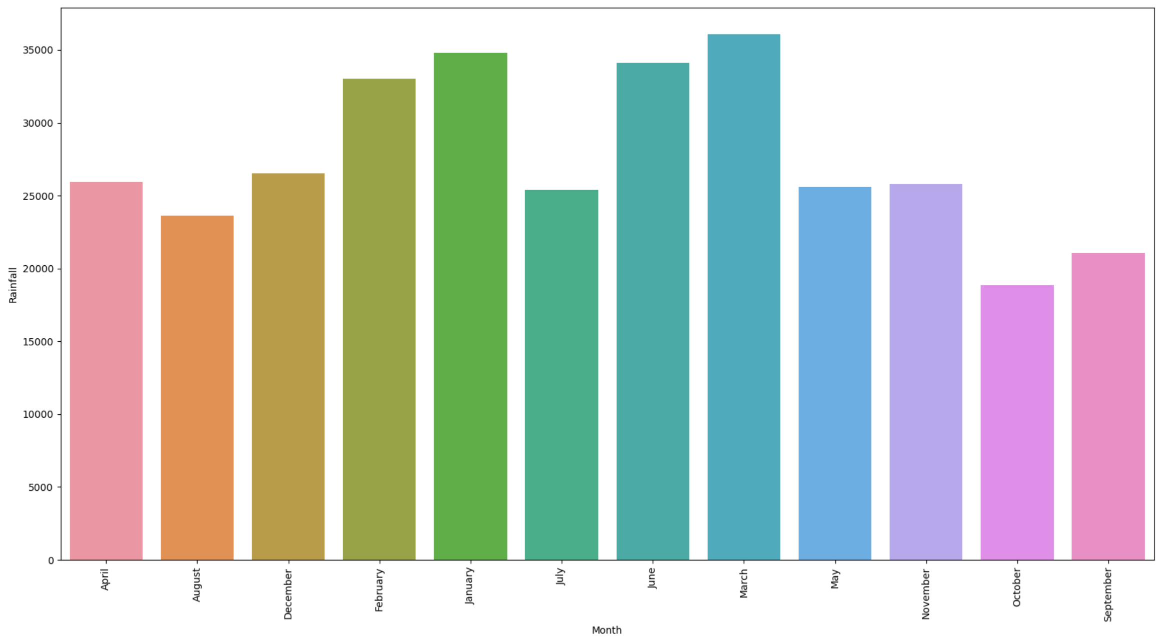
- The bar chart highlights a significant number of days with rainfall during June, July, and August.
- In Australia, the winter season corresponds to the months of June, July, and August. This observation aligns with our dataset.

4.2 Week vs Rain today:



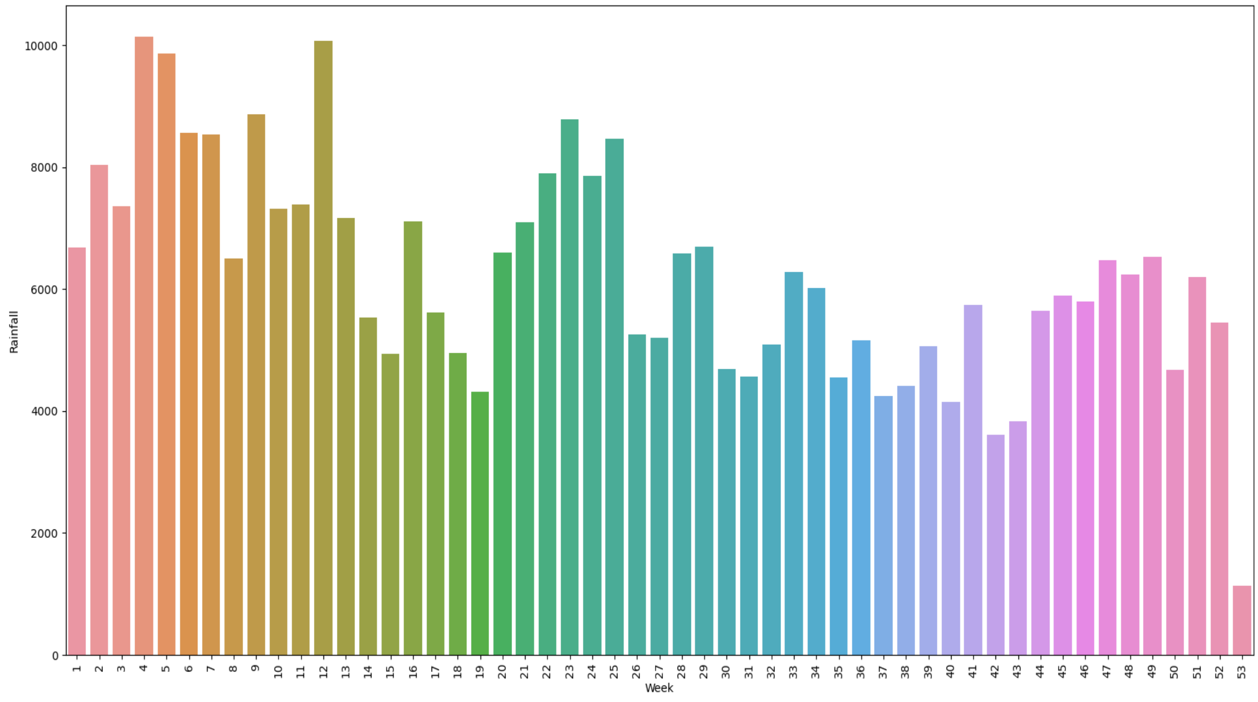
- In the above mentioned bar chart, we observe a significant number of days with rainfall falling within the week range of 21 - 34, corresponding to the months of June, July, and August.
- Notably, weeks 26 and 31 stand out with a lower count of rainy days compared to other weeks experiencing rainfall.

4.3 **Month vs Rainfall :**



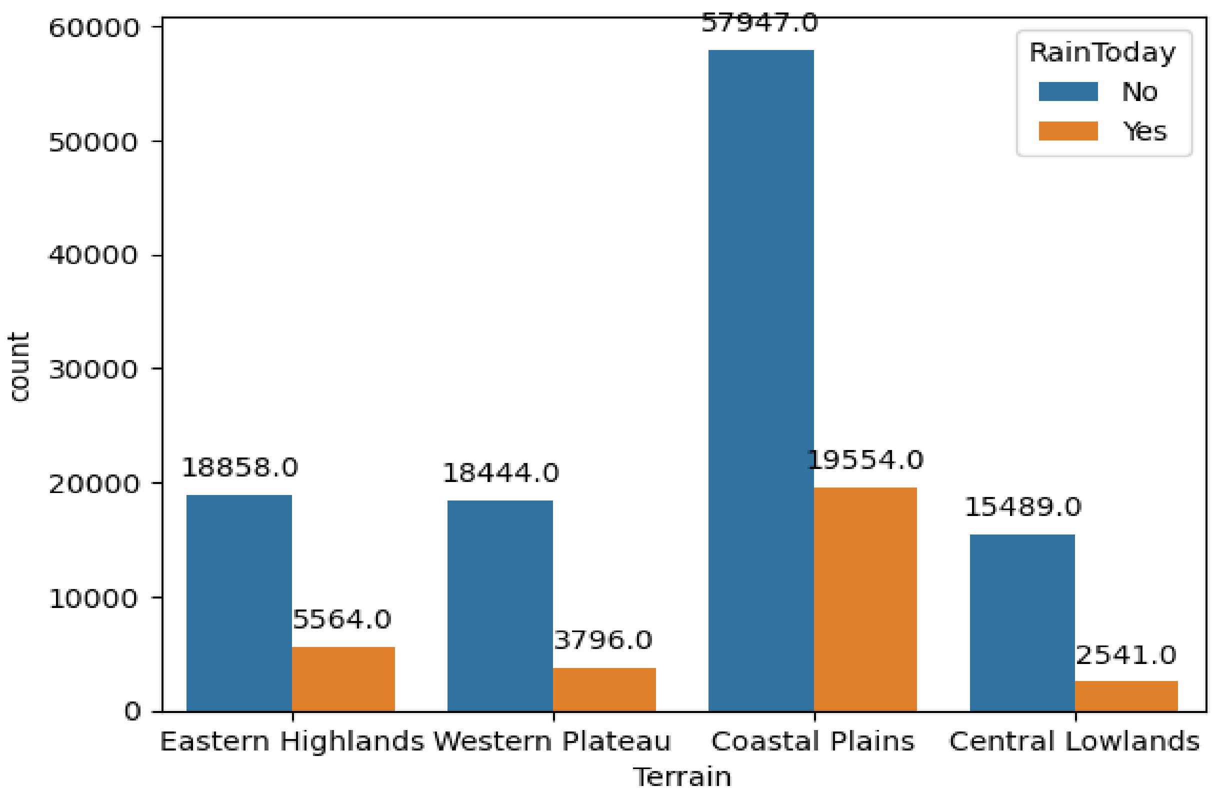
- Over the period from 2008 to 2017, Australia experienced rainfall exceeding 30000 mm primarily in January, February, June, and March.
- Interestingly, Australia receives higher rainfall during summer months compared to other seasons.
- In the northern regions of Australia, the majority of rainfall is linked to the active phases of the monsoon, prevalent during the northern wet season (October to April). Conversely, in southern Australia, the principal period of rainfall is the southern wet season (April to November), usually connected with frontal systems. Nonetheless, recent reports indicate that Australia has experienced substantial flooding, particularly from January to April.

4.4 <u>Week vs Rainfall:</u>



- Weeks 4, 5, and 12 consistently recorded elevated levels of rainfall throughout the ten-year span from 2008 to 2017.

4.5 Terrain vs Rain Today :



- The Coastal Plains, encompassing the majority of city areas, exhibit a 33% likelihood of experiencing rainfall on any given day. This ratio is derived from observed data, indicating a rainfall yes/no ratio of 19554/57947, approximately equivalent to 1 in 3.
- The Western Plateau has a 20% chance of encountering rainfall per day. This probability is determined from observed data, reflecting a rainfall yes/no ratio of 3796/18444, roughly translating to 1 in 5.
- The Eastern Highlands showcase a 30% chance of encountering rainfall for a day, as derived from observed data indicating a

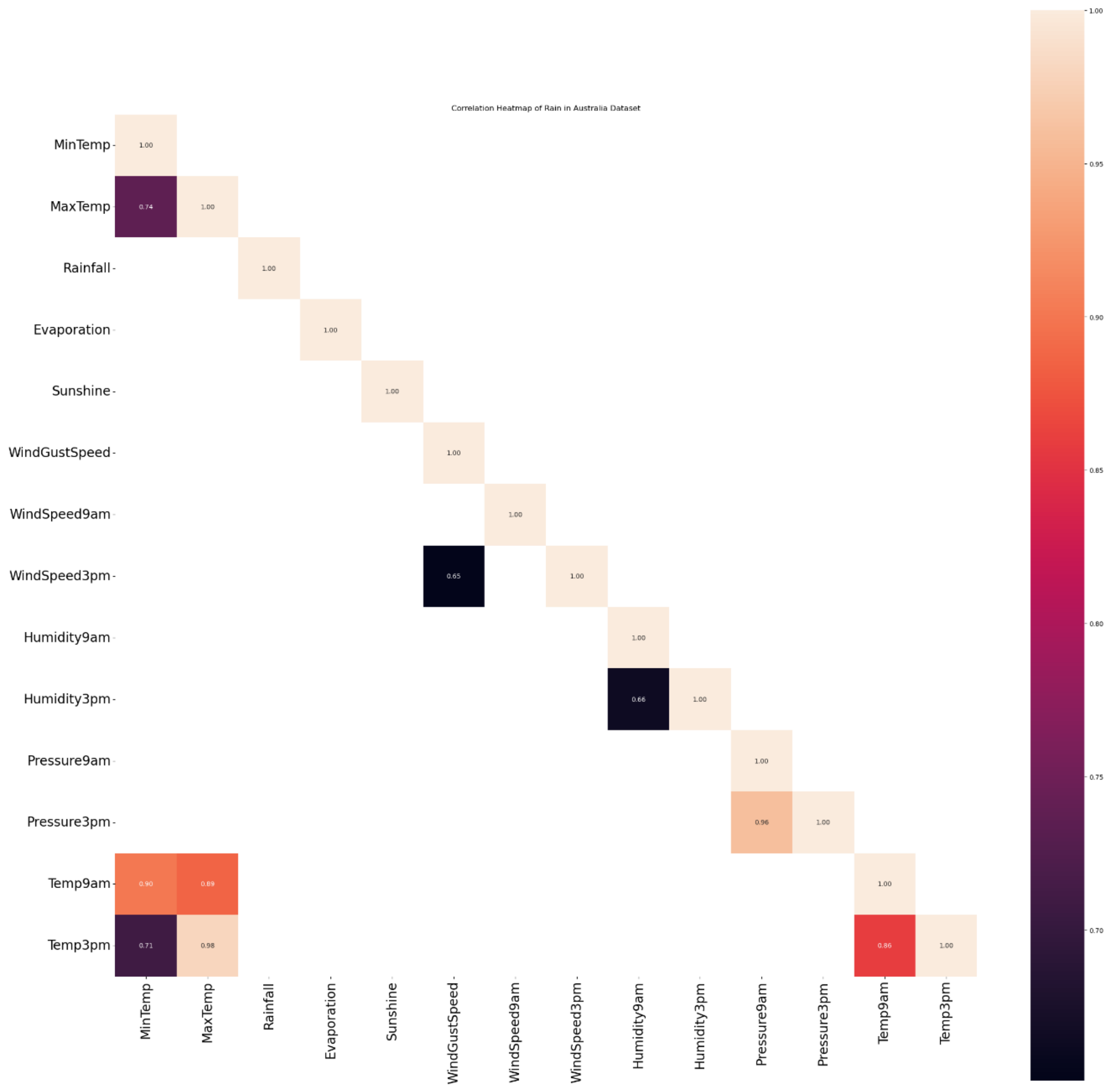
rainfall yes/no ratio of 5564/18858, approximately equivalent to 1 in 3.

- The Central Lowlands cover a smaller portion of city areas and display a 16% probability of experiencing rainfall on any given day. This probability is calculated from observed data, indicating a rainfall yes/no ratio of 2541/15489, roughly translating to 1 in 6.

❖ Within the terrains, the Central Lowlands display the least probability of receiving rainfall. These lowlands typically encounter moderate to low rainfall, characterized by drier conditions in comparison to coastal or mountainous regions.

❖ Conversely, the Coastal Plains distinguish themselves as the terrain most likely to experience rainfall. Due to their proximity to oceans, coastal plains frequently receive higher rainfall, exhibiting more consistent precipitation throughout the year. There's also a possibility of heightened rainfall during specific seasons.

4.6 High correlation between independent variables:



Here, the filters in the correlation matrix is to show only values that are less than or equal to -0.6 or greater than or equal to 0.6. This is done to highlight correlations that are considered strong between independent variables.

5. Feature engineering :

Feature engineering is a meticulous process involving the strategic selection, manipulation, and transformation of raw data to generate features that provide meaningful insights in supervised learning scenarios. This multifaceted procedure aims to introduce new variables that were not initially present in the training data, contributing to a more nuanced understanding of the underlying patterns. The primary goal of feature engineering is to enhance the efficiency and accuracy of machine learning models by refining and expediting the data transformation process.

In the context of feature engineering, encoding emerges as a pivotal step, particularly when dealing with categorical data. Categorical variables, embodying qualitative characteristics, necessitate conversion into a numerical format for seamless integration with machine learning algorithms. This conversion facilitates model interpretation and ensures compatibility with the computational requirements of advanced algorithms.

- A practical illustration of feature engineering and encoding is evident in the analysis of the dataset. Initially, the variables 'cloud9am' and 'cloud3pm' were numeric. However, to improve interpretability, encoding was implemented to categorize them into distinct sky conditions—clear sky, partly cloudy, mostly cloudy, and overcast. This not only simplified the representation of these variables but also made the data more amenable to analysis.

```
df['Cloud9am'].value_counts()
```

```
Mostly_cloudy    86988
Clear_sky        24145
Partly_cloudy    16669
Overcast         14391
Name: Cloud9am, dtype: int64
```

```
a= {0:'Clear_sky',
    1:'Clear_sky',
    2:'Partly_cloudy',
    3:'Partly_cloudy',
    4:'Partly_cloudy',
    5:'Mostly_cloudy',
    6:'Mostly_cloudy',
    7:'Mostly_cloudy',
    8:'Overcast',
    9:'Overcast'}
```

```
df['Cloud9am'] = df['Cloud9am'].map(a)
```

```
df['Cloud3pm'] = df['Cloud3pm'].map(a)
```

```
dff['Cloud3pm'].replace({'Clear_sky':0,'Partly_cloudy':1,'Mostly_cloudy':2,'Overcast':3},inplace=True)
```

```
dff['Cloud9am'].replace({'Clear_sky':0,'Partly_cloudy':1,'Mostly_cloudy':2,'Overcast':3},inplace=True)
```

- Furthermore, within the dataset, the 'location' column encompasses 49 cities and towns. To enhance analytical insights, a strategic grouping based on terrain types was executed. This grouping facilitates a more comprehensive exploration of regional patterns and characteristics.

```
# Feature Engineering

Terrain = {'Albury' : 'Eastern Highlands','BadgerysCreek': 'Western Plateau','Cobar': 'Western Plateau',
           'CoffsHarbour': 'Coastal Plains','Moree': 'Central Lowlands','Newcastle': 'Coastal Plains',
           'NorahHead': 'Coastal Plains','NorfolkIsland': 'Coastal Plains','Penrith': 'Western Plateau',
           'Richmond': 'Western Plateau','Sydney': 'Coastal Plains','SydneyAirport': 'Coastal Plains',
           'WaggaWagga': 'Central Lowlands','Williamtown': 'Coastal Plains','Wollongong': 'Coastal Plains',
           'Canberra': 'Eastern Highlands','Tuggeranong': 'Eastern Highlands','MountGinini': 'Eastern Highlands',
           'Ballarat':'Eastern Highlands','Bendigo':'Central Lowlands','Sale':'Coastal Plains',
           'MelbourneAirport':'Coastal Plains','Melbourne':'Coastal Plains','Mildura':'Central Lowlands',
           'Nhil':'Central Lowlands','Portland':'Coastal Plains','Watsonia':'Coastal Plains',
           'Dartmoor':'Western Plateau','Brisbane':'Coastal Plains','Cairns':'Coastal Plains','GoldCoast':'Coastal Plains',
           'Townsville':'Coastal Plains','Adelaide':'Coastal Plains','MountGambier':'Eastern Highlands'
           ,'Nuriootpa':'Eastern Highlands','Woomera':'Western Plateau','Albany':'Coastal Plains',
           'Witchcliffe':'Coastal Plains','PearceRAAF':'Coastal Plains','PerthAirport':'Coastal Plains',
            'Perth':'Coastal Plains','SalmonGums':'Western Plateau','Walpole':'Coastal Plains',
           'Hobart':'Coastal Plains','Launceston':'Eastern Highlands','AliceSprings':'Central Lowlands',
           'Darwin':'Coastal Plains','Katherine':'Central Lowlands','Uluru':'Western Plateau'}

dff['Terrain'] = dff['Location'].replace(Terrain)
```

- The binary attributes 'RainToday' and 'RainTomorrow,' signifying 'yes' and 'no,' underwent transformation through encoding. This transformation involved assigning the numeric values 0 for 'no' and 1 for 'yes,' providing a standardized representation that facilitates consistent interpretation and analysis.

```
dff['RainToday'].replace({'No':0,'Yes':1},inplace=True)
dff['RainTomorrow'].replace({'No':0,'Yes':1},inplace=True)
```

- Additionally, categorical variables such as 'WindGustDir,' 'WindDir9am,' and 'WindDir3pm,' each initially featuring 16 unique values, underwent a simplification process. Through the application of dummy encoding, these variables were streamlined into four major directions: north, east, west, and south. This simplification not only enhances interpretability but also improves the efficiency of subsequent analyses by reducing the dimensionality of the dataset.

```
dff['WindGustDir'].replace({'SE': 'GustS', 'SSE': 'GustS', 'SW': 'GustS', 'SSW': 'GustS',
                            'NW': 'GustN', 'NNE': 'GustN', 'NNW': 'GustN', 'NE': 'GustN',
                            'ENE': 'GustE', 'ESE': 'GustE',
                            'WNW': 'GustW', 'WSW': 'GustW',
                            'S': 'GustS','N': 'GustN','E': 'GustE','W': 'GustW'}, inplace=True)
```

```
dff['WindDir9am'].replace({'SE': 'WindDir9am_S', 'SSE': 'WindDir9am_S', 'SW': 'WindDir9am_S', 'SSW': 'WindDir9am_S',
                            'NW': 'WindDir9am_N', 'NNE': 'WindDir9am_N', 'NNW': 'WindDir9am_N', 'NE': 'WindDir9am_N',
                            'ENE': 'WindDir9am_E', 'ESE': 'WindDir9am_E',
                            'WNW': 'WindDir9am_W', 'WSW': 'WindDir9am_W',
                           'W': 'WindDir9am_W','E': 'WindDir9am_E','N': 'WindDir9am_N',
                           'S': 'WindDir9am_S'}, inplace=True)
```

```
dff['WindDir3pm'].replace({'SE': 'WindDir3pm_S', 'SSE': 'WindDir3pm_S', 'SW': 'WindDir3pm_S', 'SSW': 'WindDir3pm_S',
                            'NW': 'WindDir3pm_N', 'NNE': 'WindDir3pm_N', 'NNW': 'WindDir3pm_N', 'NE': 'WindDir3pm_N',
                            'ENE': 'WindDir3pm_E', 'ESE': 'WindDir3pm_E',
                            'WNW': 'WindDir3pm_W', 'WSW': 'WindDir3pm_W',
                           'W': 'WindDir3pm_W','E': 'WindDir3pm_E','N': 'WindDir3pm_N',
                           'S': 'WindDir3pm_S'}, inplace=True)
```

- The date column was disaggregated into separate columns for day, month, and year to facilitate a more detailed and insightful analysis.

```
df['Date']=pd.to_datetime(df['Date'])
```

```
df['iso_week_number'] = df['Date'].dt.isocalendar().week
```

```
a=[]
for i in df['Date']:
    a.append(i.strftime("%U"))
```

```
a=pd.DataFrame(a,columns=['Week'])
```

```
df = pd.concat([df,a],1)
```

```
df['Week'] = df['Week'].astype(int)
```

```
df['Month'] = df['Date'].apply(lambda x: pd.to_datetime(x).strftime('%B'))
```

```
df['Year']=df['Date'].dt.year
```

In summary, the intricate interplay between feature engineering and encoding is demonstrated in the meticulous crafting of variables to better align with the objectives of supervised learning. This not only refines the dataset but also empowers machine learning models to discern patterns and relationships with increased accuracy and efficiency.

6. FUTURE ENHANCEMENTS FOR CLASSIFICATION MODELS:

- ***Feature Engineering:*** Identify and generate features that better capture the underlying patterns in the data. This may involve creating interaction terms, polynomial features, or transforming existing variables to better represent the relationships.
- ***Cross-Validation:*** Cross-validation is a valuable tool for fine-tuning a classification model's hyperparameters and estimating its generalization performance. Techniques such as K-fold cross-validation allow the model to be evaluated on different subsets of data. Grid search or randomized search can then be employed to identify the optimal hyperparameter values, contributing to better model performance.
- ***Ensemble Methods:*** Ensemble methods, including bagging, boosting, and stacking, can enhance the performance of classification models.
    1. **Bagging**: Involves training multiple models on different subsets of the data and averaging their predictions, providing a more robust and stable model.
    2. **Boosting**: Focuses on combining weak learners to create a strong classifier, iteratively giving more weight to misclassified instances.
    3. **Stacking**: Encompasses training multiple models and combining their predictions using a meta-model, leveraging the strengths of individual models.

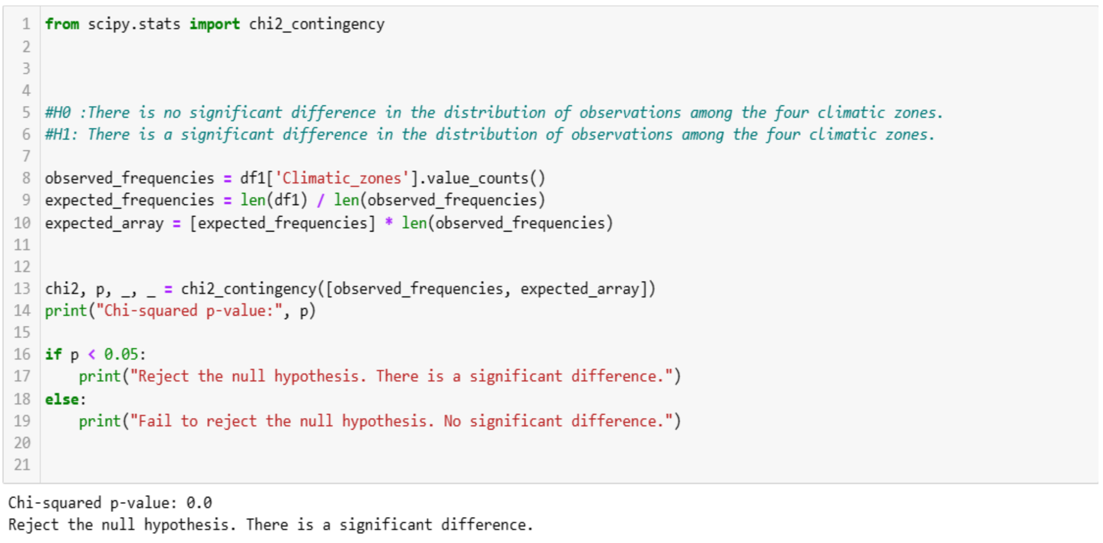
- ***Handling Imbalanced Data:*** Address class imbalance by employing techniques such as oversampling the minority class, under sampling the majority class, or using advanced methods like SMOTE (Synthetic Minority Over-sampling Technique).
- ***Advanced Algorithms:*** Using more sophisticated algorithms or ensemble models beyond traditional ones. Techniques like support vector machines, gradient boosting, or neural networks may offer improved performance depending on the nature of the data.

7. Hypothesis Testing :

Hypothesis testing is employed in ML to draw statistically significant conclusions regarding model performance, validate assumptions, and ensure reliable generalization to new data. In our dataset, we applied hypothesis testing, specifically utilizing chi-square and ANOVA, to assess the significance of variations in the categorical variable "climatic zones."

7.1 Chi – square :

Null Hypothesis: There is no significant difference in the distribution of observations among the four climatic zones.
Alternate Hypothesis: There is a significant difference in the distribution of observations among the four climatic zones.



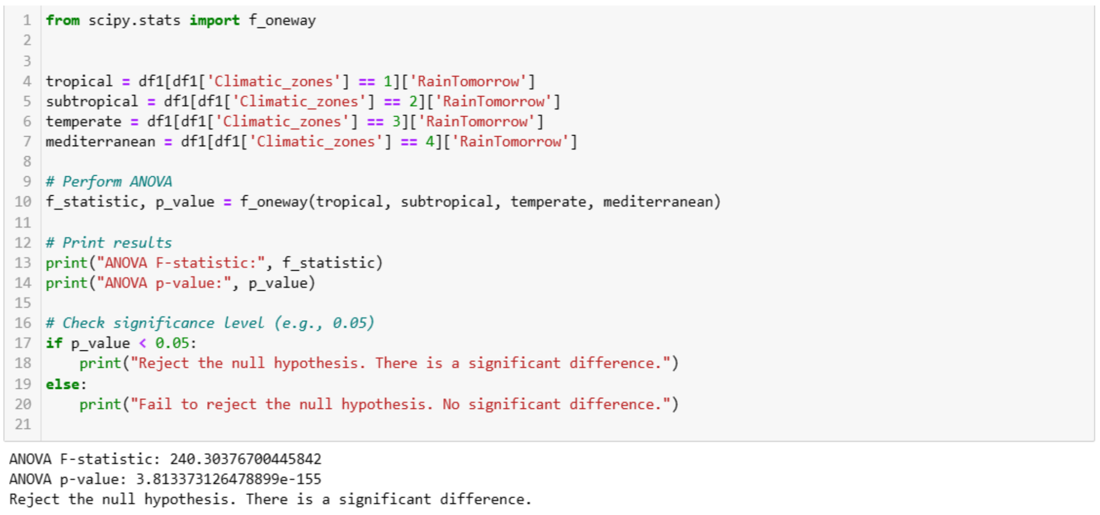
```python
from scipy.stats import chi2_contingency



#H0 :There is no significant difference in the distribution of observations among the four climatic zones.
#H1: There is a significant difference in the distribution of observations among the four climatic zones.

observed_frequencies = df1['Climatic_zones'].value_counts()
expected_frequencies = len(df1) / len(observed_frequencies)
expected_array = [expected_frequencies] * len(observed_frequencies)


chi2, p, _, _ = chi2_contingency([observed_frequencies, expected_array])
print("Chi-squared p-value:", p)

if p < 0.05:
    print("Reject the null hypothesis. There is a significant difference.")
else:
    print("Fail to reject the null hypothesis. No significant difference.")

```

```
Chi-squared p-value: 0.0
Reject the null hypothesis. There is a significant difference.
```

7.2 <u>Anova:</u>

Null hypothesis: There is no significant difference in the mean of "RainTomorrow" across different climatic zones.

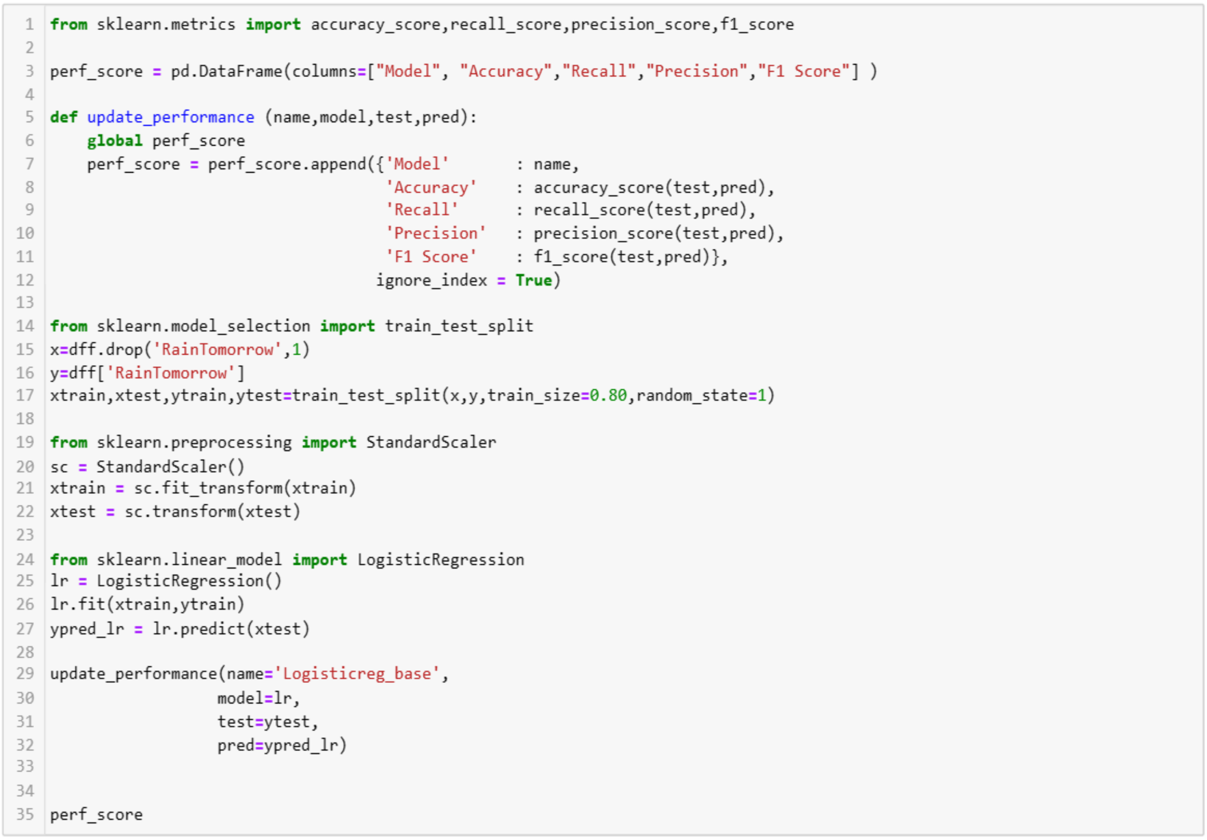
Alternate hypothesis: There is a significant difference in the mean of "RainTomorrow" across different climatic zones.



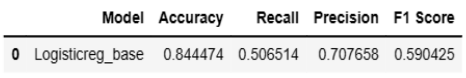
```python
from scipy.stats import f_oneway


tropical = df1[df1['Climatic_zones'] == 1]['RainTomorrow']
subtropical = df1[df1['Climatic_zones'] == 2]['RainTomorrow']
temperate = df1[df1['Climatic_zones'] == 3]['RainTomorrow']
mediterranean = df1[df1['Climatic_zones'] == 4]['RainTomorrow']

# Perform ANOVA
f_statistic, p_value = f_oneway(tropical, subtropical, temperate, mediterranean)

# Print results
print("ANOVA F-statistic:", f_statistic)
print("ANOVA p-value:", p_value)

# Check significance level (e.g., 0.05)
if p_value < 0.05:
    print("Reject the null hypothesis. There is a significant difference.")
else:
    print("Fail to reject the null hypothesis. No significant difference.")

```

```
ANOVA F-statistic: 240.30376700445842
ANOVA p-value: 3.813373126478899e-155
Reject the null hypothesis. There is a significant difference.
```

8. Base Model:



```python
from sklearn.metrics import accuracy_score,recall_score,precision_score,f1_score

perf_score = pd.DataFrame(columns=["Model", "Accuracy","Recall","Precision","F1 Score"] )

def update_performance (name,model,test,pred):
    global perf_score
    perf_score = perf_score.append({'Model'       : name,
                                    'Accuracy'    : accuracy_score(test,pred),
                                    'Recall'      : recall_score(test,pred),
                                    'Precision'   : precision_score(test,pred),
                                    'F1 Score'    : f1_score(test,pred)},
                                   ignore_index = True)

from sklearn.model_selection import train_test_split
x=dff.drop('RainTomorrow',1)
y=dff['RainTomorrow']
xtrain,xtest,ytrain,ytest=train_test_split(x,y,train_size=0.80,random_state=1)

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
xtrain = sc.fit_transform(xtrain)
xtest = sc.transform(xtest)

from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(xtrain,ytrain)
ypred_lr = lr.predict(xtest)

update_performance(name='Logisticreg_base',
                  model=lr,
                  test=ytest,
                  pred=ypred_lr)


perf_score
```



	Model	Accuracy	Recall	Precision	F1 Score
0	Logisticreg_base	0.844474	0.506514	0.707658	0.590425

Interpretation of the Base Model:

- Accuracy: The model achieves an accuracy of approximately 84.45%. This metric represents the proportion of correctly classified instances out of the total instances in the dataset. While accuracy provides an overall assessment of the model's performance, it might not be sufficient in cases of imbalanced datasets.
- Recall (Sensitivity): The recall, also known as sensitivity or true positive rate, is calculated at approximately 50.65%. This metric measures the ability of the model to correctly identify positive

instances (relevant cases). A higher recall is desirable when the cost of missing positive instances is high.

- Precision: The precision of the model is approximately 70.77%. Precision represents the proportion of correctly identified positive instances out of all instances predicted as positive. It indicates the model's ability to avoid false positives. A higher precision is favourable when the cost of false positives is significant.
- F1 Score: The F1 Score, computed as approximately 59.04%, is the harmonic mean of precision and recall. It provides a balanced measure of a model's performance, especially in scenarios where there is an imbalance between positive and negative classes. A higher F1 Score indicates a better balance between precision and recall.

<u>Overall Assessment:</u>

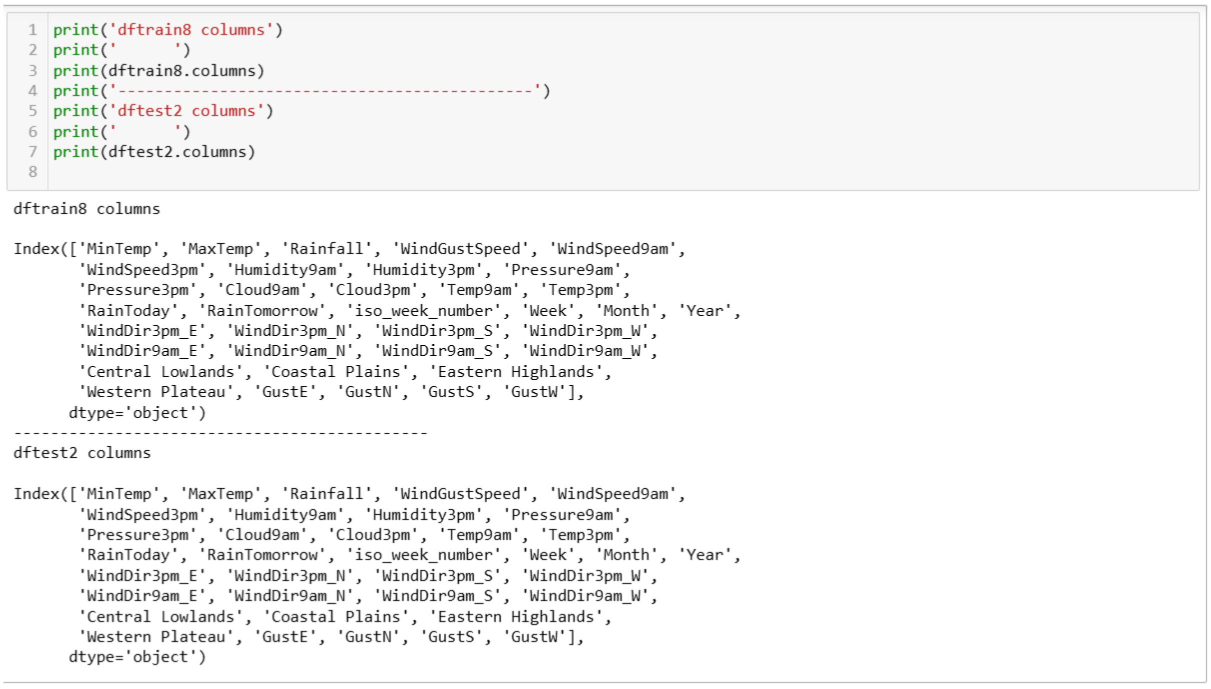
The Logistic Regression model, in its baseline configuration, demonstrates a reasonably good level of accuracy. However, the trade-off between precision and recall should be carefully considered based on the specific objectives of the classification problem. The F1 Score provides a comprehensive measure that balances these considerations.

9. Model building with other Algorithms

In the preceding base model, we adhered to the conventional 80:20 train-test data split. However, for our forthcoming models, we intend to explore two alternative approaches:

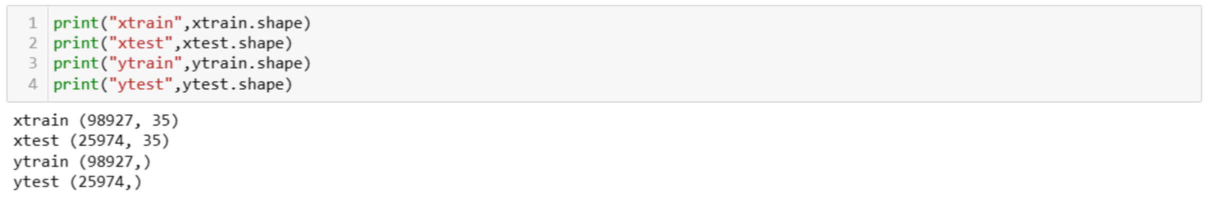
- Implementing a split where 8 years are designated for training data and 2 years for testing data.
- Dividing the 10-year dataset into 9 years for training data and 1 year for testing data.

9.1 8-Year Training, 2-Year Testing Models :



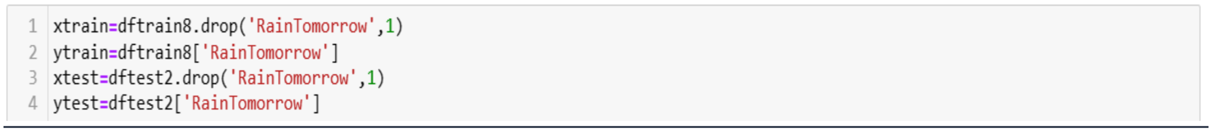
```
print('dftrain8 columns')
print('      ')
print(dftrain8.columns)
print('---------------------------------------------')
print('dftest2 columns')
print('      ')
print(dftest2.columns)

```

```
dftrain8 columns

Index(['MinTemp', 'MaxTemp', 'Rainfall', 'WindGustSpeed', 'WindSpeed9am',
       'WindSpeed3pm', 'Humidity9am', 'Humidity3pm', 'Pressure9am',
       'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am', 'Temp3pm',
       'RainToday', 'RainTomorrow', 'iso_week_number', 'Week', 'Month', 'Year',
       'WindDir3pm_E', 'WindDir3pm_N', 'WindDir3pm_S', 'WindDir3pm_W',
       'WindDir9am_E', 'WindDir9am_N', 'WindDir9am_S', 'WindDir9am_W',
       'Central Lowlands', 'Coastal Plains', 'Eastern Highlands',
       'Western Plateau', 'GustE', 'GustN', 'GustS', 'GustW'],
      dtype='object')
---------------------------------------------
dftest2 columns

Index(['MinTemp', 'MaxTemp', 'Rainfall', 'WindGustSpeed', 'WindSpeed9am',
       'WindSpeed3pm', 'Humidity9am', 'Humidity3pm', 'Pressure9am',
       'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am', 'Temp3pm',
       'RainToday', 'RainTomorrow', 'iso_week_number', 'Week', 'Month', 'Year',
       'WindDir3pm_E', 'WindDir3pm_N', 'WindDir3pm_S', 'WindDir3pm_W',
       'WindDir9am_E', 'WindDir9am_N', 'WindDir9am_S', 'WindDir9am_W',
       'Central Lowlands', 'Coastal Plains', 'Eastern Highlands',
       'Western Plateau', 'GustE', 'GustN', 'GustS', 'GustW'],
      dtype='object')
```



```
print("xtrain",xtrain.shape)
print("xtest",xtest.shape)
print("ytrain",ytrain.shape)
print("ytest",ytest.shape)
```

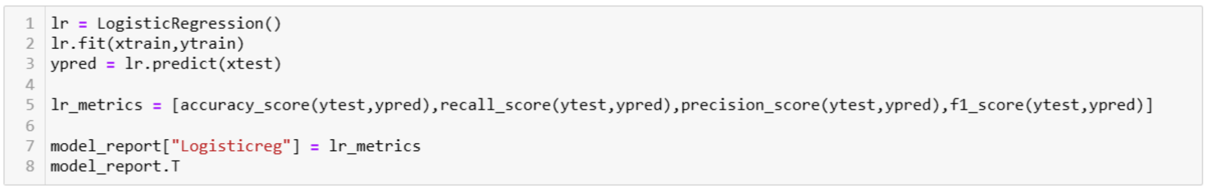
```
xtrain (98927, 35)
xtest (25974, 35)
ytrain (98927,)
ytest (25974,)
```



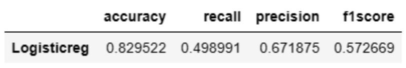
```python
xtrain=dftrain8.drop('RainTomorrow',1)
ytrain=dftrain8['RainTomorrow']
xtest=dftest2.drop('RainTomorrow',1)
ytest=dftest2['RainTomorrow']
```

9.1.1 Logistic Regression:

Logistic Regression is a statistical method used for binary classification tasks, such as predicting rain tomorrow; its advantages include simplicity, interpretability, and efficiency for linearly separable data, but it may struggle with complex relationships and multicollinearity.



```python
lr = LogisticRegression()
lr.fit(xtrain,ytrain)
ypred = lr.predict(xtest)

lr_metrics = [accuracy_score(ytest,ypred),recall_score(ytest,ypred),precision_score(ytest,ypred),f1_score(ytest,ypred)]

model_report["Logisticreg"] = lr_metrics
model_report.T
```



	accuracy	recall	precision	f1score
Logisticreg	0.829522	0.498991	0.671875	0.572669

Accuracy: Overall correctness of predictions (82.95%).

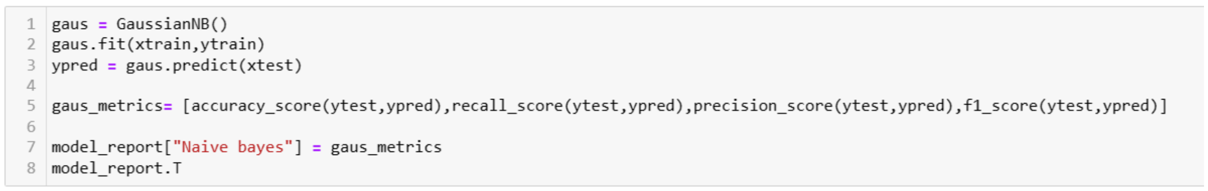
Recall: Proportion of actual positives correctly identified (49.90%).

Precision: Proportion of predicted positives correctly identified (67.19%).

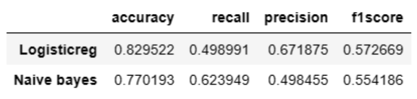
F1 Score: Harmonic mean of precision and recall (57.27%) in Logistic Regression predictions.

9.1.2 Gaussian Naïve Bayes :

Gaussian Naive Bayes, a probabilistic classifier, assumes features are independent and follows a normal distribution. Its simplicity facilitates quick training, but the independence assumption may limit accuracy. Despite this, it is employed for predicting rain tomorrow due to its efficiency in handling multiple features and its ability to provide probabilistic predictions.



```python
gaus = GaussianNB()
gaus.fit(xtrain,ytrain)
ypred = gaus.predict(xtest)

gaus_metrics= [accuracy_score(ytest,ypred),recall_score(ytest,ypred),precision_score(ytest,ypred),f1_score(ytest,ypred)]

model_report["Naive bayes"] = gaus_metrics
model_report.T
```



	accuracy	recall	precision	f1score
Logisticreg	0.829522	0.498991	0.671875	0.572669
Naive bayes	0.770193	0.623949	0.498455	0.554186

Accuracy: The Naive Bayes model correctly predicted 77.0% of instances overall.

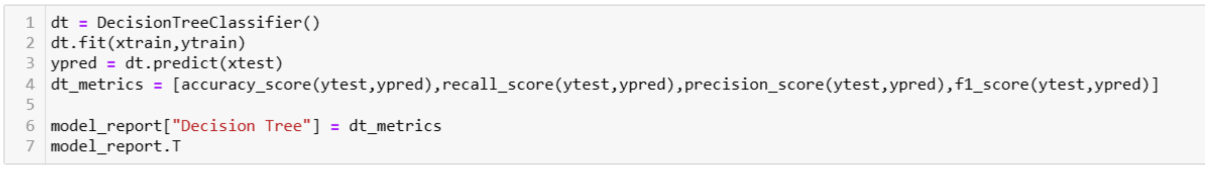
Recall: 62.4% of actual positive instances were successfully identified by the model.

Precision: The model's positive predictions were accurate 49.8% of the time.

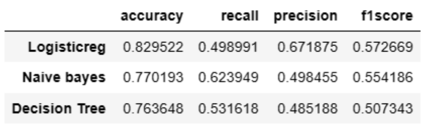
F1 Score: The balance between precision and recall yielded an F1 score of 55.4%, summarizing overall performance.

9.1.3 Decision Tree classifier model :

It is intuitive and easy to interpret, it handles various data types, but it's prone to overfitting and sensitive to small data variations; predicting rain tomorrow by recursively partitioning features, providing insights based on historical patterns and current conditions.



```
dt = DecisionTreeClassifier()
dt.fit(xtrain,ytrain)
ypred = dt.predict(xtest)
dt_metrics = [accuracy_score(ytest,ypred),recall_score(ytest,ypred),precision_score(ytest,ypred),f1_score(ytest,ypred)]

model_report["Decision Tree"] = dt_metrics
model_report.T
```



	accuracy	recall	precision	f1score
Logisticreg	0.829522	0.498991	0.671875	0.572669
Naive bayes	0.770193	0.623949	0.498455	0.554186
Decision Tree	0.763648	0.531618	0.485188	0.507343

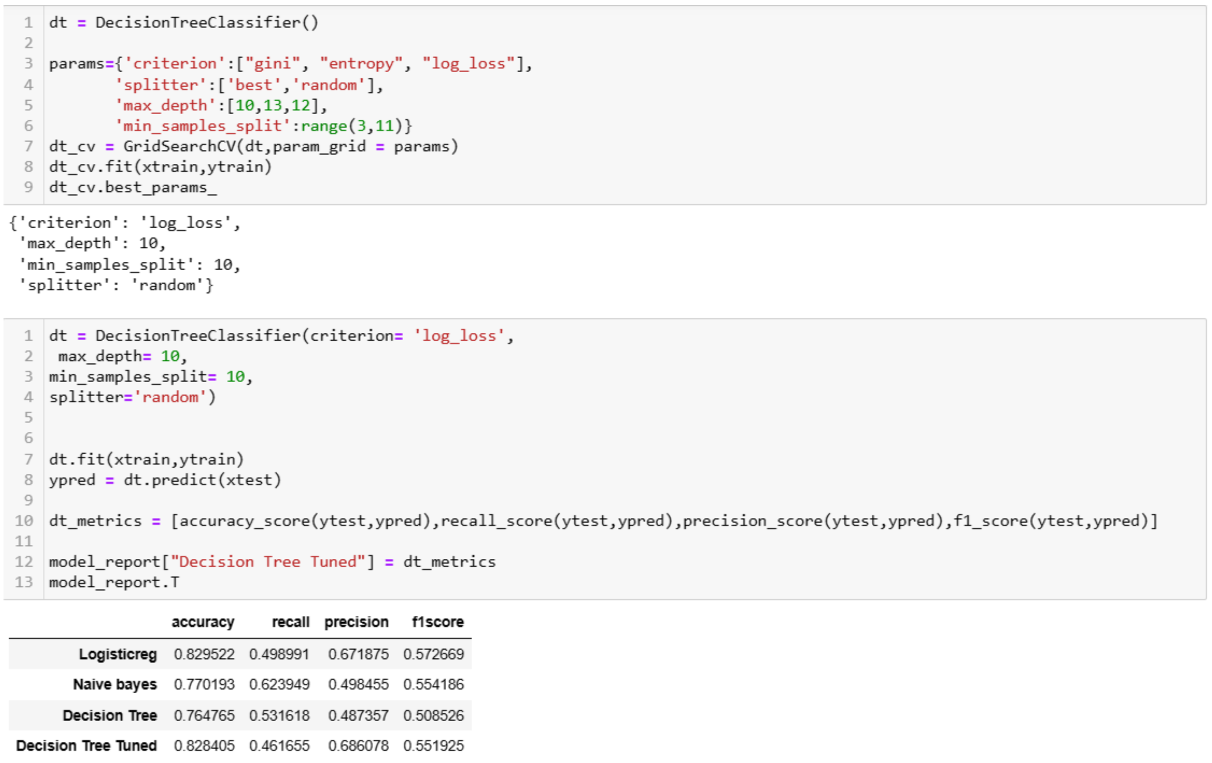
Accuracy: The Decision Tree model achieved an overall correctness of 76.4% in its predictions.

Recall: Successfully identified 53.2% of actual positive instances, showcasing its ability to capture relevant cases.

Precision: With a precision of 48.5%, the model accurately classified positive predictions in nearly half of the cases.

F1 Score: Striking a balance between precision and recall, the F1 score stands at 50.7%, summarizing the model's overall performance.

9.1.4 Decision tree classifier hyper parameter tuned model:



```python
dt = DecisionTreeClassifier()

params={'criterion':["gini", "entropy", "log_loss"],
       'splitter':['best','random'],
       'max_depth':[10,13,12],
       'min_samples_split':range(3,11)}
dt_cv = GridSearchCV(dt,param_grid = params)
dt_cv.fit(xtrain,ytrain)
dt_cv.best_params_
```

```
{'criterion': 'log_loss',
 'max_depth': 10,
 'min_samples_split': 10,
 'splitter': 'random'}
```

```python
dt = DecisionTreeClassifier(criterion= 'log_loss',
 max_depth= 10,
min_samples_split= 10,
splitter='random')


dt.fit(xtrain,ytrain)
ypred = dt.predict(xtest)

dt_metrics = [accuracy_score(ytest,ypred),recall_score(ytest,ypred),precision_score(ytest,ypred),f1_score(ytest,ypred)]

model_report["Decision Tree Tuned"] = dt_metrics
model_report.T
```

	accuracy	recall	precision	f1score
Logisticreg	0.829522	0.498991	0.671875	0.572669
Naive bayes	0.770193	0.623949	0.498455	0.554186
Decision Tree	0.764765	0.531618	0.487357	0.508526
Decision Tree Tuned	0.828405	0.461655	0.686078	0.551925

Accuracy: The tuned Decision Tree model demonstrated an improved overall correctness, achieving an accuracy of 82.8% in its predictions.

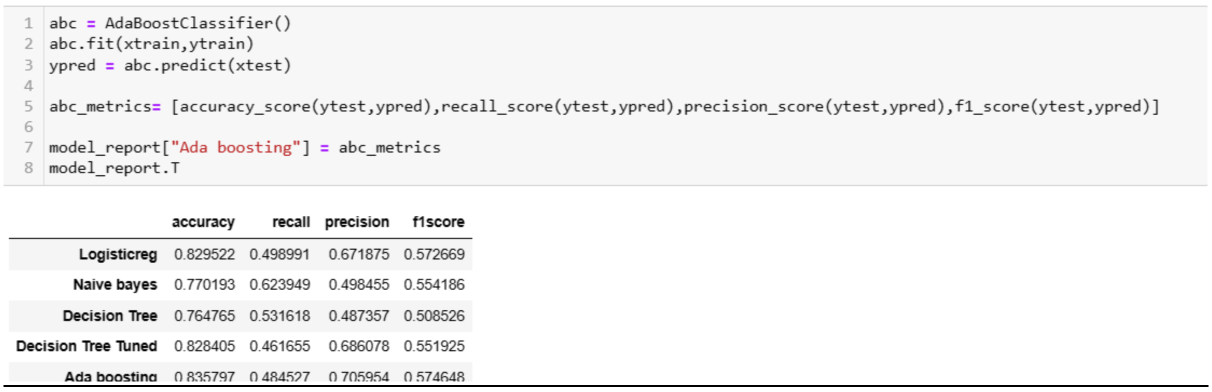
Recall: While the recall decreased to 46.2%, the model still captured a significant portion of actual positive instances.

Precision: Markedly improved precision of 68.6%, indicating higher accuracy in positive predictions.

F1 Score: The tuned model maintains a balanced performance with an F1 score of 55.2%, reflecting a harmonious blend of precision and recall.

9.1.5 Ada Boost Model :

A boosting algorithm that adapts by combining weak learners, its pros include improved accuracy and handling of complex datasets, while cons involve sensitivity to noisy data; predicts rain tomorrow by iteratively emphasizing misclassified instances, enhancing overall model performance.



```python
abc = AdaBoostClassifier()
abc.fit(xtrain,ytrain)
ypred = abc.predict(xtest)

abc_metrics= [accuracy_score(ytest,ypred),recall_score(ytest,ypred),precision_score(ytest,ypred),f1_score(ytest,ypred)]

model_report["Ada boosting"] = abc_metrics
model_report.T
```

	accuracy	recall	precision	f1score
Logisticreg	0.829522	0.498991	0.671875	0.572669
Naive bayes	0.770193	0.623949	0.498455	0.554186
Decision Tree	0.764765	0.531618	0.487357	0.508526
Decision Tree Tuned	0.828405	0.461655	0.686078	0.551925
Ada boosting	0.835797	0.484527	0.705954	0.574648

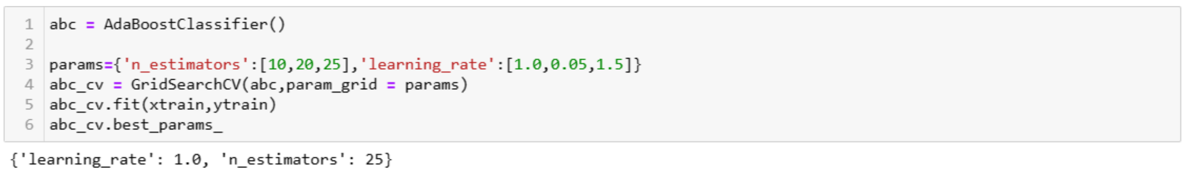
Accuracy: The AdaBoost model achieved an accuracy of 83.6%, reflecting its overall correctness in predictions.

Recall: Successfully identified 48.5% of actual positive instances, demonstrating its ability to capture relevant cases.

Precision: With a precision of 70.6%, the model accurately classified positive predictions in a substantial portion of cases.

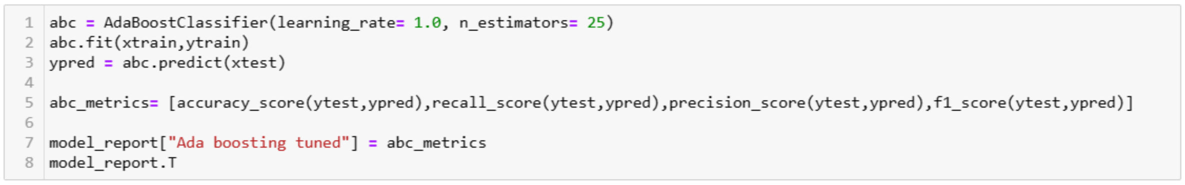
F1 Score: Striking a balance between precision and recall, the F1 score stands at 57.5%, summarizing the model's overall performance in harmonizing precision and recall.

9.1.6 Ada Boost Hyper parameter tuned model:

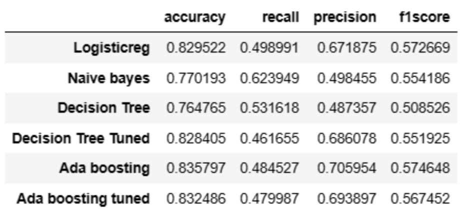


```
abc = AdaBoostClassifier()

params={'n_estimators':[10,20,25],'learning_rate':[1.0,0.05,1.5]}
abc_cv = GridSearchCV(abc,param_grid = params)
abc_cv.fit(xtrain,ytrain)
abc_cv.best_params_
```

```
{'learning_rate': 1.0, 'n_estimators': 25}
```



```
abc = AdaBoostClassifier(learning_rate= 1.0, n_estimators= 25)
abc.fit(xtrain,ytrain)
ypred = abc.predict(xtest)

abc_metrics= [accuracy_score(ytest,ypred),recall_score(ytest,ypred),precision_score(ytest,ypred),f1_score(ytest,ypred)]

model_report["Ada boosting tuned"] = abc_metrics
model_report.T
```



	accuracy	recall	precision	f1score
Logisticreg	0.829522	0.498991	0.671875	0.572669
Naive bayes	0.770193	0.623949	0.498455	0.554186
Decision Tree	0.764765	0.531618	0.487357	0.508526
Decision Tree Tuned	0.828405	0.461655	0.686078	0.551925
Ada boosting	0.835797	0.484527	0.705954	0.574648
Ada boosting tuned	0.832486	0.479987	0.693897	0.567452

Accuracy: The tuned AdaBoost model exhibited an accuracy of 83.2%, indicating its improved overall correctness in predictions.

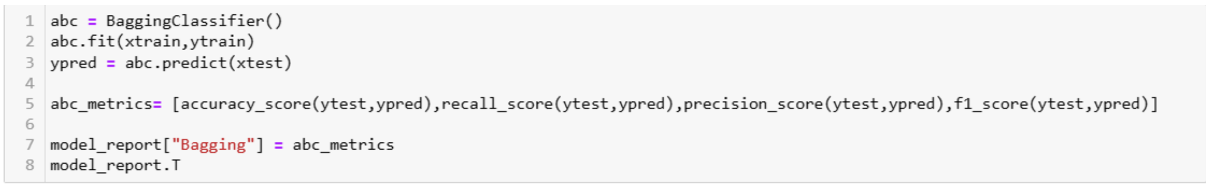
Recall: While the recall slightly decreased to 48.0%, the model continued to capture a significant portion of actual positive instances.

Precision: Maintaining a high precision of 69.4%, the model accurately classified positive predictions in a substantial proportion of cases.

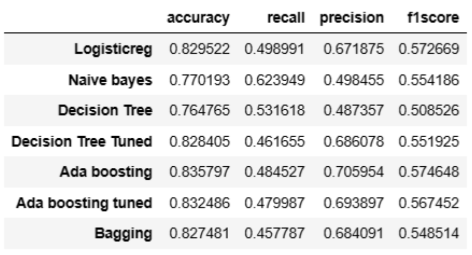
F1 Score: The tuned model demonstrated a balanced performance with an F1 score of 56.7%, emphasizing its ability to harmonize precision and recall effectively.

9.1.7 Bagging model :

Bagging is a parallel ensemble technique that reduces overfitting and enhances stability, its pros include improved generalization, while cons involve potential loss of interpretability; predicts rain tomorrow by aggregating predictions from diverse base models, fostering robustness and accuracy.



```python
abc = BaggingClassifier()
abc.fit(xtrain,ytrain)
ypred = abc.predict(xtest)

abc_metrics= [accuracy_score(ytest,ypred),recall_score(ytest,ypred),precision_score(ytest,ypred),f1_score(ytest,ypred)]

model_report["Bagging"] = abc_metrics
model_report.T
```



	accuracy	recall	precision	f1score
Logisticreg	0.829522	0.498991	0.671875	0.572669
Naive bayes	0.770193	0.623949	0.498455	0.554186
Decision Tree	0.764765	0.531618	0.487357	0.508526
Decision Tree Tuned	0.828405	0.461655	0.686078	0.551925
Ada boosting	0.835797	0.484527	0.705954	0.574648
Ada boosting tuned	0.832486	0.479987	0.693897	0.567452
Bagging	0.827481	0.457787	0.684091	0.548514

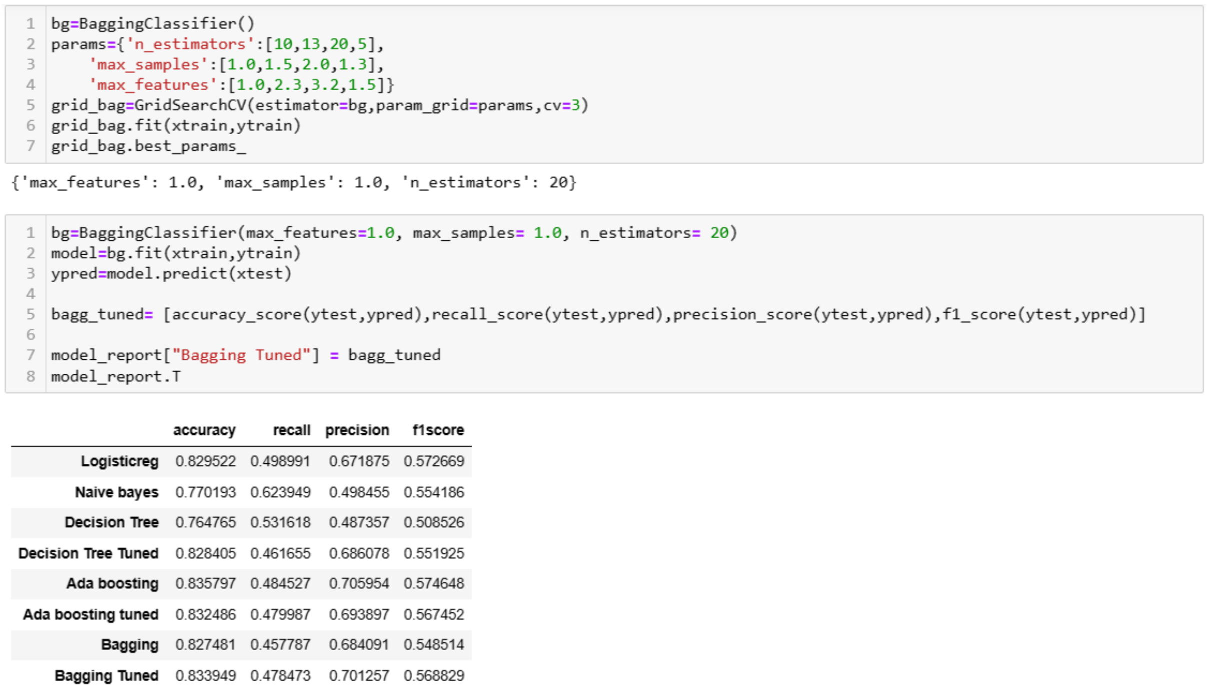
Accuracy: The Bagging ensemble achieved an accuracy of 82.7%, showcasing its overall correctness in predictions.

Recall: Successfully identified 45.8% of actual positive instances, indicating its ability to capture relevant cases.

Precision: With a precision of 68.4%, the Bagging model accurately classified positive predictions in a substantial proportion of cases.

F1 Score: Striking a balance between precision and recall, the F1 score stands at 54.9%, summarizing the ensemble's overall performance in harmonizing precision and recall.

9.1.8 Bagging hyper parameter tuned model :



```
bg=BaggingClassifier()
params={'n_estimators':[10,13,20,5],
     'max_samples':[1.0,1.5,2.0,1.3],
     'max_features':[1.0,2.3,3.2,1.5]}
grid_bag=GridSearchCV(estimator=bg,param_grid=params,cv=3)
grid_bag.fit(xtrain,ytrain)
grid_bag.best_params_
```

```
{'max_features': 1.0, 'max_samples': 1.0, 'n_estimators': 20}
```

```
bg=BaggingClassifier(max_features=1.0, max_samples= 1.0, n_estimators= 20)
model=bg.fit(xtrain,ytrain)
ypred=model.predict(xtest)

bagg_tuned= [accuracy_score(ytest,ypred),recall_score(ytest,ypred),precision_score(ytest,ypred),f1_score(ytest,ypred)]

model_report["Bagging Tuned"] = bagg_tuned
model_report.T
```

	accuracy	recall	precision	f1score
Logisticreg	0.829522	0.498991	0.671875	0.572669
Naive bayes	0.770193	0.623949	0.498455	0.554186
Decision Tree	0.764765	0.531618	0.487357	0.508526
Decision Tree Tuned	0.828405	0.461655	0.686078	0.551925
Ada boosting	0.835797	0.484527	0.705954	0.574648
Ada boosting tuned	0.832486	0.479987	0.693897	0.567452
Bagging	0.827481	0.457787	0.684091	0.548514
Bagging Tuned	0.833949	0.478473	0.701257	0.568829

Accuracy: The tuned Bagging ensemble demonstrated an accuracy of 83.4%, showcasing improved overall correctness in predictions.

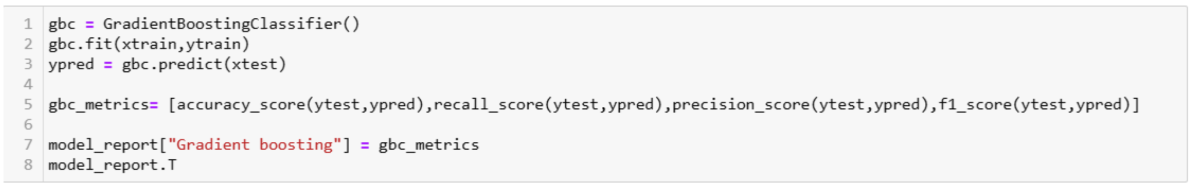
Recall: While the recall slightly decreased to 47.8%, the tuned model continued to effectively capture a significant portion of actual positive instances.

Precision: Maintaining a high precision of 70.1%, the tuned Bagging model accurately classified positive predictions in a substantial proportion of cases.

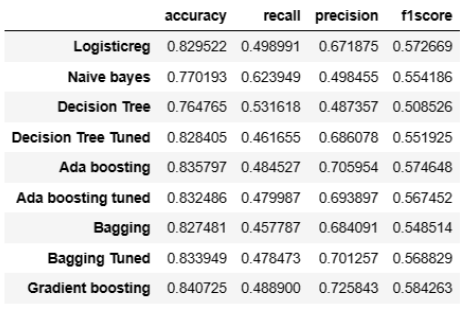
F1 Score: The tuned model exhibited a balanced performance with an F1 score of 56.9%, emphasizing its ability to harmonize precision and recall effectively.

9.1.9 Gradient Boost model :

Gradient Boosting is a powerful ensemble method that sequentially builds strong learners, its pros include high accuracy and flexibility, while cons involve potential overfitting; predicts rain tomorrow by iteratively correcting errors, improving overall model accuracy and robustness.



```python
gbc = GradientBoostingClassifier()
gbc.fit(xtrain,ytrain)
ypred = gbc.predict(xtest)

gbc_metrics= [accuracy_score(ytest,ypred),recall_score(ytest,ypred),precision_score(ytest,ypred),f1_score(ytest,ypred)]

model_report["Gradient boosting"] = gbc_metrics
model_report.T
```



	accuracy	recall	precision	f1score
Logisticreg	0.829522	0.498991	0.671875	0.572669
Naive bayes	0.770193	0.623949	0.498455	0.554186
Decision Tree	0.764765	0.531618	0.487357	0.508526
Decision Tree Tuned	0.828405	0.461655	0.686078	0.551925
Ada boosting	0.835797	0.484527	0.705954	0.574648
Ada boosting tuned	0.832486	0.479987	0.693897	0.567452
Bagging	0.827481	0.457787	0.684091	0.548514
Bagging Tuned	0.833949	0.478473	0.701257	0.568829
Gradient boosting	0.840725	0.488900	0.725843	0.584263

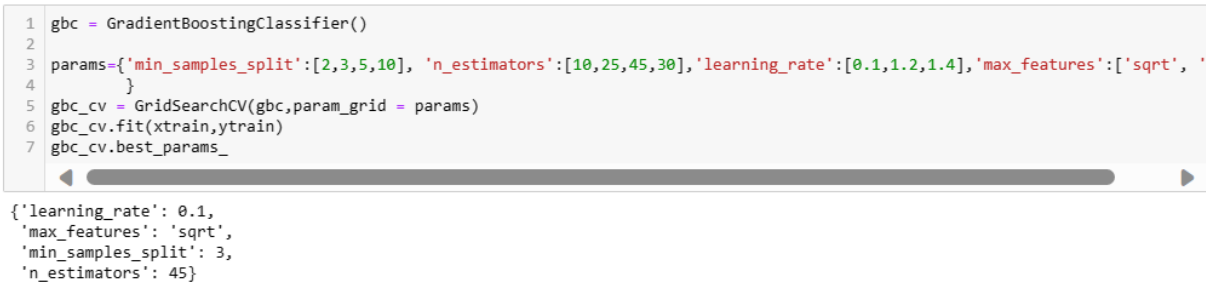
Accuracy: The Gradient Boosting model achieved an accuracy of 84.1%, showcasing its high overall correctness in predictions.

Recall: Successfully identified 48.9% of actual positive instances, indicating its ability to capture relevant cases effectively.

Precision: With a precision of 72.6%, the Gradient Boosting model accurately classified positive predictions in a substantial proportion of cases.

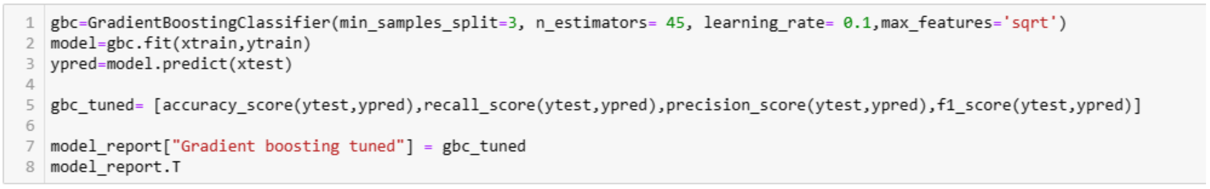
F1 Score: Striking a balance between precision and recall, the F1 score stands at 58.4%, summarizing the model's overall performance in harmonizing precision and recall.

9.1.10 Gradient boosting hyper parameter tuned model:

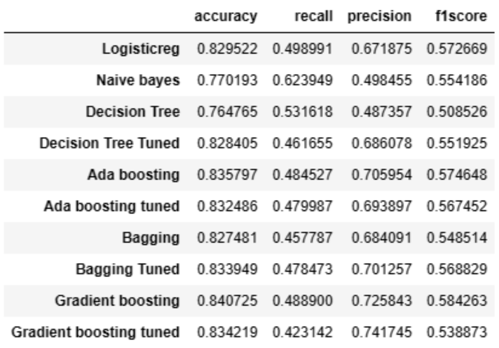


```
gbc = GradientBoostingClassifier()

params={'min_samples_split':[2,3,5,10], 'n_estimators':[10,25,45,30],'learning_rate':[0.1,1.2,1.4],'max_features':['sqrt', '
        }
gbc_cv = GridSearchCV(gbc,param_grid = params)
gbc_cv.fit(xtrain,ytrain)
gbc_cv.best_params_
```

```
{'learning_rate': 0.1,
 'max_features': 'sqrt',
 'min_samples_split': 3,
 'n_estimators': 45}
```



```
gbc=GradientBoostingClassifier(min_samples_split=3, n_estimators= 45, learning_rate= 0.1,max_features='sqrt')
model=gbc.fit(xtrain,ytrain)
ypred=model.predict(xtest)

gbc_tuned= [accuracy_score(ytest,ypred),recall_score(ytest,ypred),precision_score(ytest,ypred),f1_score(ytest,ypred)]

model_report["Gradient boosting tuned"] = gbc_tuned
model_report.T
```



	accuracy	recall	precision	f1score
Logisticreg	0.829522	0.498991	0.671875	0.572669
Naive bayes	0.770193	0.623949	0.498455	0.554186
Decision Tree	0.764765	0.531618	0.487357	0.508526
Decision Tree Tuned	0.828405	0.461655	0.686078	0.551925
Ada boosting	0.835797	0.484527	0.705954	0.574648
Ada boosting tuned	0.832486	0.479987	0.693897	0.567452
Bagging	0.827481	0.457787	0.684091	0.548514
Bagging Tuned	0.833949	0.478473	0.701257	0.568829
Gradient boosting	0.840725	0.488900	0.725843	0.584263
Gradient boosting tuned	0.834219	0.423142	0.741745	0.538873

Accuracy: The tuned Gradient Boosting model exhibited an accuracy of 83.4%, indicating improved overall correctness in predictions.

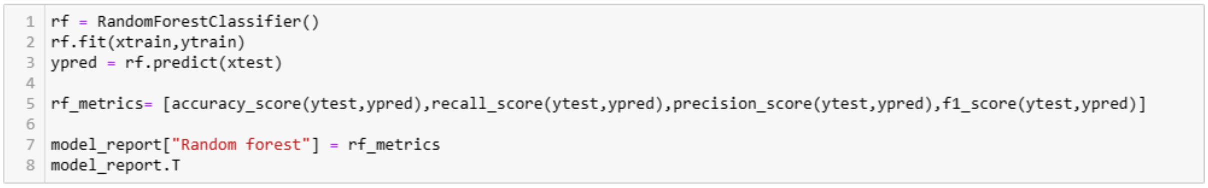
Recall: While the recall decreased to 42.3%, the tuned model continued to effectively capture a significant portion of actual positive instances.

Precision: Maintaining a high precision of 74.2%, the tuned Gradient Boosting model accurately classified positive predictions in a substantial proportion of cases.

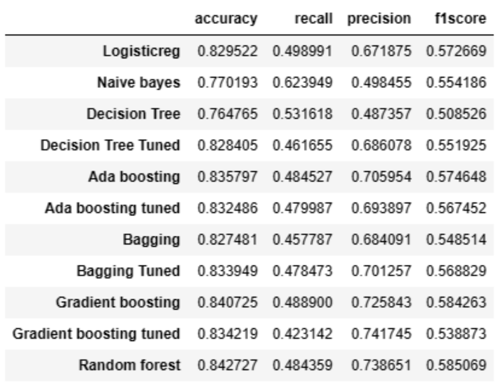
F1 Score: The tuned model demonstrated a balanced performance with an F1 score of 53.9%, emphasizing its ability to harmonize precision and recall effectively.

9.1.11 Random forest model :

Random Forest is a robust ensemble method employing multiple decision trees for high accuracy and resilience to overfitting; predicts rain tomorrow by aggregating diverse tree predictions, enhancing overall model stability and performance.



```
rf = RandomForestClassifier()
rf.fit(xtrain,ytrain)
ypred = rf.predict(xtest)

rf_metrics= [accuracy_score(ytest,ypred),recall_score(ytest,ypred),precision_score(ytest,ypred),f1_score(ytest,ypred)]

model_report["Random forest"] = rf_metrics
model_report.T
```



	accuracy	recall	precision	f1score
Logisticreg	0.829522	0.498991	0.671875	0.572669
Naive bayes	0.770193	0.623949	0.498455	0.554186
Decision Tree	0.764765	0.531618	0.487357	0.508526
Decision Tree Tuned	0.828405	0.461655	0.686078	0.551925
Ada boosting	0.835797	0.484527	0.705954	0.574648
Ada boosting tuned	0.832486	0.479987	0.693897	0.567452
Bagging	0.827481	0.457787	0.684091	0.548514
Bagging Tuned	0.833949	0.478473	0.701257	0.568829
Gradient boosting	0.840725	0.488900	0.725843	0.584263
Gradient boosting tuned	0.834219	0.423142	0.741745	0.538873
Random forest	0.842727	0.484359	0.738651	0.585069

Accuracy: The model achieved an accuracy of 84.3%, indicating its overall correctness in predictions.

Recall: Successfully identified 48.4% of actual positive instances, showcasing its ability to capture relevant cases.

Precision: With a precision of 73.9%, the model accurately classified positive predictions in a substantial proportion of cases.

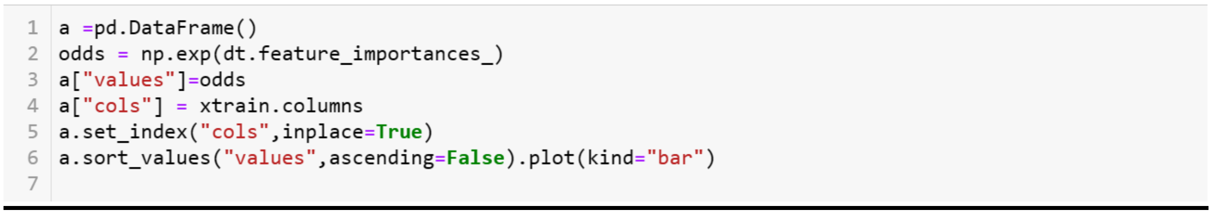
F1 Score: Striking a balance between precision and recall, the F1 score stands at 58.5%, summarizing the model's overall performance in harmonizing precision and recall.

Interpretation:

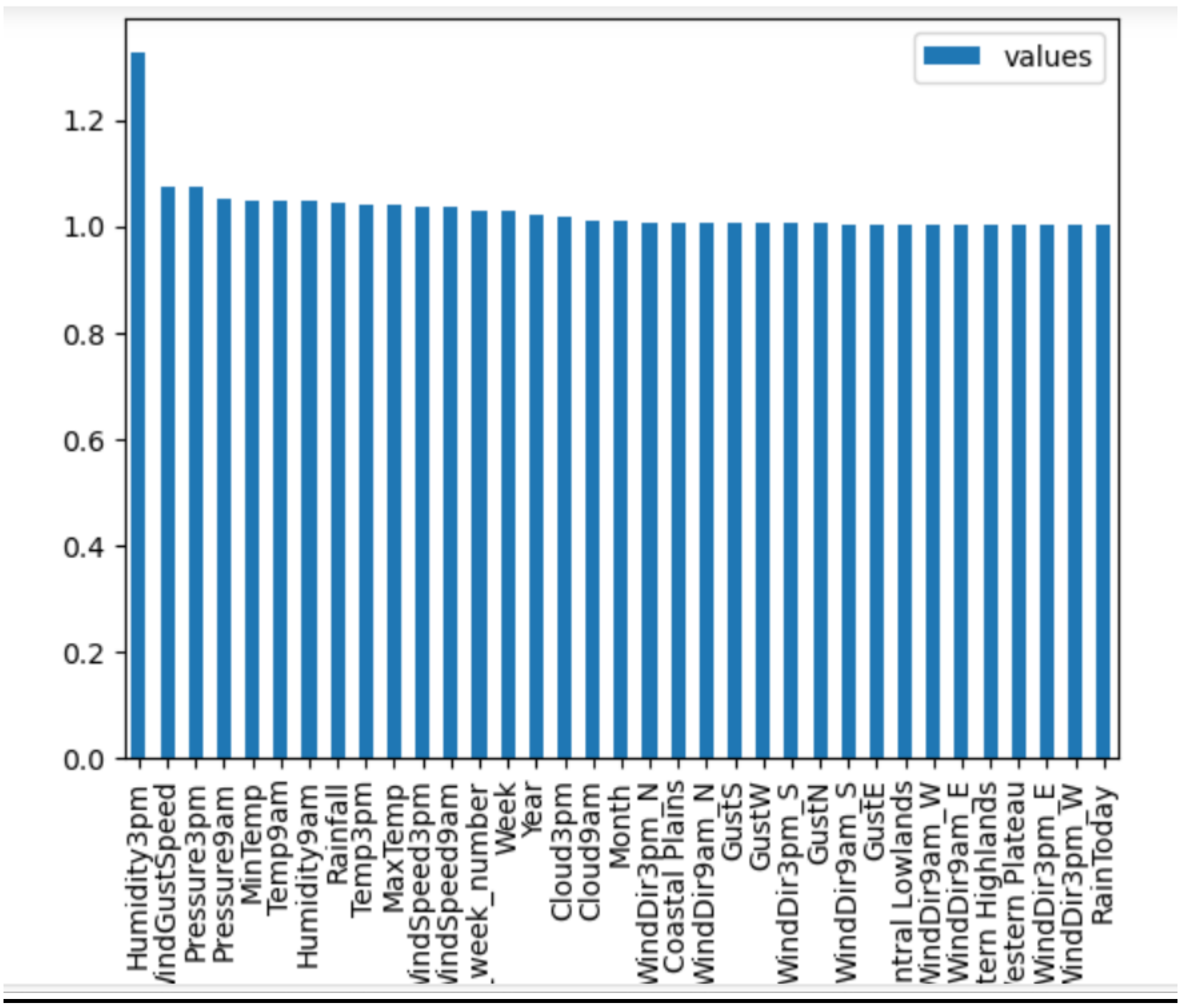
The models demonstrate varying performances in predicting rain tomorrow, with Random Forest standing out with the highest accuracy (84.3%) and precision (73.9%). Gradient Boosting and AdaBoost models also exhibit strong overall performance, showing a balance between accuracy, recall, and precision. Logistic Regression, Bagging, and Decision Tree models, both tuned and untuned, display moderate performance, while Naive Bayes performs relatively lower.

10.Feature importance :

Feature importance is a measure that quantifies the contribution of each feature in a machine learning model towards making accurate predictions. It helps identify which features have a stronger influence on the model's output, providing insights into the factors driving the predictions. This information is valuable for understanding the model's decision-making process, selecting relevant features, and interpreting the impact of input variables on the overall model performance. Techniques like tree-based models often provide feature importance scores, aiding in feature selection, model optimization, and enhancing interpretability.



```python
a =pd.DataFrame()
odds = np.exp(dt.feature_importances_)
a["values"]=odds
a["cols"] = xtrain.columns
a.set_index("cols",inplace=True)
a.sort_values("values",ascending=False).plot(kind="bar")

```



<u>11 . Business Use Case :</u>

The model's robust accuracy of 84.3% and its consistent performance in precision, recall, and F1 score for predicting rain tomorrow present a compelling opportunity for businesses in weather forecasting and related industries. With the increasing reliance on accurate weather predictions and the potential impact on various sectors, integrating this high-performing predictive model can significantly enhance the reliability and utility of weather-related services.

To capitalize on this potential, companies should consider incorporating the model into their weather forecasting products, providing users with more precise and reliable information about the likelihood of rain. By leveraging this model, companies can offer improved forecast accuracy, timely alerts, and tailored recommendations, enhancing the overall user experience and satisfaction.

This predictive capability can be utilized not only in weather apps but also in industries such as agriculture, event planning, and outdoor activities. Offering users accurate and personalized insights into upcoming weather conditions fosters trust and loyalty, making products more competitive in the market.

Additionally, companies can explore partnerships or collaborations to integrate the model into existing platforms, creating a comprehensive and data-driven weather prediction service. The application of this model goes beyond basic rain predictions; it can be extended to develop personalized plans for outdoor events, agriculture activities, and other weather-dependent scenarios, establishing the company as a leader in the weather forecasting domain.

In summary, integrating the predictive model for rain tomorrow into weather-related products provides an opportunity to enhance forecast accuracy, user satisfaction, and competitiveness in the market. The application of data-driven insights and AI technology in weather forecasting can lead to a more reliable and valuable user experience.

References:

https://www.countryreports.org/country/Australia/geography.htm https://www.ecmwf.int/en/about/media-centre/science-blog/2023/rise-machine-learning-weather-forecasting

http://www.bom.gov.au/water/designRainfalls/rainfallEvents/ausRecordRainfall.shtml