Stakeholder and Problem Statement:

Our stakeholder is a regional water resource management agency in Australia responsible for the sustainable management of water resources, including reservoir management and water supply planning.

Problem: The water resource management agency needs accurate predictions of rainfall for the next day to make informed decisions related to reservoir management, water allocation, and drought mitigation strategies. This prediction is essential for various weather-related decisions, including disaster preparedness.

Dataset Selection:

To address this problem, I have chosen the "Weather Dataset - Rattle Package" available on Kaggle. This dataset contains about 10 years of daily weather observations from many locations across Australia. This dataset includes historical weather data, including temperature, humidity, pressure and wind speed measurements at different times and rainfall measurements.

Target Variable:

The target variable is the prediction of rainfall for the next day, **RainTomorrow** (binary: yes/no). This column indicates whether there was rain the following day, with a "Yes" if the precipitation reached or exceeded 1mm. I aim to forecast the likelihood of rainfall based on historical weather data.

Features:

Column Name	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

Column Name	Description
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. This is measured in "oktas", which are a unit of eigths. It records how many eigths of the sky are obscured by cloud. A 0 measure indicates completely clear sky whilst an 8 indicates that it is completely overcast.
Cloud3pm	Fraction of sky obscured by cloud (in "oktas": eighths) at 3pm. See Cload9am for a description of the values
Temp9am	Temperature (degrees C) at 9am
Temp3pm	Temperature (degrees C) at 3pm
RainTomorrow	The amount of next day rain in mm. Used to create response variable RainTomorrow. A kind of measure of the "risk".

Dataset Exploration:

- **Data Overview:** The dataset contains historical weather data with records of various meteorological features and whether it rained the following day.
- **Categorical Features:** The dataset contains categorical features like 'Date', 'Location', 'WindGustDir', 'WindDir9am', 'WindDir3pm', 'RainToday', 'RainTomorrow' that need encoding.
- **Numerical Features:** The dataset contains numerical features like 'MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine', 'WindGustSpeed', 'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm', 'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am', 'Temp3pm' that need scaling.
- **Missing Values:** The dataset contains missing values in all columns except 'Date' and 'Location'.
- **Outliers:** The dataset contains outliers in many numerical columns.

Feature Transformations:

- **Converting Date column into day, month and year:** Splitting a date column into its day, month, and year components enhances the ability to perform temporal analysis, facilitates feature engineering, and supports various data analysis and modeling tasks that involve temporal data. It is a common practice to gain a more comprehensive understanding of how time influences the dataset and its relationships.
- **Handling Missing Values :** In **categorical columns**, I have replaced missing values with the mode. Mode imputation is commonly used for

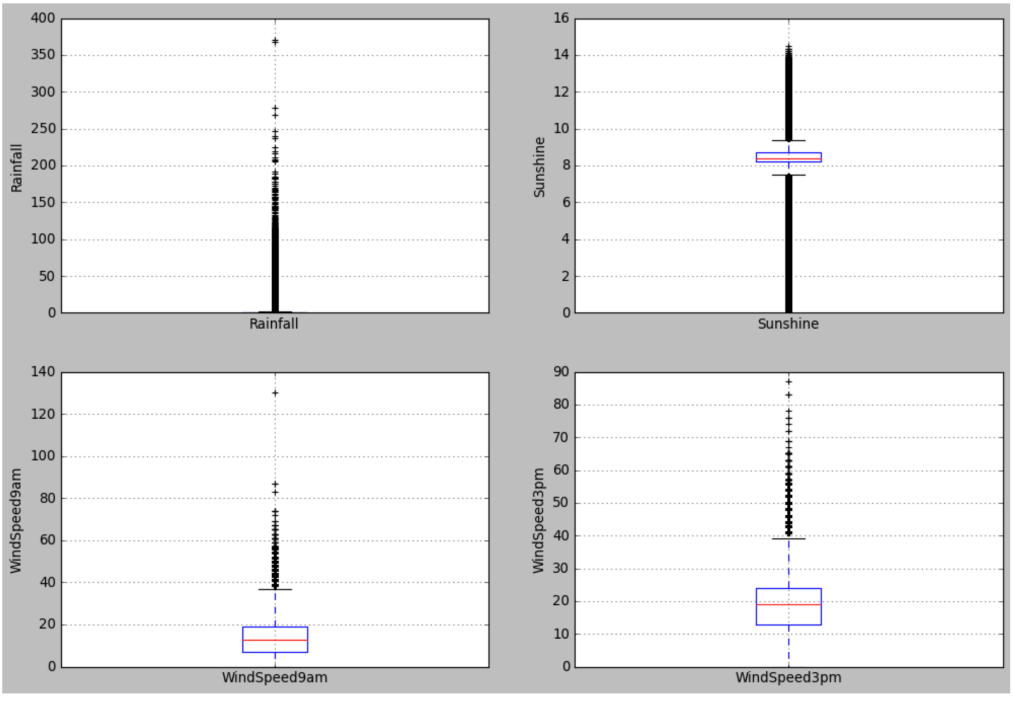
	Data Type	Number of null values	Percentage of missing values
Date	object	0	0.000000
Location	object	0	0.000000
MinTemp	float64	1485	1.020899
MaxTemp	float64	1261	0.866905
Rainfall	float64	3261	2.241853
Evaporation	float64	62790	43.166506
Sunshine	float64	69835	48.009762
WindGustDir	object	10326	7.098859
WindGustSpeed	float64	10263	7.055548
WindDir9am	object	10566	7.263853
WindDir3pm	object	4228	2.906641
WindSpeed9am	float64	1767	1.214767
WindSpeed3pm	float64	3062	2.105046
Humidity9am	float64	2654	1.824557
Humidity3pm	float64	4507	3.098446
Pressure9am	float64	15065	10.356799
Pressure3pm	float64	15028	10.331363
Cloud9am	float64	55888	38.421559
Cloud3pm	float64	59358	40.807095
Temp9am	float64	1767	1.214767
Temp3pm	float64	3609	2.481094
RainToday	object	3261	2.241853
RainTomorrow	object	3267	2.245978

categorical variables, where the concept of "average" or "mean" doesn't

make sense. In categorical data, the mode represents the most frequently occurring category, which is a logical choice to replace missing values. Imputing with the mode doesn't introduce bias into the data. It ensures that the imputed values are representative of the most common category, making it a neutral choice.

In **numerical columns**, I have dropped the 'Cloud9am', 'Cloud3pm', 'Evaporation', since they have very high percentage of missing values. It was a tough decision whether to drop the 'Sunshine' column or not. But, I have decided to impute the values with median since the distribution was skewed/ non-normal distribution. The median is robust to outliers and less influenced by extreme values.

- **Categorical Encoding:** Once I dealt with the missing values, I used label encoding for certain categorical features, preserving ordinal relationships where applicable. Label encoding is straightforward and easy to implement. It assigns a unique integer to each category, which simplifies the data representation.
  Label encoding can mislead machine learning algorithms, especially linear models, into assuming a false ordinal relationship between categories. This can lead to inaccurate predictions. Label encoding can be sensitive to new categories in the test data that were not present in the training data. These new categories can create issues during prediction.
- **Scaling Numerical Features:** Once I dealt with the missing values in numerical columns, I have used standard scaling to numerical features to ensure consistent magnitude. StandardScaler scales your features to have a mean of 0 and a standard deviation of 1. This can be advantageous for algorithms that assume features are normally distributed, and it ensures that all features have a similar scale. StandardScaler doesn't change the distribution or the shape of the data. It retains all the information in the original data, which is essential for some models. **I have done scaling on the entire dataset, but in production, we have to perform scaling**



separately on the train dataset and then do the same scaling on test dataset separately.
If you have extreme outliers, StandardScaler may distort the data distribution and compress the majority of your data into a very narrow range, making it less representative of the true data distribution.

- **Outliers:** There were some outliers in numerical columns.
I have removed the outliers based on their IQR range. Doing this, I have not really lost a lot of data.

```
In [69]: features.shape

Out[69]: (145460, 19)
```

```
In [70]: #Dropping the outliers
features["RainTomorrow"] = target

features = features[(features["MinTemp"]<2.3)&(features["MinTemp"]>-2.3)]
features = features[(features["MaxTemp"]<2.3)&(features["MaxTemp"]>-2)]
features = features[(features["Rainfall"]<4.5)]
features = features[(features["Sunshine"]<2.1)]
features = features[(features["WindGustSpeed"]<4)&(features["WindGustSpeed"]>-4)]
features = features[(features["WindSpeed9am"]<4)]
features = features[(features["WindSpeed3pm"]<2.5)]
features = features[(features["Humidity9am"]>-3)]
features = features[(features["Humidity3pm"]>-2.2)]
features = features[(features["Pressure9am"]< 2)&(features["Pressure9am"]>-2.7)]
features = features[(features["Pressure3pm"]< 2)&(features["Pressure3pm"]>-2.7)]

features.shape

Out[70]: (130981, 20)
```

Model Building:

I have not put much emphasis on this. I used Logistic Regression to build my model. Logistic regression is well-suited for binary classification tasks, where the goal is to predict one of two possible outcomes (in this case, whether it will rain or not). The dependent variable "RainTomorrow" is binary (Yes/No), making logistic regression a natural choice. Logistic regression can serve as a baseline model. It allows you to establish a benchmark for model performance, and you can then explore more complex models if needed.

I was able to get a model accuracy score of 84.38%
Here's an interpretation of the metrics:

```
In [79]: print(classification_report(y_test, y_pred_test))
```

	precision	recall	f1-score	support
0	0.86	0.95	0.91	20576
1	0.71	0.46	0.56	5621
accuracy			0.84	26197
macro avg	0.79	0.70	0.73	26197
weighted avg	0.83	0.84	0.83	26197

Precision (for class 0): Precision for class 0 is 0.86. This means that, of all the instances predicted as class 0, 86% were correctly classified as class 0.

Recall (for class 0): Recall for class 0 is 0.95. This indicates that the model was able to correctly identify 95% of all the actual instances of class 0.

F1-Score (for class 0): The F1-score for class 0 is 0.91. The F1-score is the harmonic mean of precision and recall. An F1-score of 0.91 is a good balance between precision and recall for class 0.

Support (for class 0): The support for class 0 is 20,576, meaning that there are 20,576 instances in the dataset that belong to class 0.

Precision (for class 1): Precision for class 1 is 0.71. This indicates that, of all the instances predicted as class 1, 71% were correctly classified as class 1.

Recall (for class 1): Recall for class 1 is 0.46. This means that the model was able to correctly identify 46% of all the actual instances of class 1.

F1-Score (for class 1): The F1-score for class 1 is 0.56. An F1-score of 0.56 is a moderate balance between precision and recall for class 1.

Support (for class 1): The support for class 1 is 5,621, indicating that there are 5,621 instances in the dataset that belong to class 1.

Limitations:

- Predicting rain for the next day has limitations due to the complexity of weather systems and short lead times. Key challenges include the accuracy of initial conditions, local variability, and numerical model limitations.
- Weather forecasts are expressed as probabilities, and there's always some level of uncertainty.
- Rainfall prediction often involves temporal dependencies and patterns. Logistic regression does not inherently account for time series data, which can limit its ability to capture seasonality and long-term trends.
- Standard scaling assumes that data follows a normal distribution and centers it around zero with a standard deviation of one. However, it can be sensitive to outliers. Outliers can heavily influence the mean and standard deviation, potentially distorting the scaled data.