

**CSE422: Artificial Intelligence**

**Project Name: Rainfall Prediction**

**Group: 04**

|  |  |  |
| --- | --- | --- |
| **No.** | **Name** | **ID** |
| 1 | Murshed Hasan | 23141066 |
| 2 | Shafin Abrar Mufdi | 23341122 |
| 3 | Ishrat Jahan | 20301152 |
| 4 | Syeda Jannatul Ferdous | 20301067 |

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Section no** | **Content** | **Page no** |
| **1** | Introduction | **3** |
| **2** | Dataset Description | **4** |
| **3** | Dataset Pre-processing | **8** |
| **4** | Feature scaling | **9** |
| **5** | Dataset Splitting | **10** |
| **6** | Model training and testing | **10** |
| **7** | Model selection | **16** |
| **8** | Conclusion | **20** |

**Introduction**

Rainfall is a climatic factor that significantly influences a country's socio-economic development, particularly agriculture and the management of water resources and disasters. Efficient planning and decision-making for various sectors require accurate rainfall predictions, which data-driven techniques have made possible with advancements in technology. Few of the promising approaches are the Random Forest, the Logistic Regression and the Decision Tree algorithms, which are known for their ability to tackle intricate and nonlinear data relationships.

Our report delves into predicting rainfall using these algorithms by examining their efficacy on a collection of rain pattern datasets gathered across multiple regions in Australia.

The precision of the forecasts created by these algorithms are assessed, with an analysis of the factors impacting their performance presented. The report additionally explores the potential uses of these algorithms regarding rainfall prediction.

In summary, this report offers an extensive examination of the Random Forest, the Logistic Regression and the Decision Tree algorithms and their usefulness in predicting rainfall. This report also shows the visualize comparisons between these algorithms. It underscores how data-driven techniques can enhance forecasting accuracy while presenting valuable observations about how well this application performs.

**Data Description**

**Source: Kaggle**

**Link:** https://www.kaggle.com/code/gauravduttakiit/rainfall-prediction-with-ml/input

**Reference:**

[1] P. Grover, “Intuitive Interpretation of Random Forest,” USF-Data Science, Nov. 29, 2017.<https://medium.com/usf-msds/intuitive-interpretation-of-random-forest-2238687cae45>

[2] “Random forest interpretation with scikit-learn | Diving into data.”<https://blog.datadive.net/random-forest-interpretation-with-scikit-learn/>

**Data Acquisition:** We obtained the rainfall dataset for this investigation from Kaggle. The dataset includes details on the monthly rainfall trends for several Australian areas. The dataset was gathered from a number of sources, including research organizations, government agencies, and meteorological departments.

There are 24 features in our dataset. There are 24 columns and 142,913 total observations in the dataset. The data covers a nine-year period, from 2008 to 2017. The features are of both quantitative and categorical. Among the features location, evaporation, sunshine, WindGustDir, WindDir9am, WindDir3am, RainToday consist of categorical feature. Rest of them have quantitative features.

We obtained the dataset from Kaggle when it was first made accessible in CSV format. The data was loaded and preprocessed using Python and the Pandas module  
  
**Problem category:** The problem that we are working on is a classification problem. It is because we are trying to predict the rainfall of the next day based on the features available in the dataset.

Since we’re predicting whether it will rain or not on a given day, our problem becomes a binary classification problem. We are classifying each day into two classes: “Rain” (positive class) or “No Rain” (Negative class).   
  
**Data Points:** The dataset we have chosen has 142,913 data points.

Our dataset has both quantitative and categorical features. But we impute the categorical features using Mode Imputation.

**Correlation of features:**

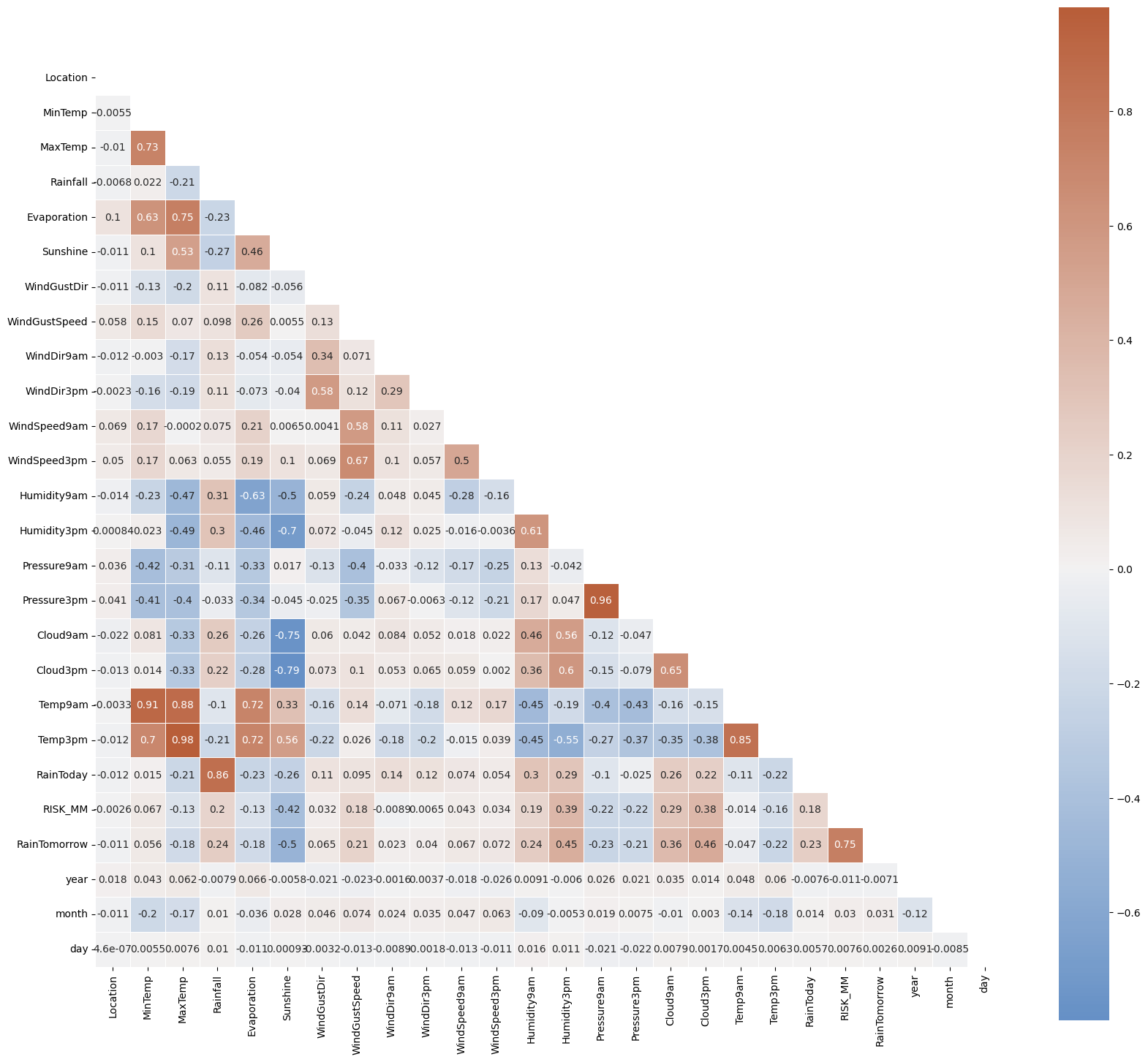
After cleaning the data we have used Seaborn library to plot the correlation heatmap of the input and output features. From this we can easily visualize how closely the features are related to each other by seeing the positive and negative correlations. ****

Figure . Correlation Heatmap

**Imbalanced Dataset:** No, for the output features unique classes do not have an equal number of instances.

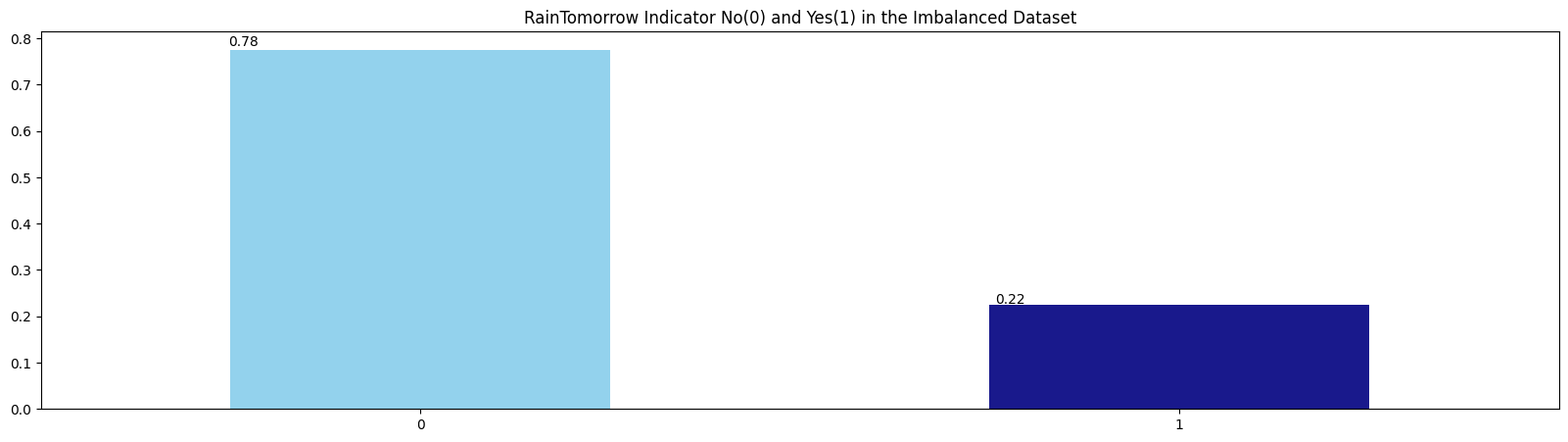


Figure . Imbalanced Dataset

From here we can see that the output feature is RainTomorrow which has two classes: yes and no. No has large amount of instances than Yes. Therefore, it has a significant imbalance which could affect the models’ performances.  
  
**Balanced Dataset:** In the preprocessing phase, it’s important to ensure that the dataset is balanced, especially when dealing with imbalanced classes. Imbalanced classes can lead to biased model perfomances, where the model might favor the majority class and not perform well on the minority class. To address this, We applied oversampling to balance the distribution of the target output variable “RainTomorrow”. The goal was to ensure an equal representation of both classes (“No” and “Yes”) in the dataset.

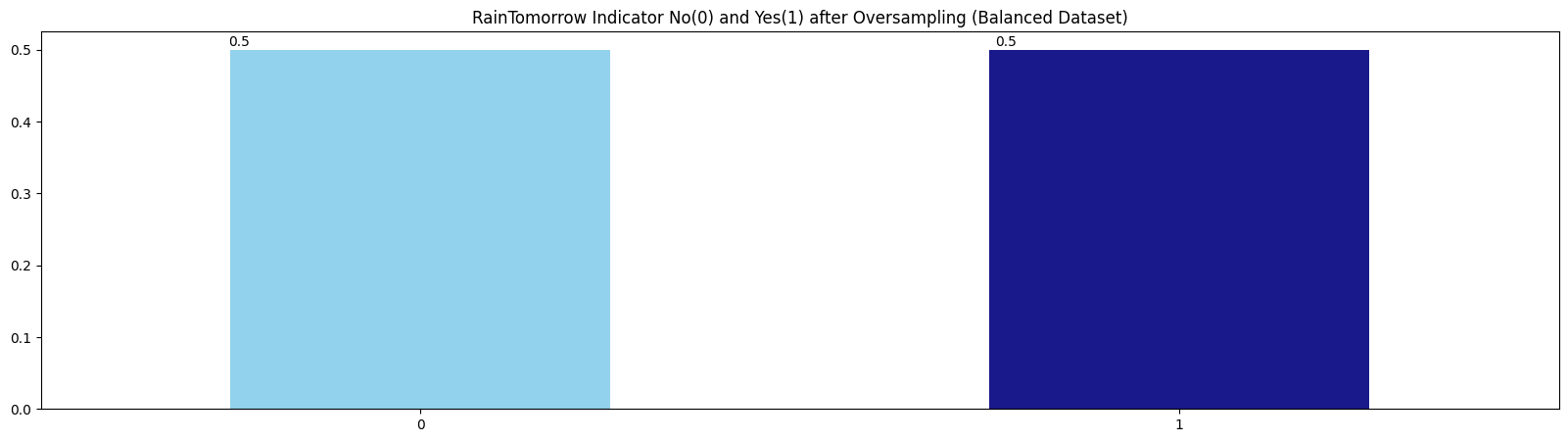


Figure . Balanced Dataset

From the above plotting we can see that the target output is balanced now after applying oversampling.

**Data Pre-Processing**

**Missing Data Pattern:** After oversampling we tried to find the missing data pattern using a heatmap which is shown below

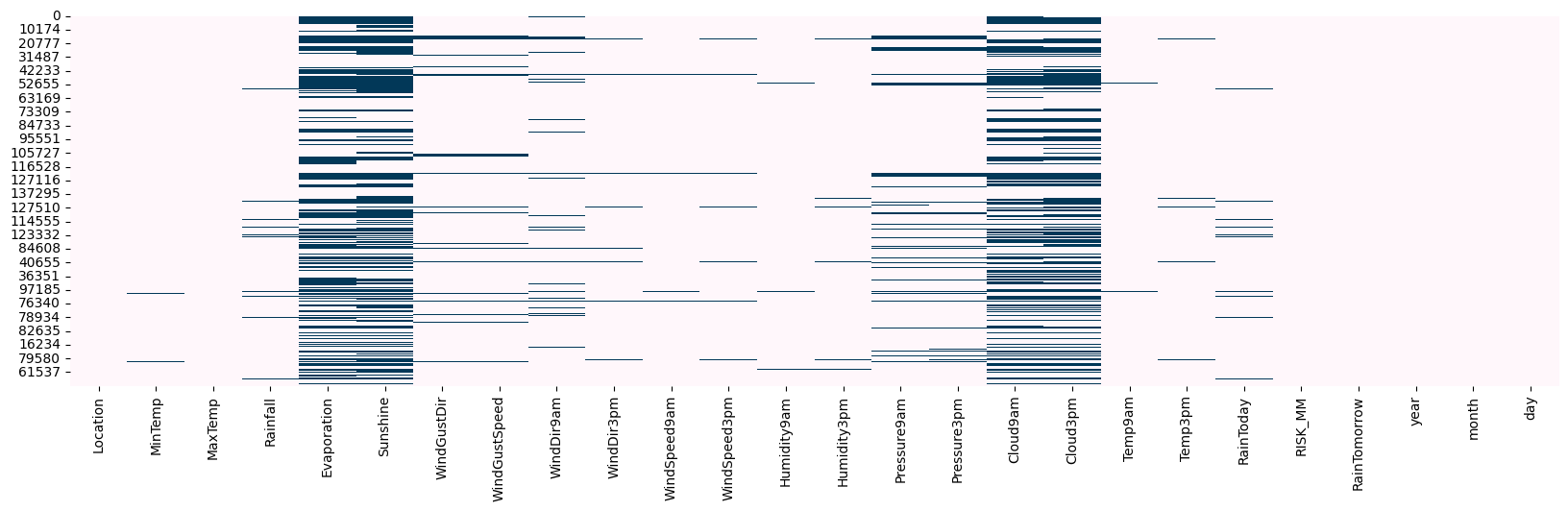


Figure . Missing Data Pattern

The missing data is likely due to a variety of reasons, such as human error, data entry errors. It is important to address this before training the model.  
  
We have also found the categorical features which are later converted into continuous features using label encoding after imputing them with mode imputation for a better classification accuracy.

As we have seen that there are a large number of missing values in the missing data pattern, we handle theses missing values by using mode imputation to fill these missing values. Then after label encoding we again run them through multiple mode imputation so that there are no features left with missing values.   
  
We have used label encoding to convert the categorical features to continuous features. This is because machine learning models are not able to directly understand categorical features. They need to be converted to continuous features before they can be used. Label encoding is a common way to do this, and it involves assigning a unique integer to each category in the feature. This makes the features more understandable to the machine learning model and can help to improve the performance of the model.

We have also used Multiple Imputation Equations (MICE) to impute the missing values in the dataset. MICE is a multivariate imputation method that iteratively imputes the missing values in a dataset. This makes it a more effective way to impute missing values than univariate imputation methods, which only impute the missing values in one feature at a time.

**Detecting and Removing Outliers:** Outliers are data points that are significantly different from the rest of the data. We used the interquartile range (IQR) to detect and remove the outliers from the dataset. The IQR is a measure of the variability of the data, and it is calculated by subtracting the 25th percentile from the 75th percentile. Outliers are any data pints that are more than 1.5 IQRs away from the percentiles. We removes these outliers because they can negatively impact the performance of the machine learning models.

**Feature Scaling**

In the process of preparing the dataset for modeling, We carried out feature scaling and selecting techniques to ensure that the input features are appropriately normalized and relevant for accurate model training and evaluation. Below is a breakdown of the steps we took.

1. **Standardization using Min-Max Scaling**: I began by standardizing the dataset using Min-Max scaling a common method that scales the features to a specified range, typically between 0 and 1. This scaling technique ensures that all features have the same scale, making the dataset suitable for these algorithms.
2. **Feature Importance using Filter Method (Chi-Square)**: Next, we used the SelectKBest method with a chi-squared (chi2) statistic to select the top k features that have the highest impact on the target variable. This helps in reducing the dimensionality of the dataset and focusing on the most relevant features for modeling.
3. **Feature Importance using Model-Based selection(Random Forest)**: Additionally, we employed a model-based feature selection technique using Random Forest as the classifier. This method estimates feature importance based on the contribution of each feature to the predictive power of the model.

**Dataset Splitting**

1. **Splitting intro Train and Test Sets**: After feature selection, we split the dataset into training and testing sets using a 70-30 split ratio. This division allows us to train the model on one subset and evaluate its performance on another, ensuring that the model’s effectiveness can be accurately assessed. The dataset is stratified. The “stratify=target” argument used in the “train\_test\_split()” function ensures that the proportion of the target variable in the train and test sets is the same.
2. **Feature Scaling using StandardScaler**: Finally, we applied feature scaling using the StandarScaler, which standardizes features by removing the mean and scaling to unit variance. This step ensures that all features have a similar scale, which is important for many machine learning algorithms that are sensitive to feature magnitudes.

**Model Training and Testing**

In the phase of building and evaluating predictive models, we implemented a comprehensive approach that involved training several machine learning classifiers including Logistic Regression, Decision Tree and Random Forest. Here’s and in-depth explanation of the methodology we used:

1. **Defining the Evaluation Metrics**: We began by importing necessary libraries and defining key evaluation metrics to asses the models performance. The metrics include Accuracy, ROC area under curve, Cohen’s Kappa, and the execution time of each model. These metrics offer a comprehensive view of how well the models are performing.
2. **The run\_model Function**: We designed a versatile function to streamline the process of training and evaluating models. This function takes as input the model to be trained, the training and testing datasets, and optional verbosity settings. Inside the function:

* The model is trained on the training dataset.
* Predictions are made on the testing dataset.
* The detailed evaluation metrics are computed and printed
* A detailed classification report is displayed providing precision and recall , F1 score and support for both classes.
* ROC curve is plotted using the plot\_roc\_cur function.
* A normalized confusion matrix is displayed using the ConfusionMatrixDisplay function.

1. **Logistical Regression**: We defined the hyperparameters and created and instance of the classifier, It was trained and evaluated using the testing dataset. We utilized this algorithm with specific parameters (penalty: ‘l1’, solver: ‘liblinear’). Below are the ROC and Confusion matrix plot for this algorithm given.

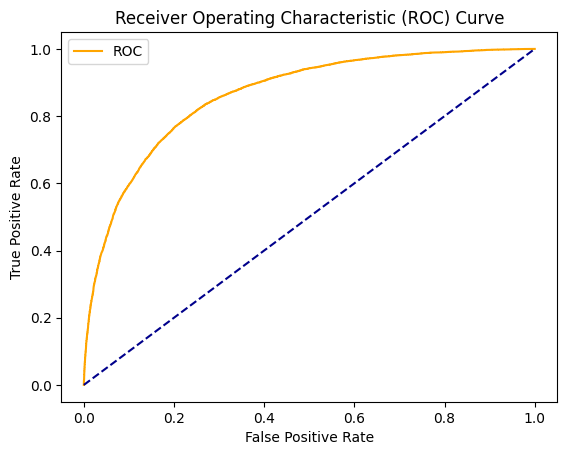


Figure . ROC Curve for Logistical Regression

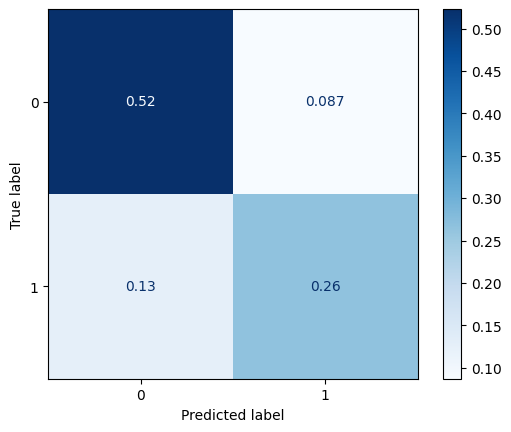


Figure . Confusion Matrix for Logistical Regression

1. **Decision Tree**: We defined the hyperparameters and created and instance of the classifier, It was trained and evaluated using the testing dataset. We employed this classifier, setting the maximum depth and maximum features. Below are the ROC and Confusion matrix plot for this algorithm given.

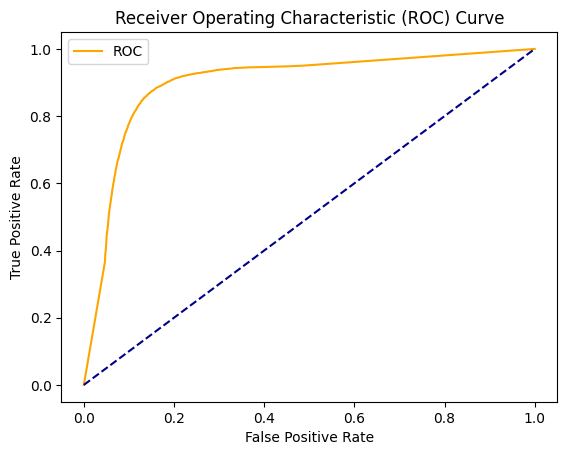


Figure . ROC Curve for Decision Tree

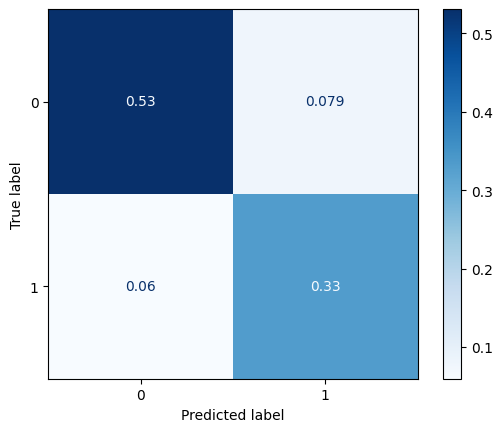


Figure . Confusion Matrix for Decision Tree

1. **Random Forest**: We defined the hyperparameters and created and instance of the classifier, It was trained and evaluated using the testing dataset. We implemented this algorithm with various hyperparameters including maximum depth, minimum samples leaf and split, number of estimators and random state. Below are the ROC and Confusion matrix plot for this algorithm given.

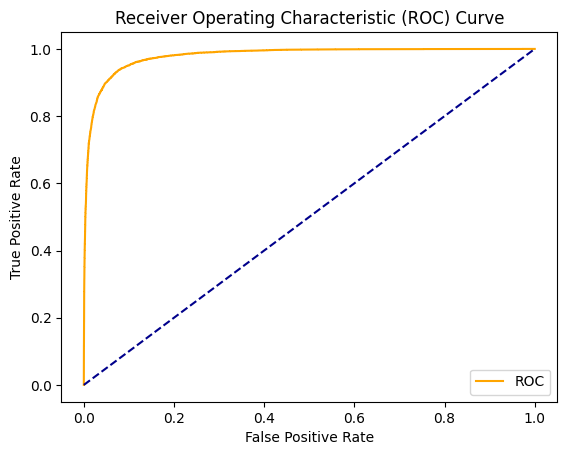


Figure . ROC Curve for Random Forest

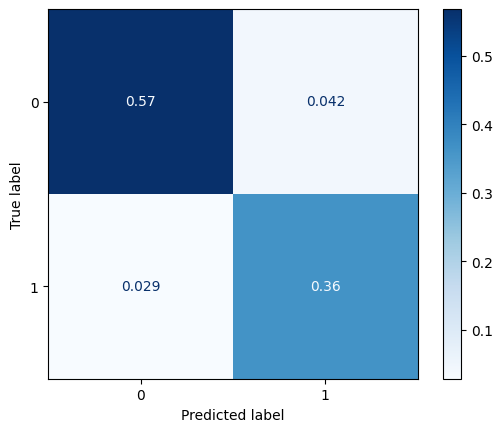


Figure . Confusion Matrix for Random Forest

**Model Selection/Comparison Analysis**

For model selection and comparison, we have plotted Precision-Recall curve. We have also compared the time taken to train these models and their accuracy. We have also used bar charts to compare their ROC and Cohen Kappa. Below are all the comparison charts shown,

**Precision-Recall Curve**: Here, we can see that Logistic Regression is doing much better than the other two models.

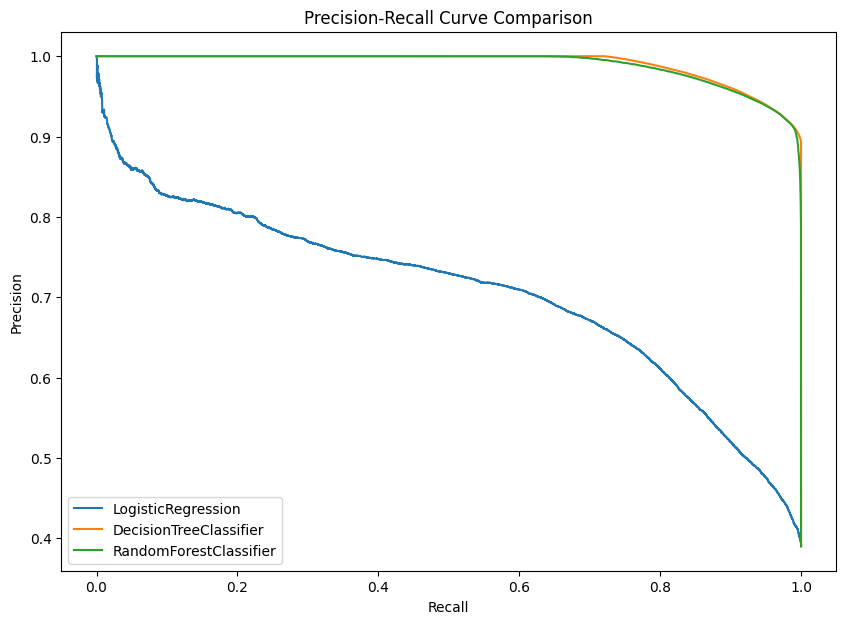


Figure . Precision-Recall Curve Comparison

**Accuracy Prediction and Time taken for execution**: We can see from the chart that Random Forest takes a lot of time to be trained where as Decision Tree and Logistic Regression are much faster. But when it comes to accuracy score Random performs better than the other two. The model's accuracy is 91.70%, indicating that it correctly classified 91.70% of the instances in the test dataset. This metric is helpful for understanding the model's overall performance but should be interpreted with caution, as it may not accurately reflect the model's ability to predict both positive and negative instances.

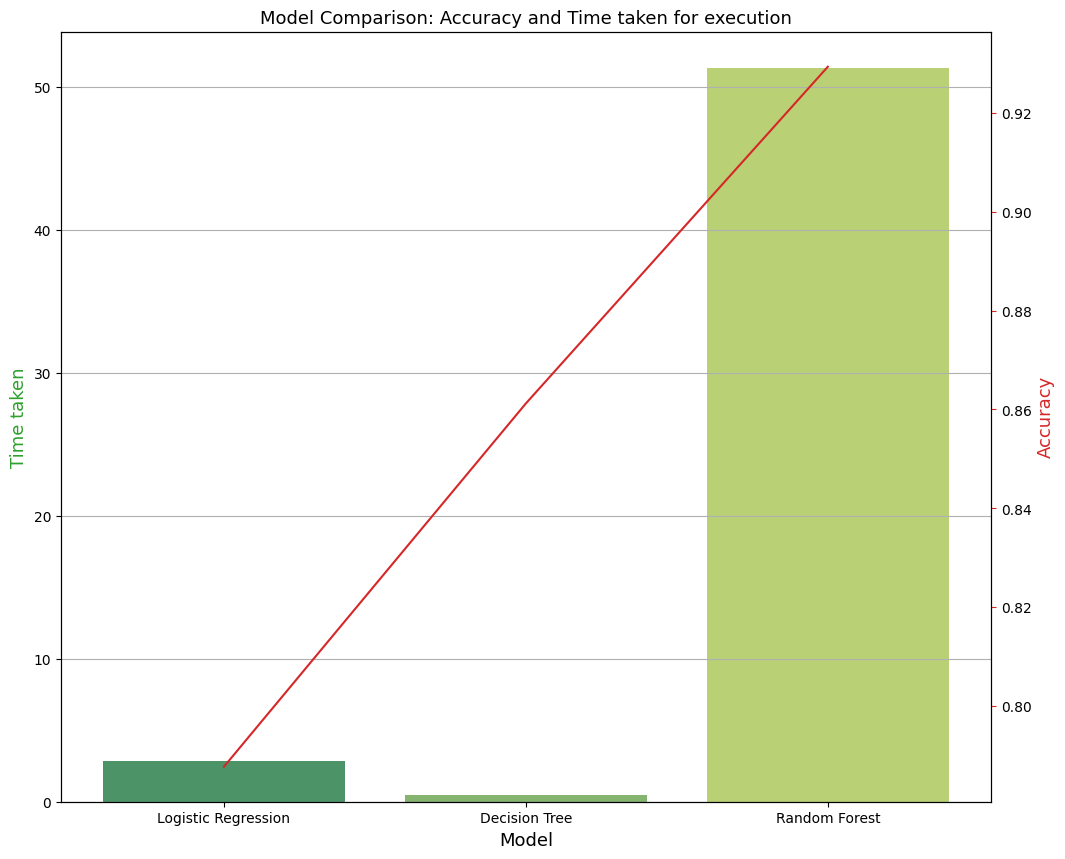


Figure . Model Comparison: Accuracy and Time taken for execution

**Area under ROC and Cohen Kappa:** In this comparison, Random Forest again performs better than the other two algorithms.

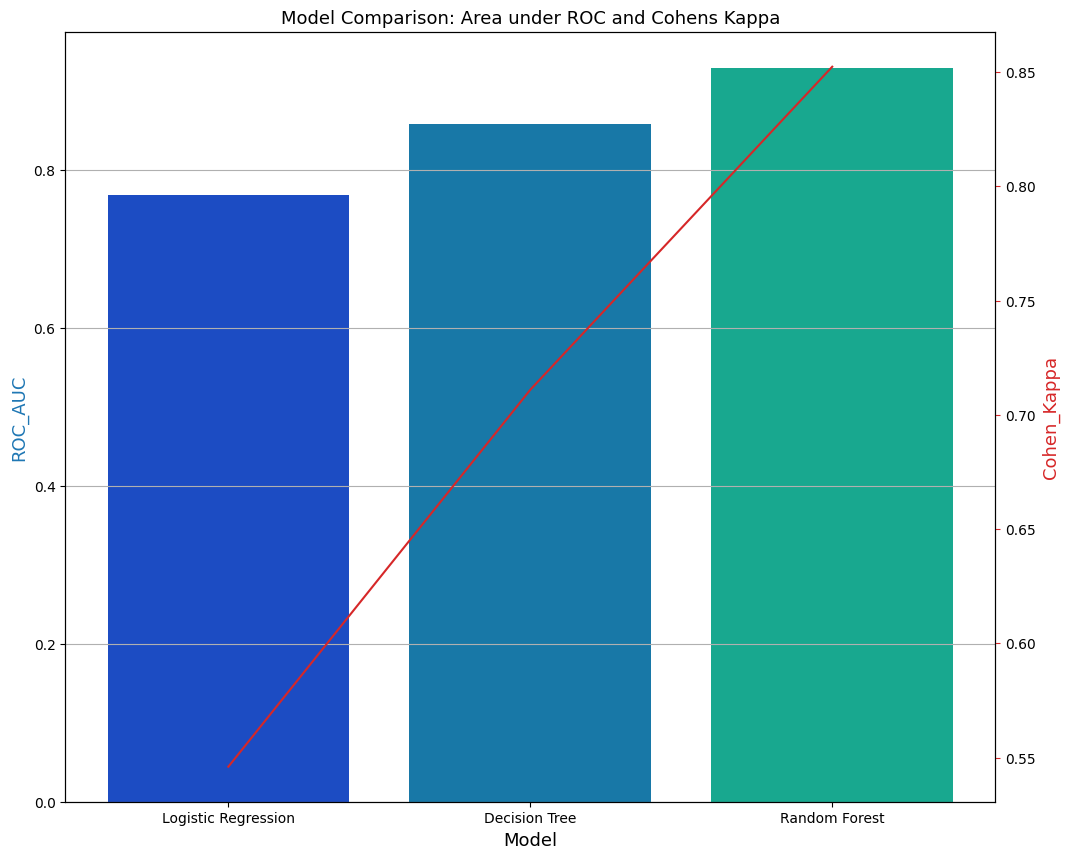


Figure . Model Comparison: Area under ROC and Cohens Kappa

So from the above comparisons, we can conclude by saying that the Random Forest algorithm is the best suited model for Rainfall Prediction because of it’s higher accuracy.

**Conclusion**

In conclusion, this work offers invaluable insights into the functionality and prospective uses of the Random Forest, Decision Tree and the Logistic Regression method. We anticipate that the application of machine learning techniques in the field of rainfall prediction will be of interest to researchers, decision-makers, and practitioners. We also discovered that a variety of variables, like the quantity of decision trees, the depth of the trees, and the amount of characteristics utilized in the model, affect how well the Random Forest approach performs. To attain the optimum results, it is crucial to properly set these parameters.