ABSTRACT

Rainfall prediction is helpful to avoid many disasters. Rainfall is a climatic factor that affects many human activities like agriculture, production, and forestry. Predicting the rainfall before itself helps in the prevention of the losses. The dataset consisting of various climatic attributes was sourced from Kaggle spanning 2008-2018. The existing system includes the following techniques Random Forest, K-Nearest Neighbors (KNN), Logistic Regression, Decision Tree, Ensemble AdaBoost which yielded an accuracy of 88% ,87% ,84% ,70% and 85% respectively. In the proposed system we used machine learning algorithms such as Logistic Regression, Light GBM, CatBoost and XGBoost. The accuracy obtained after performing these techniques are79%, 87% ,94.7% and 94% respectively. The performance of the algorithms are examined based on Precision, Recall, F1-score, Accuracy, Time Taken for Execution, Area under ROC, Support and Cohen's Kappa. The prediction models developed are compared to one another to display the best algorithm. The first comparison is done based on Accuracy and Time Taken for Execution and based on Area under ROC and Cohen's Kappa. After the performance analysis XGBoost is said to have high performance compared to other models whereas Logistic Regression has less performance.

TABLE OF CONTENTS

9.

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Rainfall is a climatic factor that affects many human activities like agricultural production, construction, power generation, forestry, among others. To this extent, rainfall prediction is essential since this variable is the one with the highest correlation with adverse natural events such as landslides, flooding, mass movements and avalanches [8]. Rainfall Prediction is the application of science and technology to predict the amount of rainfall over a region. It is important to exactly determine the rainfall for the effective use of water resources, crop productivity and pre-planning of water structures. We aim to provide the user with the best model for rainfall prediction.

1.1. OVERVIEW

Rainfall is a crucial occurrence within a meteorological system because its irregular nature immediately affects water-useful planning, agricultural, and biological processes. An important component of the hydrological cycle is rainfall, and changes to its pattern have an immediate effect on water reserves [7]. These incidents have affected society for years. Therefore, having an appropriate approach for rainfall prediction makes it possible to take preventive and mitigation measures for these natural phenomena.

The prediction mainly helps farmers and water resources can be utilized efficiently. Rainfall prediction is a challenging task, and the results should be accurate. There are many hardware devices for predicting rainfall by using the weather conditions like temperature, humidity, pressure.

These traditional methods cannot work in an efficient way so by using machine learning techniques we can produce accurate results. We can just do it by having the historical data analysis of rainfall and can predict the rainfall

for future seasons. We can apply many techniques like classification, regression according to the requirements. This studypresents a set of experim--ents that involve the use of common machine learning techniques to create models that can predict rainfall.

Our project involves utilizing machine learning techniques to build various models to predict rainfall. This project is intended to be one tool that Materials and Methods

1.2 PROBLEM STATEMENT

Rainfall prediction is very important because heavy and irregular rainfall can have many impacts like the destruction of crops and farms, damage of property so a better forecasting model is required for an early warning that can reduce the risks to life and property and helps to manage the agricultural farmsin a better way. Heavy rainfall is a cause for natural disasters like floods and drought that square measure encountered by individuals across the world each year.

Many models are developed to evaluate the rainfall and for predicting the likeliness of rain. These models are based on both supervised and unsupervised machine learning algorithms. Taking into consideration of overall rainfall will not help us to know if it rains in specific conditions. Accuracy is the major concern in machine learning. We are going to understand the data and then train the model accordingly to predict rainfall and which model is best suitable for theprediction of rainfall.

1.3 EXISTING SYSTEM

There exists a large source of research on the topic of machine learning methods for rainfall prediction, most of it has been focusing on classification, regression-based techniques. Nikhil Oswal [7] outlines various approaches to predict rainfall. He used Logistic Regression, Decision Tree, K-Nearest

Neighbors, Decision Tree, Random Forest, AdaBoost and Gradient Boosting. The analysis of dataset to predict rainfall was done by 3 ways namely experimenting with the original dataset, undersampled dataset, oversampled dataset. Were able to achieve 84-91%accuracy by using 3 ways.

K. Nanda Kumar, V. Chandrasekar [6] implemented a prediction system to compare Linear Regression over Logistic Regression. Sample size is calculated using G power software and determined as 10 per group with pretest power 80%, threshold 0.05% and CI 95%. They were able to achieve an accuracy of 91.18% for Linear Regression.

1.3.1 Materials and Methods

a. Dataset

The datasets we used in this study are open source and freely available in Kaggle. The dataset contains about 10 years of daily weather observations from numerous Australian weather stations from 2008-2018. The dataset contains 145460 entries of data.

b. Libraries

In order to perform this classification, you need the basic Data Scientist starter pack (sklearn, pandas, NumPy, matplotlib, seaborn) plus some specific libraries like resample, LabelEncoder, preprocessing, SelectKBest, chi2, train_test_split, StandardScaler, time, accuracy_score, roc_auc_score, cohen_kappa_score, plot_confusion_matrix, roc_curve, classification_report, LogisticRegression, lgb, cb, xgb.

c. Data Cleaning and Preparation

In this project, we can't use text data directly because it is necessary to clean any data before using the data. If we used it directly without cleaning, then it is very hard for the ML algorithm to detect patterns in that data and sometimesit will also generate an error.

d. Data Exploration

In this project, we have explored how machine algorithms work for the prediction? How is each parameter correlated with each other? What are the parameters that are necessary to predict the rainfall?

e. Training

In this project, we are using supervised learning algorithms such as Logistic Regression, Decision Tree, K-Nearest Neighbor, Decision Table, AdaBoost, Gradient Boosting and Random Forest.

Logistic regression: It predicts the output of a categorical dependent variable.

Decision Tree: It usually mimic human thinking ability while making a decision, so it is easy to understand. It shows a tree-like structure.

Random Forest: It takes the prediction from each tree and based on the majorityvotes of predictions, and it predicts the final output.

K-Nearest Neighbors: is a non-parametric and lazy algorithm. Non-parametric means there is no assumption for underlying data distribution. Lazy algorithm means it does not need any training data point for model generation.

Decision Table: provides a handy and compact way to represent complex business logic. In a decision table, business logic is well divided into conditions,actions (decisions) and rules for representing the various components that form the business logic.

AdaBoost: fits a sequence of weak learners on different weighted training data. It starts by predicting original data set and gives equal weight to each observation.

Gradient Boosting: The main idea behind this algorithm is to construct new base learners which can be optimally correlated with negative gradient of the loss function, relevant to the whole ensemble.

f. Classification Metrics

To check how well our model, we use some metrics to find the accuracy of our model. There are many types of classification metrics available in Scikit learn, Confusion Matrix, Accuracy Score, Precision, Recall, F1-Score, Area under ROC, Stratified k-fold, Statistical Testing.

1.4 PROPOSED SYSTEM

Due to the uncertain weather, it always remains a concern to predict the rainfall. This is why we proposed machine learning methods such as Logistic Regression, LightGBM, CatBoost and XGBoost. The proposed method is entirely composed of Machine Learning approaches, which yields higher accuracy.

We selected datasets from Kaggle for our experiments which contain about 10 years of daily weather observations from numerous Australian weather stations from 2008-2018. And the next step involves changing the Rain today and Rain tomorrow (Yes/No) into binary values (0/1). Perform EDA for the dataset if dataset not balanced then oversample and balance it.

Then, use LabelEnocer to convert categorical values to numerical values and imputing them by using Multiple Imputations by Chained Equations (MICE). After that detecting any outliers is present or not. If present remove the outliers.

Normalizing the data and then using Filter method for Feature Selection. Then, the dataset is split into train and test set. Each model incorporated is trained with the dataset with a set of evaluation metrics to find out the optimal model that is which has high accuracy. The trained are compared to one other to one another to calculate the performance.

CHAPTER 2

LITERATURE REVIEW

B. Revathi, et.al [1], proposed Rainfall Prediction using Machine Learning Classification Algorithms. In this paper two algorithms are used namely Classification and Regression trees (CART) and Intelligent Decision Tree algorithm (IDA). The dataset is preprocesses, split into test and train test and then training the models using CART and IDA. Then a comparison is taken between CART and IDA. CART performs the best with an accuracy of 80%.

D.Sirisha et.al [2] proposed Predicting Rainfall using Machine Learning Techniques. In this paper, a rainfall prediction model is proposed using Multiple Linear Regression (MLR) using the dataset of India. The data taken is from 1901 to 2015 monthly wise. The proposed method is compared with existing methods namely Quantitative Precipitation Forecasts, Logistic Regression. The proposed machine learning model devised is evaluated using the parameters viz., Mean Square Error (MSE), accuracy, and correlation. Thus, MLR performed best with MSE of 12.374, RMSE of 3.789 and correlation of 0.492.

Devanshi Shukla et.al [3] proposed rainfall prediction using Neural Networks In this paper, we propose the Multilayered neural network with Back-propagation learning algorithm and showed that Temperature, relative humidity, vapour pressure are factors to predict rainfall. After a series of analysis, it is found that rainfall prediction better accurate results are produced during mid monsoon that is from June to September. Have configured feed forward and cascade network with 1000 epoch from which feed forward yielded a better accuracy. Accuracy yielded is 82%.

G. Bala Sai Tarun et al.[4] proposed rainfall prediction using machine learning. The techniques used were Artificial Neural Network (ANN), Logistic Regression, Decision Tree, Support Vector Machine (SVM). As a result of

prediction and comparison among the models ANN performed well compared to other models. The primary aim of using ANN was its ability to handle large data, inconsistencies, and the ability to yield high accuracy. Accuracy yielded is 87%.

Gowtham Sethupathi et.al [5] proposed efficient rainfall prediction and analysis using Machine Learning Techniques. The proposed methods are Random Forest and Logistic Regression. Among the models after testing the accuracy of Logistic regression is said to perform well with an accuracy of 95.9%.

K. Nanda Kumar et.al [6] proposed Rainfall Accuracy Prediction using Machine Learning Technique based on Linear Regression over Logistic Regression. This paper describes mainly 2 machine learning techniques Logistic regression and Linear regression. Sample size is calculated using G power software and determined as 10 per group with pretest power 80%, threshold 0.05% and CI 95%. Linear Regression provides a higher of 91.18% compared to Logistic Regression algorithm with 87.05% in predicting rainfall.

Nikhil Oswal [7] proposed rainfall prediction using machine learning techniques**.** This paper we propose the techniques such as Logistic Regression, Decision Tree, Random Forest, K-Nearest Neighbors, Decision Table, Ada Boost, Gradient Boosting. The models were tested by splitting the dataset into train and test set and performed oversampling, under sampling and testing the original data. The results obtained were testing under the original data yielded random forest as the best method with an accuracy of 88.21%. By oversampling the data decision tree proves to be the best method with an accuracy of 91.00% and Logistic Regression performed well with under sampling the data.

Nisha Thakur et.al [8] proposed Rainfall Forecasting using various Artificial Neural Network Techniques-A Review. Various algorithms such as MLP (Multi-Layer Perceptron), BPN (Backpropagation Network), ARIMA (Auto Regressive integrated moving average), MT (Model Tree), ANFIS

(Artificial Neural Fuzzy Inference System), WNN (Wavelet Neural network) and SVR (Support Vector Regression) were used for prediction .Among these methods the best correlation coefficient for training and testing was 0.99 & 0.96 respectively which were found using the Backpropagation neural network.

Prem Kumar. B et.al [9] proposed Performance Analysis and Evaluation of Machine Learning Algorithms in Rainfall Prediction. This paper encompasses six machine learning algorithms such as Lasso regression, ridge regression, elastic net regression, random forest, gradient boosting, and decision tree regressor. The models are trained by performing exploratory data analysis, cleaning then splitting the dataset into trainand test in the ratio of 70:30 and 80:20 and then test the model based onevaluation metrics and finally predicting the target variable. Those models performances have been calculated through the evaluation metrics such as R^2 score, Mean Absolute Error (MAE), Mean Square Error (MSE), and Root Mean Square Error (RMSE). As a result, Lasso regression of the linear model is thebest model among six ML models. Lasso model given more R squared score is 99.21%, MAE is 13.68, MSE is 6432.41 and RMSE is 80.20 at 80 % train dataset and 20% at test dataset.

R. Arya et.al [10] proposed prediction of rainfall using an optimized Genetic-Artificial Neural Network Model. This paper describes Feed Forward - Back Propagation (FFBP) algorithm to train the networks, and to get optimized results Genetic Algorithm (GA) is used. This paper aims to analyse the four-months (June, July, August, and September) of rainfall data of 30 years from 1982-2012 in Goa, India. After, the analysis of training, validating, and testing the data Genetic Algorithm yielded a better accuracy of 98.78% and performed well under testing data.

R. Kingsy Grace et.al [11] proposed Machine Learning based Rainfall prediction using Multiple Linear Regression (MLR) for Indian Dataset**.** The dataset is split into train and test data in the ratio of 70:30 for

training the model. The parameters namely Mean Squared Error, Root Mean Squared Error and Accuracy are used for evaluation. Accuracy obtained for MLR is 99%.

R. Senthil Kumar et al.[12] proposed rainfall prediction using Machine Learning and Deep Learning processes such as Naïve-Bayes, K-Nearest Neighbors, Neural network and fuzzy logic were used. Various data mining techniques were used to predict rainfall based on parameters such as humidity, temperature, and wind gust. This paper provides a comparison based on authors, data mining techniques algorithms, attributes, time period, dataset size, accuracy percentage. MATLAB software tool is used. Decision tree comes out to be the best algorithm other than deep learning process. Accuracy yielded is 99%.

S. Sakthivel et.al [13] proposed an Effective Procedure to predict rainfall using Hybrid Machine Learning Techniques. In this paper, a novel dataset is utilized from Regional Meteorological Centre Chennai to predict the rainfall summary. This proposed approach of INNM analyze the rainfall prediction scenario based on two different machine learning logics such as Back Propagation Neural Network and the Rapid Miner. The general machine learning algorithms train the machine with respect to the dataset features and predict the result based on testing input. In this approach two different variants ofmachine learning principles are utilized to classify the resulting nature with better accuracy levels and cross validations are providing best probabilistic results in outcome. The proposed approach of INNM assures the resulting accuracy levels around 96.5% in prediction with lowest error ratio of 0.04%.

Urooj Kaimkhani et.al [14] proposed Rainfall Prediction Using Time Series Nonlinear Autoregressive Neural Network. This research work reflects the prediction of the metrological parameterin the time series, i.e., prediction of Rainfall by using ANN (Artificial Neural Network) based model NARX (Nonlinear Autoregressive with exogenous input). In this research paper, more than a few ANN models that rely on real-time sequence recurring NARX-based

ANN techniques are initiated, trained, and tested with different parameter settings to find the network model's best possible output to its most-wanted prediction function. The model network's performance is evaluated based on the Mean Squared Error (MSE) performance of the model when the data sets are trained, validated, and tested. Although one step ahead of prediction, multi-phase ahead prediction is more complicated and difficult to carry out because of its underlying added complication. It is found that NARX is the best algorithm which yielded 0.9947 near to 1 which is the closest approximation of results.

V.P. Tharun et.al [15] proposed a system that predict rainfall in Coonoor in Nilgiris District, Tamil Nādu. This paper provides prediction models and implemented them using statistical modeling and regression techniques such as Support Vector Regression (SVR), Decision Tree, Random Forest. Initially data is collected then it is preprocessed for development of model. The evaluation metrics used to evaluate the performance are R-squared and Adjusted R-square. Thus, Random Forest performance is best among all methods used with 0.981 R-square value and 0.980 Adjusted R-square value.

Vikas Kumar et.al [16] proposed Rainfall Prediction using Machine Learning. This research was done using techniques of Logistic Regression, Random Forest, Decision Tree, K-Nearest Neighbor on the dataset. For the experimental purpose we have given the actual real time values of maximum and minimum temperature, relative humidity, wind speed etc. Dataset was segregated into training and testing data and after those models were trained and the accuracy score was noted andanalyzed before final prediction. Random Forest performs the best with an accuracy of 88.21% and a precision of 0.844%.

CHAPTER 3

SYSTEM DESIGN

In this chapter, the various UML diagrams for the Rainfall Prediction using Machine Learning is represented and the various functionalities are explained.

3.1 UNIFIED MODELING LANGUAGE

Unified Modeling language (UML) is a standardized modeling language enabling developers to specify, visualize, construct and document artifacts of a software system. Thus, UML makes these artifacts scalable, secure and robust in execution. It uses graphic notation to create visual models of software systems. UML is designed to enable users to develop an expressive, ready to use visual modeling language. In addition, it supports high-level development concepts such as frameworks, patterns and collaborations. Some of the UML diagrams are discussed.

3.1.1 Use Case Diagram of Rainfall Prediction

Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analyzed the functionalities are captured in use cases. So it can be said that uses cases are nothing but the system functionalities written in an organized manner. Now the second things which are relevant to the use cases are the actors.

Actors can be defined as something that interacts with the system. The actors can be human user, some internal applications or may be some external applications.

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements.

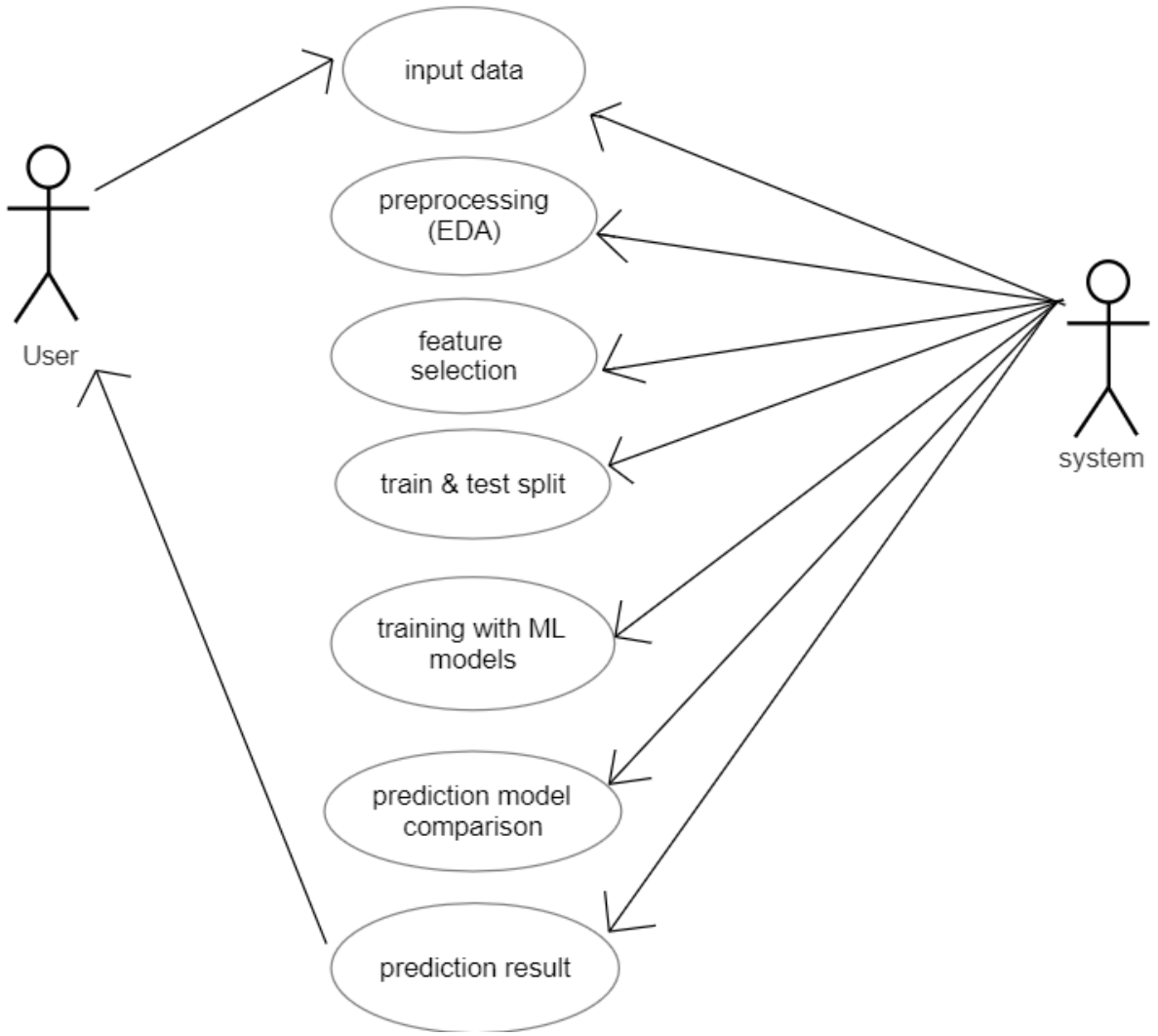


Fig 3.1 Use case diagram of Rainfall Prediction

Figure 3.1 shows that how the whole process is represented in form of an use case diagram. Initially, the user gives the input dataset to the system. Then, the system preprocess, perform feature selection, split the data set and train the data with different models. The system performs prediction, compares the models, and displays the result to the user.

3.1.2 Class Diagram of Rainfall Prediction

Figure 3.2 shows that class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. So a collection of class diagrams represent the whole system. The name of the class diagram should be meaningful to describe the aspect of the system.

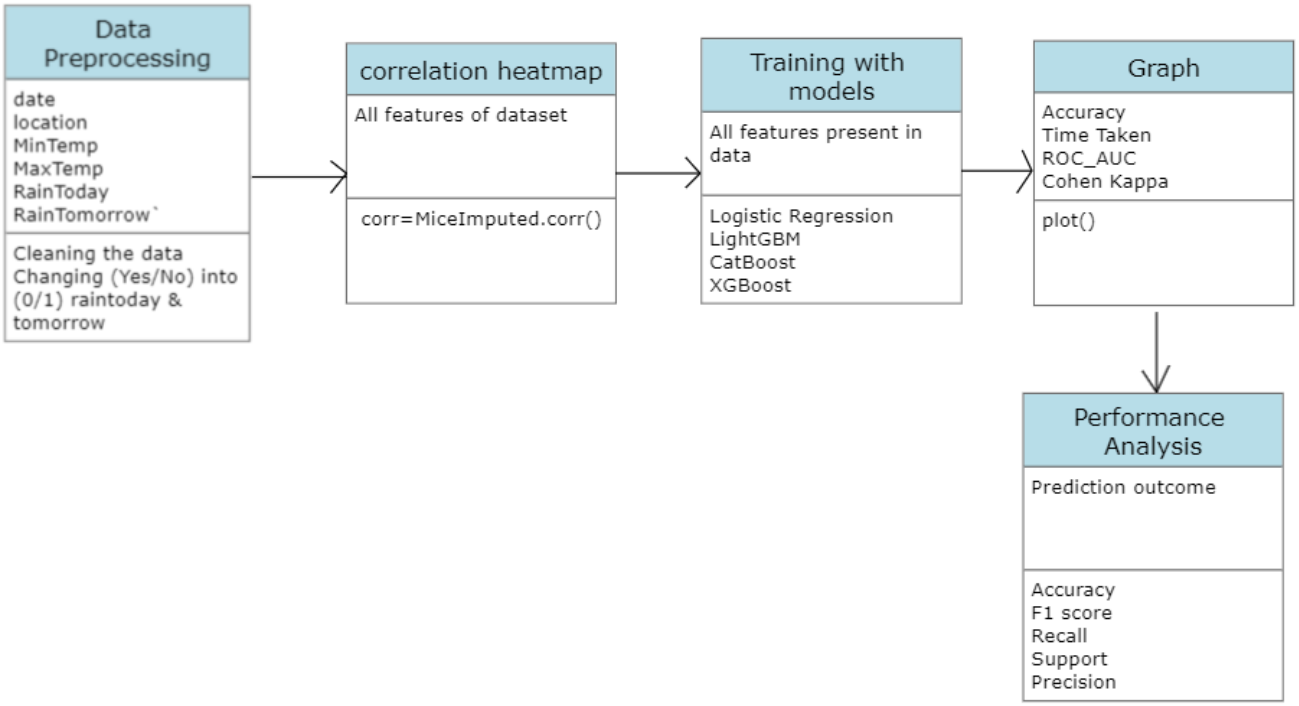


Fig 3.2 Class Diagram of Rainfall Prediction

Each element and their relationships should be identified in advance responsibility (attributes and methods)of each class should be clearly identified. The text present in the upper layer depicts the class name, the middle layer determines the attributes, and the lower layer contains the methods. The methods demonstrate how a class interacts with the data. The above diagram depicts the processes right from preprocessing ,finding correlation between the attributes of dataset and training with the ML models to obtain the result. Finally, the models are analyzed based on certain metrics that are depicted in the last layer of the last class.

3.1.3 Sequence Diagram of Rainfall Prediction

Figure 3.3 shows that UML sequence diagrams model the flow of logic within the system in a visual manner, enabling to both document and validate the logic, and are commonly used for both analysis and design purposes.

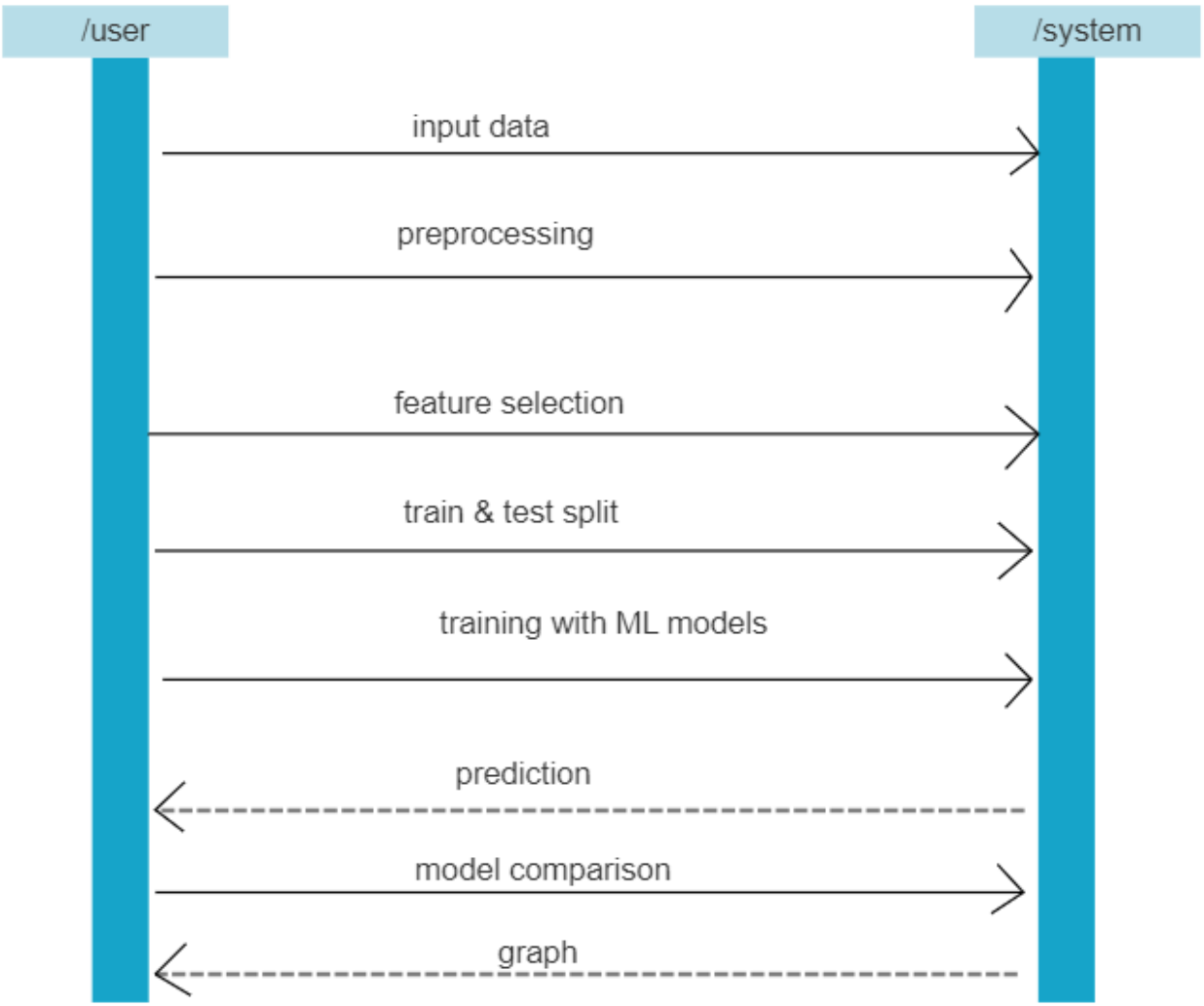


Fig 3.3 Sequence diagram of fake news detection

The various actions that take place in the application in the correct sequence are shown in Figure 3.3 Sequence diagrams are the most popular UML for dynamic modeling. The above diagram shows how user interacts with the system. The normal line represents the request from user then , the dotted lines represent the result from the system side.

3.1.4 Activity Diagram of Rainfall Prediction

Figure 3.4 shows that activity is a particular operation of the system. Activity diagram is suitable for modeling the activity flow of the system.

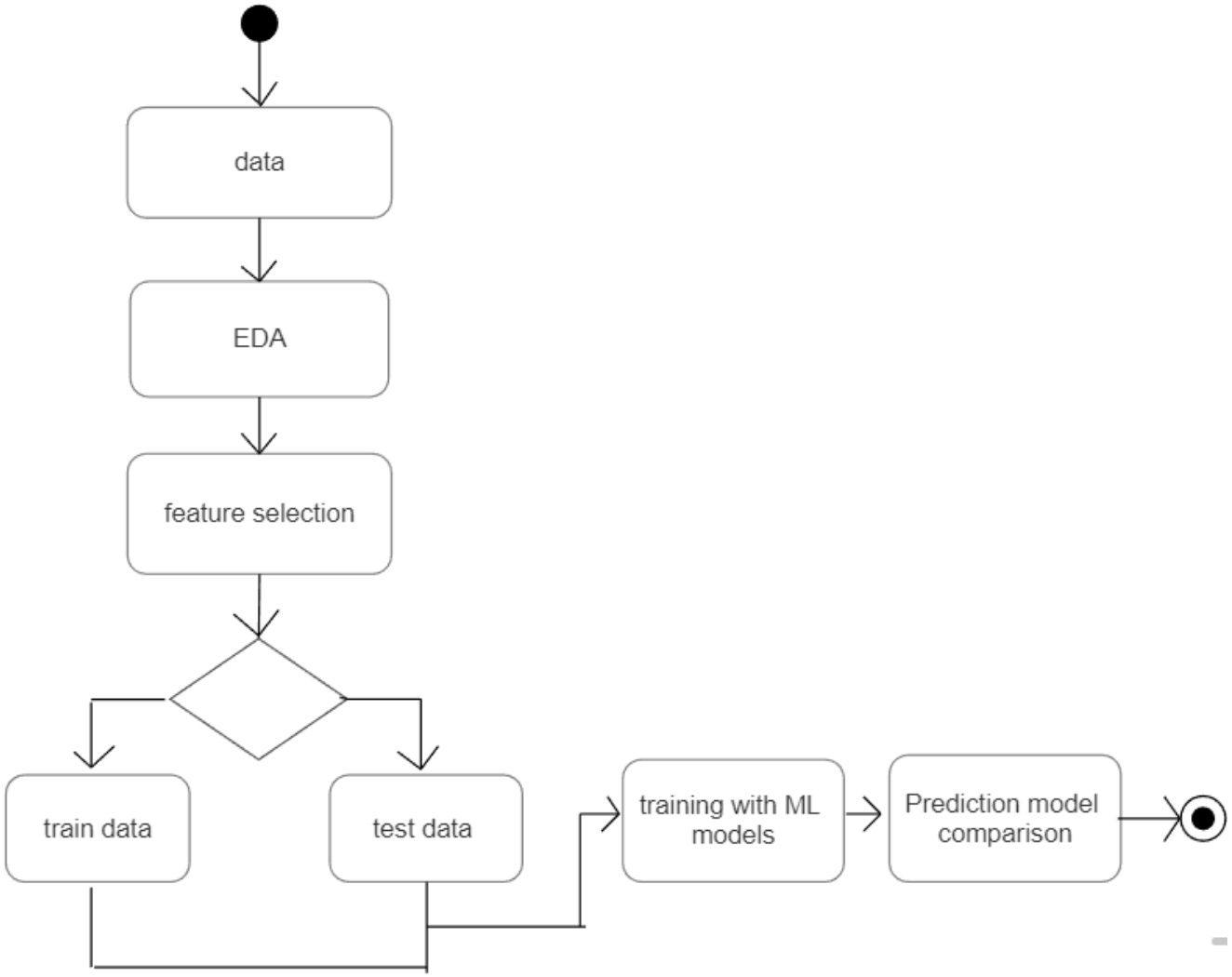


Fig 3.4 Activity Diagram of Rainfall Prediction

The above activity diagram indicates the flow of process right from collecting dataset from the user and how the system processes the data before using it, performs feature selection then splits them and perform training with different models and finally displays the result according to the user input.

CHAPTER 4
SYSTEM ARCHITECTURE

In this chapter, the System Architecture for the Fake News Detection using Machine Learning is represented and the modules are explained.

4.1 SYSTEM ARCHITECTURE DIAGRAM

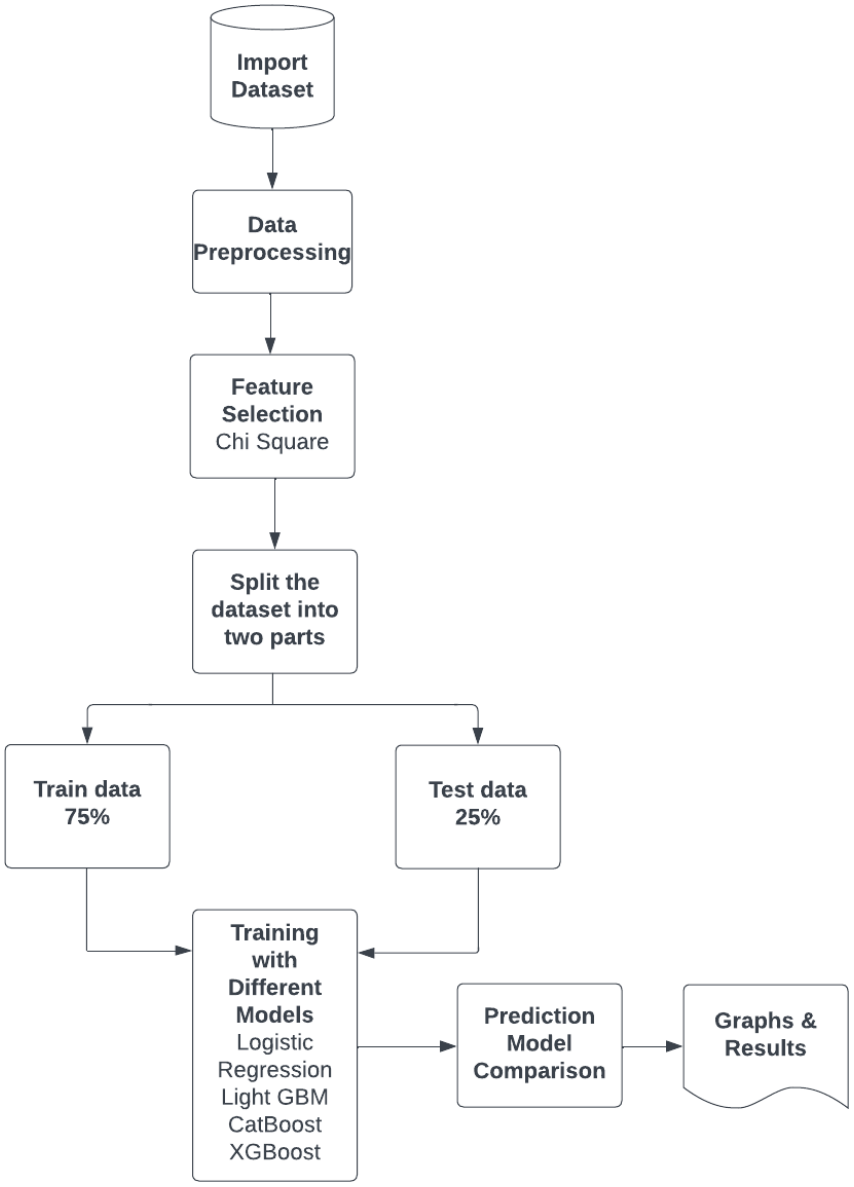


Fig 4.1 System Architecture Diagram

4.2 ARCHITECTURE DESCRIPTION

System architecture of our project provides the detailed description about the datasets, pre-processing of data, normalization, and feature selection, splitting the dataset into train and test set, training of data using various models, comparing the models based on the evaluation metrics. The dataset we used in this study are open source and freely available in Kaggle spanning from 2008-2018. The dataset consists of various climatic attributes such as date, location, MinTemp, MaxTemp, rainfall, evaporation, sunshine, windgustdirection, windgustspeed, winddirection9am, winddirection3pm, windspeed 9am, windspeed 3pm, humidity 9am, humidity 3pm, pressure 9am, pressure 3pm, cloud 9am, cloud 3pm, temp 9am, temp 3pm, raintoday, raintomorrow.

The dataset contains a total of 145460 entries. There are 23 columns present in the dataset. The basic libraries used in the prediction are Numpy, pandas, seaborn, matplot, sklearn, resample, LabelEncoder, preprocessing, SelectKBest, chi2, train_test_split, StandardScaler, time, accuracy_score, roc_auc_score, cohen_kappa_score, plot_confusion_matrix, roc_curve, classification_report, LogisticRegression, lgb, cb, xgb.

In this proposed system, we cannot use data directly because in machine learning it is necessary to clean the data before using it. This is done to reduce error So, we have to always first clean the data. In this project, we have changed the raintoday and raintomorrow factors from yes or no into binary values 0/1. In this project, we are using algorithms such as Logistic Regression, LightGBM, CatBoost and XGBoost.

Performance analysis is done between the models and then graphs are plotted based on certain parameters and displayed to the user.

CHAPTER 5

SYSTEM IMPLEMENTATION

In this chapter, the System Implementation for the Rainfall Prediction using Machine Learning is explained in detail.

5.1 IMPLEMENTATION OF RAINFALL PREDICTION USING MACHINE LEARNING

The project is implemented in Google Colaboratory. Here, the various functionalities required for the application are implemented by coding them in Python.

-Dataset source – Kaggle

Data-(Date, Location, MinTemp, MaxTemp, Rainfall, Evaporation, Sunshine, WindGustDir, WindGustSpeed, WindDir9am, WindDir3pm, WindSpeed9am, WindSpeed3pm, Humidity9am, Humidity3pm, Pressure9am, Pressure3pm, Cloud9am, Cloud3pm, Temp9am, Temp3pm, RainToday, RainTomorrow)

5.2 MODULES

5.2.1 Dataset

A dataset is a structured collection of data generally associated with a unique body of work. A database is an organized collection of data stored as multiple datasets. The datasets we used in this study are open source and freely available in Kaggle. The data includes various attributes of weather such as rain, location, windspeed and so on. The dataset contains a total of 145460 entries.

5.2.2 Libraries

In order to perform this classification, you need the basic Data Scientist starter pack (sklearn, pandas, NumPy, matplotlib, seaborn) plus some specific libraries like resample, LabelEncoder, preprocessing, SelectKBest, chi2,train_test_split,StandardScaler,time,accuracy_score, roc_auc_score, cohen_kappa_score, plot_confusion_matrix, roc_curve, classification_report,LogisticRegression, lgb, cb, xgb.

5.2.3 Data Cleaning and Preparation

Data cleaning refers to identifying and correcting errors in the dataset thatmay negatively impact a predictive model. Data cleaning is used to refer to all kinds of tasks and activities to detect and repair errors in the data. In this project,we can't use data directly because it is necessary to clean the data before using it.

If we use it directly without cleaning, then it is very hard for the ML algorithm to detect patterns in that data and sometimes it will also generate an error. In this project, we have converted the RainToday and RainTomorrow objects (Yes/No) into binary values (0/1).

5.2.4 Data Exploration

Data exploration, also known as exploratory data analysis (EDA), is a process where users look at and understand their data with statistical and visualization methods. This step helps identifying patterns and problems in the dataset, as well as deciding which model or algorithm to use in subsequent steps.In this project, we have explored that the dataset is balanced or not. If not balanced, then oversampling the dataset to balance it. Then, exploring whether there are any missing values if there are any missing values then imputing them with Multiple Imputation by Chained Equation (MICE) and then detecting any outliers are present or not. If they are present, then removing the outliers. After the removal of outliers correlation heatmap is plotted to find out the correlation among the features.

5.2.5. Feature Selection

Feature selection is the process of reducing the number of input variables when developing a predictive model. It is desirable to reduce the number of input variables to both reduce the computational cost of modeling and, in some cases, to improve the performance of the model. First normalizing the data, we use MinMaxScaler instead of StandardScaler to avoid negative values and then using Filter method for feature selection.

5.2.6 Training

Model training in machine language is the process of feeding an ML algorithm with data to help identify and learn good values for all attributes involved. There are several types of machine learning models, of which the mostcommon ones are supervised and unsupervised learning. In this project, we are using supervised learning algorithms such as Logistic Regression. Also, LightGBM, CatBoost and XGBoost are used.

5.2.7 Algorithm

The algorithm used in our project is Logistic Regression, Decision Tree and Random Forest.

a. Logistic Regression

Step 1: Converting the text data into a machine-readable form.

Step 2: Fitting Logistic Regression to the Training set

Step 3: Calculate df(t) = occurrence of t in documents

Step 4: Calculate idf(t) = log [n / df(t)] + 1

Step 5: Calculate tf-idf(t, d) = tf(t, d) * idf(t)

Step 6: Target variable (or output), y, can take discrete values

for given set of features (or inputs), X

LR(z)=1/1+e^z

where, z-threshold value

Step 7:Test accuracy of the result(Creation of Confusion matrix)

Step 8:Visualizing the test set result.

b. LightGBM

Step1: Fit LightGBM into Training set.

Step 2: Combine features that are mutually exclusive of training databy exclusive feature bundling (EFB) technique.

Step 3: Set $\theta_0(x)= \arg\min_c \sum^{N}_{i} L(y_i,c)$;

Step 4: For m=1 to M do

Calculate gradient absolute values

Step 5: Resample data using Gradient-based one side sampling (GOSS) process.

topN= a x len(D); randN=bxlen(D);

sorted=GetSortedIndices(abs(r));

A=sorted[1:topN]; B=RandomPick(sorted[topN:len(D)],randN);

D`=A+B;

Step 6: Calculate information gains.

$$V_{j|O}(d) = \frac{1}{n_o}\left(\frac{\left(\sum_{\{x_i \in O: x_{ij} \le d\}} g_i\right)^2}{n^j_{l|O}(d)} + \frac{\left(\sum_{\{x_i \in O: x_{ij} > d\}} g_i\right)^2}{n^j_{r|O}(d)}\right)$$

where $n_O = \sum I[xi \in O]$, $n^j_{l|O}(d) = \sum I[xi \in O : xij \le d]$ and $n^j_{r|O}(d) = \sum I[xi \in O : xij > d]$.

Step 7: Develop a new decision tree $q_m(X)$`on set D`

Step 8: Update $q_m(X)=q_{(m-1)}(X)+q_m(X)$

Step 9: End for

Step 10: Return $q^{\sim}(X)=q_M$

c. CatBoost

Step 1: pip install catboost

Step 2: Import cb library.

Step 3: Set up the data for the classifier .

Step 4: Calculate the minimized function to avoid loss of information.

$$h^t = \underset{h \epsilon H}{arg\ min}\ \mathbb{E}(\frac{\delta \mathcal{L} y}{\delta F^{t-1}} - h)^2 \approx \underset{h \epsilon H}{arg\ min}\ \frac{1}{n}(\frac{\delta \mathcal{L} y}{\delta F^{t-1}} - h)^2$$

Step 5: Set up the model.

d. XGBoost

Step 1: Build a sequence from the function gradient.

Step 2: Calculate the objective function.

$$obj^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \Omega(f_i)$$
$$= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) + constant$$
$$obj^{(t)} = \sum_{i=1}^{n} (y_i - (\hat{y}_i^{(t-1)} + f_t(x_i)))^2 + \sum_{i=1}^{t} \Omega(f_i)$$
$$= \sum_{i=1}^{n} [2(\hat{y}_i^{(t-1)} - y_i) f_t(x_i) + f_t(x_i)^2] + \Omega(f_t) + constant$$

Step 3: Continue until decision tree is obtained.

5.2.8 Classification

In machine learning, classification refers to a predictive modeling problem where a class label is predicted for a given example of input data.

To check how well our model, we use some metrics to find the accuracy of our model. There are many types of classification metrics available in Scikit learn: Accuracy Score, Precision, Recall, F1-Score, Support, AUC-ROC, Cohen's Kappa.

Accuracy = TP+TN/TP+FP+FN+TN.

Precision=True Positive/True Positive+False Positive

Recall=True Positive/True Positive+False Negative

F1=2*Precision*Recall/Precision+Recall

Cohen's Kappa

P_e=P(correct)+P(incorrect)

$K=P_0-P_e/1-P_e$

Accuracy: is the ratio of number of correct predictions to the total number of input samples. It works well only if there are equal number of samples belonging to each class.

Precision: is the number of correct positive results divided by the number of positive results predicted by the classifier.

Recall: is the number of correct positive results divided by the number of all relevant samples (all samples that should have been identified as positive)

F1 Score: is the Harmonic Mean between precision and recall. The range for F1 Score is [0, 1]. The greater the F1 Score, the better is the performance of our model

Support : Support may be defined as the number of samples of the true response that lies in each class of target values.

AUC-ROC: metric will tell us about the capability ofmodel in distinguishing the classes. Higher the AUC, better the model.

Cohen's Kappa: is a quantitative measure of reliability for two raters that are rating the same thing, corrected for how often that the raters may agree by chance.

CHAPTER 6

CODING AND SCREENSHOTS

6.1 Sample Code

```
from google.colab import files
upload=files.upload()
import pandas as pd
data=pd.read_csv('rainfall.csv')
data.head()
data.shape
data.info()
data.describe()
# Converting RainToday and RainTomorrow (Yes/No) into binary values (0/1)
data['RainToday'].replace({'No': 0, 'Yes': 1},inplace = True)
data['RainTomorrow'].replace({'No': 0, 'Yes': 1},inplace = True)
# Plotting a barplot of RainToday and RainTomorrow
import matplotlib.pyplot as plt
fig=plt.figure(figsize=(3,5))
data.RainTomorrow.value_counts(normalize = True).plot(kind='bar', color= ['skyblue','pink'],
alpha = 0.9, rot=0)
plt.title('Rain Tomorrow indicator No(0) Yes(1) in the Imbalanced dataset')
plt.show()
# Handling Class Imbalance
from sklearn.utils import resample
no=data[data.RainTomorrow==0]
yes=data[data.RainTomorrow==1]
```

```
 yes_oversampled=resample(yes,replace=True,n_samples=len(no),random_state=123)
oversampled=pd.concat([no,yes_oversampled])
fig=plt.figure(figsize=(8,5))
oversampled.RainTomorrow.value_counts(normalize=True).plot(kind='bar',color=['skyblue',
'pink'],alpha=0.9,rot=1)
plt.title('RainTomorrow indocator No(0) and Yes(1) after Oversampling (Balanced Datase)')
plt.show()
# Plotting missing data patterns
import seaborn as sns
sns.heatmap(oversampled.isnull(), cbar=False, cmap='PuBu')
# Checking details of missing values
 total = oversampled.isnull().sum().sort_values(ascending=False)
 percent = (oversampled.isnull().sum()/oversampled.isnull().count()).sort_values(ascending=
 False)
 missing = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
 missing.head(4)
  oversampled.select_dtypes(include=['object']).columns
# Import categorical var with data
oversampled['Date'] = oversampled['Date'].fillna(oversampled['Date'].mode()[0])
oversampled['Location'] = oversampled['Location'].fillna(oversampled['Location'].mode()[0])
oversampled['WindGustDir'] = oversampled['WindGustDir'].fillna(oversampled['WindGustDir
'].mode()[0])
oversampled['WindDir9am'] = oversampled['WindDir9am'].fillna(oversampled['WindDir9am']
.mode()[0])
oversampled['WindDir3pm'] = oversampled['WindDir3pm'].fillna(oversampled['WindDir3pm']
.mode()[0])
# Converting categorical features to continuous features with label encoder
from sklearn.preprocessing import LabelEncoder
lencoders = {}
```

```
  for col in oversampled.select_dtypes(include=['object']).columns:
    lencoders[col] = LabelEncoder()
    oversampled[col] = lencoders[col].fit_transform(oversampled[col])


  import warnings
  warnings.filterwarnings("ignore")
  # Multiple Imputation by Chained Equations
  from sklearn.experimental import enable_iterative_imputer
  from sklearn.impute import IterativeImputer
  MiceImputed = oversampled.copy(deep=True)
  mice_imputer = IterativeImputer()
  MiceImputed.iloc[:, :] = mice_imputer.fit_transform(oversampled)
# Detecting Outliers with IQR
   Q1=MiceImputed.quantile(0.25)
   Q3=MiceImputed.quantile(0.75)
   IQR=Q3-Q1
   print(IQR)
# Removing outliers from the dataset
   MiceImputed = MiceImputed[~((MiceImputed < (Q1 1.5 * IQR)) |(MiceImputed > (Q3 +
   1.5 * IQR))).any(axis=1)]
   MiceImputed.shape
# Correlation Heatmap
   import numpy as np
   import matplotlib.pyplot as plt
   import seaborn as sns
   corr=MiceImputed.corr()
   mask=np.triu(np.ones_like(corr,dtype=np.bool))
   f,ax=plt.subplots(figsize=(20,20))
```

```
    cmap=sns.diverging_palette(250,25,as_cmap=True)
    sns.heatmap(corr,mask=mask,cmap=cmap,vmax=None,center=0,square=True,annot=True,
    lineWidth=.5,cbar_kws={"shrink": .9})
# Standardizing data
    from sklearn import preprocessing
    r_scaler=preprocessing.MinMaxScaler()
    r_scaler.fit(MiceImputed)
   modified_data=pd.DataFrame(r_scaler.transform(MiceImputed),index=MiceImputed.index,
    columns=MiceImputed.columns)
# Feature Importance using Filter method
    from sklearn.feature_selection import SelectKBest,chi2
    X=modified_data.iloc[:,modified_data.columns!='RainTomorrow']
    y=modified_data[['RainTomorrow']]
    selector=SelectKBest(chi2,k=10)
    selector.fit(X,y)
    X_new=selector.transform(X)
    print(X.columns[selector.get_support(indices=True)])
# Training rainfall prediction with different models
   features = MiceImputed[['Location', 'MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshin
   e', 'WindGustDir', 'WindGustSpeed', 'WindDir9am', 'WindDir3pm', 'WindSpeed9am', 'WindSp
   eed3pm', 'Humidity9am',  'Humidity3pm', 'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud
   3pm', 'Temp9am', 'Temp3pm', 'RainToday']]
   target = MiceImputed['RainTomorrow']
# Split into train and test set
   from sklearn.model_selection import train_test_split
   X_train,X_test,y_train,y_test=train_test_split(features,target,test_size=0.25,random_state=12
   345)
# Normalize Features
    from sklearn.preprocessing import StandardScaler
```

```
scaler=StandardScaler()
X_train=scaler.fit_transform(X_train)
X_test=scaler.fit_transform(X_test)

 def plot_roc_cur(fper, tper):
 plt.plot(fper, tper, color='orange', label='ROC')
 plt.plot([0, 1], [0, 1], color='darkblue', linestyle='--')
 plt.xlabel('False Positive Rate')
 plt.ylabel('True Positive Rate')
 plt.title('Receiver Operating Characteristic (ROC) Curve')
 plt.legend()
 plt.show()
import time
from sklearn.metrics import accuracy_score, roc_auc_score, cohen_kappa_score, plot_confu
sion_matrix, roc_curve, classification_report
def run_model(model, X_train, y_train, X_test, y_test, verbose=True):
   t0=time.time()
   if verbose == False:
       model.fit(X_train,y_train, verbose=0)
   else:
       model.fit(X_train,y_train)
  y_pred = model.predict(X_test)
  accuracy = accuracy_score(y_test, y_pred)
  roc_auc = roc_auc_score(y_test, y_pred)
  coh_kap = cohen_kappa_score(y_test, y_pred)
  time_taken = time.time()-t0
  print("Accuracy = {}".format(accuracy))
  print("ROC Area under Curve = {}".format(roc_auc))
  print("Cohen's Kappa = {}".format(coh_kap))
```

```
    print("Time taken = {}".format(time_taken))
    print(classification_report(y_test,y_pred,digits=5))
    probs = model.predict_proba(X_test)
    probs = probs[:, 1]
    fper, tper, thresholds = roc_curve(y_test, probs)
    plot_roc_cur(fper, tper)
    plot_confusion_matrix(model, X_test, y_test,cmap=plt.cm.Blues, normalize = 'all')
    return model, accuracy, roc_auc, coh_kap, time_taken
# Logistic Regression
  from sklearn.linear_model import LogisticRegression
  params_lr = {'penalty': 'l1', 'solver':'liblinear'}
  model_lr = LogisticRegression(**params_lr)
  model_lr, accuracy_lr, roc_auc_lr, coh_kap_lr, tt_lr = run_model(model_lr, X_train, y_train,
  X_test, y_test)
# LightGBM
  import lightgbm as lgb
  params_lgb ={'colsample_bytree': 0.95,
       'max_depth': 16,
       'min_split_gain': 0.1,
       'n_estimators': 200,
       'num_leaves': 50,
       'reg_alpha': 1.2,
       'reg_lambda': 1.2,
       'subsample': 0.95,
       'subsample_freq': 20
       }
  model_lgb = lgb.LGBMClassifier(**params_lgb)

# CatBoost
```

```
!pip install catboost
import catboost as cb
params_cb ={'iterations': 50,
        'max_depth': 16}
model_cb = cb.CatBoostClassifier(**params_cb)
model_cb, accuracy_cb, roc_auc_cb, coh_kap_cb, tt_cb = run_model(model_cb, X_train,
y_train, X_test, y_test, verbose=False)
model_lgb, accuracy_lgb, roc_auc_lgb, coh_kap_lgb, tt_lgb = run_model(model_lgb,
X_train, y_train, X_test, y_test)
# XGBoost
import xgboost as xgb
params_xgb ={'n_estimators': 500,
        'max_depth': 16}
model_xgb = xgb.XGBClassifier(**params_xgb)
model_xgb, accuracy_xgb, roc_auc_xgb, coh_kap_xgb, tt_xgb = run_model(model_xgb,
X_train, y_train, X_test, y_test, verbose=False)
# Prediction Model Comparison based on Accuracy and Time taken
accuracy_scores = [accuracy_lr,accuracy_lgb, accuracy_cb, accuracy_xgb]
roc_auc_scores = [roc_auc_lr,roc_auc_lgb, roc_auc_cb, roc_auc_xgb]
coh_kap_scores = [coh_kap_lr,coh_kap_lgb, coh_kap_cb, coh_kap_xgb]
tt = [tt_lr,tt_lgb, tt_cb, tt_xgb]
model_data = {'Model': ['LogisticRegression','LightGBM','Catboost','XGBoost'],
        'Accuracy': accuracy_scores,
        'ROC_AUC': roc_auc_scores,
        'Cohen_Kappa': coh_kap_scores,
        'Time taken': tt}
data = pd.DataFrame(model_data)
fig, ax1 = plt.subplots(figsize=(12,10))
ax1.set_title('Model Comparison: Accuracy and Time taken for execution', fontsize=13)
```

```
    color = 'tab:green'
    ax1.set_xlabel('Model', fontsize=12)
    ax1.set_ylabel('Time taken', fontsize=10, color=color)
    ax2 = sns.barplot(x='Model', y='Time taken', data = data, palette='summer')
    ax1.tick_params(axis='y')
    ax2 = ax1.twinx()
    color = 'tab:red'
    ax2.set_ylabel('Accuracy', fontsize=13, color=color)
    ax2 = sns.lineplot(x='Model', y='Accuracy', data = data, sort=False, color=color)
    ax2.tick_params(axis='y', color=color)
# Prediction Model Comparison based on AUC_ROC and Cohen’s Kappa
  fig, ax3 = plt.subplots(figsize=(8,10))
  ax3.set_title('Model Comparison: Area under ROC and Cohens Kappa', fontsize=13)
  color = 'tab:blue'
  ax3.set_xlabel('Model', fontsize=13)
  ax3.set_ylabel('ROC_AUC', fontsize=13, color=color)
  ax4 = sns.barplot(x='Model', y='ROC_AUC', data = data, palette='winter')
  ax3.tick_params(axis='y')
  ax4 = ax3.twinx()
  color = 'tab:red'
  ax4.set_ylabel('Cohen_Kappa', fontsize=13, color=color)
  ax4 = sns.lineplot(x='Model', y='Cohen_Kappa', data = data, sort=False, color=color)
  ax4.tick_params(axis='y', color=color)
  plt.show()
```

6.2 Results

```
[ ] from google.colab import files
    upload=files.upload()

    Choose Files  No file chosen    Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
    Saving rainfall.csv to rainfall.csv

[ ] import pandas as pd
    data=pd.read_csv('rainfall.csv')
    data.head()
```

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	...	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	Clou
0	01/12/2008	Albury	13.4	22.9	0.6	NaN	NaN	W	44.0	W	...	71.0	22.0	1007.7	1007.1	8.0	
1	02/12/2008	Albury	7.4	25.1	0.0	NaN	NaN	WNW	44.0	NNW	...	44.0	25.0	1010.6	1007.8	NaN	
2	03/12/2008	Albury	12.9	25.7	0.0	NaN	NaN	WSW	46.0	W	...	38.0	30.0	1007.6	1008.7	NaN	
3	04/12/2008	Albury	9.2	28.0	0.0	NaN	NaN	NE	24.0	SE	...	45.0	16.0	1017.6	1012.8	NaN	
4	05/12/2008	Albury	17.5	32.3	1.0	NaN	NaN	W	41.0	ENE	...	82.0	33.0	1010.8	1006.0	7.0	

5 rows × 23 columns

```
[ ] data.shape

    (145460, 23)
```

Fig 6.2.1 Reading the Dataset

Read csv file using read_csv function and found the total rows and column of our rainfall csv file using shape.

```
data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 145460 entries, 0 to 145459
Data columns (total 23 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   Date           145460 non-null  object
 1   Location       145460 non-null  object
 2   MinTemp        143975 non-null  float64
 3   MaxTemp        144199 non-null  float64
 4   Rainfall       142199 non-null  float64
 5   Evaporation    82670 non-null   float64
 6   Sunshine       75625 non-null   float64
 7   WindGustDir    135134 non-null  object
 8   WindGustSpeed  135197 non-null  float64
 9   WindDir9am     134894 non-null  object
 10  WindDir3pm     141232 non-null  object
 11  WindSpeed9am   143693 non-null  float64
 12  WindSpeed3pm   142398 non-null  float64
 13  Humidity9am    142806 non-null  float64
 14  Humidity3pm    140953 non-null  float64
 15  Pressure9am    130395 non-null  float64
 16  Pressure3pm    130432 non-null  float64
 17  Cloud9am       89572 non-null   float64
 18  Cloud3pm       86102 non-null   float64
 19  Temp9am        143693 non-null  float64
 20  Temp3pm        141851 non-null  float64
 21  RainToday      142199 non-null  object
 22  RainTomorrow   142193 non-null  object
dtypes: float64(16), object(7)
memory usage: 25.5+ MB
```

Fig 6.2.2 Parameters of Dataset

BASIC EDA PROCESS

```
[ ] data.describe()
```

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Cl
count	143975.000000	144199.000000	142199.000000	82670.000000	75625.000000	135197.000000	143693.000000	142398.000000	142806.000000	140953.000000	130395.00000	130432.000000	89572
mean	12.194034	23.221348	2.360918	5.468232	7.611178	40.035230	14.043426	18.662657	68.880831	51.539116	1017.64994	1015.255889	4
std	6.398495	7.119049	8.478060	4.193704	3.785483	13.607062	8.915375	8.809800	19.029164	20.795902	7.10653	7.037414	2
min	-8.500000	-4.800000	0.000000	0.000000	0.000000	6.000000	0.000000	0.000000	0.000000	0.000000	980.50000	977.100000	0
25%	7.600000	17.900000	0.000000	2.600000	4.800000	31.000000	7.000000	13.000000	57.000000	37.000000	1012.90000	1010.400000	1
50%	12.000000	22.600000	0.000000	4.800000	8.400000	39.000000	13.000000	19.000000	70.000000	52.000000	1017.60000	1015.200000	5
75%	16.900000	28.200000	0.800000	7.400000	10.600000	48.000000	19.000000	24.000000	83.000000	66.000000	1022.40000	1020.000000	7
max	33.900000	48.100000	371.000000	145.000000	14.500000	135.000000	130.000000	87.000000	100.000000	100.000000	1041.00000	1039.600000	9

Converting raintoday and raintomorrow objects as binary.

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

Changing the RainToday and RainTomorrow (Yes/No) objects into binary values (0/1).

Plotting the graph after changing RainToday and RainTomorrow into binary values.

```
import matplotlib.pyplot as plt
fig=plt.figure(figsize=(3,5))
data.RainTomorrow.value_counts(normalize = True).plot(kind='bar', color= ['skyblue','pink'], alpha = 0.9, rot=0)
plt.title('Rain Tomorrow indicator No(0) Yes(1) in the Imbalanced dataset')
plt.show()
```

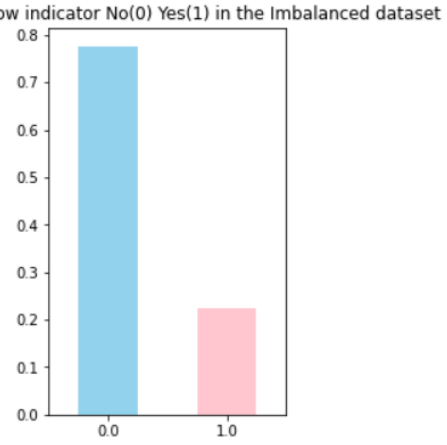


Fig 6.2.3 EDA Process

Handling class imbalance

```
from sklearn.utils import resample
no=data[data.RainTomorrow==0]
yes=data[data.RainTomorrow==1]
yes_oversampled=resample(yes,replace=True,n_samples=len(no),random_state=123)
oversampled=pd.concat([no,yes_oversampled])
fig=plt.figure(figsize=(8,5))
oversampled.RainTomorrow.value_counts(normalize=True).plot(kind='bar',color=['skyblue','pink'],alpha=0.9,rot=1)
plt.title('RainTomorrow indocator No(0) and Yes(1) after Oversampling (Balanced Datase)')
plt.show()
```

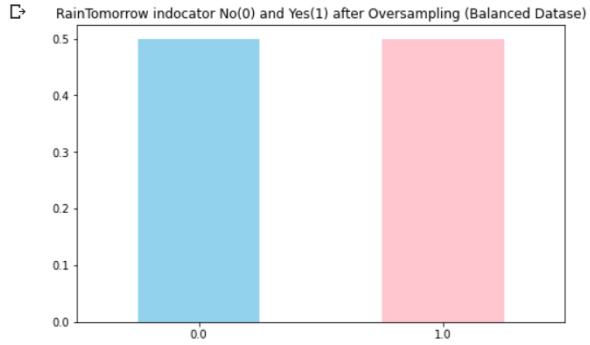


Fig 6.2.4 Handling Class Imbalance

Missing data pattern in training data

```
import seaborn as sns
sns.heatmap(oversampled.isnull(), cbar=False, cmap='PuBu')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1497e0f520>
```

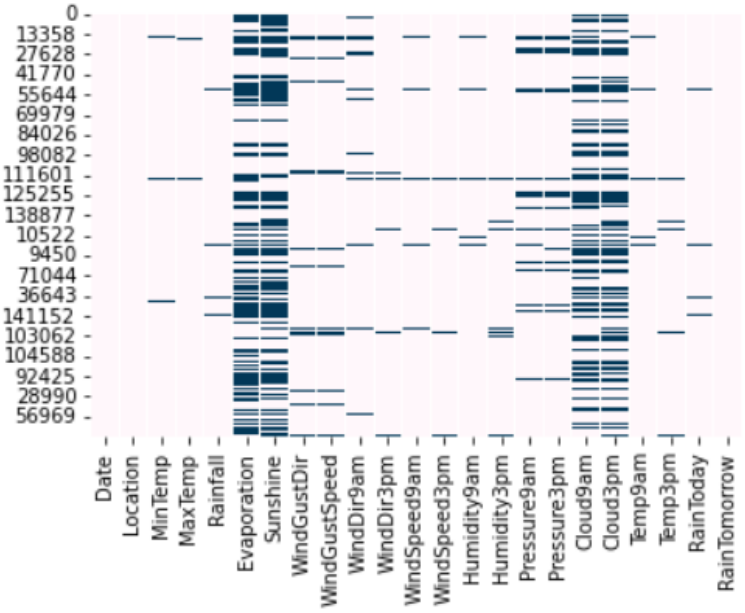


Fig 6.2.5 Missing Data Patterns

Checking details of missing data

```
[ ] total = oversampled.isnull().sum().sort_values(ascending=False)
    percent = (oversampled.isnull().sum()/oversampled.isnull().count()).sort_values(ascending=False)
    missing = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
    missing.head(4)
```

	Total	Percent
Sunshine	104831	0.475140
Evaporation	95411	0.432444
Cloud3pm	85614	0.388040
Cloud9am	81339	0.368664

```
oversampled.select_dtypes(include=['object']).columns
```

```
Index(['Date', 'Location', 'WindGustDir', 'WindDir9am', 'WindDir3pm'], dtype='object')
```

Fig 6.2.6 Percent of Missing values

Import categorical var with data

```
[ ] oversampled['Date'] = oversampled['Date'].fillna(oversampled['Date'].mode()[0])
    oversampled['Location'] = oversampled['Location'].fillna(oversampled['Location'].mode()[0])
    oversampled['WindGustDir'] = oversampled['WindGustDir'].fillna(oversampled['WindGustDir'].mode()[0])
    oversampled['WindDir9am'] = oversampled['WindDir9am'].fillna(oversampled['WindDir9am'].mode()[0])
    oversampled['WindDir3pm'] = oversampled['WindDir3pm'].fillna(oversampled['WindDir3pm'].mode()[0])
```

Convert categorical features to continuous features with label encoder

```
from sklearn.preprocessing import LabelEncoder
lencoders = {}
for col in oversampled.select_dtypes(include=['object']).columns:
    lencoders[col] = LabelEncoder()
    oversampled[col] = lencoders[col].fit_transform(oversampled[col])
```

```
[ ] import warnings
    warnings.filterwarnings("ignore")
    # Multiple Imputation by Chained Equations
    from sklearn.experimental import enable_iterative_imputer
    from sklearn.impute import IterativeImputer
    MiceImputed = oversampled.copy(deep=True)
    mice_imputer = IterativeImputer()
    MiceImputed.iloc[:, :] = mice_imputer.fit_transform(oversampled)
```

Detecting outliers with IQR

```
Q1=MiceImputed.quantile(0.25)
Q3=MiceImputed.quantile(0.75)
IQR=Q3-Q1
print(IQR)
```

```
Date             1712.000000
Location           25.000000
MinTemp             9.300000
MaxTemp            10.200000
Rainfall            2.400000
Evaporation         4.069582
Sunshine            5.983632
WindGustDir         9.000000
WindGustSpeed      19.000000
WindDir9am          8.000000
WindDir3pm          8.000000
WindSpeed9am       13.000000
WindSpeed3pm       11.000000
Humidity9am        26.000000
Humidity3pm        30.000000
Pressure9am         8.800000
Pressure3pm         8.800000
Cloud9am            4.000000
Cloud3pm            3.691648
Temp9am             9.300000
Temp3pm             9.800000
RainToday           1.000000
RainTomorrow        1.000000
dtype: float64
```

Fig 6.2.7 Detecting Outliers with IQR

Removing outliers from the dataset

```
MiceImputed = MiceImputed[~((MiceImputed < (Q1 - 1.5 * IQR)) |(MiceImputed > (Q3 + 1.5 * IQR))).any(axis=1)]
MiceImputed.shape
```

```
(170498, 23)
```

Correlation Heatmap

+ Code + Text

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
corr=MiceImputed.corr()
mask=np.triu(np.ones_like(corr,dtype=np.bool))
f,ax=plt.subplots(figsize=(20,20))
cmap=sns.diverging_palette(250,25,as_cmap=True)
sns.heatmap(corr,mask=mask,cmap=cmap,vmax=None,center=0,square=True,annot=True,lineWidth=.5,cbar_kws={"shrink": .9})
```

CORRELATION HEATMAP

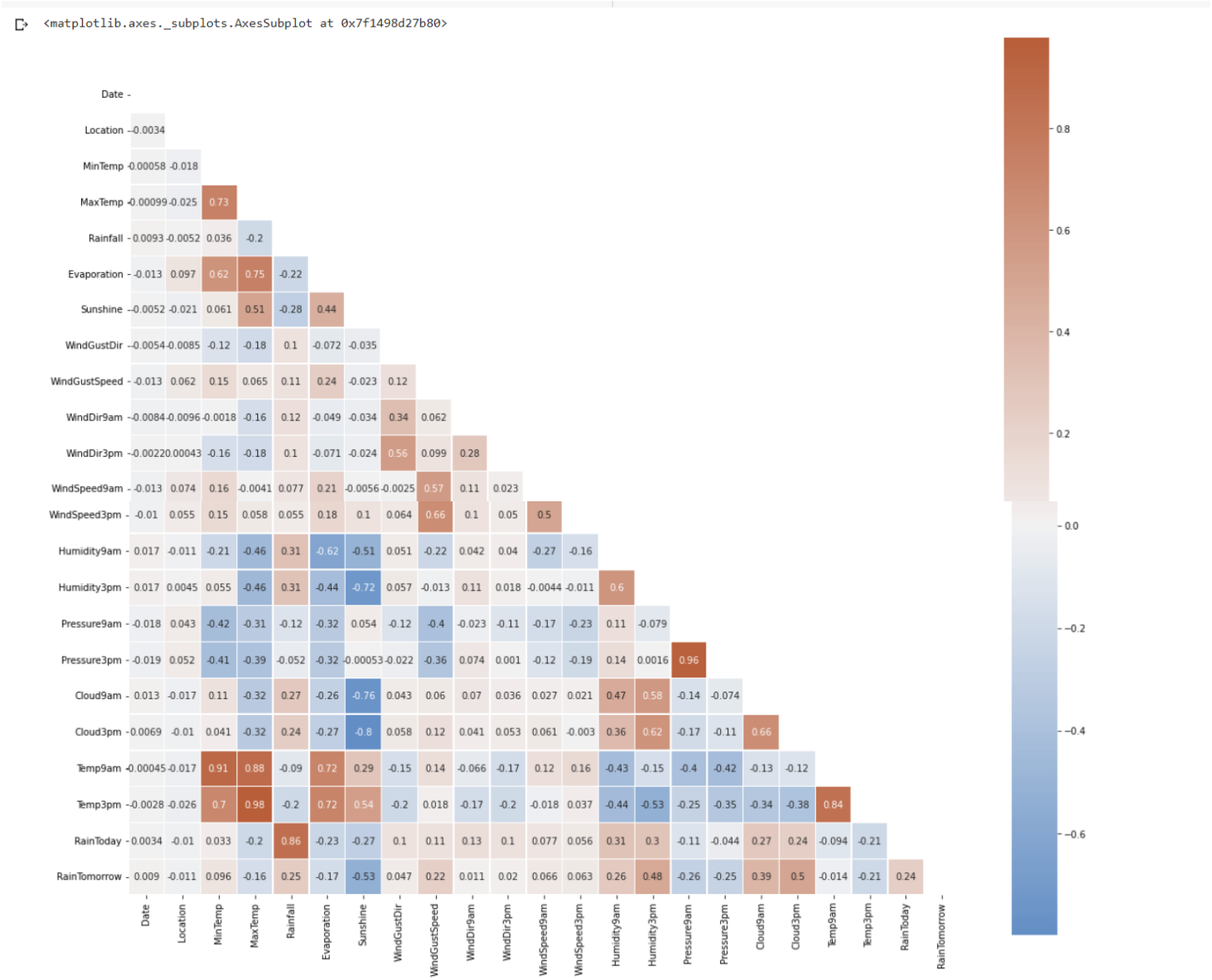


Fig 6.2.8 Correlation Heatmap

From the correlation heatmap it is clear that the(MaxTemp,MinTemp), (Pressure9h,Pressure3h),(Temp9am,3pm), (Evaporation, MaxTemp) and (MaxTemp,Temp3pm) have strong correlation between them.

FEATURE SELECTION

Standardizing data

```
[ ] from sklearn import preprocessing
    r_scaler=preprocessing.MinMaxScaler()
    r_scaler.fit(MiceImputed)
    modified_data=pd.DataFrame(r_scaler.transform(MiceImputed),index=MiceImputed.index,columns=MiceImputed.columns)
```

Feature Importance using Fiter method

```
from sklearn.feature_selection import SelectKBest,chi2
X=modified_data.iloc[:,modified_data.columns!='RainTomorrow']
y=modified_data[['RainTomorrow']]
selector=SelectKBest(chi2,k=10)
selector.fit(X,y)
X_new=selector.transform(X)
print(X.columns[selector.get_support(indices=True)])
```

```
Index(['Rainfall', 'Sunshine', 'WindGustSpeed', 'Humidity9am', 'Humidity3pm',
       'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'RainToday'],
      dtype='object')
```

Training rainfall prediction with different models

```
[ ] features = MiceImputed[['Location', 'MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine', 'WindGustDir',
                            'WindGustSpeed', 'WindDir9am', 'WindDir3pm', 'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am',
                            'Humidity3pm', 'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am', 'Temp3pm',
                            'RainToday']]
    target = MiceImputed['RainTomorrow']
```

Split into train and test set

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(features,target,test_size=0.25,random_state=12345)
```

Normalize Features

```
[ ] from sklearn.preprocessing import StandardScaler
    scaler=StandardScaler()
    X_train=scaler.fit_transform(X_train)
    X_test=scaler.fit_transform(X_test)
```

```
[ ] def plot_roc_cur(fper, tper):
        plt.plot(fper, tper, color='orange', label='ROC')
        plt.plot([0, 1], [0, 1], color='darkblue', linestyle='--')
        plt.xlabel('False Positive Rate')
        plt.ylabel('True Positive Rate')
        plt.title('Receiver Operating Characteristic (ROC) Curve')
        plt.legend()
        plt.show()
```

TRAINING WITH MODELS

```
import time
from sklearn.metrics import accuracy_score, roc_auc_score, cohen_kappa_score, plot_confusion_matrix, roc_curve, classification_report
def run_model(model, X_train, y_train, X_test, y_test, verbose=True):
    t0=time.time()
    if verbose == False:
        model.fit(X_train,y_train, verbose=0)
    else:
        model.fit(X_train,y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    roc_auc = roc_auc_score(y_test, y_pred)
    coh_kap = cohen_kappa_score(y_test, y_pred)
    time_taken = time.time()-t0
    print("Accuracy = {}".format(accuracy))
    print("ROC Area under Curve = {}".format(roc_auc))
    print("Cohen's Kappa = {}".format(coh_kap))
    print("Time taken = {}".format(time_taken))
    print(classification_report(y_test,y_pred,digits=5))

    probs = model.predict_proba(X_test)
    probs = probs[:, 1]
    fper, tper, thresholds = roc_curve(y_test, probs)
    plot_roc_cur(fper, tper)

    plot_confusion_matrix(model, X_test, y_test,cmap=plt.cm.Blues, normalize = 'all')

    return model, accuracy, roc_auc, coh_kap, time_taken
```

LOGISTIC REGRESSION

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
params_lr = {'penalty': 'l1', 'solver':'liblinear'}

model_lr = LogisticRegression(**params_lr)
model_lr, accuracy_lr, roc_auc_lr, coh_kap_lr, tt_lr = run_model(model_lr, X_train, y_train, X_test, y_test)
```

```
Accuracy = 0.7970909090909091
ROC Area under Curve = 0.7910899001499573
Cohen's Kappa = 0.5858795459636794
Time taken = 2.981065511703491
              precision    recall  f1-score   support

         0.0    0.80402   0.84214   0.82264     23818
         1.0    0.78731   0.74004   0.76294     18807

    accuracy                        0.79709     42625
   macro avg    0.79567   0.79109   0.79279     42625
weighted avg    0.79665   0.79709   0.79630     42625
```

Fig 6.2.9 Accuracy of Logistic Regression

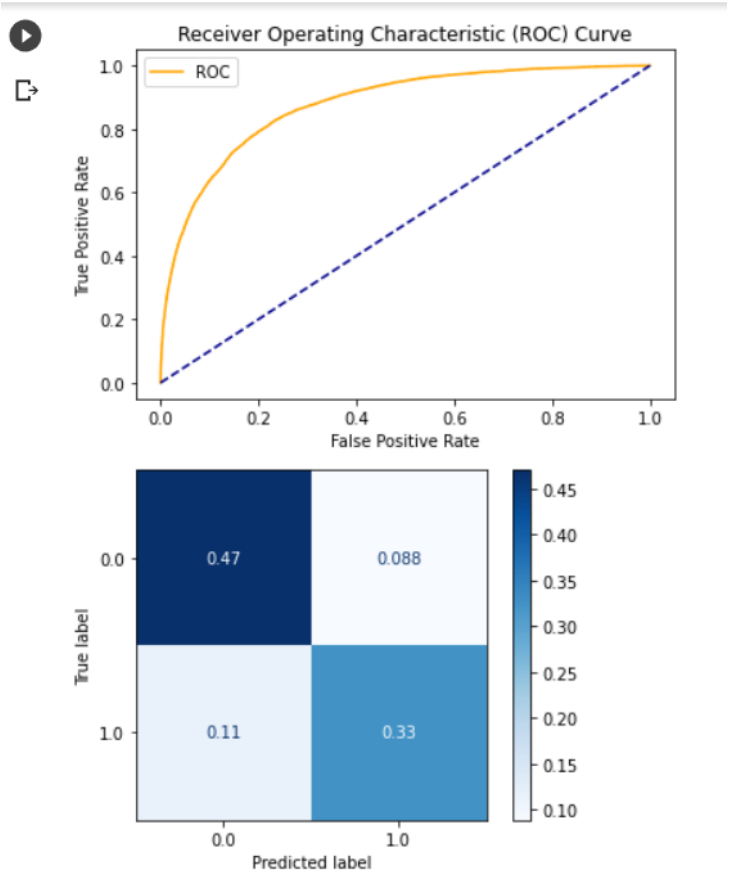


Fig 6.2.10 AUC_ROC of Logistic Regression

LIGHTGBM

Light GBM

```
import lightgbm as lgb
params_lgb ={'colsample_bytree': 0.95,
             'max_depth': 16,
             'min_split_gain': 0.1,
             'n_estimators': 200,
             'num_leaves': 50,
             'reg_alpha': 1.2,
             'reg_lambda': 1.2,
             'subsample': 0.95,
             'subsample_freq': 20
             }
model_lgb = lgb.LGBMClassifier(**params_lgb)
model_lgb, accuracy_lgb, roc_auc_lgb, coh_kap_lgb, tt_lgb = run_model(model_lgb, X_train, y_train, X_test, y_test)
```

```
Accuracy = 0.8708973607038123
ROC Area under Curve = 0.8711604331643068
Cohen's Kappa = 0.7392569056344488
Time taken = 8.708011388778687
              precision    recall  f1-score   support

         0.0    0.89682   0.86892   0.88265     23818
         1.0    0.84029   0.87340   0.85652     18807

    accuracy                        0.87090     42625
   macro avg    0.86856   0.87116   0.86959     42625
weighted avg    0.87188   0.87090   0.87112     42625
```

Fig 6.2.11 Accuracy of LightGBM

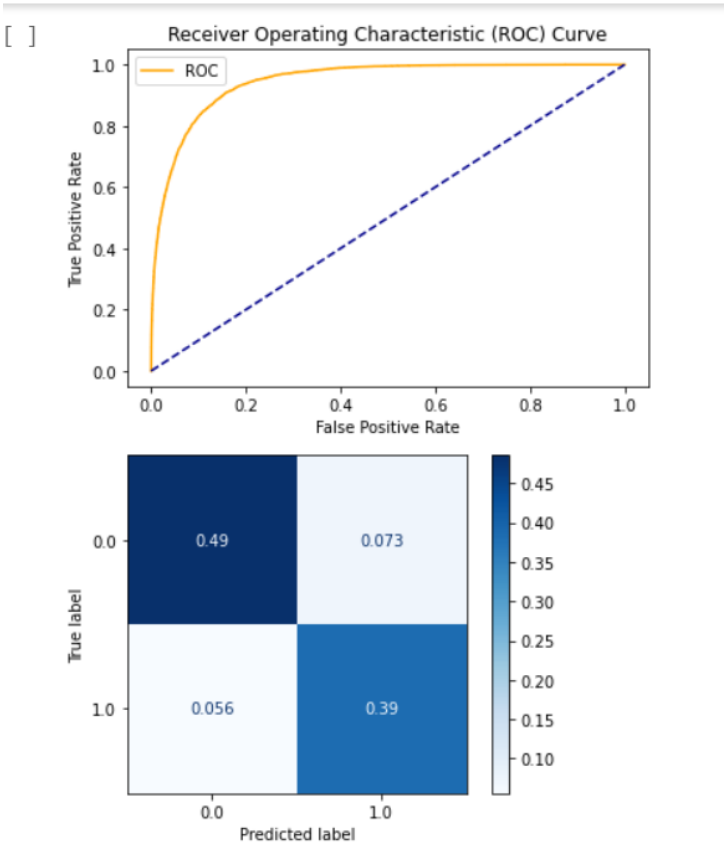


Fig 6.2.12 AUC_ROC of LightGBM

CATBOOST

```
!pip install catboost
import catboost as cb
params_cb ={'iterations': 50,
            'max_depth': 16}

model_cb = cb.CatBoostClassifier(**params_cb)
model_cb, accuracy_cb, roc_auc_cb, coh_kap_cb, tt_cb = run_model(model_cb, X_train, y_train, X_test, y_test, verbose=False)
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: catboost in /usr/local/lib/python3.8/dist-packages (1.1.1)
Requirement already satisfied: graphviz in /usr/local/lib/python3.8/dist-packages (from catboost) (0.10.1)
Requirement already satisfied: plotly in /usr/local/lib/python3.8/dist-packages (from catboost) (5.5.0)
Requirement already satisfied: numpy>=1.16.0 in /usr/local/lib/python3.8/dist-packages (from catboost) (1.21.6)
Requirement already satisfied: six in /usr/local/lib/python3.8/dist-packages (from catboost) (1.15.0)
Requirement already satisfied: pandas>=0.24.0 in /usr/local/lib/python3.8/dist-packages (from catboost) (1.3.5)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.8/dist-packages (from catboost) (3.2.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.8/dist-packages (from catboost) (1.7.3)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.8/dist-packages (from pandas>=0.24.0->catboost) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.8/dist-packages (from pandas>=0.24.0->catboost) (2022.7.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib->catboost) (1.4.4)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.8/dist-packages (from matplotlib->catboost) (0.11.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib->catboost) (3.0.9)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.8/dist-packages (from plotly->catboost) (8.1.0)
```

```
Accuracy = 0.9477536656891495
ROC Area under Curve = 0.950352303866791
Cohen's Kappa = 0.8947441417111319
Time taken = 360.137487411499
              precision    recall  f1-score   support

         0.0    0.97711   0.92825   0.95205     23818
         1.0    0.91454   0.97246   0.94261     18807

    accuracy                        0.94775     42625
   macro avg    0.94582   0.95035   0.94733     42625
weighted avg    0.94950   0.94775   0.94789     42625
```

Fig 6.2.13 Accuracy of CatBoost

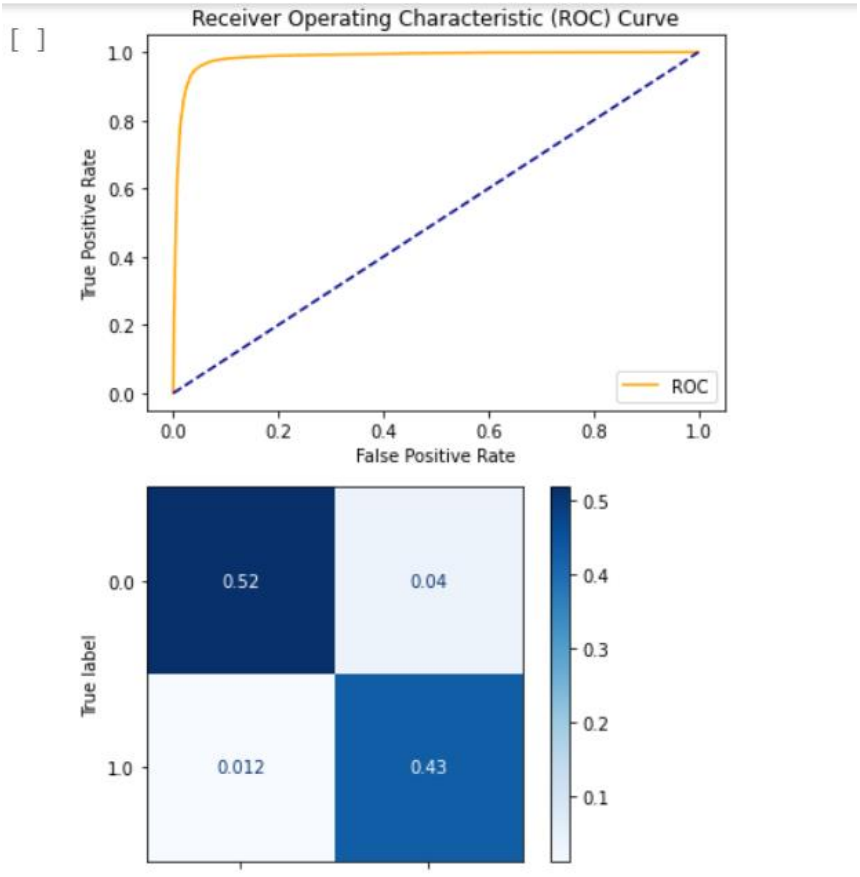


Fig 6.2.14 AUC_ROC of CatBoost

XGBOOST

XGBoost

```
import xgboost as xgb
params_xgb ={'n_estimators': 500,
             'max_depth': 16}

model_xgb = xgb.XGBClassifier(**params_xgb)
model_xgb, accuracy_xgb, roc_auc_xgb, coh_kap_xgb, tt_xgb = run_model(model_xgb, X_train, y_train, X_test, y_test, verbose=False)
```

```
Accuracy = 0.9401400602939624
ROC Area under Curve = 0.9415994194522621
Cohen's Kappa = 0.8791149659086246
Time taken = 310.314480304718
              precision    recall  f1-score   support

         0.0    0.96238   0.92936   0.94558     47707
         1.0    0.91398   0.95384   0.93349     37542

    accuracy                        0.94014     85249
   macro avg    0.93818   0.94160   0.93953     85249
weighted avg    0.94107   0.94014   0.94026     85249
```

Fig 6.2.15 Accuracy of XGBoost

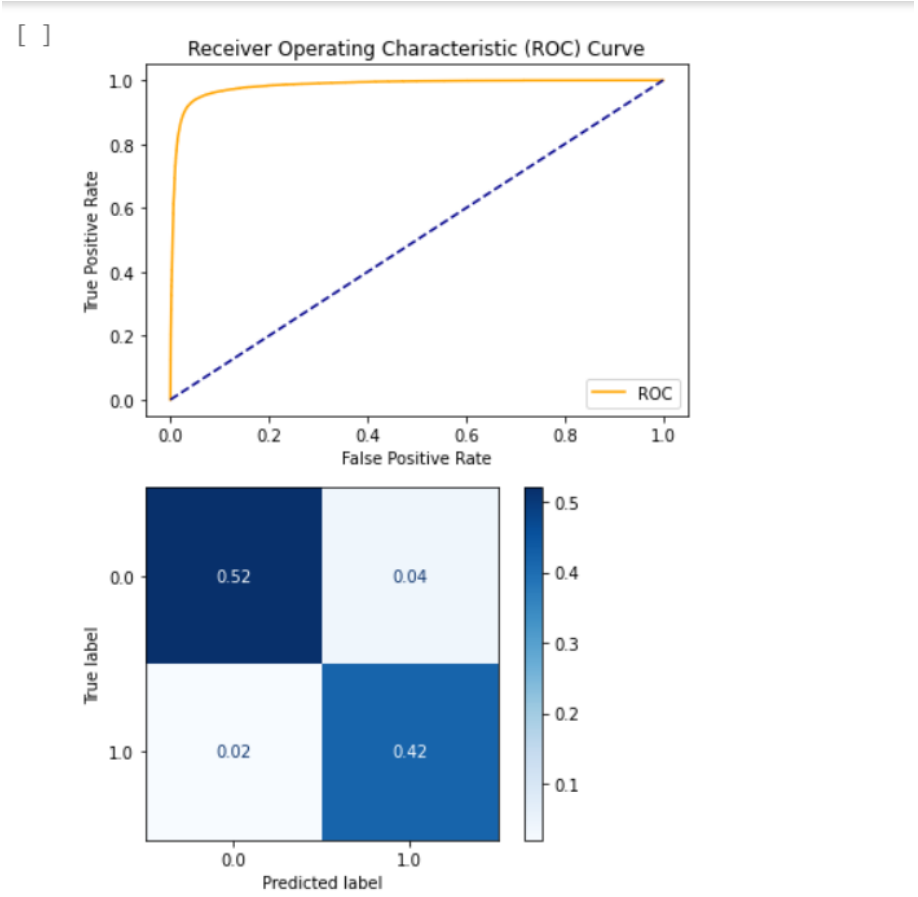


Fig 6.2.16 AUC_ROC of XGBoost

6.3 PERFORMANCE ANALYSIS

Prediction Model Comparison

```
accuracy_scores = [accuracy_lr,accuracy_lgb, accuracy_cb, accuracy_xgb]
roc_auc_scores = [roc_auc_lr,roc_auc_lgb, roc_auc_cb, roc_auc_xgb]
coh_kap_scores = [coh_kap_lr,coh_kap_lgb, coh_kap_cb, coh_kap_xgb]
tt = [tt_lr,tt_lgb, tt_cb, tt_xgb]

model_data = {'Model': ['LogisticRegression','LightGBM','Catboost','XGBoost'],
              'Accuracy': accuracy_scores,
              'ROC_AUC': roc_auc_scores,
              'Cohen_Kappa': coh_kap_scores,
              'Time taken': tt}
data = pd.DataFrame(model_data)

fig, ax1 = plt.subplots(figsize=(12,10))
ax1.set_title('Model Comparison: Accuracy and Time taken for execution', fontsize=13)
color = 'tab:green'
ax1.set_xlabel('Model', fontsize=12)
ax1.set_ylabel('Time taken', fontsize=10, color=color)
ax2 = sns.barplot(x='Model', y='Time taken', data = data, palette='summer')
ax1.tick_params(axis='y')
ax2 = ax1.twinx()
color = 'tab:red'
ax2.set_ylabel('Accuracy', fontsize=13, color=color)
ax2 = sns.lineplot(x='Model', y='Accuracy', data = data, sort=False, color=color)
ax2.tick_params(axis='y', color=color)
```

[ ]

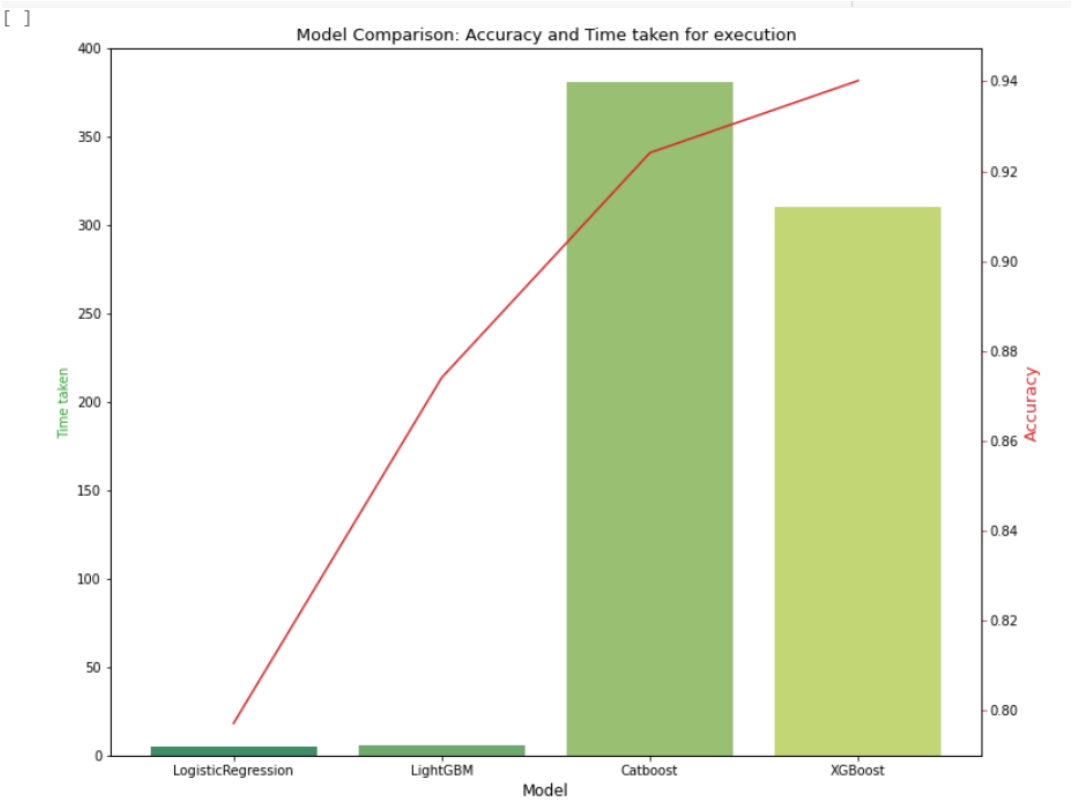
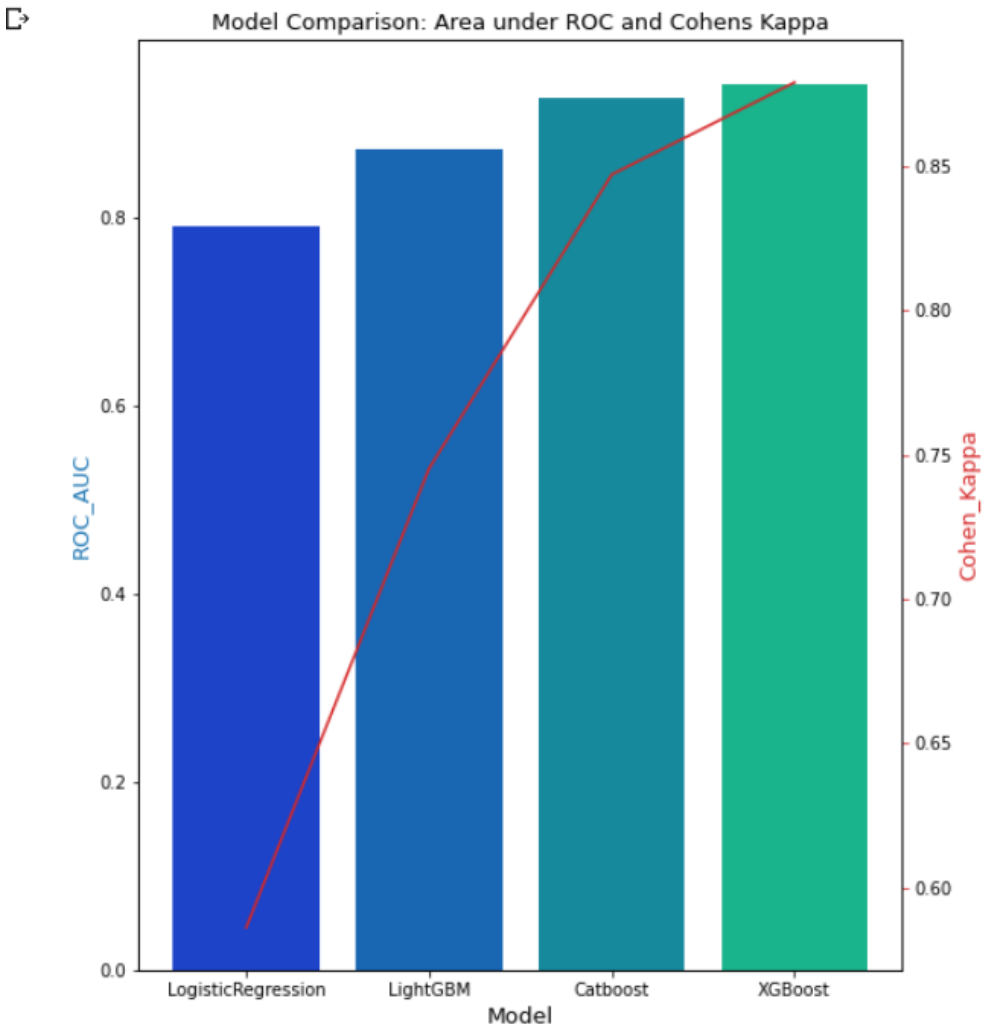


Fig 6.3.1 Performance Analysis based on Accuracy and Time Taken

```
fig, ax3 = plt.subplots(figsize=(8,10))
ax3.set_title('Model Comparison: Area under ROC and Cohens Kappa', fontsize=13)
color = 'tab:blue'
ax3.set_xlabel('Model', fontsize=13)
ax3.set_ylabel('ROC_AUC', fontsize=13, color=color)
ax4 = sns.barplot(x='Model', y='ROC_AUC', data = data, palette='winter')
ax3.tick_params(axis='y')
ax4 = ax3.twinx()
color = 'tab:red'
ax4.set_ylabel('Cohen_Kappa', fontsize=13, color=color)
ax4 = sns.lineplot(x='Model', y='Cohen_Kappa', data = data, sort=False, color=color)
ax4.tick_params(axis='y', color=color)
plt.show()
```



So, we conclude that XGBoost performs faster than other algorithms. XGBoost yields a better performance compared to existing method LogisticRegression.

Fig 6.3.2 Performance Analysis based on ROC_AUC and Cohen's Kappa

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 CONCLUSION

In this research, we discussed how we predict rainfall using machine learning models. The dataset contains many attributes of rainfall. We understood how the data has to be used before processing it into system and the purpose of it. We learned how to convert the categorial columns into numerical columns by imputation using LabelEncoder. Then, outliers were detected and removed from the dataset. We learnt how attributes are strongly related to other attributes from the correlation heatmap.The primary aim of the research is to predict rainfall and suggest a best model for prediction. The dataset was trained with different models to obtain optimal accuracy. We used multiple performance metrics to compare the results for each algorithm. The proposed methods yielded higher accuracy than existing methods. Performance analysis was done based on accuracy, time taken, AUC_ROC, Cohen's Kappa. XGBoost model performs relatively well.

7.2 FUTURE WORK

Although many attempts have been made to predict rainfall but still daily it comes out new solution. With huge amounts of data collected from social media websites, the best models improve every day. For future improvements, the prediction can be done before the occurrence of flood and develop an app regarding alerting the farmers before the occurrence of rain. The Farmers have to first enter their details regarding them, and the crops cultivated by them currently in the app. So, that the app gives a notification of when it rains.

CHAPTER 8
REFERENCES

[1] B. Revathi, C. Usharani,"Rainfall Prediction using Machine Learning Classification Algorithms",2021 SSRN Electronic Journal 9(1):2963-2967.

[2] D.Sirisha, P. Srijani, R. Dharani, K. Durga Vinusha, K. Pavan Kalyan, "Predicting Rainfall using Machine Learning Techniques." 2021 International Journal of Innovative Research in Technology IJIRT Volume 8 Issue 4.

[3] Devanshi Shukla, Vidhi Rajvir, Maulika S Patel,"Rainfall prediction using Neural Networks",2018 International Conference on Circuits and System in Digital Enterprise Technology.

[4] G. Bala Sai Tarun, J.V. Sriram, K. Sairam, K. Teja Sreenivas, M.V.B.T. Santhi, "Rainfall prediction using machine learning", International Journal of Innovative Technology and Exploring Engineering (IJITEE)ISSN: 2278-3075,Volume-8 2019

[5] Gowtham Sethupathi.Ma, Yenugudhati Sai Ganeshb, Mohammad Mansoor Alic,"Efficient rainfall prediction and analysis using Machine Learning Techniques", Turkish Journal of Computer and Mathematics Education Vol.12 No.6 (2021)

[6] V Nanda Kumar, V Chandrasekar," Rainfall Accuracy Prediction using Machine Learning Technique based on Linear Regression over Logistic Regression", Baltic Journal of Law & Politics a Journal of Vytautas Magnus University Volume 15, number 4 (2022)

[7] Nikhil Oswal ," Predicting Rainfall using Machine Learning Techniques",2019

[8] Nisha Thakur, Sanjeev Karmakar, Sunita Soni,"Rainfall Forecasting using various Artificial Neural Network Techniques-A Review."2021 International Journal of Scientific Research in Computer Science Engineering and Information Technology.

[9] Prem Kumar. B, R. Lakshmi, Bichitrananda Behera,"Performance Analysis and Evaluation of Machine Learning Algorithms in Rainfall Prediction" International Journal of Advanced Science and Technology,Volume-29 2020.

[10] R. Arya, Maya L Pai,"Prediction of rainfall using an optimized Genetic-Artificial Neural Network Model." International Journal of Pure and Applied Mathematics Volume 119 No. 10 2018, 669-678

[11] R. Kingsy Grace, B. Suganya, ”Machine Learning based Rainfall prediction using Multiple Linear Regression (MLR)” for Indian Dataset**.** 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS),IEEE.

[12] R. Senthil Kumar, C. Ramesh**,”**Rainfall prediction using Machine Learning and Deep Learning processes”,2016 International Conference on Inventive ComputationTechnologies (ICICT).

[13] S. Sakthivel, Dr. (Mrs.) G.Thailambal., “Effective Procedure to Predict Rainfall Conditions using Hybrid Machine Learning Strategies”, Turkish Journal of Computer and Mathematics Education Vol.12 No.6(2021), 209-216.

[14] Urooj Kaimkhani, Bushra Naz, Sanam Narejo**,**“Rainfall Prediction Using Time Series Nonlinear Autoregressive Neural Network.” SSRG International Journal of Computer Science and Engineering Volume 8 Issue 1, 30-38, January 2021.

[15] V.P Tharun, Ramya Prakash, S. Renuga DeviAriYair,”Prediction of Rainfall using DataMining Techniques”,2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT).

[16] Vikas Kumar, Vishal Kumar Yadav, Er. Sandeep Dubey**,”**Rainfall Prediction using Machine Learning” International Journal for Research in Applied Science & Engineering Technology (IJRASET) Volume 10 Issue V May 2022