# PROFESSIONAL TRAINING REPORT

**at**

**Sathyabama Institute of Science and Technology (Deemed to be University)**

Submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering

By

## CHELLUBOINA JYOTHI SWARUP

**REG. NO. 39110220**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**SCHOOL OF COMPUTING**

**SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY**

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI,**

**CHENNAI – 600119, TAMILNADU**

**APRIL 2022**

|  |  |  |
| --- | --- | --- |
|  | **SATHYABAMA**  **INSTITUTE OF SCIENCE AND TECHNOLOGY** (DEEMED TO BE UNIVERSITY) **Accredited with Grade “A” by NAAC**  (Established under Section 3 of UGC Act, 1956)  JEPPIAAR NAGAR, RAJIV GANDHI SALAI  CHENNAI– 600119  [**www.sathyabama.ac.in**](http://www.sathyabama.ac.in) |  |

# DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**BONAFIDE CERTIFICATE**

This is to certify that this Project Report is the bonafide work of **CHELLUBOINA JYOTHI SWARUP (Reg. No: 39110220)** who carried out the project entitled “**Prediction of Rainfall using Machine Learning**” under my supervision from January 2022 to April 2022.

## Internal Guide

## C. M. Suja, M.S., Ph.D.

**Head of the Department**

**Dr. L. Lakshmanan, M.E., Ph.D.**



## Submitted for Viva voce Examination held on

**InternalExaminer ExternalExaminer**

**DECLARATION**

I, **CHELLUBOINA JYOTHI SWARUP** hereby declare that the project report entitled **PREDICTION OF RAINFALL USING MACHINE LEARNING** done by me under the guidance of **Mrs. C. M. Suja** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering.

## DATE: (CHELLUBOINA JYOTHI SWARUP)

**PLACE: SIGNATURE OF THE CANDIDATE**

**ACKNOWLEDGEMENT**

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D**, **Dean**, School of Computing, **Dr. S. Vigneshwari, M.E., Ph.D. and Dr. L. Lakshmanan, M.E., Ph.D., Heads of the Department** of **Computer Science and Engineering** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Mrs. C. M. Suja, M.S., Ph.D.,** for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

**TRAINING CERTIFICATE**

# ABSTRACT

Rainfall prediction is one of the difficult and questionable assignments which essentially affect human society. Convenient and accurate predictions can serve to proactively reduce human and financial loss. For agriculture, rainfall is generally significant. Nowadays rainfall expectation has turned into a significant issue. Prediction of rainfall gives attention to individuals and knows ahead of time about rainfall to avoid the potential risk to shield their harvest from rainfall. There are numerous equipment gadgets for anticipating rainfall by utilizing the weather patterns like temperature, humidity, and pressure. These traditional techniques can't work proficiently so by utilizing Machine Learning techniques we can deliver precise outcomes. Machine Learning algorithms are generally helpful in predicting rainfall.

We can apply numerous strategies like classification, and regression as per the requirements and additionally we can calculate the error between the actual and predicted and accuracy. Various strategies produce various accuracies so it is important to pick the right algorithm and model it according to the requirements.

This study presents a set of experiments that include the utilization of prevalent machine learning techniques to construct models to predict whether it will rain tomorrow or not based on climate information for that particular day in major cities of Australia. This near study is led by focusing on three aspects: modeling inputs, modeling methods, and pre-processing techniques. The outcomes give a comparison of different evaluation metrics of these machine learning techniques and their reliability to anticipate rainfall by analyzing the weather data.

|  |  |  |
| --- | --- | --- |
|  | **TABLE OF CONTENTS** |  |
| **CHAPTER No** | **TITLE** | **PAGE No** |
|  | ABSTRACT | i |
|  | LIST OF FIGURES | iii |
|  | LIST OF TABLES | iv |
|  | LIST OF ABBREVIATIONS | v |
| 1 | **INTRODUCTION** | 1 |
|  | 1.1 Area of research | 2 |
| 2 | **AIM AND SCOPE OF THE PRESENT INVESTIGATION** | 3 |
|  | 2.1 Aim | 3 |
|  | 2.2 Scope of the project | 3 |
| 3 | **MATERIALS AND METHODS, ALGORITHMS USED** | 4 |
|  | 3.1 Software Requirements | 5 |
|  | 3.2 Hardware Requirements | 5 |
|  | 3.3 Understanding Dataset | 5 |
|  | 3.4 Data Preprocessing | 10 |
|  | 3.5 RandomForest Algorithm | 12 |
|  | 3.6 XGBoost Algorithm | 13 |
|  | 3.7 CatBoost Algorithm | 14 |
|  | 3.8 Understanding The Model Algorithm | 15 |
|  | 3.9 Building And Testing The Model | 16 |
|  | 3.10 Performance Metrics | 17 |
| 4 | **RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS** | 21 |
| 5 | **SUMMARY AND CONCLUSIONS** | 23 |
|  | **REFERENCES** | 24 |
|  | **APPENDIX** | 26 |
|  | A. SCREENSHOTS | 26 |
|  | B. SOURCE CODE | 42 |

# LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| FIGURE NO. | FIGURE NAME | PAGE NO. |
| 3.1 | Architecture map of the project | 4 |
| 3.2 | Information of the data | 8 |
| 3.3 | correlation of the dataset | 9 |
| 3.4 | Label encoding process of categorical data | 11 |
| 3.5 | RainTomorrow indicator in imbalanced dataset | 12 |
| 3.6 | RainTomorrow indicator in balanced dataset | 12 |
| 3.7 | Random Forest general structure | 13 |
| 3.8 | XGBoost classifier | 14 |
| 3.9 | Model complexity factors | 16 |
| 3.10 | Structure of confusion matrix for binary classification | 18 |
| 3.11 | ROC curve of RF model | 20 |
| 4.1 | visualizing the accuracy score | 22 |

# LIST OF TABLES

|  |  |  |
| --- | --- | --- |
| TABLE NO. | TABLE NAME | PAGE NO. |
| 3.1 | Gives information about the dataset | 6 |
| 3.2 | Sample of the dataset at the beginning i.e first five rows | 7 |
| 3.3 | Description of the data | 8 |
| 3.4 | Performance Measurements | 19 |
| 4.1 | Result of the models | 22 |

# LIST OF ABBREVIATIONS

|  |  |
| --- | --- |
| **ABBREVIATION** | **EXPANSION** |
| ML | Machine Learning |
| RF | Random Forest |
| AI | Artificial Intelligence |
| b/w | between |
| No. | Number of |
| min | Minimum |
| max | Maximum |
| DT | Decision Tree |
| int | integer |
| tp | True Positive |
| fp | False Positive |
| tn | True Negative |
| fn | False Negative |

**CHAPTER 1**

**INTRODUCTION**

Rainfall is a climatic factor that influences numerous human exercises like agricultural production, development, power generation, forestry, and so on. Rainfall prediction stays a genuine concern and has drawn the consideration of governments, enterprises, risk management entities, as well as the scientific community. Rainfall prediction is fundamental since this variable is the one with the most elevated connection with antagonistic normal events like avalanches, flooding, mass movements, and landslides. These incidents have impacted society for quite a long time. Hence, having a suitable methodology for rainfall prediction makes it conceivable to go to preventive and alleviation lengths for these natural phenomena.

In this project, we have proposed machine learning algorithms for the data collected. The objective here is to create the ensemble model which includes three machine learning algorithms to predict whether it will rain tomorrow based on features of the weather factors today in the dataset. As the values are categorical, we are using classification algorithms. In this project, we are using RandomForest, XGBoost and Catboost algorithms. It is well known Machine Learning Algorithms which can handle large datasets easily and efficiently and give us better results. These project aims to give end to end machine learning life cycle right from Data preprocessing to carrying out models to evaluating them. Data Preprocessing steps include imputing missing values, balancing the imbalanced data, encoding categorical features, feature scaling, feature selection and visualizing the data with graphs for comparisons and better understanding.

**1.1 AREA OF RESEARCH**

Machine learning (ML) is the investigation of computer science algorithmic calculations that can work on naturally through experience and by the utilization of data. Machine learning algorithms fabricate a model dependent on example information, known as "Training data", to make predictions or decisions without actually programming for every application.

AI (ML) is a sort of man-made brainpower (AI) that permits programming applications to turn out to be more exact at foreseeing results without being unequivocally modified to do as such. AI calculations utilize authentic information as a contribution to anticipate new yield esteems. AI is significant because it provides endeavors with a perspective on patterns in client conduct and business functional examples, just as supports the advancement of new items.

There are four basic approaches: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning. The type of algorithm data scientists chooses to use depends on what type of data they want to predict.

Machine learning algorithms are utilized in a wide range of applications, for example, in medication, email sorting, speech recognition, and computer vision, where it is troublesome or unworkable to develop ordinary algorithms to perform the required tasks.

A subset of machine learning is firmly identified with computational statistics, which centers around making predictions utilizing computers; yet not all machine learning is statistical learning. The investigation of mathematical optimization conveys strategies, hypotheses, and application areas to the field of machine learning.

**CHAPTER 2**

**AIM AND SCOPE OF THE PRESENT INVESTIGATION**

**2.1 AIM OF THE PROJECT**

The primary goal of the project is to train an ensemble model with the algorithms, that predicts the rainfall, based on the input features given to it. The main challenge of this project is to understand the dataset, deal with missing values, extract the relation between them and clean it and find necessary features that help in the prediction of rainfall and train the model using required parameters to get a good accuracy from the model. Using python and python integrated modules helps to face the challenges of a dataset and make an efficient model for predicting things.

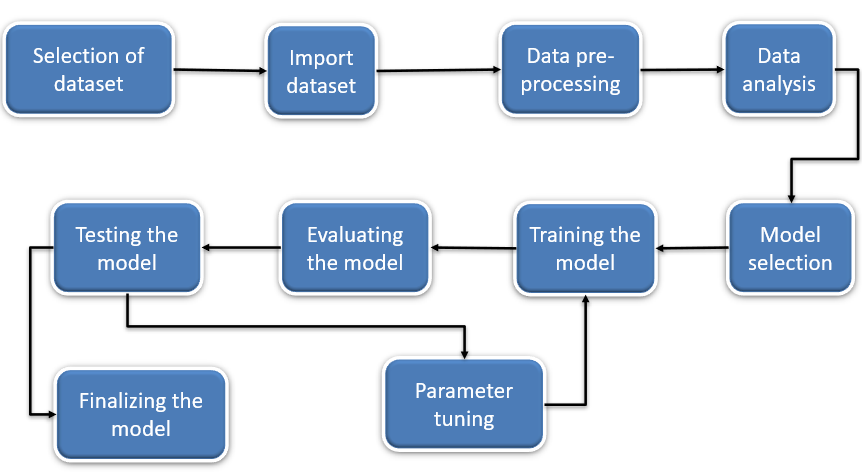
**2.2 SCOPE OF THE PROJECT**

The project is aimed to train and develop an ensemble model for the prediction of rainfall. The ensembled model is obtained by using three machine learning algorithms i.e., RandomForest, XGBoost, CatBoost. Here, the majority value which is obtained by the three models is taken as the final output which will increase the optimal result for the given problem. In this project, the data in the real world is utilized to train and test the prediction. The whole raw data contains the factors which affect the rainfall such as humidity, pressure, temperature, year and soon. Numerous ML Algorithms can tackle this issue, yet in this task, we are taking on the RandomForest, XGBoost, CatBoost algorithms for predicting the rainfall which is one of the better performing ML algorithms. The ensemble model can be used in various places to predict the rainfall which will give us the ideal outcome.

**CHAPTER 3**

**MATERIALS AND METHODS, ALGORITHMS USED**

In my study, various factors which affect the rainfall and values that are required for the prediction are noted in the dataset. Before developing the model, we need to understand the dataset, examine each feature, find the relations b/w the features, how the relations impact my results, and extract essential features. This phase is called understanding data. After understanding the data, we need to clean the dataset, drop unnecessary values, fill the missing values, and deal with categorical data. This phase is called data preprocessing. This data preprocessing helps to make compatible data structure for the model. After this preprocessing data phase, we need to understand the model, how it works, the advantages and disadvantages of algorithms, and its hyperparameters that help in performance. After understanding the model, we need to build the model with studied hyperparameters and test them with different variables of hyperparameters. This phase is called testing and validation. In this phase, we also need to understand the performance metrics of my problem statement, which helps in validating the model to reach my goal. All of these phases of the project are explained in detail in upcoming chapters and subchapters.



***Fig:3.1:*** *Architecture map of the project*

**3.1 SOFTWARE REQUIREMENTS:**

* Installing Anaconda Individual edition 64-bit (PY 3.8)
* Use Jupyter notebook in Anaconda Navigator for running project Python notebook.
* Python Notebook also works in Google Colab, Kaggle Notebook editor.

**3.2 HARDWARE REQUIREMENTS:**

* **License:** Free use and redistribution under the terms of the [EULA for Anaconda Individual Edition](https://www.anaconda.com/eula-anaconda-commercial-edition).
* **Operating system:** Windows 8 or newer, 64-bit macOS 10.13+, or Linux, including Ubuntu, RedHat, CentOS 7+, and others.
* **System architecture:** Windows- 64-bit x86, 32-bit x86; MacOS- 64-bit x86; Linux- 64-bit x86, 64-bit aarch64 (AWS Graviton2 / arm64), 64-bit Power8/Power9, s390x (Linux on IBM Z & LinuxONE).

Minimum 5 GB disk space to download and install.

**3.3 UNDERSTANDING DATASET:**

For understanding the data with python and its integrated libraries we need to import all necessary libraries for working with the dataset and import the dataset to the appropriate python notebook using pandas. Pandas module is designed to work with datasets. Using Pandas it is easy to understand the dataset. Using pandas commands, we can easily import the dataset and read it. As we observe the dataset the values are categorical, so we use a classification algorithm to predict the rainfall based on input features given to it. We use the train\_test\_split() method available in the sklearn library to split the data into train test sets. The train-test split procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model. The train-test split procedure is appropriate when you have a very large dataset, a costly model to train, or require a good estimate of model performance quickly. It is used for**splitting** data arrays into two subsets:**training** data and**testing** data. The trained set is used to fit the data into your machine learning model.

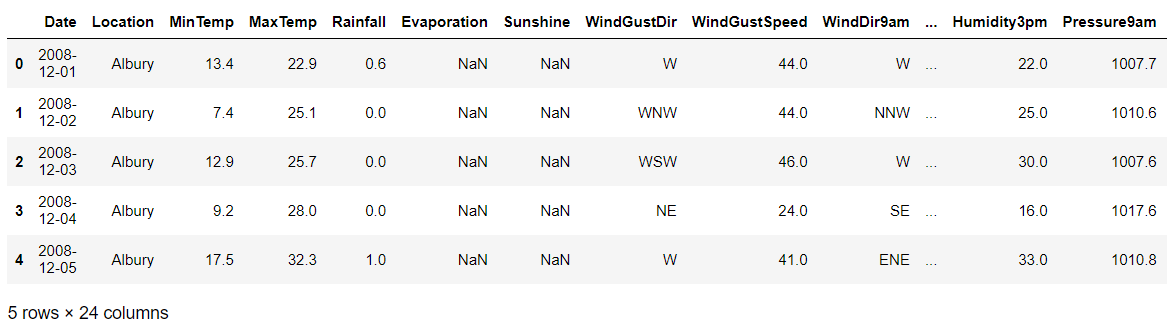
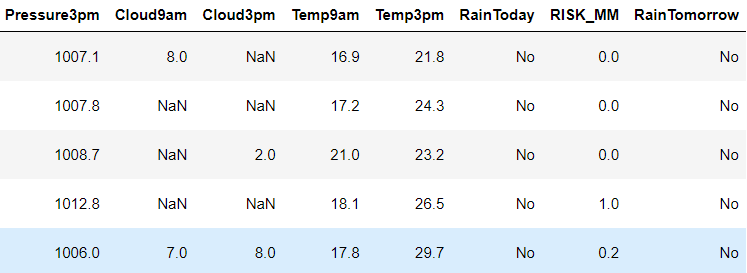
**Table No.3.1**: Gives information about the dataset

|  |  |  |
| --- | --- | --- |
| S.no | FEATURES | DESCRIPTION |
| 1 | Date | The date of observation |
| 2 | Location | The common name of the location of the weather station |
| 3 | MinTemp | The minimum temperature in degrees Celsius |
| 4 | MaxTemp | The maximum temperature in degrees Celsius |
| 5 | Rainfall | The amount of rainfall recorded for the day in mm |
| 6 | Evaporation | The so-called Class A pan evaporation (mm) in the 24 hours to 9am |
| 7 | Sunshine | The number of hours of bright sunshine in the day |
| 8 | WindGustDir | The direction of the strongest wind gust in the 24 hours |
| 9 | WindGustSpeed | The speed (km/h) of the strongest wind gust in the 24 hours |
| 10 | WindDir9am | Direction of the wind at 9am |
| 11 | WindDir3pm | Direction of the wind at 3pm |
| 12 | WindSpeed9am | Wind speed (km/hr) averaged over 10 minutes prior to 9am |
| 13 | WindSpeed3pm | Wind speed (km/hr) averaged over 10 minutes prior to 3pm |
| 14 | Humidity9am | Humidity (percent) at 9am |
| 15 | Humidity3pm | Humidity (percent) at 3pm |
| 16 | Pressure9am | Atmospheric pressure (hpa) reduced to mean sea level at 9am |
| 17 | Pressure3pm | Atmospheric pressure (hpa) reduced to mean sea level at 3pm |
| 18 | Cloud9am | Fraction of sky obscured by cloud at 9am |
| 19 | Cloud3pm | Fraction of sky obscured by cloud at 3pm. |
| 20 | Temp9am | Temperature (degrees C) at 9am |
| 21 | Temp3pm | Temperature (degrees C) at 3pm |
| 22 | RainToday | 1 if precipitation exceeds 1mm, otherwise 0 |
| 23 | RISK MM | The amount of next day rain in mm |
| 24 | RainTomorrow | The target variable. Did it rain tomorrow? |

# *3.3.1. Understanding datasets via tables*

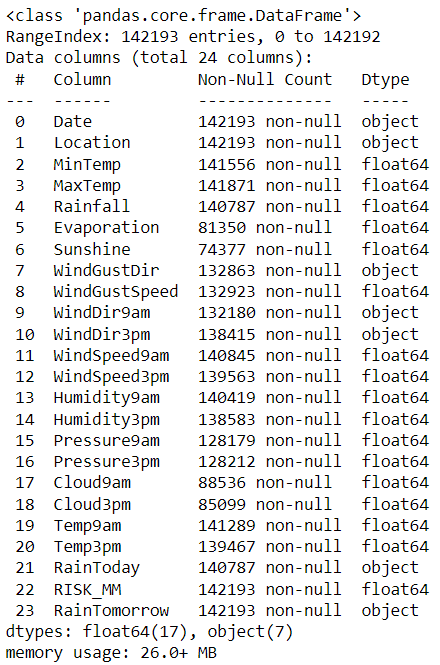
These features used in the dataset are understood by definition. Still, we also need to understand the dataset's structure, how data is represented in the tabular form, find out the missing values, and fill the missing values. This data is described in the below tables.

# Table 3.2: Sample of the dataset at the beginning i.e first five rows

We need to have the description of the dataset for the data preprocessing.

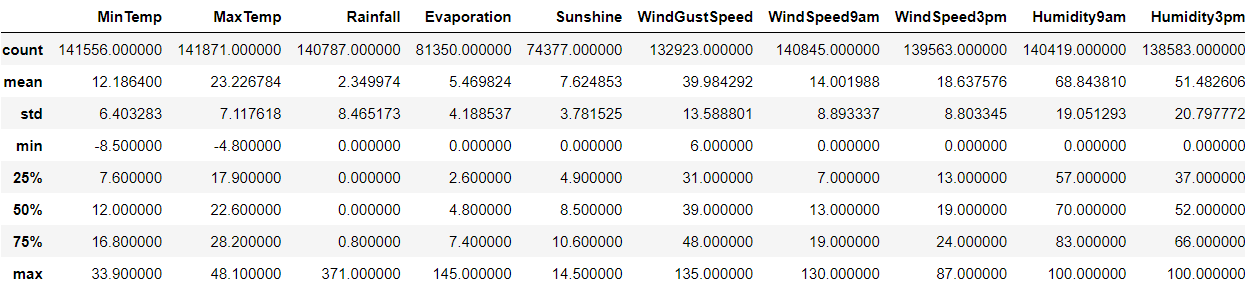
The columns names and non-null values count datatypes are represented in *fig 3.2****.***

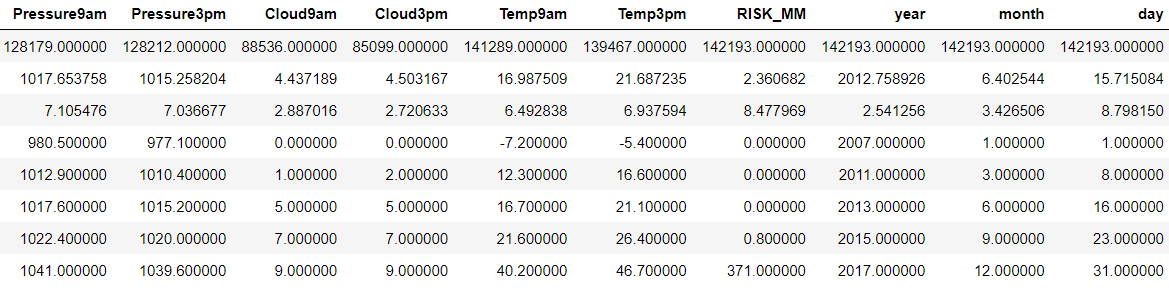


***Fig 3.2:*** *Information of the data*

The number of records, minimum value, maximum value, standard deviation, mean, 25% of max value, 50%(median) of the max value, 75% of the max values on the min-max range values of the dataset are represented in Table 3.3.

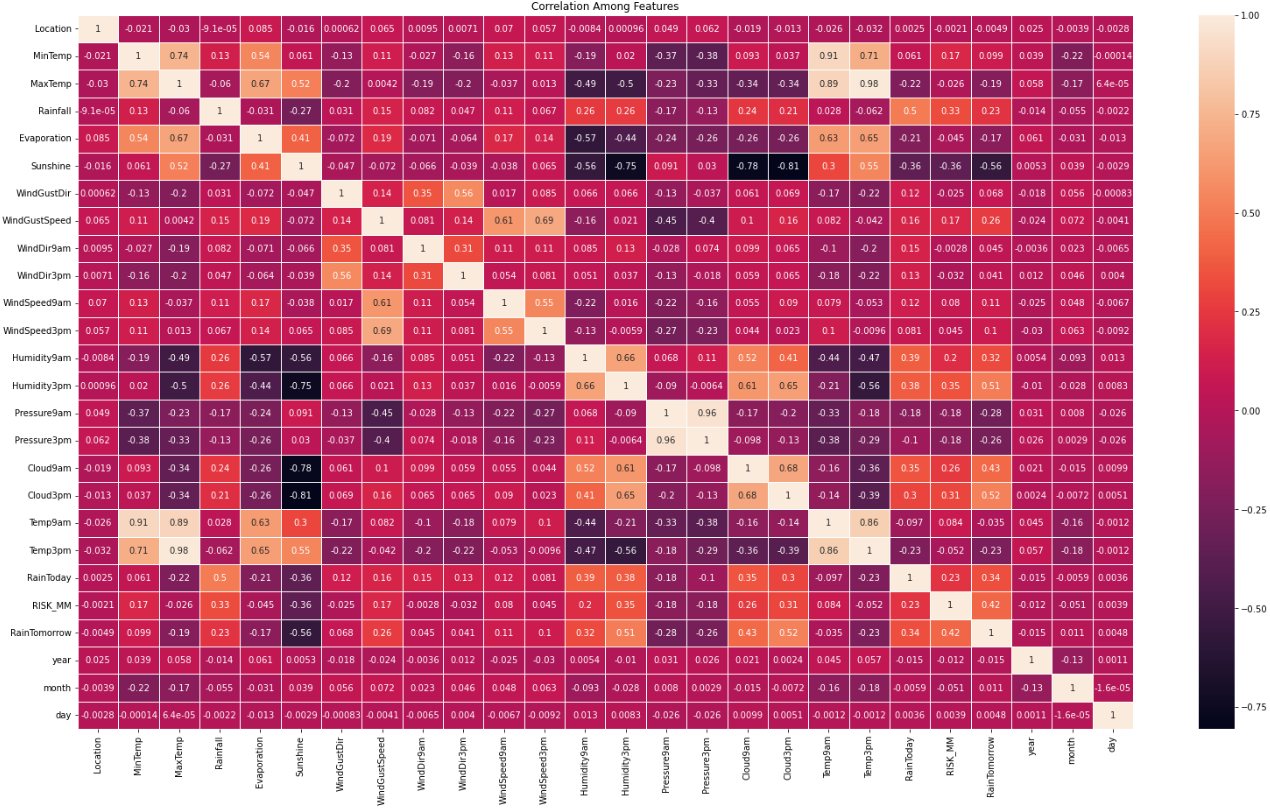
**Table 3.3:** Description of the data

****

****

***3.3.2 Understanding datasets via graphs***

Correlation is a statistical term portraying how much two variables move in coordination with each other. It can also be said as how two variables are dependent on each other. If the two variables move in a similar way, those variables are said to have a positive correlation. If they move in inverse ways, they have a negative correlation. It gives us the thought regarding the level of the relationship between the two factors. The correlation among the features in the dataset has been shown in figure 3.3.

****

***Figure 3.3:*** *correlation of the dataset*

***3.3.3 Data visualization***

Data visualization is the process of transforming large data sets into a statistical and graphical representation. It helps people understand the significance of data by summarizing and presenting a huge amount of data in a simple and easy-to

understand format and helps communicate information clearly and effectively. Visualization takes a huge complex amount of data to represent charts or graphs for quick information to absorb and better understandability. With the help of**data visualization,** we can see what the data looks like and what kind of correlation is held by the attributes of the data. **MATLAB** is a high-performance library for technical computing. It integrates computation,**visualization,** and programming in an easy-to-use environment where problems and solutions are expressed in common mathematical notation. Matplotlib is an easy-to-use, low-level data visualization library that is built on NumPy arrays. It consists of various plots like scatter plot, line plot, histograms, etc. Matplotlib provides a lot of flexibility.

# 3.4 DATA PREPROCESSING

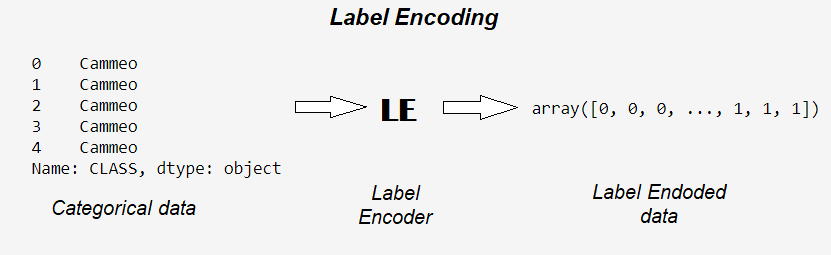
This phase is vitalfor building the model because, in this phase, we build a data structure that is compatible with the model, and we make random shuffles with the data to make the model learn the efficient way. Here we divide the data in terms of x and y, where x is the input of the model and y is the model's output.

*F(x) is the ML model, x is the input, and y is the model's output.*

After dividing input and output (x, y), we observe that output y that needs to be trained is an object data type. Random forest or any machine learning algorithms doesn't understand string or object datatype. This type of data is also known as categorical data. For making the machine understand object data type, it needs to be encoded. To deal with this categorical data, we need methods for encoding the data that the model can understand. There are different methods for encoding the categorical data like dropping columns, label encoding, and one-hot encoding, but we need to choose the method that suits the requirement. For my problem statement, we have a binary classification. The categorical data is prediction class. Here, we choose label encoding to deal with categorical data.

***3.4.1 Label Encoding***

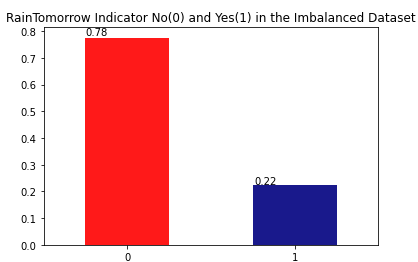
Label Encoding refers to changing over the labels into a numeric structure to change over them into the machine-understandable structure. ML Algorithms would then be able to choose in a superior manner how those labels should be worked. It is a significant preprocessing step for the structured dataset in supervised learning. So basically, converts No and Yes to 0 and 1 as labeling as integer data type, which becomes machine-readable data. The conversation of objects using python, in which there is a module called "LabelEncoder" under sklearn. Pre-processing is used for label encoding, as shown in figure 3.4.



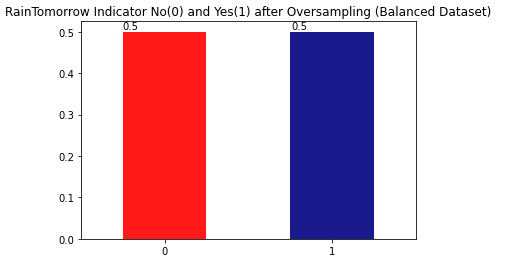
***Fig:3.4:*** *Label encoding process of categorical data*

***3.4.2 Resampling the data***

The Imbalanced classification problem is what we face when there is a severe skew in the class distribution of our training data. The skew may not be extremely severe, but the reason we identify imbalanced classification as a problem is because it can influence the performance of our Machine Learning algorithms. One way the imbalance may affect our Machine Learning algorithm is when our algorithm completely ignores the minority class. The reason this is an issue is because the minority class is often the class that we are most interested in. Here, in this dataset, the event of rainfall is extremely less contrasted with no rainfall. It would be a worry later while predicting the rainfall.



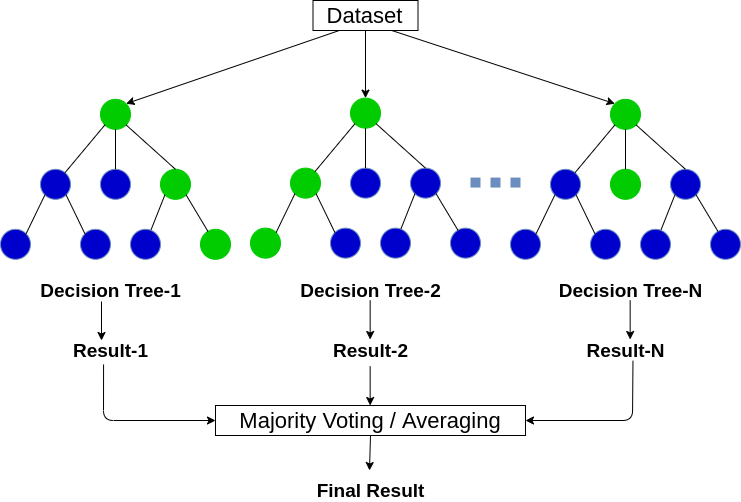
***Fig3.5:*** *RainTomorrow indicator in imbalanced dataset*



***Fig 3.6:*** *RainTomorrow indicator in balanced dataset*

**3.5 RANDOMFOREST ALGORITHM**

Random forest (RF) is a classifier that is a combination of many DT's. To make another classification, each DT gives a classification to the input sources. Then, in the end, RF evaluates the classifications and chooses the classification which has the most votes. RF can accommodate a large number of variables in the dataset. It is also excellent at estimating missing data. The most significant disadvantage of RF is the need for reproducibility. In addition, it is likewise hard to interpret the final model and its subsequent results. This is because of the way it contains numerous independent decision trees. Figure 3.7 shows the general construction of RF.

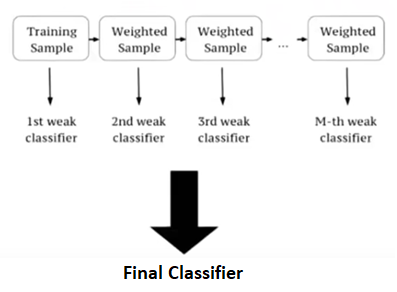


***Fig:3.7:*** *Random Forest general structure*

Building a model needs a methodology to achieve good accuracy for the problem. We are following the bagging method in this study. It is a combination of learning methods to increase the result. The Randomforest algorithm is based on the bagging method, which is based on the decision tree algorithm.

**3.6 XGBOOST ALGORITHM**

XGBoost is an implementation of Gradient Boosted decision trees. In this algorithm, decision trees are created in sequential form. Weights play an important role in XGBoost. Weights are assigned to all the independent variables which are then fed into decision trees that predict results. The weights of the variables predicted wrong by the tree are increased and these variables are then fed to the second decision tree. These individual classifiers/predictors then ensemble to give a strong and more precise model. It can work on regression, classification, ranking, and user-defined prediction problems.



***Fig:3.8:*** *XGBoost classifier*

**3.7 CATBOOST ALGORITHM**

The term CatBoost is an acronym that stands for "Category” and boosting. CatBoost supports **numerical**, **categorical**, and **text features** but has a good handling technique for categorical data. The CatBoost algorithm has quite a number of parameters to tune the features in the processing stage. "Boosting" in CatBoost refers to the gradient boosting in machine learning. Gradient boosting is a machine learning technique for regression and classification problems that produces a prediction model in an ensemble of weak prediction models, typically decision trees. CatBoost can improve the **performance** of the model while reducing overfitting and the time spent on tuning. It has several parameters to tune. Still, it reduces the need for extensive hyper-parameter tuning because the **default parameters** produce a great result. The CatBoost algorithm is a high-performance and **greedy novel** gradient boosting implementation.

**3.8 UNDERSTANDING THE MODEL ALGORITHM**

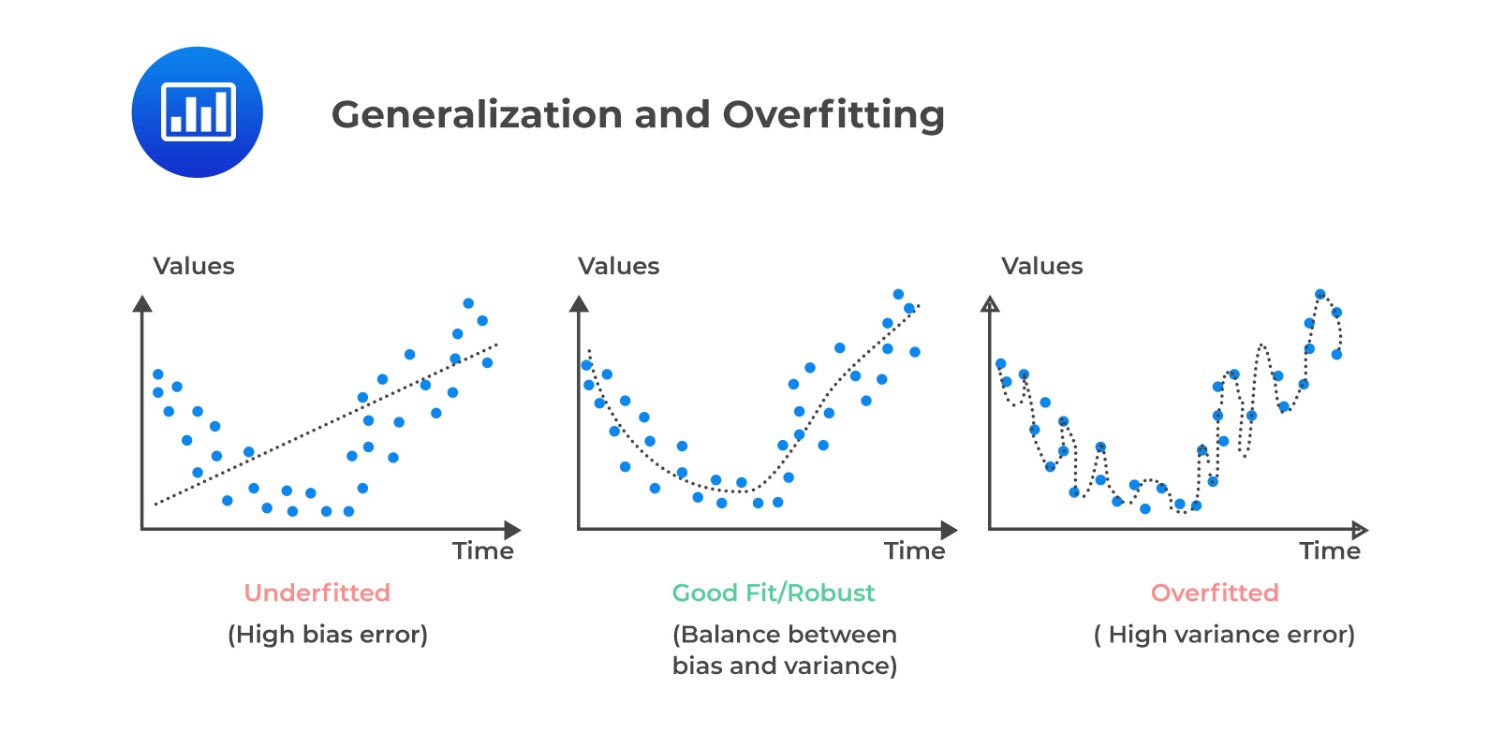
Before developing the model, we need to understand the model's nature. ML algorithms has parameters called hyperparameters which can be tuned to achieve maximum performance. The hyperparameters used in the model are

* + n\_estimators: The number of decision trees in the forest, (int, default=100).
  + bootstrap: Whether bootstrap samples are used when building trees. If False, the whole dataset is used to build each tree. (bool, default=True).
  + max\_depth: The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or contain less than min\_samples\_split samples. (int, default=None).
  + min\_samples\_leaf: The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least ``min\_samples\_leaf`` training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression. - If int, then considers `min\_samples\_leaf` as the minimum number. - If float, then `min\_samples\_leaf` is a fraction, and `ceil(min\_samples\_leaf \* n\_samples)` is the minimum number of samples for each node. (int or float, default=1).
  + random\_state: Controls both the randomness of the bootstrapping of the samples used when building trees and the sampling of the features to consider when looking for the best split at each node (int, default=None).

Here the n\_estimators(default=100), bootstrap(bootstrap =True) and random\_state(random\_state =12) are unchanged values. n\_estimator represents no. of decision trees, random state is given to test values for same shuffle data.

**3.9 BUILDING AND TESTING THE MODEL**

After understanding the model, we need to train and test the model to achieve better performance results. Model complexity is one of the factors that affect performance. The complexity of the model decides the overfitting, underfitting, and appropriate fitting decisions made by the model. Overfitting occurs when your model learns the training data too well and incorporates details and noise specific to your dataset. A model is overfitting when it performs great on your training/validation set but poorly on the test set (or new real-world data). Underfitting occurs when your model over-generalizes and fails to incorporate relevant variations in your data that would give your model more predictive power. A model is underfitting when it performs poorly on both training and test sets. Best Fit /Appropriate fit occurs when your model actualizes and learns the patterns of data to incorporate relevant variations in your data that would give your model more predictive power. A model is Best /Appropriate fitting when the model performs excellently in both training and testing datasets. Overfitting, Underfitting, and Appropriate fitting are explained in figure 3.9.



***Fig:3.9:*** *Model complexity factors*

We need a model to appropriate fit where it can go wrong in some values in prediction but gets the good majority of it. The parameters which affect the performance of the model when they variate are max\_depth and min\_sample\_leaf. If max\_depth increases, model complexity increases, and min\_sample\_leaf decreases, model complexity increases. So, in terms of model complexity, max\_depth is directly proportional, and min\_sample\_leaf is inversely proportional.

α Max depth

α

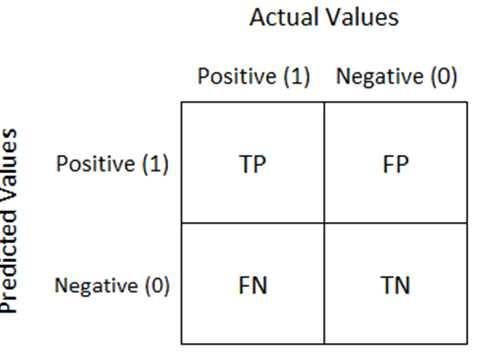
**3.10 Performance metrics**

Many learning algorithms have been proposed to date. It is generally expected that it is essential to evaluate the efficiency of an algorithm. Individuals even ended up making their metrics that suit the application. Performance metrics are often used to test and evaluate the model performance based on the requirement. For this project, the problem type is classification. In this review, we see the absolute most common measurements in a classification setting of a problem. Possible Performance metrics for classification problems are

* + - Confusion Matrix
    - F1score (balances the precision and recall)
    - Log loss
    - Accuracy score
    - ROC curve
    - ROC-AUC score (Area Under Curve in ROC Graph)

***3.10.1 Confusion Matrix***

A confusion matrix is a technique used for the performance of a classification algorithm. A confusion matrix is used to evaluate the accuracy of classification. By definition, a confusion matrix C is such that Ci,j equals the number of observations known to be in group i and predicted to be in group j. Thus, in binary classification, the count of true negatives is C0,0, false negatives are C1,0, true positives are C1,1, and false positives are C0,1. The tp, fp, tn, fn are shown infigure 3.10.



***Fig:3.10:*** *Structure of confusion matrix for binary classification*

The Confusion Matrix has four parameters, as seen in figure 3.10. They are:

• tp: true positive

• fp: false positive

• fn: false negative

• tn: true negative

The confusion matrix structure is a 2 x 2 matrix for binary classification because of tp, fp, tn, fn. Performance measurements such as Accuracy, Sensitivity, Specificity, Precision, F1-Score, Negative Predictive Value, False Positive Rate, False Discovery Rate, False Negative Rate are calculated using figure 3.10. Calculation formulas are given in Table 3.4**.**

**Table 3.4:** Performance Measurements

|  |  |  |
| --- | --- | --- |
| Sno. | Performance measure | Formula |
| 1 | Accuracy |  |
| 2 | Sensitivity |  |
| 3 | Specificity |  |
| 4 | Precision |  |
| 5 | F1-Score |  |
| 6 | Negative Predictive Value |  |
| 7 | Negative Positive Rate |  |
| 8 | False Discovery Rate |  |
| 9 | False Negative Rate |  |

* + The F1 score can be explained as a weighted average of precision and recall.
  + It is calculated with tp, tn, fp, and fn obtained from the confusion matrix
  + F1 score reaches its best at value 1 and worst score at 0.

***3.10.2 Log loss***

Log loss is also known as cross-entropy loss function. The loss function used in logistic regression and extensions such as neural networks is the negative log-likelihood of a logistic model that returns output predictions probabilities for its training data actual output. The log loss is only defined for two or more labels. For a single sample with true label  and a probability estimate  the log loss is defined as:

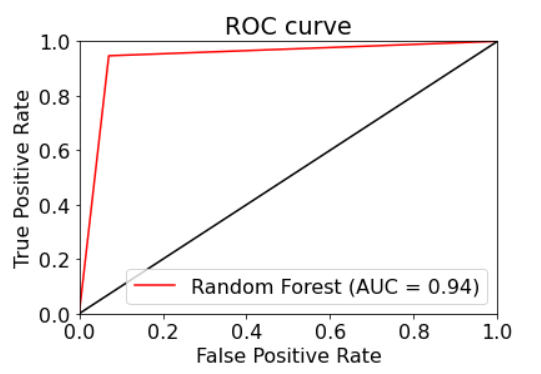
***3.10.3 Accuracy score***

The accuracy score is the measurement that tests the model's performance by an average of output prediction vs. actual output.

* + It ranges from 0 to 1, where 0 is worst, and 1 is Best.
  + It is often multiplied by 100 to get the accuracy percentage

***3.10.4 ROC AUC curve***

ROC curve is plotted against True Positives VS False Positives. Greater the area under the curve (AUC) better the model. The roc curve of the RF model is shown in figure 3.11.



***Fig:3.11:*** *ROC curve of RF model*

**CHAPTER 4**

**RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS**

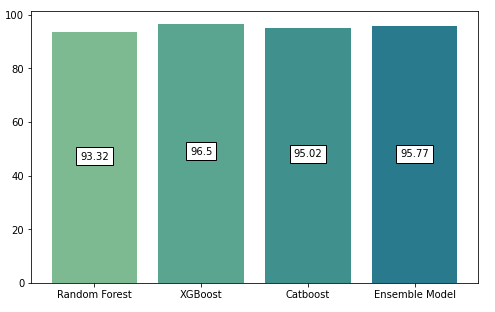
As mentioned in the above, the dataset is based on the factors that affect the rainfall. It gives us the idea of the complete factors which are responsible for the rainfall. The aim is to develop an ensemble model which will predict whether it will rain tomorrow by the data given to it. The models have been trained and tested to give a good accuracy model.

Here, three ML algorithms RandomForest, XGBoost, CatBoost, are taken to create the ensemble model. These are the best ML algorithms for classification-based learning. While exploring our data, we came to that one of our feature (Risk MM) was used for generating the target variable and hence it made no sense to use this feature for predictions. As the values to which we have to predict the rainfall is imbalanced, the first task is to balance the values because as mentioned above the machine learning model would completely ignore the minor values which would severely affect the accuracy of the model. After balancing the model, all the values of object datatype in the dataset have to be converted to categorial data by using the Label Encoding and the null values are replaced with the mode values of the particular feature.

As data cleaning and feature selection is done, now we have to split the data for training and testing the model. The data is splitted into 80% for the training and 20% for testing the data. Now, we are feeding the data to machine learning models, by changing parameters inversely, the model is been tested. We use the train\_test\_split() method available in the sklearn library to split the data into train test sets, Sklearn train\_test\_split will make random partitions for the two subsets.

The test set is used to evaluate the fit in your machine learning model. The train set is used to teach the machine learning model. Then the test set will be used to predict the output using the trained model and compare the output with the expected output to check whether the machine learning model is trained properly.

By using machine learning algorithms, we trained the data and predicted the values of log loss, accuracy score, f1 score and plotted the confusion matrix. With these data, we trained an ensembled model to predict the rainfall of tomorrow based on weather factors of today. Here the outcome is been finalized by the majority value that has been obtained by the three algorithms.



***Fig 4.1:*** *visualizing the accuracy score*

**Table 4.1:** Result of the models

|  |  |  |
| --- | --- | --- |
| S.NO | ALGORITHMS | ACCURACY SCORE |
| 1 | **RANDOMFOREST** | **0.9332154916491038** |
| 2 | **XGBOOST** | **0.9650100845287466** |
| 3 | **CATBOOST** | **0.9502345502753416** |
| 4 | **ENSEMBLED MODEL** | **0.9577129648514515** |

**CHAPTER 5**

**SUMMARY AND CONCLUSION**

In this study, a qualitative methodology has been used to reach our aim. The analytical process started with data cleaning and processing, missing value, exploratory analysis, and finally model building and evaluation. To achieve the best accuracy from the model, we have to check both training and testing score and we need to check the difference between train and test score and it should be minimum.

In this paper, we explored and applied a few preprocessing steps and learned their impact on the general execution of our classifiers. We additionally conveyed a comparative study of all the classifiers with different input data and noticed how the input data can affect the model predictions. We can presume that Australian weather conditions are uncertain and there is no such correlation between rainfall and the respective region and time. We figured out certain patterns and relationships among data which helped in determining important features. While exploring our data, we came to that one of our features (Risk MM) was used for generating the target variable and hence it made no sense to use this feature for predictions. The Machine Learning models that we developed is used to predict the rainfall efficiently and can also use in future purposes in predicting rainfall.

**REFERENCES**

[1] Python tutorial. (n.d.). Retrieved April 9, 2022, from <https://www.w3schools.com/python/>

[2] The absolute basics for beginners¶. NumPy. (n.d.). Retrieved April 6, 2022, from <https://numpy.org/doc/stable/user/absolute_beginners.html>

[3] Matplotlib.pyplot¶. matplotlib.pyplot - Matplotlib 3.5.1 documentation. (n.d.). Retrieved April 6, 2022, from <https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.html>

[4] Pandas.dataframe¶. pandas.DataFrame - pandas 1.4.2 documentation. (n.d.). Retrieved April 6, 2022, from <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html>

[5] 5. visualizations. scikit. (n.d.). Retrieved April 6, 2022, from <https://scikit-learn.org/stable/visualizations.html>

[6] Sklearn.preprocessing.LabelEncoder. scikit. (n.d.). Retrieved April 7, 2022, from <https://scikitlearn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html>

[7]Sklearn.ensemble.randomforestclassifier. scikit. (n.d.). Retrieved April 6, 2022, from <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>

[8] Sklearn.impute.IterativeImputer. scikit. (n.d.). Retrieved April 7, 2022, from <https://scikitlearn.org/stable/modules/generated/sklearn.impute.IterativeImputer.html>

[9] XGBoost documentation¶. XGBoost Documentation - xgboost 1.5.2 documentation. (n.d.). Retrieved April 7, 2022, from <https://xgboost.readthedocs.io/en/stable/>

[10] CatBoost: CatBoost categorical features. Analytics Vidhya. (2020, June 7). Retrieved April 7, 2022, from <https://www.analyticsvidhya.com/blog/2017/08/catboost-automated-categorical-data/>

[11] Narkhede, S. (2021, June 15). Understanding confusion matrix. Medium. Retrieved April 6, 2022, from <https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62?gi=5109a2d7d068>

[12] Sklearn.metrics.roc\_auc\_score. scikit. (n.d.). Retrieved April 6, 2022, from <https://scikitlearn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html>

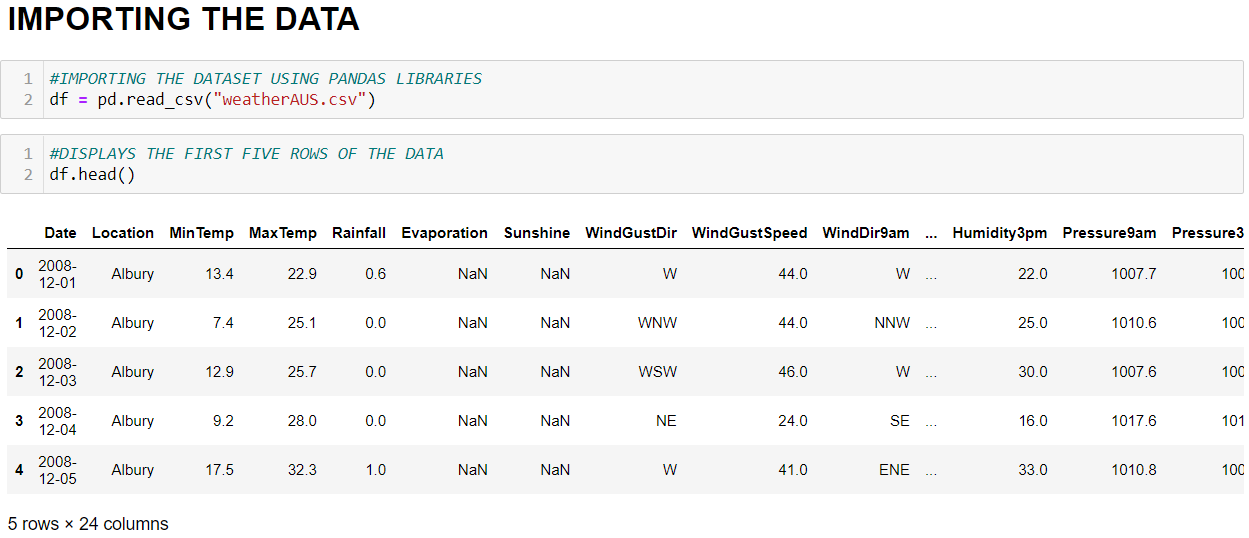
[13] 1.11. ensemble methods. scikit. (n.d.). Retrieved April 6, 2022, from <https://scikit-learn.org/stable/modules/ensemble.html#bagging-meta-estimator>

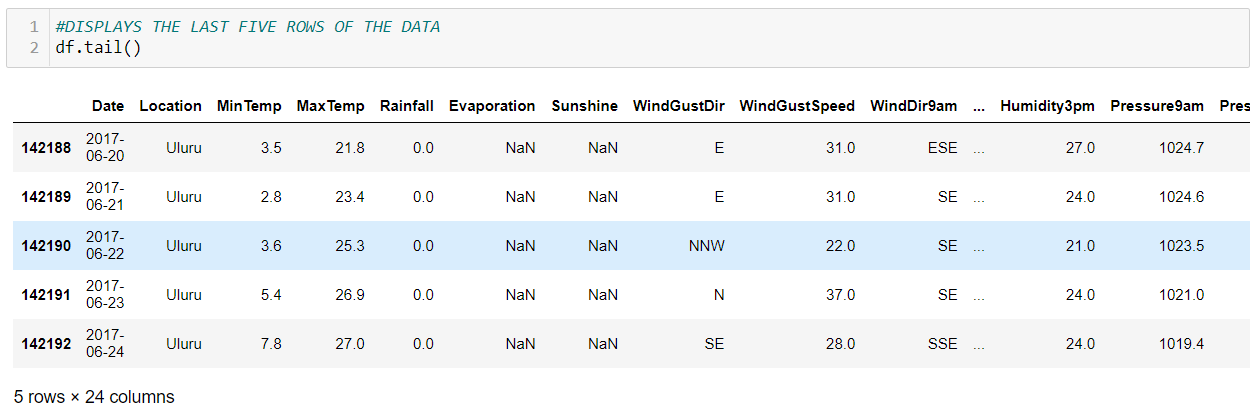
[14] Vidiyala, R. (2020, July 26). Performance metrics for classification machine learning problems. Medium. Retrieved April 9, 2022, from <https://towardsdatascience.com/performance-metrics-for-classification-machine-learning-problems-97e7e774a007>

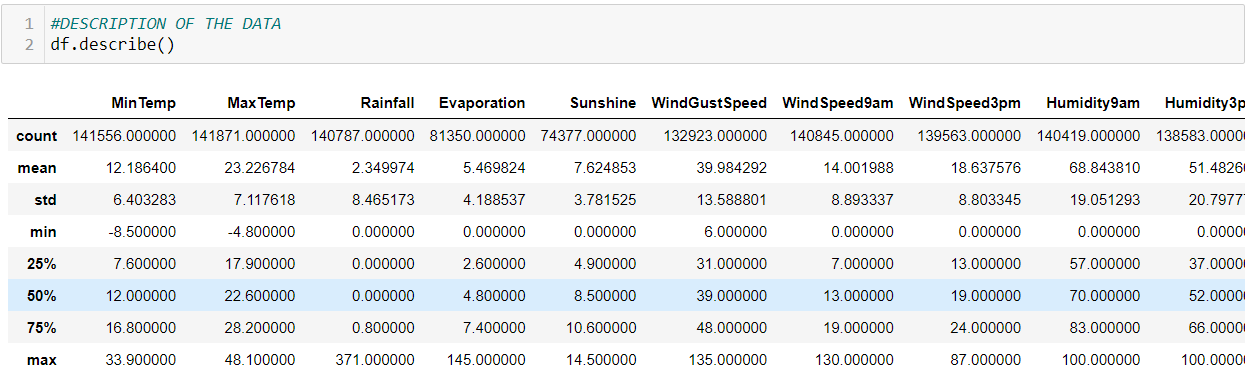
**APPENDIX**

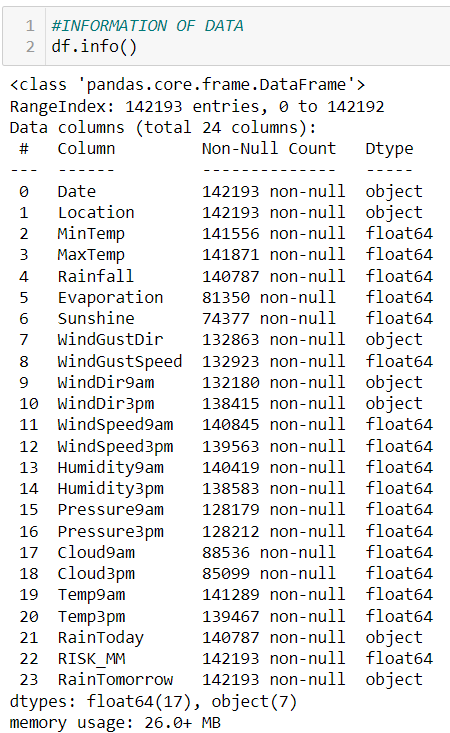
1. **SCREENSHOTS**

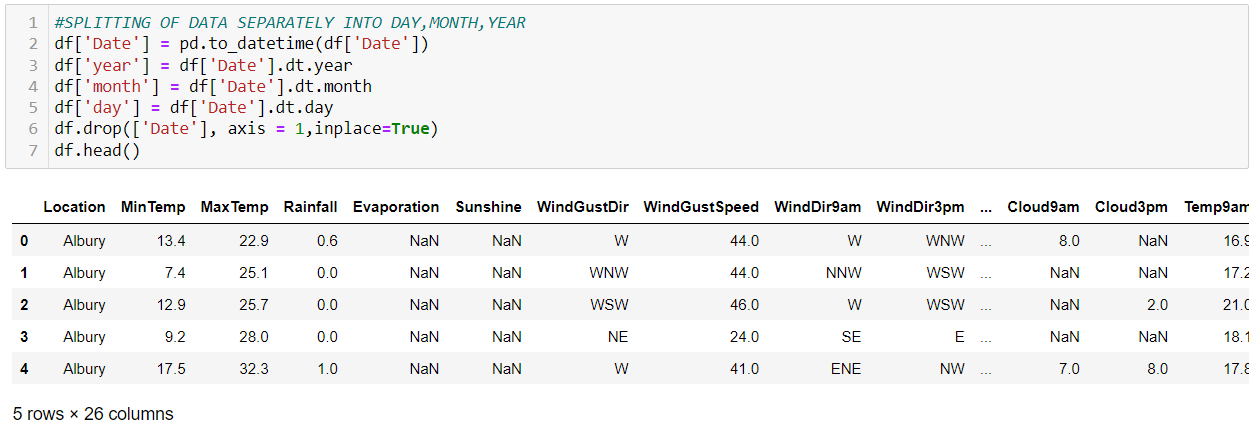


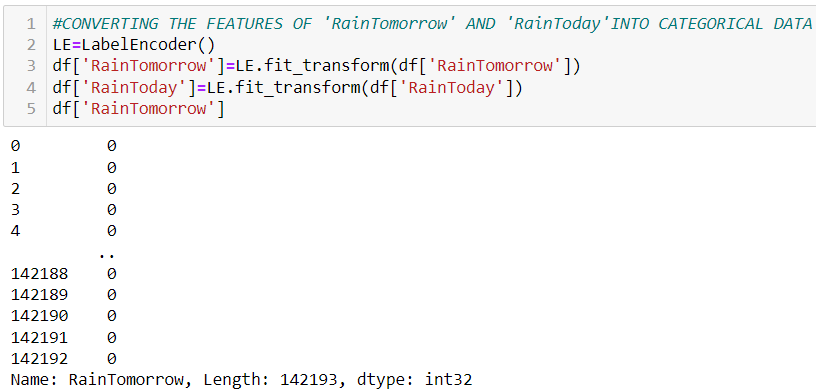


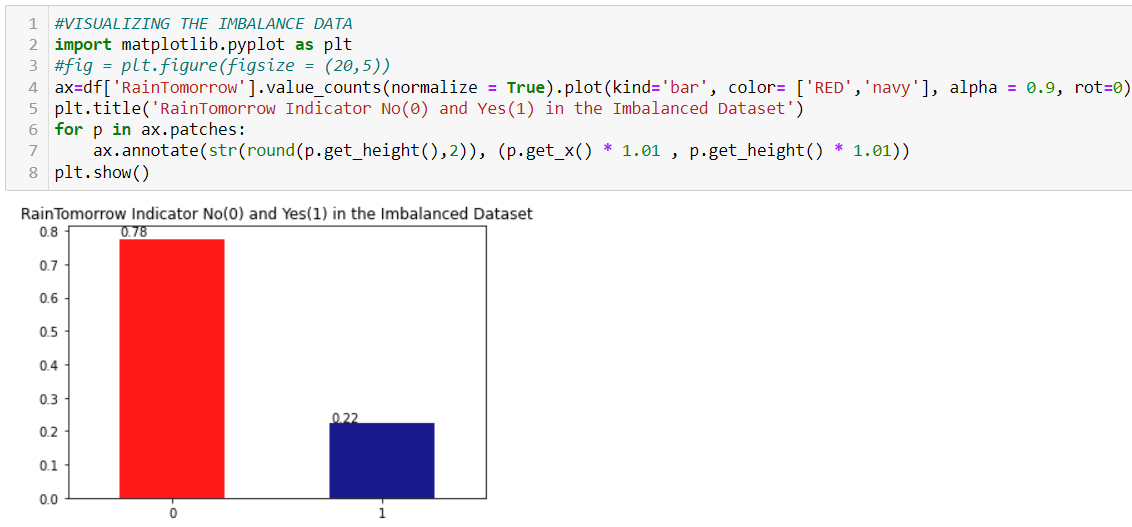




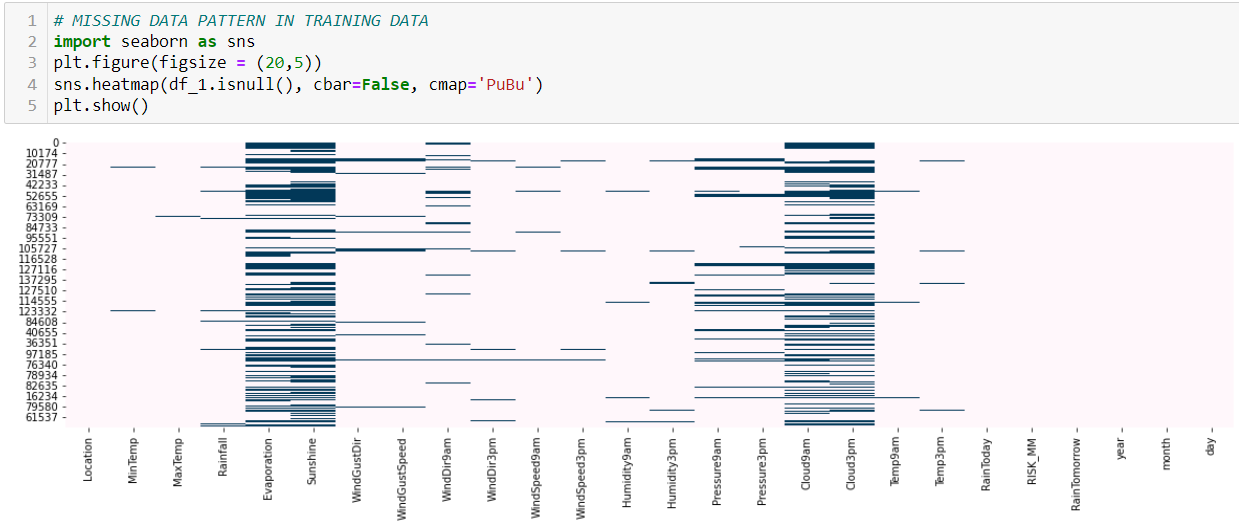


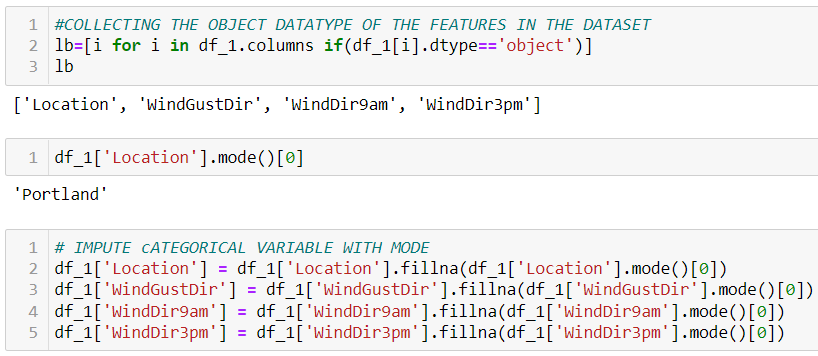


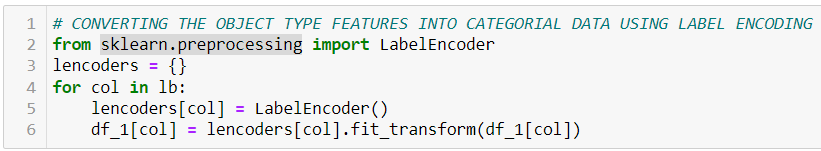


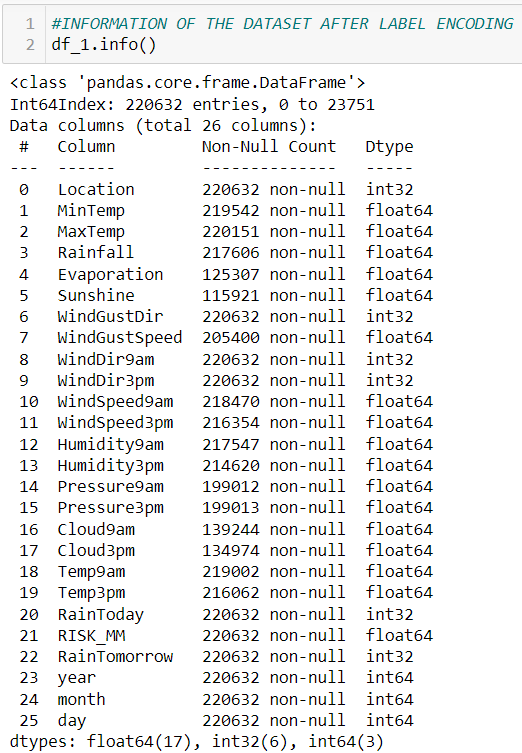


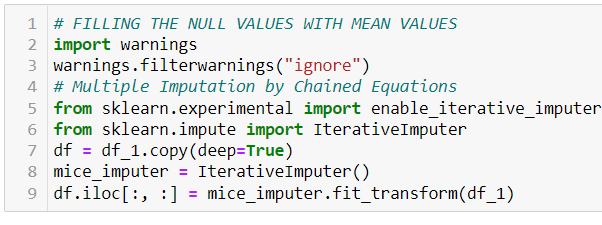






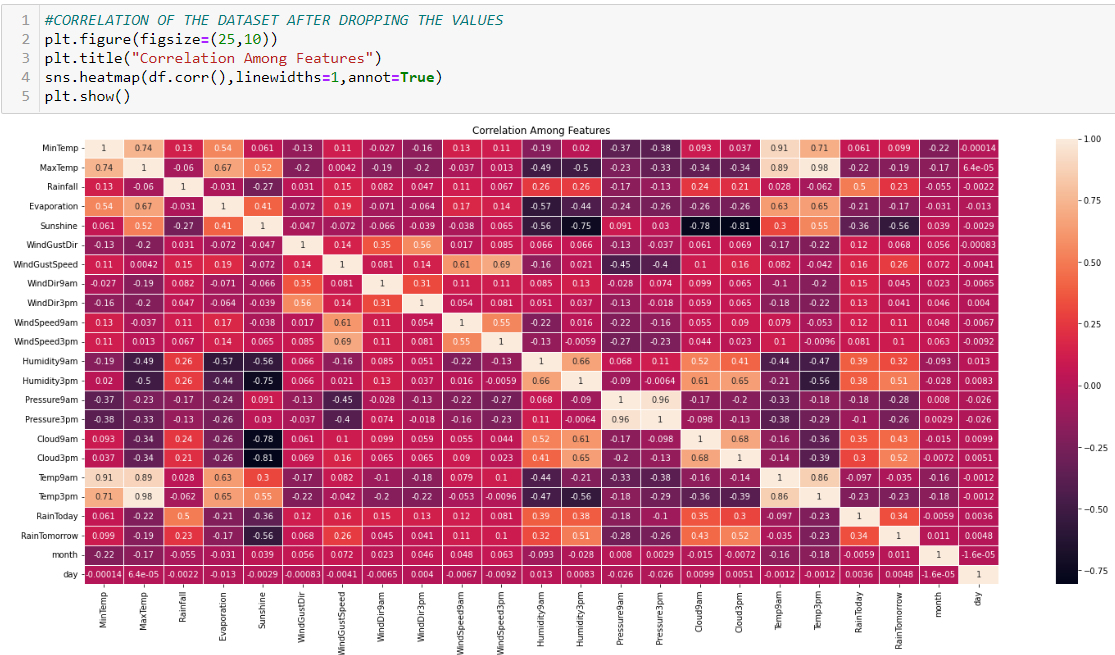


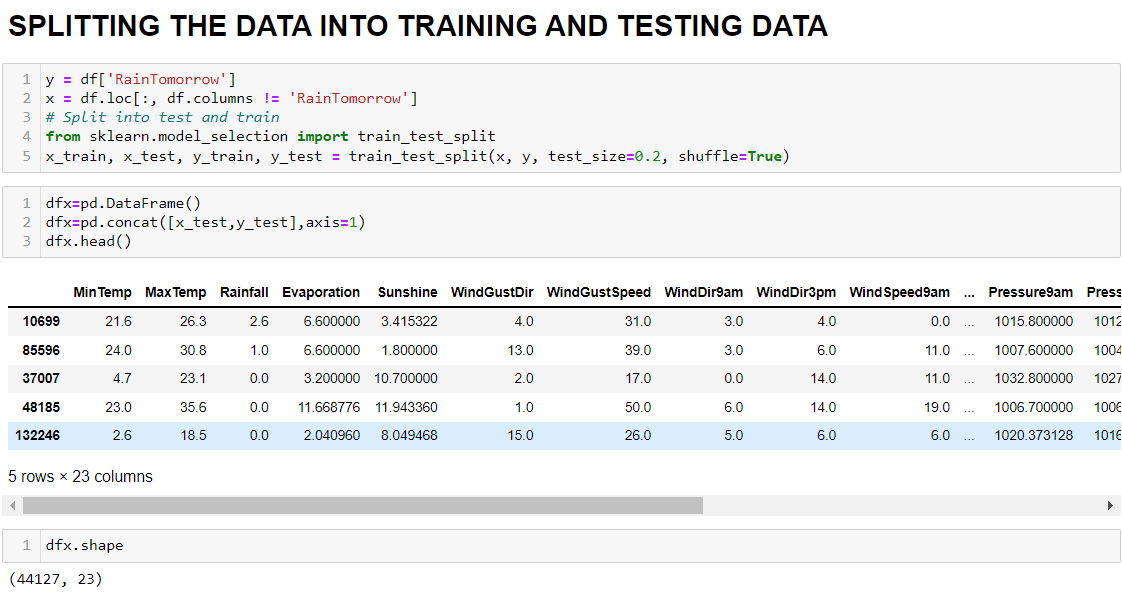


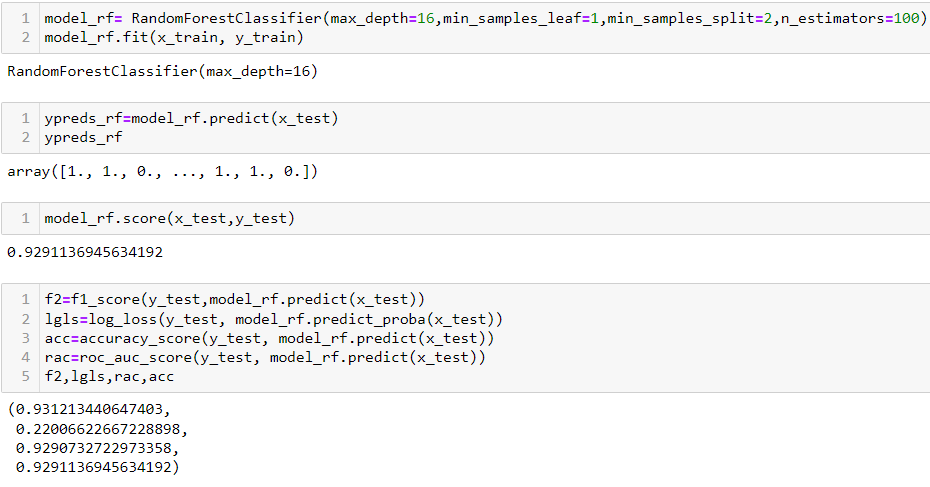


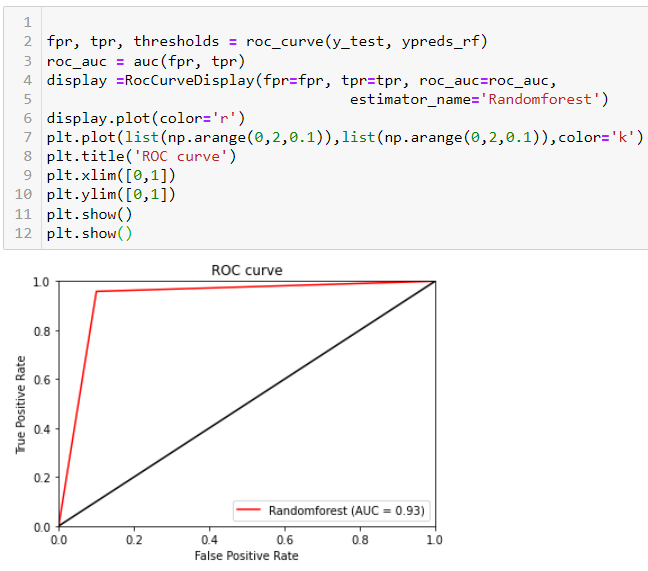


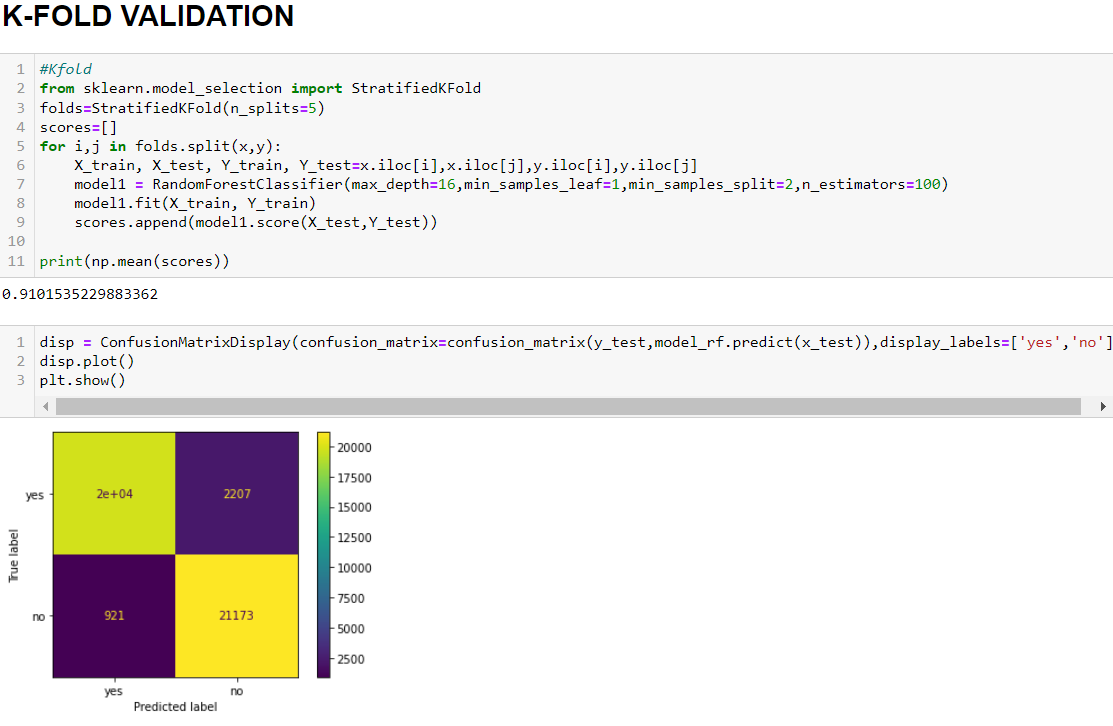


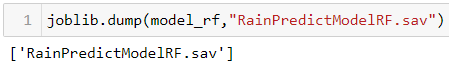


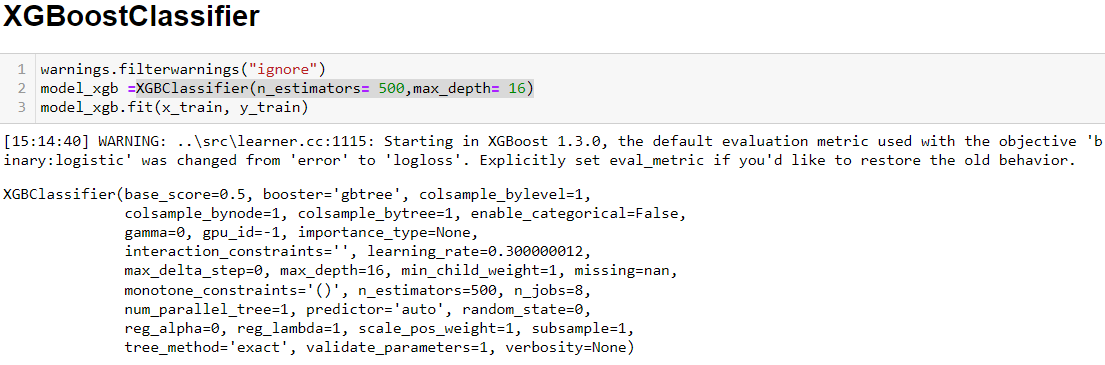




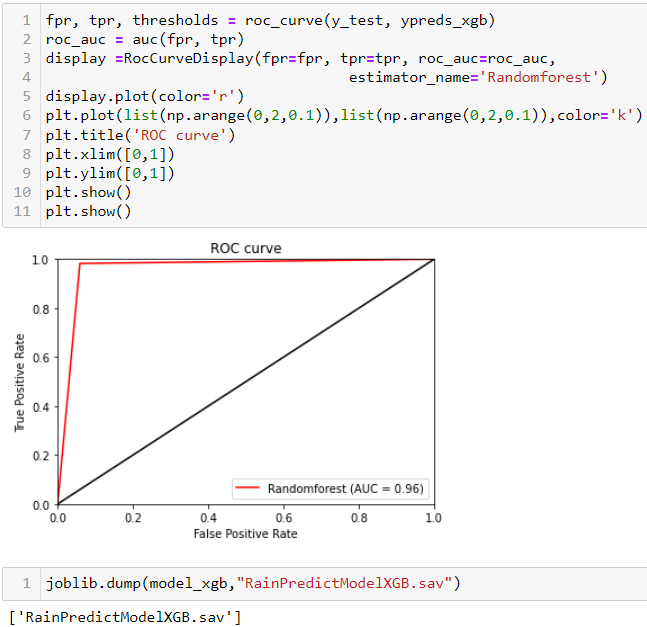


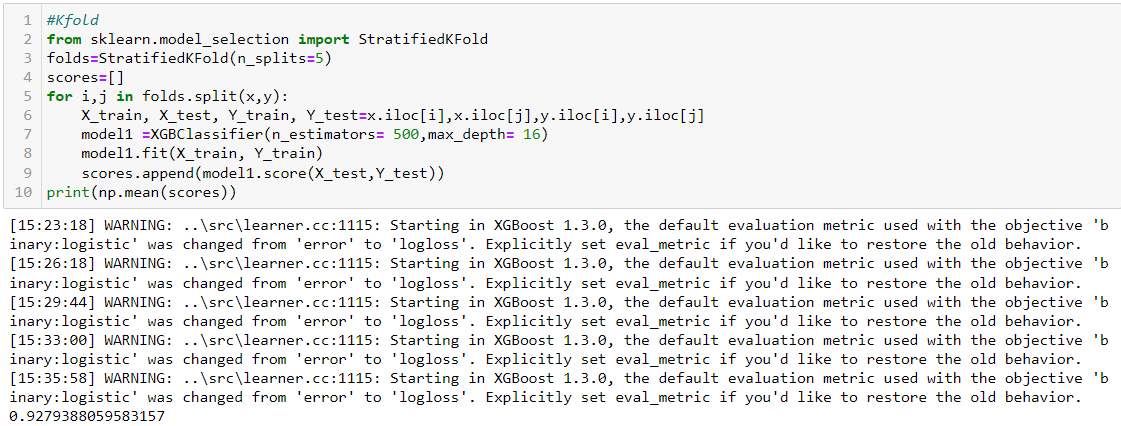


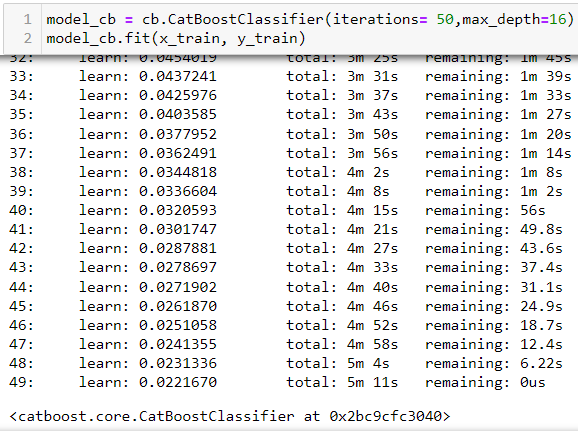


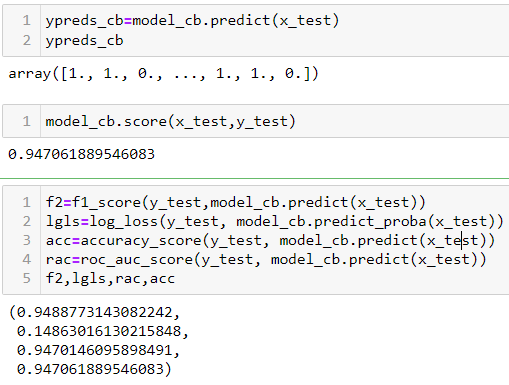


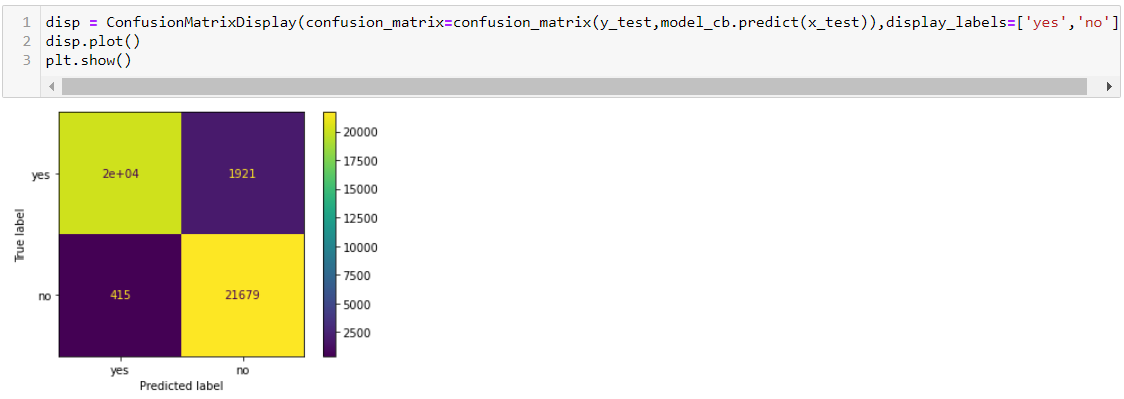


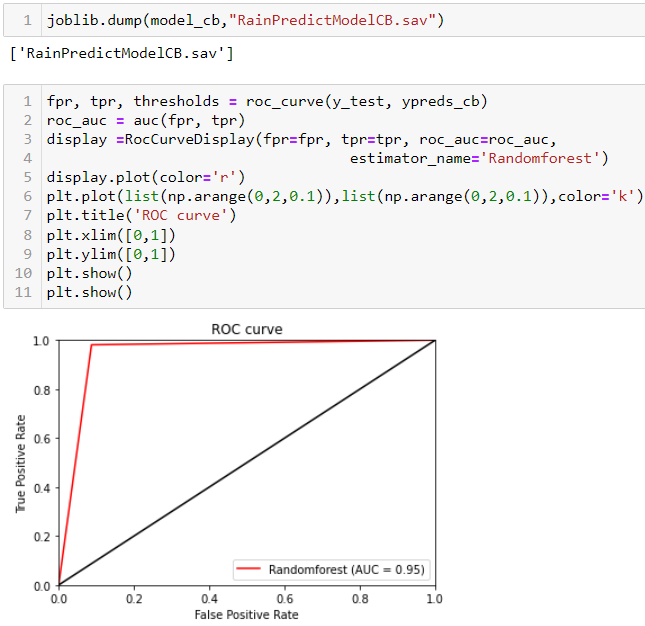


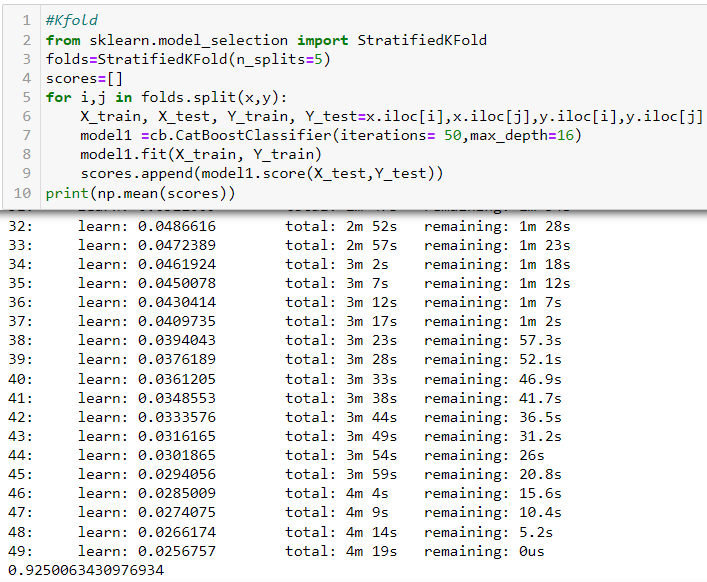




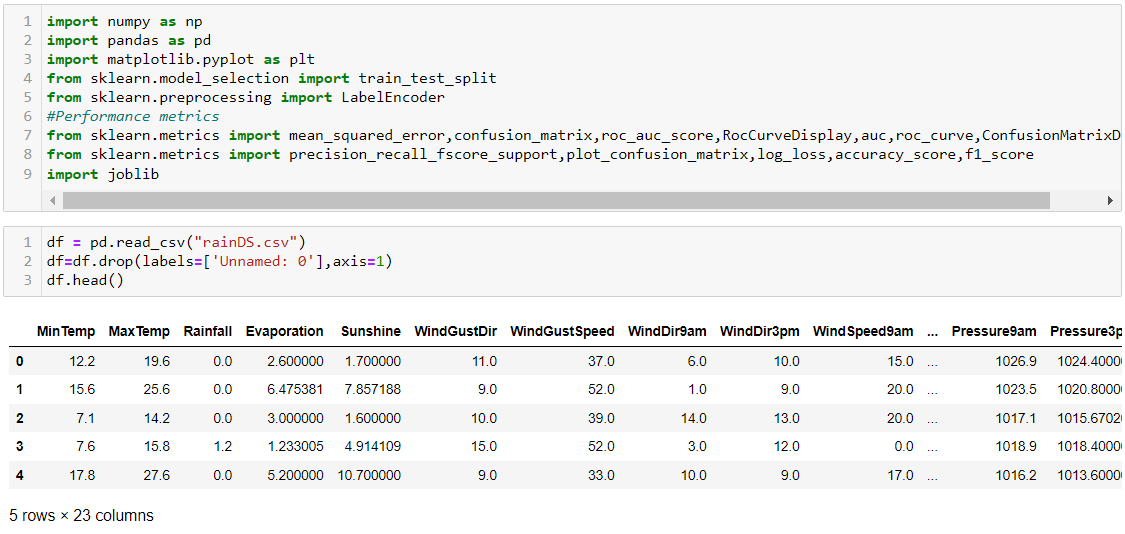


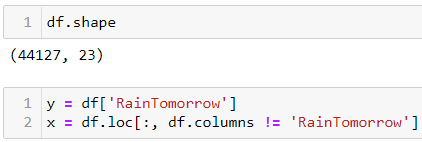


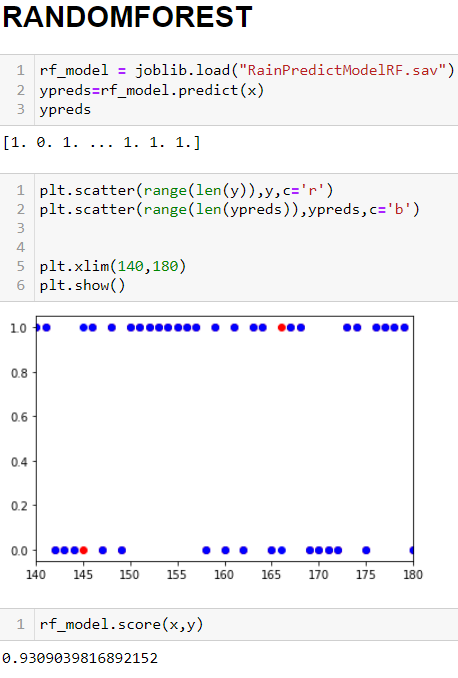




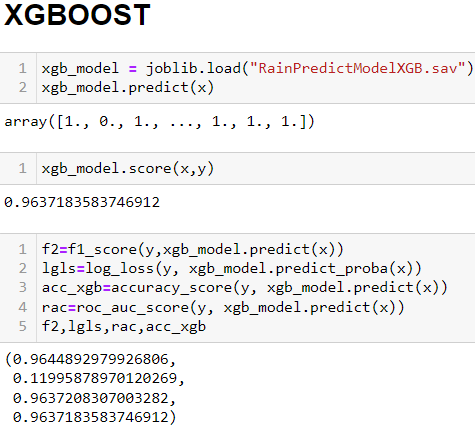
**MODEL DEPLOYMENT**

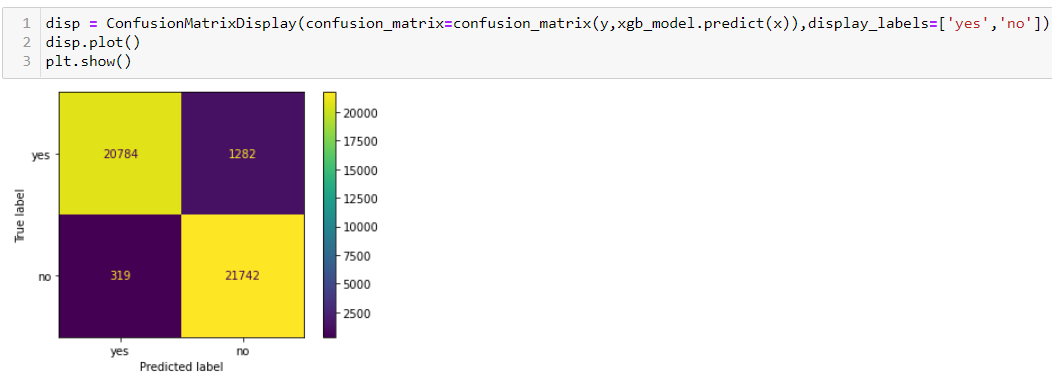
****

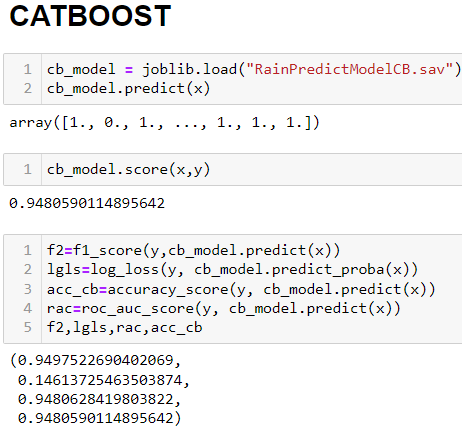
****

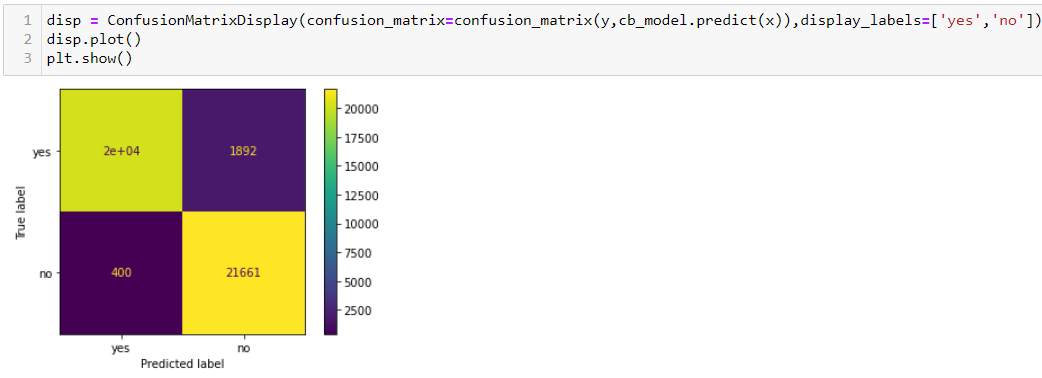
****

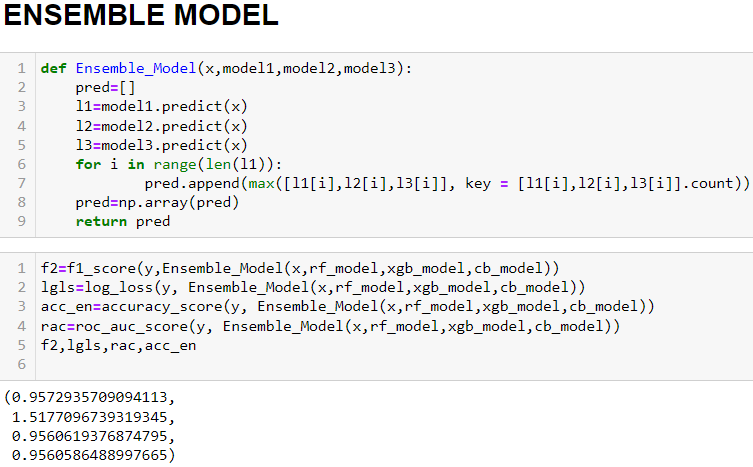
****

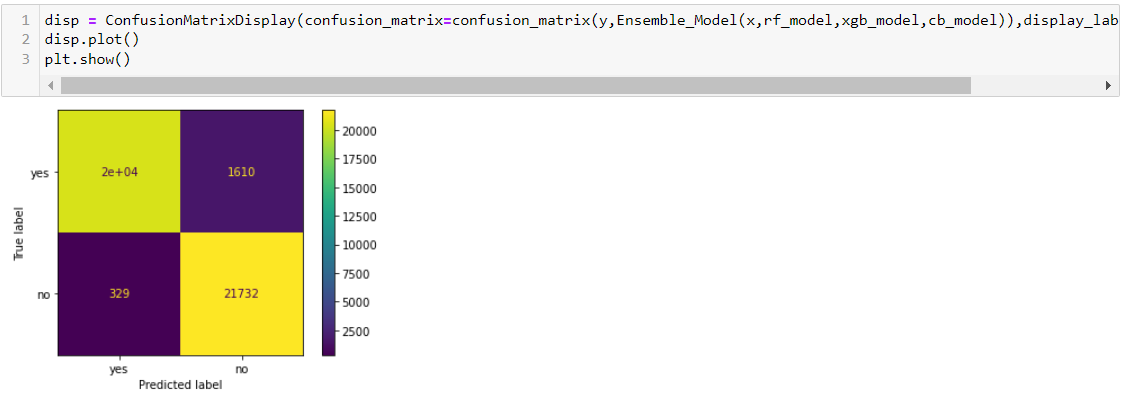
****

****

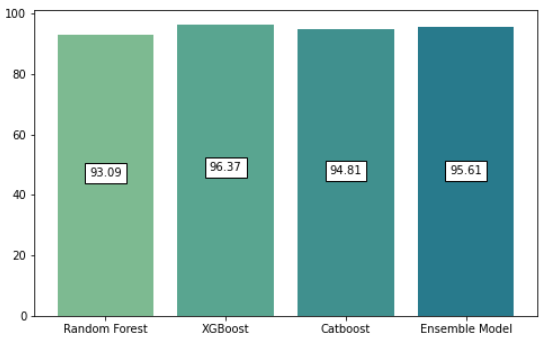
****

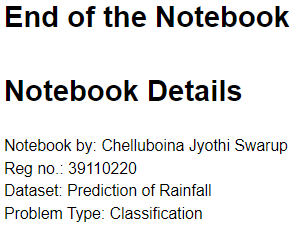
****

****

****

****

****

****

**B. SOURCE CODE**

%matplotlib inline

#Visualization Libraries

import matplotlib.pyplot as plt

import seaborn as sns

#Data processing Libraries

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder

from math import sqrt

#from sklearn.feature\_selection import VarianceThreshold

import warnings

#Machine model Algorithm module

from sklearn.ensemble import RandomForestClassifier

from xgboost import XGBClassifier

#from sklearn.tree import plot\_tree,DecisionTreeRegressor

import catboost as cb

import joblib

#Performance metrics

from sklearn.metrics import mean\_squared\_error,confusion\_matrix,roc\_auc\_score,RocCurveDisplay,auc,roc\_curve,ConfusionMatrixDisplay

from sklearn.metrics import precision\_recall\_fscore\_support,plot\_confusion\_matrix,log\_loss,accuracy\_score,f1\_score

#IMPORTING THE DATASET USING PANDAS LIBRARIES

df = pd.read\_csv("weatherAUS.csv")

#DISPLAYS THE FIRST FIVE ROWS OF THE DATA

df.head()

#DISPLAYS THE LAST FIVE ROWS OF THE DATA

df.tail()

#DESCRIPTION OF THE DATA

df.describe()

#INFORMATION OF DATA

df.info()

#SPLITTING OF DATA SEPARATELY INTO DAY,MONTH,YEAR

df['Date'] = pd.to\_datetime(df['Date'])

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

df['month'] = df['Date'].dt.month

df['day'] = df['Date'].dt.day

df.drop(['Date'], axis = 1,inplace=True)

df.head()

#DIMENSION OF THE DATASET

df.shape

#DESCRIPTION OF THE DATASET AFTER SPLIITING THE DATE

df.describe()

#CONVERTING THE FEATURES OF 'RainTomorrow' AND 'RainToday'INTO CATEGORICAL DATA

LE=LabelEncoder()

df['RainTomorrow']=LE.fit\_transform(df['RainTomorrow'])

df['RainToday']=LE.fit\_transform(df['RainToday'])

df['RainTomorrow']

#VISUALIZING THE IMBALANCE DATA

import matplotlib.pyplot as plt

#fig = plt.figure(figsize = (20,5))

ax=df['RainTomorrow'].value\_counts(normalize = True).plot(kind='bar', color= ['RED','navy'], alpha = 0.9, rot=0)

plt.title('RainTomorrow Indicator No(0) and Yes(1) in the Imbalanced Dataset')

for p in ax.patches:

ax.annotate(str(round(p.get\_height(),2)), (p.get\_x() \* 1.01 , p.get\_height() \* 1.01))

plt.show()

#BALANCING THE IMBALANCED DATA AND VISUALIZING IT

from sklearn.utils import resample

no = df[df['RainTomorrow'] == 0]

yes = df[df['RainTomorrow'] == 1]

yes\_oversampled = resample(yes, replace=True, n\_samples=len(no), random\_state=42)

df\_1 = pd.concat([no, yes\_oversampled])

#fig = plt.figure(figsize = (20,5))

ax=df\_1.RainTomorrow.value\_counts(normalize = True).plot(kind='bar', color= ['RED','navy'], alpha = 0.9, rot=0)

plt.title('RainTomorrow Indicator No(0) and Yes(1) after Oversampling (Balanced Dataset)')

for p in ax.patches:

ax.annotate(str(round(p.get\_height(),2)), (p.get\_x() \* 1.01 , p.get\_height() \* 1.01))

plt.show()

# MISSING DATA PATTERN IN TRAINING DATA

import seaborn as sns

plt.figure(figsize = (20,5))

sns.heatmap(df\_1.isnull(), cbar=False, cmap='PuBu')

plt.show()

#COLLECTING THE OBJECT DATATYPE OF THE FEATURES IN THE DATASET

lb=[i for i in df\_1.columns if(df\_1[i].dtype=='object')]

lb

df\_1['Location'].mode()[0]

# IMPUTE cATEGORICAL VARIABLE WITH MODE

df\_1['Location'] = df\_1['Location'].fillna(df\_1['Location'].mode()[0])

df\_1['WindGustDir'] = df\_1['WindGustDir'].fillna(df\_1['WindGustDir'].mode()[0])

df\_1['WindDir9am'] = df\_1['WindDir9am'].fillna(df\_1['WindDir9am'].mode()[0])

df\_1['WindDir3pm'] = df\_1['WindDir3pm'].fillna(df\_1['WindDir3pm'].mode()[0])

# CONVERTING THE OBJECT TYPE FEATURES INTO CATEGORIAL DATA USING LABEL ENCODING

from sklearn.preprocessing import LabelEncoder

lencoders = {}

for col in lb:

lencoders[col] = LabelEncoder()

df\_1[col] = lencoders[col].fit\_transform(df\_1[col])

# FILLING THE NULL VALUES WITH MEAN VALUES

import warnings

warnings.filterwarnings("ignore")

# Multiple Imputation by Chained Equations

from sklearn.experimental import enable\_iterative\_imputer

from sklearn.impute import IterativeImputer

df = df\_1.copy(deep=True)

mice\_imputer = IterativeImputer()

df.iloc[:, :] = mice\_imputer.fit\_transform(df\_1)

plt.figure(figsize=(27,15))

plt.title("Correlation Among Features")

sns.heatmap(df.corr(),linewidths=1,annot=True)

plt.show()

df.drop(['RISK\_MM','Location','year'],axis=1,inplace=True,)

plt.figure(figsize=(25,10))

plt.title("Correlation Among Features")

sns.heatmap(df.corr(),linewidths=1,annot=True)

plt.show()

y = df['RainTomorrow']

x = df.loc[:, df.columns != 'RainTomorrow']

# Split into test and train

from sklearn.model\_selection import train\_test\_split

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.2, shuffle=True)

dfx=pd.DataFrame()

dfx=pd.concat([x\_test,y\_test],axis=1)

dfx.head()

model\_rf= RandomForestClassifier(max\_depth=16,min\_samples\_leaf=1,min\_samples\_split=2,n\_estimators=100)

model\_rf.fit(x\_train, y\_train)

ypreds\_rf=model\_rf.predict(x\_test)

ypreds\_rf

model\_rf.score(x\_test,y\_test)

f2=f1\_score(y\_test,model\_rf.predict(x\_test))

lgls=log\_loss(y\_test, model\_rf.predict\_proba(x\_test))

acc=accuracy\_score(y\_test, model\_rf.predict(x\_test))

rac=roc\_auc\_score(y\_test, model\_rf.predict(x\_test))

f2,lgls,rac,acc

fpr, tpr, thresholds = roc\_curve(y\_test, ypreds\_rf)

roc\_auc = auc(fpr, tpr)

display =RocCurveDisplay(fpr=fpr, tpr=tpr, roc\_auc=roc\_auc,

estimator\_name='Randomforest')

display.plot(color='r')

plt.plot(list(np.arange(0,2,0.1)),list(np.arange(0,2,0.1)),color='k')

plt.title('ROC curve')

plt.xlim([0,1])

plt.ylim([0,1])

plt.show()

plt.show()

#Kfold

from sklearn.model\_selection import StratifiedKFold

folds=StratifiedKFold(n\_splits=5)

scores=[]

for i,j in folds.split(x,y):

X\_train, X\_test, Y\_train, Y\_test=x.iloc[i],x.iloc[j],y.iloc[i],y.iloc[j]

model1 = RandomForestClassifier(max\_depth=16,min\_samples\_leaf=1,min\_samples\_split=2,n\_estimators=100)

model1.fit(X\_train, Y\_train)

scores.append(model1.score(X\_test,Y\_test))

print(np.mean(scores))

disp = ConfusionMatrixDisplay(confusion\_matrix=confusion\_matrix(y\_test,model\_rf.predict(x\_test)),display\_labels=['yes','no'])

disp.plot()

plt.show()

joblib.dump(model\_rf,"RainPredictModelRF.sav")

warnings.filterwarnings("ignore")

model\_xgb =XGBClassifier(n\_estimators= 500,max\_depth= 16)

model\_xgb.fit(x\_train, y\_train)

ypreds\_xgb=model\_xgb.predict(x\_test)

ypreds\_xgb

model\_xgb.score(x\_test,y\_test)

disp = ConfusionMatrixDisplay(confusion\_matrix=confusion\_matrix(y\_test,model\_xgb.predict(x\_test)),display\_labels=['yes','no'])

disp.plot()

plt.show()

fpr, tpr, thresholds = roc\_curve(y\_test, ypreds\_xgb)

roc\_auc = auc(fpr, tpr)

display =RocCurveDisplay(fpr=fpr, tpr=tpr, roc\_auc=roc\_auc,

estimator\_name='Randomforest')

display.plot(color='r')

plt.plot(list(np.arange(0,2,0.1)),list(np.arange(0,2,0.1)),color='k')

plt.title('ROC curve')

plt.xlim([0,1])

plt.ylim([0,1])

plt.show()

plt.show()

joblib.dump(model\_xgb,"RainPredictModelXGB.sav")

#Kfold

from sklearn.model\_selection import StratifiedKFold

folds=StratifiedKFold(n\_splits=5)

scores=[]

for i,j in folds.split(x,y):

X\_train, X\_test, Y\_train, Y\_test=x.iloc[i],x.iloc[j],y.iloc[i],y.iloc[j]

model1 =XGBClassifier(n\_estimators= 500,max\_depth= 16)

model1.fit(X\_train, Y\_train)

scores.append(model1.score(X\_test,Y\_test))

print(np.mean(scores))

model\_cb = cb.CatBoostClassifier(iterations= 50,max\_depth=16)

model\_cb.fit(x\_train, y\_train)

ypreds\_cb=model\_cb.predict(x\_test)

ypreds\_cb

model\_cb.score(x\_test,y\_test)

f2=f1\_score(y\_test,model\_cb.predict(x\_test))

lgls=log\_loss(y\_test, model\_cb.predict\_proba(x\_test))

acc=accuracy\_score(y\_test, model\_cb.predict(x\_test))

rac=roc\_auc\_score(y\_test, model\_cb.predict(x\_test))

f2,lgls,rac,acc

disp = ConfusionMatrixDisplay(confusion\_matrix=confusion\_matrix(y\_test,model\_cb.predict(x\_test)),display\_labels=['yes','no'])

disp.plot()

plt.show()

joblib.dump(model\_cb,"RainPredictModelCB.sav")

fpr, tpr, thresholds = roc\_curve(y\_test, ypreds\_cb)

roc\_auc = auc(fpr, tpr)

display =RocCurveDisplay(fpr=fpr, tpr=tpr, roc\_auc=roc\_auc,

estimator\_name='Randomforest')

display.plot(color='r')

plt.plot(list(np.arange(0,2,0.1)),list(np.arange(0,2,0.1)),color='k')

plt.title('ROC curve')

plt.xlim([0,1])

plt.ylim([0,1])

plt.show()

plt.show()

#Kfold

from sklearn.model\_selection import StratifiedKFold

folds=StratifiedKFold(n\_splits=5)

scores=[]

for i,j in folds.split(x,y):

X\_train, X\_test, Y\_train, Y\_test=x.iloc[i],x.iloc[j],y.iloc[i],y.iloc[j]

model1 =cb.CatBoostClassifier(iterations= 50,max\_depth=16)

model1.fit(X\_train, Y\_train)

scores.append(model1.score(X\_test,Y\_test))

print(np.mean(scores))

**MODEL DEPLOYMENT**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder

#Performance metrics

from sklearn.metrics import mean\_squared\_error,confusion\_matrix,roc\_auc\_score,RocCurveDisplay,auc,roc\_curve,ConfusionMatrixDisplay

from sklearn.metrics import precision\_recall\_fscore\_support,plot\_confusion\_matrix,log\_loss,accuracy\_score,f1\_score

import joblib

df = pd.read\_csv("rainDS.csv")

df=df.drop(labels=['Unnamed: 0'],axis=1)

df.head()

y = df['RainTomorrow']

x = df.loc[:, df.columns != 'RainTomorrow']

rf\_model = joblib.load("RainPredictModelRF.sav")

ypreds=rf\_model.predict(x)

ypreds

plt.scatter(range(len(y)),y,c='r')

plt.scatter(range(len(ypreds)),ypreds,c='b')

plt.xlim(140,180)

plt.show()

rf\_model.score(x,y)

f2=f1\_score(y,rf\_model.predict(x))

lgls=log\_loss(y, rf\_model.predict\_proba(x))

acc\_rf=accuracy\_score(y, rf\_model.predict(x))

rac=roc\_auc\_score(y, rf\_model.predict(x))

f2,lgls,rac,acc\_rf

disp = ConfusionMatrixDisplay(confusion\_matrix=confusion\_matrix(y,rf\_model.predict(x)),display\_labels=['yes','no'])

disp.plot()

plt.show()

xgb\_model = joblib.load("RainPredictModelXGB.sav")

xgb\_model.predict(x)

xgb\_model.score(x,y)

f2=f1\_score(y,xgb\_model.predict(x))

lgls=log\_loss(y, xgb\_model.predict\_proba(x))

acc\_xgb=accuracy\_score(y, xgb\_model.predict(x))

rac=roc\_auc\_score(y, xgb\_model.predict(x))

f2,lgls,rac,acc\_xgb

disp = ConfusionMatrixDisplay(confusion\_matrix=confusion\_matrix(y,xgb\_model.predict(x)),display\_labels=['yes','no'])

disp.plot()

plt.show()

cb\_model = joblib.load("RainPredictModelCB.sav")

cb\_model.predict(x)

cb\_model.score(x,y)

f2=f1\_score(y,cb\_model.predict(x))

lgls=log\_loss(y, cb\_model.predict\_proba(x))

acc\_cb=accuracy\_score(y, cb\_model.predict(x))

rac=roc\_auc\_score(y, cb\_model.predict(x))

f2,lgls,rac,acc\_cb

disp = ConfusionMatrixDisplay(confusion\_matrix=confusion\_matrix(y,cb\_model.predict(x)),display\_labels=['yes','no'])

disp.plot()

plt.show()

def Ensemble\_Model(x,model1,model2,model3):

pred=[]

l1=model1.predict(x)

l2=model2.predict(x)

l3=model3.predict(x)

for i in range(len(l1)):

pred.append(max([l1[i],l2[i],l3[i]], key = [l1[i],l2[i],l3[i]].count))

pred=np.array(pred)

return pred

f2=f1\_score(y,Ensemble\_Model(x,rf\_model,xgb\_model,cb\_model))

lgls=log\_loss(y, Ensemble\_Model(x,rf\_model,xgb\_model,cb\_model))

acc\_en=accuracy\_score(y, Ensemble\_Model(x,rf\_model,xgb\_model,cb\_model))

rac=roc\_auc\_score(y, Ensemble\_Model(x,rf\_model,xgb\_model,cb\_model))

f2,lgls,rac,acc\_en

disp = ConfusionMatrixDisplay(confusion\_matrix=confusion\_matrix(y,Ensemble\_Model(x,rf\_model,xgb\_model,cb\_model)),display\_labels=['yes','no'])

disp.plot()

plt.show()

import seaborn as sns

l=[acc\_rf,acc\_xgb,acc\_cb,acc\_en]

l=[i\*100 for i in l]

n=['Random Forest','XGBoost','Catboost','Ensemble Model']

plt.figure(figsize=(8,5))

plt.bar(n,l,color=sns.color\_palette("crest"))

for i in range(4):

plt.text(i,l[i]//2,round(l[i],2),ha='center',Bbox = dict(facecolor = 'white', alpha =1))

plt.show()

GITHUB LINK: <https://github.com/chelluboinajyothiswarup/PREDICTION-OF-RAINFALL.git>