

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 Technology in Information Technology

By

KALVA POOJITHA
REG. NO. 40120042



DEPARTMENT OF INFORMATION TECHNOLOGY

SCHOOL OF COMPUTING

SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119, TAMILNADU

APRIL 2023



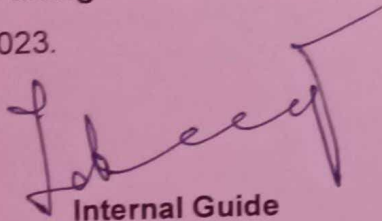
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC
(Established under Section 3 of UGC Act, 1956)
JEPPIAAR NAGAR, RAJIV GANDHI SALAI
CHENNAI- 600119
www.sathyabama.ac.in



DEPARTMENT OF INFORMATION TECHNOLOGY

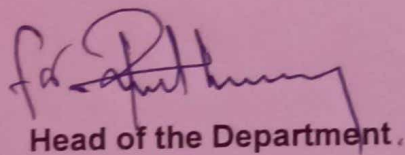
BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of KALVA POOJITHA(Reg. No: 40120042) who carried out the project entitled "Prediction of Rainfall using Machine Learning" under my supervision from January 2023 to April 2023.



Internal Guide

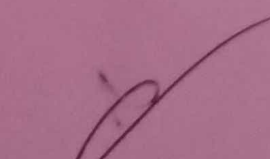
Dr.R Jeberson Retna Raj,M.Tech,Ph.D.,



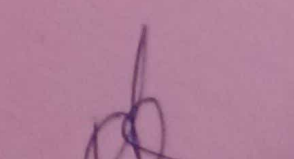
Head of the Department,

Dr. R.Subhashini, M.E., Ph.D.

Submitted for Viva voce Examination held on 28/04/23



Internal Examiner



External Examiner

DECLARATION

I, KALVA POOJITHA hereby declare that the project report entitled **PREDICTION OF RAINFALL USING MACHINE LEARNING** done by me under the guidance of **Dr.R Jeberson Retna Raj,M.Tech.,Ph.D.**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology Degree in Information Technology.

DATE: 28/04/23

PLACE: Chennai



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.R.Subhashini, M.E., Ph.D. Heads of the Department of Information Technology** 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 **Dr.Jeberson Retna Raj, M.Tech., 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 Information Technology** who were helpful in many ways for the completion of the project.

TRAINING CERTIFICATE



Elite

NPTEL Online Certification

(Funded by the MoE, Govt. of India)



This certificate is awarded to

KALVA POOJITHA

for successfully completing the course

Foundation of Cloud IoT Edge ML

with a consolidated score of **69** %

Online Assignments	23.54/25	Proctored Exam	45.75/75
--------------------	----------	----------------	----------

Total number of candidates certified in this course: **1649**

Prof. B. V. Ratish Kumar
Chairman, Centre for Continuing Education
IIT Kanpur

Feb-Apr 2023

(8 week course)

Prof. Satyaki Roy
NPTEL Coordinator
IIT Kanpur



Indian Institute of Technology Kanpur



Roll No: NPTEL23CS65S43364658

To validate the certificate



No. of credits recommended: 2 or 3

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. In this project 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 project 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 Positives
fp	False Positive
tn	True Negatives
fn	False Negatives

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.

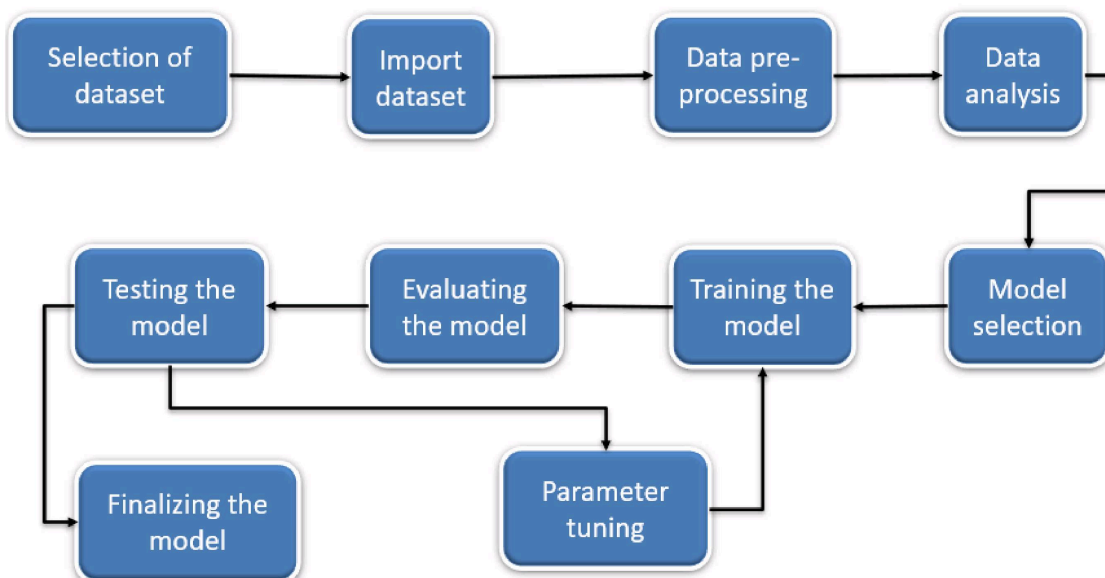


Fig:3.1: Architecture map of the project

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](#).
- **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

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	...	Humidity3pm	Pressure9am
0	2008-12-01	Albury	13.4	22.9	0.6	NaN	NaN	W	44.0	W	...	22.0	1007.7
1	2008-12-02	Albury	7.4	25.1	0.0	NaN	NaN	WNW	44.0	NNW	...	25.0	1010.6
2	2008-12-03	Albury	12.9	25.7	0.0	NaN	NaN	WSW	46.0	W	...	30.0	1007.6
3	2008-12-04	Albury	9.2	28.0	0.0	NaN	NaN	NE	24.0	SE	...	16.0	1017.6
4	2008-12-05	Albury	17.5	32.3	1.0	NaN	NaN	W	41.0	ENE	...	33.0	1010.8

5 rows × 24 columns

Pressure3pm	Cloud9am	Cloud3pm	Temp9am	Temp3pm	RainToday	RISK_MM	RainTomorrow
1007.1	8.0	NaN	16.9	21.8	No	0.0	No
1007.8	NaN	NaN	17.2	24.3	No	0.0	No
1008.7	NaN	2.0	21.0	23.2	No	0.0	No
1012.8	NaN	NaN	18.1	26.5	No	1.0	No
1006.0	7.0	8.0	17.8	29.7	No	0.2	No

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*.

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 142193 entries, 0 to 142192
Data columns (total 24 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Date                                  142193 non-null object
1   Location                             142193 non-null object
2   MinTemp                             141556 non-null float64
3   MaxTemp                             141871 non-null float64
4   Rainfall                            140787 non-null float64
5   Evaporation                         81350 non-null float64
6   Sunshine                            74377 non-null float64
7   WindGustDir                         132863 non-null object
8   WindGustSpeed                       132923 non-null float64
9   WindDir9am                         132180 non-null object
10  WindDir3pm                         138415 non-null object
11  WindSpeed9am                       140845 non-null float64
12  WindSpeed3pm                       139563 non-null float64
13  Humidity9am                        140419 non-null float64
14  Humidity3pm                       138583 non-null float64
15  Pressure9am                       128179 non-null float64
16  Pressure3pm                       128212 non-null float64
17  Cloud9am                          88536 non-null float64
18  Cloud3pm                          85099 non-null float64
19  Temp9am                           141289 non-null float64
20  Temp3pm                           139467 non-null float64
21  RainToday                         140787 non-null object
22  RISK_MM                           142193 non-null float64
23  RainTomorrow                      142193 non-null object
dtypes: float64(17), object(7)
memory usage: 26.0+ MB

```

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

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm
count	141556.000000	141871.000000	140787.000000	81350.000000	74377.000000	132923.000000	140845.000000	139563.000000	140419.000000	138583.000000
mean	12.186400	23.226784	2.349974	5.469824	7.624853	39.984292	14.001988	18.637576	68.843810	51.482606
std	6.403283	7.117618	8.465173	4.188537	3.781525	13.588801	8.893337	8.803345	19.051293	20.797772
min	-8.500000	-4.800000	0.000000	0.000000	0.000000	6.000000	0.000000	0.000000	0.000000	0.000000
25%	7.600000	17.900000	0.000000	2.600000	4.900000	31.000000	7.000000	13.000000	57.000000	37.000000
50%	12.000000	22.600000	0.000000	4.800000	8.500000	39.000000	13.000000	19.000000	70.000000	52.000000
75%	16.800000	28.200000	0.800000	7.400000	10.600000	48.000000	19.000000	24.000000	83.000000	66.000000
max	33.900000	48.100000	371.000000	145.000000	14.500000	135.000000	130.000000	87.000000	100.000000	100.000000
	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm	Temp9am	Temp3pm	RISK_MM	year	month	day
128179.000000	128212.000000	88536.000000	85099.000000	141289.000000	139467.000000	142193.000000	142193.000000	142193.000000	142193.000000	142193.000000
1017.653758	1015.258204	4.437189	4.503167	16.987509	21.687235	2.360682	2012.758926	6.402544	15.715084	
7.105476	7.036677	2.887016	2.720633	6.492838	6.937594	8.477969	2.541256	3.426506	8.798150	
980.500000	977.100000	0.000000	0.000000	-7.200000	-5.400000	0.000000	2007.000000	1.000000	1.000000	
1012.900000	1010.400000	1.000000	2.000000	12.300000	16.600000	0.000000	2011.000000	3.000000	8.000000	
1017.600000	1015.200000	5.000000	5.000000	16.700000	21.100000	0.000000	2013.000000	6.000000	16.000000	
1022.400000	1020.000000	7.000000	7.000000	21.600000	26.400000	0.800000	2015.000000	9.000000	23.000000	
1041.000000	1039.600000	9.000000	9.000000	40.200000	46.700000	371.000000	2017.000000	12.000000	31.000000	

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.

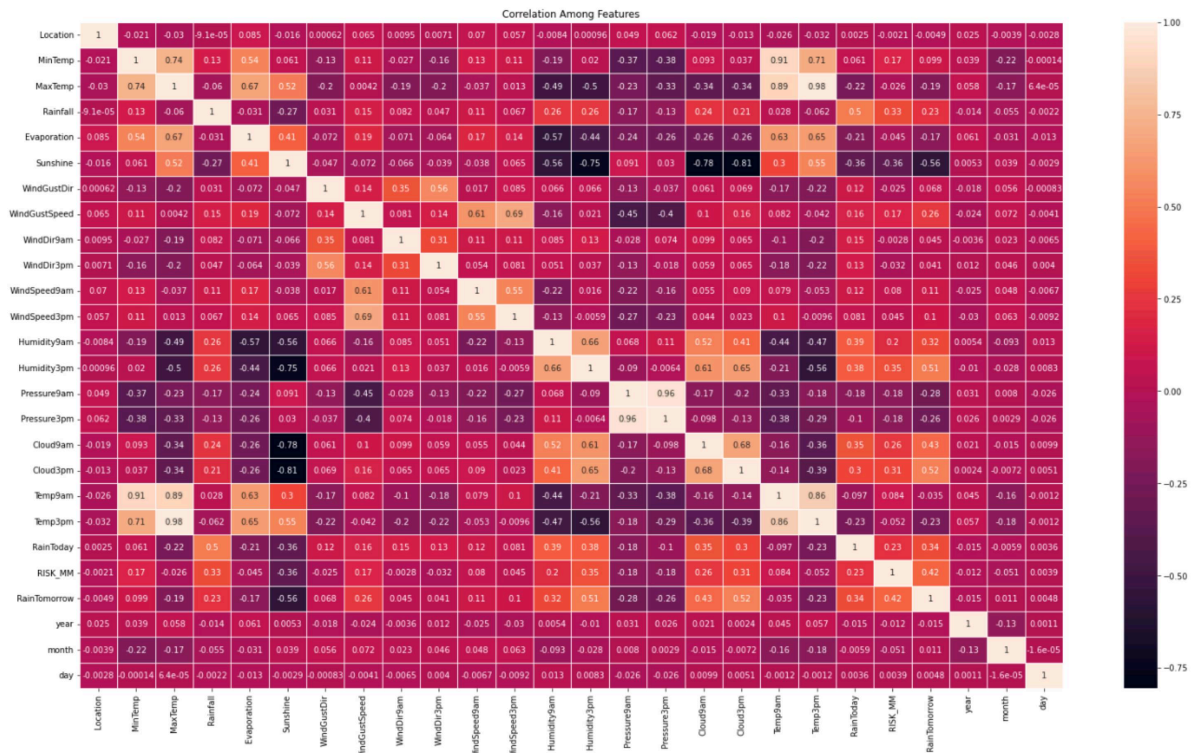


Figure 3.3: correlation of the dataset

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 vital for 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.

$$f(x) = y$$

$F(x)$ 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.

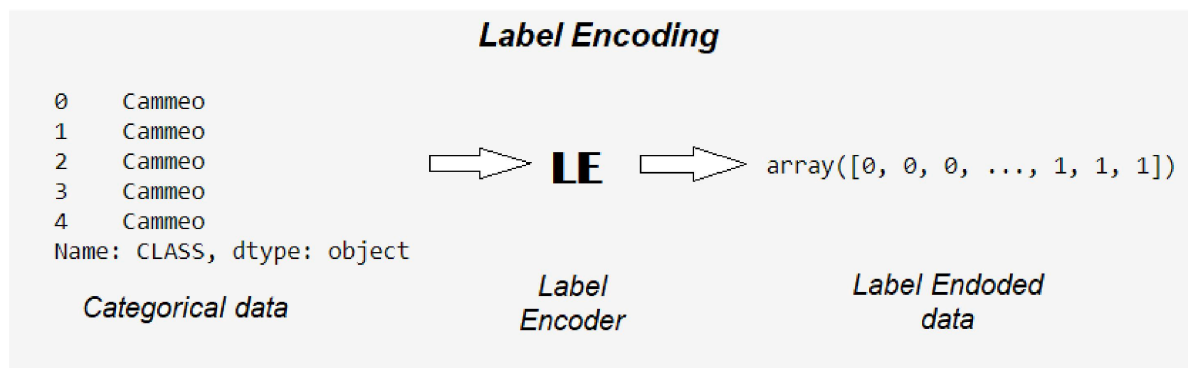


Fig:3.4: Label encoding process of categorical data

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.

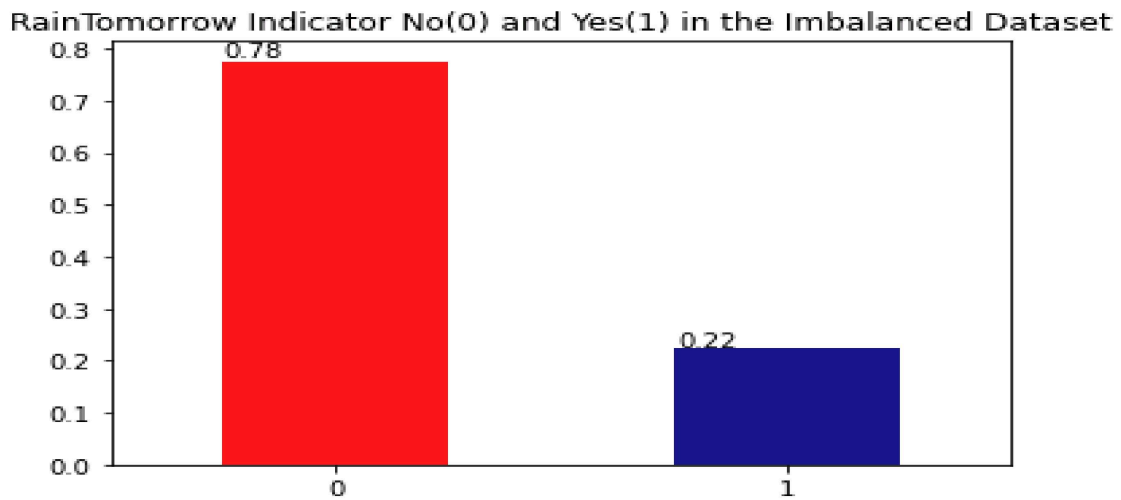


Fig3.5: *RainTomorrow indicator in imbalanced dataset*

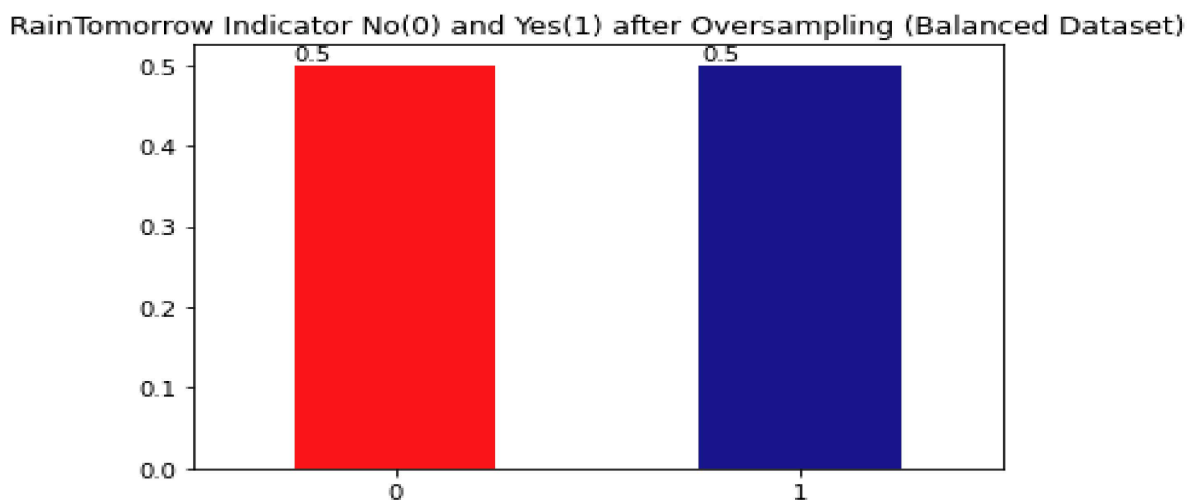


Fig 3.6: *RainTomorrow indicator in balanced dataset*

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.

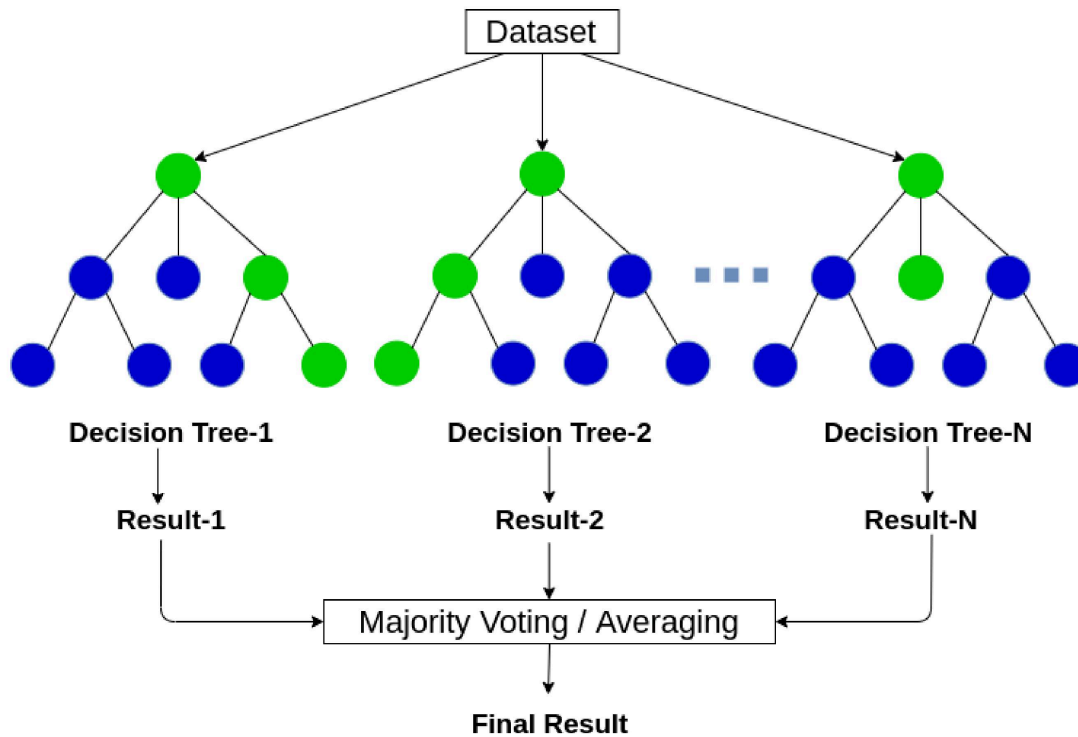


Fig:3.7: Random Forest general structure

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.

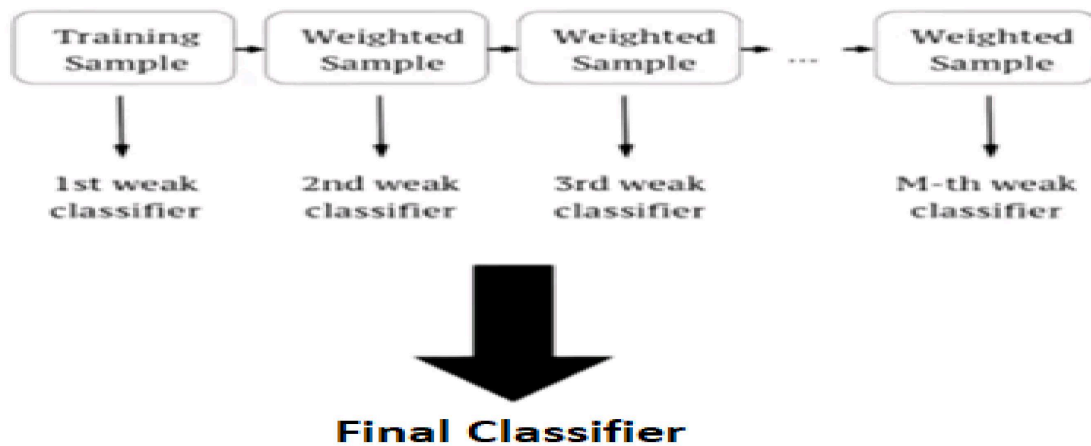


Fig:3.8: XGBoost classifier

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.

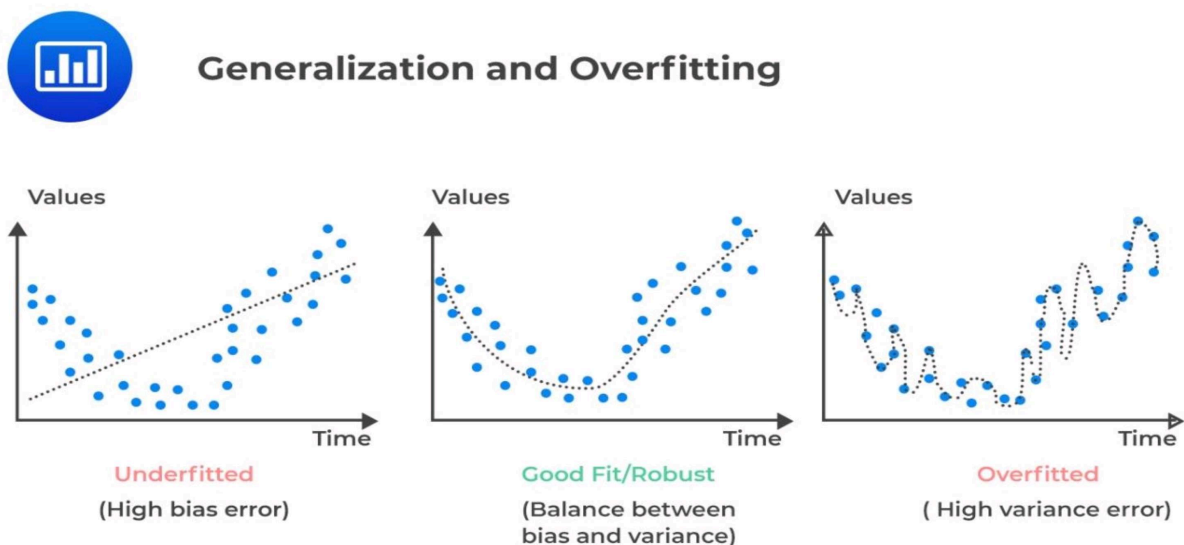


Fig:3.9: Model complexity factors

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 vary 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.

Time complexity $\propto$ Max depth

Time complexity $\propto \frac{1}{\text{Min sample leaf}}$

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 C_{ij} 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 $C_{0,0}$, false negatives are $C_{1,0}$, true positives are $C_{1,1}$, and false positives are $C_{0,1}$. The tp, fp, tn, fn are shown in figure 3.10.

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

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	$ACC = \frac{tp + tn}{tp + fp + tn + fn} \times 100$
2	Sensitivity	$TPR = \frac{tp}{tp + fn} \times 100$
3	Specificity	$SPC = \frac{tn}{tn + fp} \times 100$
4	Precision	$PPV = \frac{tp}{tp + fp} \times 100$
5	F1-Score	$F1 = \frac{2tp}{2tp + fp + fn} \times 100$
6	Negative Predictive Value	$NPV = \frac{tn}{tn + fn} \times 100$
7	Negative Positive Rate	$FPR = \frac{fp}{tn + fp} \times 100$
8	False Discovery Rate	$FDR = \frac{fp}{tp + fp} \times 100$
9	False Negative Rate	$FNR = \frac{fn}{tp + fn} \times 100$

- 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.

$$F1\ Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

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 $y \in \{0,1\}$ and a probability estimate $p = Pr(y = 1)$, the log loss is defined as:

$$L_{\log}(y, p) = -(y \log(p) + (1 - y) \log(1 - p))$$

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.

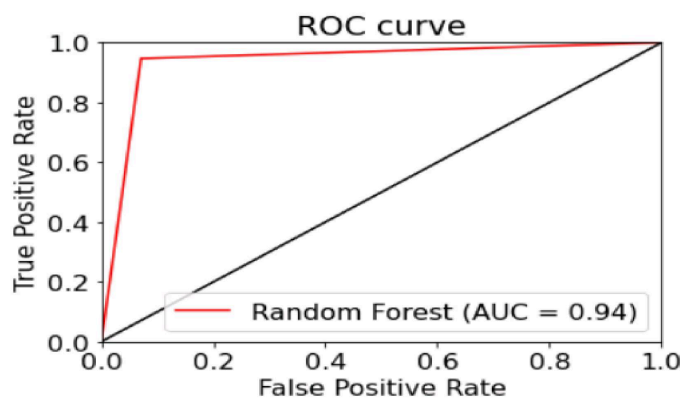


Fig:3.11: ROC curve of RF model

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 categorical 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 ensemble 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.

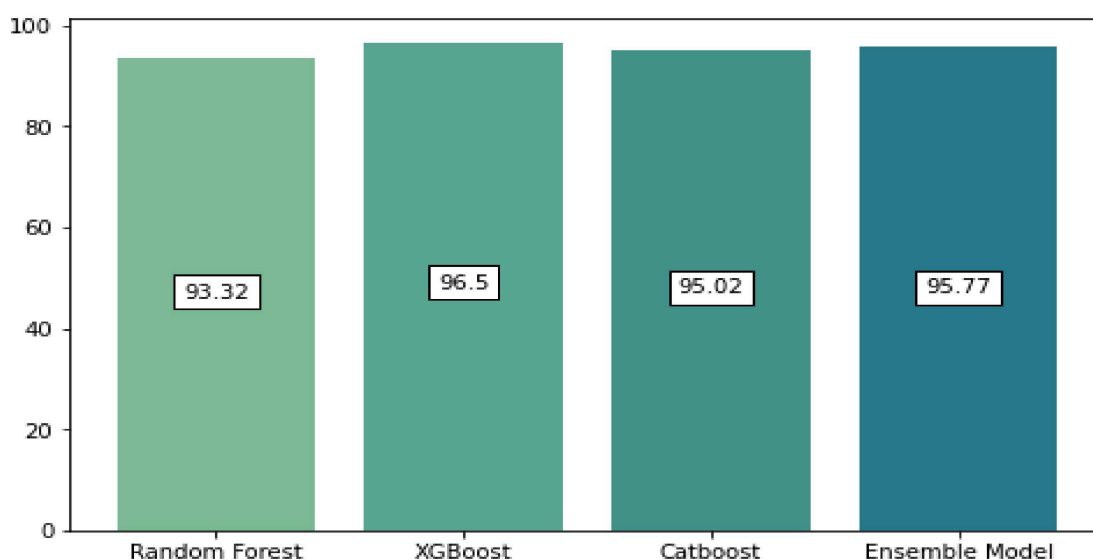


Fig 4.1: visualizing the accuracy score

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

A. SCREENSHOTS

IMPORTING LIBRARIES

```
1 %matplotlib inline
2 #Visualization Libraries
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5
6 #Data processing Libraries
7 import numpy as np
8 import pandas as pd
9 from sklearn.model_selection import train_test_split
10 from sklearn.preprocessing import LabelEncoder
11 from math import sqrt
12 #from sklearn.feature_selection import VarianceThreshold
13 import warnings
14
15 #Machine model Algorithm module
16 from sklearn.ensemble import RandomForestClassifier
17 from xgboost import XGBClassifier
18 #from sklearn.tree import plot_tree, DecisionTreeRegressor
19 import catboost as cb
20 import joblib
21
22 #Performance metrics
23 from sklearn.metrics import mean_squared_error, confusion_matrix, roc_auc_score, RocCurveDisplay, auc, roc_curve, ConfusionMatrixDisplay
24 from sklearn.metrics import precision_recall_fscore_support, plot_confusion_matrix, log_loss, accuracy_score, f1_score
```

IMPORTING THE DATA

```
1 #IMPORTING THE DATASET USING PANDAS LIBRARIES
2 df = pd.read_csv("weatherAUS.csv")
```

```
1 #DISPLAYS THE FIRST FIVE ROWS OF THE DATA
2 df.head()
```

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	...	Humidity3pm	Pressure9am	Pressure3
0	2008-12-01	Albury	13.4	22.9	0.6	NaN	NaN	W	44.0	W	...	22.0	1007.7	100
1	2008-12-02	Albury	7.4	25.1	0.0	NaN	NaN	WNW	44.0	NNW	...	25.0	1010.6	100
2	2008-12-03	Albury	12.9	25.7	0.0	NaN	NaN	WSW	46.0	W	...	30.0	1007.6	100
3	2008-12-04	Albury	9.2	28.0	0.0	NaN	NaN	NE	24.0	SE	...	16.0	1017.6	101
4	2008-12-05	Albury	17.5	32.3	1.0	NaN	NaN	W	41.0	ENE	...	33.0	1010.8	100

5 rows × 24 columns

```
1 #DISPLAYS THE LAST FIVE ROWS OF THE DATA
2 df.tail()
```

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	...	Humidity3pm	Pressure9am	Pressure3
142188	2017-06-20	Uluru	3.5	21.8	0.0	NaN	NaN	E	31.0	ESE	...	27.0	1024.7	
142189	2017-06-21	Uluru	2.8	23.4	0.0	NaN	NaN	E	31.0	SE	...	24.0	1024.6	
142190	2017-06-22	Uluru	3.6	25.3	0.0	NaN	NaN	NNW	22.0	SE	...	21.0	1023.5	
142191	2017-06-23	Uluru	5.4	26.9	0.0	NaN	NaN	N	37.0	SE	...	24.0	1021.0	
142192	2017-06-24	Uluru	7.8	27.0	0.0	NaN	NaN	SE	28.0	SSE	...	24.0	1019.4	

5 rows × 24 columns

```
1 #DESCRIPTION OF THE DATA
2 df.describe()
```

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm
count	141556.000000	141871.000000	140787.000000	81350.000000	74377.000000	132923.000000	140845.000000	139563.000000	140419.000000	138583.000000
mean	12.186400	23.226784	2.349974	5.469824	7.624853	39.984292	14.001988	18.637576	68.843810	51.482691
std	6.403283	7.117618	8.465173	4.188537	3.781525	13.588801	8.893337	8.803345	19.051293	20.797701
min	-8.500000	-4.800000	0.000000	0.000000	0.000000	6.000000	0.000000	0.000000	0.000000	0.000000
25%	7.600000	17.900000	0.000000	2.600000	4.900000	31.000000	7.000000	13.000000	57.000000	37.000000
50%	12.000000	22.600000	0.000000	4.800000	8.500000	39.000000	13.000000	19.000000	70.000000	52.000000
75%	16.800000	28.200000	0.800000	7.400000	10.600000	48.000000	19.000000	24.000000	83.000000	66.000000
max	33.900000	48.100000	371.000000	145.000000	14.500000	135.000000	130.000000	87.000000	100.000000	100.000000

```
1 #INFORMATION OF DATA
2 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 142193 entries, 0 to 142192
Data columns (total 24 columns):
#   Column                                  Non-Null Count  Dtype
---  -
0   Date                                   142193 non-null object
1   Location                               142193 non-null object
2   MinTemp                               141556 non-null float64
3   MaxTemp                               141871 non-null float64
4   Rainfall                              140787 non-null float64
5   Evaporation                           81350 non-null float64
6   Sunshine                              74377 non-null float64
7   WindGustDir                           132863 non-null object
8   WindGustSpeed                         132923 non-null float64
9   WindDir9am                            132180 non-null object
10  WindDir3pm                             138415 non-null object
11  WindSpeed9am                           140845 non-null float64
12  WindSpeed3pm                           139563 non-null float64
13  Humidity9am                            140419 non-null float64
14  Humidity3pm                            138583 non-null float64
15  Pressure9am                            128179 non-null float64
16  Pressure3pm                            128212 non-null float64
17  Cloud9am                               88536 non-null float64
18  Cloud3pm                               85099 non-null float64
19  Temp9am                                141289 non-null float64
20  Temp3pm                                139467 non-null float64
21  RainToday                              140787 non-null object
22  RISK_MM                                142193 non-null float64
23  RainTomorrow                           142193 non-null object
dtypes: float64(17), object(7)
memory usage: 26.0+ MB
```

```
1 #SPLITTING OF DATA SEPARATELY INTO DAY,MONTH, YEAR
2 df['Date'] = pd.to_datetime(df['Date'])
3 df['year'] = df['Date'].dt.year
4 df['month'] = df['Date'].dt.month
5 df['day'] = df['Date'].dt.day
6 df.drop(['Date'], axis = 1,inplace=True)
7 df.head()
```

	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	...	Cloud9am	Cloud3pm	Temp9am
0	Albury	13.4	22.9	0.6	NaN	NaN	W	44.0	W	WNW	...	8.0	NaN	16.9
1	Albury	7.4	25.1	0.0	NaN	NaN	WNW	44.0	NNW	WSW	...	NaN	NaN	17.2
2	Albury	12.9	25.7	0.0	NaN	NaN	WSW	46.0	W	WSW	...	NaN	2.0	21.0
3	Albury	9.2	28.0	0.0	NaN	NaN	NE	24.0	SE	E	...	NaN	NaN	18.1
4	Albury	17.5	32.3	1.0	NaN	NaN	W	41.0	ENE	NW	...	7.0	8.0	17.8

5 rows × 26 columns


```

1 #CONVERTING THE FEATURES OF 'RainTomorrow' AND 'RainToday' INTO CATEGORICAL DATA
2 LE=LabelEncoder()
3 df['RainTomorrow']=LE.fit_transform(df['RainTomorrow'])
4 df['RainToday']=LE.fit_transform(df['RainToday'])
5 df['RainTomorrow']

```

```

0      0
1      0
2      0
3      0
4      0
..
142188  0
142189  0
142190  0
142191  0
142192  0

```

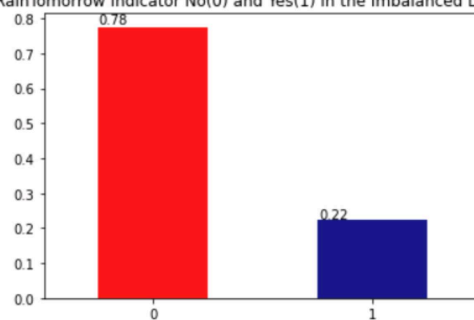
Name: RainTomorrow, Length: 142193, dtype: int32

```

1 #VISUALIZING THE IMBALANCE DATA
2 import matplotlib.pyplot as plt
3 #fig = plt.figure(figsize = (20,5))
4 ax=df['RainTomorrow'].value_counts(normalize = True).plot(kind='bar', color= ['RED','navy'], alpha = 0.9, rot=0)
5 plt.title('RainTomorrow Indicator No(0) and Yes(1) in the Imbalanced Dataset')
6 for p in ax.patches:
7     ax.annotate(str(round(p.get_height(),2)), (p.get_x() * 1.01 , p.get_height() * 1.01))
8 plt.show()

```

RainTomorrow Indicator No(0) and Yes(1) in the Imbalanced Dataset

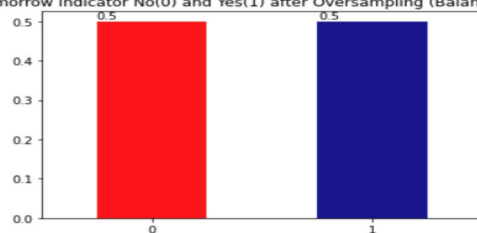


```

1 #BALANCING THE IMBALANCED DATA AND VISUALIZING IT
2 from sklearn.utils import resample
3
4 no = df[df['RainTomorrow'] == 0]
5 yes = df[df['RainTomorrow'] == 1]
6 yes_oversampled = resample(yes, replace=True, n_samples=len(no), random_state=42)
7 df_1 = pd.concat([no, yes_oversampled])
8
9 #fig = plt.figure(figsize = (20,5))
10 ax=df_1.RainTomorrow.value_counts(normalize = True).plot(kind='bar', color= ['RED','navy'], alpha = 0.9, rot=0)
11 plt.title('RainTomorrow Indicator No(0) and Yes(1) after Oversampling (Balanced Dataset)')
12 for p in ax.patches:
13     ax.annotate(str(round(p.get_height(),2)), (p.get_x() * 1.01 , p.get_height() * 1.01))
14 plt.show()

```

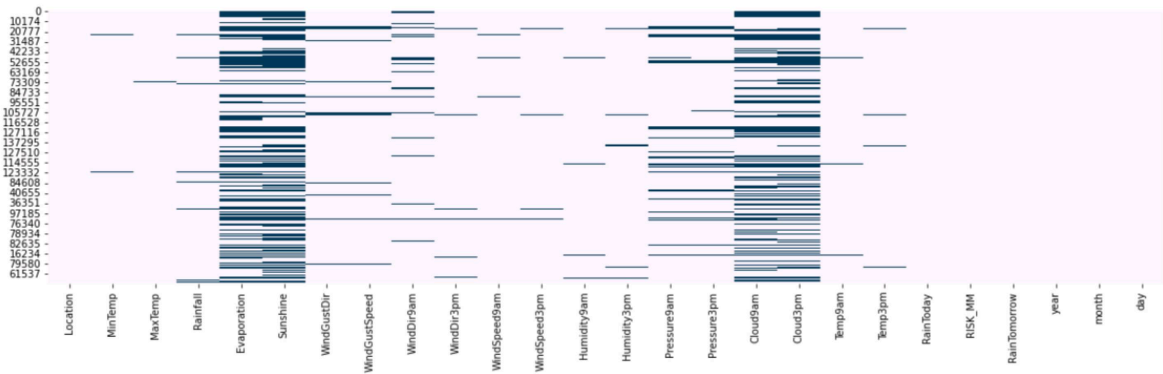
RainTomorrow Indicator No(0) and Yes(1) after Oversampling (Balanced Dataset)




```

1 # MISSING DATA PATTERN IN TRAINING DATA
2 import seaborn as sns
3 plt.figure(figsize = (20,5))
4 sns.heatmap(df_1.isnull(), cbar=False, cmap='PuBu')
5 plt.show()

```



```

1 #COLLECTING THE OBJECT DATATYPE OF THE FEATURES IN THE DATASET
2 lb=[i for i in df_1.columns if(df_1[i].dtype=='object')]
3 lb

```

```
['Location', 'WindGustDir', 'WindDir9am', 'WindDir3pm']
```

```
1 df_1['Location'].mode()[0]
```

```
'Portland'
```

```

1 # IMPUTE CATEGORICAL VARIABLE WITH MODE
2 df_1['Location'] = df_1['Location'].fillna(df_1['Location'].mode()[0])
3 df_1['WindGustDir'] = df_1['WindGustDir'].fillna(df_1['WindGustDir'].mode()[0])
4 df_1['WindDir9am'] = df_1['WindDir9am'].fillna(df_1['WindDir9am'].mode()[0])
5 df_1['WindDir3pm'] = df_1['WindDir3pm'].fillna(df_1['WindDir3pm'].mode()[0])

```

```

1 # CONVERTING THE OBJECT TYPE FEATURES INTO CATEGORIAL DATA USING LABEL ENCODING
2 from sklearn.preprocessing import LabelEncoder
3 lencoders = {}
4 for col in lb:
5     lencoders[col] = LabelEncoder()
6     df_1[col] = lencoders[col].fit_transform(df_1[col])

```

```

1 #INFORMATION OF THE DATASET AFTER LABEL ENCODING
2 df_1.info()

```

```

<class 'pandas.core.frame.DataFrame'>
Int64Index: 220632 entries, 0 to 23751
Data columns (total 26 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Location              220632 non-null  int32
1   MinTemp               219542 non-null  float64
2   MaxTemp               220151 non-null  float64
3   Rainfall              217606 non-null  float64
4   Evaporation           125307 non-null  float64
5   Sunshine              115921 non-null  float64
6   WindGustDir           220632 non-null  int32
7   WindGustSpeed         205400 non-null  float64
8   WindDir9am            220632 non-null  int32
9   WindDir3pm            220632 non-null  int32
10  WindSpeed9am          218470 non-null  float64
11  WindSpeed3pm          216354 non-null  float64
12  Humidity9am           217547 non-null  float64
13  Humidity3pm           214620 non-null  float64
14  Pressure9am           199012 non-null  float64
15  Pressure3pm           199013 non-null  float64
16  Cloud9am              139244 non-null  float64
17  Cloud3pm              134974 non-null  float64
18  Temp9am               219002 non-null  float64
19  Temp3pm               216062 non-null  float64
20  RainToday             220632 non-null  int32
21  RISK_MM               220632 non-null  float64
22  RainTomorrow          220632 non-null  int32
23  year                  220632 non-null  int64
24  month                 220632 non-null  int64
25  day                   220632 non-null  int64
dtypes: float64(17), int32(6), int64(3)

```

```

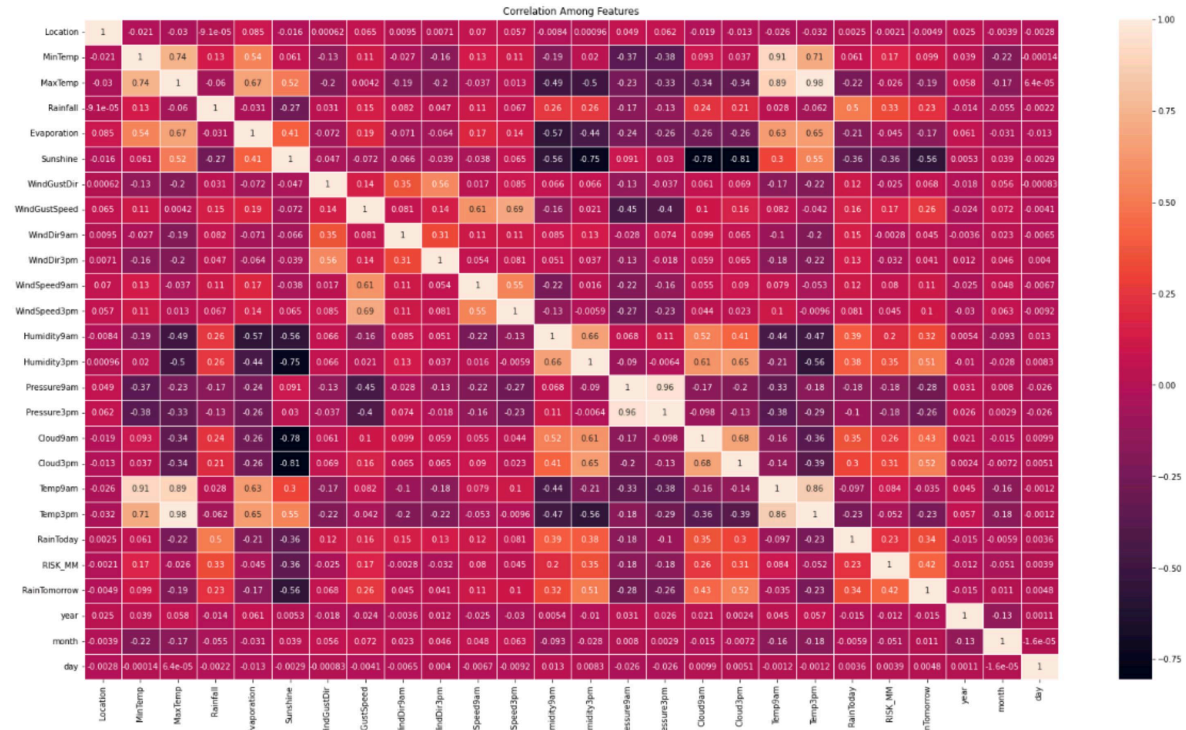
1 # FILLING THE NULL VALUES WITH MEAN VALUES
2 import warnings
3 warnings.filterwarnings("ignore")
4 # Multiple Imputation by Chained Equations
5 from sklearn.experimental import enable_iterative_imputer
6 from sklearn.impute import IterativeImputer
7 df = df_1.copy(deep=True)
8 mice_imputer = IterativeImputer()
9 df.iloc[:, :] = mice_imputer.fit_transform(df_1)

```

```

1 plt.figure(figsize=(27,15))
2 plt.title("Correlation Among Features")
3 sns.heatmap(df.corr(),linewidths=1,annot=True)
4 plt.show()

```



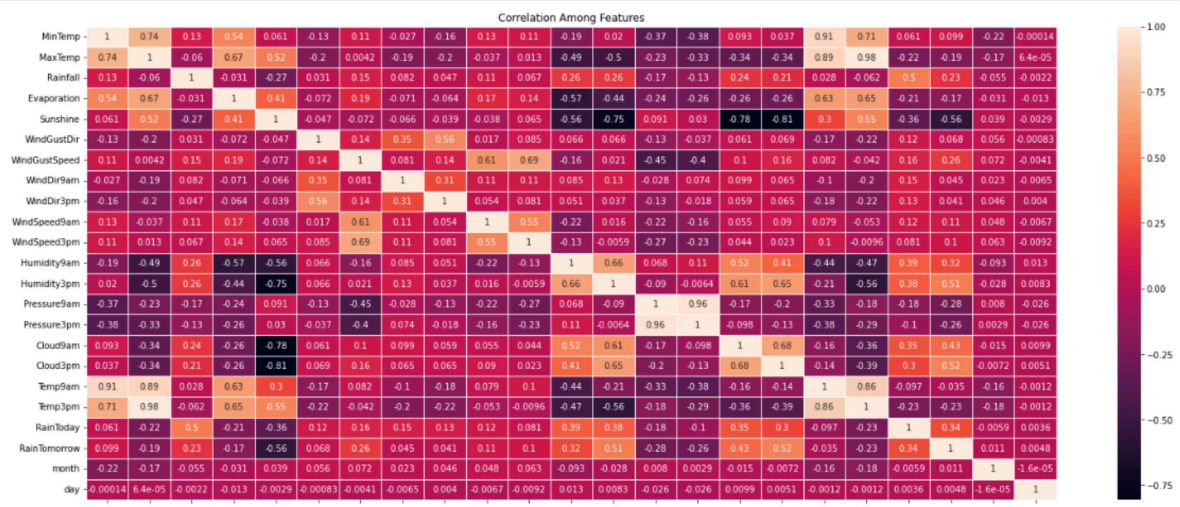
```
# DROPPING THE FEATURES
```

```
df.drop(['RISK_MM', 'Location', 'year'],axis=1,inplace=True,)
```

```

1 #CORRELATION OF THE DATASET AFTER DROPPING THE VALUES
2 plt.figure(figsize=(25,10))
3 plt.title("Correlation Among Features")
4 sns.heatmap(df.corr(),linewidths=1,annot=True)
5 plt.show()

```



SPLITTING THE DATA INTO TRAINING AND TESTING DATA

```
1 y = df['RainTomorrow']
2 x = df.loc[:, df.columns != 'RainTomorrow']
3 # Split into test and train
4 from sklearn.model_selection import train_test_split
5 x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, shuffle=True)
```

```
1 dfx=pd.DataFrame()
2 dfx=pd.concat([x_test,y_test],axis=1)
3 dfx.head()
```

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	...	Pressure9am	Press
10699	21.6	26.3	2.6	6.600000	3.415322	4.0	31.0	3.0	4.0	0.0	...	1015.800000	1012
85596	24.0	30.8	1.0	6.600000	1.800000	13.0	39.0	3.0	6.0	11.0	...	1007.600000	1004
37007	4.7	23.1	0.0	3.200000	10.700000	2.0	17.0	0.0	14.0	11.0	...	1032.800000	1027
48185	23.0	35.6	0.0	11.668776	11.943360	1.0	50.0	6.0	14.0	19.0	...	1006.700000	1006
132246	2.6	18.5	0.0	2.040960	8.049468	15.0	26.0	5.0	6.0	6.0	...	1020.373128	1016

5 rows × 23 columns



```
1 dfx.shape
```

(44127, 23)

```
1 model_rf= RandomForestClassifier(max_depth=16,min_samples_leaf=1,min_samples_split=2,n_estimators=100)
2 model_rf.fit(x_train, y_train)
```

RandomForestClassifier(max_depth=16)

```
1 ypreds_rf=model_rf.predict(x_test)
2 ypreds_rf
```

array([1., 1., 0., ..., 1., 1., 0.])

```
1 model_rf.score(x_test,y_test)
```

0.9291136945634192

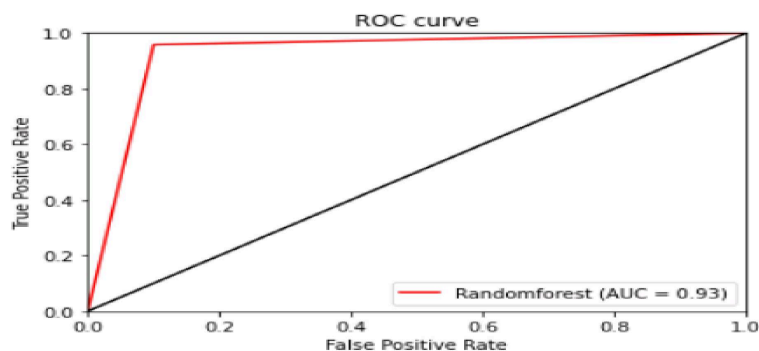
```
1 f2=f1_score(y_test,model_rf.predict(x_test))
2 lgls=log_loss(y_test, model_rf.predict_proba(x_test))
3 acc=accuracy_score(y_test, model_rf.predict(x_test))
4 rac=roc_auc_score(y_test, model_rf.predict(x_test))
5 f2,lgls,rac,acc
```

(0.931213440647403,
0.22006622667228898,
0.9290732722973358,
0.9291136945634192)


```

1 fpr, tpr, thresholds = roc_curve(y_test, ypreds_rf)
2 roc_auc = auc(fpr, tpr)
3 display = RocCurveDisplay(fpr=fpr, tpr=tpr, roc_auc=roc_auc,
4                             estimator_name='Randomforest')
5
6 display.plot(color='r')
7 plt.plot(list(np.arange(0,2,0.1)),list(np.arange(0,2,0.1)),color='k')
8 plt.title('ROC curve')
9 plt.xlim([0,1])
10 plt.ylim([0,1])
11 plt.show()
12 plt.show()

```



K-FOLD VALIDATION

```

1 #Kfold
2 from sklearn.model_selection import StratifiedKFold
3 folds=StratifiedKFold(n_splits=5)
4 scores=[]
5 for i,j in folds.split(x,y):
6     X_train, X_test, Y_train, Y_test=x.iloc[i],x.iloc[j],y.iloc[i],y.iloc[j]
7     model1 = RandomForestClassifier(max_depth=16,min_samples_leaf=1,min_samples_split=2,n_estimators=100)
8     model1.fit(X_train, Y_train)
9     scores.append(model1.score(X_test,Y_test))
10
11 print(np.mean(scores))

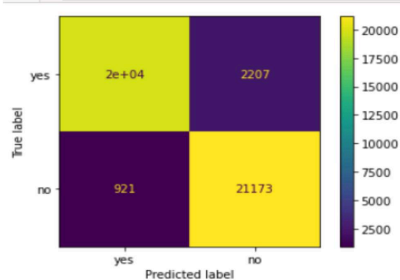
```

0.9101535229883362

```

1 disp = ConfusionMatrixDisplay(confusion_matrix=confusion_matrix(y_test,model_rf.predict(x_test)),display_labels=['yes','no'])
2 disp.plot()
3 plt.show()

```



```
1 joblib.dump(model_rf,"RainPredictModelRF.sav")
```

['RainPredictModelRF.sav']

XGBoostClassifier

```
1 warnings.filterwarnings("ignore")
2 model_xgb = XGBClassifier(n_estimators=500, max_depth=16)
3 model_xgb.fit(x_train, y_train)
```

[15:14:40] WARNING: ..\src\learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
              gamma=0, gpu_id=-1, importance_type=None,
              interaction_constraints='', learning_rate=0.300000012,
              max_delta_step=0, max_depth=16, min_child_weight=1, missing=nan,
              monotone_constraints=(), n_estimators=500, n_jobs=8,
              num_parallel_tree=1, predictor='auto', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

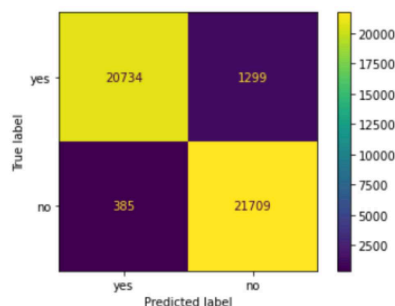
```
1 ypreds_xgb = model_xgb.predict(x_test)
2 ypreds_xgb
```

```
array([1., 1., 0., ..., 1., 1., 0.])
```

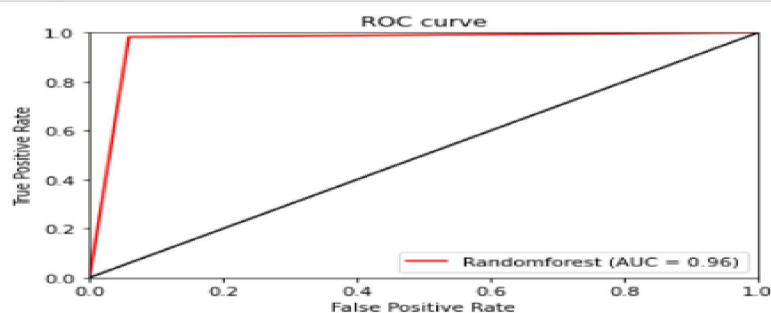
```
1 model_xgb.score(x_test, y_test)
```

```
0.9618374237994879
```

```
1 isp = ConfusionMatrixDisplay(confusion_matrix=confusion_matrix(y_test, model_xgb.predict(x_test)), display_labels=['yes', 'no'])
2 isp.plot()
3 lt.show()
```



```
1 fpr, tpr, thresholds = roc_curve(y_test, ypreds_xgb)
2 roc_auc = auc(fpr, tpr)
3 display = RocCurveDisplay(fpr=fpr, tpr=tpr, roc_auc=roc_auc,
4                           estimator_name='Randomforest')
5 display.plot(color='r')
6 plt.plot(list(np.arange(0,2,0.1)), list(np.arange(0,2,0.1)), color='k')
7 plt.title('ROC curve')
8 plt.xlim([0,1])
9 plt.ylim([0,1])
10 plt.show()
11 plt.show()
```



```
1 joblib.dump(model_xgb, "RainPredictModelXGB.sav")
```

```
['RainPredictModelXGB.sav']
```

```

1 #Kfold
2 from sklearn.model_selection import StratifiedKFold
3 folds=StratifiedKFold(n_splits=5)
4 scores=[]
5 for i,j in folds.split(x,y):
6     X_train, X_test, Y_train, Y_test=x.iloc[i],x.iloc[j],y.iloc[i],y.iloc[j]
7     model1 =XGBClassifier(n_estimators= 500,max_depth= 16)
8     model1.fit(X_train, Y_train)
9     scores.append(model1.score(X_test,Y_test))
10 print(np.mean(scores))

```

[15:23:18] WARNING: ..\src\learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
[15:26:18] WARNING: ..\src\learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
[15:29:44] WARNING: ..\src\learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
[15:33:00] WARNING: ..\src\learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
[15:35:58] WARNING: ..\src\learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
0.9279388059583157

```

1 model_cb = cb.CatBoostClassifier(iterations= 50,max_depth=16)
2 model_cb.fit(x_train, y_train)

```

32:	learn: 0.0434019	total: 3m 25s	remaining: 1m 45s
33:	learn: 0.0437241	total: 3m 31s	remaining: 1m 39s
34:	learn: 0.0425976	total: 3m 37s	remaining: 1m 33s
35:	learn: 0.0403585	total: 3m 43s	remaining: 1m 27s
36:	learn: 0.0377952	total: 3m 50s	remaining: 1m 20s
37:	learn: 0.0362491	total: 3m 56s	remaining: 1m 14s
38:	learn: 0.0344818	total: 4m 2s	remaining: 1m 8s
39:	learn: 0.0336604	total: 4m 8s	remaining: 1m 2s
40:	learn: 0.0320593	total: 4m 15s	remaining: 56s
41:	learn: 0.0301747	total: 4m 21s	remaining: 49.8s
42:	learn: 0.0287881	total: 4m 27s	remaining: 43.6s
43:	learn: 0.0278697	total: 4m 33s	remaining: 37.4s
44:	learn: 0.0271902	total: 4m 40s	remaining: 31.1s
45:	learn: 0.0261870	total: 4m 46s	remaining: 24.9s
46:	learn: 0.0251058	total: 4m 52s	remaining: 18.7s
47:	learn: 0.0241355	total: 4m 58s	remaining: 12.4s
48:	learn: 0.0231336	total: 5m 4s	remaining: 6.22s
49:	learn: 0.0221670	total: 5m 11s	remaining: 0us

<catboost.core.CatBoostClassifier at 0x2bc9cfc3040>

```

1 ypreds_cb=model_cb.predict(x_test)
2 ypreds_cb

```

array([1., 1., 0., ..., 1., 1., 0.])

```
1 model_cb.score(x_test,y_test)
```

0.947061889546083

```

1 f2=f1_score(y_test,model_cb.predict(x_test))
2 lgls=log_loss(y_test, model_cb.predict_proba(x_test))
3 acc=accuracy_score(y_test, model_cb.predict(x_test))
4 rac=roc_auc_score(y_test, model_cb.predict(x_test))
5 f2,lgls,rac,acc

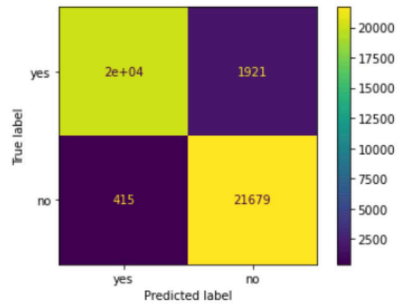
```

(0.9488773143082242,
0.14863016130215848,
0.9470146095898491,
0.947061889546083)


```

1 disp = ConfusionMatrixDisplay(confusion_matrix=confusion_matrix(y_test,model_cb.predict(x_test)),display_labels=['yes','no'])
2 disp.plot()
3 plt.show()

```



```

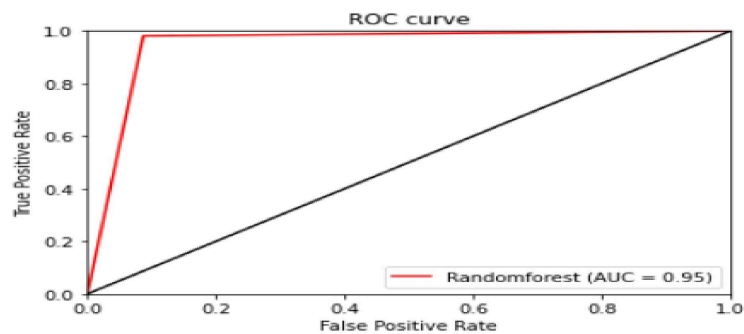
1 joblib.dump(model_cb,"RainPredictModelCB.sav")
[ 'RainPredictModelCB.sav' ]

```

```

1 fpr, tpr, thresholds = roc_curve(y_test, ypreds_cb)
2 roc_auc = auc(fpr, tpr)
3 display = RocCurveDisplay(fpr=fpr, tpr=tpr, roc_auc=roc_auc,
4                             estimator_name='Randomforest')
5 display.plot(color='r')
6 plt.plot(list(np.arange(0,2,0.1)),list(np.arange(0,2,0.1)),color='k')
7 plt.title('ROC curve')
8 plt.xlim([0,1])
9 plt.ylim([0,1])
10 plt.show()
11 plt.show()

```



```

1 #Kfold
2 from sklearn.model_selection import StratifiedKFold
3 folds=StratifiedKFold(n_splits=5)
4 scores=[]
5 for i,j in folds.split(x,y):
6     X_train, X_test, Y_train, Y_test=x.iloc[i],x.iloc[j],y.iloc[i],y.iloc[j]
7     model1 =cb.CatBoostClassifier(iterations= 50,max_depth=16)
8     model1.fit(X_train, Y_train)
9     scores.append(model1.score(X_test,Y_test))
10 print(np.mean(scores))

```

32:	learn: 0.0486616	total: 2m 52s	remaining: 1m 28s
33:	learn: 0.0472389	total: 2m 57s	remaining: 1m 23s
34:	learn: 0.0461924	total: 3m 2s	remaining: 1m 18s
35:	learn: 0.0450078	total: 3m 7s	remaining: 1m 12s
36:	learn: 0.0430414	total: 3m 12s	remaining: 1m 7s
37:	learn: 0.0409735	total: 3m 17s	remaining: 1m 2s
38:	learn: 0.0394043	total: 3m 23s	remaining: 57.3s
39:	learn: 0.0376189	total: 3m 28s	remaining: 52.1s
40:	learn: 0.0361205	total: 3m 33s	remaining: 46.9s
41:	learn: 0.0348553	total: 3m 38s	remaining: 41.7s
42:	learn: 0.0333576	total: 3m 44s	remaining: 36.5s
43:	learn: 0.0316165	total: 3m 49s	remaining: 31.2s
44:	learn: 0.0301865	total: 3m 54s	remaining: 26s
45:	learn: 0.0294056	total: 3m 59s	remaining: 20.8s
46:	learn: 0.0285009	total: 4m 4s	remaining: 15.6s
47:	learn: 0.0274075	total: 4m 9s	remaining: 10.4s
48:	learn: 0.0266174	total: 4m 14s	remaining: 5.2s
49:	learn: 0.0256757	total: 4m 19s	remaining: 0us

0.9250063430976934

MODEL DEPLOYMENT

```

1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 from sklearn.model_selection import train_test_split
5 from sklearn.preprocessing import LabelEncoder
6 #Performance metrics
7 from sklearn.metrics import mean_squared_error, confusion_matrix, roc_auc_score, RocCurveDisplay, auc, roc_curve, ConfusionMatrixDisplay
8 from sklearn.metrics import precision_recall_fscore_support, plot_confusion_matrix, log_loss, accuracy_score, f1_score
9 import joblib

```

```

1 df = pd.read_csv("rainDS.csv")
2 df=df.drop(labels=['Unnamed: 0'],axis=1)
3 df.head()

```

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	...	Pressure9am	Pressure3p
0	12.2	19.6	0.0	2.600000	1.700000	11.0	37.0	6.0	10.0	15.0	...	1026.9	1024.4000
1	15.6	25.6	0.0	6.475381	7.857188	9.0	52.0	1.0	9.0	20.0	...	1023.5	1020.8000
2	7.1	14.2	0.0	3.000000	1.600000	10.0	39.0	14.0	13.0	20.0	...	1017.1	1015.6702
3	7.6	15.8	1.2	1.233005	4.914109	15.0	52.0	3.0	12.0	0.0	...	1018.9	1018.4000
4	17.8	27.6	0.0	5.200000	10.700000	9.0	33.0	10.0	9.0	17.0	...	1016.2	1013.6000

5 rows x 23 columns

```
1 df.shape
```

(44127, 23)

```

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

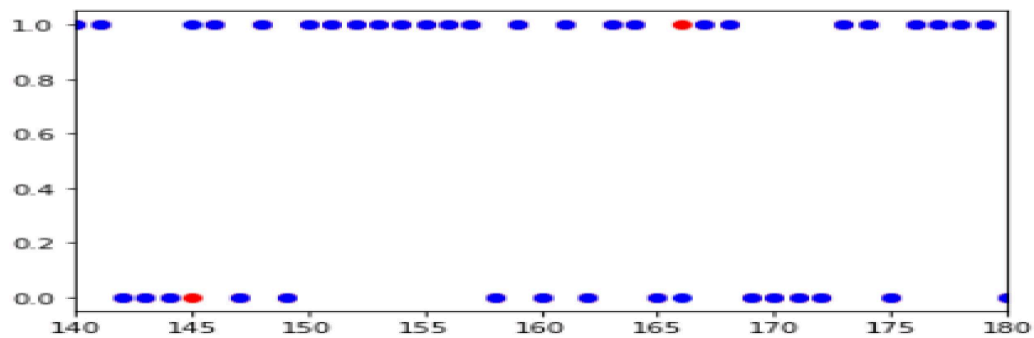
```

RANDOMFOREST

```
1 rf_model = joblib.load("RainPredictModelRF.sav")
2 ypreds=rf_model.predict(x)
3 ypreds
```

```
[1.  0.  1.  ...  1.  1.  1.]
```

```
1 plt.scatter(range(len(y)),y,c='r')
2 plt.scatter(range(len(ypreds)),ypreds,c='b')
3
4
5 plt.xlim(140,180)
6 plt.show()
```



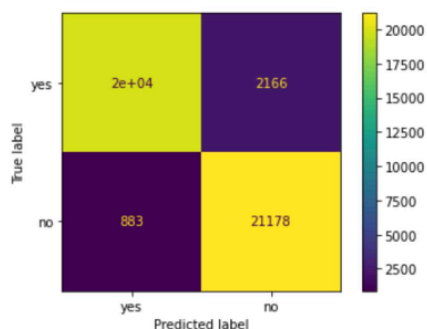
```
1 rf_model.score(x,y)
```

```
0.9309039816892152
```

```
1 f2=f1_score(y,rf_model.predict(x))
2 lgls=log_loss(y, rf_model.predict_proba(x))
3 acc_rf=accuracy_score(y, rf_model.predict(x))
4 rac=roc_auc_score(y, rf_model.predict(x))
5 f2,lgls,rac,acc_rf
```

```
(0.9328488051976656,
0.2181580897425044,
0.9309072752895674,
0.9309039816892152)
```

```
1 disp = ConfusionMatrixDisplay(confusion_matrix=confusion_matrix(y,rf_model.predict(x)),display_labels=['yes','no'])
2 disp.plot()
3 plt.show()
```



XGBOOST

```
1 xgb_model = joblib.load("RainPredictModelXGB.sav")
2 xgb_model.predict(x)
```

```
array([1., 0., 1., ..., 1., 1., 1.])
```

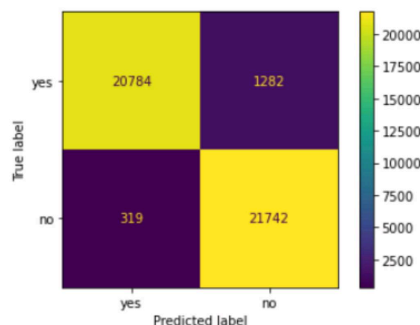
```
1 xgb_model.score(x,y)
```

```
0.9637183583746912
```

```
1 f2=f1_score(y,xgb_model.predict(x))
2 lgls=log_loss(y, xgb_model.predict_proba(x))
3 acc_xgb=accuracy_score(y, xgb_model.predict(x))
4 rac=roc_auc_score(y, xgb_model.predict(x))
5 f2,lgls,rac,acc_xgb
```

```
(0.9644892979926806,
0.11995878970120269,
0.9637208307003282,
0.9637183583746912)
```

```
1 disp = ConfusionMatrixDisplay(confusion_matrix=confusion_matrix(y,xgb_model.predict(x)),display_labels=['yes','no'])
2 disp.plot()
3 plt.show()
```



CATBOOST

```
1 cb_model = joblib.load("RainPredictModelCB.sav")
2 cb_model.predict(x)
```

```
array([1., 0., 1., ..., 1., 1., 1.])
```

```
1 cb_model.score(x,y)
```

```
0.9480590114895642
```

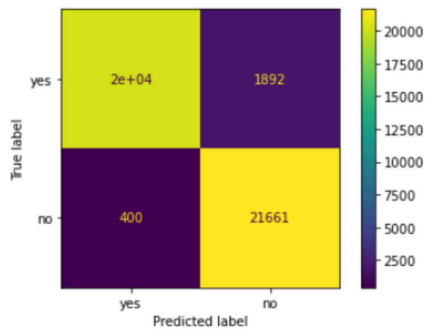
```
1 f2=f1_score(y,cb_model.predict(x))
2 lgls=log_loss(y, cb_model.predict_proba(x))
3 acc_cb=accuracy_score(y, cb_model.predict(x))
4 rac=roc_auc_score(y, cb_model.predict(x))
5 f2,lgls,rac,acc_cb
```

```
(0.9497522690402069,
0.14613725463503874,
0.9480628419803822,
0.9480590114895642)
```

```

1 disp = ConfusionMatrixDisplay(confusion_matrix=confusion_matrix(y,cb_model.predict(x)),display_labels=['yes','no'])
2 disp.plot()
3 plt.show()

```



ENSEMBLE MODEL

```

1 def Ensemble_Model(x,model1,model2,model3):
2     pred=[]
3     l1=model1.predict(x)
4     l2=model2.predict(x)
5     l3=model3.predict(x)
6     for i in range(len(l1)):
7         pred.append(max([l1[i],l2[i],l3[i]], key = [l1[i],l2[i],l3[i]].count))
8     pred=np.array(pred)
9     return pred

```

```

1 f2=f1_score(y,Ensemble_Model(x,rf_model,xgb_model,cb_model))
2 lgls=log_loss(y, Ensemble_Model(x,rf_model,xgb_model,cb_model))
3 acc_en=accuracy_score(y, Ensemble_Model(x,rf_model,xgb_model,cb_model))
4 rac=roc_auc_score(y, Ensemble_Model(x,rf_model,xgb_model,cb_model))
5 f2,lgls,rac,acc_en
6

```

```

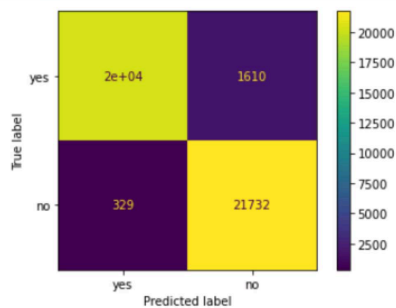
(0.9572935709094113,
1.5177096739319345,
0.9560619376874795,
0.9560586488997665)

```

```

1 disp = ConfusionMatrixDisplay(confusion_matrix=confusion_matrix(y,Ensemble_Model(x,rf_model,xgb_model,cb_model)),display_lab
2 disp.plot()
3 plt.show()

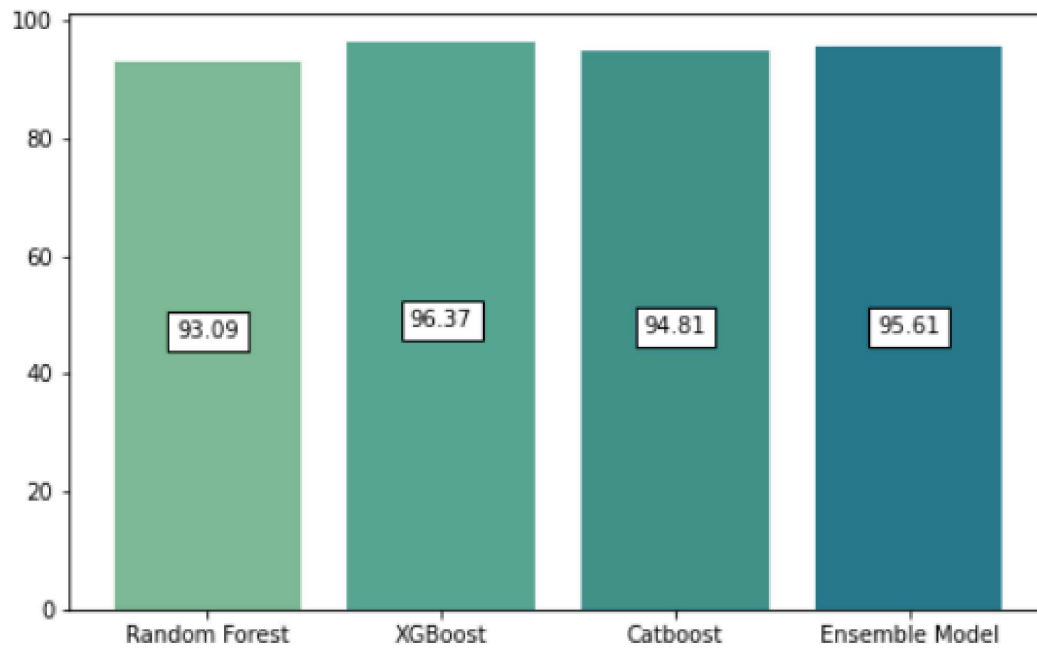
```



```

1 import seaborn as sns
2 l=[acc_rf,acc_xgb,acc_cb,acc_en]
3 l=[i*100 for i in l]
4 n=['Random Forest','XGBoost','Catboost','Ensemble Model']
5 plt.figure(figsize=(8,5))
6 plt.bar(n,l,color=sns.color_palette("crest"))
7 for i in range(4):
8     plt.text(i,l[i]//2,round(l[i],2),ha='center',Bbox = dict(facecolor = 'white', alpha =1))
9
10 plt.show()

```



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,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 SPLITTING 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)),di
splay_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()

```