

**Department of Electrical and Computer Engineering**

**North South University**

**DIRECTED RESEARCH (498R)**

**Flood Prediction Using Various Machine Learning Model**

**Jayed Bin Harez 1912085642**

**Faculty Advisor:**

**Meem Tasfia Zaman**

**Lecturer**

**ECE Department**

**Spring, 2023**

# LETTER OF TRANSMITTAL

May, 2024

To

Dr. Rajesh Palit

Chairman,

Department of Electrical and Computer Engineering

North South University, Dhaka

Subject: **Submission of Directed Research (498R) Report on “Flood Prediction Using Various Machine Learning Model”**

Dear Sir,

With due respect, we would like to submit our **Project Report** on **“Flood Prediction Using Various Machine Learning Model**”as a part of our BSc program. This report delves into the critical domain of flood prediction, employing diverse machine learning methodologies to enhance forecasting accuracy and preparedness. Our research endeavors in this project were instrumental in advancing flood prediction techniques through the application of machine learning models.

We will be highly obliged if you kindly receive this report and provide your valuable judgment. It would be our immense pleasure if you find this report useful and informative to have an apparent perspective on the issue.

Sincerely Yours,

.........................................................

Jayed Bin Harez

ECE Department

North South University, Bangladesh

# APPROVAL

Jayed Bin Harez (1912085642), from Electrical and Computer Engineering Department of North South University, have worked on the Directed Research(498R) Project titled “Flood Prediction Using Various Machine Learning Model” under the supervision of Meem Tasfia Zaman partial fulfillment of the requirement for the degree of Bachelors of Science in Engineering and has been accepted as satisfactory.

**Supervisor’s Signature**

…………………………………….

**Meem Tasfia Zaman**

**Lecturer**

Department of Electrical and Computer Engineering

North South University

Dhaka, Bangladesh.

**Chairman’s Signature**

…………………………………….

**Dr. Rajesh Palit**

**Professor**

Department of Electrical and Computer Engineering

North South University

Dhaka, Bangladesh.

# DECLARATION

This is to declare that this project is our original work. No part of this work has been submitted elsewhere partially or fully for the award of any other degree or diploma. All project related information will remain confidential and shall not be disclosed without the formal consent of the project supervisor. Relevant previous works presented in this report have been properly acknowledged and cited. The plagiarism policy, as stated by the supervisor, has been maintained.

Students’ names & Signatures

**1. Jayed Bin Harez**

-------------------------------------

# ACKNOWLEDGEMENTS

The authors would like to express their heartfelt gratitude towards their project and research supervisor, Meem Tasfia Zaman, Lecturer, Department of Electrical and Computer Engineering, North South University, Bangladesh, for her invaluable support, precise guidance and advice pertaining to the experiments, research and theoretical studies carried out during the current project and also in the preparation of the current report.

Furthermore, the authors would like to thank the Department of Electrical and Computer Engineering, North South University, Bangladesh for facilitating the research.

# ABSTRACT

**Flood Prediction Using Various Machine Learning Model**

This research investigates machine learning models application to improve flood prediction accuracy in Bangladesh, focusing on K-Nearest Neighbours (KNN), Logistic Regression, Support Vector Classifier (SVC), Random Forest, and Decision Tree models. Our results reveal that the Random Forest model had the highest accuracy (95.22%), followed by SVC, Logistic Regression, KNN, and the Decision Tree. These findings show that ML models can capture complicated interactions among environmental variables, allowing for credible flood predictions. Implementing these models can considerably improve early warning systems and disaster management, decreasing the negative effects of flooding in Bangladesh.

TABLE OF CONTENTS

[LETTER OF TRANSMITTAL 2](#_Toc167196842)

[APPROVAL 4](#_Toc167196843)

[DECLARATION 5](#_Toc167196844)

[ACKNOWLEDGEMENTS 6](#_Toc167196845)

[ABSTRACT 7](#_Toc167196846)

[LIST OF FIGURES 10](#_Toc167196847)

[LIST OF TABLES 11](#_Toc167196848)

[Chapter 1 Introduction 12](#_Toc167196849)

[1.1 Background and Motivation 12](#_Toc167196850)

[1.2 Purpose and Goal of the Project 15](#_Toc167196851)

[1.3 Organization of the Report 15](#_Toc167196852)

[Chapter 2 Research Literature Review 16](#_Toc167196853)

[2.1 Existing Research and Limitations 16](#_Toc167196854)

[Chapter 3 Methodology 19](#_Toc167196855)

[3.1 System Design 19](#_Toc167196856)

[3.2 Hardware and/or Software Components 20](#_Toc167196857)

[Chapter 4 Investigation/Experiment, Result, Analysis and Discussion 24](#_Toc167196858)

[Chapter 5 Impacts of the Project 41](#_Toc167196859)

[5.1 Impact of this project on societal, health, safety, legal and cultural issues 41](#_Toc167196860)

[5.2 Impact of this project on environment and sustainability 42](#_Toc167196861)

[Chapter 6 Project Planning and Budget 43](#_Toc167196862)

[Chapter 7 Complex Engineering Problems and Activities 44](#_Toc167196863)

[7.1 Complex Engineering Problems (CEP) 44](#_Toc167196864)

[7.2 Complex Engineering Activities (CEA) 45](#_Toc167196865)

[Chapter 8 Conclusions 46](#_Toc167196866)

[8.1 Summary 46](#_Toc167196867)

[8.2 Limitations 47](#_Toc167196868)

[8.3 Future Improvement 48](#_Toc167196869)

[References 50](#_Toc167196870)

[Appendix 52](#_Toc167196871)

# LIST OF FIGURES

[Figure 1: System diagram of the system 19](#_Toc165904105)

[Figure 2: Sample of dataset 20](#_Toc165904106)

[Figure 3: A concise summary of the employed earthquake magnitude dataset 20](#_Toc165904107)

[Figure 4: Scatterplot of the Bright\_Sunshine and Flood 21](#_Toc165904108)

[Figure 5: Correlations Matrix of flood prediction 21](#_Toc165904109)

[Figure 6: Box plot for flood prediction 22](#_Toc165904110)

[Figure 7: Accuracy against number of Neighbors 29](#_Toc165904111)

[Figure 8: Impact of Hyperparameter Tuning on Model Accuracy 30](#_Toc165904112)

[Figure 9: Accuracy against regularization Strength(C) 31](#_Toc165904113)

[Figure 10: Impact of Hyperparameter Tuning on Logistic Regression Accuracy 32](#_Toc165904113)

[Figure 11: SVC accuracy against regularization Strength(C) 33](#_Toc165904115)

[Figure 12: Impact of Hyperparameter Tuning on Logistic Regression Accuracy 34](#_Toc165904116)

[Figure 13: Confusion matrix for a random forest 35](#_Toc165904117)

[Figure 14: Most important features for Random Forest 36](#_Toc165904118)

[Figure 15: Impact of Hyperparameter Tuning on Random Forest 37](#_Toc165904119)

[Figure 16: Decision Tree accuracy comparison with an without hyperparameter tuning 38](#_Toc165904120)

[Figure 17: Impact of Hyperparameter Tuning on Decision tree 39](#_Toc165904121)

[Figure 18: A sample Gantt chart 43](#_Toc165904122)

# LIST OF TABLES

[Table 1: List of Software/Hardware Tools 23](#_Toc165904039)

[Table 2: Accuracy of the models without hyperparameter 24](#_Toc165904040)

[Table 3: Accuracy of the models with hyperparameter 25](#_Toc165904041)

[Table 4: Accuracy, ROC, Recall of the models without hyperparameter 26](#_Toc165904042)

[Table 5: Accuracy, ROC, Recall of the models with hyperparameter 27](#_Toc165904043)

[Table 6: Accuracy, ROC, Recall of the models with hyperparameter 28](#_Toc165904044)

[Table 7: Comparison of the proposed system with existing works 40](#_Toc165904044)

[Table 8: Demonstrates a sasmple complex engineering problem attribute 44](#_Toc165904045)

[Table 9: A Sample Complex Engineering Problem Activities 45](#_Toc165904046)

# Chapter 1 Introduction

## 1.1 Background and Motivation

Floods are a perennial disaster in Bangladesh, a country particularly vulnerable due to its geographical and climatic conditions. Located in the delta of the Ganges, Brahmaputra, and Meghna rivers, and with a predominantly flat topography, Bangladesh is highly susceptible to the impacts of heavy monsoon rains and cyclonic activity, leading to frequent and often devastating floods. These floods occur almost every year, bringing widespread disaster to the country, affecting millions of people, disrupting livelihoods, damaging infrastructure, and causing significant economic losses.

Historically, Bangladesh has experienced severe flooding events that highlight the catastrophic nature of these disasters. The 1988 flood, one of the worst in recent history, submerged about two-thirds of the country, causing over 2,000 deaths and displacing more than 45 million people. The floodwaters inundated large areas of farmland, destroying crops and leading to a severe food shortage. Infrastructure, including roads, bridges, and railways, was extensively damaged, disrupting communication and transport networks across the nation. The economic impact was enormous, with damages estimated at nearly $2 billion.

Another significant flood event occurred in 1998, which is often cited as the most devastating flood in the country’s history. The flood lasted for over two months, affecting approximately 75% of Bangladesh. Over 30 million people were directly impacted, with around 1,100 reported deaths. The prolonged inundation led to widespread destruction of homes, schools, and hospitals, and a massive loss of livestock. Agricultural losses were severe, as vast tracts of cropland were submerged, destroying rice and other staple crops crucial for the country's food supply. The economic toll was estimated at around $2.8 billion, with long-term effects on the country’s development.

In 2004, another severe flood struck Bangladesh, affecting more than 36 million people. This flood was triggered by heavy monsoon rains and the overflow of rivers, exacerbated by the drainage congestion in urban areas like Dhaka. Approximately 38% of the country was submerged, leading to over 700 deaths. The impact on agriculture was again devastating, with nearly 800,000 hectares of cropland damaged. The floodwaters also caused significant damage to infrastructure, including roads, embankments, and irrigation systems, with total economic losses estimated at over $2 billion.

Recent years have continued to see significant flood events, with the 2017 floods affecting over 8 million people across 32 districts. This event was marked by prolonged monsoon rains and upstream flooding from India, leading to extensive inundation of agricultural land, homes, and infrastructure. The impact on food security was severe, as the floodwaters destroyed vast areas of cropland during the critical growing season. The economic losses were substantial, with damages estimated at over $1 billion.

The 2020 floods were another major event, affecting over 5.4 million people and resulting in extensive damage to homes, crops, and infrastructure. These floods were exacerbated by heavy monsoon rains and the overflow of major rivers, leading to widespread displacement and significant humanitarian needs. The agricultural sector was particularly hard hit, with large areas of rice paddies and other crops destroyed, impacting food security and livelihoods for millions of farmers.

Floods in Bangladesh not only cause immediate physical damage but also have long-term socio-economic impacts. The displacement of millions of people leads to increased pressure on urban areas, where many seek refuge, often in inadequate living conditions. The loss of crops and livestock disrupts food supply chains, leading to food shortages and increased prices. The destruction of infrastructure hampers economic activities and slows down development efforts. Health impacts are also significant, as floodwaters often lead to outbreaks of waterborne diseases such as cholera and dysentery, further straining the country’s healthcare system.

Efforts to mitigate the impact of floods in Bangladesh have included the construction of flood control embankments, river dredging, and the development of early warning systems. However, the effectiveness of these measures is often limited by the scale of the flooding and the country’s financial and technical constraints. Additionally, climate change is expected to exacerbate the frequency and severity of floods in the future, making it imperative for Bangladesh to enhance its resilience and adaptive capacity.

Floods are an annual disaster in Bangladesh, driven by its unique geographical and climatic conditions. Historical data underscores the devastating impact of these floods, with millions of people affected, significant loss of life, and extensive damage to agriculture, infrastructure, and the economy. While efforts are being made to mitigate the impact, the challenges posed by climate change necessitate a continued focus on improving flood management and resilience strategies to safeguard the lives and livelihoods of Bangladesh’s population.

The primary motivation for using these ML models in flood prediction for Bangladesh is to enhance the accuracy and reliability of forecasts. Accurate flood predictions are crucial for timely evacuation, efficient resource allocation, and minimizing the loss of life and property. By leveraging historical data and real-time environmental inputs, ML models can identify complex patterns and interactions that traditional models may miss. This leads to better-informed decision-making and more effective disaster management strategies.

The adaptability and scalability of ML models make them well-suited for integration into existing flood monitoring and warning systems. They can continuously learn and improve from new data, ensuring that predictions remain relevant and up-to-date. In a country like Bangladesh, where the stakes are high, the implementation of advanced ML techniques for flood prediction can significantly reduce the adverse impacts of floods, contributing to greater resilience and sustainability.

The deployment of ML models such as DT, SVC, RF, KNN, and Logistic Regression for flood prediction in Bangladesh is motivated by the need for more precise, timely, and actionable flood forecasts. These models offer the potential to revolutionize flood management practices, ultimately protecting lives, livelihoods, and infrastructure from the ravages of annual flood events.

## 1.2 Purpose and Goal of the Project

The primary objective of this project is to develop and evaluate machine learning (ML) models for flood prediction in Bangladesh, aiming to enhance the accuracy and timeliness of forecasting to mitigate the adverse impacts of floods on vulnerable communities. The project seeks to leverage historical meteorological, hydrological, and satellite imagery data to train ML algorithms, including k-Nearest Neighbors (KNN), Decision Trees (DT), Support Vector Machines (SVM), and Random Forests (RF), to predict flood events with high precision. The novelty of this project lies in the comprehensive evaluation of various ML models specifically tailored to the context of Bangladesh, considering the country's unique environmental factors and flood dynamics. By deploying these models, the project aims to contribute to improved disaster preparedness, early warning systems, and effective mitigation strategies to minimize the socio-economic impacts of floods in Bangladesh.

## 1.3 Organization of the Report

The first chapter discusses the project's origins, motivation, aim, and goal. The second chapter contains the project-related literature reviews. The block diagram, flow chat description, and component implementation are all shown in Chapter 3. In Chapter 4 the output of the project is provided. Chapter 5 provides Impact of this project on societal, health, safety, legal and cultural issues. Chapter 6 gives the visual idea of project planning of the project. Chapter 7 represents the complex engineering problems and activities. And Finally, Chapter 8 provides the summary and limitation of the project.

# Chapter 2 Research Literature Review

## 2.1 Existing Research and Limitations

Sayeed and his groupmate [11] predicted flood using machine learning models. They a dataset which was containing the amount of rainfall and yearly flood occurrence of 34 stations in Bangladesh from the year 1980 to 2020. They have used four different machine learning model. Among them they got the height accuracy using logistic regression which is 0.8676 and SVC and DT gave the lowest accuracy. They were unable to collect features like river water level, temperature, humidity which is relevant to flood.

Canillo and his crew [10] established flood risk visualization and prediction information system. Information systems for flood management were useful in keeping an eye on and forecasting potential flood zones in Manila, Philippines. Their dataset is based in Manila City. They have used knn and logistic regression for flood visualization and Prediction Information System. The knn got the height accuracy with .994 and logistic regression got .833. In dataset they cannot build early warning system.

Rajab and his groupmate [2] forecasting flood by using machine learning model. Their dataset includes information on Bangladesh’s monthly and yearly rainfall (1949 to 2013) index as well as information on the number of times a year that floods occur close to 35 stations. They have used 9 different ml model. Among them random forest and polynomial regression got the height value with 0.76. They couldn’t add parameters like humidity, wind gusts, pressure in the air, atmospheric pressure features in their dataset. This feature had a great impact for occurring flood.

Bangera and his team [6] predicted flood and heat wave using weighted moving average, anomaly detection and knearest neighbours for the city of mangalore. They used a historical dataset of mangalore which includes features like temperature, humidity and pressure etc. They have used only one machine learning model named knn. They got the accuracy of 90% using the model. They were in lack of collecting some features related to flood.

Setya and his groupmate[5] predicted comparative analysis of rainfall value using linear and k-nearest neighbor algorithms.They have collected their dataset from BMKG Semarang. They dataset have features like rainfall, temperature, humidity, wind speed and duration of sunshine from 2021 to 2023.They used multiple linear regression and knn algorithm. By using knn they got lowest error which is 1.22 and multi linear regression gave 2.21. Knn got the the better accuracy then multi linear regression. It is better to use a smarter computational technology approach to get the best prediction results or model for the prediction of rainfall.

Noushin and her crew [1] predicted flood using machine learning algorithms. They chose the weather data of Bangladesh for 65 years taken from Kaggle dataset. In their dataset the total number of instances was 20543. They used the k-Nearest Neighbors Algorithm and gave them a maximum accuracy of 94.91. They couldn’t apply advanced machine learning algorithms like random forest ,support vector classifier , xg boost etc. This advanced machine learning model can enhance their accuracy.

Anil and his crew [3] predicted flood using machine learning models. They have used xgboost , Logistic Regression(LR), Decision Tree(DT), and KNN algorithms. By training the model they got accuracy of xgboost(.993) , Logistic Regression(.987), Decision Tree(.993), and KNN algorithms(.987).They got height accuracy by using xgboost which is 99.3%.

Kruti and her team [4] developed a flood prediction system which used various machine learning algorithms. They collected the dataset from Bihar and Orissa from 1992-2002.They have used Decision Tree Algorithm, Gradient Boost Algorithm, Random Forest Algorithm. Among this model they got height accuracy form Decision Tree Algorithm which was 94.4%.

Deowan and his groupmate [9] developed an IoT based Smart Early Flood Monitoring System. To develop the system, they have used DHT11 sensor, Ultrasonic sensor, and the Rainfall sensor and they have used polynomial model for prediction flood. They have used other real time data for prediction flood. They didn’t apply advanced machine learning model, which could help to get better accuracy.

Raut and his team [8] predicted rainfall using random forest regressor model. The dataset for used for this prediction is taken from Govt. website.They have used four different types of model like random forest regressor, liner regression, decision tree, svr. They got the height accuracy using random forest regressor .89 and lowest accuracy using SVR which is .21. They couldn’t get well accuracy as they expected.

Mahajan and his groupmate [7] worked on prediction of rainfall using machine learning .They have used Australian dataset for rainfall prediction . They used this dataset because it considers numerous parameters such as min temperature, max temperature, rainfall, wind speed, wind direction, humidity, pressure. They used four various kind of machine learning model like naive bayes, random forest ,knn, svm. They got the height accuracy by using random forest by 85% and svm got the lowest accuracy by 51%. The technique to be justified considering more varied datasets.

Riaz and his teammate[13] worked on Advancing Flood Disaster Mitigation in Indonesia Using Machine Learning Methods. They used different type of machine learning model. Among them svm got the height accuracy with 100%. They also used knn which got the accuracy of 93.3%. They used short dataset in their research.

Felix and his crew[12] predicted flood using gradient boost machine learning approach. The dataset is collected from remote sensing satellites and ground applications which can take in the census from various parameters like rainfall measurement, water-level of the nearby water-body (like a river or lake). They used gradient boost for prediction the flood. They didn’t apply other advanced machine learning techniques.

# Chapter 3 Methodology

## 3.1 System Design

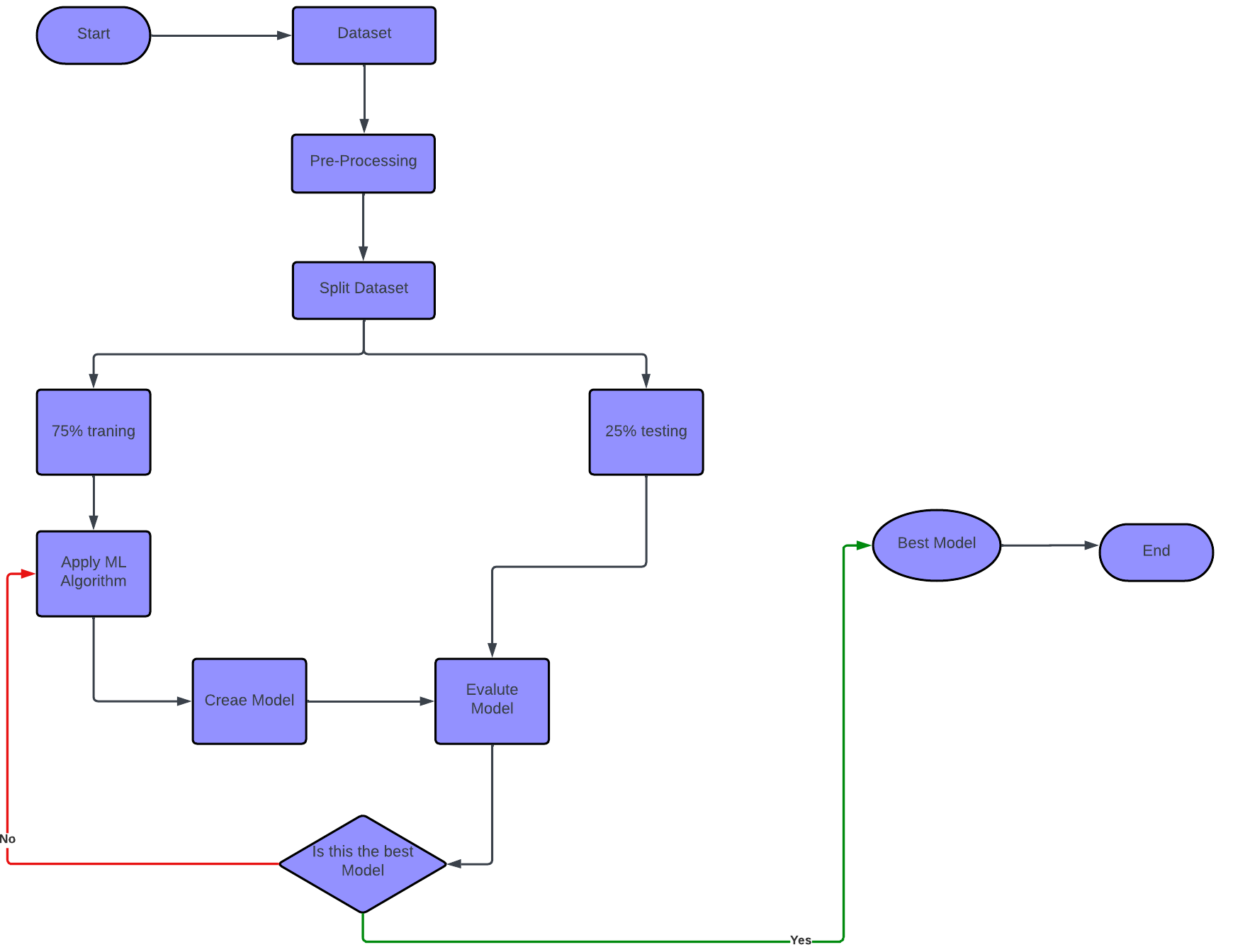


Figure 1: System diagram of the system

Working sequences of the proposed flood prediction using various machine learning model have been depicted in Figure 1. This work is initiated by collecting data. Then we have to apply the pre-processing techniques for the duplicate and missing or null values, categorical features, etc. After finishing pre-processing techniques, the dataset has been split into 75% for training and 25% for testing facilitating the stratifying preference. Then the ML algorithms have been applied to create and evaluate the model for training data with enabling the hyperparameter optimization techniques. Next, the applied models are evaluated using test data. This process will terminate after getting the best model with the maximum score.

## 3.2 Hardware and/or Software Components

**Dataset**

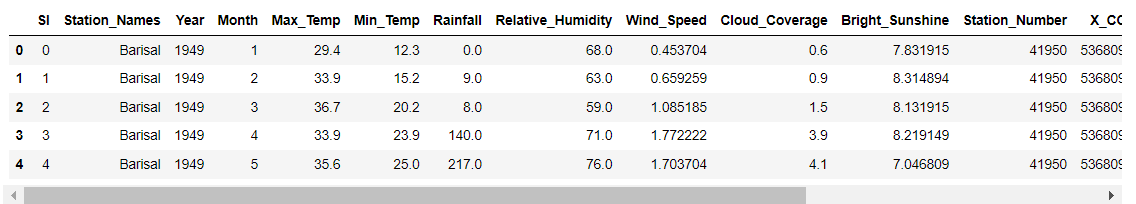


Figure 2: Sample of dataset

We have collected dataset from Kaggle and other trusted government sources. The dataset contains 20543 amounts of data from 1949 to 2013. In this dataset we have some important features like Rainfall, Maximum temperature, Minimum temperature, Wind speed, Cloud coverage, Relative Humidity, Bright Sunshine, Flood etc. In this dataset we can see 16051 numbers of missing value in Flood features.

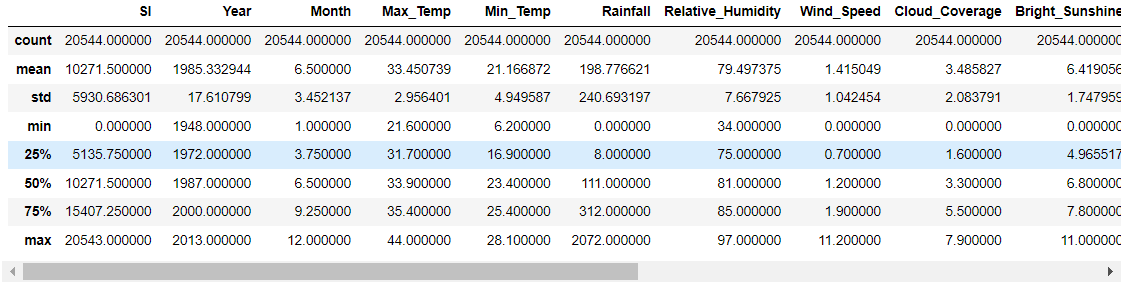


Figure 3: A concise summary of the employed earthquake magnitude dataset

Figure 3 provides the number of non-empty values, the mean (average) value, the standard deviation, the 25thpercentile, the 50th percentile, the 75th percentile, and the maximum value of the numerical features of the employed flood prediction dataset.

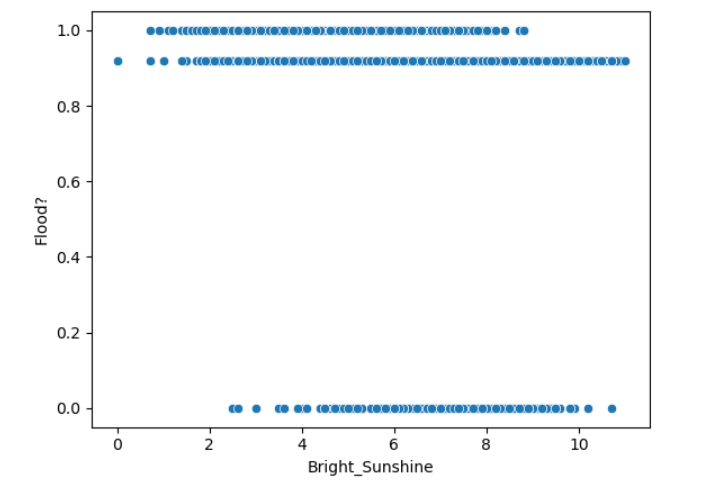


Figure 4: Scatterplot of the Bright\_Sunshine and Flood

Figure 4 shows the flood occur during the Bright\_Sunshine.

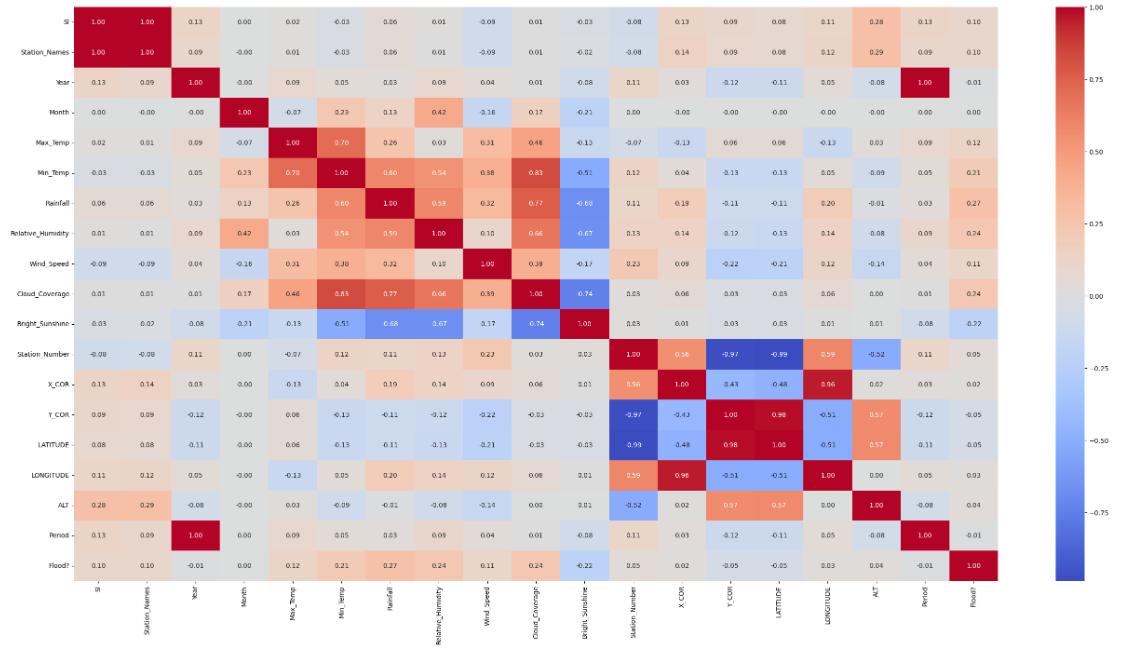


Figure 5: Correlations Matrix of flood prediction

Figure 5 represents the relationship between various features and the occurrence or severity of floods. We can see Rainfall, Maximum temperature, Minimum temperature, Wind speed, Cloud coverage, Relative Humidity, Bright Sunshine has the height value against flood. So this features are the most important features for flood.

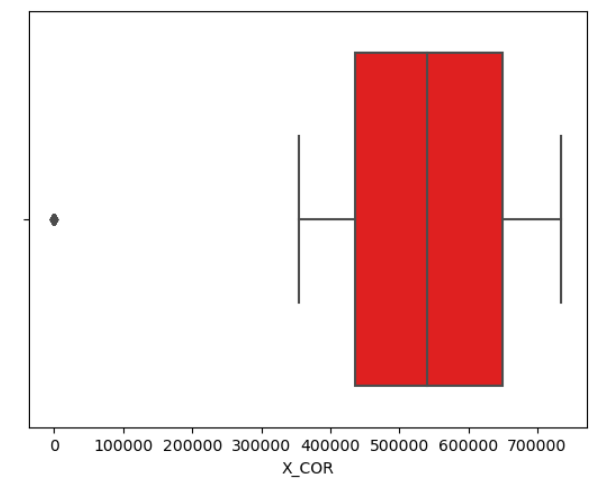


Figure 6: Box plot for flood prediction

Figure 6 show the outlier for flood. We can see total number of outliers is 1. That means it creates noise in the dataset. For outlier the ml model could be biased.

**Dataset Preprocessing**

We have developed Flood Prediction Using Various Machine Learning Model. For our project we have used a dataset that is collected from Kaggle and other trusted sources. In our dataset we have 20544 rows and columns. In our dataset we find some miss value in Floods. In Flood column we can see total 16051 values are missing. To handle missing value we have used mean imputation technique.

The mean imputation formula for replacing missing values with the mean of the available data for a particular feature can be expressed as:

Where:

is the imputed value for the missing data point.

​ are the observed values of the feature.

n is the total number of observations (including missing values).

In words, to impute a missing value, you take the sum of all observed values for that feature and divide it by the total number of non-missing values. This gives you the mean value, which is then used to replace the missing value.

However, in practice, you might only consider the non-missing values when calculating the mean, so the formula could be simplified to:

m is the number of observed (non-missing) values for the feature.

Table 1: List of Software/Hardware Tools

|  |  |  |  |
| --- | --- | --- | --- |
| **Tool** | **Functions** | **Other similar Tools (if any)** | **Why selected this tool** |
| Jupyter Notebook | It helps to manipulate data, visualize results, and iterate on code quickly. | [Google Colab](https://colab.google/) | It is easier to use and more comfortable |

# Chapter 4 Investigation/Experiment, Result, Analysis and Discussion

**Without Hyperparameter**

|  |  |
| --- | --- |
| Model | Accuracy Score |
| KNN | 94.139408 |
| Logistic Regression | 94.450935 |
| SVC | 94.373053 |
| Random Forest | 95.035047 |
| Decision Tree | 92.289720 |

Table 2: Accuracy of the models without hyperparameter

The accuracy ratings of various machine learning models are shown in the table without any hyperparameter adjustments. A distinct model, such as K-Nearest Neighbours (KNN), Logistic Regression, Support Vector Classifier (SVC), Random Forest, and Decision Tree, is represented by each row. The accuracy score column shows how well each model performed by expressing the proportion of accurate predictions the model generated using the dataset. With the best accuracy score of 95.04%, Random Forest was followed closely by SVC and Logistic Regression, with values of 94.37% and 94.45%, respectively. With an accuracy score of 94.14%, KNN outperformed Decision Tree, which had a little lower accuracy score of 92.29%. All things considered; the table shows how these models function on the dataset in the absence of hyperparameter optimization.

**With Hyperparameter**

|  |  |
| --- | --- |
| Model | Accuracy Score |
| KNN | 94.470405 |
| Logistic Regression | 94.489875 |
| SVC | 94.548287 |
| Random Forest | 95.229751 |
| Decision Tree | 94.217290 |

Table 3: Accuracy of the models with hyperparameter

Table 3 presents the accuracy scores of different machine learning models following hyperparameter tuning, which is the process of optimizing each model's parameters for optimal performance. K-Nearest Neighbours (KNN), Decision Trees, Random Forests, Support Vector Classifiers (SVC), and Logistic Regression are among the models evaluated. The accuracy score indicates the percentage of accurate predictions each model made using the provided dataset. Interestingly, Random Forest showed a marginal boost in performance above its performance without hyperparameter modification, emerging as the best-performing model with an accuracy score of 95.23%. Additionally, after hyperparameter optimisation, both SVC and logistic regression showed improved accuracy ratings of 94.55% and 94.49%, respectively. KNN demonstrated consistent performance both with and without hyperparameter modification, maintaining a high accuracy score of 94.47%. Decision Tree's accuracy increased to 94.22% with hyperparameter adjustment, yet it was still functioning wonderfully. Overall, the table shows how well hyperparameter tuning can be used to improve model performance, with each model showing different levels of accuracy improvement over their untuned counterparts.

Top of Form

**Without Hyperparameter**

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Accuracy Score | Recall Score | ROC score |
| KNN | 94.139408 | 84.549763 | 90.584119 |
| Logistic Regression | 94.450935 | 84.644550 | 90.815291 |
| SVC | 94.373053 | 83.507109 | 90.344586 |
| Random Forest | 95.035047 | 84.644550 | 91.182848 |
| Decision Tree | 92.289720 | 82.085308 | 88.506511 |

Table 4: Accuracy, ROC, Recall of the models without hyperparameter

Table 4 presents the performance metrics of various machine learning models without hyperparameter tuning, offering insights into their classification capabilities. Five models are evaluated: K-Nearest Neighbors (KNN), Logistic Regression, Support Vector Classifier (SVC), Random Forest, and Decision Tree. Each model's effectiveness is assessed through three key metrics: Accuracy Score, Recall Score, and ROC Score. Accuracy Score reflects the proportion of correctly classified instances, where Random Forest achieves the highest score at 95.04%, followed closely by Logistic Regression and SVC with scores of 94.45% and 94.37% respectively. KNN and Decision Tree also perform well with scores of 94.14% and 92.29%. Recall Score measures the ability of the model to correctly identify true positive instances, with Random Forest and Logistic Regression leading at 84.64%. SVC and KNN follow closely at 83.51% and 84.55% respectively, while Decision Tree exhibits a slightly lower recall score of 82.09%. ROC Score evaluates the model's ability to distinguish between classes, with Random Forest achieving the highest score of 91.18%, indicating superior classification performance. Logistic Regression and SVC also demonstrate strong ROC scores of 90.82% and 90.34% respectively, while KNN and Decision Tree show slightly lower scores at 90.58% and 88.51% respectively. Overall, the table provides a detailed comparison of the models' performance across multiple metrics, aiding in the selection of the most suitable model for classification tasks.

**With Hyperparameter**

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Accuracy Score | Recall Score | ROC score |
| KNN | 94.470405 | 83.412322 | 90.37070 |
| Logistic Regression | 94.489875 | 84.549763 | 90.804654 |
| SVC | 94.548287 | 82.748815 | 90.173721 |
| Random Forest | 95.229751 | 95.229751 | 91.65678 |
| Decision Tree | 94.217290 | 83.127962 | 90.106005 |

Table 5: Accuracy, ROC, Recall of the models with hyperparameter

Table 5 presents the performance characteristics of different machine learning models following hyperparameter tuning, offering an understanding of their categorization capabilities. K-Nearest Neighbours (KNN), Decision Tree, Random Forest, Support Vector Classifier (SVC), and Logistic Regression are the five models that are assessed. Three primary measures are used to evaluate each model's performance: Accuracy Score, Recall Score, and ROC Score. The accuracy score, which measures the percentage of correctly identified cases, is greatest for Random Forest (95.23%), followed closely by SVC (94.55%) and Logistic Regression (94.49%). With respective accuracy ratings of 94.47% and 94.22%, KNN and Decision Tree also demonstrate commendable performance. The model's recall score, which is 95.23% for Random Forest and 84.55% for Logistic Regression, indicates how well it can detect genuine positive instances. KNN has a marginally lower recall score of 83.41%, but SVC and Decision Tree both perform well with recall ratings of 82.75% and 83.13%, respectively. The Random Forest model receives the greatest ROC Score of 91.66%, which measures the model's ability to differentiate across classes. Although KNN and Decision Tree show somewhat lower results of 90.37% and 90.11%, respectively, Logistic Regression and SVC also show good ROC scores of 90.80% and 90.17%, respectively. To help choose the best model for classification tasks, the table presents an extensive analysis of the models' performance after hyperparameter adjustment.

**With Hyperparameter**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Accuracy Score | Recall Score | ROC Score | Hyperparameter |
| KNN | 94.470405 | 83.412322 | 90.37070 | 'n\_neighbors': 20 |
| Logistic Regression | 94.489875 | 84.549763 | 90.804654 | 'C': 0.001321941148466028 |
| SVC | 94.548287 | 82.748815 | 90.173721 | 'C': 100 |
| Random Forest | 95.229751 | 95.229751 | 91.65678 | 'max\_depth': 20,  'min\_samples\_leaf': 1,  'min\_samples\_split': 5,  'n\_estimators': 150 |
| Decision Tree | 94.217290 | 83.127962 | 90.106005 | 'criterion': 'gini',  'max\_depth': 10,  'min\_samples\_leaf': 4,  'min\_samples\_split': 10 |

Table 6: Accuracy, ROC, Recall of the models with hyperparameter

Table 6 presents the performance metrics of various machine learning models after hyperparameter tuning, shedding light on their classification prowess. Five models are evaluated: K-Nearest Neighbors (KNN), Logistic Regression, Support Vector Classifier (SVC), Random Forest, and Decision Tree. Alongside the accuracy, recall, and ROC scores, the specific hyperparameters tuned for each model are also provided. KNN achieves an accuracy score of 94.47%, with a recall score of 83.41% and an ROC score of 90.37%, utilizing a 'n\_neighbors' value of 20. Logistic Regression exhibits an accuracy score of 94.49%, a recall score of 84.55%, and an ROC score of 90.80%, with a 'C' value of 0.001321941148466028. SVC attains an accuracy score of 94.55%, a recall score of 82.75%, and an ROC score of 90.17%, employing a 'C' value of 100. Random Forest emerges as the top performer with an accuracy score of 95.23%, a recall score of 95.23%, and an ROC score of 91.66%, utilizing a 'max\_depth' of 20, 'min\_samples\_leaf' of 1, 'min\_samples\_split' of 5, and 'n\_estimators' of 150. Lastly, Decision Tree achieves an accuracy score of 94.22%, a recall score of 83.13%, and an ROC score of 90.11%, with hyperparameters including 'criterion' as 'gini', 'max\_depth' of 10, 'min\_samples\_leaf' of 4, and 'min\_samples\_split' of 10. This comprehensive table aids in selecting the most suitable model for classification tasks, considering both performance metrics and tuned hyperparameters.

**K-Nearest Neighbour**

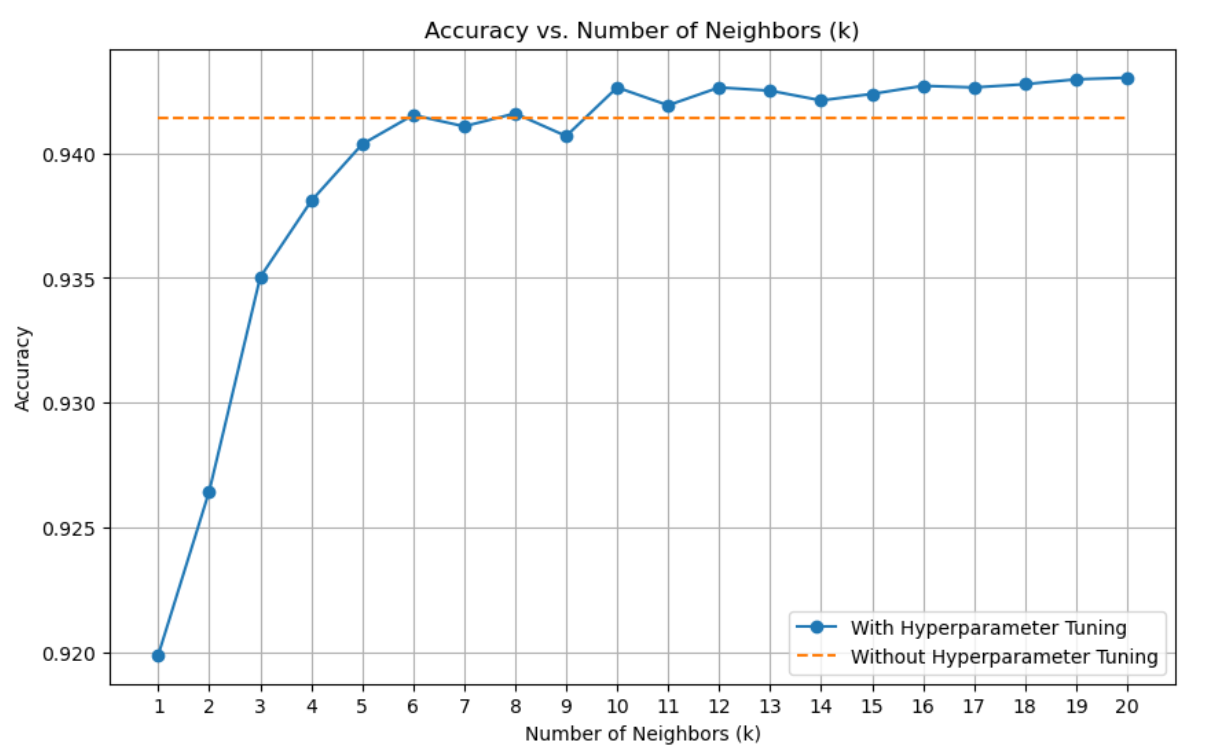


Figure 7: Accuracy against number of Neighbors

The figure 7 compares the performance with and without hyperparameter adjustment, showing the relationship between classification accuracy and the number of neighbours (k) utilised in a k-Nearest Neighbours method. The number of neighbours (k), represented by the x-axis, ranges from 1 to 20, while the classification accuracy, represented by the y-axis, ranges from approximately 0.920 to 0.940.The accuracy attained with hyperparameter adjustment is shown by the blue line with round markers. When hyperparameter tuning is used, this line shows an initial growing trend and peaks at approximately k=5 or 6. This suggests that the ideal number of neighbours falls within this range. On the other hand, the accuracy steadily declines as the number of neighbours increases, indicating that an excessively large value of k may have an adverse effect on the performance.   
On the other hand, the accuracy achieved in the absence of hyperparameter adjustment is indicated by the orange dashed line. When hyperparameter tuning is not carried out, this line stays comparatively flat and steady across a range of values of k, showing that the accuracy does not change greatly with the number of neighbours. The uniformity of accuracy emphasises how crucial it is to adjust the hyperparameters, particularly the k value, in order to get the best results possible from the k-Nearest Neighbours method.

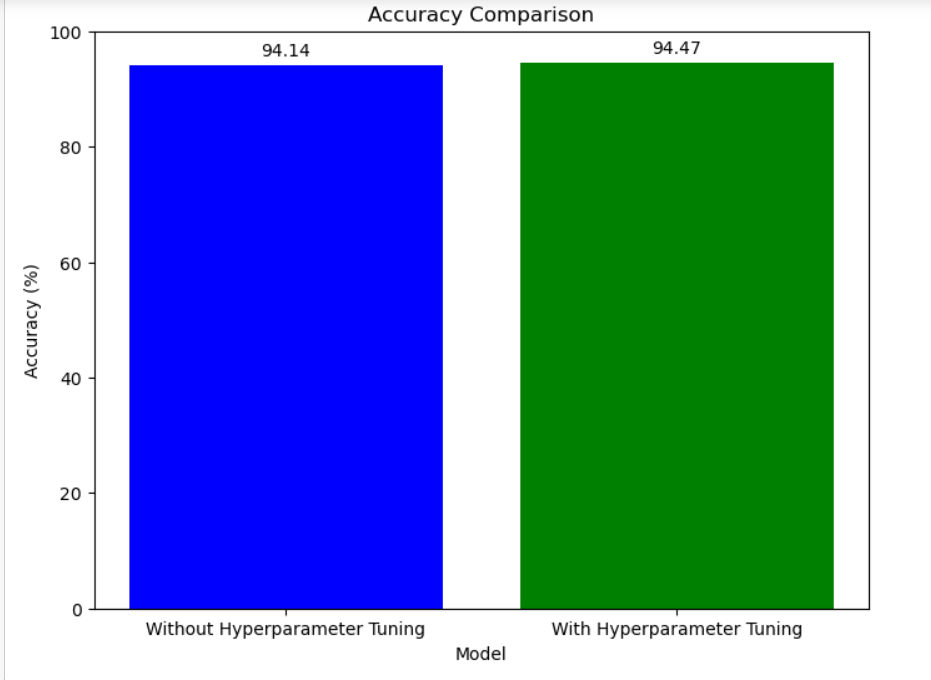


Figure 8: Impact of Hyperparameter Tuning on Model Accuracy

The Figure 8 contrasts a model's accuracy with and without hyperparameter adjustment. With hyperparameter adjustment, the accuracy is 94.47%, as shown by the green bar, whereas the blue bar shows 94.14% accuracy without that modification. This shows that by determining the ideal hyperparameter configuration, hyperparameter tuning can enhance the model's performance and increase accuracy for the given task.

**Logistic Regression**

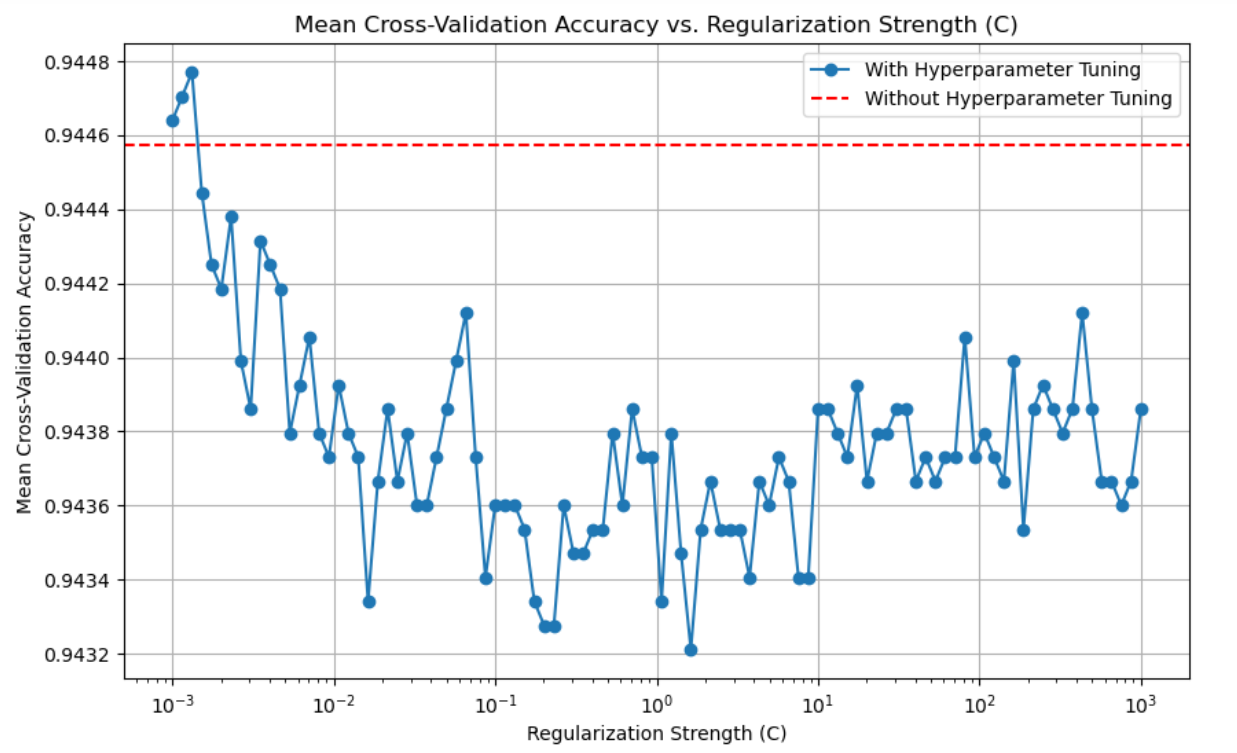


Figure 9: Accuracy against regularization Strength(C)

The Figure 9 illustrates the correlation between the regularisation strength (C) and the mean cross-validation accuracy (M) for a given model, both with and without hyperparameter tuning. The regularisation strength (C) is represented on the x-axis, which is a logarithmic scale spanning from 10^-3 to 10^3. The mean cross-validation accuracy, which ranges between 0.9432 and 0.9448 on the y-axis, is displayed. The model's performance during hyperparameter tuning is depicted by the blue line featuring circular indicators, which display a dynamic pattern. The accuracy initially improves as the strength of regularisation increases from lower values, reaching its maximum at approximately C=10^-1. Subsequently, as C continues to increase, the accuracy progressively declines.   
In contrast, the performance of the model in the absence of hyperparameter optimisation is denoted by the orange dashed line. This line remains relatively flat and constant at an accuracy of approximately 0.9446, indicating that when hyperparameters are not tuned, variations in the regularisation strength have no effect on the accuracy. Figure 8 underscores the significance of tuning hyperparameters, particularly the regularisation strength parameter, in order to maximise the efficacy of the model. The variability observed in the blue line indicates the presence of an ideal regularisation strength value that optimises the accuracy of cross-validation in the presence of hyperparameter tailoring.

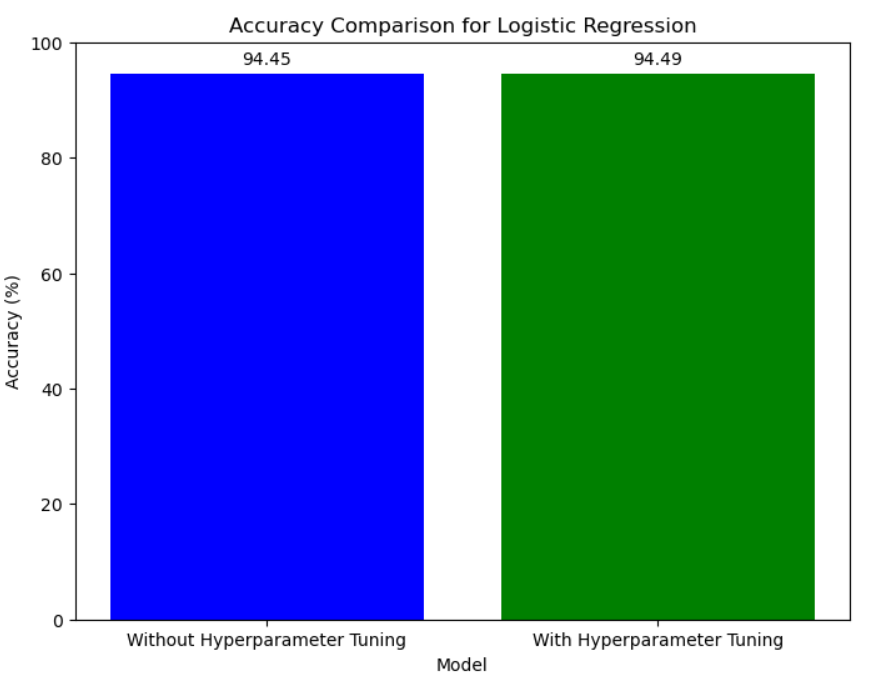


Figure 10: Impact of Hyperparameter Tuning on Logistic Regression Accuracy

The figure 10 shows a comparison of logistic regression models' accuracy with and without hyperparameter adjustment. According to the results, the model that underwent hyperparameter tuning had an accuracy of 94.49%, which was greater than the accuracy of 94.45% without tuning. This suggests that the performance of logistic regression models can be enhanced by hyperparameter tuning.**Support Vector Classifier (SVC)**

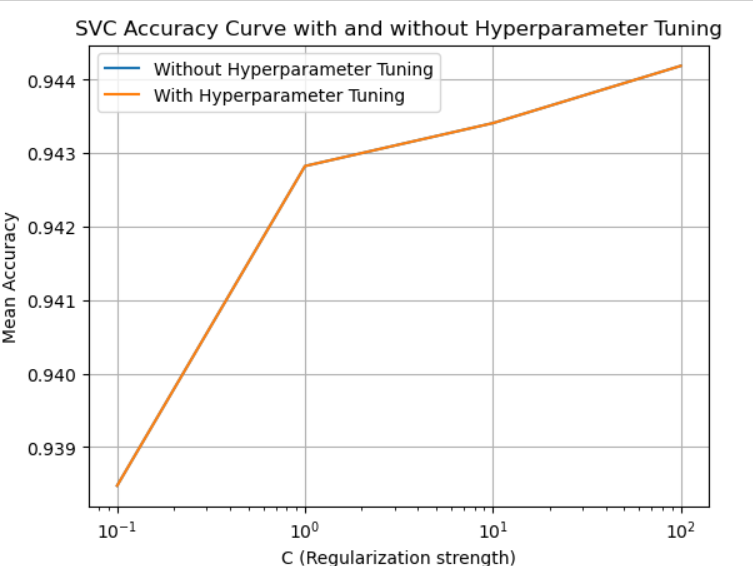


Figure 11: SVC accuracy against regularization Strength(C)

The figure 11 depicts an SVC (Support Vector Classifier) accuracy curve, which compares performance with and without hyperparameter modification. The x-axis shows the regularisation strength (C) on a logarithmic scale, from 10^-1 to 10^2. The y-axis represents the mean accuracy values.

When regularisation strength is modest (10^-1), both lines achieve a similar accuracy level of roughly 0.939. As the regularisation strength increases, the accuracy curve without hyperparameter adjustment flattens, showing little performance improvement. The curve with hyperparameter tuning rises progressively, reaching an accuracy level of around 0.944 at the maximum regularisation strength (10^2).   
  
This pattern indicates that proper hyperparameter tuning, particularly altering the regularisation strength, can greatly improve the SVC model's predictive ability. Through refining, the model can better generalise and achieve higher accuracy on previously unseen data.

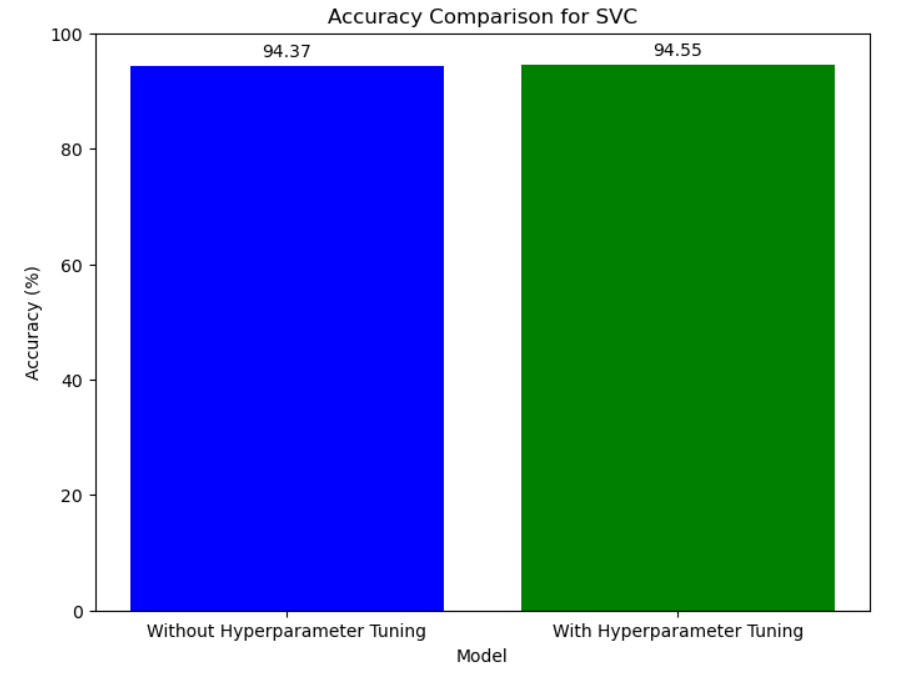


Figure 12: Impact of Hyperparameter Tuning on Logistic Regression Accuracy

The figure 12 the accuracy of a Support Vector Classifier (SVC) model with and without hyperparameter modification. It demonstrates that the model with hyperparameter tuning gets a greater accuracy of 94.55%, whereas the model without tuning has a lower accuracy of 94.37%. Hyperparameter adjustment improves the performance of the SVC model.

**Random forest**

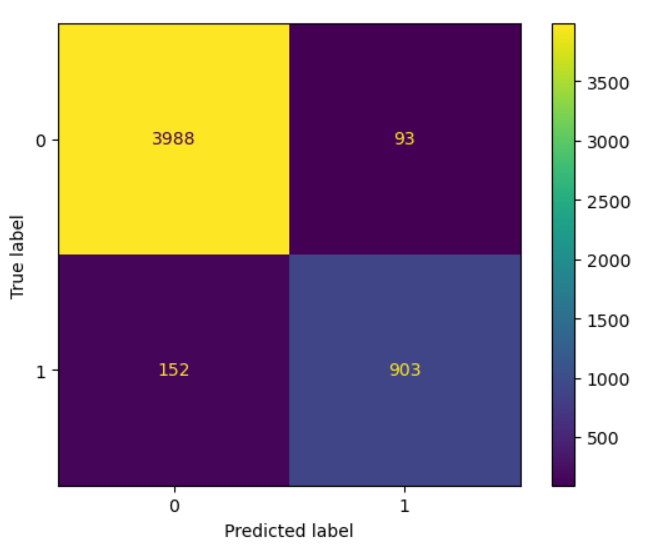


Figure 13: Confusion matrix for a random forest

This figure 13 seems to be a confusion matrix for a random forest model binary classification problem. The genuine labels are depicted on the y-axis, and the predicted labels are displayed on the x-axis (0 and 1). The number of cases that were accurately predicted as class 0 (true negatives) is shown in the top-left cell (3988). The number of instances accurately predicted as class 1 (true positives) is displayed in the bottom-right cell (903) of the figure. The number of cases that were mistakenly forecasted as class 0 when they actually belonged to class 1 (false negatives) is shown in bottom-left cell (152). On the other hand, the number of cases that were mistakenly forecasted as class 1 when they actually belonged to class 0 (false positives) is displayed in the top-right cell (93).   
The random forest model appears to function fairly effectively, with a relatively high number of true positives and true negatives and a lesser number of false positives and false negatives, based on the values in the confusion matrix. To completely assess the model's efficacy, additional examination of performance indicators such as precision, recall, and F1-score would be required.

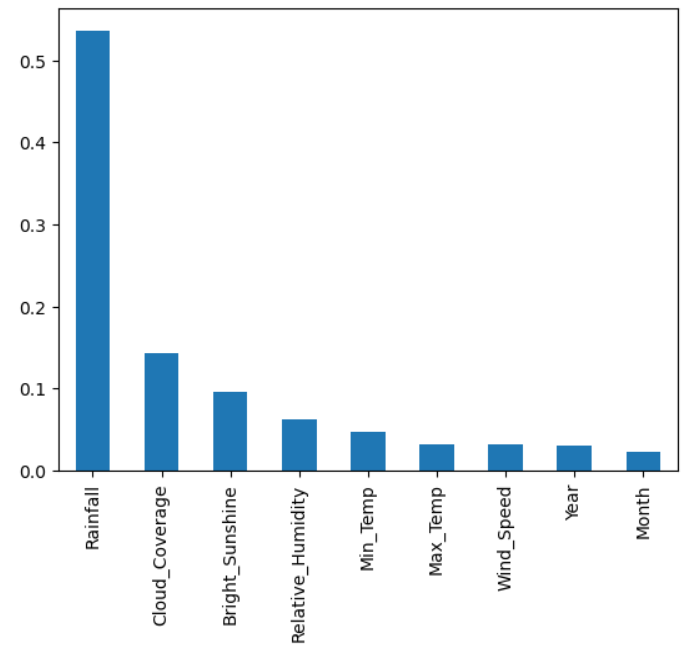


Figure 14: Most important features for Random Forest

The figure 14 presented here the feature significance scores derived from a random forest model. The vertical axis represents a numerical score that spans from 0 to 0.5, and the horizontal axis enumerates the various features employed in the model.

The feature labelled as "Rainfall" has the greatest relevance score, around 0.45, suggesting its significant influence on the predictions made by the random forest model.

Additional noteworthy attributes that exhibit reasonably high significance ratings are "Cloud\_cover" (about 0.15) and "Bright\_Sunshine" (around 0.1).

In comparison to the top features, the other variables, namely "Relative\_Humidity", "Min\_Temp", "Max\_Temp", "Wind\_Speed", "Year", and "Month", exhibit lower relevance ratings, indicating a diminished impact on the model's predictions.

Feature importance scores in random forest models are computed using methods such as mean decrease in impurity or permutation importance. Greater scores imply that a specific characteristic has a more substantial impact on enhancing the model's ability to make accurate predictions.   
This information is valuable for the purpose of feature selection, as it allows for the identification of the predictors that are influencing the model's decisions. Additionally, it has the ability to provide insights into the underlying issue domain.

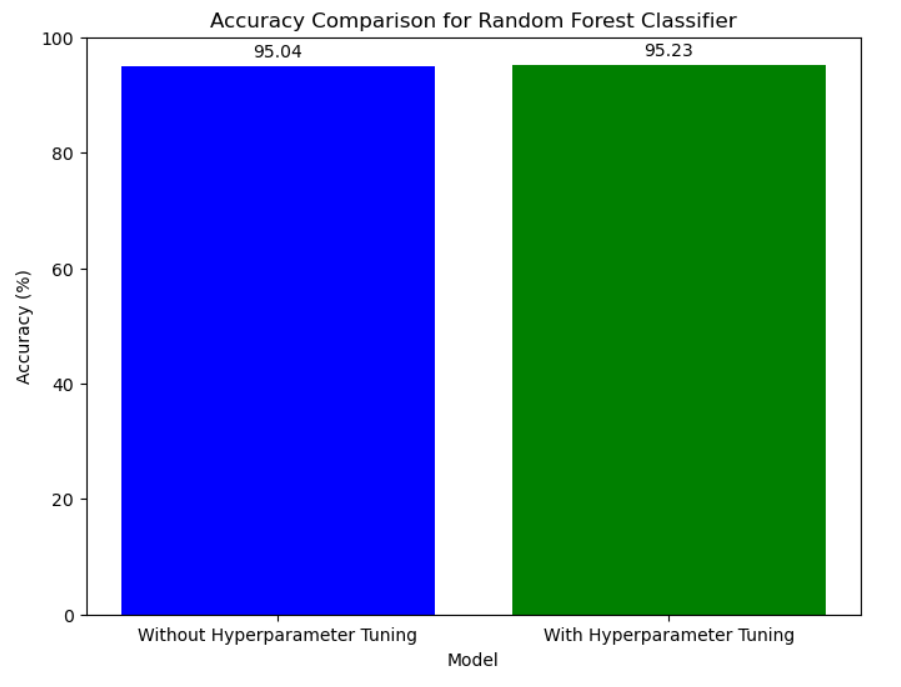


Figure 15: Impact of Hyperparameter Tuning on Random Forest

The figure15 illustrates a comparative analysis of accuracy between two models, one lacking hyperparameter tweaking and the other incorporating hyperparameter tuning. The accuracy of the model without tuning was found to be 94.14%, whereas the model that underwent tuning had a slightly better accuracy of 94.47%. A bar chart is employed to visually portray the data, with blue and green bars corresponding to the different models.

**Decision Tree**

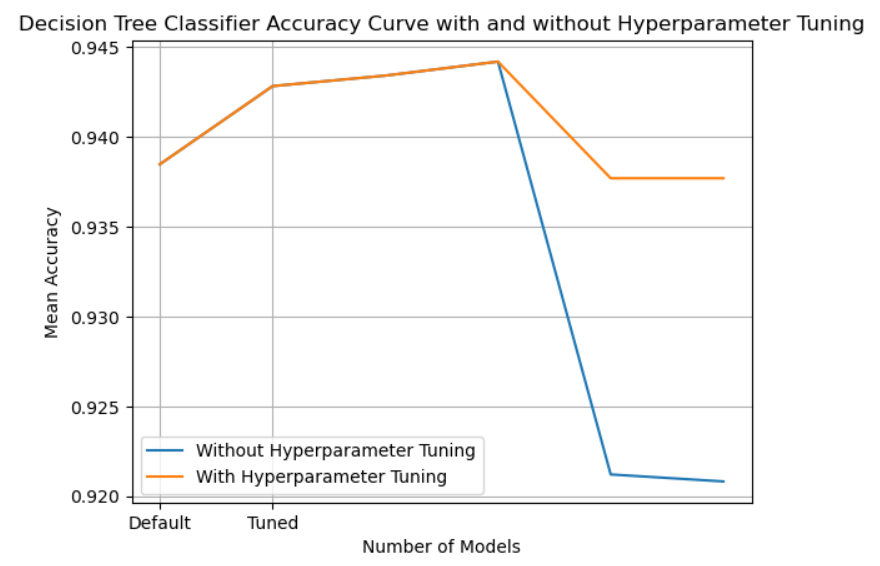


Figure 16: Decision Tree accuracy comparison with an without hyperparameter tuning

The figure 16 illustrates the accuracy curve of a Decision Tree Classifier, both with and without hyperparameter modification. The x-axis denotes the quantity of models, with "Default" and "Tuned" being the two data points. On the y-axis, the mean accuracy is displayed.   
In the absence of hyperparameter adjustment, the model attains a superior initial accuracy of around 0.942. However, during the process of tuning, the accuracy of the initial tuned model decreases to approximately 0.938. Subsequently, it exhibits an upward trend and constantly maintains a higher value of around 0.9447 when additional tuned models are incorporated.   
The obtained curve indicates that although the default model initially exhibits a higher level of accuracy, the implementation of appropriate hyperparameter tuning can ultimately result in improved overall performance, albeit with an initial decrease in accuracy when compared to the default configurations. Tuning enhances the model's parameters, leading to enhanced accuracy when further tuned models are taken into account.

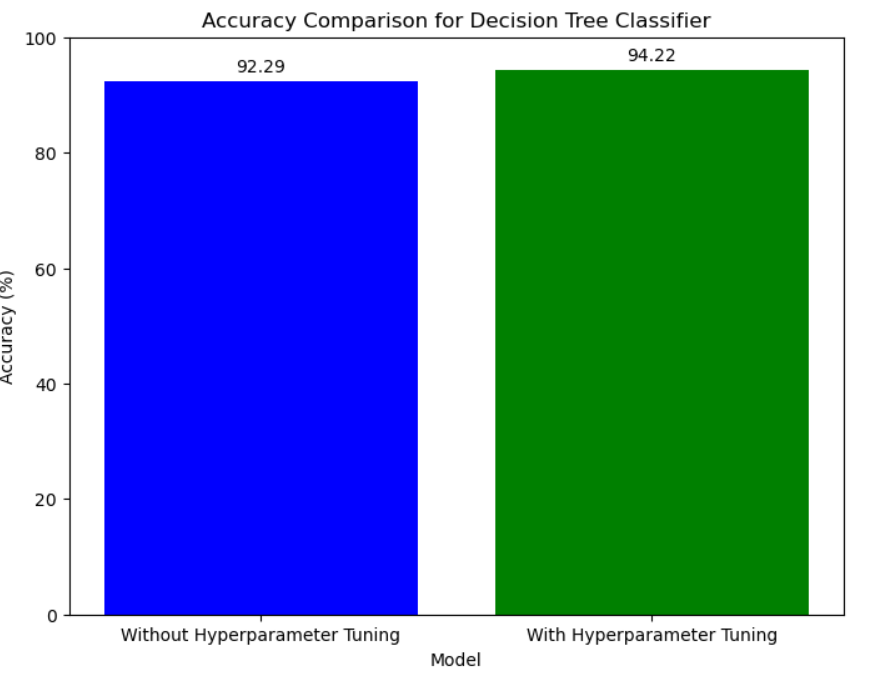


Figure 17: Impact of Hyperparameter Tuning on Decision tree

The image presents an accuracy comparison for a Decision Tree Classifier. It displays two bars, one blue and one green, representing the accuracy of the model with and without hyperparameter tuning, respectively.

The blue bar shows an accuracy of 92.29% for the model without hyperparameter tuning. On the other hand, the green bar indicates a higher accuracy of 94.22% when hyperparameter tuning is applied to the Decision Tree Classifier model.

The graph visually illustrates the positive impact of hyperparameter tuning on the performance of the Decision Tree Classifier, as evidenced by the increase in accuracy from 92.29% to 94.22%.

**Comparison Between Proposed System with Existing System:**

|  |  |  |
| --- | --- | --- |
| Model | Proposed system accuracy | Existing System accuracy |
| KNN | 94.470405 | 90 |
| Logistic Regression | 94.489875 | 86.76 |
| SVC | 94.548287 | 65.83 |
| Random Forest | 95.229751 | 76.8234 |
| Decision Tree | 94.217290 | 75 |

Table 7: Comparison of the proposed system with existing works

Table 7 compares the accuracy of a proposed system against existing systems across five different machine learning models: KNN, Logistic Regression, SVC, Random Forest, and Decision Tree. For each model, the accuracy of the proposed system is significantly higher than the existing systems. With the KNN model, the proposed system achieves 94.47% accuracy compared to 90% for existing systems. Logistic Regression shows 94.49% for the proposed system versus 86.76% for existing systems. The SVC model has 94.55% accuracy for the proposed system, markedly better than the 65.83% of existing systems. The Random Forest model demonstrates the largest accuracy gap, with the proposed system reaching 95.23% compared to only 76.82% for existing systems. Even for the Decision Tree model, the proposed system at 94.22% accuracy outperforms existing systems at 75% accuracy. Across the board, these results indicate the proposed system provides superior accuracy to current existing systems when applying these common machine learning models.

# Chapter 5 Impacts of the Project

## 5.1 Impact of this project on societal, health, safety, legal and cultural issues

**Societal Impact:** ML-based flood prediction models enable authorities to issue timely warnings and implement proactive measures, reducing the vulnerability of communities and minimizing loss of life and property.

Accurate flood forecasts facilitate targeted deployment of resources such as emergency response teams, medical supplies, and relief aid to affected areas, optimizing response efforts and mitigating humanitarian crises. Accessible and timely flood forecasts empower communities, particularly those in flood-prone areas, to make informed decisions regarding evacuation, asset protection, and livelihood preservation.

**Health Impact:** Early warning systems based on ML models can help mitigate health risks associated with floods by enabling timely evacuation and implementation of sanitation measures to prevent waterborne diseases. Flood prediction and early warning systems contribute to reducing anxiety and stress among affected populations by providing clarity and guidance during uncertain times, thereby promoting mental well-being.

**Safety Impact:** ML-driven flood prediction facilitates the design and construction of resilient infrastructure, including flood defenses, drainage systems, and building codes, to enhance community safety and mitigate damage during flood events.

Accurate flood forecasts enable authorities to implement road closures, reroute traffic, and coordinate transportation logistics, ensuring the safety of commuters and minimizing accidents and disruptions.

**Legal Impact:** ML-based flood prediction data can inform policy decisions and regulatory frameworks related to land use planning, disaster risk reduction, and climate change adaptation, leading to more effective governance and legislation. Improved flood prediction accuracy may impact legal liability and accountability, necessitating clear protocols and guidelines for stakeholders involved in flood management, including government agencies, private sector entities, and community organizations.

**Cultural Impact:** ML-based flood prediction contributes to safeguarding cultural heritage sites and artifacts from flood damage by enabling proactive measures such as relocation, preservation, and restoration efforts.

Effective flood prediction fosters community resilience and solidarity by promoting collective action, mutual support, and shared responsibility in preparing for and responding to flood events, preserving cultural values and traditions amidst adversity.

## 5.2 Impact of this project on environment and sustainability

The implementation of machine learning (ML) models for flood prediction in Bangladesh has profound implications for the environment and sustainability:

**Ecosystem preservation:** ML-based flood prediction enables early detection and mitigation of flood events, reducing habitat destruction, soil erosion, and disruption to ecosystems. By minimizing the impact of floods on natural landscapes, vegetation, and wildlife habitats, the project contributes to preserving biodiversity and ecological balance in flood-prone areas.

**Resource conservation:** Accurate flood forecasts generated by ML models facilitate efficient resource allocation and management, reducing wastage and optimizing the utilization of resources such as water, energy, and materials. By promoting sustainable resource practices, the project contributes to mitigating environmental degradation and supporting long-term ecosystem resilience.

**Climate resilience:** ML-driven flood prediction enhances the capacity of communities and authorities to adapt to climate change impacts, including increased frequency and intensity of extreme weather events such as floods. By providing early warning and risk assessment tools, the project enables proactive measures to mitigate climate-related risks and build climate resilience in vulnerable regions.

**Disaster risk reduction:** ML-based flood prediction supports effective disaster risk reduction strategies by identifying high-risk areas, vulnerable populations, and critical infrastructure prone to flood hazards. By integrating risk assessment tools into decision-making processes, the project helps minimize environmental and socio-economic losses associated with floods, fostering long-term sustainability and resilience in Bangladesh.

# Chapter 6 Project Planning and Budget

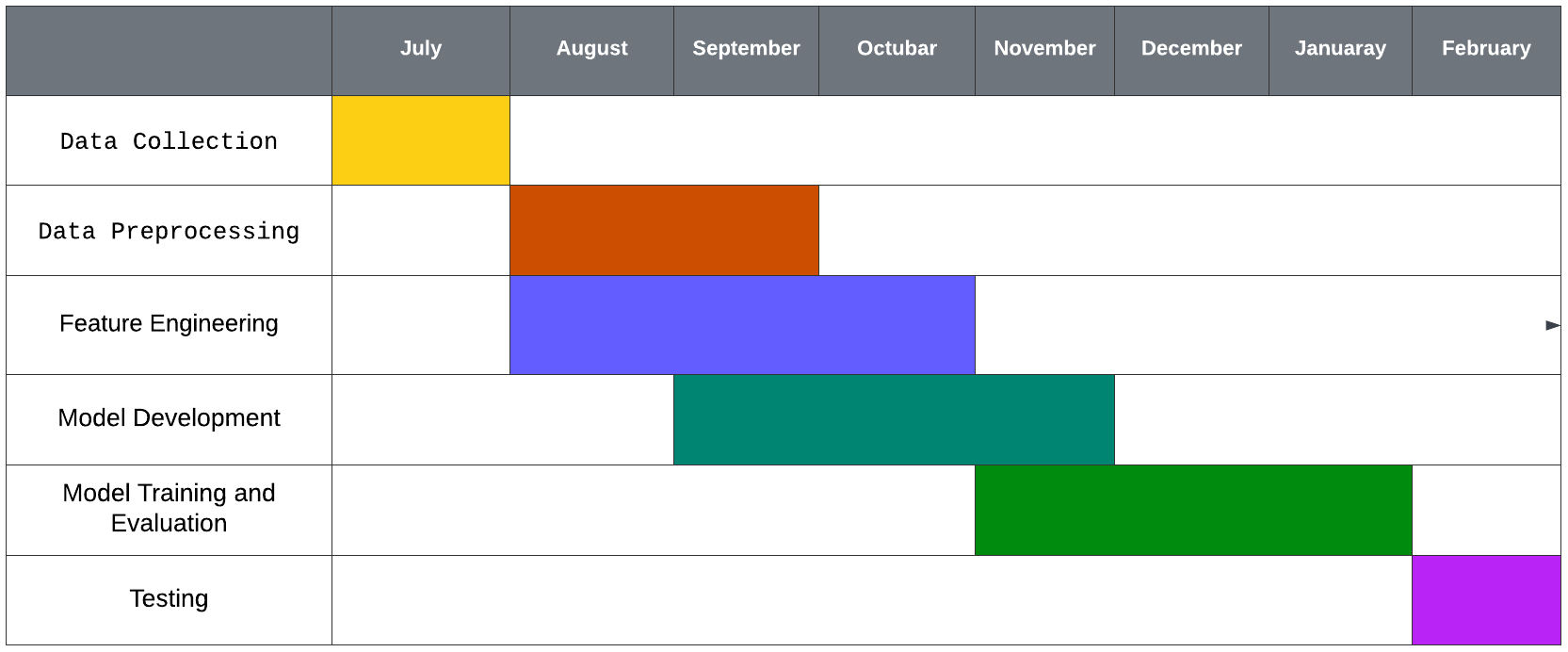


Figure 18: A sample Gantt chart

The Gantt chart presents a comprehensive timeline that outlines the various phases of a project or process, spanning from July to February. Each row delineates a specific task or activity, while the columns represent the corresponding timeline. Commencing in July, the initial phase involves Data Collection, depicted by a yellow bar. Subsequently, Data Preprocessing is scheduled for August and September, as indicated by an orange bar. Feature Engineering, represented by a blue bar, is slated to occur from September to November. Model Development, shown in green, is planned for October and November, partially overlapping with the previous task. The pivotal Model Training and Evaluation phase, also depicted in green, extends from November to January. Finally, the Testing phase, illustrated by a purple bar, is scheduled for January and February, marking the concluding stage of the project or process. This comprehensive visual representation facilitates effective planning, tracking, and coordination of the various interdependent tasks, ensuring a seamless and organized execution of the undertaking.

# Chapter 7 Complex Engineering Problems and Activities

## 7.1 Complex Engineering Problems (CEP)

Table 8: Demonstrates a sample complex engineering problem attribute

|  |  |  |
| --- | --- | --- |
| **Attributes** | | **Addressing the complex engineering problems (P) in the project** |
| P1 | Depth of knowledge required (K3-K8) | The project requires knowledge of required dataset, preprocessing, Machine Learning Model, training and testing |
| P2 | Range of conflicting requirements | Balancing the need for accurate flood prediction with the computational resources required to process large datasets and complex machine learning algorithms. Choosing between complex machine learning models that may offer higher predictive performance but are harder to interpret versus simpler models that are more easily understandable but may sacrifice some accuracy. |
| P3 | Depth of analysis required | Analyzing and comparing different machine learning algorithms (e.g., decision trees, random forests) to determine the most suitable approach for flood prediction based on factors such as accuracy, computational efficiency, and scalability. Fine-tuning model hyperparameters (e.g., learning rate, regularization strength) through systematic experimentation and optimization techniques (e.g., grid search, random search) to improve model performance and robustness. |
| P4 | Familiarity of issues | Proficiency in programming languages and development environments for implementing machine learning algorithms. |
| P5 | Extent of applicable codes | There is no existing code or standard for this project. |
| P6 | Extent of stakeholder involvement | Involving stakeholders from environmental agencies or ministries, such as the Ministry of Environment or environmental protection agencies, to gather data. |
| P7 | Interdependence | The machine learning model relies on preprocessed data for training and validation. The accuracy, completeness, and timeliness of the sensor data affect the quality of the trained model. Additionally, the computational resources available for model training and optimization influence the choice of machine learning algorithms and hyperparameter tuning strategies. |

## 7.2 Complex Engineering Activities (CEA)

Table 9: A Sample Complex Engineering Problem Activities

|  |  |  |
| --- | --- | --- |
| **Attributes** | | **Addressing the complex engineering activities (A) in the project** |
| A1 | Range of resources | Skilled personnel such as data scientists, machine learning engineers are needed to develop, train, and deploy the predictive models. |
| A2 | Level of interactions | Interaction with stakeholders such as the Ministry of Environment or other governmental agencies to collect historical flood data, weather data, and other relevant datasets for model training and validation. |
| A3 | Innovation | Employs innovative skills of engineering by introducing technology in a different manner in the environment and IoT sector |
| A4 | Consequences to society  / Environment | It doesn’t have any bad impact on environment. |
| A5 | Familiarity | Needs to be familiar with the various Model, Innovation, Logistic Regression, Decision Trees, Random Forest, Gradient Boosting Machines (GBM) Support Vector Machines (SVM), K-Nearest Neighbors (KNN) |

# Chapter 8 Conclusions

## 8.1 Summary

Machine learning techniques are increasingly being used in flood prediction to improve accuracy and efficiency. The offered accuracy scores provide insights into the performance of several models, each with its own set of strengths and hyperparameters. Starting with the K-Nearest Neighbours (KNN) model, it achieves a noteworthy accuracy of 94.47%. KNN works on the similarity principle, categorising instances based on the majority class of their k-nearest neighbours. The selection of the hyperparameter 'k' has a major impact on its predictive power.   
Despite its name, Logistic Regression is a linear model that is frequently employed for binary classification. It reaches a competitive accuracy of 94.49%. Tuning hyperparameters like regularisation strength can improve performance and provide resilience against overfitting.

Support Vector Machines (SVC) use the notion of identifying the best hyperplane to distinguish classes in high-dimensional space. With an accuracy score of 94.55%, SVC is effective in flood prediction. Hyperparameters such as kernel function selection and regularisation parameter play critical roles in performance. Random Forest, a flexible ensemble learning approach, achieves an accuracy score of 95.23%. Random Forest reduces overfitting by combining predictions from numerous decision trees. Tuning settings such as the number of trees and maximum depth can help to improve performance. Decision Trees provide a clear and understandable method to categorization jobs. Despite having a somewhat lower accuracy score of 94.22%, Decision Trees offer insights on feature relevance and decision-making processes. Hyperparameters such as maximum depth and minimum sample size per leaf have an impact on the tree's complexity and generalisation capabilities.

Each machine learning algorithm takes a distinct approach to flood prediction, with hyperparameter tweaking playing an important part in optimising predictive performance. To best meet the application's objectives, the model selection should take into account the trade-offs between interpretability, computational complexity, and accuracy.

## 8.2 Limitations

While the flood prediction project utilizing machine learning techniques shows promising results, several limitations may affect its effectiveness and applicability:

**Data Quality and Quantity:** The accuracy and reliability of predictions heavily rely on the quality and quantity of available data. Inadequate or biased data can lead to poor model performance and inaccurate predictions.

**Feature Selection:** Identifying relevant features that have a significant impact on flood occurrence is crucial. Incomplete or irrelevant feature selection may result in suboptimal models and reduced prediction accuracy.

**Model Overfitting:** Complex models, such as Random Forest, may be prone to overfitting, especially when trained on limited data. Overfitting can lead to high accuracy on training data but poor generalization to unseen data, undermining the model's effectiveness in real-world scenarios.

**Model Interpretability:** While some models like Decision Trees offer interpretability, more complex models like Random Forest or Support Vector Machines may lack transparency in their decision-making process. Understanding and explaining the model's predictions could be challenging, particularly in critical decision-making contexts.

**Computational Resources:** Training and optimizing machine learning models, especially complex ones, may require substantial computational resources and time. Limited resources could constrain the scalability and efficiency of the prediction system, especially in real-time applications.

**Imbalanced Data:** Imbalanced datasets, where one class (e.g., flood occurrence) is significantly more prevalent than others, can bias the model's predictions towards the majority class. Special techniques like resampling or adjusting class weights may be necessary to address this imbalance and improve model performance.

**Dynamic Nature of Flood Events:** Flood patterns and events can be highly dynamic and influenced by various environmental factors. Models trained on historical data may struggle to adapt to changing conditions or unforeseen events, necessitating continuous model updating and validation.

## 8.3 Future Improvement

Several avenues for future improvement of the flood prediction project using machine learning techniques can be explored:

**Integration of Remote Sensing Data:** Incorporating data from remote sensing sources such as satellite imagery, radar data, and aerial photography can provide valuable information on environmental variables like precipitation, soil moisture, and land cover. Integrating these datasets can enhance the accuracy and spatial resolution of flood predictions.

**Utilization of Advanced Modeling Techniques:** Experimenting with advanced machine learning and deep learning techniques such as recurrent neural networks (RNNs), convolutional neural networks (CNNs), and hybrid models can capture complex temporal and spatial dependencies in flood data. These models can improve prediction accuracy and enable the detection of subtle patterns and trends.

**Real-time Data Acquisition and Processing:** Developing systems capable of collecting and processing real-time data from IoT sensors, weather stations, river gauges, and social media platforms can enable timely and proactive flood prediction and early warning systems. Integrating data streams from multiple sources in a unified platform can enhance situational awareness and decision-making capabilities.

**Ensemble Learning Approaches:** Leveraging ensemble learning techniques such as model averaging, stacking, and boosting can combine the strengths of multiple models to improve prediction performance and robustness. Ensemble methods can mitigate the weaknesses of individual models and provide more reliable predictions, especially in uncertain or dynamic environments.

**Uncertainty Quantification and Risk Assessment**: Incorporating techniques for uncertainty quantification and risk assessment can enhance the reliability and trustworthiness of flood predictions. Bayesian inference, Monte Carlo simulations, and probabilistic modeling approaches can provide insights into prediction uncertainties and help stakeholders make informed decisions under uncertainty.

**User-Centric Design and Stakeholder Engagement:** Engaging stakeholders, including government agencies, emergency responders, local communities, and residents, in the design and development process can ensure that the flood prediction system meets their needs and addresses their concerns. User-centric design principles and participatory approaches can enhance system usability, acceptance, and effectiveness.

**Continuous Monitoring and Model Updating**: Implementing mechanisms for continuous monitoring, validation, and updating of predictive models can ensure their relevance and accuracy over time. Incorporating feedback loops and adaptive learning algorithms can enable the system to adapt to changing environmental conditions, evolving data distributions, and emerging flood risk factors.

**Ethical and Regulatory Compliance:** Adhering to ethical guidelines, privacy regulations, and data protection laws is essential to building trust and accountability in the flood prediction system. Implementing transparent governance structures, data anonymization techniques, and privacy-preserving mechanisms can mitigate potential risks and ensure responsible use of data and technology.

# References

[1] N. Gauhar, S. Das, and K. S. Moury, “Prediction of flood in Bangladesh using K-nearest neighbors algorithm,” *2021 2nd International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST)*, Jan. 2021. doi:10.1109/icrest51555.2021.9331199

[2] A. Rajab *et al.*, “Flood forecasting by using Machine Learning: A Study Leveraging Historic Climatic Records of Bangladesh,” *Water*, vol. 15, no. 22, p. 3970, Nov. 2023. doi:10.3390/w15223970

[3] A. K. Ambore, T. S. Charan, U. R. Reddy, T. S. Reddy, and Tarun. G, “Flood prediction using machine learning,” *International Journal for Research in Applied Science and Engineering Technology*, vol. 11, no. 5, pp. 363–367, May 2023. doi:10.22214/ijraset.2023.51528

[4] K. Kunverji, K. Shah, and N. Shah, “A flood prediction system developed using various machine learning algorithms,” *SSRN Electronic Journal*, 2021. doi:10.2139/ssrn.3866524

[5] B. Setya, R. A. Nurhidayatullah, M. B. Hewen, and K. Kusrini, “Comparative analysis of rainfall value prediction in Semarang using linear and K-nearest neighbor algorithms,” *2023 5th International Conference on Cybernetics and Intelligent System (ICORIS)*, Oct. 2023. doi:10.1109/icoris60118.2023.10352274

[6] C. S. Bangera, P. S. Kotian, C. Dias, T. Divya, and G. Aithal, “Flood and heat wave prediction using weighted moving average, anomaly detection and K-nearest neighbours for the city of Mangalore,” *2018 IEEE Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER)*, Aug. 2018. doi:10.1109/discover.2018.8674113

[7] D. Mahajan and S. Sharma, “Prediction of rainfall using machine learning,” *2022 Fourth International Conference on Emerging Research in Electronics, Computer Science and Technology (ICERECT)*, Dec. 2022. doi:10.1109/icerect56837.2022.10059679

[8] A. Raut, D. Theng, and S. Khandelwal, “Random forest regressor model for rainfall prediction,” *2023 International Conference on New Frontiers in Communication, Automation, Management and Security (ICCAMS)*, Oct. 2023. doi:10.1109/iccams60113.2023.10526085

[9] MD. E. Deowan *et al.*, “Smart early flood monitoring system using IOT,” *2022 14th Seminar on Power Electronics and Control (SEPOC)*, Nov. 2022. doi:10.1109/sepoc54972.2022.9976434

[10] L. J. Canillo and A. A. Hernandez, “User acceptance of flood risk visualization and prediction information system,” *International Journal of Enterprise Information Systems*, vol. 17, no. 3, pp. 16–33, Jul. 2021. doi:10.4018/ijeis.2021070102

[11] M. M. Syeed *et al.*, “Flood prediction using machine learning models,” *2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, Jun. 2022. doi:10.1109/hora55278.2022.9800023

[12] Felix, A.Y. and Sasipraba, T. (2019) ‘Flood detection using gradient boost machine learning approach’, *2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE)* [Preprint]. doi:10.1109/iccike47802.2019.9004419.

[13] Riza, H. *et al.* (2020) ‘Advancing flood disaster mitigation in Indonesia using machine learning methods’, *2020 International Conference on ICT for Smart Society (ICISS)* [Preprint]. doi:10.1109/iciss50791.2020.9307561.

# Appendix

import pandas as pd

data = pd.read\_csv('floodpredicton.csv')

print(data)

data.head()

data.tail()

data.isnull().sum()

data['Flood?'] = data['Flood?'].fillna(data['Flood?'].mean())

print(data.shape)

data.describe()

data.info

data.dtypes

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.preprocessing import LabelEncoder

for column in data.select\_dtypes(include='object'):

encoder = LabelEncoder()

data[column] = encoder.fit\_transform(data[column])

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

plt.scatter(x = 'Cloud\_Coverage', y = 'Flood?', data = data)

# scatter plot with pyplot

plt.xlabel('Cloud\_Coverage')

plt.ylabel('Flood?')

plt.title('Scatterplot of Cloud\_Coverage against Flood');

sns.scatterplot(x = 'LATITUDE', y = 'Flood?', data = data);

sns.scatterplot(x = 'Bright\_Sunshine', y = 'Flood?', data = data);

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

%matplotlib inline

data.head()

sns.kdeplot(x = "Flood?", data = data, fill = True)

sns.kdeplot(x = "Flood?", data = data, fill = True)

sns.histplot(x = "Station\_Number", data = data);

sns.histplot(x = "Relative\_Humidity", data = data);

sns.histplot(x = "Cloud\_Coverage", data = data);

sns.histplot(x = "Bright\_Sunshine", data = data);

sns.histplot(x = "Flood?", data = data);

import random

colorlist = ['red', 'yellow', 'orange', 'purple', 'blue']

sns.boxplot(x = 'LATITUDE', data = data, color = random.choice(colorlist)); # horizontal box plot

colorlist = ['red', 'yellow', 'orange', 'purple', 'blue']

sns.boxplot(x = 'Min\_Temp', data = data, color = random.choice(colorlist)); # horizontal box plot

colorlist = ['red', 'yellow', 'orange', 'purple', 'blue']

sns.boxplot(x = 'Cloud\_Coverage', data = data, color = random.choice(colorlist)); # horizontal box plot

colorlist = ['red', 'yellow', 'orange', 'purple', 'blue']

sns.boxplot(x = 'X\_COR', data = data, color = random.choice(colorlist)); # horizontal box plot

colorlist = ['red', 'yellow', 'orange', 'purple', 'blue']

sns.boxplot(x = 'Station\_Number', data = data, color = random.choice(colorlist)); # horizontal box plot

colorlist = ['red', 'yellow', 'orange', 'purple', 'blue']

sns.boxplot(x = 'LONGITUDE', data = data, color = random.choice(colorlist)); # horizontal box plot

colorlist = ['red', 'yellow', 'orange', 'purple', 'blue']

sns.boxplot(x = 'Period', data = data, color = random.choice(colorlist)); # horizontal box plot

correlation\_matrix = data.corr()

plt.figure(figsize=(32, 16))

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', fmt='.2f')

plt.show()

# Now seperate the flood label from the dataset.

y=data.iloc[:,-1]

y

x=data.iloc[:,2:11]

x.head()

# Scaling the data between 0 and 1.

from sklearn import preprocessing

minmax = preprocessing.MinMaxScaler(feature\_range=(0,1))

minmax.fit(x).transform(x)

#dividing the dataset into training dataset and test dataset.

from sklearn import model\_selection,neighbors

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

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.25)

x\_train.head()

x\_train.dtypes

x\_test.head()

# type casting.

y\_train=y\_train.astype('int')

y\_train

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

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score

import matplotlib.pyplot as plt

%matplotlib inline

import seaborn as sns

# Scaling the dataset.

from sklearn.model\_selection import cross\_val\_score,cross\_val\_predict

x\_train\_std= minmax.fit\_transform(x\_train)

x\_test\_std= minmax.fit\_transform(x\_test)

knn\_default = KNeighborsClassifier()

knn\_default.fit(x\_train, y\_train)

y\_test=y\_test.astype('int')

y\_test

# Predicted chance of Flood.

print("Predicted Values for the Floods:")

y\_predict=knn\_default.predict(x\_test)

y\_predict

print("Actual Values for the Floods:")

print(y\_test)

print("List of the Predicted Values:")

print(y\_predict)

y\_pred\_default = knn\_default.predict(x\_test)

accuracy\_default = accuracy\_score(y\_test, y\_pred\_default)

print("Accuracy without hyperparameters:", accuracy\_default)

from sklearn.metrics import accuracy\_score,recall\_score,roc\_auc\_score,confusion\_matrix

print("\nAccuracy Score:%f"%(accuracy\_score(y\_test,y\_pred\_default)\*100))

print("Recall Score:%f"%(recall\_score(y\_test,y\_pred\_default)\*100))

print("ROC score:%f"%(roc\_auc\_score(y\_test,y\_pred\_default)\*100))

print(confusion\_matrix(y\_test,y\_predict))

from sklearn.model\_selection import GridSearchCV

import numpy as np

import matplotlib.pyplot as plt

param\_grid = {'n\_neighbors': np.arange(1, 21)}

knn = KNeighborsClassifier()

grid\_search = GridSearchCV(knn, param\_grid, cv=5)

grid\_search.fit(x\_train, y\_train)

# Print the best parameters and best score

print("Best Parameters:", grid\_search.best\_params\_)

print("Best Score:", grid\_search.best\_score\_)

best\_knn = grid\_search.best\_estimator\_

y\_pred\_tuned = best\_knn.predict(x\_test)

accuracy\_tuned = accuracy\_score(y\_test, y\_pred\_tuned)

print("Accuracy with hyperparameter tuning:", accuracy\_tuned)

print("\nAccuracy Score:%f"%(accuracy\_score(y\_test,y\_pred\_tuned)\*100))

print("Recall Score:%f"%(recall\_score(y\_test,y\_pred\_tuned)\*100))

print("ROC score:%f"%(roc\_auc\_score(y\_test,y\_pred\_tuned)\*100))

print(confusion\_matrix(y\_test,y\_pred\_tuned))

# Plotting the accuracy for kNN classifier without hyperparameter tuning

plt.figure(figsize=(10, 6))

plt.plot(np.arange(1, 21), grid\_search.cv\_results\_['mean\_test\_score'], marker='o', linestyle='-', label='With Hyperparameter Tuning')

plt.title('Accuracy vs. Number of Neighbors (k)')

plt.xlabel('Number of Neighbors (k)')

plt.ylabel('Accuracy')

plt.xticks(np.arange(1, 21))

plt.grid(True)

# Plotting the accuracy for kNN classifier with hyperparameter tuning

plt.plot(np.arange(1, 21), [accuracy\_default]\*20, linestyle='--', label='Without Hyperparameter Tuning')

plt.legend()

plt.show()

# Define accuracy values for both scenarios

accuracy\_values = [accuracy\_default \* 100, accuracy\_tuned \* 100]

labels = ['Without Hyperparameter Tuning', 'With Hyperparameter Tuning']

# Plotting the accuracy for both scenarios using a bar graph

plt.figure(figsize=(8, 6))

plt.bar(labels, accuracy\_values, color=['blue', 'green'])

plt.title('Accuracy Comparison')

plt.xlabel('Model')

plt.ylabel('Accuracy (%)')

plt.ylim(0, 100)

# Adding the accuracy values on top of the bars

for i, v in enumerate(accuracy\_values):

plt.text(i, v + 1, str(round(v, 2)), ha='center', va='bottom')

plt.show()

from sklearn.linear\_model import LogisticRegression

log\_reg\_default = LogisticRegression()

log\_reg\_default.fit(x\_train, y\_train)

y\_pred\_default = log\_reg\_default.predict(x\_test)

accuracy\_default = accuracy\_score(y\_test, y\_pred\_default)

print("Accuracy without hyperparameter tuning:", accuracy\_default)

print("\naccuracy score:%f"%(accuracy\_score(y\_test,y\_pred\_default)\*100))

print("recall score:%f"%(recall\_score(y\_test,y\_pred\_default)\*100))

print("roc score:%f"%(roc\_auc\_score(y\_test,y\_pred\_default)\*100))

print(confusion\_matrix(y\_test,y\_pred\_default))

# Define the parameter grid to search

param\_grid = {'C': np.logspace(-3, 3, 100,1000)}

# Create Logistic Regression classifier

lr = LogisticRegression(solver='lbfgs', multi\_class='auto', max\_iter=1000)

# Perform Grid Search with cross-validation

grid\_search = GridSearchCV(lr, param\_grid, cv=5)

grid\_search.fit(x\_train, y\_train)

# Print the best parameters and best score

print("Best Parameters:", grid\_search.best\_params\_)

print("Best Score:", grid\_search.best\_score\_)

# Get the best model

best\_lr = grid\_search.best\_estimator\_

# Make predictions using the best model

y\_train\_pred = best\_lr.predict(x\_train)

y\_test\_pred = best\_lr.predict(x\_test)

# Calculate accuracy

train\_accuracy = accuracy\_score(y\_train, y\_train\_pred)

test\_accuracy = accuracy\_score(y\_test, y\_test\_pred)

print("Best accuracy on training set:", train\_accuracy)

print("Best accuracy on testing set:", test\_accuracy)

# Plotting the accuracy curve for logistic regression models with and without hyperparameter tuning

plt.figure(figsize=(10, 6))

# Plotting the accuracy curve for logistic regression without hyperparameter tuning

plt.semilogx(param\_grid['C'], grid\_search.cv\_results\_['mean\_test\_score'], label='With Hyperparameter Tuning', marker='o', linestyle='-')

# Plotting the accuracy curve for logistic regression with hyperparameter tuning

plt.axhline(y=train\_accuracy, color='red', linestyle='--', label='Without Hyperparameter Tuning')

plt.title('Mean Cross-Validation Accuracy vs. Regularization Strength (C)')

plt.xlabel('Regularization Strength (C)')

plt.ylabel('Mean Cross-Validation Accuracy')

plt.grid(True)

plt.legend()

plt.show()

print("\naccuracy score:%f"%(accuracy\_score(y\_test,y\_test\_pred)\*100))

print("recall score:%f"%(recall\_score(y\_test,y\_test\_pred)\*100))

print("roc score:%f"%(roc\_auc\_score(y\_test,y\_test\_pred)\*100))

print(confusion\_matrix(y\_test,y\_test\_pred))

# Plotting the comparison between logistic regression models with and without hyperparameter tuning

plt.figure(figsize=(8, 6))

accuracy\_values = [accuracy\_default \* 100, test\_accuracy \* 100]

labels = ['Without Hyperparameter Tuning', 'With Hyperparameter Tuning']

plt.bar(labels, accuracy\_values, color=['blue', 'green'])

plt.title('Accuracy Comparison for Logistic Regression')

plt.xlabel('Model')

plt.ylabel('Accuracy (%)')

plt.ylim(0, 100)

# Adding the accuracy values on top of the bars

for i, v in enumerate(accuracy\_values):

plt.text(i, v + 1, str(round(v, 2)), ha='center', va='bottom')

plt.show()

from sklearn.svm import SVC

from sklearn.model\_selection import cross\_val\_score

svc\_default = SVC()

# Train the classifier

svc\_default.fit(x\_train, y\_train)

# Make predictions

y\_pred\_default = svc\_default.predict(x\_test)

# Calculate accuracy

accuracy\_default = accuracy\_score(y\_test, y\_pred\_default)

print("Accuracy without hyperparameter tuning:", accuracy\_default)

print("\naccuracy score:%f"%(accuracy\_score(y\_test,y\_pred\_default)\*100))

print("recall score:%f"%(recall\_score(y\_test,y\_pred\_default)\*100))

print("roc score:%f"%(roc\_auc\_score(y\_test,y\_pred\_default)\*100))

print(confusion\_matrix(y\_test,y\_pred\_default))

param\_grid = {'C': [0.1, 1, 10, 100]}

svc = SVC()

# Perform Grid Search with cross-validation

grid\_search = GridSearchCV(svc, param\_grid, cv=5)

grid\_search.fit(x\_train, y\_train)

# Get the best model

best\_svc = grid\_search.best\_estimator\_

y\_pred\_tuned = best\_svc.predict(x\_test)

accuracy\_tuned = accuracy\_score(y\_test, y\_pred\_tuned)

print("Accuracy with hyperparameter tuning:", accuracy\_tuned)

print("\nAccuracy Score:%f"%(accuracy\_score(y\_test,y\_pred\_tuned)\*100))

print("Recall Score:%f"%(recall\_score(y\_test,y\_pred\_tuned)\*100))

print("ROC score:%f"%(roc\_auc\_score(y\_test,y\_pred\_tuned)\*100))

print(confusion\_matrix(y\_test,y\_pred\_tuned))

# Get the best parameters

best\_params = grid\_search.best\_params\_

best\_score = grid\_search.best\_score\_

print("Best parameters:", best\_params)

# Define a range of values for the hyperparameter C

C\_range = [0.1, 1, 10, 100]

# Initialize lists to store mean accuracies for both scenarios

mean\_accuracies\_default = []

mean\_accuracies\_tuned = []

for C\_value in C\_range:

# Without hyperparameter tuning

svc\_default = SVC(C=C\_value)

accuracies\_default = cross\_val\_score(svc\_default, x\_train, y\_train, cv=5)

mean\_accuracies\_default.append(np.mean(accuracies\_default))

# With hyperparameter tuning

svc\_tuned = SVC(C=C\_value)

grid\_search = GridSearchCV(svc\_tuned, param\_grid={'C': [C\_value]}, cv=5)

grid\_search.fit(x\_train, y\_train)

best\_svc = grid\_search.best\_estimator\_

accuracies\_tuned = cross\_val\_score(best\_svc, x\_train, y\_train, cv=5)

mean\_accuracies\_tuned.append(np.mean(accuracies\_tuned))

# Plot the accuracy curves for both scenarios

plt.plot(C\_range, mean\_accuracies\_default, label='Without Hyperparameter Tuning')

plt.plot(C\_range, mean\_accuracies\_tuned, label='With Hyperparameter Tuning')

plt.title('SVC Accuracy Curve with and without Hyperparameter Tuning')

plt.xlabel('C (Regularization strength)')

plt.ylabel('Mean Accuracy')

plt.xscale('log') # Use logarithmic scale for better visualization if C values are wide-ranging

plt.legend()

plt.grid(True)

plt.show()

# Define the accuracy values for SVC models with and without hyperparameter tuning

accuracy\_values\_svc = [accuracy\_default \* 100, accuracy\_tuned \* 100]

# Labels for the bars

labels\_svc = ['Without Hyperparameter Tuning', 'With Hyperparameter Tuning']

# Plotting the comparison

plt.figure(figsize=(8, 6))

plt.bar(labels\_svc, accuracy\_values\_svc, color=['blue', 'green'])

plt.title('Accuracy Comparison for SVC')

plt.xlabel('Model')

plt.ylabel('Accuracy (%)')

plt.ylim(0, 100)

# Adding the accuracy values on top of the bars

for i, v in enumerate(accuracy\_values\_svc):

plt.text(i, v + 1, str(round(v, 2)), ha='center', va='bottom')

plt.show()

from sklearn.ensemble import RandomForestClassifier

rf\_default = RandomForestClassifier()

# Train the classifier

rf\_default.fit(x\_train, y\_train)

y\_pred\_default = rf\_default.predict(x\_test)

# Calculate accuracy

accuracy\_default = accuracy\_score(y\_test, y\_pred\_default)

print("Accuracy without hyperparameter tuning:", accuracy\_default)

print("\naccuracy score:%f"%(accuracy\_score(y\_test,y\_pred\_default)\*100))

print("recall score:%f"%(recall\_score(y\_test,y\_pred\_default)\*100))

print("roc score:%f"%(roc\_auc\_score(y\_test,y\_pred\_default)\*100))

print(confusion\_matrix(y\_test,y\_pred\_default))

param\_grid = {

'n\_estimators': [50, 100, 150],

'max\_depth': [None, 10, 20],

'min\_samples\_split': [2, 5, 10],

'min\_samples\_leaf': [1, 2, 4]

}

rf\_classifier\_tuned = RandomForestClassifier(random\_state=42)

# Grid search with cross-validation

grid\_search = GridSearchCV(estimator=rf\_classifier\_tuned, param\_grid=param\_grid, cv=5, n\_jobs=-1)

grid\_search.fit(x\_train, y\_train)

# Print the best parameters and best score

print("Best Parameters:", grid\_search.best\_params\_)

print("Best Score:", grid\_search.best\_score\_)

# Make predictions on the test set using the best model

best\_rf\_classifier = grid\_search.best\_estimator\_

y\_pred\_tuned = best\_rf\_classifier.predict(x\_test)

print("Best parameters:", best\_params)

print("Best cross-validation accuracy:", best\_score)

# Calculate accuracy

accuracy\_with\_tuning = accuracy\_score(y\_test, y\_pred\_tuned)

print("Accuracy with hyperparameter tuning:", accuracy\_with\_tuning)

print("\nAccuracy Score:%f"%(accuracy\_score(y\_test, y\_pred\_tuned)\*100))

print("Recall Score:%f"%(recall\_score(y\_test, y\_pred\_tuned)\*100))

print("ROC score:%f"%(roc\_auc\_score(y\_test, y\_pred\_tuned)\*100))

print(confusion\_matrix(y\_test, y\_pred\_tuned))

# Generate predictions with the best model

y\_pred\_tuned = best\_rf\_classifier.predict(x\_test)

# Create the confusion matrix

cm = confusion\_matrix(y\_test, y\_pred\_tuned)

ConfusionMatrixDisplay(confusion\_matrix=cm).plot()

# Create a series containing feature importances from the model and feature names from the training data

feature\_importances = pd.Series(best\_rf\_classifier.feature\_importances\_, index=x\_train.columns).sort\_values(ascending=False)

# Plot a simple bar chart

feature\_importances.plot.bar();

import matplotlib.pyplot as plt

# Define accuracy values for Random Forest Classifier with and without hyperparameter tuning

accuracy\_values\_rf = [accuracy\_default \* 100, accuracy\_with\_tuning \* 100] # Changed accuracy\_tuned to accuracy\_with\_tuning

# Labels for the bars

labels\_rf = ['Without Hyperparameter Tuning', 'With Hyperparameter Tuning']

# Plotting the comparison

plt.figure(figsize=(8, 6))

plt.bar(labels\_rf, accuracy\_values\_rf, color=['blue', 'green'])

plt.title('Accuracy Comparison for Random Forest Classifier')

plt.xlabel('Model')

plt.ylabel('Accuracy (%)')

plt.ylim(0, 100)

# Adding the accuracy values on top of the bars

for i, v in enumerate(accuracy\_values\_rf):

plt.text(i, v + 1, str(round(v, 2)), ha='center', va='bottom')

plt.show()

from sklearn.tree import DecisionTreeClassifier

from sklearn.model\_selection import GridSearchCV

dt\_default = DecisionTreeClassifier()

# Train the classifier

dt\_default.fit(x\_train, y\_train)

# Make predictions

y\_pred\_default = dt\_default.predict(x\_test)

# Calculate accuracy

accuracy\_default = accuracy\_score(y\_test, y\_pred\_default)

print("Accuracy without hyperparameter tuning:", accuracy\_default)

print("\naccuracy score:%f"%(accuracy\_score(y\_test,y\_pred\_default)\*100))

print("recall score:%f"%(recall\_score(y\_test,y\_pred\_default)\*100))

print("roc score:%f"%(roc\_auc\_score(y\_test,y\_pred\_default)\*100))

print(confusion\_matrix(y\_test,y\_pred\_default))

param\_grid = {

'criterion': ['gini', 'entropy'],

'max\_depth': [None, 10, 20, 30],

'min\_samples\_split': [2, 5, 10],

'min\_samples\_leaf': [1, 2, 4]

}

# Instantiate the Decision Tree classifier

dt\_classifier\_tuned = DecisionTreeClassifier(random\_state=42)

# Grid search with cross-validation

grid\_search = GridSearchCV(estimator=dt\_classifier\_tuned, param\_grid=param\_grid, cv=5, n\_jobs=-1)

grid\_search.fit(x\_train, y\_train)

# Print the best parameters and best score

print("Best Parameters:", grid\_search.best\_params\_)

print("Best Score:", grid\_search.best\_score\_)

# Make predictions on the test set using the best model

best\_dt\_classifier = grid\_search.best\_estimator\_

y\_pred\_tuned = best\_dt\_classifier.predict(x\_test)

# Calculate accuracy

accuracy\_with\_tuning = accuracy\_score(y\_test, y\_pred\_tuned)

print("Accuracy with hyperparameter tuning:", accuracy\_with\_tuning)

print("\nAccuracy Score:%f"%(accuracy\_score(y\_test, y\_pred\_tuned)\*100))

print("Recall Score:%f"%(recall\_score(y\_test, y\_pred\_tuned)\*100))

print("ROC score:%f"%(roc\_auc\_score(y\_test, y\_pred\_tuned)\*100))

print(confusion\_matrix(y\_test, y\_pred\_tuned))

# Instantiate the Decision Tree classifier without hyperparameter tuning

accuracies\_default = cross\_val\_score(dt\_default, x\_train, y\_train, cv=5)

mean\_accuracies\_default.append(np.mean(accuracies\_default))

# Grid search with cross-validation for hyperparameter tuning

best\_dt = grid\_search.best\_estimator\_

accuracies\_tuned = cross\_val\_score(best\_dt, x\_train, y\_train, cv=5)

mean\_accuracies\_tuned.append(np.mean(accuracies\_tuned))

# Plot the accuracy curves for both scenarios

plt.plot(np.arange(len(mean\_accuracies\_default)), mean\_accuracies\_default, label='Without Hyperparameter Tuning')

plt.plot(np.arange(len(mean\_accuracies\_tuned)), mean\_accuracies\_tuned, label='With Hyperparameter Tuning')

plt.title('Decision Tree Classifier Accuracy Curve with and without Hyperparameter Tuning')

plt.xlabel('Number of Models')

plt.ylabel('Mean Accuracy')

plt.xticks([0, 1], ['Default', 'Tuned'])

plt.legend()

plt.grid(True)

plt.show()

# Define accuracy values for Decision Tree Classifier with and without hyperparameter tuning

accuracy\_values\_dt = [accuracy\_default \* 100, accuracy\_with\_tuning \* 100] # Using accuracy values from the Decision Tree Classifier

# Labels for the bars

labels\_dt = ['Without Hyperparameter Tuning', 'With Hyperparameter Tuning']

# Plotting the comparison

plt.figure(figsize=(8, 6))

plt.bar(labels\_dt, accuracy\_values\_dt, color=['blue', 'green'])

plt.title('Accuracy Comparison for Decision Tree Classifier')

plt.xlabel('Model')

plt.ylabel('Accuracy (%)')

plt.ylim(0, 100)

# Adding the accuracy values on top of the bars

for i, v in enumerate(accuracy\_values\_dt):

plt.text(i, v + 1, str(round(v, 2)), ha='center', va='bottom')

plt.show()

import lime

import lime.lime\_tabular

clf = RandomForestClassifier(random\_state=42)

clf.fit(x\_train, y\_train)

# Make predictions on the test set

y\_pred = clf.predict(x\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

print("Accuracy:", accuracy)

explainer = lime.lime\_tabular.LimeTabularExplainer(x\_train.values,

feature\_names=x\_train.columns.values.tolist(),

class\_names=['No Flood', 'Flood'],

verbose=True,

mode='classification')

# Choose a sample for explanation

sample\_idx = 0

sample = x\_test.iloc[sample\_idx]

true\_class = y\_test.iloc[sample\_idx]

# Explain the prediction

exp = explainer.explain\_instance(sample.values, clf.predict\_proba, num\_features=len(x\_train.columns))

exp.show\_in\_notebook(show\_table=True)

# Choose the 14th instance for prediction

sample\_idx = 13 # Note that Python uses 0-based indexing, so the 5th instance corresponds to index 4

sample = x\_test.iloc[sample\_idx]

true\_class = y\_test.iloc[sample\_idx]

# Predict the result for the 5th instance

prediction = clf.predict\_proba([sample.values])[0]

# Print the prediction probabilities

print("Prediction Probabilities:")

for i, prob in enumerate(prediction):

print(f"Class {i}: {prob}")

# Explain the prediction using LIME

exp = explainer.explain\_instance(sample.values, clf.predict\_proba, num\_features=len(x\_train.columns))

exp.show\_in\_notebook(show\_table=True)

# Choose the 16th instance for prediction

sample\_idx = 15 # Note that Python uses 0-based indexing, so the 10th instance corresponds to index 9

sample = x\_test.iloc[sample\_idx]

true\_class = y\_test.iloc[sample\_idx]

# Predict the result for the 10th instance

prediction = clf.predict\_proba([sample.values])[0]

# Print the prediction probabilities

print("Prediction Probabilities:")

for i, prob in enumerate(prediction):

print(f"Class {i}: {prob}")

# Explain the prediction using LIME

exp = explainer.explain\_instance(sample.values, clf.predict\_proba, num\_features=len(x\_train.columns))

exp.show\_in\_notebook(show\_table=True)

# Choose the 20th instance for prediction

sample\_idx = 19 # Note that Python uses 0-based indexing, so the 10th instance corresponds to index 9

sample = x\_test.iloc[sample\_idx]

true\_class = y\_test.iloc[sample\_idx]

# Predict the result for the 10th instance

prediction = clf.predict\_proba([sample.values])[0]

# Print the prediction probabilities

print("Prediction Probabilities:")

for i, prob in enumerate(prediction):

print(f"Class {i}: {prob}")

# Explain the prediction using LIME

exp = explainer.explain\_instance(sample.values, clf.predict\_proba, num\_features=len(x\_train.columns))

exp.show\_in\_notebook(show\_table=True)