A MINI PROJECT

on

"FLOOD PREDICTION USING MACHINE LEARNING"

Submitted

In partial fulfillment for the requirement for the award of degree of

BACHELOR OF TECHNOLOGY

in

Computer Science and Engineering (AI & ML).

By

D.HARSHITHA	**20641A6658**
D.SHRAVYA	**20641A6607**
K.VISHWAS	**20641A6611**
CH.SIDHARTHA	**20641A6655**

Under the Guidance of

SALIM AMIRALI JIWANI

Associate Professor, Department of CSE (AI&ML).

Department of Computer Science & Engineering (AI & ML)

Vaagdevi College of Engineering

(UGC Autonomous, Accredited by NAAC with "A")

Bollikunta, Khila Warangal (Mandal), Warangal Urban – 506005(T.S)

(2023-2024)

VAAGDEVI COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING(AI & ML)

(UGC Autonomous, Accredited by NAAC with "A")

Bollikunta, Khila Warangal (Mandal), Warangal Urban – 506005(T.S)

(2023-2024)

CERTIFICATE

This is to certify that the m i n i project titled **" FLOOD PREDICTION USING MACHINE LEARNING"** is a bonafide record of work done by ***D.HARSHITHA, D.SHRAVYA, K.VISHWAS, CH.SIDHARTHA*** bearing ***20641A6658, 20641A6607, 20641A6611, 20641A6655*** in partial fulfillment of the requirements for the award of the degree in Bachelor of Technology in Computer Science & Engineering (AI & ML) for the academic year 2023-2024.

Project Guide **Head of the Department** **External Examiner**

Submitted for the Project Viva-Voice Examination on_____

ACKNOWLEDGEMENT

We wish to take this opportunity to express my sincere gratitude and deep sense of respect to beloved **Dr.K.Prakash, Principal** of Vaagdevi College of Engineering for making me available all the required assistance and for his support and inspiration to carry out this industry Oriented Major Project in the institute.

We extend our heartfelt thanks to **Dr. Thanveer Jahan, Head of the Department of CSE (AI&ML)**, Vaagdevi College of Engineering for providing us necessary infrastructure and there by giving us freedom to carry out the Industry Oriented Major Project.

We express heartfelt thanks to the guide, **Salim Amirali Jiwani, Associate Professor**, Department of CSE (AI&ML) for his constant support and giving necessary guidance for completion of this Industry Oriented Minor Degree Project.

We are also thankful to **Mrs. G Vijayalaxmi**, **Assistant professor, Mini Project Coordinator,** for their valuable suggestions, encouragement and motivations for completing this project successfully.

We are thankful to all other faculty members for their encouragement. We convey our heartfelt thanks to the lab staff for allowing me to use the required equipment whenever needed.

Finally, we would like to take this opportunity to thank our family for their support through the work. We sincerely acknowledge and thank all those who gave directly or indirectly their support in completion of this work.

TEAM MEMBERS

D.HARSHITHA	**20641A6658**
D.SHRAVYA	**20641A6607**
K.VISHWAS	**20641A6611**
CH.SIDHARTHA	**20641A6655**

DECLARATION

We here by declare that this project entitled **"FLOOD PREDICTION USING MACHINE LEARNING"** is submitted in partial fulfillment of requirements for the award of bachelor of technology at **VAAGDEVI COLLEGE OF ENGINEERING** affiliated to Jawaharlal Nehru Technology University. The report has not been submitted either in part or full for degree earlier to this University.

TEAM MEMBERS

D.HARSHITHA	**20641A6658**
D.SHRAVYA	**20641A6607**
K.VISHWAS	**20641A6611**
CH.SIDHARTHA	**20641A6655**

ABSTRACT

Floods are among the most destructive natural disasters, which are highly complex to model. The research on the advancement of flood prediction models contributed to risk reduction, policy suggestion, minimization of the loss of human life, and reduction the property damage associated with floods. To mimic the complex mathematical expressions of physical processes of floods, during the past two decades, machine learning (ML) methods contributed highly in the advancement of prediction systems providing better performance and cost-effective solutions. Due to the vast benefits and potential of ML, its popularity dramatically increased among hydrologists. Researchers through introducing novel ML methods and hybridizing of the existing ones aim at discovering more accurate and efficient prediction models. The main contribution of this paper is to demonstrate the state of the art of ML models in flood prediction and to give insight into the most suitable models. In this paper, the literature where ML models were benchmarked through a qualitative analysis of robustness, accuracy, effectiveness, and speed are particularly investigated to provide an extensive overview on the various ML algorithms used in the field. The performance comparison of ML models presents an in-depth understanding of the different techniques within the framework of a comprehensive evaluation and discussion. As a result, this paper introduces the most promising prediction methods for both long-term and short-term floods. Furthermore, the major trends in improving the quality of the flood prediction models are investigated. Among them, hybridization, data decomposition, algorithm ensemble, and model optimization are reported as the most effective strategies for the improvement of ML methods. This survey can be used as a guideline for hydrologists as well as climate scientists in choosing the proper ML method according to the prediction task.

INDEX

CHAPTER-01

INTRODUCTION

Floods are among the most destructive natural disasters and it causes lots of damage to property and human life. The yearly data shows that the amount of rainfall is increasing and it's due to climate change. Flood is predicted in several locations using some advanced technologies which just helps the people to be prepared for upcoming disasters. It is very difficult to create a predictive model using machine learning. Machine learning gives computers the capability to learn without being explicitly programmed. Machine learning has a role in preventing many natural disasters like earthquakes, floods and many more. Machine learning make decisions using past data and these data are fed into the algorithms and the output is predicted. Machine learning (ML) can be classified into three categories Supervised learning, Unsupervised learning and Reinforcement learning. Supervised learning is the types of machine learning in which machines are trained using well "labelled" training data, and on the basis of that data, machines predict the output. The labelled data means some input data is already tagged with the correct output. The Supervised learning can be again further divided into two types Regression and Classification. Regression algorithms are used if there is a relationship between the input variable and output variable. It is used for the prediction of continuous variables. Classification algorithms are used when the output variable is categorical which means there are two classes such as Yes-No, Male-Female, True-False. Unsupervised learning is a machine learning technique in which models are not supervised using training dataset. Instead models itself find the hidden patterns and insights from the given data. The Unsupervised learning algorithm can be further categorized into two types: Clustering and Association. Clustering is a method of grouping objects into clusters such that objects with most similarities remain into a group and have less or no similarities with the objects of another group. Association rule is an unsupervised learning method which is used for finding the relationships between variables in the large database. The aim of this project is to develop a flood prediction model which is real time. This could be helpful in the areas where the flash flood occurs. This system takes the input as the rainfall data all over the india process it using different machine learning models and the best model is determined with the help of accuracy of different algorithms, which would help people prior, save lives and also save lots of meteorological efforts.

1.1 EXISTING SOLUTIONS

Flood forecasting is highly complicated and expensive. The weather and rainfall is a factor of predicting the flood. The advanced technology uses simulations supported by physics and differential equations. The satellite images are used to get the rainfall data. In recent times rapid urbanization, global climate change and extreme rainfall have resulted in flash floods. In orthodox methods of flood forecasting, using satellite images and radar also involving mathematical equations, current weather conditions are detected. At present machine learning technologies are implemented to detect such kinds of natural disasters. The floods are predicted by considering the parameters causing the flash flood. There are some drawbacks in machine learning that lead to wrong predictions of floods. The results cannot be accurate in predicting flash floods.

1.2 PROPOSED SOLUTIONS

The aim of this project is to get all the rainfall data of India and from a dataset containing yearly rainfall data. By providing real time input to different models of machine learning, those are Logistic Regression, Support Vector Machine, K-Nearest Neighbours and Decision Tree Classifier. The input provided to models are pre-processed and patterns are extracted by getting maximum accuracy. The data provided is split into a Training set and Test set. It is split in the ratio of 7:3. The all four models are used to predict and by comparing all the results of model and considering the confusion matrix of all the models the accuracy is determined. The best model is chosen by comparing the accuracy of each model.

1.3 SOFTWARE REQUIREMENTS

- Windows 10, 11 OS
- Jupyter nootbook (anaconda 3)

1.4 HARDWARE REQUIREMENTS

- Intel dual core
- 512 MB Ram
- Mouse
- Hard disk 80 GB
- Processor-i5
- Key board Standard

CHAPTER-02

RELATED WORK/ LITERATURE SURVEY

CHAPTER – 03

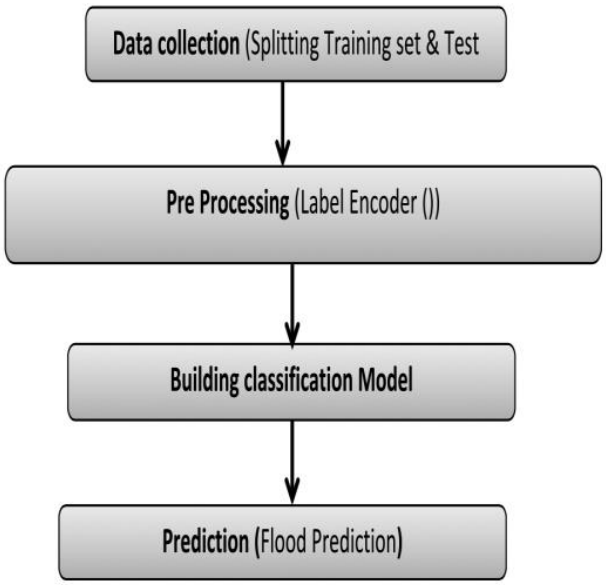
DESIGN OF PROJECT

3.1 FLOW CHART

The prediction accuracy of the different models is evaluated using data validation, and the results are compared to get accuracy. The accuracy of the training dataset, accuracy of the testing dataset, false-positive rate, specification, precision, and recall are calculated by comparing algorithms using python code.

The steps involved are:

- Define a problem
- Preparing data
- Evaluating algorithms
- Predicting results



The goal of cleaning data is to detect and remove errors. The process of transforming data prior to feeding it to the algorithm is called data pre-processing. For better results such a process takes place where raw data is converted to clean data and fed to an algorithm. Some of the Machine Learning models need data in some format, random forest algorithm does not support null values. Therefore, to execute random forest algorithm the raw data has to be transformed into null free data set.

3.2 SALGORITHMS AND TECHNIQUES

Logistic Regression

Logistic Regression may be a machine learning algorithm that predicts the probability of a categorical variable. It is a statistical way of analyzing a group of knowledge that comprises quite one experimental variable that determines the result. The outcome is then measured with a dichotomous variable. The goal of this algorithm is to seek out the simplest model to explain the connection between a dichotomous characteristic of interest and a group of independent variables. In this algorithm, the dependent variable is a binary variable that contains data coded as 1 or 0. In other words, the logistic regression model predicts P(Y=1) as a function of X.

Support Vector Machines

SVM uses a classifier that categorizes the info set by setting an optimal hyperplane between data. This classifier is chosen as it is incredibly versatile in the number of different kernel functions that can be applied, and this model can yield a high predictability rate. Support Vector Machine is one among the foremost popular and widely used clustering algorithms. It belongs to a gaggle of generalized linear classifiers and is taken into account as an extension of the perceptron. It was developed in the 1990s and continues to be the desired method for a high-performance algorithm with a little tuning.

K-Nearest Neighbor (KNN)

K-Nearest Neighbor is one among the supervised machine learning algorithms that stores all instances like training data points in an n-dimensional space. For real-valued data, the algorithm returns the mean of Nearest Neighbours, and in case of receiving unknown discrete data, it analyses the closest k number of instances that is saved and returns the most common class as the result of the prediction. In the distance-weighted Nearest Neighbor algorithm, the contribution of each of the K neighbours is weighed according to their distance, giving higher weight to the closest neighbours. The K-Nearest Neighbour algorithm is a classification algorithm and is robust to noisy data as it averages the K-Nearest Neighbours. The algorithm first takes a bunch of labelled points and analyses them to find out the way to label the opposite points. Hence, to label a new point, it looks at the closest labelled points to that new

point and has those neighbours vote, so whichever label most of the neighbours have is the label for the new point. This algorithm makes predictions about the validation set using the whole training set. Only by rummaging through the whole training set to seek out the closest instances, the new instance is predicted. Closeness is a value that is determined using a proximity measurement across all the features involved.

Decision Tree

A Decision Tree algorithm is a non-parametric supervised learning algorithm used for both classification and regression tasks**.** It has a hierarchical tree structure consisting of a root node, branches, internal nodes, and leaf nodes. Algorithms for constructing decision trees usually work top-down, by choosing a variable at each step that best splits the set of items. Different algorithms use different metrics for measuring "best", generally measuring the homogeneity of the target variable within the subsets .Decision trees are powerful tools for classifying data and weighing costs, risks, and potential benefits of ideas. They present alternatives in an easily interpretable format, making them useful in an array of environments. Decision trees are constructed via an algorithmic process that identifies ways to split, classify, and visualize a dataset based on different conditions

CHAPTER-04

IMPLIMENTATIONS

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
df = pd.read_csv('kerala.csv')
df.head()
df.isnull()
print(df.shape)
df.describe()
df['FLOODS'].replace(['YES','NO'],[1,0],inplace=True)
df.drop('SUBDIVISION',axis = 1,inplace=True)
df.head()
x = df.iloc[:,:-1]
y = df.iloc[:,-1]
x.head()
y.head()
AvgIndex = df[['JUN','JUL','AUG','SEP']]
AvgIndex.hist()
plt.show()
hm = sns.heatmap(data =  df[['JUN','JUL','AUG','SEP']])
plt.show()
from sklearn import preprocessing
minmax = preprocessing.MinMaxScaler(feature_range=(0,1))
minmax.fit_transform(x)
from sklearn.model_selection import train_test_split
```

```
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.3,random_state=1)
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
def mymodel(model):
    model.fit(xtrain, ytrain)
    ypred = model.predict(xtest)

    train = model.score(xtrain, ytrain)
    test = model.score(xtest, ytest)

    print(f"Training Accuracy : {train}\nTesting Accuracy : {test}\n\n")
    print(classification_report(ytest, ypred))

    return model
knn = mymodel(KNeighborsClassifier())
logreg = mymodel(LogisticRegression())
svm = mymodel(SVC())
dt = mymodel(DecisionTreeClassifier())
rn=mymodel(RandomForestClassifier())
from sklearn.model_selection import cross_val_score
knn_accuracy = cross_val_score(knn,xtest,ytest,cv=3,scoring='accuracy',n_jobs=-1)
knn_accuracy.mean()
logreg_accuracy = cross_val_score(logreg,xtest,ytest,cv=3,scoring='accuracy',n_jobs=-1)
logreg_accuracy.mean()
svm_accuracy = cross_val_score(svm,xtest,ytest,cv=3,scoring='accuracy',n_jobs=-1)
```

```
svm_accuracy.mean()

dt_accuracy = cross_val_score(dt,xtest,ytest,cv=3,scoring='accuracy',n_jobs=-1)

dt_accuracy.mean()

rn_accuracy = cross_val_score(dt,xtest,ytest,cv=3,scoring='accuracy',n_jobs=-1)

rn_accuracy.mean()

names = ['KNN','LogReg','SVM','DecisionTree','RandomForest']

score
=[knn_accuracy.mean(),logreg_accuracy.mean(),svm_accuracy.mean(),dt_accuracy.mean(),r
n_accuracy.mean()]

scores = pd.DataFrame({'Algorithm Name':names,'Score':score})

axis = sns.barplot(x='Algorithm Name',y='Score',data = scores)

axis.set(xlabel='Classifier', ylabel='Accuracy')


for p in axis.patches:

    height = p.get_height()

    axis.text(p.get_x() + p.get_width()/2, height + 0.005, '{:1.4f}'.format(height), ha="center")

    scores # Cross-validation

#2nd Models

##HyperParameter Tuning for decsion Tree
dt = mymodel (DecisionTreeClassifier ()) # Current Accuracy

parameters = {

    "criterion":["gini", "entropy"],

    "max_depth": list(range(1,50, 5)),

    "min_samples_leaf": list(range(1, 50, 5))

}

from sklearn.model_selection import GridSearchCV

grid = GridSearchCV(DecisionTreeClassifier(), parameters, verbose=2)

grid.fit(xtrain, ytrain)
```

```
grid.best_params_
grid.best_score_
grid.best_estimator_
dt2 = mymodel(grid.best_estimator_) # Post- HyperParameter Tuning For Decision

##HyperParameter Tuning For SVM

parameters = {

    'C': [0.1, 1, 10, 100, 1000],

    'gamma': [1, 0.1, 0.01, 0.001, 0.0001],

    'kernel': ['rbf']

}

from sklearn.model_selection import GridSearchCV

grid = GridSearchCV(SVC(), parameters, verbose=3)

grid.fit(xtrain, ytrain)
grid.best_params_
grid.best_score_
grid.best_estimator_
svm = mymodel(grid.best_estimator_)

parameters = {

    'penalty' : ['l1','l2'],

    'C'       : np.logspace(-3,3,7),

    'solver'  : ['newton-cg', 'lbfgs', 'liblinear'],

}

from sklearn.model_selection import GridSearchCV

grid = GridSearchCV(LogisticRegression(), parameters, verbose=2)

grid.fit(xtrain, ytrain)
grid.best_params_
grid.best_score_
grid.best_estimator_
logreg = mymodel(grid.best_estimator_)
```

```
##HyperParameter Tuning For KNN
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler

pipe = Pipeline(
    [
        ("sc", StandardScaler()),
        ("knn", KNeighborsClassifier())
    ]
)
from sklearn.model_selection import GridSearchCV
parameters = [{"knn__n_neighbors": [3, 5, 7, 9],
               "knn__weights": ["uniform", "distance"],
               "knn__leaf_size": [15, 20]}]
grid = GridSearchCV(pipe, parameters, cv=5, scoring="accuracy")
grid.fit(xtrain, ytrain)
grid.best_params_
grid.best_score_
grid.best_estimator_
knn2 = mymodel(grid.best_estimator_)

param_grid = {
    'n_estimators': [25, 50, 100, 150],
    'max_features': ['sqrt', 'log2', None],
    'max_depth': [3, 6, 9],
    'max_leaf_nodes': [3, 6, 9],
}
```

```
grid_search = GridSearchCV(RandomForestClassifier(),
                           param_grid=param_grid)
grid_search.fit(xtrain, ytrain)
print(grid_search.best_estimator_)
model_grid = RandomForestClassifier(max_depth=9,
                                    max_features="log2",
                                    max_leaf_nodes=9,
                                    n_estimators=25)
model_grid.fit(xtrain, ytrain)
y_pred_grid = model_grid.predict(xtest)
print(classification_report(y_pred_grid, ytest))
xtrain, xtest, ytrain, ytest = train_test_split(x,y, test_size=0.3, random_state=1, stratify=y)
knn_accuracy = cross_val_score(knn,xtest,ytest,cv=3,scoring='accuracy')
knn_accuracy.mean()
logreg_accuracy = cross_val_score(logreg,xtest,ytest,cv=3,scoring='accuracy',n_jobs=-1)
logreg_accuracy.mean()
svm_accuracy = cross_val_score(svm,xtest,ytest,cv=3,scoring='accuracy',n_jobs=-1)
svm_accuracy.mean()
dt_accuracy = cross_val_score(dt,xtest,ytest,cv=3,scoring='accuracy',n_jobs=-1)
dt_accuracy.mean()
rn_accuracy = cross_val_score(rn,xtest,ytest,cv=3,scoring='accuracy')
rn_accuracy.mean()
names = ['KNN','LogReg','SVM','DecisionTree','RandomForest']
score
=[knn_accuracy.mean(),logreg_accuracy.mean(),svm_accuracy.mean(),dt_accuracy.mean(),r
n_accuracy.mean()]
scores_2nd = pd.DataFrame({'Algorithm Name':names,'Score':score})
axis = sns.barplot(x='Algorithm Name',y='Score',data = scores_2nd)
axis.set(xlabel='Classifier', ylabel='Accuracy')
```

```
for p in axis.patches:
    height = p.get_height()
    axis.text(p.get_x() + p.get_width()/2, height + 0.005, '{:1.4f}'.format(height), ha="center")
scores_2nd # Cross-validation post HyperParameter Tuning
scores # Cross-validation for the Base model
Cross-validation post HyperParameter Tuning
scores_2nd['Score'].max()
Cross-validation for the Base model
scores['Score'].max()
```

CHAPTER-05

APPLICATIONS

CHAPTER-06

EVALUATION

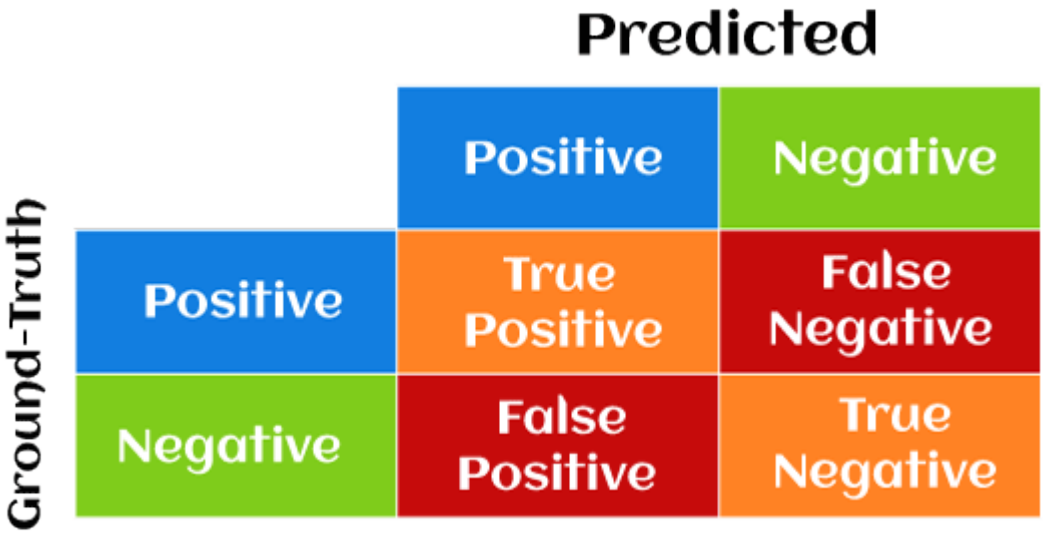
Performance Analysis Metrics

True Positive: It is an outcome where the model correctly predicts the positive class. The outcome is considered as true positive when the system can correctly predict that an incident has indeed occurred.

True Negative: It is an outcome where the model correctly predicts the negative class. The outcome is considered as true negative when the system can correctly predict that the particular incident has not occurred.

False Positive: False Positive is an accuracy measure where the model mis-predicts the positive class. The outcome is considered as False Positive when the system cannot correctly predict that the particular incident has occurred.

False Negative: False Negative is an accuracy value where the model mis-predicts the negative class. The outcome is considered as False Negative when the system cannot correctly predict that the particular incident has not occurred.



Sensitivity: Sensitivity is a measure of the proportion of true positive values, that is, the actual number of positive cases that are correctly predicted as positive. It is also known as Recall value. There exists another proportion of actual positive cases that are mis-predicted, which can be represented in the form of a false negative rate. Therefore, the sum of sensitivity and false-negative rate value is 1.

Mathematically sensitivity can be calculated as:

Sensitivity/Recall = (True Positive) / (True Positive + False Negative)

The higher value of sensitivity would mean a higher value of the true positive and lower value of false negative. The lower value of sensitivity would mean a lower value of the true positive and higher value of false negative.

Precision: The proportion of positive predictions that are actually correct.

Precision = TP / (TP + FP)

Precision is calculated by dividing the number of correctly predicted positive observations by the total number of predicted positive observations. High precision relates to the low false-positive rate.

Recall: The proportion of positive observed values correctly predicted. (The proportion of actual defaulters that the model correctly predicts)

Recall = TP / (TP + FN)

Recall (Sensitivity) – Recall is calculated by dividing the number of correctly predicted observations to the total number of observations in an actual class.

F1 Score: F1 Score is defined as the weighted average of Precision and Recall values. Hence both false negative and false positives values are taken into account. When there's an uneven class distribution, the F1 Score value is generally more useful when compared to Accuracy value. On the other hand, Accuracy value worksheets best when the values of false positive and false negatives have a similar cost. If both values are different, then Precision and Recall values are taken into account.

GENERAL FORMULA:

F- Measure = 2TP / (2TP + FP + FN)

F1-SCORE FORMULA ;

F1 Score = 2*(Recall * Precision) / (Recall + Precision)

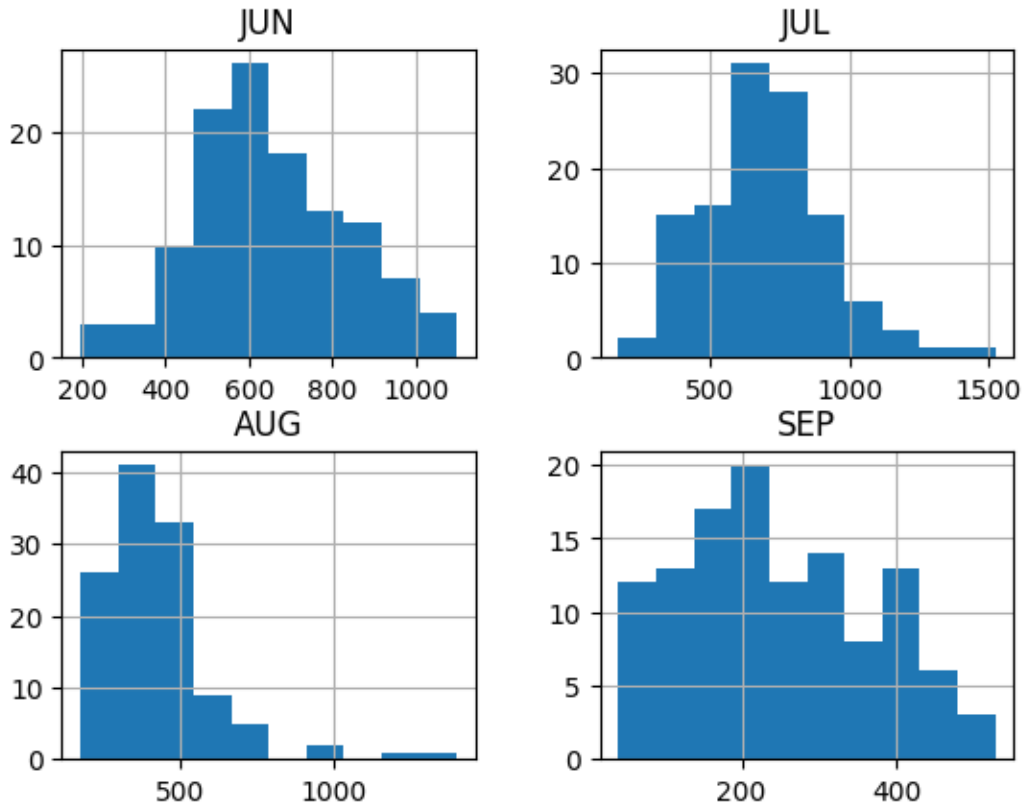
CHAPTER – 07

TESTING

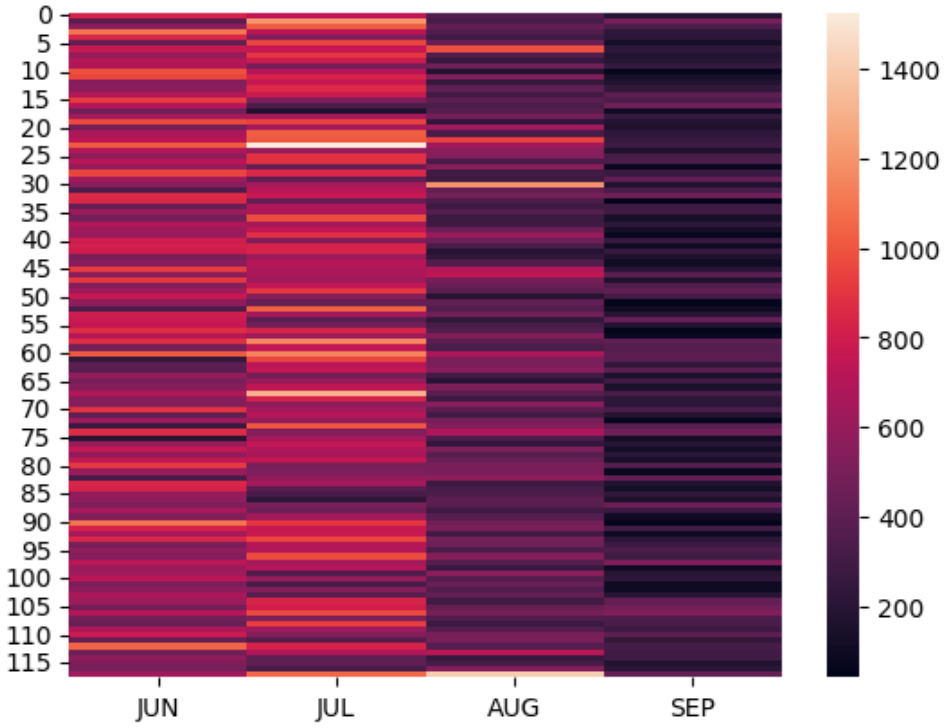
CHAPTER-08

RESULTS

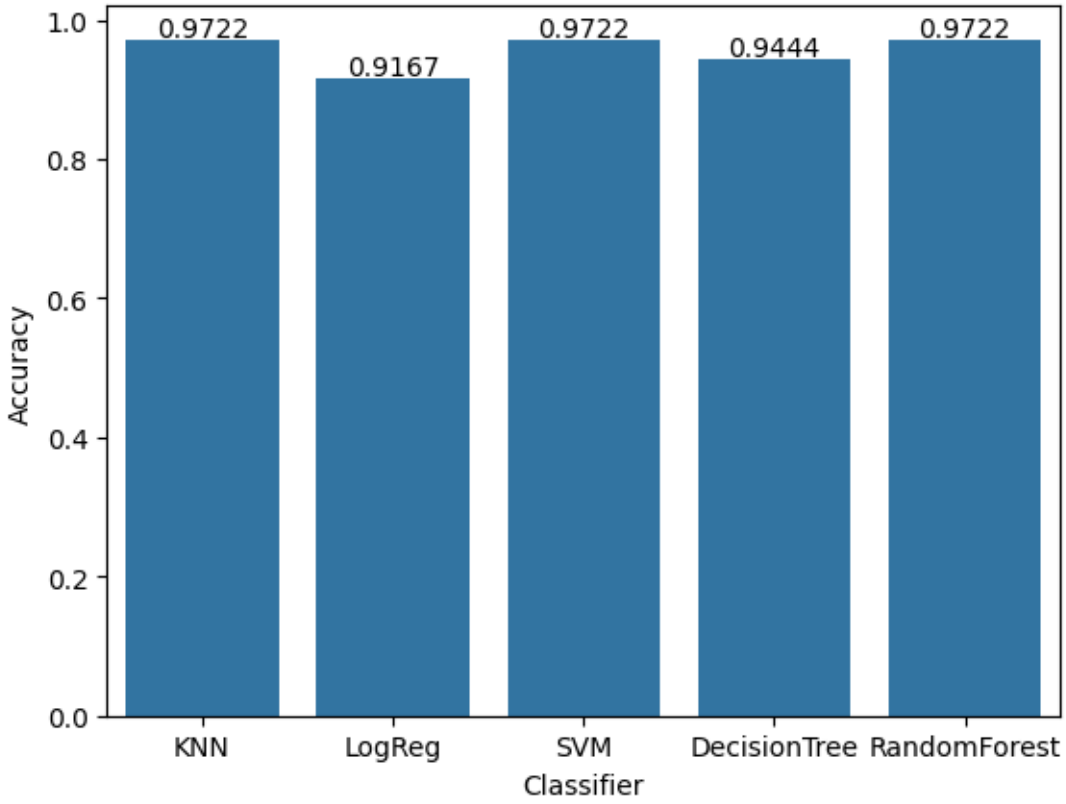
A histogram graph which shows about the floods during rainy season



Heat map displaying the effect in rainy season



Graph displaying the accuracies of each algorithm used in this ML model



Cross-validation post HyperParameter Tuning

	Algorithm Name	Score
0	KNN	0.972222
1	LogReg	0.916667
2	SVM	0.972222
3	DecisionTree	0.944444
4	RandomForest	0.972222

Cross-validation for the Base model

	Algorithm Name	Score
0	KNN	0.916667
1	LogReg	0.944444
2	SVM	0.916667
3	DecisionTree	0.944444
4	RandomForest	0.944444

Cross-validation post HyperParameter Tuning

```
0.9722222222222222
```

Cross-validation for the Base model

```
0.9444444444444445
```

CHAPTER-09

CONCLUSION AND FUTURE SCOPE

9.1 CONCLUSION

This analysis aims to select a prediction model for rainfall forecasting. Since Kerala contributes a high amount of agricultural products in India, maximizing the outputs and minimizing the risks of confronting natural disasters are the concern of this analysis. We can see that the baseline model gave us an Accuracy of 94%, not disappointing but after tuning the model with the hyperparameters we were able to achieve 97% Accuracy over out. In conclusion, flood prediction using machine learning is a promising strategy that can aid in flood mitigation by offering early warning systems and assisting in the development of better decisions. The architecture for machine learning-based flood prediction entails gathering and preprocessing pertinent data, extracting, and engineering features, choosing, and training an appropriate machine learning algorithm, assessing its performance, deploying it in an appropriate environment, and maintaining it over time. With this architecture, we can create flood prediction models that are precise and trustworthy and that can be utilized to save lives, safeguard property, and lessen the impact of floods on our communities. But it is necessary to remember that machine learning models are only as good as the data they're trained on, therefore it's essential to guarantee the accuracy and dependability of the data used to create such models. In general, machine learning-based flood prediction has the potential to significantly improve our capacity to control and lessen the effects of floods.

9.2 FUTURE WORK

This model is currently analysing the data which was taken from a predefined data set and performing predictions using the same dataset. The architecture for machine learning-based flood prediction can be improved in a number of ways in the future. Real-time data integration like usage of IOT devices for data collection, ensemble modelling, uncertainty estimation methods, enhancing interpretability, taking into account social and economic aspects, and scaling the model to cover bigger areas are a few of these. These improvements can assist ensure the relevance and effectiveness of flood prediction models in the face of shifting climatic and environmental conditions. They can also help increase the accuracy, dependability, and usefulness of flood prediction models.

ADVANTAGES AND DISADVANTAGES

ADVANTAGES :

- Improved accuracy: By using advanced machine learning algorithms and incorporating multiple data sources, proposed systems can provide more accurate flood predictions than existing system.
- Increased coverage: Proposed systems can expand the geographic coverage of flood prediction, allowing more areas to benefit from the system's predictions.
- Enhanced efficiency: By using parallel processing or cloud computing, proposed systems can speed up the training and operation of the system, making it more efficient.
- Improved interpretability: Proposed systems can incorporate explainable AI techniques to improve the interpretability and transparency of the system, which can increase trust and adoption of the system.

DISADVANTAGES :

Although This flood prediction model offer several benefits, it does have some drawbacks that include difficulties with data quality, coverage gaps, expensive computations and unpredictability of results. Since the incomplete or huge amount of data might harm this model’s accuracy, quality of data is essential and certain methods can only work well in selected geographic areas where data is accessible, which limits the capability in predicting floods in locations with sparse data coverage. Machine learning can be expensive, which is limited due to low funding. Basically, these are some of the drawbacks of our flood prediction model.

CHAPTER-10

BIBLIOGRAPHY

- An Innovative Flood Prediction System Using Improved Machine Learning Approach (csfjournal.com) https://encyclopedia.pub/73
- PyQt5 Reference Guide — PyQt v5.15 Reference Guide (riverbankcomputing.com)
- Requests: HTTP for Humans™ — Requests 2.25.1 documentation (python-requests.org)
- Tutorials — Matplotlib 3.4.2 documentation
- https://www.irjet.net/archives/V7/i5/IRJETV7I51189.pdf
- Escarameia M, Karanxha A, Tagg A. 2007 Quantifying the flood resilience properties of walls in typical UK dwellings. Build. Serv. Eng. Res. Technol. 28, 249–263. (doi:10.1177/0143624407079093) Handmer J, Proudley B. 2007 Communicating
- uncertainty via probabilities: the case of weather forecasts.Environ. Hazards7, 79–87. (doi:10.1016/j. envhaz.2007.05.002)