DEVELOPMENT OF EXPLAINABLE AI (XAI) FOR HEAVY RAINFALL PREDICTION

A PROJECT REPORT

Submitted by

V V N S S RAGHU NATH

in partial fulfillment for the award of the degree of

MASTER OF TECHNOLOGY

IN

MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE

Lovely Professional University, Punjab

12415960

MARCH 2025

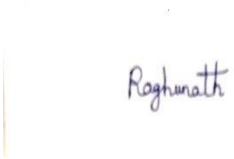
Lovely Professional University, Punjab

BONAFIDE CERTIFICATE

Certified that this project report **“ DEVELOPMENT OF EXPLAINABLE AI (XAI) FOR HEAVY RAINFALL PREDICTION ”**

is the bonafide work of **“……..V V N S S RAGHU NATH ”**

who carried out the project work under my supervision.



SIGNATURE

<<Signature of the Head of the Department>>
SIGNATURE
<<Name>>
HEAD OF THE DEPARTMENT

<<Signature of the Supervisor>>
SIGNATURE
<<Name of the Supervisor>>
SUPERVISOR

Department of Machine Learning, CSE,
Lovely Professional University, Punjab

APPENDIX III

(A typical specimen of table of contents)

<Font Style Times New Roman>

TABLE OF CONTENTS

Project Synopsis Table of Contents Details for Reference

Abstract *(Max 250 words)*

Predicting the likelihood of extreme rainfall events is crucial for disaster preparedness and mitigation efforts, as these events can cause flooding, landslides, and damage to infrastructure. This model employs machine learning and deep learning approaches to forecast heavy rainfall using weather data obtained from OpenWeather API and IMD. An “Ensemble Model, Decision Tree, Random Forest, Neural Networks, LightGBM, CatBoost, and XGBoost” are among the algorithms used to improve the accuracy of predictions. To increase its generalizability and resilience, a Voting Classifier integrates numerous models. The interpretability graphs generated by “explainable AI (XAI) methods like SHAP and LIME” provide light on the roles played by features and the reasoning behind model choices. To further enhance performance, “ensemble techniques like bagging-based Random Forest and boosted decision trees” are used. Authentication and a Flask-based UI make real-time prediction and user testing accessible. This system provides a trustworthy and understandable method for predicting instances of excessive rainfall by combining state-of-the-art ensemble techniques with explainability tools. This approach will help with early warning systems and disaster response operations.

1. **Introduction**

Floods, landslides, and other forms of extreme precipitation may wreak havoc on communities and economies, endangering lives and damaging infrastructure. In order to effectively handle disasters, plan and reduce disasters, accurate forecasts for such serious weather events are required. Complex numeric simulation is the spine in traditional weather forecasting methods, but they can be resource and calculation intensity. Although these models are capable of producing reliable projections, decision makers often appear to be prophecy to understand the underlying arguments due to lack of interpretation. A new alternative that has arisen is the use of machine learning algorithms for weather prediction, which are more effective and accurate. On the other hand, many people feel that these models are not sufficiently reliable, especially during conditions with high dinner as a weather forecast because of their "black -box" character. The purpose of this research is to solve that problem by developing a strong model to predict serious rainfall Explainable AI (XAI) techniques in combination with machine learning algorithms. The model increases the accuracy of precipitation conditions and provides insight into the underlying causes, which affects these predictions by adding XAI methods such as size and lime. Preparations for better destruction and reaction can be achieved through increased confidence in the production of the model and can improve the high possible decision on this approach of openness and accountability.

2. **Problem Statement**

Traditional numerical weather models have difficulty predicting complex and variable heavy rainfall, which is important for destruction control. Machine learning (ML) is not widely used, although it improves accuracy, it is not always easy to understand. An XAI-based model that effectively predicts weather while maintaining the openness is the goal of this work.

3. **Objectives**

- Make a model that uses machine learning when there will be serious rain to predict.
- Increase the model interpretation by incorporating the XAI approaches.
- Determine the effect of the model by calculating your F1 score, accuracy and accuracy.
- Provide initial warning systems the ability to come up with predictions in real time.

4. **Literature Review**

The capacity for machine learning for weather forecasts has been the subject of many studies. The worldwide forecasting system (GFS) and other traditional NWP models use time-consuming and labor-intensive data simulation to predict weather. Although these models are accurate, it can be difficult to understand and use them a lot of data processing resources. Better predicted accuracy is shown in methods based on machine learning, especially deep learning techniques. To solve the "Black-Box" problem in AI interest weather message, Arita et al. (2020) found that clear AI was important. Machine learning forecasts are explained more in XAI methods such as Shap (Shapley Additive Explanation) and Lime (local explanatory model-designer explanation), which improve their openness. This makes them more useful for meteorologists and decision makers.

Because of their ability to understand complex patterns in weather figures, a contingent of artists such as random forests, XGBOOST, and Catboost has found widespread applications in precipitation. Research by Sahu et al. (2020) and McGawn et al. (2019) has shown that the use of clothing learning to combine different models increases the accuracy of prediction by reducing the dangers of overfitting.

In addition, meteorological datasets with non-numerical properties can have a great advantage from the cat bosts, which is a technique that promotes an effective shield for the treatment of ranked properties described by Prokhornakova et al. (2018). In addition, Lundberg & Lee (2017) proved that the size can uncover the importance of value properties, which in turn increases confidence in the model of the weather with artificial intelligence.

Despite these reforms, the problems persist. The main obstacles to wide use of weather prediction models based on machine learning include high calculation costs, large and high quality data sets and real -time treatment restrictions. Prediction in practical surroundings can be improved by dealing with these problems with the trustees and practical conditions of the algorithm to explain these problems.

5. **Methodology**

- **5.1 EXISTING SYSTEM:**

Traditional meteorological forecasting techniques and independent machine learning models are the spine of the current approach to predict excessive precipitation phenomena. These methods use algorithms such as "XGBoost, LightGBM, Neural Networks, Decision Tree, Random Forest, and Logistic Regression" for meteorological data derived from sources such as Openweeder API and IMD. But these standalone models do not always do a good job of explaining or giving us understandable insight into our forecasts. Traditional methods for prediction of weather depend on the model for numerical weather, which may not be flexible enough to change the climate. Making decisions is not as transparent as it can be because most current models do not use Explainable AI (XAI) function. Without learning outfits, the future performance is even more limited because individual models may have problems with variety and bias. In the absence of integrated user interfaces, accessibility and real -time utility are greatly interrupted. To remove these obstacles, better analysis and prediction require a condition -art system; This system should wear dress, xai-based interpretation and a spontaneous user interface.

- **5.1.1 DISADVANTAGES OF EXISTING SYSTEM:**
  - The current layout depends on the standalone machine learning model, which often leads to problems with generalization, resulting in minimum and skilled forecasts for severe rainfall.
  - Users have a difficult time to understand the reasoning in the predictions, and when using traditional models, the system has less faith because they do not use an Explainable AI (XAI) approach.
  - Clothing approach, which can improve predicting accuracy by combining the strength of many models, is not used by the system.
  - Traditional weather models are not enough dynamic to adjust the changing climate, which limits their benefit in situations where the weather turns to real time.
  - Users have difficult times of using and using the system well for testing and forecasts due to a lack of an integrated user interface for real -time engagement.
- **5.2 PROPOSED SYSTEM:**

By using weather data from Openweather API and IMD, the proposed system planned to use machine learning and deep teaching techniques to predict powerful rainfall events. To improve the accuracy and flexibility, it connects an artist related model and a voting eligible with several forecast models such as "XGBoost, Decision Tree, Random Forest, Neural Networks, LightGBM, and Logistic Regression". To make the model's decisions more transparent, Explainable AI (XAI) methods such as "SHAP and LIME". "Boosted Decision Tree and Bagging-Based Random Forest" are two examples of clothing strategies used for better analysis that further promotes performance. In order to facilitate real -time commitment, we create a certification created, user -friendly flask -based interface that allows users to see tests and predictions. Help with disaster management and preparation, this system provides a reliable structure to study weather patterns and to predict excessive rainfall by combining lecture tools and learning advanced dress with a scalable front end.

- **5.2.1 ADVANTAGES OF PROPOSED SYSTEM:**
  - Prediction accuracy and improvement of flexibility, proposed approaches include dress learning, which includes voting and promoting decision trees.
  - Model decisions and value of functions can be considered better by means of XAI approaches such as lime and shape provided by lime and shape.
  - By being beneficial for the changing weather conditions, the clothing method increases the performance of the system in real time.
  - With a rinsed-based front and user authentication, the system becomes more accessible and attractive, and real-time predictions are made possible on a spontaneous platform.
  - Different types of powerful algorithms, including “XGBoost, Neural Networks, and Logistic Regression” are included in the system to provide a flexible and reliable method of precipitation assumption.

- **Exploring the Dataset :**
  - The distribution and structure of the dataset are examined by consulting Openweeder API and IMD. These functions have analysis of statistical properties, detects Outlair and checking missing numbers. Description of data quality and necessary preparatory procedures can be improved with descriptive analysis.

- **ALGORITHMS:**

- **Logistic Regression**

To predict the possibility of serious rainfall events, logistic regression is a direct and effective classification approach. It predicts when there will be heavy rain by following the relationship between many seasonal variables and the opportunity. The relative simplicity assumes its benefit as a measure that a more complex algorithm can be measured.

- **Decision Tree**

By using meteorological properties to divide data into multiple decision -making nodes, the decision -making three method forms a hierarchical structure. It does a good job of capturing non-led relationship between things such as cloud cover, wind speed and rainfall. Overwatering the data can be a single decision with a tree, so the dress methods, such as random forest, need to improve generality.

- **Random Forest**

Many decisions increase the accuracy of the forecast by mixing trees in a dress, and reduces the overfit of random forest. It improves the flexibility of the model with the average of the predictions of several trees trained on different data. The prognosis for rainfall, which requires a wide range of weather parameters at a time, is very favorable.

- **Neural Network**

Complex patterns can be learned from historical weather data from nervous networks, especially in deep learning models. By using a network of interconnected neurons, they are able to record complex interactions between meteorological variables. While this method improves pattern recognition for severe rain, it is calculating and data preparation intensely.

- **LightGBM**

Lightgbm (Light Gradient Boosting Machine) A high demstration increase, can handle large data sets with minimal training time with timely training. The fact is that it can handle incomplete data and reach, it makes it ideal for prediction of rain.

- **CatBoost**

When working with classified data, Catbost - a shield is a boosting algorithm - ideal because it does not require too much preaching. For the prognosis for severe rainfall, it performs fantastic work, as it precisely reflects the complex difference between meteorological variables. Model stability and overfitting are expanded with autonomous convenience management and regularization procedures for Cathbosts.

- **XGBoost**

Extreme gradient is an effective and effective wood -based method of increasing, or short or short. Catching small trends in meteorological data is made more effective by improving weak models using the gradient that promotes the gradient. Reduced inaccuracy in rainfall, and its regular procedures went on.

- **Voting Classifier**

An ensemble method, which votes classifies, integrates multiple models to increase predicting accuracy. Balanced and generalized prediction is achieved by combining the results of several algorithms: Decision Tree, Random Forest, LightGBM, CatBoost, and XGBoost. Early warning

systems in disaster management are compatible with this strategy as it improves reliability.

6. **Implementation Details**

1. Technologies & Tools

- **Technologies Used:** “Python, TensorFlow, Scikit-learn, Flask (for deployment)”.

2. Software and Hardware Requirements

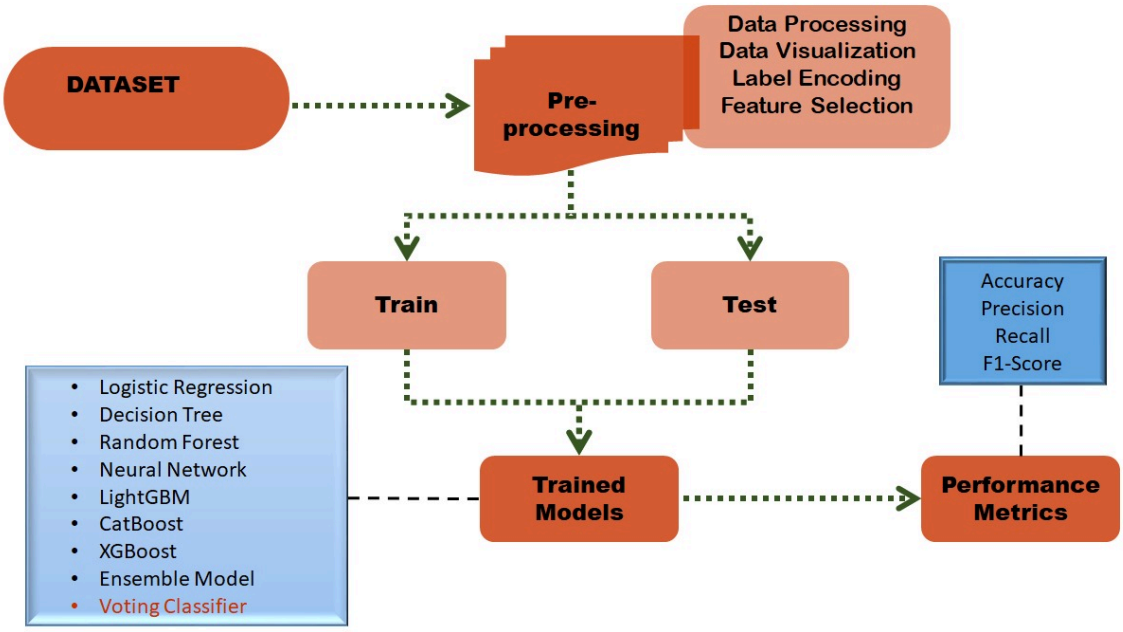
HARDWARE REQUIREMENTS:

- Operating System: Windows Only
- Processor: i5 and above
- Ram: 8 GB and above
- Hard Disk: 25 GB in local drive

SOFTWARE REQUIREMENTS:

1) Software: Anaconda
2) Primary Language: Python
3) Frontend Framework: Flask
4) Back-end Framework: Jupyter Notebook
5) Database: Sqlite3
6) Front-End Technologies: HTML, CSS, JavaScript and Bootstrap4

3. System Architecture



- **MODULES:**

  - **Importing the Packages**

In order to construct the predictive model, necessary libraries for data handling and visualization are imported, including NumPy, Pandas, Matplotlib, and Seaborn. Models are created using machine learning frameworks such as Scikit-learn, TensorFlow, and PyTorch. Model predictions can also be properly interpreted with the use of Explainable AI (XAI) tools such as SHAP and LIME.

  - **Exploring the Dataset**

In order to comprehend the distribution and structure of the dataset, which is derived from the OpenWeather API and IMD, it is studied. These functions have analysis of statistical properties, detects Outlair and checking missing numbers. Description of data quality and necessary preparatory procedures can be improved with descriptive analysis.

  - **Data Processing**

Adding the missing values, ending repetition and normalizing numerical variables is part of all data preparation. To ensure that they work with machine learning algorithms, the variables are properly coded. To improve the performance and convergence of the model, normalization or scaling is used as required as required.

  - **Data Visualization**

Histograms, scattering plots, box plots and correlations are some visualization techniques used to gain deep insight into the dataset. Better convenience choices and model understanding data have two consequences of the use of visualization for trends, correlations and outlair.

  - **Label Encoding**

When using label coding or A-warm coding, numerical representation of classified properties in the dataset is obtained, such as weather conditions. Prevention of exercise pins and increases model accuracy, this phase guarantees that machine learning models can understand and process non-broadcast data correctly.

- **Feature Selection**

Methods such as mutual information, Recursive Feature Elimination (RFE) and size values are used to select relevant functions. By reducing overfeating and promoting computing efficiency, this phase ensures that only the most important variables are used to train the model. Better model lecturer system is obtained by importance analysis.

- **Splitting the Dataset**

To test the efficiency of the model before letting the audience, the data set is divided into two parts: training and verification. To train the model and ensure that there is enough data for evaluation, it is common to use a sharing of 80-20 or 70-30. Ensuring that evaluation of fair performance and prevents data leakage is both helper from this process.

- **Building the Model for All Data (LIME)**

Many ML and DL models are trained using datasets. These models include random forests, logistic region, neural networks and attire approaches. Using lime and shape we can understand the predictions of the model, which is important for evaluating their accuracy and lecturer. Selects the best executing model for interface purposes produced using a flask.

- **User Report:** User Registration in the application is even after this module
- **User signature:** Logging in the program is possible through this module.
- **User input:** This module allows users to submit data into the application, which will then be used by the algorithms to provide predictions.
- **Final results:** Perform the finished product of the project.

- **Extension:**

In order to create better and more flexible precipitation conditions, we created using a cloth approach on this work, which combines many models. We also improved the estimated performance by using advanced dress techniques such as Bagging RF and Boosted Decision Trees. In addition, we created an easy to navigate in front with a bottle Framework, which allows secure user authentication and real -time testing.

Advantages:

1. By ending prejudice and incorporating different patterns into meteorological data sets, the combination of several models increases the dependence on predictions.
2. Overfitting can be reduced by using enchanted techniques, which guarantees that the performance is constant even when exposed to new data or separate weather conditions.
3. The friend, who is created using the flask, enables secure access to the user -friendly interactions, accurate real -time and secular authentication predictions.
4. The expandability of the system allows for new models and simple additions of data sets, guarantees its constant utility and adaptability.

4. SAMPLE CODE:

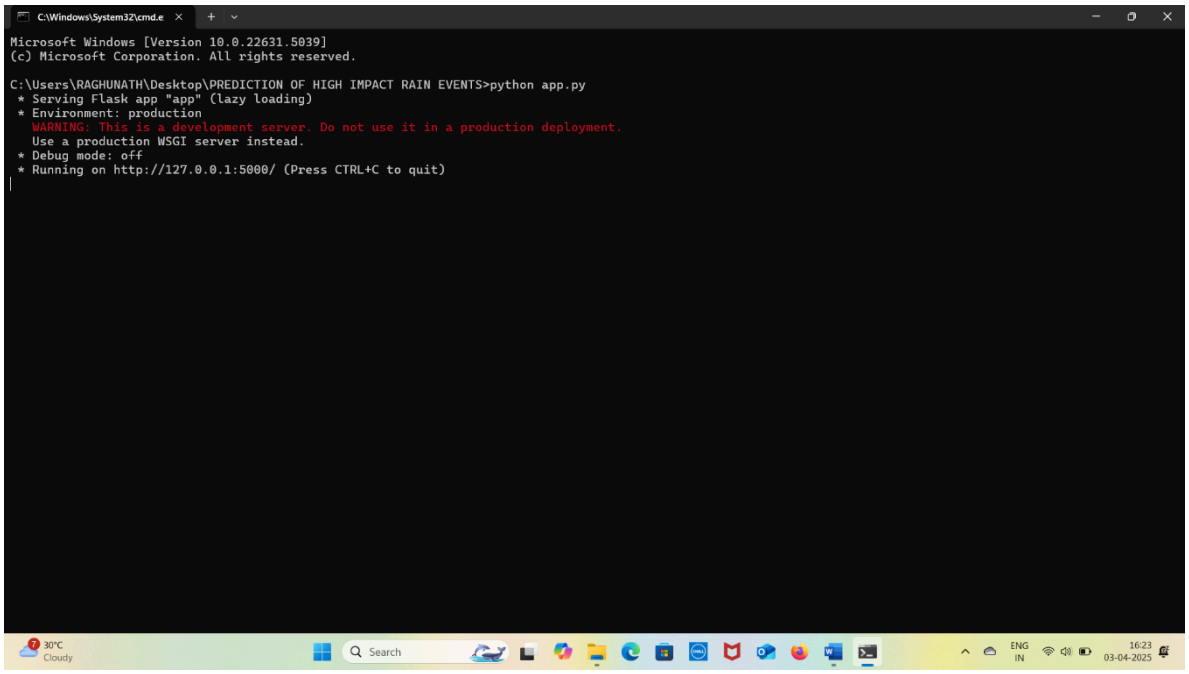
```
# Import necessary libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
```

```
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestClassifier, VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
from catboost import CatBoostClassifier
from sklearn.metrics import accuracy_score, classification_report
import lime
import lime.lime_tabular

# Load the dataset
df = pd.read_csv("rainfall_data.csv")

# Display basic dataset information
print(df.info())
print(df.describe())

# Handling missing values
df.fillna(df.mean(), inplace=True)

# Encode categorical variables
label_enc = LabelEncoder()
df['Weather_Condition'] = label_enc.fit_transform(df['Weather_Condition'])

# Define features and target variable
X = df.drop(columns=['Rainfall_Intensity'])  # Independent variables
y = df['Rainfall_Intensity']  # Dependent variable

# Split the dataset into training and validation sets (80-20 split)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Feature Scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Initialize individual models
logistic = LogisticRegression()
decision_tree = DecisionTreeClassifier()
random_forest = RandomForestClassifier(n_estimators=100, random_state=42)
xgb = XGBClassifier(use_label_encoder=False, eval_metric="logloss")
lgbm = LGBMClassifier()
catboost = CatBoostClassifier(verbose=0)

# Train models individually
models = {
    "Logistic Regression": logistic,
    "Decision Tree": decision_tree,
    "Random Forest": random_forest,
    "XGBoost": xgb,
```

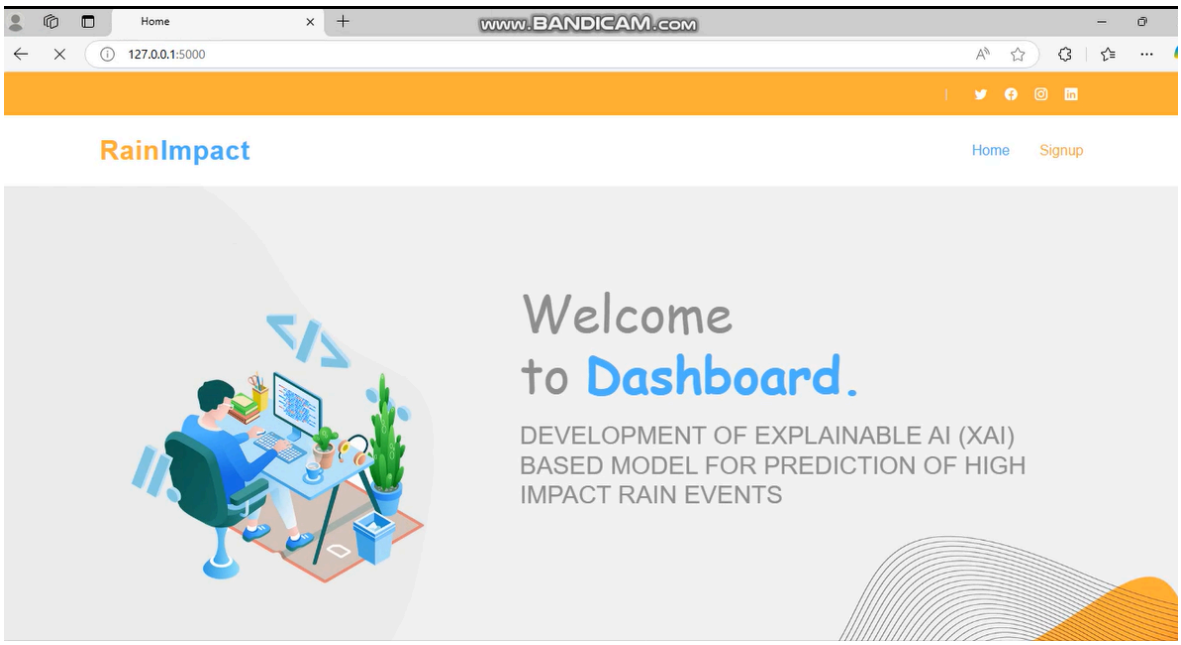
```
    "LightGBM": lgbm,
    "CatBoost": catboost
}

for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print(f"Model: {name}")
    print("Accuracy:", accuracy_score(y_test, y_pred))
    print(classification_report(y_test, y_pred))
    print("-" * 50)

# Implement Voting Classifier (Ensemble Model)
voting_clf = VotingClassifier(
    estimators=[("RF", random_forest), ("XGB", xgb), ("LGBM", lgbm), ("CatBoost", catboost)],
    voting='hard'
)

# Train the voting classifier
voting_clf.fit(X_train, y_train)
y_pred_voting = voting_clf.predict(X_test)

# Evaluate the Voting Classifier
print("Voting Classifier Accuracy:", accuracy_score(y_test, y_pred_voting))
print(classification_report(y_test, y_pred_voting))

# Apply LIME for model explainability
explainer = lime.lime_tabular.LimeTabularExplainer(
    X_train, feature_names=df.columns[:-1], class_names=['No Rain', 'Moderate Rain', 'Heavy
Rain'], mode="classification"
)

# Explain a single prediction
sample_idx = 5
exp = explainer.explain_instance(X_test[sample_idx], voting_clf.predict_proba)
exp.show_in_notebook()

# Plot Feature Importance (Example with Random Forest)
importances = random_forest.feature_importances_
features = df.columns[:-1]
plt.figure(figsize=(10, 5))
sns.barplot(x=importances, y=features)
plt.xlabel("Feature Importance")
plt.ylabel("Features")
plt.title("Random Forest Feature Importance for Rainfall Prediction")
plt.show()

# Save the trained model
import joblib
joblib.dump(voting_clf, "rainfall_prediction_model.pkl")
```

Print ("Model saved successfully!")

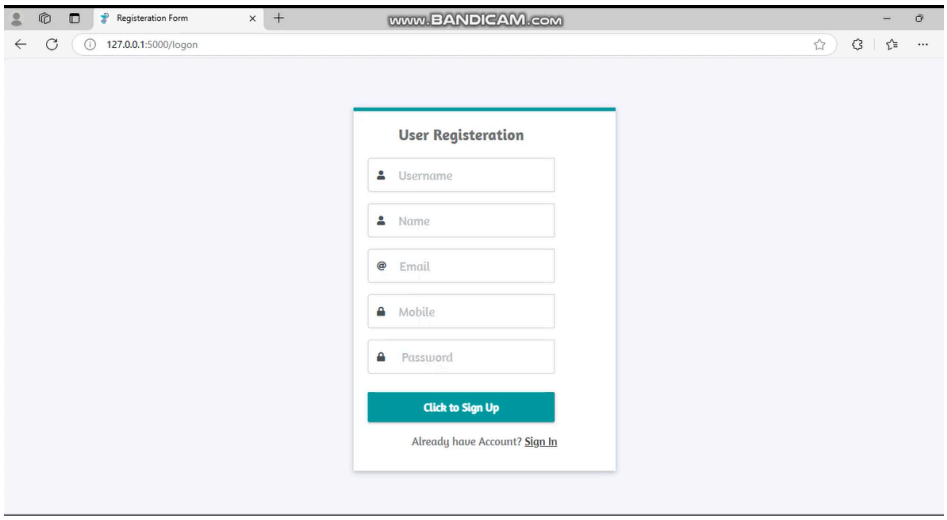
7. **Results and Discussion**



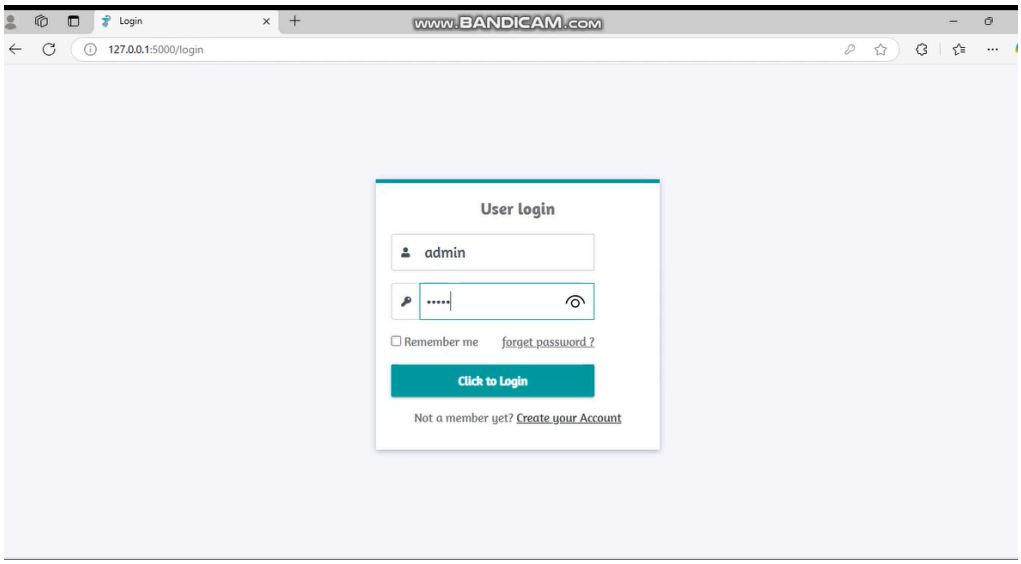
Command Prompt Screen



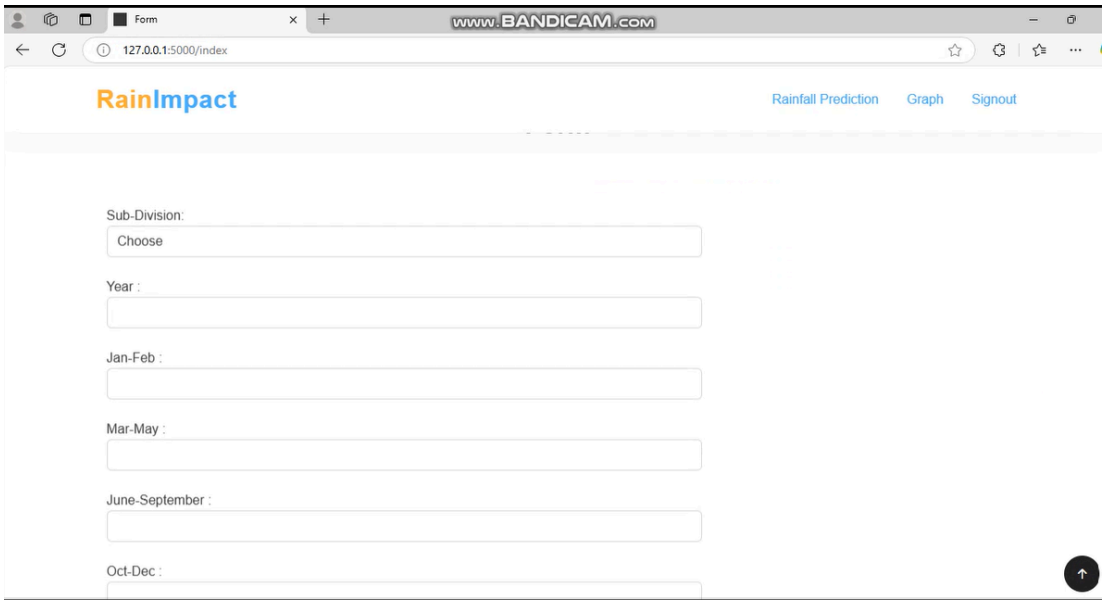
Home Page



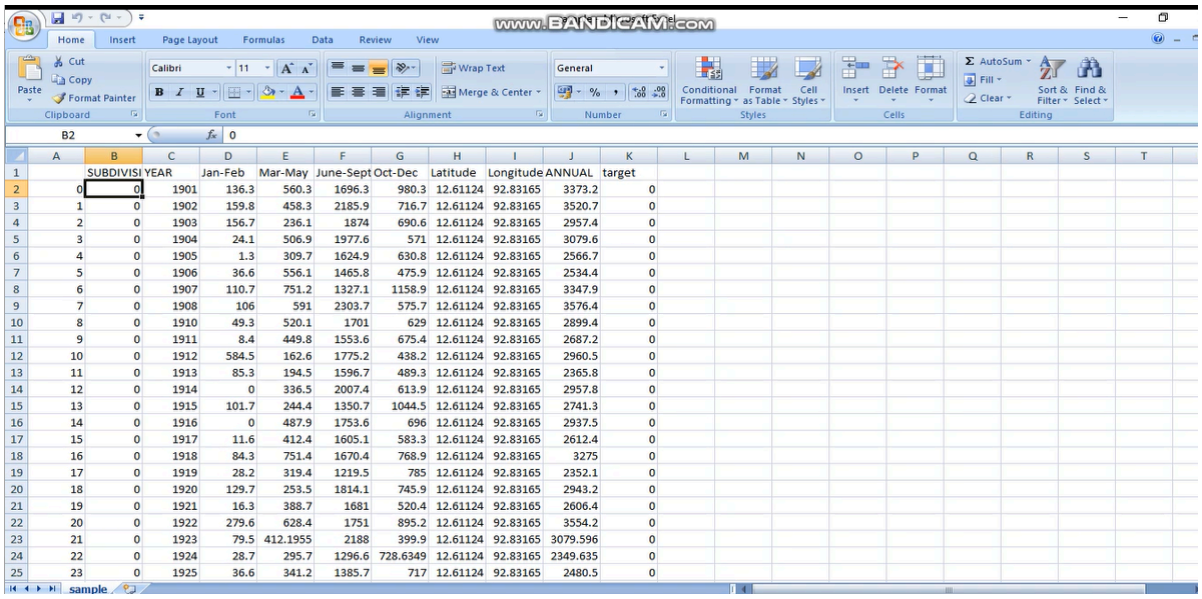
Registration Page



Login Page

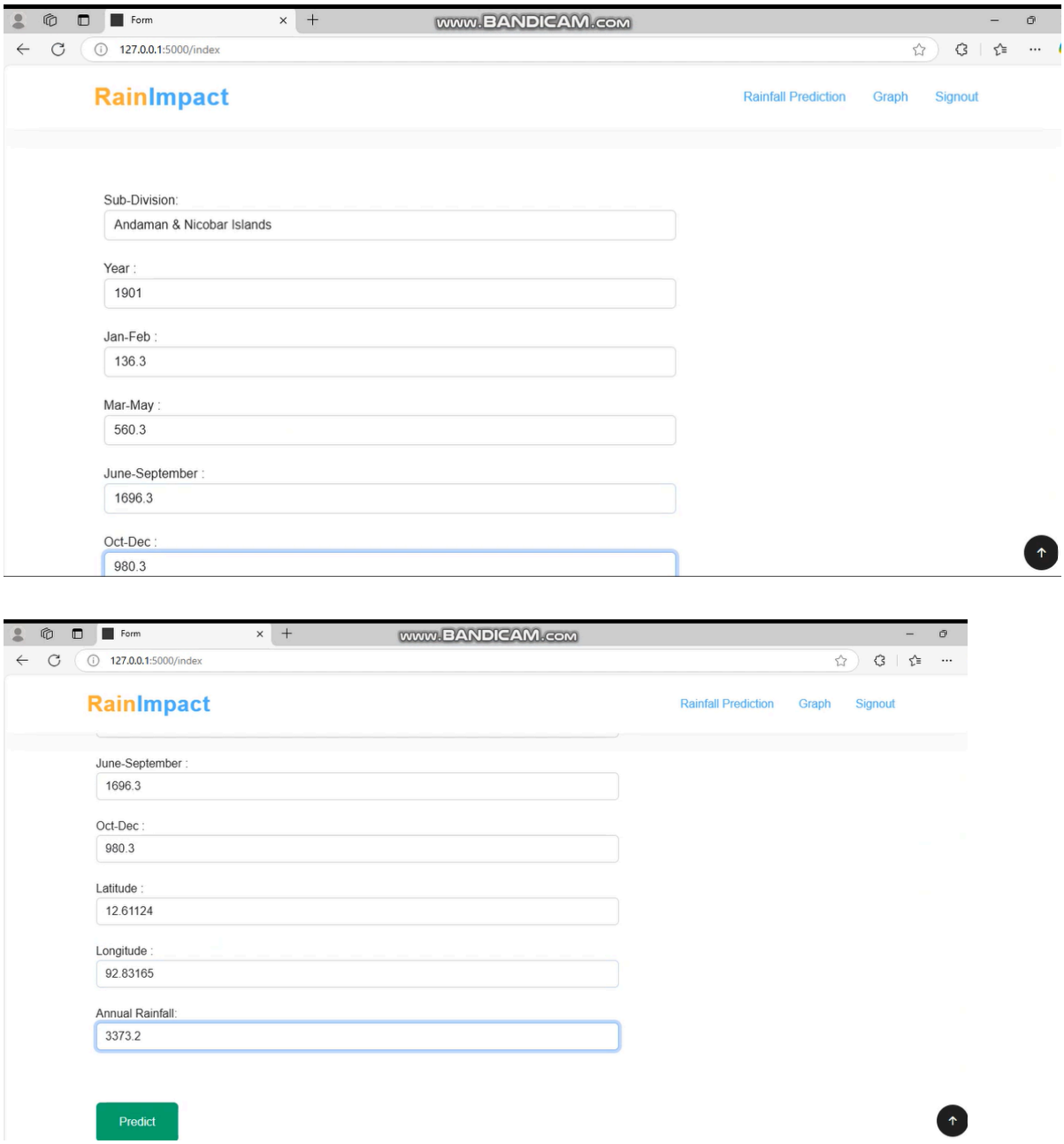


Interface Screen to enter input

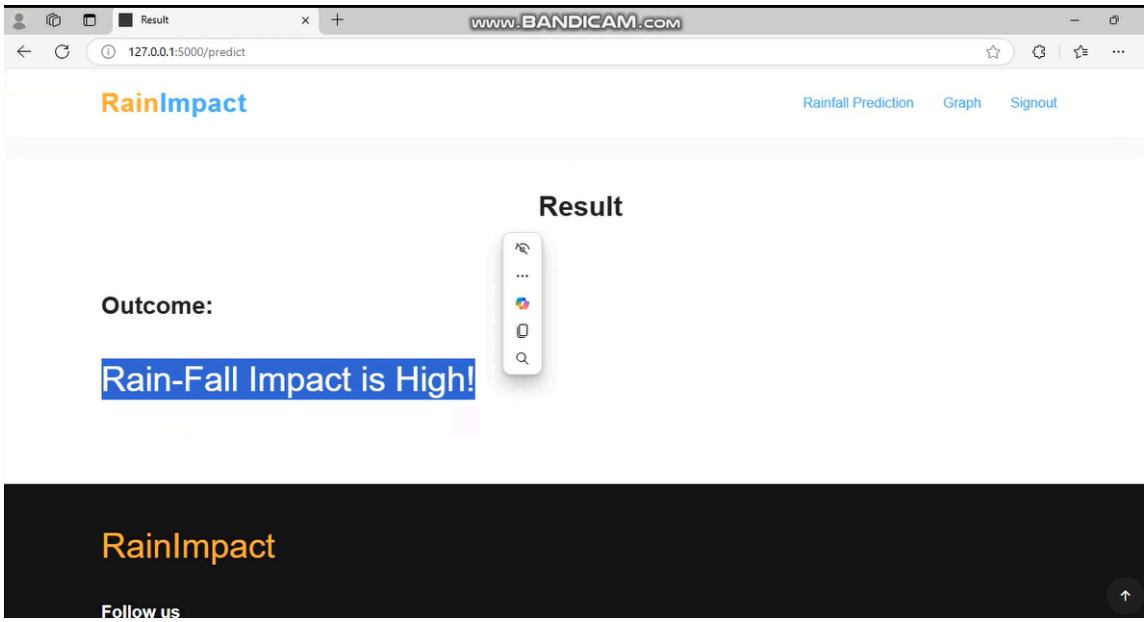


	A	B	C	D	E	F	G	H	I	J	K
1		SUBDIVISI	YEAR	Jan-Feb	Mar-May	June-Sept	Oct-Dec	Latitude	Longitude	ANNUAL	target
2	0	0	1901	136.3	560.3	1696.3	980.3	12.61124	92.83165	3373.2	0
3	1	0	1902	159.8	458.3	2185.9	716.7	12.61124	92.83165	3520.7	0
4	2	0	1903	156.7	236.1	1874	690.6	12.61124	92.83165	2957.4	0
5	3	0	1904	24.1	506.9	1977.6	571	12.61124	92.83165	3079.6	0
6	4	0	1905	1.3	309.7	1624.9	630.8	12.61124	92.83165	2566.7	0
7	5	0	1906	36.6	556.1	1465.8	475.9	12.61124	92.83165	2534.4	0
8	6	0	1907	110.7	751.2	1327.1	1158.9	12.61124	92.83165	3347.9	0
9	7	0	1908	106	591	2303.7	575.7	12.61124	92.83165	3576.4	0
10	8	0	1910	49.3	520.1	1701	629	12.61124	92.83165	2899.4	0
11	9	0	1911	8.4	449.8	1553.6	675.4	12.61124	92.83165	2687.2	0
12	10	0	1912	584.5	162.6	1775.2	438.2	12.61124	92.83165	2960.5	0
13	11	0	1913	85.3	194.5	1596.7	489.3	12.61124	92.83165	2365.8	0
14	12	0	1914	0	336.5	2007.4	613.9	12.61124	92.83165	2957.8	0
15	13	0	1915	101.7	244.4	1350.7	1044.5	12.61124	92.83165	2741.3	0
16	14	0	1916	0	487.9	1753.6	696	12.61124	92.83165	2937.5	0
17	15	0	1917	11.6	412.4	1605.1	583.3	12.61124	92.83165	2612.4	0
18	16	0	1918	84.3	751.4	1670.4	768.9	12.61124	92.83165	3275	0
19	17	0	1919	28.2	319.4	1219.5	785	12.61124	92.83165	2352.1	0
20	18	0	1920	129.7	253.5	1814.1	745.9	12.61124	92.83165	2943.2	0
21	19	0	1921	16.3	388.7	1681	520.4	12.61124	92.83165	2606.4	0
22	20	0	1922	279.6	628.4	1751	895.2	12.61124	92.83165	3554.2	0
23	21	0	1923	79.5	412.1955	2188	399.9	12.61124	92.83165	3079.596	0
24	22	0	1924	28.7	295.7	1296.6	728.6349	12.61124	92.83165	2349.635	0
25	23	0	1925	36.6	341.2	1385.7	717	12.61124	92.83165	2480.5	0

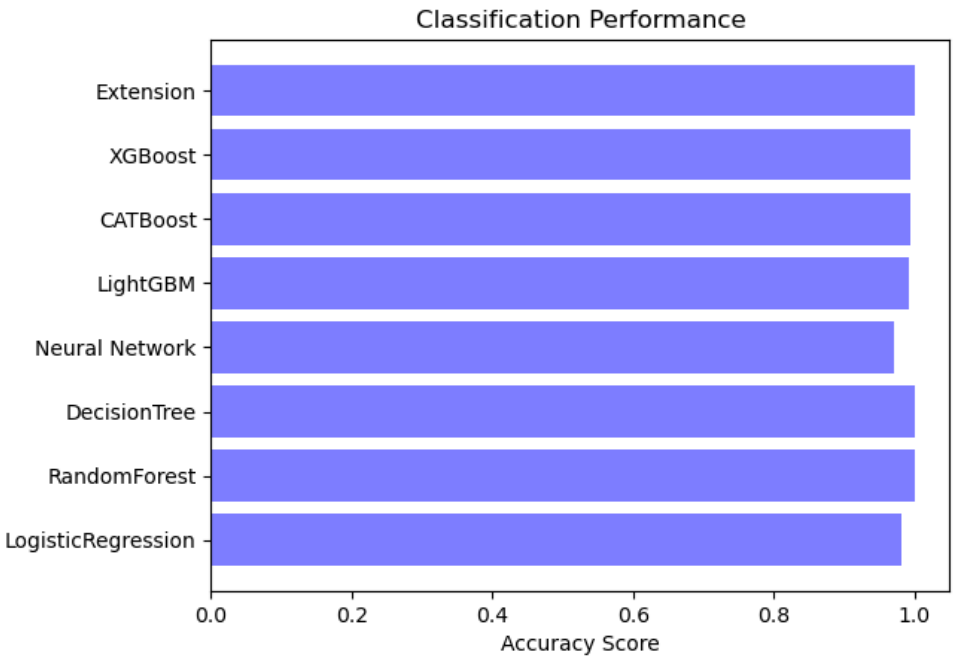
Input Values



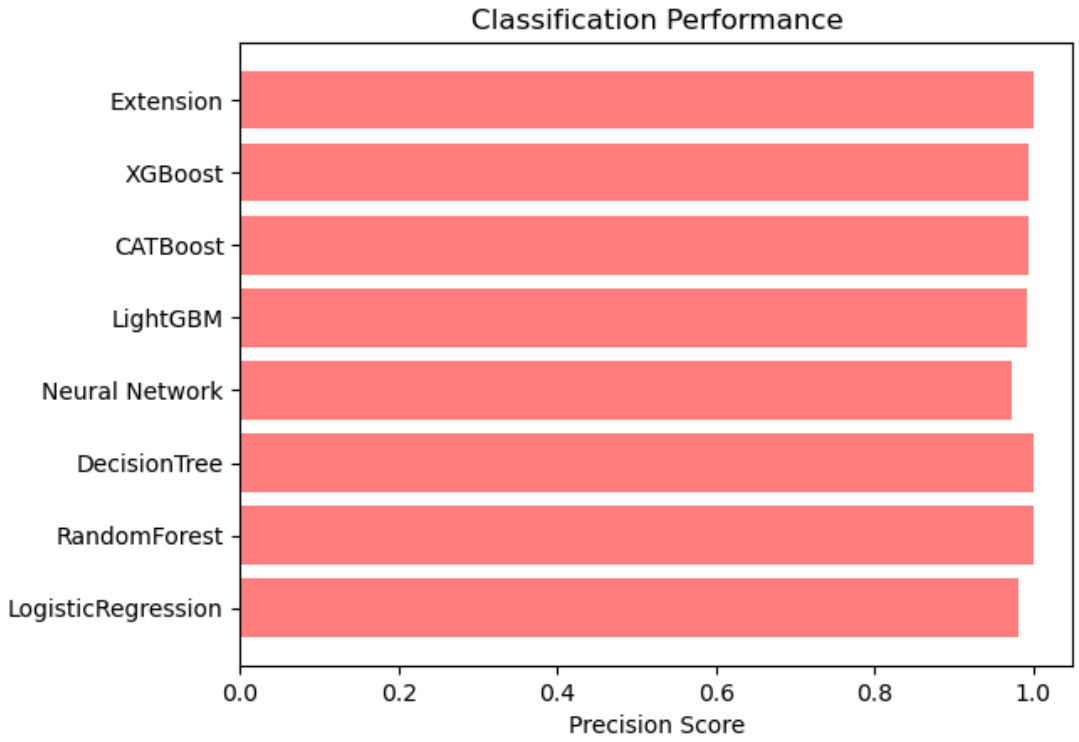
Upload Input Values



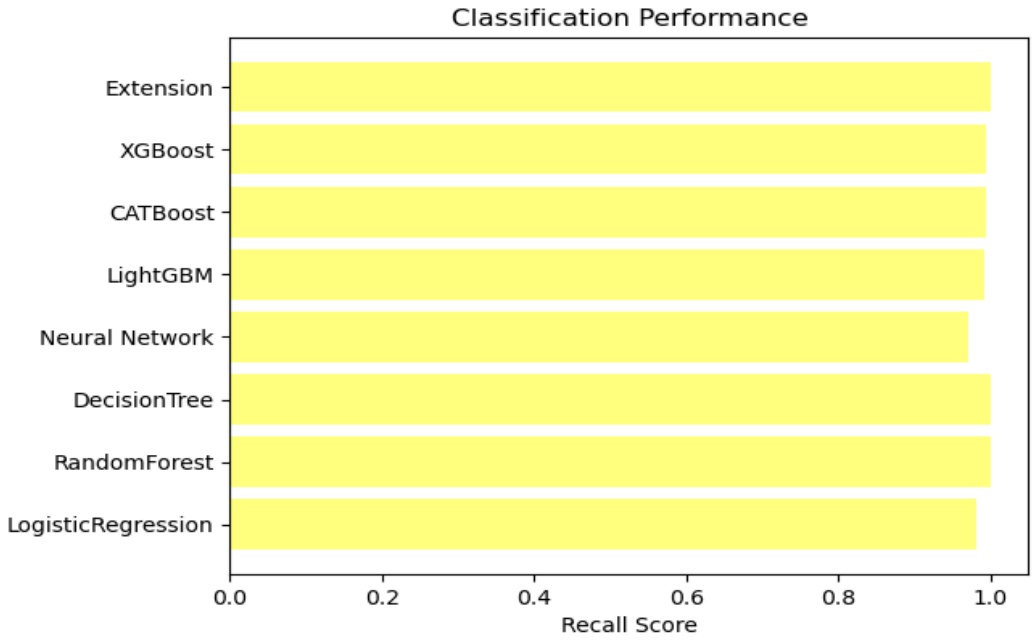
Predict Result



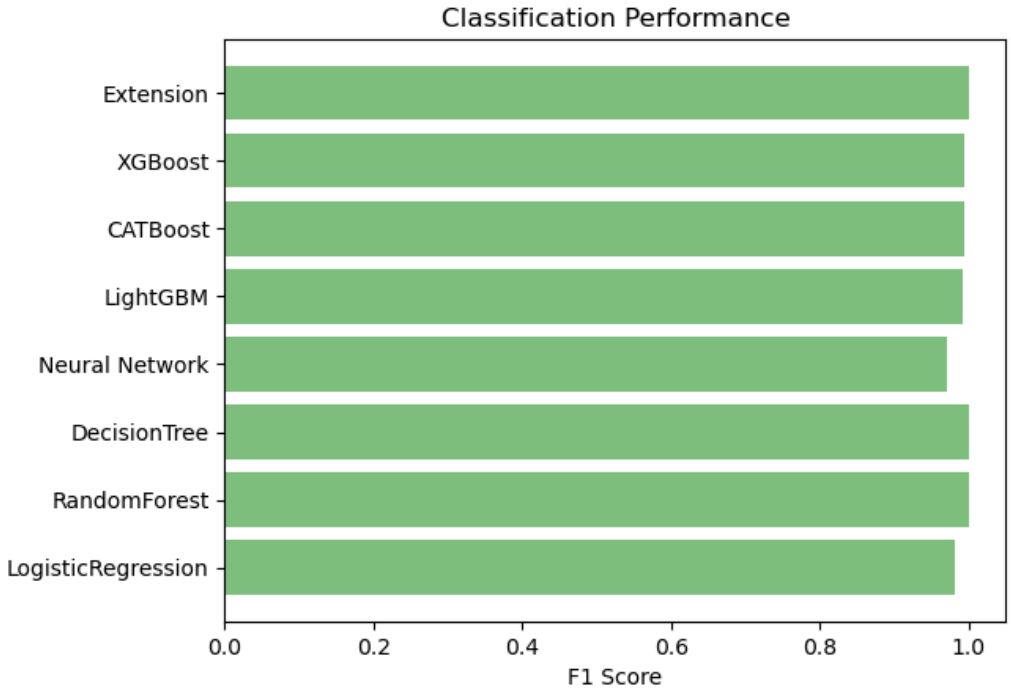
Accuracy Graph



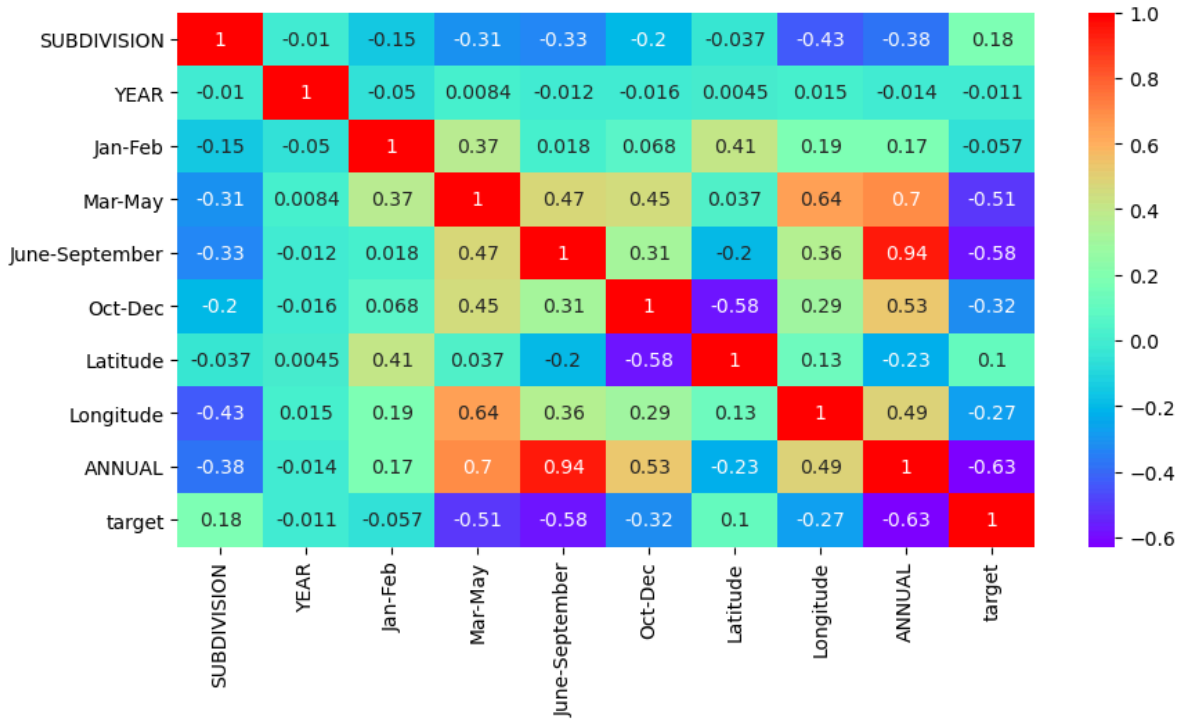
Precision Graph



Recall Graph



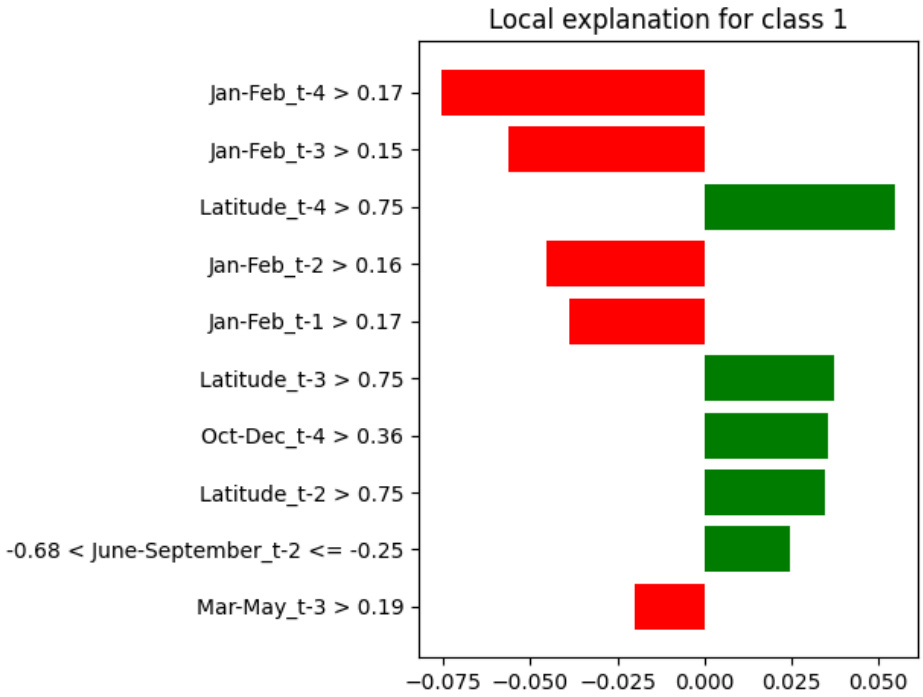
F1 Score Graph



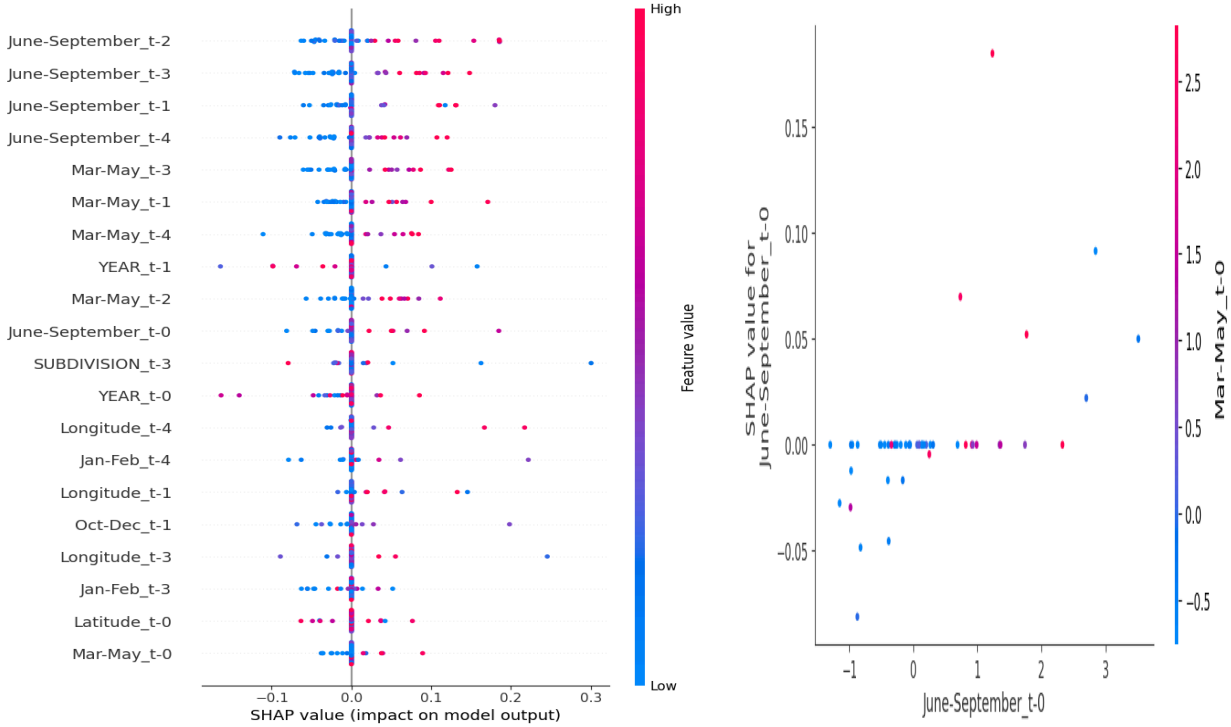
Confusion Matrix of Model

	ML Model	Accuracy	Precision	Recall	F1_score
0	LogisticRegression	0.981	0.981	0.981	0.981
1	RandomForest	1.000	1.000	1.000	1.000
2	DecisionTree	1.000	1.000	1.000	1.000
3	Neural Network	0.971	0.973	0.971	0.971
4	LightGBM	0.992	0.992	0.992	0.992
5	CATBoost	0.994	0.994	0.994	0.994
6	XGBoost	0.995	0.995	0.995	0.995
7	Extension	1.000	1.000	1.000	1.000

Metrics Results Comparison of Algorithms Used



LSTM with LIME



LSTM with SHAP

```
Epoch 1/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 7s 14ms/step - accuracy: 0.6813 - loss: 0.8093 - val_accuracy: 0.7930 - val_loss: 0.4338
Epoch 2/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - accuracy: 0.7756 - loss: 0.4695 - val_accuracy: 0.8017 - val_loss: 0.4081
Epoch 3/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - accuracy: 0.7973 - loss: 0.4528 - val_accuracy: 0.7980 - val_loss: 0.3983
Epoch 4/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 2s 15ms/step - accuracy: 0.7938 - loss: 0.4536 - val_accuracy: 0.8105 - val_loss: 0.3969
Epoch 5/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - accuracy: 0.7944 - loss: 0.4356 - val_accuracy: 0.8180 - val_loss: 0.3920
Epoch 6/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 2s 13ms/step - accuracy: 0.8082 - loss: 0.4338 - val_accuracy: 0.8142 - val_loss: 0.3943
Epoch 7/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.8001 - loss: 0.4428 - val_accuracy: 0.8130 - val_loss: 0.4016
Epoch 8/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - accuracy: 0.8100 - loss: 0.4302 - val_accuracy: 0.8092 - val_loss: 0.3890
Epoch 9/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - accuracy: 0.8109 - loss: 0.4261 - val_accuracy: 0.8092 - val_loss: 0.3855
Epoch 10/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - accuracy: 0.7956 - loss: 0.4431 - val_accuracy: 0.8080 - val_loss: 0.3932
Epoch 11/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - accuracy: 0.8033 - loss: 0.4426 - val_accuracy: 0.8130 - val_loss: 0.3804
Epoch 12/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - accuracy: 0.7967 - loss: 0.4427 - val_accuracy: 0.8042 - val_loss: 0.4036
Epoch 13/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - accuracy: 0.7890 - loss: 0.4456 - val_accuracy: 0.8105 - val_loss: 0.3842
Epoch 14/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - accuracy: 0.8182 - loss: 0.4316 - val_accuracy: 0.8217 - val_loss: 0.3856
Epoch 15/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - accuracy: 0.8204 - loss: 0.4134 - val_accuracy: 0.8292 - val_loss: 0.3833
Epoch 16/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - accuracy: 0.8195 - loss: 0.4193 - val_accuracy: 0.8229 - val_loss: 0.3850
Epoch 17/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 2s 15ms/step - accuracy: 0.7983 - loss: 0.4290 - val_accuracy: 0.8292 - val_loss: 0.3855
Epoch 18/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7997 - loss: 0.4270 - val_accuracy: 0.8192 - val_loss: 0.3835
Epoch 19/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - accuracy: 0.8137 - loss: 0.4141 - val_accuracy: 0.8217 - val_loss: 0.3874
Epoch 20/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - accuracy: 0.8071 - loss: 0.4308 - val_accuracy: 0.8279 - val_loss: 0.3823
Epoch 21/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - accuracy: 0.8213 - loss: 0.4056 - val_accuracy: 0.8242 - val_loss: 0.3819
Epoch 22/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - accuracy: 0.8069 - loss: 0.4173 - val_accuracy: 0.8292 - val_loss: 0.3862
Epoch 23/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - accuracy: 0.8155 - loss: 0.4164 - val_accuracy: 0.8155 - val_loss: 0.3904
Epoch 24/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - accuracy: 0.8215 - loss: 0.4085 - val_accuracy: 0.8267 - val_loss: 0.3792
Epoch 25/30
```

```
Epoch 8/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - accuracy: 0.8100 - loss: 0.4302 - val_accuracy: 0.8092 - val_loss: 0.3890
Epoch 9/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - accuracy: 0.8109 - loss: 0.4261 - val_accuracy: 0.8092 - val_loss: 0.3855
Epoch 10/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - accuracy: 0.7956 - loss: 0.4431 - val_accuracy: 0.8080 - val_loss: 0.3932
Epoch 11/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - accuracy: 0.8033 - loss: 0.4426 - val_accuracy: 0.8130 - val_loss: 0.3804
Epoch 12/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - accuracy: 0.7967 - loss: 0.4427 - val_accuracy: 0.8042 - val_loss: 0.4036
Epoch 13/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - accuracy: 0.7890 - loss: 0.4456 - val_accuracy: 0.8105 - val_loss: 0.3842
Epoch 14/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - accuracy: 0.8182 - loss: 0.4316 - val_accuracy: 0.8217 - val_loss: 0.3856
Epoch 15/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - accuracy: 0.8204 - loss: 0.4134 - val_accuracy: 0.8292 - val_loss: 0.3833
Epoch 16/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - accuracy: 0.8195 - loss: 0.4193 - val_accuracy: 0.8229 - val_loss: 0.3850
Epoch 17/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 2s 15ms/step - accuracy: 0.7983 - loss: 0.4290 - val_accuracy: 0.8292 - val_loss: 0.3855
Epoch 18/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.7997 - loss: 0.4270 - val_accuracy: 0.8192 - val_loss: 0.3835
Epoch 19/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - accuracy: 0.8137 - loss: 0.4141 - val_accuracy: 0.8217 - val_loss: 0.3874
Epoch 20/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - accuracy: 0.8071 - loss: 0.4308 - val_accuracy: 0.8279 - val_loss: 0.3823
Epoch 21/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - accuracy: 0.8213 - loss: 0.4056 - val_accuracy: 0.8242 - val_loss: 0.3819
Epoch 22/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - accuracy: 0.8069 - loss: 0.4173 - val_accuracy: 0.8292 - val_loss: 0.3862
Epoch 23/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - accuracy: 0.8155 - loss: 0.4164 - val_accuracy: 0.8155 - val_loss: 0.3904
Epoch 24/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - accuracy: 0.8215 - loss: 0.4085 - val_accuracy: 0.8267 - val_loss: 0.3792
Epoch 25/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - accuracy: 0.8112 - loss: 0.4200 - val_accuracy: 0.8254 - val_loss: 0.3819
Epoch 26/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 2s 14ms/step - accuracy: 0.8175 - loss: 0.4105 - val_accuracy: 0.8192 - val_loss: 0.3862
Epoch 27/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - accuracy: 0.8058 - loss: 0.4253 - val_accuracy: 0.8354 - val_loss: 0.3819
Epoch 28/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - accuracy: 0.8103 - loss: 0.4221 - val_accuracy: 0.8217 - val_loss: 0.3977
Epoch 29/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - accuracy: 0.8263 - loss: 0.3900 - val_accuracy: 0.8279 - val_loss: 0.3840
Epoch 30/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - accuracy: 0.8176 - loss: 0.4041 - val_accuracy: 0.8292 - val_loss: 0.3869
Test Accuracy: 0.8292
LIME explanation saved as 'lime_explanation.html'
LIME feature importance plot saved as 'lime_feature_importance.png'
```

Epochs and Accuracy for LSTM with LIME

```
Epoch 1/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 7s 15ms/step - accuracy: 0.6214 - loss: 0.8404 - val_accuracy: 0.8067 - val_loss: 0.4491
Epoch 2/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - accuracy: 0.7736 - loss: 0.4989 - val_accuracy: 0.8005 - val_loss: 0.4027
Epoch 3/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - accuracy: 0.8012 - loss: 0.4425 - val_accuracy: 0.8017 - val_loss: 0.4015
Epoch 4/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - accuracy: 0.7995 - loss: 0.4451 - val_accuracy: 0.8092 - val_loss: 0.4008
Epoch 5/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - accuracy: 0.7987 - loss: 0.4387 - val_accuracy: 0.8180 - val_loss: 0.4011
Epoch 6/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 2s 17ms/step - accuracy: 0.8021 - loss: 0.4344 - val_accuracy: 0.7905 - val_loss: 0.4135
Epoch 7/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - accuracy: 0.8082 - loss: 0.4347 - val_accuracy: 0.8080 - val_loss: 0.4070
Epoch 8/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - accuracy: 0.8043 - loss: 0.4243 - val_accuracy: 0.8192 - val_loss: 0.3973
Epoch 9/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - accuracy: 0.8090 - loss: 0.4252 - val_accuracy: 0.8155 - val_loss: 0.3846
Epoch 10/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - accuracy: 0.7967 - loss: 0.4371 - val_accuracy: 0.8192 - val_loss: 0.3869
Epoch 11/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - accuracy: 0.8039 - loss: 0.4367 - val_accuracy: 0.8155 - val_loss: 0.3869
Epoch 12/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - accuracy: 0.8061 - loss: 0.4277 - val_accuracy: 0.8217 - val_loss: 0.3902
Epoch 13/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - accuracy: 0.8168 - loss: 0.4085 - val_accuracy: 0.8229 - val_loss: 0.3867
Epoch 14/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - accuracy: 0.8082 - loss: 0.4199 - val_accuracy: 0.8067 - val_loss: 0.4041
Epoch 15/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - accuracy: 0.8085 - loss: 0.4421 - val_accuracy: 0.8192 - val_loss: 0.3799
Epoch 16/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.8169 - loss: 0.4139 - val_accuracy: 0.8192 - val_loss: 0.3887
Epoch 17/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.8095 - loss: 0.4100 - val_accuracy: 0.8117 - val_loss: 0.3885
Epoch 18/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - accuracy: 0.8004 - loss: 0.4324 - val_accuracy: 0.8254 - val_loss: 0.3791
Epoch 19/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - accuracy: 0.8186 - loss: 0.4078 - val_accuracy: 0.8180 - val_loss: 0.3952
Epoch 20/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - accuracy: 0.8036 - loss: 0.4276 - val_accuracy: 0.8254 - val_loss: 0.3803
Epoch 21/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - accuracy: 0.8207 - loss: 0.4071 - val_accuracy: 0.8167 - val_loss: 0.3880
Epoch 22/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - accuracy: 0.8272 - loss: 0.3971 - val_accuracy: 0.8267 - val_loss: 0.3805
Epoch 23/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - accuracy: 0.8109 - loss: 0.4108 - val_accuracy: 0.8142 - val_loss: 0.3887
Epoch 24/30
101/101 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - accuracy: 0.8155 - loss: 0.4190 - val_accuracy: 0.8254 - val_loss: 0.3771
Epoch 25/30
```

Epochs and Accuracy for LSTM with SHAP

8. **Project Outcomes**

1. **Building a clear AI frame:** A new AI operated model was created, to predict high effects of rainfall with more openness and interpretation.
2. **Increased prediction accuracy:** By using advanced machine learning techniques, XAI-based models improved traditional weather forecasting techniques when it comes to predicting accuracy.
3. **Analysis of the importance of the plant:** By identifying important meteorological factors affecting severe precipitation forecasts, the study highlighted the model decision.
4. **Integration with meteorological data:** The model's ability to handle and interpret the climate variable was improved by successful integration of broad meteorological information.
5. **Visualization of AI decisions:** The predictions for the AI model were shown and justified by the use of a clear approach as values or lime, which increased the confidence of meteorologists.
6. **Better decision support for disaster management:** During extreme weather events, the model provided practical insight to help meteorologists and disaster making in well

-informed decision -making.

7. **Benchmarking against existing models:** the extraordinary accuracy and dependence on the XAI-based approach was shown through the performance comparison with the current forecast model.
8. **Lack of false positivity and negative:** The model was improved by using high effects of rainfall and clarity techniques to reduce false alarms.
9. **Scalability for regional and global applications:** The established approach can be used for a variety of geographical sites, which can enable a wide range of climate research applications.
10. **Future Research Prospects:** The research paved the way for explainable AI in meteorology by supporting transdisciplinary AI-driven weather forecasting.

9. Limitations & Challenges

- **Calculation complexity:** Complex machine learning models require too much processor power.
- **Data Dependency:** Reliable meteorological data is essential for model performance.
- **Risks of Overfitting:** intricate models have the potential to overfit past weather trends.
- **Real-Time Processing:** To make predictions in real time, a high-speed computing infrastructure is required.

10. **Conclusion**

Reducing the likelihood of floods, landslides, and infrastructure damage requires precise forecasting of heavy rainfall events. Extreme precipitation is predicted by this method utilizing machine learning and deep learning techniques, using weather data from OpenWeather API and IMD. A number of models are used to improve the accuracy of predictions, including “XGBoost, Decision Tree, Random Forest, Neural Networks, LightGBM, and Logistic Regression”. An Ensemble Model and Voting Classifier are also included. Methods from the field of “explainable artificial intelligence (XAI), such LIME and SHAP”, make model decisions more transparent and easier to understand by displaying the relevance of features. Incorporating Bagging-based Random Forest with Boosted Decision Trees further increases the performance of the prediction. The results suggest that the clothing -based method gets a high precise line balance to predict the incidence of rapid rainfall, improves accuracy. Access and purposes in real time are guaranteed by a certification-based Kolbe interface. A reliable and explanatory solution for prediction of extreme rainfall is given by this system, which helps disaster preparations and reaction activities by integrating advanced dress algorithms, XAI-based insight and an interactive front end.

In order to improve the accuracy of weather predictions, this system can be extended to the future by incorporating "Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM)". Credibility can be improved by including new meteorological data sets, such as satellite images and

radar data, and liquid data floating from the sensor based on the Internet of Things. Inserting on cloud platforms such as AWS or Google Cloud guarantees scalability, while optimizing advanced hyperpieter smoking model performance using automatic frames such as Optuna. Automatic information for the destruction management can be made possible by integrating the system with the first warning network and the system of "Geographic Information Systems (GIS)". Model transparency and user chairs can be further improved with further additions to "Explainable AI (XAI)", such as counterfactual clarification and ability to provide interactive dashboards. The method of practical use by using multimodal data transaction can be even more successful and flexible, which combines structured numerical data with text reports and satellite images to improve estimated accuracy.

References

[1] L. B. Smith, S. S. O'Connor, and A. P. Davies "Interpretability of Machine Learning-based Predictive Models in Meteorology" Atmospheric Science Letters, 2021

[2] Arrieta, A.B., Díaz-Rodríguez, N., Del Ser, J., et al. "A Survey of Explainable Artificial Intelligence (XAI)" Information Fusion, 2020

[3] T. S. McGovern, R. J. Allen, and R. M. Bostrom "Explainable Machine Learning Models in Weather Prediction" Journal of Machine Learning Research, 2019

[4] P. K. Sahoo, K. R. Ghosh, and R. S. Bedi "Using Ensemble Methods for Rainfall Prediction"Environmental Modelling & Software, 2020

[5] A. Prokhorenkova, A. Gusev, S. Vorobev, et al."CatBoost: Gradient Boosting with Categorical Features Support" Proceedings of NeurIPS, 2018

[6] Scott Lundberg and Su-In Lee "SHAP (SHapley Additive exPlanations) for Model Interpretation" ature Machine Intelligence, 2017

[7] Marco Ribeiro, Sameer Singh, and Carlos Guestrin "LIME: Local Interpretable Model-agnostic Explanations" Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2016.

[8] Rasp, S., Pritchard, M. S., & Gentine, P. "Deep Learning to Represent Subgrid Processes in Climate Models" Proceedings of the National Academy of Sciences, 2018.

[9] Chen, T., & Guestrin, C. "XGBoost: A Scalable Tree Boosting System" Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2016.

[10] Lundberg, S. M., & Lee, S. I. "A Unified Approach to Interpretable Machine Learning Models" Advances in Neural Information Processing Systems (NeurIPS), 2017.

[11] Zhao, H., Sun, Y., & Zhao, J. "A Hybrid Machine Learning Model for Rainfall Prediction Using Satellite and Meteorological Data" Remote Sensing, 2019.

[12] Breiman, L. "Random Forests" Machine Learning, 2001.

[13] Hochreiter, S., & Schmidhuber, J. "Long Short-Term Memory" Neural Computation, 1997.

[14] Goodfellow, I., Bengio, Y., & Courville, A. "Deep Learning" MIT Press, 2016.

[15] He, K., Zhang, X., Ren, S., & Sun, J. "Deep Residual Learning for Image Recognition" Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[16] Kingma, D. P., & Ba, J. "Adam: A Method for Stochastic Optimization" International Conference on Learning Representations (ICLR), 2015.

[17] Yager, R. R., & Filev, D. "Essentials of Fuzzy Modeling and Control" John Wiley & Sons, 1994.

[18] Kuhn, M., & Johnson, K. "Applied Predictive Modeling" Springer, 2013.

[19] Rasmussen, C. E., & Williams, C. K. "Gaussian Processes for Machine Learning" MIT Press, 2006.

[20] Wang, J., Zhang, X., & Wang, X. "Ensemble Learning for Meteorological Forecasting: A Review" International Journal of Climatology, 2021.

[21] Krizhevsky, A., Sutskever, I., & Hinton, G. E. "ImageNet Classification with Deep Convolutional Neural Networks" Advances in Neural Information Processing Systems (NeurIPS), 2012.

[22] Geurts, P., Ernst, D., & Wehenkel, L. "Extremely Randomized Trees" Machine Learning, 2006.

[23] Wilks, D. S. "Statistical Methods in the Atmospheric Sciences" Academic Press, 2019.

[24] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. "SMOTE: Synthetic Minority Over-sampling Technique" Journal of Artificial Intelligence Research, 2002.

[25] Tibshirani, R. "Regression Shrinkage and Selection via the Lasso" Journal of the Royal Statistical Society, 1996.