**Leveraging Machine Learning Models for Proactive Disaster Forecasting**

Syed Affan Daimi1, Arslaan Ahmed Askari2, Syed Maaz Ahmed3

*1(**CSE, Muffakham Jah College of Engineering and Technology, India)*

*2(CSE, Muffakham Jah College of Engineering and Technology, India)*

*3(CSE, Muffakham Jah College of Engineering and Technology, India)*

*Corresponding Author: 160420733074@mjcollege.ac.in*

|  |
| --- |
| ***Abstract:*** *The increasing frequency and intensity of natural disasters, such as earthquakes, tsunamis, floods, and forest fires, necessitate the development of advanced early warning systems. Current disaster prediction systems are hampered by limitations in sensor technology, which is often expensive and primarily effective for large-scale disasters. This project aims to enhance disaster prediction accuracy and timeliness by utilizing real-time data and advanced machine learning algorithms, including XGBoost, Voting classifier, and Random Forest classifier. Additionally, recognizing the long-term health impacts of poor air quality, the project extends its scope to predict the Air Quality Index (AQI). The project seeks to enable proactive risk management and effective disaster response by integrating real-time environmental data and machine learning models. We aim to improve proactive risk management and disaster response by employing real-time data. The outcomes of this project promise significant advancements in disaster preparedness, ultimately safeguarding human lives and infrastructure.*  ***Key Word****:**Natural disasters; Early warning systems; Sensor technology; Machine learning algorithms; XGBoost; Random Forest; Air Quality Index; Disaster response; Real-time data; Disaster preparedness* |

1. **Introduction**

Natural disasters are catastrophic events that cause significant damage to human life, property, and the environment. These events, which include earthquakes, tsunamis, floods, forest fires, and others, often strike with little warning, leaving communities in chaos. The devastating impacts of natural disasters have been witnessed repeatedly across the globe. Natural disasters' escalating frequency and severity highlight an urgent need for innovative and accessible prediction systems.

For example, the 2023 Turkey-Syria earthquake resulted in over 50,000 deaths, numerous injuries, and extensive property damage. Similarly, as presented by Satake, K., et al. (2013) the 2011 Tohoku earthquake in Japan caused approximately 16,000 deaths. It led to severe nuclear accidents, highlighting the profound impact such events can have on both human lives and infrastructure. Traditional methods, like those presented by Rundle et al., (2000) for predicting earthquakes, rely heavily on historical data and physics-based models. While these methods have been useful, they often fail in accuracy and timeliness. Smith et al. (2018) and P. Banagr et al. (2020) comprehensively reviewed various machine learning approaches applied to earthquake prediction. They discussed the potential of these models, particularly deep learning and statistical learning techniques, to enhance early warning systems. By analyzing seismic data and other geophysical parameters, these models can improve earthquake detection, localization, and magnitude estimation, ultimately reducing the impact of earthquakes on vulnerable communities.

It is often observed, for example, in work presented by Titov et al., that tsunami prediction typically involves monitoring seismic activity and oceanographic data to detect potential tsunamigenic events. Johnstone et al. (2020) presented a real-time tsunami prediction model that utilizes seismic data and machine learning algorithms. Their approach can rapidly detect and analyze seismic signals to generate early warnings, improving the speed and accuracy of tsunami forecasts. A. Novianty et al. (2019) and Titov et al. (2005) also explored machine learning-based approaches for tsunami forecasting. A. Novianty et al. used K-Nearest Neighbours to model the complex relationships between seismic data and tsunami wave heights, while Titov et al. applied support vector machines to predict tsunami arrival times and amplitudes. Both studies demonstrated the advantages of ML methods regarding speed and accuracy compared to traditional tsunami prediction techniques.

Floods are the other frequently occurring disasters causing huge loss of life and property. Traditional flood prediction methods like Thielen et al. (2009) may not account for rapidly changing weather patterns, and earthquake prediction remains highly uncertain due to the complex nature of seismic activity. Agbo et al. (2019) designed a smart flood disaster prediction framework that integrates IoT (Internet of Things) devices and machine learning algorithms. Their system can provide early warnings by analyzing real-time data from the Openweather API monitoring water levels, rainfall, and other relevant factors. This work showcases how ML can be used to save lives and reduce flood damage potentially. A. Mosavi et al. (2018) and Gupta et al. (2020) reviewed various machine learning models used for flood prediction, including artificial neural networks, decision trees, and ensemble models. They found that ensemble methods, such as random forest and gradient boosting, often outperform single-model approaches in flood forecasting, improving the accuracy of early warning systems. Similarly, Jeerana Noymanee et al. (2017) collected data on flooding occurrences and utilize a Bayesian Linear Model for forecasting flooding phenomena in Pattani River.

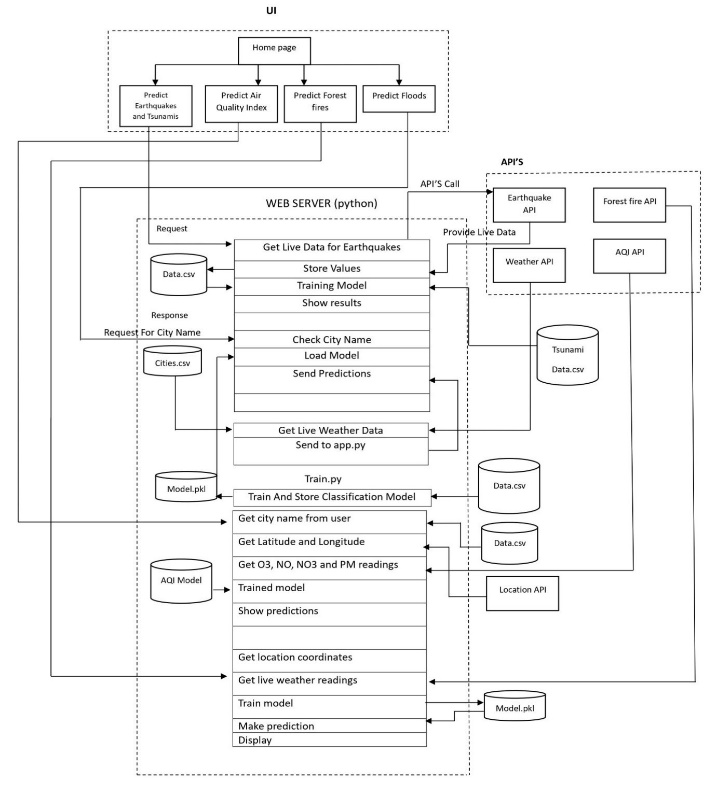
Forest Fires are another natural disaster causing huge loss to the environment. Bowman et al. (2009) discussed the complex role of fire in the Earth system, highlighting its impact on climate, ecosystems, and human societies. They emphasized the need for predictive models to help manage wildfire risks by forecasting fire behavior, spread, and intensity. This review set the stage for subsequent research on applying machine learning techniques to wildfire prediction. F Abid. et al. (2017) provide a dataset of forest fires recorded in Algeria, with basic efforts to model the same. Our work in machine learning modelling improves the accuracy presented in this work from ~80% to 95%.

All these disasters are natural; however, for urban dwellers in particular, the Air Quality Index (AQI) affects the quality of human life. B. Brunekreef et. Al. (2002) provides an overview of the evidence linking air pollution to various health outcomes, including respiratory and cardiovascular diseases, and discusses the mechanisms by which air pollutants may exert their effects. Kim et al. (2019) and Liu et al. (2018) independently developed deep learning models, specifically recurrent neural networks, to forecast air quality. These models can capture the temporal dependencies and complex relationships in air quality data, achieving more accurate predictions than traditional statistical methods. Londhe (2019) applied data mining and machine learning techniques like decision trees and neural networks to predict the Air Quality Index (AQI). By analyzing historical data on pollutants, weather conditions, and other factors, their model could provide timely and accurate air quality forecasts, helping authorities take proactive measures to improve air quality and protect public health.

Machine Learning: Machine learning (ML) is a powerful tool with many applications in disaster prediction and management. Various algorithms have been developed and applied to enhance early warning systems and improve prediction accuracy. Among the most prominent algorithms are XGBoost, presented by Chun Et al, a scalable tree-boosting system known for its state-of-the-art performance; Random Forest (Brieman L), an ensemble learning method effective in uncovering patterns in data; and Logistic Regression, a fundamental algorithm for binary classification. Additionally, ensemble learning has been extended through Voting Classifiers, which combine models for improved predictions, and Ensemble Regression, offering accurate estimates in software development efforts (Carvalho et al). These ML techniques provide automated procedures for prediction and insight generation, making them integral to modern scientific research.

Furthermore, as presented by Anderson, D. P., et. al (2019) ML enables cost-effective and accessible data processing and analysis. By leveraging algorithms that learn from data, ML reduces the need for manual labor, resulting in increased efficiency and reduced costs. The automation provided by ML enhances productivity, allowing for more data to be processed with fewer resources. Additionally, ML algorithms can identify complex patterns and relationships in data, improving prediction accuracy and decision-making. This accessibility and accuracy in data analysis make ML a valuable tool for various domains, including disaster management, healthcare, and business intelligence.

1. **System Design**



The web server, developed in Python, serves as the core processing unit and interacts with external APIs to gather live data. This workflow begins with data acquisition from multiple sources: earthquake data is fetched from an Earthquake API, weather data from a Weather API, and forest fire data from a Forest Fire API. This real-time data is stored in various CSV files for processing.

The server first collects live data for earthquake and tsunami predictions, which is stored in Data.csv. The server trains a model using this data through a structured process involving data loading, preprocessing, and model validation. The trained model predicts seismic activities, with results displayed to the user.

Flood prediction involves gathering live weather data and city-specific information stored in Cities.csv. The server checks the requested city name against the database, loads the corresponding machine-learning model, and processes the live data to generate flood predictions. These predictions are then sent back to the UI for user interpretation.

Forest fire prediction follows a similar process. The server retrieves live weather data and processes it to predict the likelihood of forest fires. The data, stored in Data.csv, is used to train the model, which then forecasts fire probabilities. The predictions are displayed on the Forest Fire Prediction page.

For AQI prediction, the system gathers environmental data, including pollutant readings (O3, NO, NO3, PM), from relevant APIs. The server processes this data and trains a classification model stored in AQI Model. The model predicts the air quality index, which is then presented to the user on the AQI Prediction page.

**User Experience:**

The user experience (UI) of the proactive disaster forecasting web application is designed to be intuitive, informative, and user-friendly, enabling users to easily navigate through various prediction tools for natural disasters. The application is structured around five main pages: the Homepage, Earthquake and Tsunami Prediction Page, Flood Prediction Page, Forest Fire Prediction Page, and Air Quality Index (AQI) Prediction Page. Each page is tailored to provide specific insights and predictions related to its respective natural phenomena.

**Homepage:**

The homepage serves as the central hub of the application, offering users a straightforward choice to select the type of disaster prediction they are interested in. The design is clean and uncluttered, featuring clear buttons or links that lead to the different prediction pages. This ensures that users can quickly and efficiently navigate to the area of their interest without unnecessary complexity.

**Earthquake and Tsunami Prediction Page:**

Upon selecting the "Predict Earthquakes'' option from the homepage, users are directed to the Earthquake and Tsunami Prediction Page. A slider allows users to choose a prediction date, dynamically updating a map that visualizes likely earthquake locations. The map is interactive, providing a geographical representation of potential earthquake hotspots. Below the map, a list of these locations is provided, with tsunami-prone areas highlighted in red text for easy identification. This visual differentiation helps users quickly identify regions at higher risk of tsunamis.

**Flood Prediction Page:**

When users opt to predict floods by selecting "Predict Floods" from the homepage, they are taken to the Flood Prediction Page. Here, users are prompted to enter the name of a city for which they seek flood predictions. The UI then displays the likelihood of a flood occurring in the specified area. If a flood is likely, a message indicating the high probability is shown along with current weather data. Conversely, if a flood is unlikely, a reassuring message is displayed, accompanied by current weather information. This immediate feedback helps users make informed decisions based on the predicted flood risks and the real-time weather conditions.

**Forest Fire Prediction Page:**

The Forest Fire Prediction Page is accessed by selecting the corresponding option from the homepage. On this page, users are presented with an interactive map where they can drag a pointer to any location they are interested in. After submitting a prediction request for the specified area, the server processes the data and analyzes the probability of a forest fire occurring in that location. The UI then displays the predicted likelihood of a fire, providing users with critical information about potential fire risks in the selected region. This interactive map-based approach allows users to explore different areas and understand the varying levels of risk associated with forest fires.

**Air Quality Index (AQI) Prediction Page:**

The AQI Prediction Page is designed to provide users with insights into air quality. Users input the name of a major city, and the server processes this request to calculate the expected Air Quality Index (AQI) for the specified location. The resulting AQI value is displayed prominently on the screen, along with information about its health effects. This includes color-coded indicators that correspond to different levels of air quality, ranging from good to hazardous, helping users easily interpret the data. Additionally, the page provides contextual information about what the AQI value means for health, enabling users to understand the potential impact of air quality on their well-being.

1. **Earthquake and Tsunami Prediction**

**Prediction Pipeline:**

Our work on Earthquake and Tsunami Prediction involved the development of a comprehensive pipeline that could predict potential earthquake locations and assess the risk of subsequent tsunamis. This section details the methodologies employed, the data utilized, the model training processes, and the application interface for presenting the predictions.

The initial phase of our project involved the collection of earthquake data, which was retrieved from the Earthquake API provided by the United States Geological Survey (USGS). The USGS offers real-time data that updates every minute, enabling us to access the most current information. We focused on data from the past 30 days, which included 22 features and a substantial number of samples. This data was then stored in a local storage system to facilitate further processing.

Preprocessing and feature selection were critical steps to ensure the data's integrity and suitability for model training. The preprocessing involved cleaning the data by removing any null values and formatting it appropriately. Feature engineering was then conducted to create meaningful features that could enhance the predictive power of our models. The selected features included rolling averages for the 'depth' and 'mag' (magnitude) columns using different window sizes (22, 15, and 7 days). This approach allowed us to predict earthquake magnitudes in the subsequent 7 days for various zones. A new column, 'mag\_outcome', was created to hold the predicted magnitude values, obtained by shifting the 7-day rolling average values to simulate future predictions. The cleaned and processed data was stored in a CSV file.

Model training was carried out using two distinct machine-learning algorithms. We employed the XGBoost algorithm for earthquake prediction, which is known for its high accuracy and robustness in handling class imbalances. The dataset was split into training and testing sets, and the XGBoost classifier was trained using these sets. The model parameters were fine-tuned to optimize performance, and early stopping was applied to prevent overfitting. The XGBoost model achieved an impressive AUC score of 0.9766 and a recall score of 0.804, making it well-suited for our task. The trained model was then used to make predictions on the live data, which involved creating a DMatrix object to store the live data and obtaining predictions using the trained XGBoost model.

The live data for earthquake prediction was obtained from the USGS website, which provided a CSV file with real-time earthquake occurrences. This data was processed and stored in a data frame, with rolling averages calculated for the 'depth' and 'mag' columns. The live data was then fed into the trained XGBoost model to make predictions for future earthquake occurrences. These predictions were added back to the live dataset, and any duplicates were aggregated. The date in the live data was incremented by the number of days to predict the future, representing the prediction horizon. This processed live data, containing predictions for future earthquake occurrences, was stored in a CSV file named df\_live.csv.

We leveraged a separate dataset for tsunami prediction that included information on tsunami occurrences. This data was sourced from an online repository provided by ArcGIS, which offered a CSV file containing 33 features. After extracting and cleaning the relevant columns, we focused on key features such as magnitude, tsunami indicator, longitude, latitude, and depth. Missing values in the "tsunami" column were filled with zeros to indicate the absence of a tsunami condition, and any remaining rows with missing values were dropped. This cleaned dataset was then stored in another CSV file named Tsunami\_data.csv.

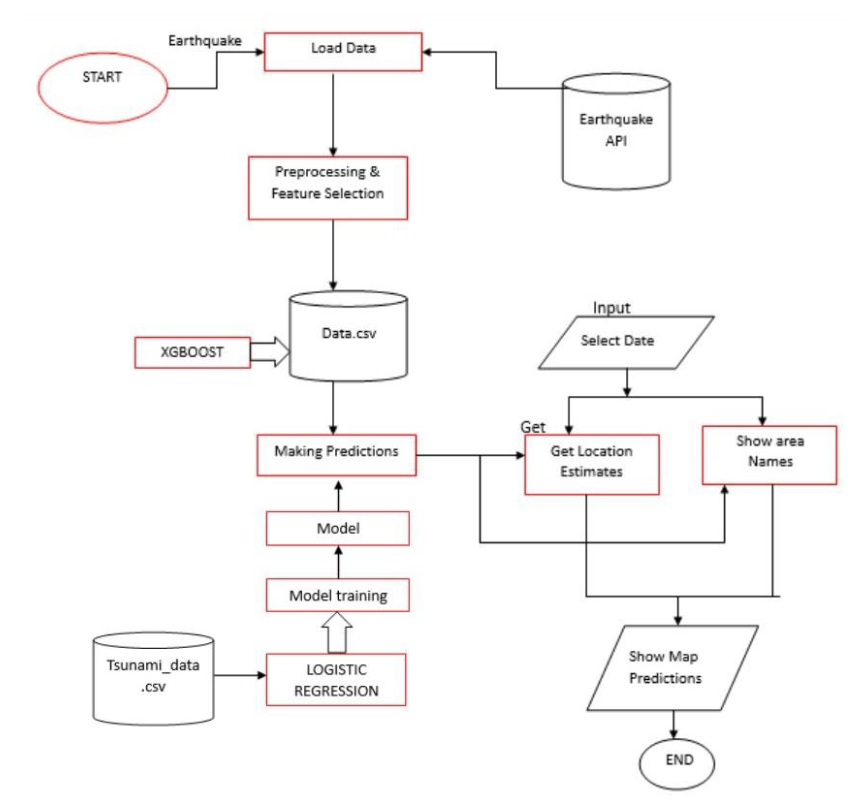
A logistic regression model was utilized on the tsunami dataset. Given the imbalanced nature of the data, class weighting was a critical aspect of the implementation. Class weights were calculated based on the distribution of the tsunami target variable, and these weights were applied to the logistic regression model to mitigate bias towards the majority class. The model pipeline included a StandardScaler for feature scaling and the logistic regression model with the calculated class weights. The model was trained using features such as magnitude, longitude, latitude, and depth, and its performance was evaluated on the test set.

The results of the earthquake predictions were used as input for the tsunami model. The logistic regression model was applied to the live earthquake data to predict the occurrence of tsunamis. The predictions were stored in a CSV file and could be accessed via an API at the view level. This API, implemented using Flask, handled GET requests and served the live dataset as a JSON response. The route '/api/df\_live' was defined to convert the data frame to a JSON string and return it as a response to the client.

The application interface allowed users to interact with the prediction system in a user-friendly manner. Users could input a specific date for which they required predictions, and the application would retrieve the relevant locations and display them on a map. The map highlighted areas at risk of earthquakes, and the names of the affected areas were also displayed. This visualization helped users easily identify regions at risk and take necessary precautions.

The overall pipeline for earthquake and tsunami prediction was structured to ensure seamless integration between data collection, preprocessing, model training, and prediction. The use of real-time data and robust machine learning models enabled accurate and timely predictions, which are crucial for disaster management and mitigation efforts. By leveraging the power of XGBoost for earthquake prediction and logistic regression for tsunami prediction, our system achieved high accuracy and reliability, making it a valuable tool for predicting natural disasters and reducing their impact on communities.

In conclusion, our project on Earthquake and Tsunami Prediction demonstrated the effectiveness of combining advanced machine learning techniques with real-time data to predict natural disasters. The comprehensive pipeline developed for this project can serve as a foundation for further research and development in disaster prediction and management. The detailed explanation of the methodologies, data processing, and model training provided in this section highlights the innovative approach taken to address the challenges of earthquake and tsunami prediction.

****

**Results:**

**Earthquake Prediction Models**

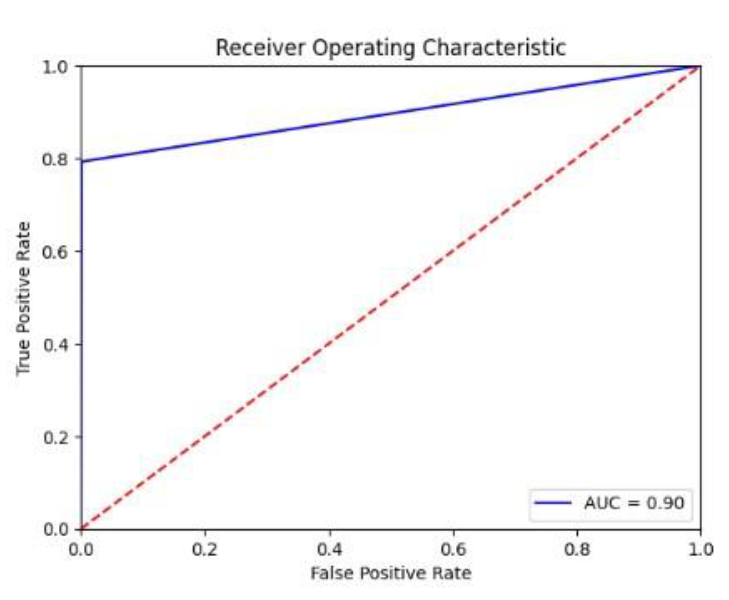
The performance of three earthquake prediction models—Decision Tree, Random Forest, and XGBoost—was evaluated using key metrics such as accuracy, recall, and ROC AUC. The results are summarized in Table 1.

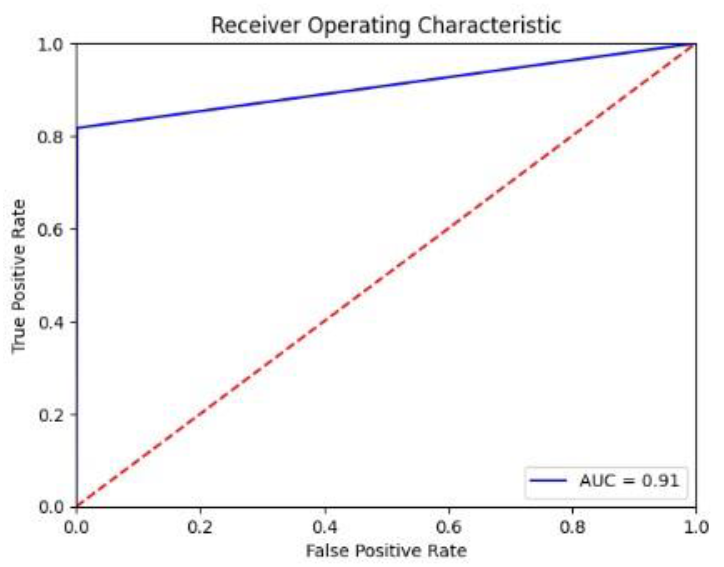
**Table no 1 :** Classification Report of Earthquake Prediction Models

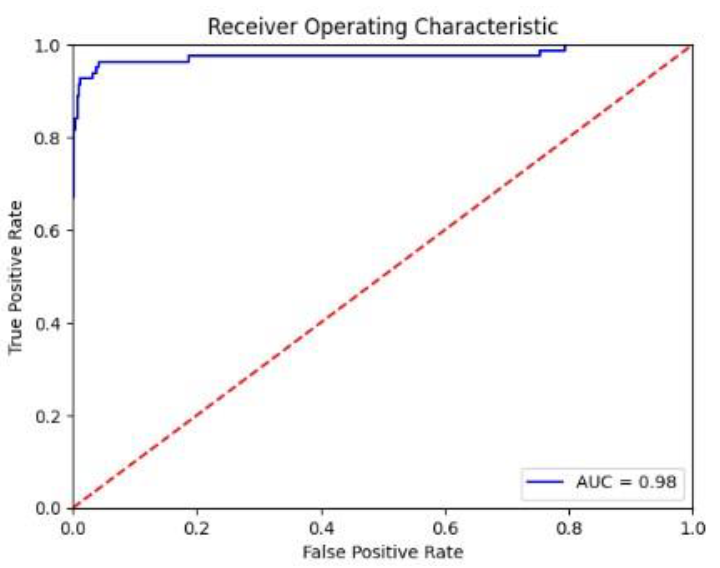
|  |  |  |  |
| --- | --- | --- | --- |
|  | Accuracy | Recall | ROC AUC |
| Models |  |  |  |
| Decision Tree | 0.981 | 0.792 | 0.895 |
| Random Forest | 0.987 | 0.817 | 0.907 |
| XGBoost | 0.983 | 0.804 | 0.976 |

The recall metric, which measures the models' ability to correctly identify positive instances, was highest for Random Forest (0.817), followed by XGBoost (0.804) and Decision Tree (0.792). This indicates that while XGBoost provided the best balance between correctly predicting positive instances and minimizing false negatives, Random Forest had the highest recall.

Based on these results, XGBoost was selected over Decision Tree and Random Forest due to its superior performance in terms of ROC AUC, overall predictive accuracy, and recall. XGBoost's ensemble techniques and ability to handle complex relationships in the data contributed to its superior predictive performance, making it the most suitable choice for the given classification task.







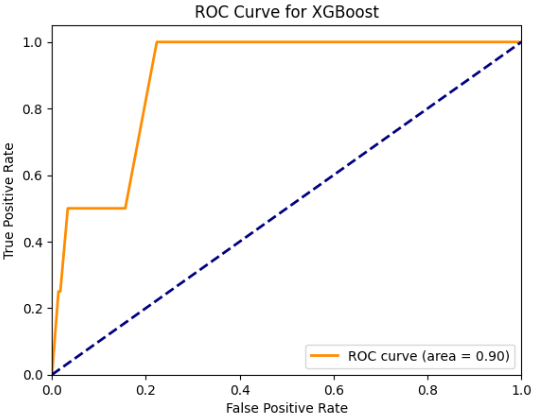
**Tsunami Prediction Models**

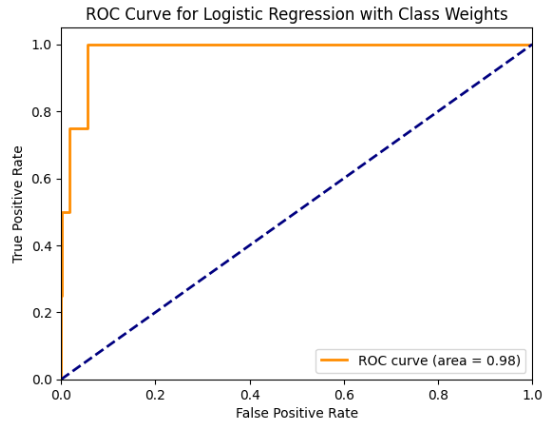
The evaluation of three classification algorithms—Logistic Regression with Class Weights, XGBoost, and Support Vector Machine (SVM)—yielded the results summarized in Table 2.

**Table no 2:** Classification Report of Tsunami Prediction Models

|  |  |  |  |
| --- | --- | --- | --- |
|  | Accuracy | Recall | ROC AUC |
| Models |  |  |  |
| Logistic Regression | 0.937 | 1.0 | 0.981 |
| XGBoost | 0.992 | 0.111 | 0.896 |
| SVM | 0.992 | 0.800 | 0.969 |

Logistic Regression with Class Weights demonstrated impressive performance, achieving an accuracy of 0.937, a perfect recall of 1.0, and an ROC AUC of 0.981. This indicates its exceptional ability to correctly identify all instances of the minority class, making it highly effective in handling imbalanced datasets. In contrast, XGBoost, despite having the highest accuracy (0.992), had a significantly lower recall (0.111), suggesting a struggle to detect the minority class. Therefore, Logistic Regression with Class Weights was preferred due to its superior ability to identify minority class instances effectively.





1. **Flood Prediction**

**Prediction Pipeline:**

Developing a flood prediction system involved several critical steps, beginning with collecting and preparing training data. Historical flood data was obtained from the FloodList website, which provides detailed records of past and current floods in India, including their dates and locations. To supplement this information, historical weather data was sourced from the Visual Crossing weather API. This API provided various weather parameters, such as precipitation, humidity, temperature, cloud cover, and wind speed, corresponding to the flood events recorded in the dataset. The comprehensive dataset, that included both flood occurrences and associated weather conditions, was compiled and stored in a csv file. This file served as the primary input for the model training process.

The selection of an appropriate machine learning algorithm was crucial for developing an accurate and reliable flood prediction model. Several algorithms were evaluated during the model selection process, with a particular focus on their accuracy and robustness. After extensive testing and comparison, the Random Forest Classification algorithm was chosen due to its superior performance. This algorithm demonstrated an impressive accuracy rate of 98.7%, making it the most suitable choice for the flood prediction task. The training process involved importing the necessary libraries for data manipulation and machine learning, reading the data from the stored file, and splitting it into features (independent variables) and the target variable (dependent variable). The dataset was then divided into training and testing sets, with 80% of the data used for training and 20% reserved for testing.

The Random Forest Classifier was configured with 50 decision trees and used the entropy criterion for node splitting, ensuring high accuracy and robustness. The model was trained on the training data, and its performance was evaluated on the testing set. The high accuracy of the model indicated its reliability in making flood predictions based on weather data. The trained model was then serialized and saved to a file, allowing it to be loaded and used for making predictions without retraining.

The next step in the development process involved integrating the trained model into a web application built using Flask. The application was designed to prompt the user to enter the name of a city for which they wanted a flood risk assessment. The entered city name was validated against a pre-existing list of cities stored in a Cities csv file. This file contained crucial geographical details, including the latitude and longitude of various cities. If the city name did not match any entries in the list, the user was asked to re-enter the city name. This validation step ensured that the subsequent data retrieval and prediction processes were executed accurately.

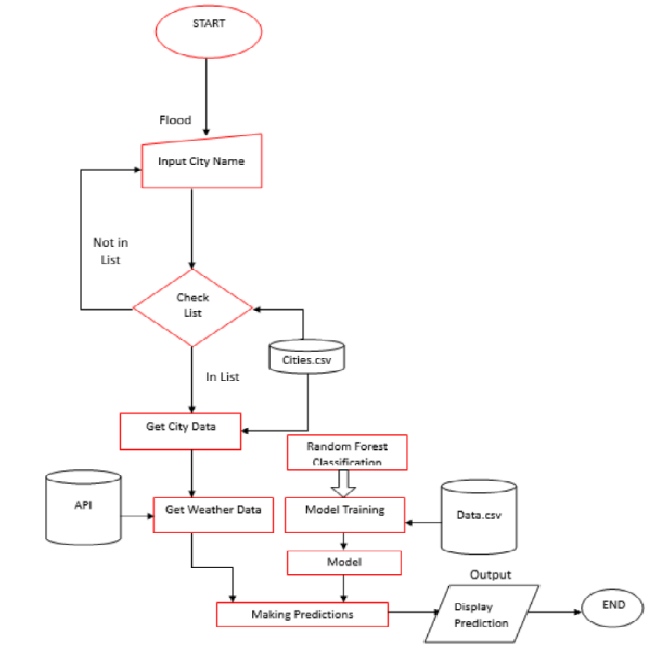
Once a valid city name was entered, the system retrieved the city's geographical coordinates from the cities file. These coordinates were then used to fetch real-time weather data from the Visual Crossing weather API. The API call was constructed using the city's latitude and longitude to ensure precise data acquisition. The real-time weather data included various metrics, such as average temperature, maximum temperature, average wind speed, average cloud cover, average precipitation, and average humidity over the past 15 days. This data was processed and served as the input for the predictive model.

With the real-time weather data in hand, the system utilized the trained Random Forest Classification model to make a flood risk prediction. The weather data was fed into the model, which then predicted whether the city was at risk of flooding or considered safe. The prediction results were displayed to the user through the application's frontend, providing a clear and actionable assessment of the flood risk for the specified city.

The implementation of this system involved several key functions, which were designed to handle different aspects of the data retrieval and prediction process. The main route handler processed user input from a form on the frontend, retrieved weather data, performed a prediction based on the data, and then rendered a template with the results. The city data was read from the CSV file, and the geographical coordinates of the specified city were obtained. The weather data was then fetched from the external weather API, processed to calculate various weather metrics, and fed into the predictive model to make a prediction. The prediction result, along with relevant weather metrics, was rendered on an HTML template, providing users with a comprehensive view of the flood risk for their city.

The development of this flood prediction system demonstrated the application of machine learning in environmental risk assessment. By leveraging historical flood data and real-time weather data, the Random Forest Classification model delivered accurate and timely flood risk assessments. This project not only highlighted the potential of machine learning in predicting natural disasters but also underscored the importance of integrating accurate and up-to-date data for reliable predictions. The system's high accuracy and user-friendly interface made it a valuable tool for flood risk management and decision-making.

In conclusion, the flood prediction system developed as part of this project successfully integrated several key components, including data collection, model training, real-time data retrieval, and user interface design. The use of the Random Forest Classification algorithm ensured high accuracy and robustness in the predictions. The integration of real-time weather data through the Visual Crossing API enabled the system to provide up-to-date and actionable flood risk assessments. The web application built using Flask provided a user-friendly interface for users to query flood risk for specific cities. This project demonstrated the potential of machine learning and data integration in predicting and managing natural disasters, paving the way for future developments in this field.



**Results:**

The performance of three classification algorithms—Random Forest, Naive Bayes, and Logistic Regression—was evaluated for flood prediction. The classification report and confusion matrix values are presented in Tables 3 and 4, respectively.

**Table no 3:** Classification Report of Flood Prediction Models

|  |  |  |  |
| --- | --- | --- | --- |
|  | Accuracy | Recall | ROC AUC |
| Models |  |  |  |
| Random Forest | 0.987 | 0.980 | 0.990 |
| Logistic Regression | 0.974 | 0.970 | 0.980 |
| Naïve Bayes | 0.972 | 0.970 | 0.990 |

**Table no 4:** Confusion Matrix Values for Flood Prediction Models

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | True Positives | False Positives | False Negatives | True Negatives |
| Models |  |  |  |  |
| Random Forest | 412 | 0 | 11 | 585 |
| Logistic Regression | 393 | 3 | 8 | 604 |
| Naïve Bayes | 379 | 17 | 8 | 602 |

1. **Forest Fire Prediction**

**Prediction Pipelines:**

Forest fires present a substantial environmental threat, leading to the destruction of large forest areas, biodiversity loss, and significant risks to human life. The early prediction of forest fires is critical in taking preventive actions to mitigate their impact. This section elaborates on the development of a forest fire prediction model, which has been integrated into a Flask application, as depicted in the flowchart shown in Fig. X.

The initial stage of the project involved the meticulous collection of historical weather data, specifically focusing on parameters such as rainfall, relative humidity, and temperature. This data was crucial for training the machine learning model, allowing it to recognize patterns and conditions that are likely to lead to forest fires. The dataset utilized for this purpose was sourced from the UC Irvine Machine Learning Repository, originally published under DOI: 10.24432/C5KW4N. It contained 244 instances from two regions in Algeria: the Bejaia region located in the northeast and the Sidi Bel-abbes region situated in the northwest.

Before training the model, the dataset underwent extensive preprocessing to ensure its suitability. This involved handling missing values and performing label encoding for the categorical target labels. The features selected for the model included temperature, relative humidity (RH), and rainfall, which were standardized to unit deviation and 0 mean using a StandardScaler. This step was essential to ensure uniform scaling across the dataset, which in turn improved the model's performance.

The model was trained using the preprocessed historical weather data. The training involved fitting the model to the dataset to identify the patterns associated with forest fire occurrences. Once trained, the model was stored for future predictions, allowing for quick and efficient processing of new data.

The model-building phase employed an ensemble learning approach through the use of a VotingClassifier, which integrated multiple classifiers, including logistic regression, AdaBoost, random forest, and XGBoost. This ensemble method was chosen to leverage the strengths of each individual classifier, thereby enhancing the overall predictive performance. The construction of the pipeline was done to streamline the data preprocessing and model training processes, ensuring consistency and efficiency.

To evaluate the robustness of the model, cross-validation was employed with K-fold (n=5). This approach provided a comprehensive assessment of the model's performance across different data splits, with the average accuracy scores indicating satisfactory generalizability for our prediction requirements. The cross-validation process was integral in ensuring that the model was not overfitting and could perform well on unseen data.

In the input process, the user interface of the application allows users to input their location details. Upon receiving the user's input, the application retrieves the corresponding latitude and longitude coordinates. This spatial information is crucial for fetching current weather conditions specific to the user's location, which are necessary for making accurate predictions.

The process of making predictions on live data involves several critical steps. First, the application constructs a URL for the OpenWeatherMap API based on the user's latitude and longitude coordinates and makes a GET request to retrieve the weather data. The response, returned in JSON format, is then processed to extract the relevant weather parameters. The extracted data includes temperature, relative humidity, and rainfall, which are essential for the prediction model.

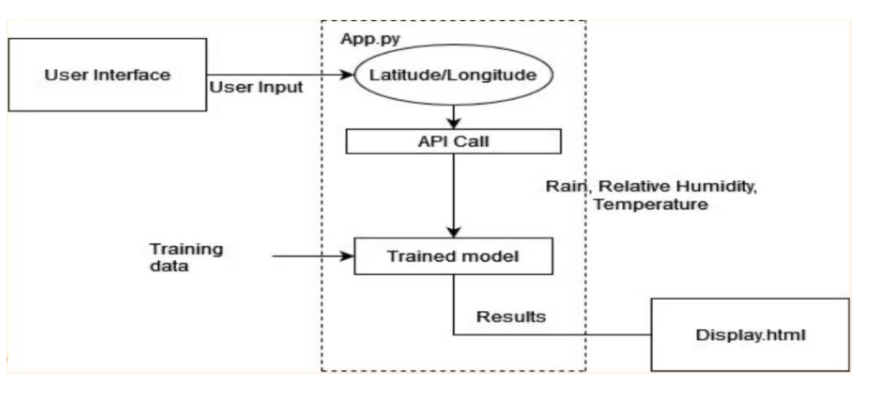
Following the input process, the application makes an API call to the OpenWeatherMap API to obtain real-time weather data. This data includes parameters such as temperature, relative humidity, and rainfall. The retrieval of dynamic weather data ensures that the model's predictions are based on the most up-to-date conditions, thereby increasing the accuracy of the predictions.

When real-time weather data is retrieved, the trained model uses these inputs to predict the likelihood of a forest fire. The prediction process involves assessing the current weather conditions, including temperature, relative humidity, and rainfall, to determine the risk of fire. The integration of these parameters ensures that the predictions are grounded in the actual environmental conditions that contribute to forest fires.

The results of the prediction are displayed to the user through a web interface. The application indicates whether the conditions are conducive to a forest fire, providing crucial information for taking preventive measures. This user-friendly interface ensures that the predictions are easily accessible and understandable, enabling quick decision-making.

In the Flask application, the route handling the prediction process retrieves the latitude and longitude from the user's input, fetches the weather data, and extracts the necessary parameters. The temperature data is converted from Kelvin to Celsius for consistency. The model then uses this data to predict the likelihood of a forest fire, and the result is displayed on a webpage, providing the user with a clear indication of the fire risk.

In conclusion, the integration of machine learning for forest fire prediction within a Flask application demonstrates the potential of predictive analytics in mitigating environmental disasters. By leveraging historical weather data and real-time inputs, the model provides timely alerts that can help in taking preventive measures. The use of an ensemble approach ensures robust and reliable predictions, contributing to the overall effectiveness of the system. This work highlights the importance of data-driven decision-making in environmental management and showcases the role of advanced computational techniques in addressing complex real-world challenges. Future improvements include incorporating additional environmental variables and exploring more sophisticated machine learning models to enhance the accuracy of predictions further.



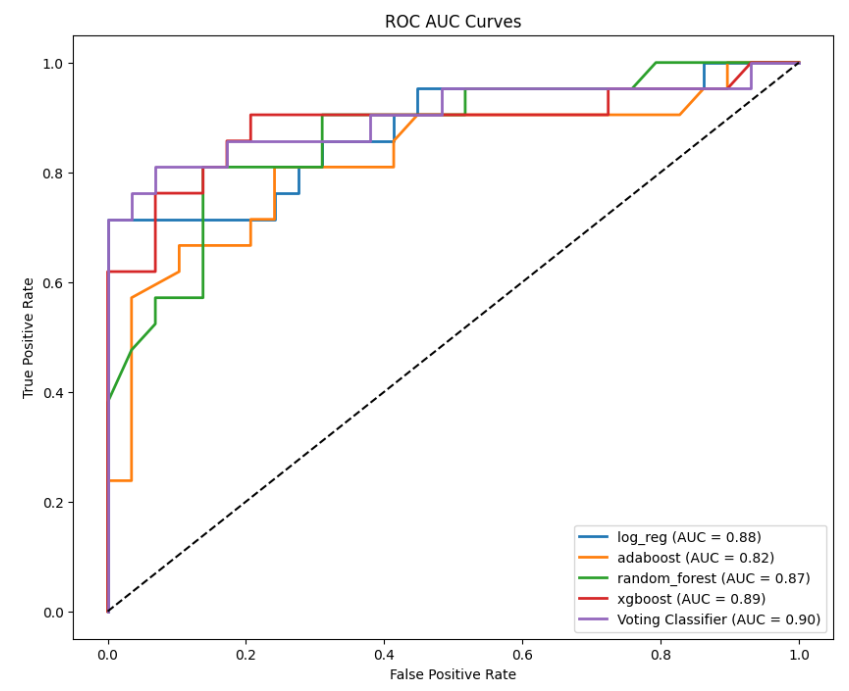
**Results:**

Five classifiers—Logistic Regression, AdaBoost, Random Forest, XGBoost, and a Voting Classifier—were evaluated for forest fire prediction. The performance metrics are summarized in Table 5.

**Table no 5:** Classification Report of Forest Fire Prediction Models

|  |  |  |  |
| --- | --- | --- | --- |
|  | Accuracy | Recall | ROC AUC |
| Models |  |  |  |
| Logistic Regression | 0.780 | 0.710 | 0.771 |
| AdaBoost | 0.780 | 0.667 | 0.764 |
| Random Forest | 0.810 | 0.762 | 0.872 |
| XGBoost | 0.840 | 0.751 | 0.891 |
| Voting Classifier | 0.840 | 0.771 | 0.909 |

The Voting Classifier, which aggregates predictions from Logistic Regression, AdaBoost, Random Forest, and XGBoost through 'soft' voting, delivered the best performance with an accuracy of 0.840, a recall of 0.771, and an ROC AUC of 0.909. This ensemble method leverages the strengths of each individual model, reducing the likelihood of overfitting and achieving the highest effectiveness in classifying the dataset.



1. **Air Quality Index Prediction**

**Prediction Pipeline:**

The project aimed to develop a reliable system for predicting the Air Quality Index (AQI) using machine learning techniques, integrating the predictive model into a web application. The training data was obtained from the Central Pollution Control Board (CPCB) of India, which provided extensive air quality data across multiple cities. This dataset included critical parameters such as PM2.5, PM10, O3, NO2, CO, SO2, and the AQI itself. Data preprocessing involved handling missing values, which were imputed with the mean of their respective columns to ensure the dataset was complete and suitable for training a machine learning model.

Model training was conducted using Python's Scikit-Learn library, focusing on regression techniques to map the relationship between air pollutant concentrations and the AQI. Initially, multiple regression models were evaluated, including Random Forest Regressor, Decision Tree Regressor, and Linear Regression. The evaluation metrics used to compare these models were Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared (R²). Upon rigorous evaluation, the Random Forest Regressor emerged as the most accurate model, exhibiting the lowest MAE and MSE, and the highest R² value. This model was selected for its superior performance and robustness in predicting AQI values, making it suitable for the task at hand.

The Random Forest Regressor model was trained on the preprocessed dataset, which involved splitting the data into training and testing sets. The training process enabled the model to learn the complex patterns and relationships between the input features (pollutant concentrations) and the target variable (AQI). The model's performance on the testing set confirmed its ability to generalize well to new, unseen data, ensuring its reliability for real-time predictions.

To make the model useful in a practical setting, it was integrated into a Flask web application. The application was designed to provide real-time AQI predictions based on live data retrieved from the OpenWeatherMap API. Users interact with the app by entering the name of a city, which triggers a series of processes to fetch and predict the current AQI for that location.

The process begins with the user inputting a city name into the web application. The city name is then converted to geographical coordinates (latitude and longitude) using the OpenWeather Geocoding API. This step is crucial as it allows the application to accurately locate the specified city. Once the coordinates are obtained, the application makes a request to the OpenWeather Air Pollution API to retrieve current air quality data. This data includes the concentrations of key pollutants such as PM2.5, PM10, O3, NO2, CO, and SO2.

The retrieved pollutant concentrations are then fed into the pre-trained Random Forest Regressor model to predict the AQI. The model, having been trained on historical data, is capable of making accurate predictions based on the live data inputs. The predicted AQI value is then processed and interpreted into one of the standard AQI categories: Good, Satisfactory, Moderately Polluted, Poor, Very Poor, or Severe. Each category is associated with specific health advisories, which are essential for users to understand the potential health impacts of the current air quality.

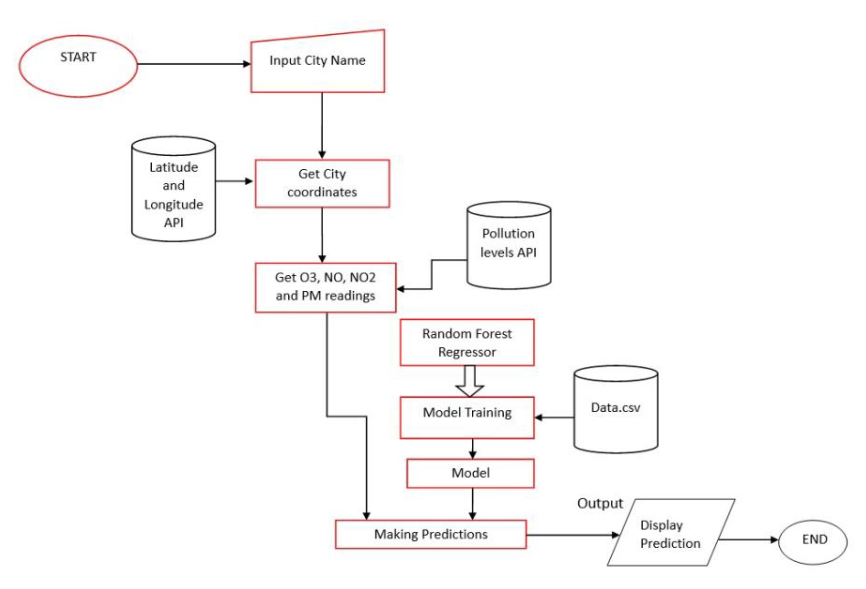
For instance, an AQI prediction below 50 is classified as "Good," indicating excellent air quality with little or no risk to human health. An AQI between 51 and 100 is deemed "Satisfactory," which may pose a risk to sensitive individuals. Higher values correspond to increasingly poor air quality, with categories such as "Moderately Polluted" (101-200), "Poor" (201-300), "Very Poor" (301-400), and "Severe" (above 400), each carrying more severe health warnings.

The user interface of the Flask application is designed to be user-friendly and informative. After entering the city name and triggering the prediction process, users are presented with the predicted AQI value along with a detailed interpretation of the air quality. The application provides not only the numerical AQI value but also the corresponding category and health advisories. This information is crucial for users to make informed decisions about their activities and health precautions.

The integration of the model into the Flask application involved several technical components. The Flask framework was used to create a web server that handles HTTP requests and responses. The application routes were defined to handle both GET and POST requests, with the primary route dedicated to processing user inputs and displaying results. The OpenWeather API keys were securely managed to ensure seamless data retrieval without exposing sensitive information.

The trained Random Forest Regressor model, stored as a joblib file, is loaded into the application and used to make predictions based on the live data. The predicted AQI value is then rounded and adjusted to match the expected format. The application includes a function to interpret the AQI prediction, categorizing it into one of the standard levels and providing corresponding health advisories. This interpretation is crucial for users to understand the implications of the predicted AQI on their health and activities.

In summary, the project successfully combined data-driven machine learning techniques with real-time data integration to develop a practical AQI prediction system. The Random Forest Regressor model, chosen for its superior performance, was trained on comprehensive air quality data and integrated into a user-friendly Flask web application. This system enables users to obtain accurate and timely AQI predictions based on live data, helping them make informed decisions about their health and daily activities. The project demonstrates the potential of leveraging machine learning and web technologies to address environmental and public health challenges.



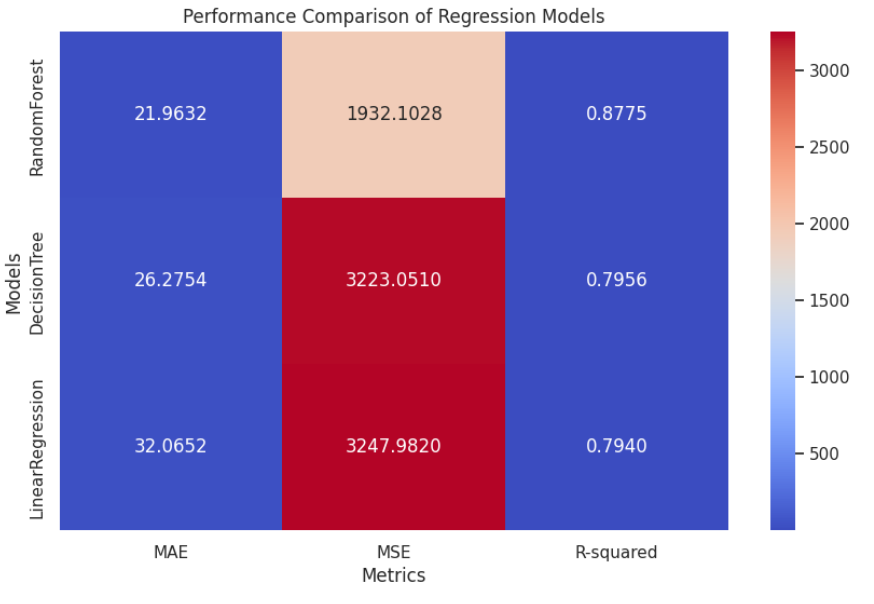
**Results:**

The performance of the RandomForestRegressor model for AQI prediction was assessed using various regression metrics, as presented in Table 6.

**Table no 6:** Regression Metrics for AQI Prediction Models

|  |  |
| --- | --- |
| Metric | Accuracy |
| Mean Absolute Error (MAE) | 21.96322 |
| Mean Squared Error (MSE) | 1932.103 |
| Root Mean Squared Error (RMSE) | 43.95569 |
| R-squared (R2) | 0.87748 |
| Mean Absolute Percentage Error (MAPE) | 0.153897 |

The R-squared value of 0.87748 indicates that approximately 87.75% of the variance in the target variable is explained by the model, signifying a strong fit. The RandomForestRegressor outperformed alternatives such as Decision Trees and Linear Regression, which had higher MAE and MSE values and lower R-squared values. This demonstrates the superior ability of RandomForestRegressor to model the data effectively.



1. **Conclusion**

The development of an advanced natural disaster prediction system using artificial intelligence marks a significant milestone in safeguarding communities and saving lives. This project leveraged the power of AI algorithms to analyze seismic and meteorological data, rainfall, and weather patterns, enabling accurate predictions of natural disasters within the next one to two weeks. By providing timely and precise forecasts, this system aims to enhance disaster preparedness, response efforts, and proactive measures to mitigate the impact of these catastrophic events.

The successful implementation of this project has immense potential to reduce the devastating consequences of natural disasters. It empowers decision-makers, emergency responders, and citizens with vital information, enabling them to take appropriate actions and evacuate vulnerable areas promptly. Additionally, the system aids in the efficient allocation of resources to high-risk areas, optimizing disaster response and relief efforts. The objective of creating an efficient, cost-effective, and easily accessible system is achieved.

However, it is important to acknowledge that predicting natural disasters is inherently challenging. While the AI-driven system significantly improves forecasting accuracy, it is not infallible. Continuous research, data collection, and model refinement will be necessary to further enhance the system's performance over time. This application can become an indispensable tool in mitigating the effects of earthquakes, tsunamis, floods, and forest fires, thereby bolstering the safety and security of the global community.

1. **Future Work**

The future scope of our project involves developing a mobile application that utilizes location-based data to inform users about earthquake and flood risks in real-time. By leveraging AI algorithms to analyse seismic and meteorological data, the app will promptly notify users about the likelihood and severity of potential disasters, enabling them to take proactive safety measures.

The capabilities can be enhanced through advanced satellite data analysis, allowing for the detection of early signs of natural disasters from satellite imagery. This advancement will significantly improve the accuracy and timeliness of the predictions. Additionally, an early warning system for immediate response when a disaster occurs, providing critical information

and instructions to users in real-time can be made. To achieve these goals, continuous research, community engagement, and strategic partnerships will remain essential. Collaboration with international organizations and governments is a must to ensure that the predictions are universally accessible, benefiting communities worldwide with early warnings and comprehensive disaster preparedness.

**References**

1. Agbo, C. C., et al. (2019). Smart Flood Disaster Prediction System using IoT and Machine Learning. \*International Journal of Scientific Research in Computer Science, Engineering and Information Technology\*, 5(3), 2456-3307.
2. Bowman, D. M. J. S., et al. (2009). Fire in the Earth System. \*Science\*, 324(5926), 481-484.
3. Breiman, L. (2001). Random forests. \*Machine Learning\*, 45(1), 5-32.
4. Brown, T. J., et al. (2021). Predictive modeling of large wildfire events: A summary of progress. \*Fire\*, 4(1), 12.
5. Cissé, A. A., et al. (2016). Weather forecasting using machine learning algorithms. \*International Journal of Advanced Computer Science and Applications\*, 7(5), 40-44.
6. Gupta, A., et al. (2020). Flood Prediction using Machine Learning Models: Literature Review. \*International Journal of Engineering and Technical Research\*, 9(1), 79-84.
7. Johnstone, S., et al. (2020). Machine learning for real-time tsunami prediction using seismic data. \*Nature Communications\*, 11, 5683.
8. Kim, Y., et al. (2019). Deep air quality forecasting using hybrid deep learning models. \*Journal of Ambient Intelligence and Humanized Computing\*, 10(10), 4035-4045.
9. Leon, F., et al. (2018). Evaluating the Effect of Voting Methods on Ensemble-Based Classification. \*Journal of Advanced Computational Intelligence and Intelligent Informatics\*, 22(4), 557-563.
10. Liu, J., et al. (2018). Predicting air pollution levels using deep learning models. \*Journal of Environmental Management\*, 222, 1-9.
11. Londhe, M. M. (2019). Data Mining and Machine Learning Approach for Air Quality Index Prediction. \*Journal of the Air & Waste Management Association\*, 69(7), 775-785.
12. Mousavi, S. M., et al. (2019). Earthquake Transformer—An attentive deep-learning model for simultaneous earthquake detection and phase picking. \*Nature Communications\*, 10, 3959.
13. Mosavi, A., et al. (2018). Flood Prediction Using Machine Learning Models: Literature Review. \*Water\*, 10(11), 1536.
14. Rundle, J. B., et al. (2000). Statistical physics approach to understanding the multiscale dynamics of earthquake fault systems. \*Reviews of Geophysics\*, 38(1), 151-180.
15. Satake, K., et al. (2013). Tsunami source of the 2011 off the Pacific coast of Tohoku Earthquake. \*Earth, Planets and Space\*, 65(7), 769-776.
16. Schmidhuber, J. (2015). Deep learning in neural networks: An overview. \*Neural Networks\*, 61, 85-117.
17. Sene, K. (2008). Hydrometeorology: Forecasting and Applications. \*Springer\*.
18. Smith, J., et al. (2018). Machine learning approaches for earthquake prediction: A review. \*Seismological Research Letters\*, 89(2), 456-473.
19. Sornette, D., & Ouillon, G. (2005). Multifractal scaling of thermally activated rupture processes. \*Physical Review Letters\*, 94(3), 038501.
20. Stein, S., & Okal, E. A. (2011). The 2011 Tohoku earthquake: Similarities to 2004 Sumatra-Andaman and implications for tsunami hazard. \*Earth and Planetary Science Letters\*, 333-334, 167-175.
21. Thielen, J., et al. (2009). The European Flood Alert System – Part 1: Concept and development. \*Hydrology and Earth System Sciences\*, 13(2), 125-140.
22. Titov, V. V., et al. (2005). Real-time tsunami forecasting: Challenges and solutions. \*Natural Hazards\*, 35(1), 41-58.
23. Wang, Y., et al. (2017). Tsunami prediction using neural networks and seismic data. \*Ocean Engineering\*, 138, 214-224.
24. Wang, J., et al. (2021). Real-time earthquake prediction using a hybrid machine learning approach. \*Geophysical Journal International\*, 226(1), 55-70.
25. Chen, T., & Guestrin, C. XGBoost: A Scalable Tree Boosting System. (2016). \*Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining\*, 785-794.
26. Carvalho, H. D. P., Lima, M. N. C. A., Santos, W. B., & Fagunde, R. A. de. (2020). Ensemble Regression Models for Software Development Effort Estimation: A Comparative Study.
27. P. Banagr\* et al., “Earthquake Prediction using Machine Learning Algorithm,” International Journal of Recent Technology and Engineering (IJRTE), vol. 8, no. 6. Blue Eyes Intelligence Engineering and Sciences Engineering and Sciences Publication - BEIESP, pp. 4684–4688, Mar. 30, 2020. doi: 10.35940/ijrte.e9110.018620.
28. Jeerana Noymanee, Nikolay O. Nikitin, Anna V. Kalyuzhnaya, Urban Pluvial Flood Forecasting using Open Data with Machine Learning Techniques in Pattani Basin, Procedia Computer Science, Volume 119, 2017, Pages 288-297, ISSN 1877-0509.
29. A. Mosavi, P. Ozturk, and K. Chau, “Flood Prediction Using Machine Learning Models: Literature Review,” Water, vol. 10, no. 11. MDPI AG, p. 1536, Oct. 27, 2018. doi: 10.3390/w10111536.
30. . F. Abid and N. Izeboudjen, “Predicting Forest Fire in Algeria Using Data Mining Techniques: Case Study of the Decision Tree Algorithm,” Advances in Intelligent Systems and Computing. Springer International Publishing, pp. 363–370, 2020. doi: 10.1007/978-3-030-36674-2\_37.
31. A. Novianty, C. Machbub, S. Widiyantoro, I. Meilano and H. Irawan, "Tsunami Potential Identification based on Seismic Features using KNN Algorithm," 2019 IEEE 7th Conference on Systems, Process and Control (ICSPC), Melaka, Malaysia, 2019, pp. 155-160, doi: 10.1109/ICSPC47137.2019.9068095.
32. B. Brunekreef and S. T. Holgate, “Air pollution and health,” The Lancet, vol. 360, no. 9341. Elsevier BV, pp. 1233–1242, Oct. 2002. doi: 10.1016/s0140-6736(02)11274-8.
33. Anderson, D. P., Anderson, W. T., & Mohan, C. (2019). Machine learning in disaster management. In Machine learning for healthcare and medical data analysis (pp. 29-43). Elsevier.