VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belgaum-590018

A PROJECT REPORT

ON

" FLOOD PREDICTION USING RAINFALL ANALYSIS"

Submitted in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF ENGINEERING

IN

INFORMATION SCIENCE AND ENGINEERING

Submitted by

Sruthi Prathapa	**1CR21IS116**
Prerana Anand	**1CR21IS120**
Navya Narayan	**1CR21IS190**
Krutika K Naik	**1CR22IS406**

Under the Guidance of

Prof. Rakesh Kumar

B.Tech, M.Tech

Assistant Professor, Department of ISE, CMRIT

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

CMR INSTITUTE OF TECHNOLOGY

AECS LAYOUT, ITPL PARK ROAD, BENGALURU - 5600372023-24

CMR Institute of Technology
AECS Layout, Bengaluru-560037
Department of Information Science and Engineering

CERTIFICATE

Certified that the project work entitled **Flood prediction using Rainfall Analysis** is a bone fide work carried out by

Sruthi Prathapa	**1CR21IS116**
Prerana Anand	**1CR21IS120**
Navya Narayan	**1CR21IS190**
Krutika K Naik	**1CR22IS406**

in partial fulfillment for the award of Bachelor of Engineering in Information Science and Engineering of the Visvesvaraya Technological University, Belgaum during the year 2024-2025. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report deposited in the department library. The project report has been approved as it is satisfied the academic requirements in respect of project work prescribed for the said Degree.

Prof. Rakesh Kumar
Project Guide

Dr. Jagadishwari V
Head of the Department

Viva

Name of the Examiner	Signature with date
1.	
2.	

ACKNOWLEDGMENT

The satisfaction and euphoria that accompany a successful completion of any task would be incomplete without the mention of people who made it possible. Success is the epitome of hard work and perseverance, but steadfast of all is encouraging guidance.

So, it is with gratitude that I acknowledge all those whose guidance and encouragement served as beacon of light and crowned our effort with success. I would like to thank **Dr. Sanjay Jain**, Principal, CMRIT, Bangalore, for providing an excellent academic environment in the college and his never-ending support for the B.E program.

I would like to express my gratitude towards **Dr. Jagadishwari V**, Professor and HOD, Department of Information Science and Engineering CMRIT, Banga-lore, who provided guidance and gave valuable suggestions regarding the project.

I consider it a privilege and honor to express my sincere gratitude to our internal guide **Rakesh Kumar, Assistant Professor** Department of Information Science and Engineering, CMRIT, Bangalore, for their valuable guidance throughout the tenure of this project work.

I would like to thank all the faculty members who have always been very cooperative and generous. Conclusively, I also thank all the non- teaching staff and all others who have done immense help directly or in- directly during our project.

Sruthi Prathapa
Prerana Anand
Navya Narayan
Krutika K Naik

ABSTRACT

Floods are one of the most destructive natural disasters affecting life, property, and structure. Effective disaster preparedness, resource management, and mitigation planning against fluctuating rainfall and dynamism surrounding climate change necessitated prompt, accurate, and timely forecasts of floods. Flood Prediction Using Rainfall Analysis" is proposed as a new project that shall develop a system of predicting rainfall with good reliability and strength using advanced analytical techniques with aid to assess the risk of flooding.

The system also analyzes large historical datasets of patterns of rainfall in the area. Background data undergoes cleaning and normalization and categorization into useful classifications on some recognized guidelines that make it easy to use or interpret. To train the model well and for evaluation, this dataset is then partitioned between a 70% training set and a 30% testing set. It integrates historical data and presents meteorological inputs to forecast a possible flood risk inside specified locality and over a specified period, mainly during the monsoon season.

Testing and validation confirm the system with support like high precision and safe confirmation from a performance monitoring capacity such as RMSE and MAE with a mixture of confusion matrices for predictions. The other major point in this system is that it has a modular design, and this can lead to further scaling up, adding an additional scaling layer which allows for seamless integration of multiple data sources or predictive models.

This project further develops on existing systems by offering location-specific forecasts from a user-centric angle. After entering the location-specific data, actionable forecasts can be retrieved through intuitive visualizations and reports. This will empower stakeholders to customize their predictions according to the specifics of a region and time frame, thereby enabling authorities, planners, and communities to better manage the risk of flooding.

Contents

List of Figures

1 PREAMBLE

1.1 Introduction

Weather forecasting & risk avoidance are very much a necessity these days as a result of continually mounting risks from life & property damages through natural disasters. Among these disasters, floods have caused quite a great amount of hassle in the matter of infrastructure, human lives, & properties. It has increased rapidly due to rising urbanization. This happens to be a basic necessity for designing a model which is strong enough to give correct & timely predictions about floods.

Our final year project is the development of an advanced flood prediction model using Artificial Intelligence & Machine Learning techniques. The model can handle large datasets with deep learning algorithms & even spatial patterns of flood risk, but at the same time, it predicts the probability & intensity level of rainfall within the same geography.

It is a model of historical data that conducts a temperature analysis in the geographical area, understands hydrological patterns, & gives an accurate prediction of rainfall & flood risk. This model is basically aimed at developing an intelligent system that can process real-time data on rainfall, temperature, & other relevant factors & provide a prediction based on the same.

The model fetches historical data, with data gathered from government datasets, old scientific publications, or even news archives. All the relevant information needs to be gathered from trusted sources since the model is used for predicting a real-time calamity. A machine learning algorithm tests the accuracy of the model & ensures that it is reliable.

It is a model using data of the last 10 years to make the appropriate predictions about the future using recorded data. Relevant data fetch to the desired year is carried out using the Open Mateo, an open-source API; using the fetched relevant data for predicting the answer of the model to the user question.

1.2 Existing System

Google's flood prediction initiative employs artificial intelligence and machine learning techniques to predict riverine flooding up to seven days in advance. The system learns from its big datasets every day, thus increasingly gaining flowering accuracy over time and capturing climate adaptation changes.

This comprises two principal AI models.

- Hydrologic Model: It predicts water in rivers based on meteorological data, topography, and historical river behavior. The prediction is then fine-tuned by real-time rainfall and river gauge data to provide an early and more accurate alert.
- Inundation Model: It predicts the areas that will be inundated as well as the expected water heights under one particular forecast of hydrology. Such geospatial analysis makes it possible to estimate the extent of flooding, allowing agencies to take targeted preparedness measures.

This scheme has so far been used in over 80 nations to provide early warning for mitigation against damages induced by floods. Google cooperates with government and local agencies to disseminate the alerts through mobile notifications, search, and maps, ensuring maximum communications penetration and impact.

1.3 Drawbacks

- Data Limitations: The accuracy emanates from the quality as well as the availability of input data; regions without this comprehensive hydrological and meteorological data have predictions that might lack reliability.
- Generalizability of the model: In the system, covering vast territorial spaces, such distinctiveness among regions concerning terrain, land cover, and structures may not always be well captured, which assures prediction accuracy of a likelihood.
- Communication challenges: In the midst of the correct predictions, the effectiveness of the warnings distributed to the communities at risk may be different. At times, the warnings may not reach the at-risk populations in time to ensure proper governance on their part.

The next challenge is channeling resources. Such deployments and maintenance of sophisticated artificial intelligence systems require an enormous whack of computational power and technical expertise, which becomes one of the major hindrances for some resource-poor regions.
Such challenges may be surmountable through building up systems for data collection models applicable to the local context; improving communication strategy; and providing equitable access to technological means.

Encouraging effective collaboration between government and private individuals concerned with establishing improved data sets increases model quality. Investment in mobile access and internet penetration will additionally bridge the gap to vulnerable communities through timely warnings on flood risks.

1.4 Problem Statement

Floods are the most destructive of all natural calamities. These floods cause enormous damage to life, property, & infrastructure. In areas with seasonal rainfall, it is vital to predict rainfall patterns & pinpoint flood-prone areas to avoid damage caused by floods. Presently, the flood forecasting system uses less data or does not give accurate information on localized flooding, especially in areas where there is incomplete or inconsistent meteorological data. The need for a more robust, reliable, & scalable system for predicting rainfall has gained much urgency following current climate change & massive urbanization. This project is focused on filling this gap by developing an advanced system that uses rainfall data for the prediction of flooding occurrences with high accuracy in terms of very early warnings that can be given to affected communities as well as authorities involved.

1.5 Objective of the Project

The aim of this project is to develop a Flood Prediction System that is based on Rainfall Analysis. The mentioned Flood Prediction System will take rainfall data from previous years and mash it with the working machine learning model to ascertain the monthly rainfall and risk of floods for any specific geographic location. It does the following:

- Analyses the historical rainfall data to detect trends and patterns.
- The use of an LSTM-based deep learning model for forecasting upcoming rain.
- Forecasting flood risk classification on a rainfall basis with Random Forest.
- Interactive User Interface, where the user feeds in latitude, longitude, and year to get the forecasted rainfall and flood risk estimation
- Alerts on flooding to decision-making authorities & the concerned regional authorities with the predicted rainfall.

This project will serve greatly in preparing any region for flood prevention in regards to data-driven analyses and predictive capabilities.

1.6 Proposed system

Resulting from the above preprocessing measures of rainfall data focused on fetching data from the Open-Meteo API, the data is then categorized into rainfall classes according to IMD standards, with 70% set aside for training and another 30% automatically assigned for model testing.

Long Short-Term Memory (LSTM) is used for rainfall prediction because of its effective handling of time series data. The model is to be trained for rain prediction between 1979-2013 to predict rains for the succeeding monsoon season and this next year ahead, whilst the performance of models will be assessed using the confusion matrix for proper prediction.

The use of a Random Forest model could calculate flood risk by studying rainfall predictions and ascertaining areas susceptible to flooding. Hence this takes better management decisions in preparing for an impending flood.

Since most of the rainfall across India is accounted for between June and December, data limited to these months are retained for experimentation. Though it can be reasonably inferred that a whole year forecast for a particular location might be more likely to yield more accurate results on account of proper differentiation of non-rainy days, efforts are still based on development of rainfall prediction and thus potential due to floods during critical monsoon months.

The work recommends the use of a data-driven approach employing deep learning and ensemble learning methods for better predictability and reliability of accuracy in contrast to the traditional heuristic and rule-based model of flood forecasting. The suggested method appropriately consolidates the time-dependent nature of the rainfall pattern and employs a Random Forest that integrates individual attributes governed by the insights derived from multiple decision trees in assessing flood risks of predicted intensity. Predictive location-based prediction will provide timely

and reliable information for proactive disaster management by the authorities. Further works could include other influence factors such as soil moisture and river water level that might add refinement to the assessment of flood risk and enhance the robustness of this system.

1.7 Plan of Implementation

The implementation process for this project, "Flood Prediction using Rainfall Analysis," is done within a four-step structure that consists of data collection, preprocessing, model training, and deployment. The project implementation shall be done in the following six steps:

1. Data Collection:

- Downloading historical rainfall for different locations through Open-Meteo API.
- Cleansing and preprocessing of daily rainfall data for further aggregation into monthly rainfall.
- Geolocation services such as Nominatim must be used to correctly map inputs of latitude and longitude.

2. Data Preprocessing:

- The raw rainfall data needs to be processed and structured using the Dataframe library Pandas and NumPy.
- Handle negative values, negative outliers, and conflicting data.
- Feature scaling is performed to normalize the data for use in machine learning models.

3. Model Development:

- To build a model for rain prediction using LSTM-based deep learning.
- To build a model for floods using Random Forest-classified algorithms to measure the risk of flooding.
- Trained models to be saved for later predictions using Joblib.

4. System Integration:

- Web UI for user input and visualizations - built in Streamlit.
- Session states that hold latitude, longitude, and predicted rainfall values.
- The navigation sidebar for easy access to the features of the application (Location Input, Rainfall Prediction, Flood Prediction).

5. Prediction & Visualization:

- Forecast and historic monthly rainfall data in tabular format.

- Real-time flood risk estimation using the forecasted rainfall.
- Improvement of the user-friendly interaction through graphical presentations like graphs and data tables.

6. Model Deployment:

- Deploy the application on Streamlit Cloud or Heroku to access the web
- MODEL files must be included in the deployment: rain_prediction.pkl, flood_prediction.pkl, scaler.pkl.

7. Testing & Validation:

- Compare predictions with historical real rainfall data to ensure model accuracy.
- Sample should have a bigger size to avoid edge cases: invalid latitude/longitude.
- Improved ability of prediction through hyperparameter tuning on the model.

8. Future Enhancements:

- Add another weather parameter set consisting of temperature and humidity to the data set.
- Discuss alternative machine learning techniques that could enhance the model.
- Deploy alert systems before floods, using real-time data on rainfall amounts.

2 LITERATURE SURVEY

Flood prediction and management have been correlated with the impacts of natural disasters for many years. Many research studies have been carried out in enhancing the precision of the flood forecasting model with the help of advanced techniques, including satellite data, artificial intelligence, machine learning, and geospatial data analysis. In this section, three prominent flood prediction systems have been discussed along with their methodologies and technical features: Global Flood Monitoring System, FloodAI, and the flood prediction model of Kyoto University.

2.1 Global Flood Monitoring System by NASA

GFMS is the satellite-based flood prediction system offering flood monitoring globally at a wide scale. Satellites, by means of observing hydrological events and using them for hydrological modeling, detect floods in real time and generally focus on the vast geography scale. Satellite data provided by the GFMS enable it to monitor floods worldwide on an uninterrupted basis with much accuracy over broad areas in predicting floods. It makes use of some source of remote sensing in collecting its data, which may be relevant for the risk analysis of flooding in some areas with scarce data in level ground.

This system is based on flood estimates, and it has used hydrological models that simulate the flow of water across the regions. Water models depend upon the satellite images and hydrological conditions to identify places prone to hazard. GFMS is of high level, large scale, and institutional usage; therefore, it is very helpful for regional or governmental organizations. However, in terms of providing more localized and specific flood risk information at the level of individual locations, it may be limited by its scale. Nonetheless, despite being exceptional in broad coverage, GFMS does not allow location-based parameters like latitude and longitude, which can allow better specificity of flood prediction.

Unlike that, this proposed system looks to user-centric personalisation so that flood risk analysis can include latitude and longitude as inputs thus responding more particularly to local demands. The accommodating ability of this system in accommodating the inputs for the users themselves is what also makes it open to smaller firms and individuals more readily.

2.2 FloodAI by Google

It is a flood prediction project by Google, using machine learning and artificial intelligence to give good and real-time flood predictions. It is a project of providing actionable flood alerts, mainly for vulnerable communities in riverine areas, hence utilizing advanced AI models that comprise deep learning networks using meteorological and geospatial data for flooding prediction purposes. FloodAI is specifically concentrating its focus on riverine flood risks and seeking areas where the accuracy of the forecast may impact preparation and response strategies.

The use of artificial intelligence helps FloodAI in analyzing complex patterns derived from historic weather data, therefore giving accurate and responsive flood predictions. The model is more directed towards providing timely warnings about immediate risks of floods.

This model gives critical short-term forecasts but the system does not comprise long-term predictions like seasonal or future rainfall patterns.

Unlike that, the model above uses LSTM networks for the forecast of future rainfalls. Such a lengthening in the forecast would help a user understand risks surfacing long term within flood situations. Further, open-source APIs like OpenMeteo make its deployment global where less or no weather data access can be available with some other source.

2.3 Kyoto University Flood Prediction Model

It makes use of machine learning techniques, specifically LSTM, on the prediction of rainfall and the risk of floods. It tries to refine its predictions using geospatial and meteorological data to make the accuracy of the flood forecasts better. The system considers parameters like topography and historical weather patterns to generate the given flood risk assessments. It offers a flood likelihood-based approach with predispositions toward historical data that could be used as the primary tool for preparation and mitigation.

The Kyoto University model makes use of pre-set datasets from credible meteorological as well as geographical sources for better designing of risk maps within flood-prone areas. The approach generally deals with a predefined dataset as well as geospatial data that limits its flexibility for final users in regard to customization. This model has proven effective in forecasting floods if it actually takes into consideration past data, but the model can't forecast annual predictions or long-term forecasts because apparently, it depends on the past for consideration. This proposed system will be unique since it shall take specific input in terms of latitude and longitude and therefore has a more adaptable approach towards the prediction of floods. Furthermore, instead of using data on past histories for flood predication, the system predicts upcoming patterns of rains and, in that sense, is a proactive type of system against the flooding risk management.

The proposed flood prediction project from these systems yields the following features, adding to its flexibility and applicability. While the models like GFMS and FloodAI and Kyoto University's model are highly successful in their domain, the inclusion of LSTM in future rainfall predictions extends beyond the immediate flood risks with which these models are concerned. This will come in handy while handling floods on a proactive manner, which models such as FloodAI and GFMS that focused on real time do not, currently. The system also ensures a user-focused approach and one can make it customize any forecasts on given spatial coordinates or intervals. This is in contrast to the approach of GFMS, which emphasizes general regional prediction and the approach by Kyoto University, which will have to depend on predefined data for flood predictions. The significant advantage of this proposed system, therefore, remains in the prediction of floods on a local level; the input for location will be allowed while customizing the year for prediction. This would add more value to those kinds of users that are not institutional such as local organizations and individuals.

Open-source APIs like OpenMeteo will scale the system towards relevance and applicability geographically, a feature that helps in adoption in different geographies and diverse data conditions under which it works, a differentiation from other models that may depend on localized availability of data.

GFMS, FloodAI, and a model developed by Kyoto University have been very essential in the building of flood prediction. Each one has its characters and strengths with applying the different data from satellites plus AI on top of machine learning algorithms for predictions of floods with risk levels of assessment. This system builds on past efforts: it emphasizes more localized, user-driven predictions which can now, of course, try to predict long-term rainfall patterns. This is where personalization, long-term prediction, and suitability around the globe make the suggested system feasible and worthwhile for flood risk management within different regions.

3 SYSTEM REQUIREMENTS SPECIFICATION

3.1 Functional Requirements

1. Improvement of Collection Data

The application uses the Open Meteo API to collect historical & current weather data to keep it real & sure about the inputs. In this step, there is data preprocessing, which collects all the oncoming daily rainfall & adds it up during the month as required to form a uniform structure & fill holes or gaps in such data using interpolation techniques. All inputs normalized range set into a range of 0-1 compliant to machine learning. This normalizing is done using latitudes & longitudes & year & month, among other time-derived attributes. This data will first go through pre-processing & stored in memory for a period, in order that crossing of various pages may be efficiently done, hence increasing efficiency & the user experience.

2. Training Models

There are two different models that form the system on which one relies: the Rainfall Analysis Model based on the LSTM model, primarily aimed at time series forecasting, & the Random Forest classifier for flood risk prediction. These models are fed by credible data sets, so it ensures accurate prediction & generalization. Training includes other iterative processes, such as cross-validation to avoid overfitting & underfitting. Techniques of data augmentation are applied on the original training dataset to make it richer, & hyperparameter tuning also makes the model more refined. As a result of training, databases of weights that are applied on the models are stored for a smooth & seamless integration with deployment.

3. Prediction

The predicting mechanism of the application is historical & current alongside future ones. Rainfall is forecasted based on LSTM, while flood risks are predicted using the Random Forest model in terms of their risk classifications with the attributed predicted rainfall data. Predictions are also reflected dynamically based on user inputs, such as geographic coordinates & the year desired. The past years have monthly rainfall summaries available, & the current year has next month predictions derived from November data-the results are represented in interactive tables, with visual indicators for easy use, like using colors to present risk levels.

4. Evaluation

Includes testing procedures related to the reliability & accuracy of the models. The foundation of the validation for prediction is given by historical data of floods based on metrics that include precision, recall, & F1-score analysis. Based on the comparative study with current models, reliability is established of the derived approach. The predictive strength can also be measured based on user suggestions over what needs improvement. By regularly backtesting models using past data, it can monitor prediction robustness. The evaluation results guide supplementing models & keep them optimized. Such improvement ensures the system continuously conforms to performance & accuracy standards.

5. Fine-tuning

Fine-tuning is of utmost importance in the adaptation of the system & is mainly done on the level of hyperparameters for the LSTM as well as the Random Forest models. Grid search & random search are examples of systematic exploration of parameter configurations. Other enhancements that extra climate variables provide are an expansion of the models & their accuracies. It also removes outliers in data & leads to error reductions. Periodic updates involve feedback & insights from performance evaluations on how accurate & relevant for specific dynamic scenarios the models remain. With this, continuous integration practices normally speed up the process of integrating such improvements.

6. Deployment

In this case, deployment refers to the strategy using Streamlit as the interactive & very user-friendly front-end interface. The back-end APIs provide straightforward data handling & processing, & the models are developed integrated in serialized formats. Introduces & still uses the current system where versioning & version control are all done in the pipeline, so it's intended to capture all the modifications within the system with the personality of having the system intact. Features with real-time processing which add to the responsiveness of the whole system, so prediction can be very soon done. Error-handling mechanisms with fallback options ensure the continuation of operation, and the use of options for cloud deployment would allow resources to be scaled as needed to cope with increased or decreased demand.

7. Multi-modal Support

The program has supported inputs in several different modes including latitude & longitude pairs & temporal inputs for global outputs. Model for rainfall & flood predictions were integrated into the same system as it could provide predictions for site-specific or broader areas. It is so designed to sometimes run under years that are either in the past, present, or future while its predictions adapt dynamically based on user input. All such flexibility ensures context relevance while its modularity opens windows to add in new data sources or new modalities as they may be necessary.

8. User-Friendly Interface

The system has simple easy & clean designs for better usage in interaction. It offers a variety of interactive pages that allow the clients to easily transition from data input to rainfall prediction to flood risk outputs. The results are visually attractive using color-coded indicators & even in structured tables. The session persistence saves user's inputs from one page to the next so that workflow is improved. Customization with light & dark theme options allows convenient user comfort. Sidebar navigation & responsive designs offer access to the systems through devices & platforms, hence making an appealing interface that is functional & visible.

9. Feedback Mechanism

Integrating a solid feedback system provides for ways through which users can report inaccuracies or suggest improvements via the application itself. The data is stored & analyzed to refine models & application logic to enable continuous improvement. Active learning techniques employ user feedback to retrain models using edge case examples. Furthermore, the system can reveal the weaknesses of the system or highlight potential areas for improvement, thereby increasing user satisfaction. Implementation of user input promptly keeps applications in relance & thus their reliability in most applications.

10. Security and Privacy

Very important is privacy & security, & they will be addressed by API connections, security, & in-memory data handling to safeguard the user's identity. Even geographical coordinates will be processed in such a way that temporary data removal will happen without a permanent archive as part of privacy. Access controls will also bar against unauthorized interaction, & methods of anonymization will improve privacy. Security audits will be conducted regularly, & application updates will be done whenever it's necessary in regard to security vulnerabilities. These above methods protect user data & retain their trust.

11. Scalability

This application can accommodate incoming & increasing user traffic & storage of bigger datasets; performance is ensured not to degrade. Its scalable architecture allows for easy integration of new models, APIs, or functionalities. Cloud-based deployment options ensure dynamic allocation of resources nationwide & flexibility during high demand peaks. Modular design enhances ease in updating & extending systems while smooth operation in diverse use cases is assured by efficient data handling mechanisms. It allows the system to expand & accommodate the changing needs of users.

3.2 Non -Functional Requirements

1. Performance

Real-time predictions are optimized & made at the lowest possible latencies by the system. These are concerning data fetching & model inference processes. Periodic performance tests are designed to identify bottlenecks & smoothen the pipeline. Caching results accessed frequently saves redundancy & accelerates response time. Application uptime shall be monitored closely to maintain its availability. Collectively, these efforts ensure that such an application meets high-performance standards in service for all users.

2. Scalability

The low latency helps the system to optimize predictions in real time. These concern data fetching & model inference processes. Periodic performance tests are designed to identify bottlenecks & smoothen the pipeline. When results are accessed frequently, it saves redundancy & accelerates response time. It will be important to monitor the uptime of an application so that its availability is maintained. Collectively, these efforts are made sure to ensure such an application meets certain high-performance standards in service to all the users.

3. Reliability

Reliability is ensured by the introduction of stringent identification mechanisms of errors & regular testing for edge cases along with fallback modes should a disruption occur. Redundant systems provide the support of high availability while accuracy of prediction is kept through regular cross-validation of the model. This detection is also done in real time to respond to such promises toward a much more dependable user experience.

4. Usability

User accessibility has been prioritized, with features like voice input & keyboard shortcuts for seamless navigation. Data security measures, including encrypted storage & secure API requests, have been implemented to protect user information. Additionally, the application is scalable & modular, allowing for future enhancements such as integration with real-time weather monitoring systems.

5. Security

Data security is maintained through end-to-end encryption, which ensures that sensitive information is protected during both storage & transmission. A secure & managed approach is implemented for handling API keys & authentication tokens, reducing the risk of unauthorized access. Role-based access controls (RBAC) ensure that only authorized users can access specific functionalities, while multi-factor authentication (MFA) adds an extra layer of protection. Session expiration & automatic logout mechanisms prevent the unauthorized use of an active session, thereby preventing potential security risks. Regular software updates, security patches, & vulnerability assessments are also conducted to update emerging risks & strengthen system defences. Periodic security audits, compliance with industry standards such as GDPR & ISO 27001, & real-time threat monitoring help maintain a secure & trustworthy environment for the users.

6. Maintainability

This maintainability is achieved through a well-documented modular codebase, which can allow update & integration to be done easily. With proper comments as well as version control, developers will appreciate the understanding & modification of the system. Test-driven development reduces the possible risk of unintended disruptions. Additionally, automated deployment pipeline ensures simplified conditions for updates making the application always adaptable & efficient.

3.3 Product Requirements

The product is the portfolio of customers, predicting flood risks accurately without geographical location. It is highly dynamic & visually engaging, with high appeal. Modular architecture with secure APIs has been integrated to provide for future growth. It further provides features of feedback on the input from the customer, real-time visualization, & scalable back-end systems that enhance the functionalities of usability. Together, they give a very strong & customer-centric solution.

- Operating system: Any OS
- Processor: Intel, AMD, or I3 or above processor
- IDE Required: VS Code
- Programming Language: Python
- Run time Environment: Python

4 SYSTEM DESIGN

4.1 System Development Methodology

The indexed application is robust & adaptable to two purposes-predictive accuracy & user satisfaction. In the design, there is a focus on modularity so that new features can be integrated without problems, new data sources, or better models, for example. The clear separation among the main tasks-data handling, prediction, & visualization-insures efficiency in processing & maintenance. It is built considering scalability, such that the system should be able to handle increased users & large datasets but with no deterioration of its performance.

Extensive documentation backs the developer to understand how these different components interact, thus making debugging & updating easy. Security considerations are also part of the design for the purpose of user data privacy & system integrity. It has applied Agile methodology for the flexibility to respond very fast to the change in the requirements.

Every sprint can introduce a new component of the application upfront, whether it is a data preprocessing pipeline, user interface development, or model integration, for progressive construction. Other things like collaboration among developers, data scientists, & UI/UX designers would ensure that the products remain aligned with the needs of the users & the goals of the project. Retrospectives conducted often allow the team to refine the process, detect bottlenecks, & increase their productivity.

It would be ideal if they can find & correct process lapses that cause low performance. Agile's nature, using an iterative approach, still enables changes at the last stages but with some minimal disruptions. It maintains the momentum without causing burnout because unit work breaks down into small but consistent bites. It consists of logical organization & smooth operation in a three-layer construction instead applicable to the system.
The data layer contacts & connects with external APIs & processes raw weather data to create structured input for analysis. The processing layer contains machine learning models, or it has orchestration mechanism to route proper data into each model.

Therefore, the presentation layer displays results to users in accessible formats through tables, graphs, & colour codes alerts. Middleware is the alternative for securing the efficient communication between layers against direct API reliance cache mechanism for frequently accessed data. The system is set up with redundant systems in backend to ensure uptime against down time as well as high availability. This architectural modularity brings about the adaptability of making this flexible if there are new requirements in the future like the addition of new predictive models or integration of advanced visualization tool.

4.2 Project Structure

The most accurate organization of the whole project under direct heads has been used to achieve a clear understanding of the tasks & its maintenance. The Data folder contains training datasets & scripts for acquiring real-time data. The Model's directory is subdivided into subdirectories for serialized models & configuration files for easy updating & versioning.

The scripts here, in this Utils folder, would be the data transformations, requests to the APIs, while in Config files save global variables & settings in a uniform behaviour of the different environments. It also contains templates & stylesheets within the UI folder to ensure the proper visual code. The Build directory contains automated build scripts for managing the process of deployment. Detailed logs & test cases support debugging & verification.

4.3 Implementation of the project

As the project is set up in phases, the first activities begin with data sourcing and pre-processing. The team has also initiated application programming interfaces to access data and error handling should the connectivity be interrupted or data not exist. The machine learning model received validation along with export of the serialized weights for use by the application. The construction of the UI via Streamlit is expanded to include user feedback and incorporates repeated changes in terms of functionality and aesthetics. Care is also needed for implementing on cloud, as can be seen in its need for containers and orchestration tools such as Docker and Kubernetes for creating scalable applications. Integration tests conducted have shown that the modules have been able to communicate with each other smoothly. There is real-time monitoring of the performance through the use of monitoring tools. Code reviews and code documentation happen at regular intervals so that the entire process is transparent and quality assured.

4.4 Technology Feasibility Report

A feasibility report is the analysis & justification of the technology considered. In this case, extensive weather dataset has been chosen from Open Meteo API because they are essential in the building of accurate predictive models. Another chosen technology is a Python supplemented by TensorFlow, scikit-learn, & Pandas which provide huge libraries to handle preprocessing, modeling, or analyzing flexibly easily. The user also chose Streamlit as it does not require heavy technical knowledge to develop interactive dashboards quickly. Among other technologies, it narrowed down to AWS since it is a cloud platform with scalability, cost efficiency, & global reach. All the above technologies ensure the viability of the project, providing for its technicality balanced with performance versus cost ratio. This report also goes on to evaluate alternative tools while affirming the superiority of the chosen stack for the specific needs in this project.

4.5 Testing

It starts from unit tests - which verify the correctness of individual functionalities. In this case, those are referred to as normalized data and API calls-in contrast to integration tests which would test the valid execution of a number of them together. Performance tests evaluate an application at the level of functionality and performance and end-user experience aspects. Regression testing is automated via scripts such that new updates do not disrupt existing functionality. Functional usability testing collects different opinions from users to judge points of improvement for the interface and navigation. Edge case testing under incomplete input data conditions or API timeout conditions checks if the system would do well in scenarios like this. Test scenario documentation logs reproducibility of the test procedures and all anticipated future enhancements.

4.6 Test Cases

All parts of the system are extensively covered through formation of test cases. They essentially encompass all situations under which preprocessing might fail-including test scenarios for extreme amounts of rainfall, for void values & formatting flaws. Prediction testing confirms that any change in either of the parameters-the state and/or the year-only changes predictions as it ought to. UI tests ensure that behavior is acceptable across devices, monitor correctness of results displayed & check whether user input works across sessions. Security test cases look for issues such as API key vulnerabilities, unauthorized intrusion, & data leakage. Stress tests would simulate extremely high traffic flows to retain their performance by the system. Each test case has a description, input data, an expected & actual result for ease of debugging & quality assurance.

4.7 Advantages of the Project

Advanced machine learning is used to predict floods within the time & accuracy required for the flood management task at hand. The global coordinates make it equally appropriate for all users regardless of whether they are in urban or rural settings. An intuitive interface with visualization & feedback mechanisms will ensure ease of use by people with different levels of technological expertise. Real-time data applied to the history of occurrences will give an immediate yet long-term accurate prediction of events & their happenings. Modular design allows for future improvements like more data sources or predictive capabilities. It will continue to scale for increasing users, while its architecture is robust enough to guarantee performance & reliability. This will also encourage communities & decision-makers to create measures that would proactively deal with flooding conditions, improving the damage mitigation, & saving lives before flooding occurs.

5 IMPLEMENTATION

This project successfully melded data management with machine-learning prediction and real-time assessment of flood risks. It thus becomes a strong-candidate application for disaster preparedness. It brought out several interesting insights and lessons from both successes and the need for improvement for future considerations, through broad testing and analysis.

Among the important accomplishments was the rainfall prediction and flood risk assessment models built up on the platform, which proved to be highly reliable and of very good accuracy. The system uses training rainfall data to forecast future trends using LSTM neural networks to ensure accuracy in prediction. The flood prediction model, built around the Random Forest classifier, offered instant assessments of flood risk and adjusted automatically to rainfall amounts expected. This system remained responsive in the presence of computation load-heavy operations and was able to provide fast and reliable forecasting. Robustness of this nature is instrumental at this particular point for securing the safety of the public and assisting disaster management exercises.

Real-time processing and visualization of data were another definite win. Users could view interactive graphs and receive real-time alerts of flood risks, allowing them to see the trend of rainfall and the risk of flooding. This assisted the users in being alert and understanding their flood risk and the trends of rainfall. The product engenders a larger user base through engagement and invites proactive planning and mitigation strategies.

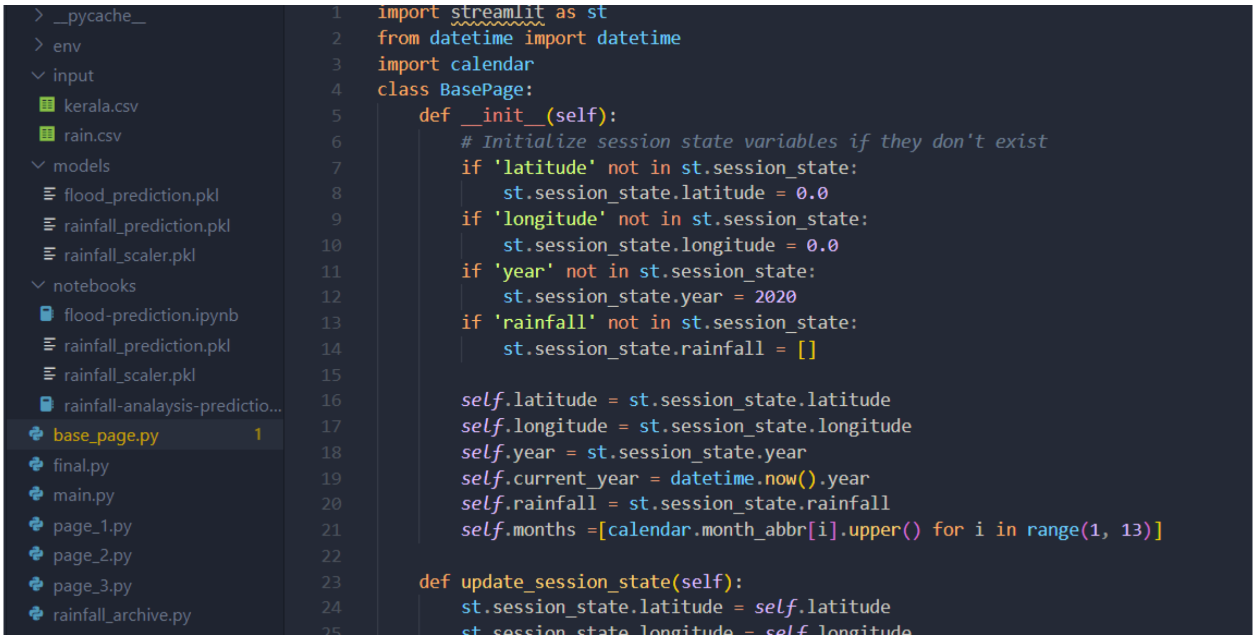
The platform is simply stunningly built on a data management system using Streamlit. It secured more or less all it was involved in: managing user data, historical rainfall, and prediction results on grounds of data integrity and availability. The systematic quick computing retrieval and its efficient analyses enabled user and administrator monitoring trends in just a few clicks. The performance of the system while working on a large dataset indicated that it could deliver a streamlined user experience and provide a strong basis for extendable growth.

There are a couple of important issues that one might address regarding the performance of the platform. To mention a few, one prime issue is the responsiveness of the frontend of the platform. On loading times combined with big traffic, some aspects of the interface would get jittery & eventually inconvenience a user. With this consideration, it becomes of utmost importance to

ensure the smooth maneuvering of the interface regardless of concurrent users.
Another area is scalability. The system was indeed scaled to moderate data loads; the test, however revealed that it hit the limit when the datasets and concurrent requests became large. Adding infrastructures enabling the performance will thus be an important area of growth towards reliable flood predictions and further expansion to bigger numbers of users in the future.

These will be solved through optimizations at both frontend & backend levels. Frontend optimization is going to help with a smooth user experience with better rendering efficiency and oversee management of the state. The backend optimization will also help in scaling up while perfecting machine learning models & refining queries in the database. Also, the load balancing & caching will be used to distribute computation loads efficiently and reduce server strain in a high-traffic scenario.
The implementation outcomes of the platform support its position as a potential strong & interactive ground for flood prediction & rainfall analysis. With elaborate forecasting models, near-real-time risk assessment, & secure data management, the platform delivers a good user experience with it. Such exceptions arose as regards scalability and [response]bring it forth further to fix and improve the system in respect to fulfilling the requirement for a growing audience, hence availing itself as a veritable tool in flood disaster management.

5.1 Code Initialization & Conceptualization

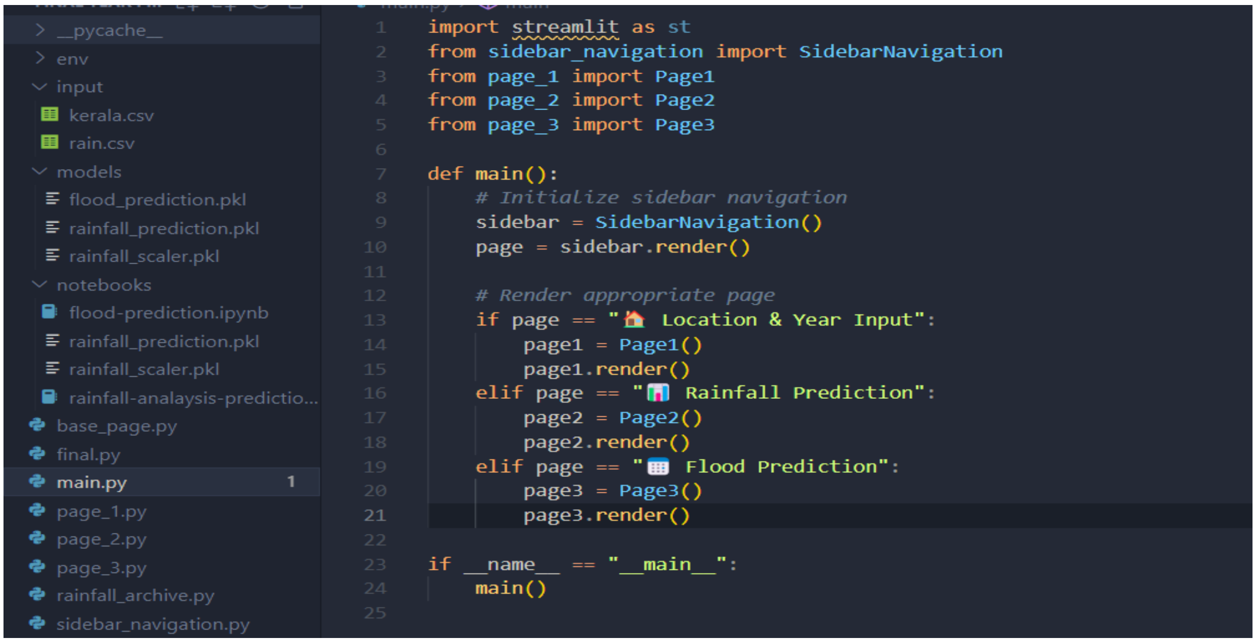


```python
import streamlit as st
from datetime import datetime
import calendar
class BasePage:
    def __init__(self):
        # Initialize session state variables if they don't exist
        if 'latitude' not in st.session_state:
            st.session_state.latitude = 0.0
        if 'longitude' not in st.session_state:
            st.session_state.longitude = 0.0
        if 'year' not in st.session_state:
            st.session_state.year = 2020
        if 'rainfall' not in st.session_state:
            st.session_state.rainfall = []

        self.latitude = st.session_state.latitude
        self.longitude = st.session_state.longitude
        self.year = st.session_state.year
        self.current_year = datetime.now().year
        self.rainfall = st.session_state.rainfall
        self.months =[calendar.month_abbr[i].upper() for i in range(1, 13)]

    def update_session_state(self):
        st.session_state.latitude = self.latitude
        st.session_state.longitude = self.longitude
```

Fig 5.1.1. base_page.py

This code serves in a flood prediction system constructed using Streamlit. It can be regarded as a state manager. Session states are more of variables such as latitude, longitude, year, and rainfall data that make sure values in the application will be remembered all the way throughout the interactions of the user.

The session should be consistent for all the three application pages because the details generally required are common for all three. So it would mean that the user has to fill in the details again and then while the page refreshes.



```
> __pycache__
> env
v input
    kerala.csv
    rain.csv
v models
    flood_prediction.pkl
    rainfall_prediction.pkl
    rainfall_scaler.pkl
v notebooks
    flood-prediction.ipynb
    rainfall_prediction.pkl
    rainfall_scaler.pkl
    rainfall-analaysis-predictio...
  base_page.py
  final.py
  main.py                        1
  page_1.py
  page_2.py
  page_3.py
  rainfall_archive.py
  sidebar_navigation.py
```

```python
import streamlit as st
from sidebar_navigation import SidebarNavigation
from page_1 import Page1
from page_2 import Page2
from page_3 import Page3

def main():
    # Initialize sidebar navigation
    sidebar = SidebarNavigation()
    page = sidebar.render()

    # Render appropriate page
    if page == "🏠 Location & Year Input":
        page1 = Page1()
        page1.render()
    elif page == "📊 Rainfall Prediction":
        page2 = Page2()
        page2.render()
    elif page == "📅 Flood Prediction":
        page3 = Page3()
        page3.render()

if __name__ == "__main__":
    main()

```

Fig 5.1.2. main.py

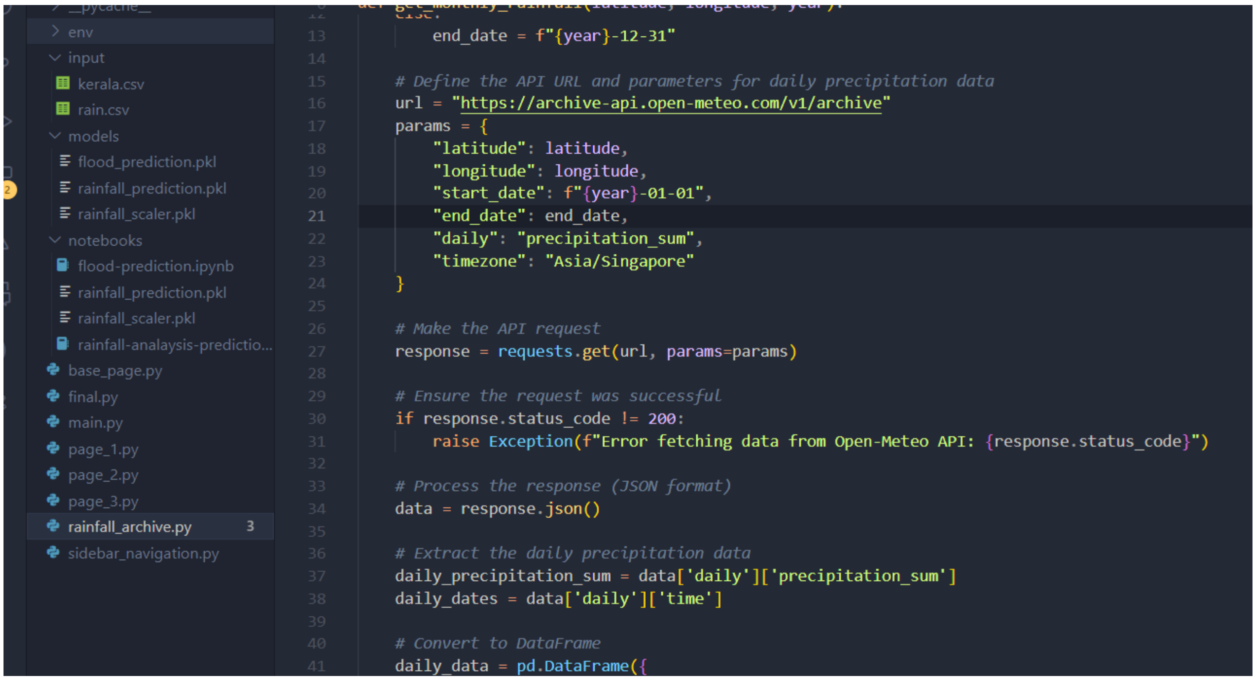
This script is the central controller of the application. It makes use of the navigation sidebar for switching among the three different pages namely, Location & Year Inputs, Rainfall Prediction, and Flood Prediction.

- The sidebar is a place where the user can select a page to go to.
- The modules corresponding to that page will be invoked in order to display the content selected.

This application runs in real time based on the interactions between the user, so it will be fluid and intuitive.

Since the main.py code is modular, it would surely ensure efficient communication between the front end and the back end of the application in order to manage different events proper and efficiently for instant response.

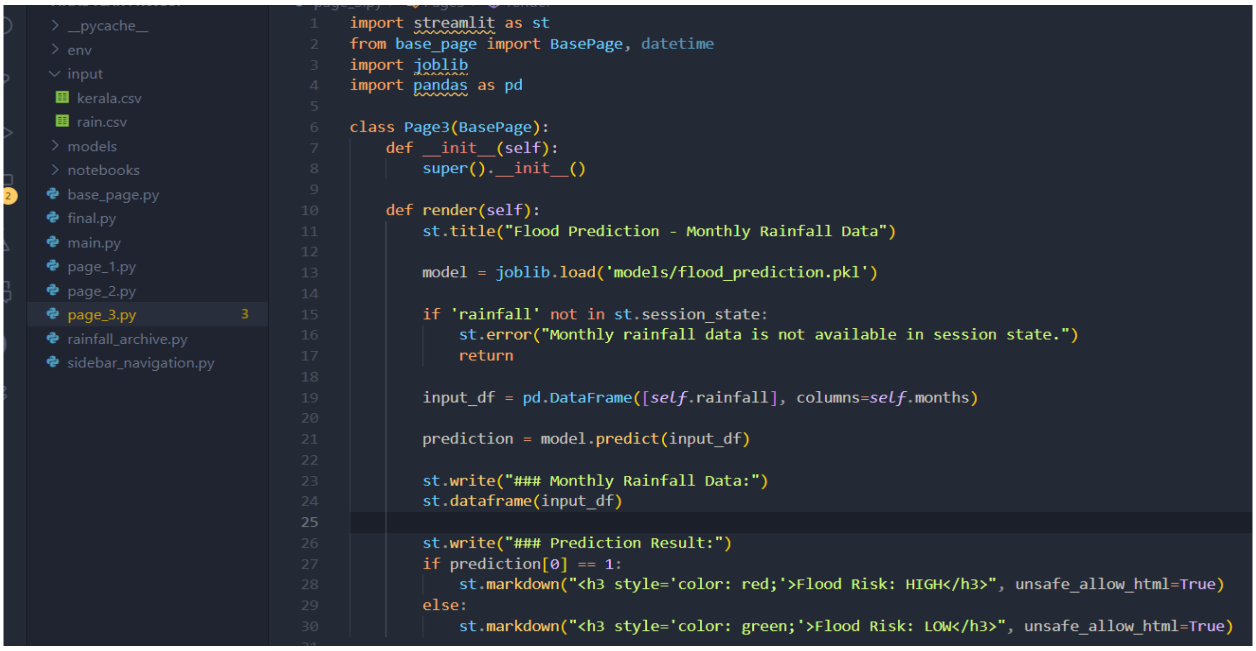
Real-time data updates would also be gathered for current predictions, thus offering an interactive and responsive flood prediction and rainfall analysis platform to the users.



```python
def get_monthly_rainfall(latitude, longitude, year):
        end_date = f"{year}-12-31"

    # Define the API URL and parameters for daily precipitation data
    url = "https://archive-api.open-meteo.com/v1/archive"
    params = {
        "latitude": latitude,
        "longitude": longitude,
        "start_date": f"{year}-01-01",
        "end_date": end_date,
        "daily": "precipitation_sum",
        "timezone": "Asia/Singapore"
    }

    # Make the API request
    response = requests.get(url, params=params)

    # Ensure the request was successful
    if response.status_code != 200:
        raise Exception(f"Error fetching data from Open-Meteo API: {response.status_code}")

    # Process the response (JSON format)
    data = response.json()

    # Extract the daily precipitation data
    daily_precipitation_sum = data['daily']['precipitation_sum']
    daily_dates = data['daily']['time']

    # Convert to DataFrame
    daily_data = pd.DataFrame({
```

Fig 5.1.3. rainfall_archive.py

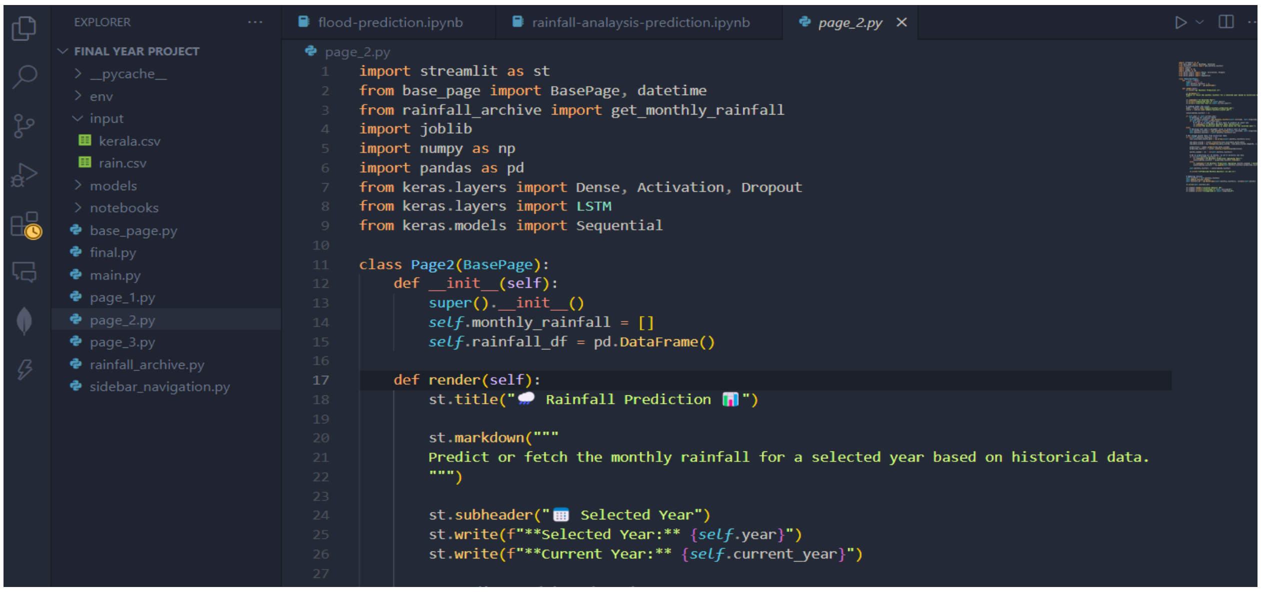
Python code for a function called get_monthly_rainfall, an API that is intended to retrieve historical precipitation data for a given location & year. The retrieved data can be used for analyzing historical rainfall trends or feeding into a machine learning model.



```python
import streamlit as st
from base_page import BasePage, datetime
import joblib
import pandas as pd

class Page3(BasePage):
    def __init__(self):
        super().__init__()

    def render(self):
        st.title("Flood Prediction - Monthly Rainfall Data")

        model = joblib.load('models/flood_prediction.pkl')

        if 'rainfall' not in st.session_state:
            st.error("Monthly rainfall data is not available in session state.")
            return

        input_df = pd.DataFrame([self.rainfall], columns=self.months)

        prediction = model.predict(input_df)

        st.write("### Monthly Rainfall Data:")
        st.dataframe(input_df)

        st.write("### Prediction Result:")
        if prediction[0] == 1:
            st.markdown("<h3 style='color: red;'>Flood Risk: HIGH</h3>", unsafe_allow_html=True)
        else:
            st.markdown("<h3 style='color: green;'>Flood Risk: LOW</h3>", unsafe_allow_html=True)
```

Fig 5.1.4. page_3.py

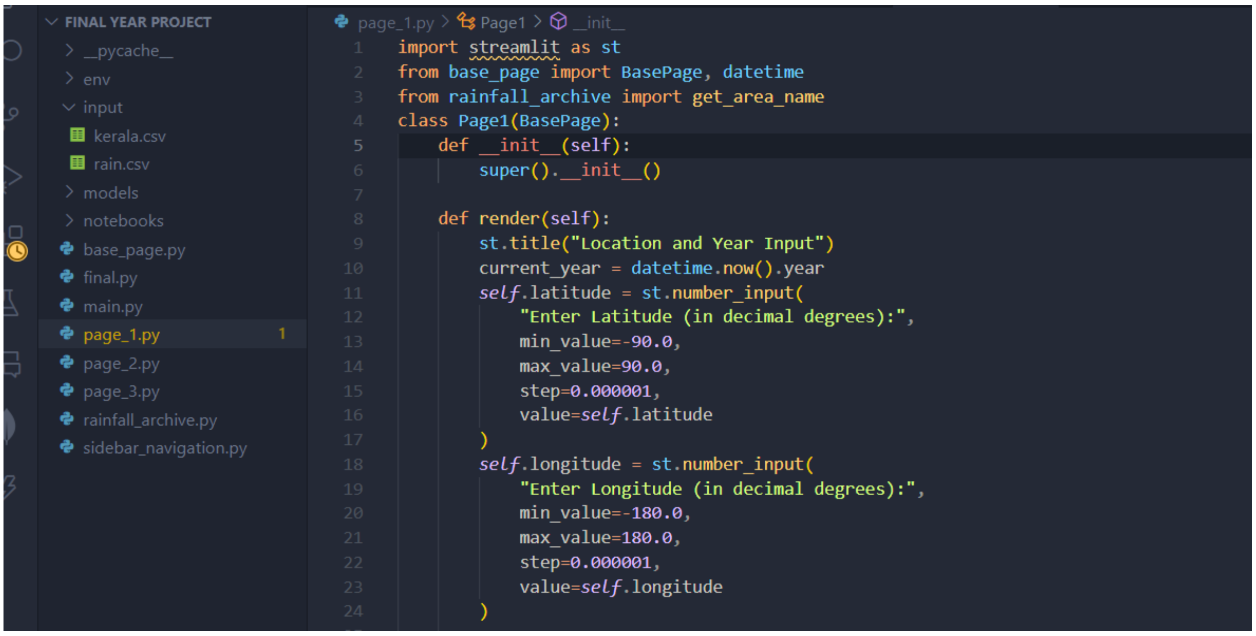
It initializes & controls session states for several variables used all through the application. The class is simply part of a larger Streamlit application related to predicting rainfall. It is the parent class of other page-specific classes (like Page2) and ensures that all the pages share the same application state.



```python
import streamlit as st
from base_page import BasePage, datetime
from rainfall_archive import get_monthly_rainfall
import joblib
import numpy as np
import pandas as pd
from keras.layers import Dense, Activation, Dropout
from keras.layers import LSTM
from keras.models import Sequential

class Page2(BasePage):
    def __init__(self):
        super().__init__()
        self.monthly_rainfall = []
        self.rainfall_df = pd.DataFrame()

    def render(self):
        st.title("🌧️ Rainfall Prediction 📊")

        st.markdown("""
        Predict or fetch the monthly rainfall for a selected year based on historical data.
        """)

        st.subheader("📅 Selected Year")
        st.write(f"**Selected Year:** {self.year}")
        st.write(f"**Current Year:** {self.current_year}")
```

Fig 5.1.5. page_2.py

- Determines if data should be downloaded for the current year or the prior one.
- This logic ensures the right information gets used in prediction or display in reference to the year, which the user inputs.



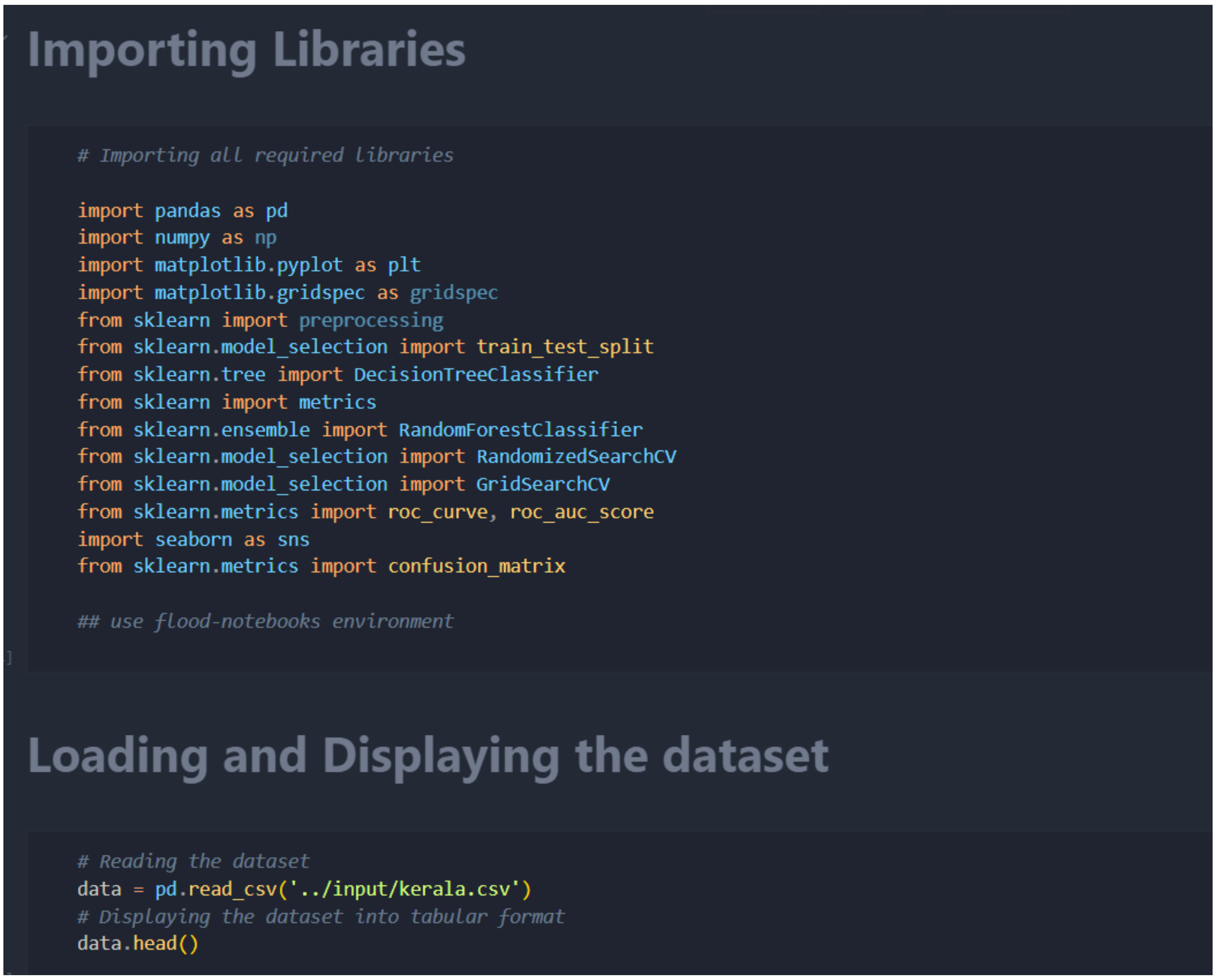
```python
import streamlit as st
from base_page import BasePage, datetime
from rainfall_archive import get_area_name
class Page1(BasePage):
    def __init__(self):
        super().__init__()

    def render(self):
        st.title("Location and Year Input")
        current_year = datetime.now().year
        self.latitude = st.number_input(
            "Enter Latitude (in decimal degrees):",
            min_value=-90.0,
            max_value=90.0,
            step=0.000001,
            value=self.latitude
        )
        self.longitude = st.number_input(
            "Enter Longitude (in decimal degrees):",
            min_value=-180.0,
            max_value=180.0,
            step=0.000001,
            value=self.longitude
        )
```

Fig 5.1.6. page_1.py

A python file using Streamlit framework for a web-based application in rainfall prediction. The code computes predicted rainfall in a month. When there are predictions for all 12 months, it is smooth and simple. And in case there aren't, it combines predictions to reflect the number of months needed now.

5.2 Model Initialization & Conceptualization

1. Rainfall Analysis & Prediction Notebook



Importing Libraries

```
# Importing all required libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import roc_curve, roc_auc_score
import seaborn as sns
from sklearn.metrics import confusion_matrix

## use flood-notebooks environment
```

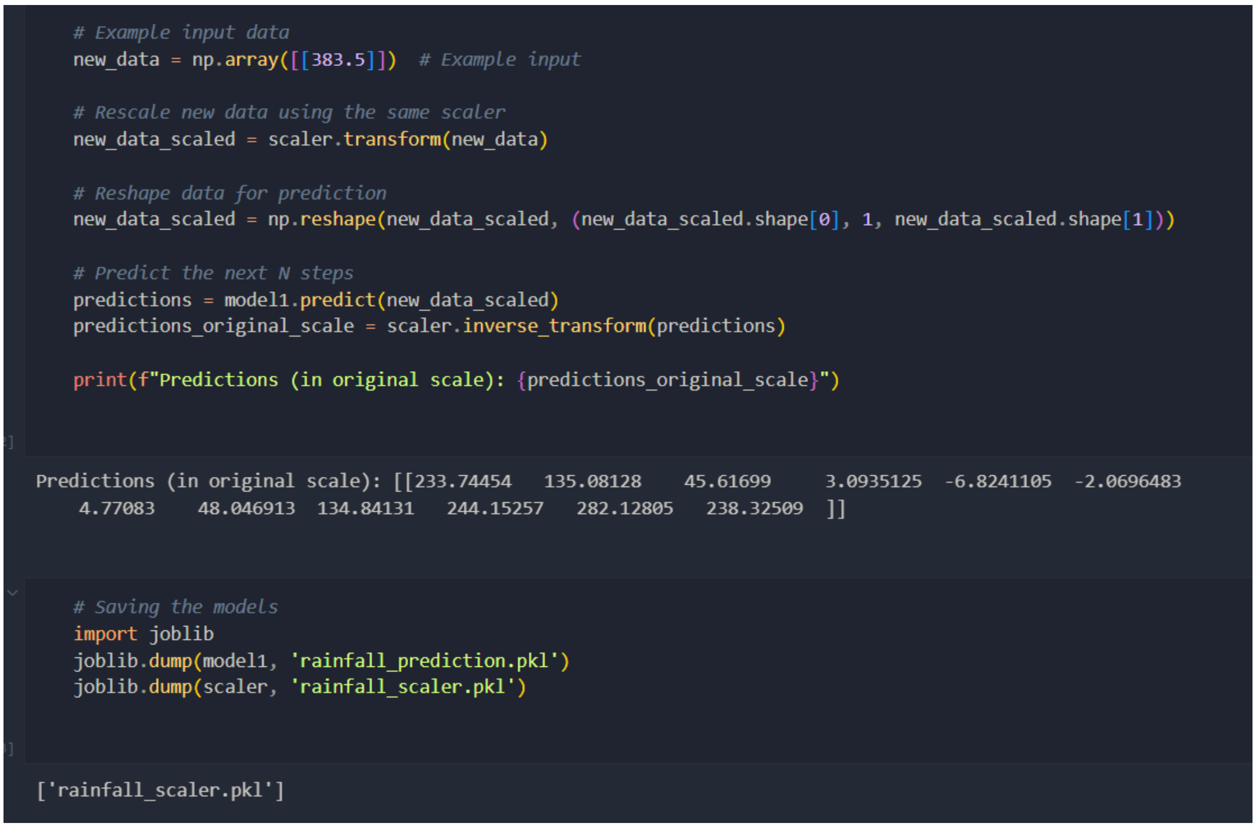
Loading and Displaying the dataset

```
# Reading the dataset
data = pd.read_csv('../input/kerala.csv')
# Displaying the dataset into tabular format
data.head()
```

Fig 5.2.1. rainfall-analysis. ipynb

This notebook covers data preprocessing, feature engineering, and the training of an LSTM-based model over rainfall predictions. It will load historical rainfall data, clean it, perform normalization, and employ certain techniques for time-series forecasting to predict rainfall in coming months. Afterward, some visualizations in terms of trend graphs and heatmaps will be added to support analysis on the patterns of rainfall data. The model will be saved and reused for an application intended for actual user execution that will provide accurate rainfall predictions.

2. Flood Prediction Notebook



```python
# Example input data
new_data = np.array([[383.5]])  # Example input

# Rescale new data using the same scaler
new_data_scaled = scaler.transform(new_data)

# Reshape data for prediction
new_data_scaled = np.reshape(new_data_scaled, (new_data_scaled.shape[0], 1, new_data_scaled.shape[1]))

# Predict the next N steps
predictions = model1.predict(new_data_scaled)
predictions_original_scale = scaler.inverse_transform(predictions)

print(f"Predictions (in original scale): {predictions_original_scale}")
```

```
Predictions (in original scale): [[233.74454   135.08128    45.61699     3.0935125  -6.8241105  -2.0696483
    4.77083    48.046913  134.84131   244.15257   282.12805   238.32509  ]]
```

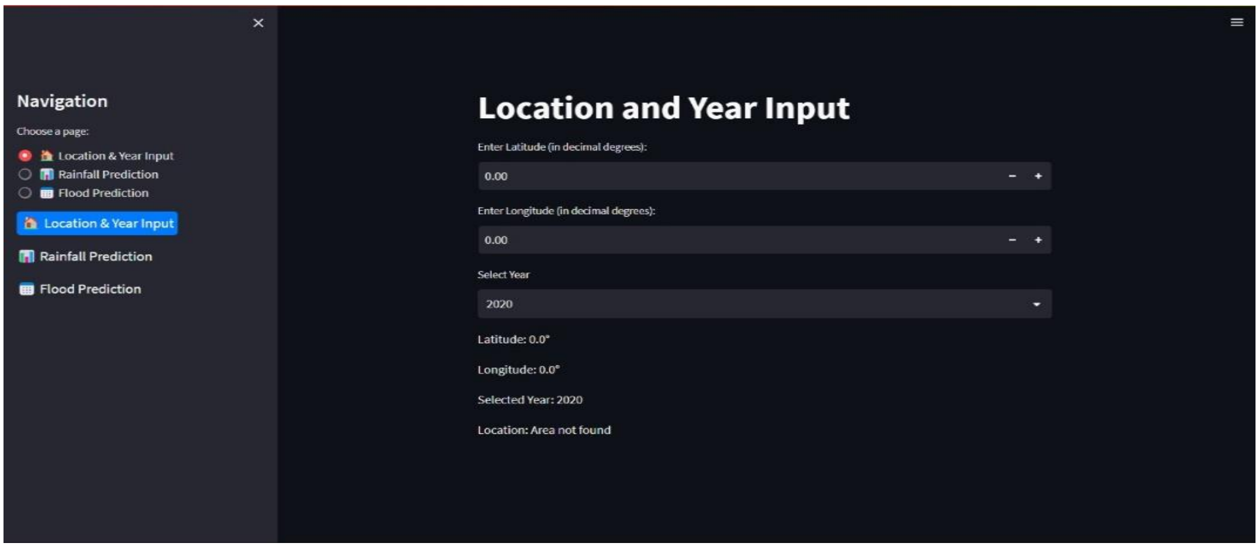
```python
# Saving the models
import joblib
joblib.dump(model1, 'rainfall_prediction.pkl')
joblib.dump(scaler, 'rainfall_scaler.pkl')
```

```
['rainfall_scaler.pkl']
```

Fig 5.2.2. flood-prediction. ipynb

This notebook takes predictions of rainfall to also analyze the flood risk estimation using a machine learning model. This predicts the rainfall values and utilizes historical flood occurrence data to train a Random Forest classifier for predicting binary flood risks: either High or Low. For completeness, the notebook implements data visualization, model performance metrics, and feature importance. Export of the model follows for real-time predictions in the application.

6 RESULTS



Location and Year Input

Enter Latitude (in decimal degrees):

0.00

Enter Longitude (in decimal degrees):

0.00

Select Year

2020

Latitude: 0.0°

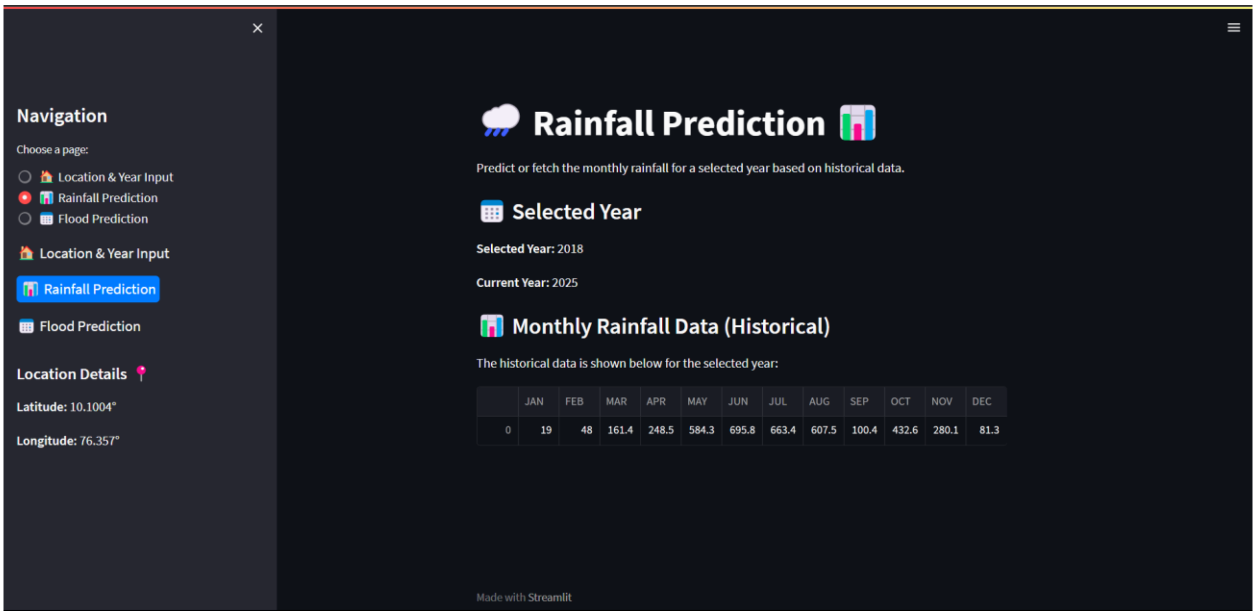
Longitude: 0.0°

Selected Year: 2020

Location: Area not found

Fig 6.1. Home screen

This figure (Fig 6.1) presents the initial screen of the application hailed up right after the launch. It does take the latitude, longitude, & year as input in order to compute the rainfall analysis, which helps predict flooding.



Location Details

Latitude: 10.1004°

Longitude: 76.357°

Rainfall Prediction

Predict or fetch the monthly rainfall for a selected year based on historical data.

Selected Year

Selected Year: 2018

Current Year: 2025

Monthly Rainfall Data (Historical)

The historical data is shown below for the selected year:

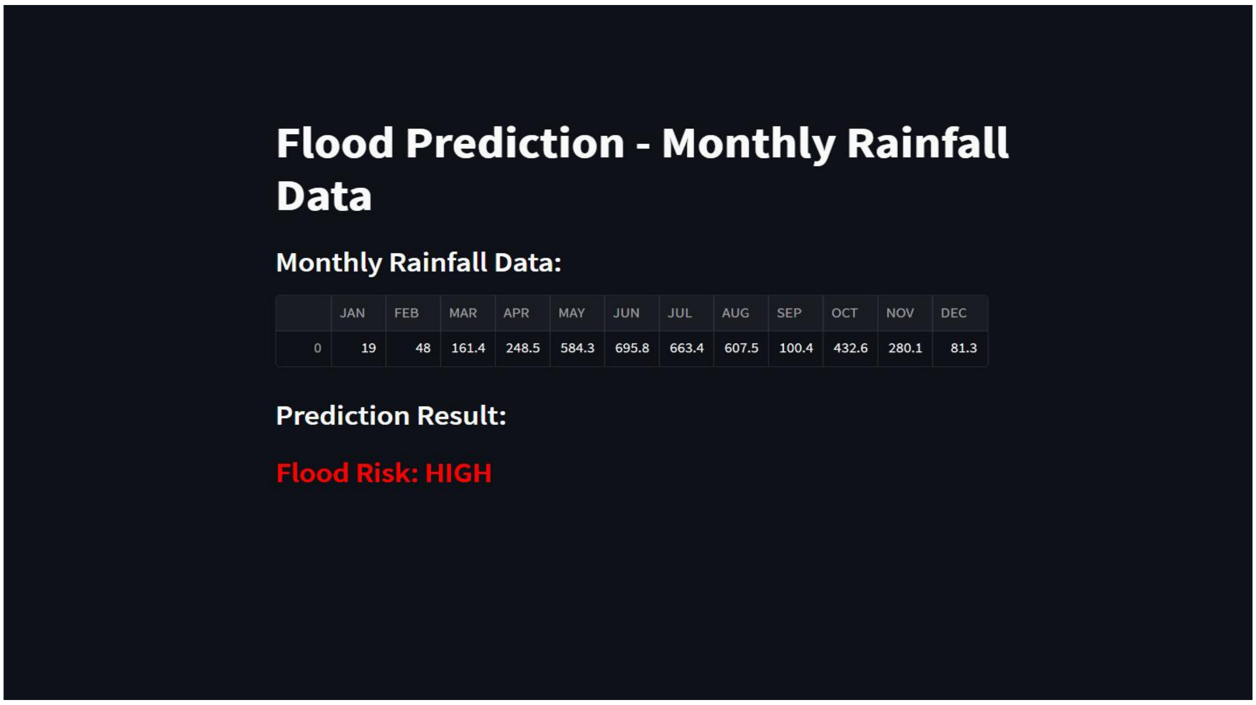
	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
0	19	48	161.4	248.5	584.3	695.8	663.4	607.5	100.4	432.6	280.1	81.3

Fig 6.2: Monthly Rainfall Data

Fig 6.2 Image showing monthly rainfall data for which the user has asked for a particular year. Monthly historical rainfall data includes predictive analysis for each month, selected according to the user's input for the specified year. It shows an application interface for predicting rainfall that

allows the selection of a year with historical monthly rainfall data in millimeters arranged in tabular form. There are pointers to "Selected Year" and "Current Year" within the interface.

The user interface is used to enter the location and the year to analyse rainfall predictions produced month-wise. Latitude and longitude selected are 10.10° & 76.36°, respectively & the selected year being 2018. The location comes up with Aluva, Kerala, India, and the proper coordinates and selected year displays. The panel navigates to rainfall & flood prediction tools.



Flood Prediction - Monthly Rainfall Data

Monthly Rainfall Data:

	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
0	19	48	161.4	248.5	584.3	695.8	663.4	607.5	100.4	432.6	280.1	81.3

Prediction Result:

Flood Risk: HIGH

Fig 6.3: Flood Prediction Result

Fig 6.3 Flood prediction risk based on monthly rainfall data forth. This image is showing the flood prediction report dependent on monthly rainfall data, providing the volume of monthly rainfall, showing throughout the middle year-months from May till September significant heights of rainfall predicted.

7 CONCLUSION & FUTURE SCOPE

7.1 Conclusion

Flood Forecasting: Rainfall Prediction is about how deep learning combining with machine learning techniques has opened the avenues of finding solutions for nearly all the tough and complex real-world challenges of today's context. While now recognized as an integral part of a set of activities termed disaster preparation, flood prediction till very recently had remained an arena where more accuracy was demanded in the predictions.

Developed are the models aimed at optimizing the power and strength of classifiers & advanced algorithms. The models work with complex patterns in data, with the central goal being the extraction & timely delivery of actionable insights that would effectively spur a response to flooding situations. The whole system puts an emphasis on the use of accurate & powerful models in making predictions.

The aspiration has taken a significant leap in successful identification of potential scenarios likely to be flooded and which can contribute towards the mechanism of early warning for averting disasters, safeguarding lives, minimizing possible loss, and building the community's resilience. The system utilizes a series of classifiers that can learn and further adapt in a fashion that will better the system's accuracy with the passage of time by gaining access to older data. This work within the study forms the basis for further research and development into flood predictions.

Fundamentally, the work intends to present an advanced and data-driven approach towards rain prediction and flood risk assessment based on applied machine learning techniques, culminating in high accuracy and dependability. The LSTM was applied for rainfall prediction while Random Forest performed the flooding prediction. These, therefore, aid in proactive management during disasters.

An ideal solution to control flood risk is enhanced with historical rainfall data and predictive analytics for easy and practical solutions.

7.2 Future Scope

Further Scope: The project aims to extend this knowledge into predicting more indicators of environmental status, such as soil moisture, river water, and land surface characteristics. Fusing a combination of multiple-source data, satellite imagery, and Internet-of-Things based weather sensors is expected to narrow down model error and provide multi-perspective flood risk. Optimized real-time data processing and dynamic model updating will allow the model to update based on the current weather patterns for better forecasting and predictions. Cloud-based deployment will also scale up application extensibility and reach into the heterogeneous geographical region based on the prevailing rainfall and flood patterns. Another promising area is developing a friendly mobile or web-based platform for the real-time predictions of rainfall and floods to be accessed by population users, government agencies, and coordinators for management of disasters.

The geospatial visualization tools will facilitate better interpretation of forecasts by users.

In addition, integration with AI-based flood early warning systems that issue automated alerts through SMS, apps, & social networking will also ensure timely alerts of at-risk communities. Future enhancements shall also include deep learning models fed on-going data inputs to enhance future extreme weather forecasting capacity, thus steering effective disaster management & climate resilience.

8 REFERENCES

[1] Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep Learning. MIT Press.

[2] Hochreiter, S., and Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), 1735-1780.

[3] Breiman, L. (2001). Random Forests. Machine Learning, 45(1), 5-32.

[4] Raschka, S., and Mirjalili, V. (2019). Python Machine Learning: Machine Learning & Deep Learning with Python, scikit-learn, & TensorFlow 2 (3rd ed.). Packt Publishing.

[5] Chollet, F. (2018). Deep Learning with Python. Manning Publications.

[6] Murphy, K. P. (2012). Machine Learning: A Probabilistic Perspective. MIT Press.

[7] Abadi, M., Barham, P., Chen, J., et al. (2016). TensorFlow: A System for Large-Scale Machine Learning. In OSDI'16: Proceedings of the 12th USENIX Conference on Operating Systems Design & Implementation (pp. 265-283).

[8] Rodriguez, D., Wijnen, R., and Maathuis, B. (2016). OpenMeteo API Documentation. OpenMeteo.

[9] Bishop, C. M. (2006). Pattern Recognition & Machine Learning. Springer