**Rainfall Prediction Using Machine Learning**

**A Project Report Submitted in Partial Fulfilment of**

**the Requirements for the Degree of**

**Bachelor of Technology**

**in**

**Computer Science (Artificial Intelligence)**

**by**

**Mr. Vinay Pratap Singh 2003481520022**

**Mr. Neeraj Singh 2003481520022**

**Mr. Prateek Singh pal 2003481520014**

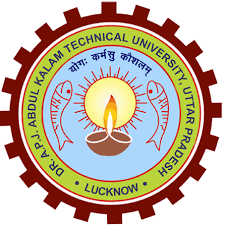
**Under the Supervision of**

**Miss. Priyanka Arya**

**(Assistant Professor)**

**PSIT COLLEGE OF ENGINEERING, KANPUR**

**to the**



**Faculty of Computer Science & Engineering**

**Dr. A.P.J. Abdul Kalam Technical University, Lucknow**

**June 2024**

# DECLARATION

I declare that this submission is solely my team oading datasetsg algorithms metioned above.work, and to the best of my knowledge and belief, it does not include any previously published or written material by another person. Furthermore, it does not contain any material that has substantially contributed to the award of any degree or diploma from a university or other institute of higher learning, except where proper acknowledgment has been provided within the text.

Name : Mr. Vinay Pratap Singh

Roll No : 2003481520022

Date :

Signature :

Name : Mr. Neeraj Singh

Roll No : 2003481520011

Date :

Signature :

Name : Mr. Prateek Singh Pal

Roll No : 2003481520014

Date :

Signature :

# ACKNOWLEDGEMENT

It gives us a great sense of pleasure to present the report of B.Tech. Project Rainfall Prediction Using Machine Learning undertaken during B.Tech. Final Year. We owe special debt of gratitude to our project mentor **Miss. Priyanka Arya (Assistant Professor, CSE), PSIT College of Engineering Kanpur** for his constant support and guide throughout course our work. His sincerity, thoroughness and perseverance have been a constant source of inspiration for us. It is only his cognizant efforts that our endeavours have seen light of the day.

We also do not like to miss the opportunity to acknowledge the contribution of my co-ordinator **Mr. Amit Kumar Sharma** **(Assistant Professor, CSE), PSIT College of Engineering Kanpur**, H.O.D **Mr. Abhay Kumar Tripathi** **(Assistant Professor, CSE), PSIT College of Engineering Kanpur** and all faculty member of the department for their kind assistance and cooperation during the development of our project. Last but not the least, we acknowledge our friends for their contribution in the completion of the project.

Name : Mr. Neeraj Singh

Roll No : 2003481520011

Date :

Signature :

Name : Mr. Vinay Pratap Singh

Roll No : 2003481520022

Date :

Signature :

Name : Mr. Prateek Singh Pal

Roll No : 2003481520011

Date :

Signature :

# CERTIFICATE

This is to certify that the project titled Rainfall Prediction Using ML is submitted by

* Mr. Neeraj Singh (2003481520011)
* Mr. Vinay Pratap Singh (2003481520022)
* Mr. Prateek Singh Pal (2003481520014)

in partial fulfilment of the requirement for the award of the degree of Bachelor of Technology in Computer Science (Artificial Intelligence) to PSIT College of Engineering, Kanpur, affiliated to Dr. A.P.J. Abdul Kalam Technical University, Lucknow, during the academic year 2023–24, is the record of the candidate’s own work carried out by him under my supervision. The matter embodied in this report is original and has not been submitted for the award of any other degree.

Mr. Abhay Kumar Tripathi Miss. Priyanka Arya

(Head of Dept. of CSE) (Asst. Professor, Dept. of CSE)

# ABSTRACT

Accurate rainfall prediction is important for managing water resources, planning agriculture, and preparing for natural disasters. Rainfall prediction is the demanding and mysterious tasks which has a extraordinary impact on human society. Timely and accurate predictions can help to decrease human and financial loss. Rainfall Prediction is the application of science and technology to predict the state of the atmosphere.

It is important to exactly determine the rainfall for effective use of water resources, crop productivity and pre planning of water structures. Traditional rainfall forecasting methods often struggle with the complex nature of weather data. This project uses machine learning (ML) to predict rainfall for the next day and for the upcoming month.

We use historical weather data, like temperature, humidity, wind speed, and past rainfall, to train various ML models such as Linear Regression, Decision Trees, Random Forests, and Support Vector Machine. We evaluate these models based on how accurately and quickly they can predict rainfall, using metrics like mean absolute error (MAE), root mean square error (RMSE), and correlation coefficient (R).

We also improve the models by identifying the most important factors that affect rainfall and testing different data preparation methods, like normalization and removing outliers.Our results show that ML models can predict rainfall more accurately than traditional methods, with neural networks performing the best at capturing complex weather patterns. These models are useful for daily and monthly planning, helping people better prepare for rainfall and reduce its negative impacts.

This project highlights the benefits of using advanced data analysis in weather forecasting and encourages further research to improve these ML models.

**Keywords: -**Mean Absolute Error (MAE), Root Mean Square Error (RMSE), Decision Trees, Random Forests, Support Vector Machine, Correlation Coefficient (R).

# TABLE OF CONTENT

TITLE PAGE………………………………………………………………………. i

[DECLARATION ii](#_Toc167096199)

[ACKNOWLEDGEMENT iii](#_Toc167096200)

[CERTIFICATE iv](#_Toc167096201)

[ABSTRACT v](#_Toc167096202)

[TABLE OF CONTENT vi](#_Toc167096203)

[LIST OF FIGURES ix](#_Toc167096204)

[LIST OF TABLE 12](#_Toc167096205)

[CHAPTER 1: INTRODUCTION 13](#_Toc167096206)

[1.1 INTRODUCTION TO PROBLEM: 13](#_Toc167096207)

[1.2 IMPORTANCE 15](#_Toc167096208)

[1.3 SOLUTION 19](#_Toc167096209)

[1.4 CONCLUSION 23](#_Toc167096210)

[1.5 SOFTWARE SPECIFICATION FOR THE RAINFALL PREDICTION PROJECT 23](#_Toc167096211)

[1.6 HARDWARE SPECIFICATION 25](#_Toc167096212)

[CHAPTER 2: LITERATURE REVIEW 27](#_Toc167096213)

[2.1 EXISTING SYSTEM 27](#_Toc167096214)

[2.1.1 OVERVIEW 27](#_Toc167096215)

[2.1.2 RAINFALL PREDICTION CHALLENGES 28](#_Toc167096216)

[2.2 PROPOSED SYSTEM 29](#_Toc167096217)

[2.2.1 BRIEF OVERVIEW 30](#_Toc167096218)

[2.2.2 ARCHITECTURE 31](#_Toc167096219)

[2.2.2.1 Rainfall Prediction Architecture 32](#_Toc167096220)

[2.2.3 ANALYSIS AND MODEL 32](#_Toc167096221)

[2.3 FEASIBILITY STUDY 38](#_Toc167096222)

[2.3.1 OBJECTIVES 38](#_Toc167096223)

[2.3.2 TYPES OF FEASIBILITY 39](#_Toc167096224)

[CHAPTER 3: SYSTEM ANALYSIS & DESIGN 40](#_Toc167096225)

[3.1 REQUIREMENT SPECIFICATION 40](#_Toc167096226)

[3.1.1 FUNCTIONAL REQUIREMENT SPECIFICATION 44](#_Toc167096227)

[3.1.2 NON-FUNCTIONAL REQUIREMENT SPECIFICATION 45](#_Toc167096228)

[3.2 FLOWCHARTS 48](#_Toc167096229)

[3.2.1 USE OF FLOWCHART IN RAINFALL PREDICTION MODEL 49](#_Toc167096230)

[3.3 DATA FLOW DIAGRAM 51](#_Toc167096231)

[3.3.1 LEVEL 0 53](#_Toc167096232)

[3.3.2 LEVEL 1 54](#_Toc167096233)

[3.3.3 Level 2 56](#_Toc167096234)

[3.3.4 Level 3 57](#_Toc167096235)

[3.4 UML Diagrams 59](#_Toc167096236)

[3.4.1 GOALS: 59](#_Toc167096237)

[3.4.2 USE CASE DIAGRAM: 60](#_Toc167096238)

[3.4.3 SEQUENCE DIAGRAM: 61](#_Toc167096239)

[3.4.4 ACTIVITY DIAGRAM: 61](#_Toc167096240)

[3.5 DESIGN AND TEST STEPS 63](#_Toc167096241)

[3.6 ALGORITHM AND PSEUDOCODE 66](#_Toc167096242)

[3.6.1 ALGORITHM FOR RAINFALL PREDICTION USING MACHINE LEARNING 66](#_Toc167096243)

[3.6.2 PSEUDOCODE FOR RAINFALL PREDICTION ALGORITHM: 69](#_Toc167096244)

[3.6.3 ACTUAL CODE FOR ANALYSIS 72](#_Toc167096245)

[3.7 TESTING 77](#_Toc167096246)

[3.7.1 VERIFICATION 77](#_Toc167096247)

[3.7.2 VALIDATION 78](#_Toc167096248)

[3.7.3 BLACK BOX 78](#_Toc167096249)

[3.7.4 WHITE BOX 79](#_Toc167096250)

[3.7.5 TYPE OF TESTING 81](#_Toc167096251)

[CHAPTER 4: RESULT AND OUTPUT 86](#_Toc167096252)

[4.1 DATA ANALYSIS RESULTS 86](#_Toc167096253)

[CHAPTER 5: CONCLUSIONS 98](#_Toc167096254)

[5.1 OBJECTIVE SOLUTIONS 98](#_Toc167096255)

[CHAPTER 6: RECOMMENDATIONS: 100](#_Toc167096256)

[CHAPTER 7: APPENDICES 102](#_Toc167096257)

[7.1 DATA ANALYSIS EDA CODE: 102](#_Toc167096258)

[7.2 WORKING ALGORITHM 104](#_Toc167096259)

[REFERENCES 106](#_Toc167096260)

# LIST OF FIGURES

|  |
| --- |
| 1.1.0.1 Basis Project Flow Diagram………………………………………….12 |
| 2.1.1.1 Prediction Application…...……………….…………………………..25 |
| 2.1.1.2 Monsoon Image.………………………….……...………………...…26 |
| 2.2.1.1 Different Rainfall Prediction Systems..….…………………………...28 |
| 2.2.2.1 Rainfall Prediction Architecture.….……………………………29 |
| 2.2.3.1 Detailed Diagram……………………….…………………………….30 |
| 2.2.3.2 Algorithm Flowchart…..…………….………………………………..31 |
| 2.2.3.3 Logistic Regression Flowchart……….………………………………32 |
| 2.2.3.4 Sigmoid Function……………...……………………………………..32 |
| 2.2.3.5 Support Vector Machine (SVM).…….………………………………33 |
| 2.2.3.6 Random Forest…………………….…………………………………34 |
| 2.2.3.7 K-Nearest Neighbour….………….…………………………………..35 |
| 2.3.0.1 Feasibility Study Parts….….…….…………………………………..36 |
| 3.1.0.1 Overview Basic Diagram……….……………………………………39 |
| 3.1.0.2 Feature Extraction…..………….…………………………………….40 |
| 3.1.0.3 Model A….…………………….…………………………………......41 |
| 3.2.0.1 Flow Chart ...……………………….…………………………….......46 |
| 3.2.1.1 Basic Methodology……………………………………………...........47 |
| 3.2.1.2 Methodology Diagram ………………………………………..…….48 |
| 3.3.1.1 Level 0 DFD……………………………………………………….…51 |
| 3.3.2.1 Level 1 DFD ………………………………………………….………52 |
| 3.3.3.1 Level 2 DFD ……………….………………………………………...54 |
| 3.3.4.1 Level 3 DFD ………………………………………………………….55 |
| 3.4.2.1 Use case diagram……………………………………………………..57 |
| 3.4.3.1 Sequence diagram…………..………………………………………58 |
| 3.4.4.1 Activity Diagram ……………………………………………………..60 |
| 3.5.0.1 Basic design………………………………………………...………...61 |
| 3.5.0.2 Algorithm Diagram …………………………………..………………63 |
| 3.6.1.1 Algorithm …………………………………………………………….64 |
| 3.6.3.1 Pseudo Snippet Splitting….…...………….………………………….70 |
| 3.6.3.2 Pseudo Snippet Linear regression ……………………...…………..71 |
| 3.6.3.3 Pseudo Snippet Random Forest…………….……...………….…...…71 |
| 3.6.3.4 Pseudo Snippet Lasso Regression……….…………………………...72 |
| 3.6.3.5 Pseudo Snippet SVM ..…………………….…………………………73 |
| 3.6.3.6 Pseudo Snippet Logistic Regression …..…….………………………73 |
| 3.6.3.7 Pseudo Snippet KNN……………………….….……………………..74 |
| 3.6.3.8 Pseudo Snippet Comparison Algorithm ..……………………………74 |
| 3.7.0.1 Testing Graph ………………………………………………………..75 |
| 3.7.3.1 Black Box Testing………………….…………………………………77 |
| 3.7.4.1 White Box Testing ……………………………………………………78 |
| 3.7.5.1 Unit Testing ………….……………………………………………….79 |
| 3.7.5.2 Integration Testing …………….……………………………………..80 |
| 3.7.5.3 Usability Testing ……………………….………………………….....81 |
| 3.7.5.4 Acceptance Testing ….………….……………………………………82 |
| 3.7.5.5 Regression Testing ….………….…………………………………….83 |
| 4.1.0.1 Data Analysis Of Feature ………………………………………….....84 |
| 4.1.0.2 Feature EDA Diagram ……………………………………………......85 |
| 4.1.0.3 Accuracy Diagram ……………………………………………...........86 |
| 4.1.0.4 Output Daily Basis ………………………………………..……….…87 |
| 4.1.0.5 **A**nnual Rainfall vs Subdivisions....……………………………….…89 |
| 4.1.0.6 This plot shows the average rainfall in mm in the year ……….…….90 |
| 4.1.0.7 This plot shows rainfall pattern through line graph yearly vs two months.…………………………………………………………………….....91 |
| 4.1.0.8 Rainfall Pattern Diagram Yearwise ……………………………………………………………………………….92 |
| 4.1.0.9 West UP Rainfall Monthly…………………………………………...93 |
| 4.1.0.10 Simple GUI (Graphical User Interface) by which user interacts and get the output generated by the model. Enter the month in numbers and year ……………………………………………………………………….………94 |
| 4.1.0.11 Monthly Result GUI ………………………………………………..95 |
| 5.1.0.1 Rainfall Improvement ………………….………………...………...97 |
| 7.2.0.1 Algorithm Flow Diagram …...………………………………………103 |

# LIST OF TABLE

Table 1 Existing System……………………………………………………………...…….…27

Table 2 Functional and Non-Functional ……………………………………………….…..…45

Table 3 Black Box Testing………………………………………………………...……….…77

Table 4 White Box Testing……………………………………………………………………78

Table 5 Accuracy Score…………………………………………………………………….....87

# CHAPTER 1: INTRODUCTION

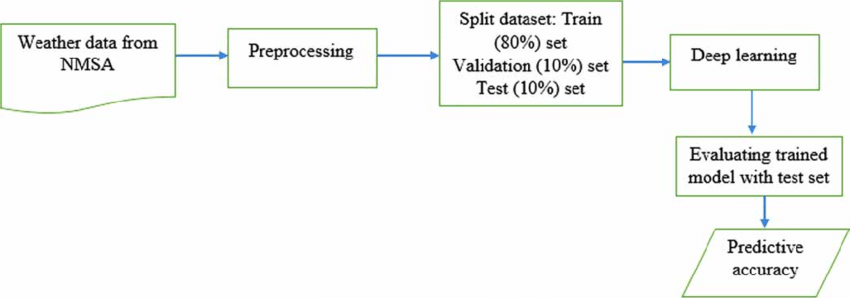
## 1.1 INTRODUCTION TO PROBLEM:

Accurate rainfall prediction is essential for a wide range of activities, from agricultural planning and water resource management to disaster preparedness and urban infrastructure development. Effective forecasting helps farmers optimize planting and harvesting times, water managers allocate resources efficiently, and authorities implement timely flood or drought mitigation strategies. However, the inherent complexity and variability of weather patterns present significant challenges to traditional forecasting methods, which often struggle to provide precise and reliable predictions.

The advent of machine learning (ML) has opened new avenues for enhancing the accuracy of rainfall predictions. ML algorithms can process vast amounts of historical weather data, uncover hidden patterns, and make predictions that account for the intricate interactions among various meteorological factors. Unlike conventional statistical models, ML techniques can dynamically adapt to new data and improve their predictive performance over time.

This project aims to harness the power of ML to predict rainfall on both daily (next day) and monthly timescales. By utilizing historical data such as temperature, humidity, wind speed, and past precipitation, we develop and evaluate several ML models, including Linear Regression, Decision Trees, Random Forests, and SVM. Each model is assessed for its ability to accurately forecast rainfall, with a focus on improving prediction reliability and computational efficiency.

The methodology involves comprehensive data preprocessing and feature engineering to optimize the models. Key performance metrics like mean absolute error (MAE), root mean square error (RMSE), and correlation coefficient (R) are used to evaluate and compare the models. Our goal is to demonstrate that ML can significantly enhance the precision of rainfall forecasts, providing valuable insights for short-term and long-term planning.



1.1.0.1 Basis Project Flow Diagram

This introduction outlines the pressing need for better rainfall prediction methods and sets the context for exploring the potential of machine learning in addressing this challenge. By improving forecast accuracy, this project aims to support more informed decision-making and effective risk management in weather-sensitive sectors.

## 1.2 IMPORTANCE

**1. Introduction**

Rainfall prediction plays a pivotal role in various sectors, from agriculture and water management to disaster preparedness and urban planning. The ability to accurately forecast rainfall can lead to better resource allocation, reduced risks from weather-related disasters, and optimized agricultural productivity. This project leverages machine learning (ML) to enhance the accuracy of short-term (daily) and long-term (monthly) rainfall predictions. Here, we delve into the importance of accurate rainfall prediction and the potential impact of this project across multiple domains.

**2. Agricultural Planning and Productivity**

Agriculture is highly dependent on weather conditions, particularly rainfall. Farmers need reliable rainfall forecasts to make informed decisions about planting, irrigation, and harvesting. Accurate predictions can help:

1. Optimal Planting and Harvesting: By knowing when rain is likely, farmers can time their planting and harvesting activities to ensure crops receive adequate water without being damaged by excessive rain.
2. Efficient Irrigation Management: Rainfall forecasts can guide irrigation schedules, ensuring that water resources are used efficiently and crops receive the right amount of water at the right time.
3. Pest and Disease Management: Certain pests and plant diseases are more prevalent in specific weather conditions. Accurate rainfall forecasts can help farmers anticipate and manage these threats more effectively.

Enhancing rainfall prediction through ML can lead to improved agricultural yields, reduced water wastage, and increased resilience against weather-related disruptions.

**3. Water Resource Management**

Water resource managers rely on accurate rainfall forecasts to ensure sustainable water supply and distribution. Key benefits include:

1. Reservoir Management: Accurate forecasts help in managing water levels in reservoirs, balancing the needs for water supply, flood control, and hydropower generation.
2. Flood Risk Mitigation: Predicting heavy rainfall events enables preemptive measures to manage flood risks, such as controlled water release from dams and the construction of temporary flood defenses.
3. Drought Preparedness: Reliable long-term rainfall predictions can aid in drought preparedness, ensuring that water conservation measures are implemented in time to mitigate the impact of prolonged dry periods.

Machine learning models that improve rainfall prediction accuracy can thus support more effective and sustainable water management practices.

**4. Disaster Preparedness and Response**

Natural disasters, particularly floods, are often triggered by heavy rainfall. Improved rainfall prediction has significant implications for disaster preparedness and response:

1. Early Warning System: Accurate short-term forecasts can trigger early warning systems, giving communities more time to prepare and evacuate if necessary.
2. Resource Allocation: Predictive insights allow authorities to allocate resources, such as emergency supplies and personnel, more efficiently and effectively.
3. Infrastructure Protection: Anticipating heavy rainfall can prompt preemptive actions to protect critical infrastructure, such as reinforcing levees and securing drainage systems.

By leveraging ML for more precise rainfall forecasts, this project can enhance the effectiveness of disaster response strategies, potentially saving lives and reducing economic losses.

**5. Urban Planning and Infrastructure Development**

Urban areas are particularly vulnerable to rainfall-related issues like flooding and waterlogging. Accurate rainfall prediction is crucial for:

1. Drainage System Design: Forecasts inform the design and capacity planning of urban drainage systems, ensuring they can handle expected rainfall volumes and prevent flooding.
2. Construction Planning: Knowing when rain is likely can help in scheduling construction activities, reducing delays, and ensuring the safety of construction sites.
3. Transportation Management: Rainfall forecasts can improve the management of transportation networks, allowing for preemptive measures to minimize disruptions and accidents.

Machine learning models that enhance rainfall prediction accuracy can thus contribute to more resilient urban environments.

**6. Technological Advancements in Meteorology**

Traditional weather prediction models often struggle with the complexity of weather systems, which are influenced by numerous interrelated factors. The application of ML offers several advantages:

1. Handling Complex Data: ML algorithms can process and learn from large datasets, identifying patterns and relationships that may not be apparent with traditional methods.
2. Adaptive Learning: ML models can continuously improve as more data becomes available, leading to progressively better forecasts over time.
3. Customization and Flexibility: Different ML models can be tailored to specific forecasting needs, whether it be short-term predictions for emergency response or long-term forecasts for strategic planning.

This project exemplifies how technological advancements in ML can revolutionize meteorology, providing more accurate and reliable weather forecasts.

**7. Environmental and Economic Impact**

Accurate rainfall prediction has broad environmental and economic implications:

1. Environmental Conservation: Better predictions support sustainable practices in agriculture and water management, contributing to environmental conservation efforts.
2. Economic Savings: Improved forecasts can lead to significant cost savings by reducing losses due to crop failure, inefficient water use, and disaster damages.
3. Policy and Planning: Reliable rainfall forecasts enable policymakers to make informed decisions regarding land use, resource allocation, and climate adaptation strategies.

By enhancing rainfall prediction, this project aims to contribute to environmental sustainability and economic resilience.

**8. Conclusion**

The importance of this project lies in its potential to transform rainfall prediction through the application of machine learning. By providing more accurate and reliable forecasts, the project addresses critical needs across agriculture, water management, disaster preparedness, urban planning, and meteorology. The far-reaching benefits underscore the significance of investing in advanced predictive technologies, ultimately fostering more resilient and sustainable communities.

## 1.3 SOLUTION

This project provides a comprehensive solution for improving rainfall prediction using machine learning (ML). The approach addresses the limitations of traditional weather forecasting methods by leveraging advanced ML techniques to analyze historical weather data and predict future rainfall with greater accuracy. The solution encompasses several key components:

**1. Data Collection and Preprocessing**

1. Gather extensive historical weather data, including variables such as temperature, humidity, wind speed, and past rainfall. This data is sourced from reliable meteorological databases and weather stations.
2. Preprocess the data to handle missing values, outliers, and inconsistencies. Techniques like interpolation for missing values, normalization for scaling data, and outlier detection are employed to ensure data quality.
3. Create new features from the raw data to enhance the predictive power of the ML models. Examples include lagged variables (e.g., rainfall from previous days), rolling averages, and interaction terms between different weather variables.

**2. Model Development and Training**

1. Develop and evaluate various ML models, each suited to capturing different aspects of the weather data:
2. Linear Regression: Simple and interpretable, used for establishing baseline performance.
3. Decision Trees: Capture non-linear relationships and interactions between variables.
4. Random Forests: An ensemble method that improves prediction accuracy by combining multiple decision trees.
5. Training and Validation:
   1. Train the selected models on the preprocessed data using techniques like cross-validation to ensure robustness and prevent overfitting. The data is split into training and validation sets to assess model performance.
6. Hyperparameter Tuning:
   1. Optimize the models' performance by tuning hyperparameters using grid search or random search methods. This step involves adjusting parameters such as the depth of decision trees, the number of trees in a random forest, and the architecture of neural networks.

**3. Model Evaluation and Comparison**

1. Performance Metrics:
   1. Evaluate the models using key metrics such as mean absolute error (MAE), root mean square error (RMSE), and correlation coefficient (R). These metrics provide insights into the accuracy and reliability of the predictions.
2. Model Comparison:
   * Compare the performance of different models to identify the best-performing one for both daily and monthly rainfall predictions. This involves analyzing the trade-offs between model complexity and predictive accuracy.

**4. Implementation and Deployment**

1. Prediction System:
   1. Develop a user-friendly prediction system that can generate daily and monthly rainfall forecasts. The system includes an interface for inputting new weather data and displaying the predictions.
2. Real-Time Data Integration:
   1. Integrate the prediction system with real-time weather data feeds to continuously update the models and provide up-to-date forecasts.
3. Automation and Scalability:
   1. Automate the data collection, preprocessing, model training, and prediction processes to ensure scalability and ease of maintenance. This allows the system to adapt to new data and improve over time.

**5. Applications and Impact**

1. Agricultural Advisory:
   1. Provide farmers with precise daily and monthly rainfall forecasts to optimize planting, irrigation, and harvesting schedules. This can lead to better crop yields and resource management.
2. Water Resource Management:
   1. Aid water managers in planning and distributing water resources more efficiently. Accurate predictions help in managing reservoir levels and preparing for drought or flood conditions.
3. Disaster Preparedness:
   1. Support authorities in implementing early warning systems and disaster response strategies. Accurate rainfall forecasts enable timely actions to mitigate the impacts of floods and other weather-related disasters.
4. Urban Planning:
   1. Assist urban planners in designing effective drainage systems and scheduling construction activities. Reliable forecasts help in reducing urban flooding and minimizing construction delays.

**6. Future Enhancements**

1. Model Refinement:
   1. Continuously refine the ML models by incorporating additional data sources, such as satellite imagery and atmospheric pressure readings, to improve prediction accuracy.
2. Climate Change Adaptation:
   1. Adapt the models to account for long-term climate change trends, providing more resilient and forward-looking predictions.
3. User Feedback Integration
4. Collect feedback from end-users to refine the system's usability and relevance, ensuring it meets the specific needs of different sectors.

## 1.4 CONCLUSION

The solution provided by this project harnesses the power of machine learning to significantly enhance rainfall prediction accuracy. By developing and deploying advanced ML models, the project offers valuable tools for agricultural planning, water resource management, disaster preparedness, and urban planning. The comprehensive approach ensures that the solution is robust, scalable, and adaptable to evolving weather patterns, ultimately contributing to more informed decision-making and resilient communities.

## 1.5 SOFTWARE SPECIFICATION FOR THE RAINFALL PREDICTION PROJECT

**NumPy:**

NumPy is a Python library that provides fast mathematical functions for calculations. It can handle large arrays and perform complex computations quickly.  
Pandas: Pandas is a library that helps read and write various file types and directories. It makes data processing easier and more efficient, allowing you to work with large data sets seamlessly.

**Matplotlib:**

Matplotlib is a plotting library for creating two-dimensional graphs and charts in Python. It can produce high-quality plots in various formats and is used in Python scripts, Jupyter notebooks, IPython shells, and web frameworks. It makes it easy to create a wide range of visualizations like graphs, histograms, bar charts, and scatterplots with just a few lines of code.

**Anaconda Navigator:**

Anaconda Navigator is a user-friendly software application that lets you launch applications and manage packages, environments, and channels without needing to use command-line functions. It works on Linux, Windows, and macOS. Jupyter Notebook, a key tool, is available through Anaconda Navigator.

**Jupyter Notebook:**

Jupyter Notebook is an open-source tool that allows researchers to combine code, computations, text, and multimedia in a single document. It's available through Anaconda Navigator and is widely used for data analysis and visualization tasks.

**Visual Studio Code (VS Code):**

VS Code is a powerful and popular code editor developed by Microsoft. It supports multiple programming languages and provides features like debugging, task running, and version control. With extensions, it can be tailored to meet specific development needs, making it a versatile tool for coding, including Python development for the rainfall prediction project.

**HTML:**

HTML (HyperText Markup Language) is the standard language for creating web pages. It is used to structure content on the web, including text, images, links, and other elements. For the rainfall prediction project, HTML can be used to create the front-end interface of the web application.  
  
**CSS:**

CSS (Cascading Style Sheets) is used to style HTML elements. It controls the layout, colors, fonts, and overall appearance of the web pages. CSS is essential for making the web application visually appealing and user-friendly.  
  
**Flask:**

Flask is a lightweight web framework for Python. It is used to build web applications quickly and with minimal setup. Flask is ideal for creating the backend of the rainfall prediction model, handling requests, processing data, and serving responses.  
  
**scikit-learn (sklearn):**

Scikit-learn is a Python library for machine learning. It provides simple and efficient tools for data mining and data analysis. For the rainfall prediction project, scikit-learn can be used to implement machine learning algorithms to predict rainfall based on historical data.

## 1.6 HARDWARE SPECIFICATION

**Hardware Specifications for the Rainfall Prediction Project**

**1. Computer System:**  
Processor: A multi-core processor with a minimum of 4 cores, such as Intel Core i3 or intel i5 or AMD Ryzen 5, is recommended for efficient data processing and model training.  
RAM: At least 8 GB of RAM is required, but 16 GB or more is recommended for handling large datasets and running multiple applications simultaneously.  
Storage: A minimum of 256 GB SSD is required for faster read/write speeds. Additionally, a 1 TB HDD can be used for storing large amounts of historical data.

**2. Network Connectivity:**  
Internet Connection: A stable high-speed internet connection is essential for real-time data acquisition from weather stations or satellites and for accessing cloud-based services and resources.

**3. Peripherals:**  
Monitor: A full HD monitor (1920x1080 resolution) is recommended for better visualization of data and results.  
Keyboard and Mouse: Standard ergonomic keyboard and mouse for comfortable coding and data entry.

**4. Backup and Storage Devices:**  
External Hard Drive: At least 1 TB external hard drive for data backup and storage.  
 **5. Power Supply:**  
Uninterruptible Power Supply (UPS): A UPS system is recommended to protect against data loss and hardware damage during power outages.

**6. Optional Hardware:**  
Weather Station Kit: For educational or research purposes, a personal weather station kit can be used to gather local weather data.  
Printer/Scanner: For printing reports and scanning documents related to the project.

These hardware specifications ensure that the system can efficiently handle the data processing, model training, and prediction tasks required for the rainfall prediction project, while also providing a reliable and secure environment for development and deployment.

# CHAPTER 2: LITERATURE REVIEW

## 2.1 EXISTING SYSTEM

### 2.1.1 OVERVIEW

Predicting rainfall accurately is crucial for various sectors, especially agriculture and construction. Traditional methods often struggle to capture the complexity of rainfall patterns. To address this challenge, modern systems utilize data mining techniques for improved forecasting.



2.1.1.1 Prediction Application

This report offers a simplified overview of an existing system for rainfall prediction using machine learning, highlighting the techniques utilized and their associated limitations.

Accurate rainfall prediction is crucial for industries like agriculture and construction. Traditional forecasting methods often struggle with the complexity of rainfall patterns. Modern systems address this challenge by using data mining techniques to improve predictions. This report offers a simplified overview of an existing rainfall prediction system using machine learning, focusing on the techniques employed and their limitations.

Agriculture is vital to the Indian economy, and farmers heavily rely on the monsoon season for their crops. Good soil, fertilizers, and favorable weather conditions are essential for a successful harvest. However, sudden changes in the weather can cause significant economic and physical harm to people. Predicting the rainfall accurately is crucial for farmers.



2.1.1.2 Monsoon Image

### 2.1.2 RAINFALL PREDICTION CHALLENGES

Predicting weather, especially rainfall, is a challenging task. Current systems use various data mining techniques to forecast weather conditions. These techniques include:

1. Regression: Detecting the pattern in the data.
2. Classification: Sorting data into different categories.
3. Clustering: Grouping similar data together.
4. Decision Trees: Using a tree-like model to make decisions based on data.
5. Neural Networks: Using complex algorithms modeled after the human brain to predict outcomes.

**Meteorological Data**

Rainfall related data, also known as meteorological data, includes important parameters such as:

1. Month and Year
2. Rainfall Amount
3. Wind Speed
4. Temperature
5. Humidity

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| Less resource required | Classification: Sorting data into categories can sometimes be inaccurate. |
| Optimized and scalable | Clustering: Grouping similar data may not always reflect real-world weather patterns. |
| Prediction accuracy is good. | Decision Trees: These models can be too simplistic and may not capture complex weather changes. |
| Predicts daily basis rainfall. | Problem Of overfitting And underfitting of dataset. |

Table 1. Existing System

These parameters are crucial for accurate rainfall predictions.

**Disadvantages of the Existing System**

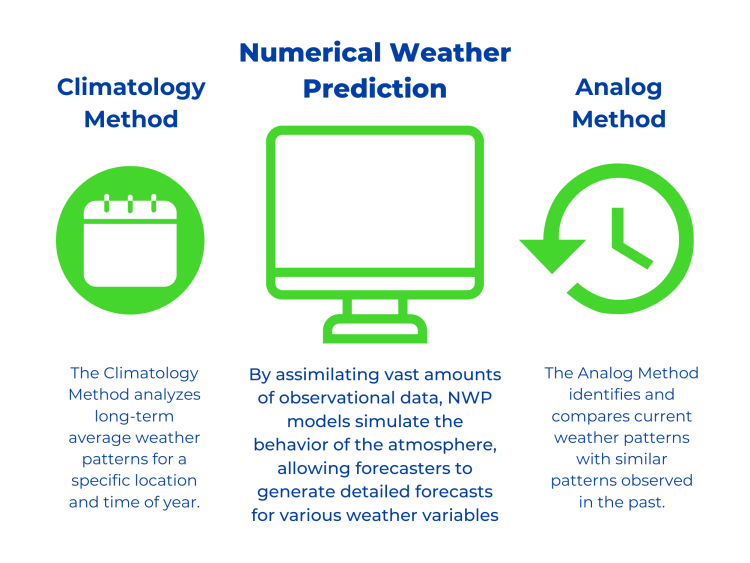
The current weather prediction system has some drawbacks:

1. Classification: Sorting data into categories can sometimes be inaccurate.
2. Clustering: Grouping similar data may not always reflect real-world weather patterns.
3. Decision Trees: These models can be too simplistic and may not capture complex weather changes.

## 2.2 PROPOSED SYSTEM

### 2.2.1 BRIEF OVERVIEW

Rainfall prediction is important for food production plan, water resource management and all activity plans in the nature. The occurrence of prolonged dry period or heavy rain at the critical stages of the crop growth and development may lead to significant reduce crop yield. India is an agricultural country and its economy is largely based upon crop productivity. Thus rainfall prediction becomes a significant factor in agricultural countries like India. Rainfall forecasting has been one of the most scientifically and technologically challenging problems around the world in the last century.



2.2.1.1 Different Rainfall Prediction Systems

While various approaches to rainfall prediction have been developed and have achieved some success, they still have limitations and room for improvement. To address these challenges in current monthly and daily rainfall prediction methods, this report presents a comprehensive methodology and a series of experiments aimed at improving prediction accuracy.

As we have already discussed, different approaches to predicting rainfall and providing effective solutions and have been relatively successful. Whereas they have a large area for enhancement and have some errors and limitations. In order to address the existing constraints and overcome the challenges in current monthly basis and daily basis rainfall prediction methodologies, the complete process of the suggested methodology, as well as a series of experiments addressing current issues in rainfall predicting tasks, are presented in this part.

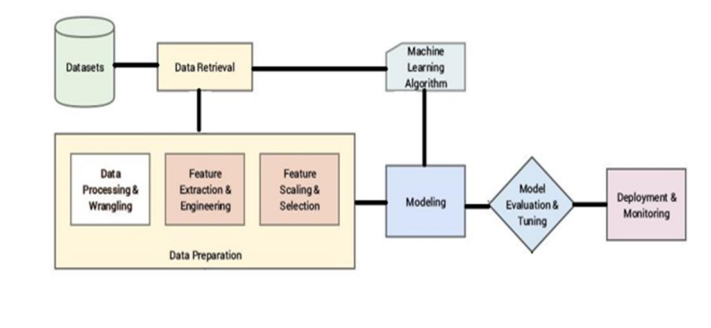
The suggested technique focuses on various critical steps of constructing a rainfall predicting model, such as the introduction of a large real time dataset and augmentation methods, the analysis of several classification and regression algorithms, and the proposal of a unique methodology for rainfall prediction methodology.

The suggested method is divided into two segments. The first part or phase is concerned with data analysis of dataset for finding relevant insights from data. Data has been gathered from various platforms such as meterological department including major regions in India. Once the dataset is completed including efficient data pattern and values samples, provide effective information for the prediction of rainfall as a precaution. The data has been input from the form using GUI webpage.

### 2.2.2 ARCHITECTURE

The proposed system is based on the given architecture in which the following steps are as follows:

1. Gather data in the dataset using different platforms.
2. Data analysis of the dataset for extracting the relevant information.
3. Picking the effective predicting algorithm with good performance from various algos.
4. Train the model by splitting the dataset and test accordingly.
5. Measure the accuracy or model evaluation.
6. Predict the rainfall by deploying the model using flask as web app.

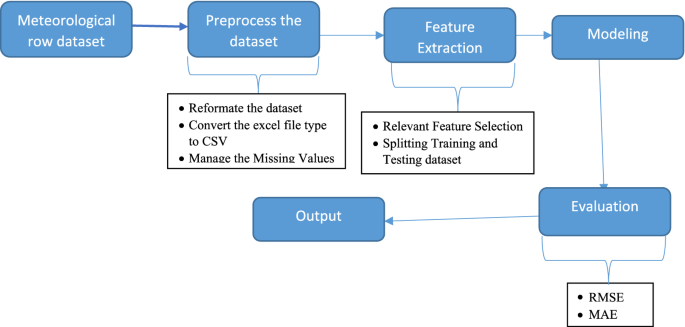
****

2.2.2.1 Rainfall Prediction Architecture

### **2.2.3 ANALYSIS AND MODEL**

The process of gathering information, organizing information, and validating it that will be done using machine learning libraries such as pandas, numpy, matplotlib is referred to as "dataset preparation” or data analysis. The goal in this context is to develop a dataset of features having data about rainfall pattern in different areas, that can be used to train and test predicting algorithms for properly identifying rainfall patterns in a particular area. Creating such a dataset necessitates a substantial amount of time and work.

1. The first step is to acquire a good amount of data that can have information about different cities and there rainfall pattern with some values
2. Second step is to analysis of data by doing exploratory data analysis and feature engineering which includes finding of the essential or dependent features by which the independent features give result. Encoding of irrelevant data has been done.
3. Third step is to split the filtered or relevant data into 80:20 ration for training and testing using various machine learning algorithms for efficient prediction.
4. Fourth step is to measure the accuracy of the model by finding the absolute error and standard deviation error values of the working algorithms such as randomforest, SVM etc.
5. Fifth step is to deploy the machine learning model to generate pickle file and predicts the rainfall.
6. Last step is for taking the data from user through HTML form and predicts the rainfall and give result using GUI web application.



2.2.3.1 Detailed Diagram

**Steps with the Algorithm:**

1. Import the libraries.

2. Import the datasets.

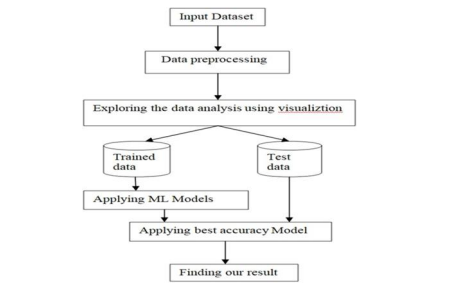
3. Define the datasets.

4. Test the datasets.

5. Run the algorithm.

6. Compare the results.

**Here is the workflow of the project**

****

**2.2.3.2 Algorithm FlowChart**

**A. Data Pre-Processing**

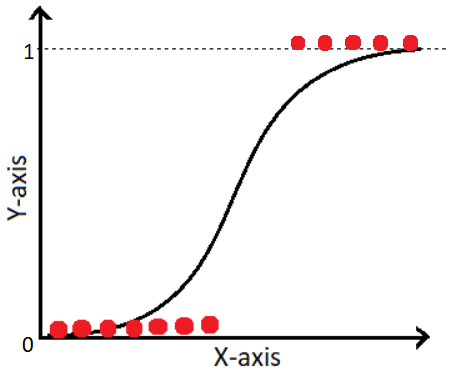
This is a crucial step in machine learning. Missing or dirty data can lower the quality of our results. Preprocessing the data helps us get better outcomes. Here are the steps for data preprocessing:

1. Removing Missing Values: Remove all entries with 0, as they are not valid.
2. Data Splitting: After cleaning the data, split it into training and testing sets. Use the training set to train the algorithm. The trained model is then used to predict outcomes based on the training data.

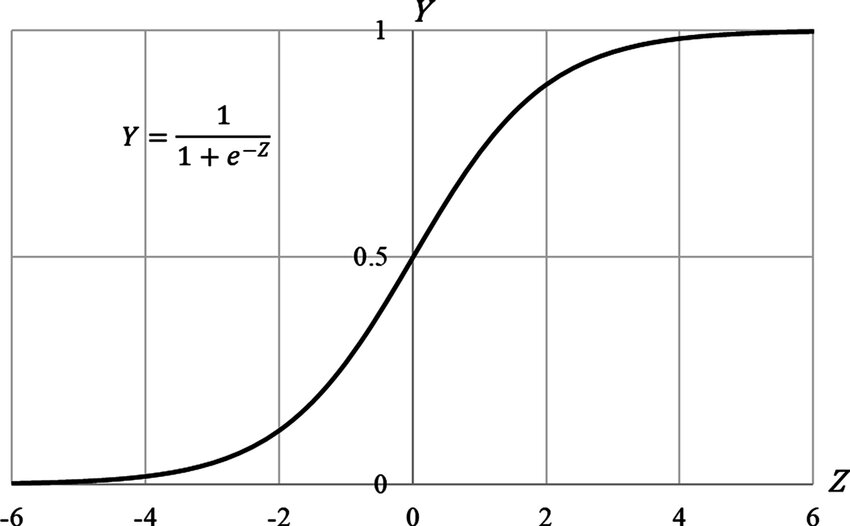
**B. Machine Learning**

Once the data is ready, we use machine learning techniques. We apply various algorithms to predict rainfall. We evaluate their performance to find the most accurate one. Here are the methods we use:

1. Logistic Regression- Logistic regression is a common machine learning algorithm. It is highly accurate and usually outputs 0s and 1s. It is used for classification tasks.



2.2.3.3 Logistic Regression Flowchart

1. Uses the sigmoid function to predict the probability of a valid or invalid class.
2. Sigmoid Function:
   * 1. 
        1. Sigmoid Function
3. Here, P is the probability, and a and b are model parameters.

2. Support Vector Machine (SVM)

SVM is a supervised machine learning algorithm. It creates a hyperplane to separate different classes. It is used for both classification and regression.

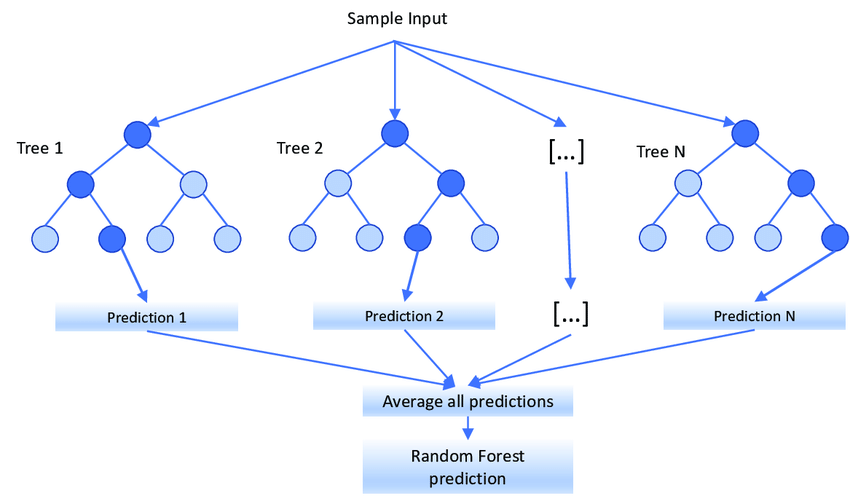


2.2.3.5 Support Vector Machine (SVM)

1. Algorithm Steps:
2. Select the hyperplane that best separates the classes.
3. Calculate the margin, the distance between the hyperplane and data points.
4. Maximize the margin to reduce misclassification.
5. Choose the class with the largest margin.
6. Margin: Distance to the negative point + Distance to the positive point.

3. Random Forest

1. This is a machine learning algorithm used for classification and regression. It is very accurate and works well with any dataset size. It improves the performance of decision trees by reducing variance.



2.2.3.6 Random Forest

1. Algorithm Steps:
   1. Select 'R' features from the total 'M' features where \( R << M \).
   2. Find the best node from the R features.
   3. Split the node into sub-nodes using the best split method.
   4. Repeat these steps until the desired number of nodes is reached.
   5. Repeat to build a forest with 'n' trees.

4. K-Neighbors Classifier:

This is an AI method used for solving problems in regression and classification. The algorithm finds the closest values around the unknown variable.



2.2.3.7 K-Nearest Neighbour

**Result**

With the Random Forest model we created, we achieved an higher accuracy of 85.67%. We used a voting classifier with an array of models, setting the voting parameter to "hard." This means the model makes predictions based on the majority vote. We tested the Random Forest model with our test data and here is the individual accuracy of the algorithms.

**C. Best Model Selected**

After training and evaluating the models, we choose the best one based on how well it performs.

**D. Deploy Model**

Deploying the model means setting it up so it can be used in real life. This usually involves integrating the model into a web application to make real-time predictions by making users interact with GUI.

## 2.3 FEASIBILITY STUDY

This phase evaluates if the project is possible and beneficial. We look at a basic plan and estimate costs. The goal is to ensure the proposed system is practical and not too expensive for the company. Understanding the main needs for the system is essential.



2.3.0.1 Feasibility Study Parts

### 2.3.1 OBJECTIVES

* 1. Identify Problems and Needs: Understand the issues and what stakeholders need.
  2. Resource Requirements: Determine the resources needed to create the system.
  3. Cost and Benefits: Evaluate the expenses and benefits of the system.
  4. Feasibility Evaluation: Consider different solutions and choose the best one for the organization.

### 2.3.2 TYPES OF FEASIBILITY

1. **Economic Feasibility**

This study checks how the system will affect the organization's finances. The company has a limited budget for research and development, so spending must be justified. We kept costs low by using mostly free technologies and only buying necessary custom products. The system fits within the budget.

* 1. Purpose: Check the financial impact of the system on the organization.
  2. Budget: Ensure the system is affordable within the company's budget.
  3. Cost Management: Use mostly free technologies and purchase only necessary customized products.
  4. Conclusion: The project is financially viable as it stays within the budget.

1. **Technical Feasibility**

This study checks the technical requirements of the system. The system should not demand too much from the current technical resources. It should work with minimal changes to existing technology. The system has modest requirements and is easy to implement.

* 1. Purpose: Assess the technical requirements for the system.
  2. Resource Misutilization: Ensure the system does not overburden the company’s technical resources.
  3. Implementation: The system should require minimal or no changes to existing technical resources.
  4. Conclusion: The project is technically feasible as it requires only modest technical resources.

1. **Social Feasibility**

This study checks how well users will accept the system. It includes training users to use the system efficiently. Users should feel comfortable with the system and see it as necessary, not as a threat.

* 1. Purpose: Determine how well users will accept the system.
  2. User Training: Ensure users can learn to use the system efficiently.
  3. Acceptance: Users should view the system as a useful tool, not a threat.
  4. Conclusion: The project is socially feasible as it is designed to be user-friendly and necessary.

1. **Benefits of the Proposed System**
   1. Accurate Predictions: Improve rainfall predictions, helping various sectors.
   2. User-Friendly Interface: Easy for users to access and understand the predictions.
   3. Cost-Effective: Uses mostly free technologies, keeping costs low.
   4. Minimal Technical Changes: Fits well with existing technical infrastructure.

**Purpose of Feasibility Study**

The feasibility study shows that the monthly and daily rainfall prediction system using machine learning is practical and beneficial. It meets economic, technical, and social requirements, making it a viable project for the organization. By leveraging machine learning, the system will provide accurate and timely rainfall predictions, benefiting agriculture, water management, and disaster preparedness.

# CHAPTER 3: SYSTEM ANALYSIS & DESIGN

## 3.1 REQUIREMENT SPECIFICATION

**What is Requirement Specification?**

Requirement specification is a detailed description of the functionalities, features, and constraints of a system or project. It outlines what the system should do, how it should perform, and any limitations it must operate within. This document is essential for developers, stakeholders, and project managers to ensure everyone has a clear understanding of the project objectives and requirements.

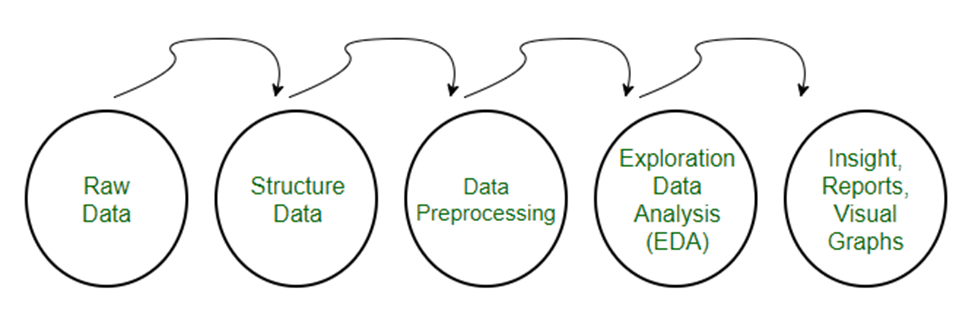
By following below requirement specifications, the rainfall prediction model will be able to provide accurate and reliable forecasts, benefiting various sectors like agriculture and water resource management.

1. **Data Collection:**

Data features such as year, month, date, evaporation, sunshine, maximum temperature, minimum temperature, humidity, wind speed, and rainfall were included. Meteorology station records the values of the environmental variable every day for each year directly from the devices in the station. The data were recorded in the Microsoft Excel into tabular format. For year and the days of the month were arranged in the row of tables related to environmental variables in the column of the table. The system should be able to collect historical weather data from reliable sources. It should also be capable of real-time data acquisition from weather stations or satellites.

1. **Data Preprocessing:**

During the data preprocessing step included the data conversion, manage missing values, categorical encoding, and splitting dataset for training and testing dataset. Since the data were raw, they contained missing values, and wrongly encoded values so that the missing values of the target variable were removed and the other features were using the mean of the data. Data should be cleaned and filtered to remove noise and inconsistencies. Missing data should be handled appropriately through imputation or other techniques.



3.1.0.1 Overview Basic Diagram

1. **Feature Extraction:**

During the data collection on which weather depend such as temperature, rainfall, wind speed, relative humidity, surface pressure. Relevant features such as temperature, humidity, wind speed, and pressure should be extracted from the data and remove the features like date, city etc.



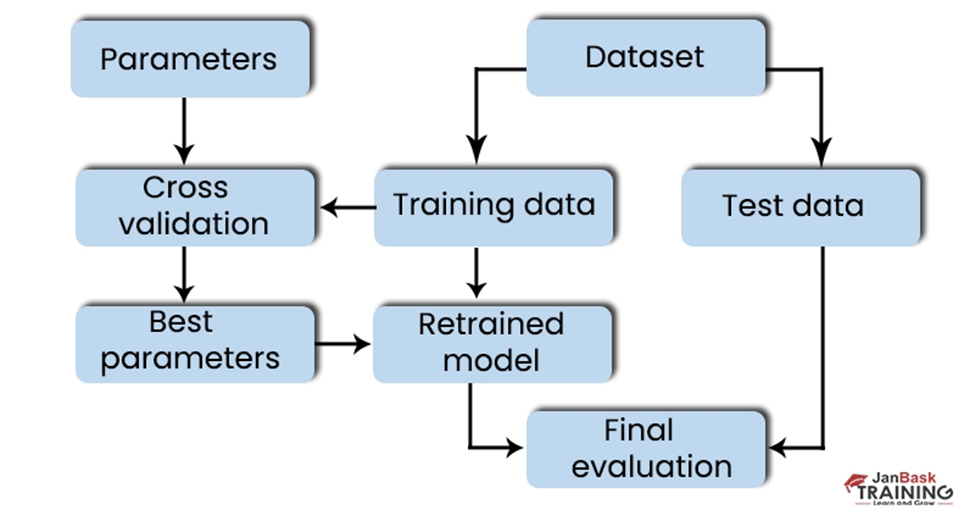
3.1.0.2 Feature Extraction

1. **Model Training:**

Model training for rainfall prediction involves several steps, including data preprocessing, feature engineering, model selection, training, and evaluation. The system should employ machine learning or statistical models for rainfall prediction. Different algorithms like regression, neural networks, or ensemble methods can be used.

1. **Model Evaluation:**

Rainfall prediction models are assessed using metrics like mean squared error (MSE), root mean squared error (RMSE), or correlation coefficient(R). These metrics quantify the disparity between predicted and actual rainfall values. Additionally, visualizations, such as scatter plots or time series graphs, aid in understanding model performance. Cross- validation techniques like MSE/RMSE and higher correlation with observed data are considered more accurate. Continual assessment and refinement of models, possibly through ensemble methods or feature engineering, are essential for improving predictive accuracy and reliability in dynamic weather conditions. Evaluation metrics such as accuracy, precision, recall, and F1-score should be used to assess the performance of the models.



**3.1.0.3 Model A**

**VI. Prediction Generation:**

Once the model is trained and validated, use it to generate prediction for future rainfall. Provide input data for the desired prediction period and utilize the trained model to forecast rainfall values. The system should generate rainfall predictions for specified time intervals (e.g., daily, weekly).

**VII. Visualization:**

Visualize the predicted rainfall value along historical data to understand trends and patterns. Provide graphical representations to communicate the predicted rainfall forecasts effectively. Interpret the predictions in the context of the location’s climate, seasonal variations, and any external factors influencing rainfall patterns. Visual representations like graphs or maps should be provided to display predicted rainfall patterns.

**Requirement Specification Types**

The requirement specification are of two types:

### 3.1.1 FUNCTIONAL REQUIREMENT SPECIFICATION

**Data Collection:**

1. **Historical Data Retrieval:** The system must retrieve historical weather data from reliable sources, including features like year, month, date, evaporation, sunshine, temperature, humidity, wind speed, and rainfall.
2. **Real-time Data Acquisition:** The system should be capable of acquiring real-time weather data from weather stations or satellites.

**Data Preprocessing:**

1. **Data Conversion:** Raw data formats should be converted into a structured format suitable for analysis.
2. **Missing Value Handling:** The system must handle missing values in the dataset by either removing them or replacing them using appropriate techniques like mean imputation.
3. **Noise Reduction:** Data should be cleaned and filtered to remove any noise or inconsistencies.
4. **Categorical Encoding:** Categorical variables should be encoded into numerical format for analysis.
5. **Dataset Splitting:** The dataset should be split into training and testing sets for model development and evaluation.

**Feature Extraction:**

**Identification of Relevant Features:** Relevant weather features such as temperature, humidity, wind speed, and pressure must be extracted from the dataset.

**Model Training:**

1. **Algorithm Selection:** The system should employ machine learning or statistical models like regression, neural networks, or ensemble methods for rainfall prediction.
2. **Model Training:** Models must be trained using the preprocessed data, including steps like feature engineering and evaluation.

**Model Evaluation:**

1. **Evaluation Metrics:** Rainfall prediction models should be assessed using metrics like mean squared error (MSE), root mean squared error (RMSE), correlation coefficient (R), accuracy, precision, recall, and F1-score.
2. **Visualization:** Visualizations such as scatter plots, time series graphs, and maps should be used to aid in understanding model performance.

**Prediction Generation:**

**Forecasting:** Once trained and validated, the system should generate predictions for future rainfall based on input data for specified time intervals (e.g., daily, weekly).

**Visualization:**

1. **Graphical Representation:** Predicted rainfall values should be visualized along with historical data using graphs or maps to identify trends and patterns effectively.
2. **Interpretation:** Predictions should be interpreted in the context of location-specific climate, seasonal variations, and external factors influencing rainfall patterns.

### 3.1.2 NON-FUNCTIONAL REQUIREMENT SPECIFICATION

**Performance:**

1. **Data Processing Speed:** The system should preprocess and analyze data efficiently to minimize processing time.
2. **Model Training Time:** Model training should be completed within a reasonable time frame to ensure timely predictions.
3. **Prediction Response Time:** The system should generate rainfall predictions quickly to provide timely information to users.

**Reliability:**

1. **Data Accuracy:** The system must ensure the accuracy of collected and processed data to produce reliable predictions.
2. **Model Accuracy:** Rainfall prediction models should be highly accurate to instill confidence in users.
3. **System Availability:** The system should be available for data collection, preprocessing, model training, and prediction generation without significant downtime.

**Scalability:**

1. **Data Scalability:** The system should be able to handle large volumes of data as the dataset grows over time.
2. **Model Scalability:** Models should scale effectively with increasing data size to maintain prediction accuracy.
3. **User Scalability:** The system should support multiple users accessing and utilizing prediction results simultaneously.

**Security:**

1. **Data Privacy:** Ensure the privacy and confidentiality of collected weather data to comply with privacy regulations.
2. **Model Security:** Implement measures to protect trained models from unauthorized access or tampering.
3. **System Access Control:** Restrict access to sensitive system components and data to authorized personnel only.

**Usability:**

1. **User Interface Intuitiveness:** The system interface should be user-friendly and intuitive for easy navigation and interaction.
2. **Documentation Clarity:** Provide clear and comprehensive documentation to guide users in system operation and data interpretation.
3. **Error Handling:** The system should handle errors gracefully, providing informative error messages to users when issues arise.

**Compatibility:**

1. **Data Source Compatibility:** Ensure compatibility with various data sources to collect weather data from different sources seamlessly.
2. **Model Compatibility:** Models should be compatible with different machine learning frameworks and libraries for flexibility.
3. Platform Compatibility: The system should be compatible with different operating systems and platforms to accommodate diverse user requirements.

**Maintainability:**

1. **Code Maintainability:** Write clean, well-documented code to facilitate system maintenance and future enhancements.
2. **Model Updatability:** Allow for easy updating or replacement of trained models to incorporate new data or improve prediction accuracy.
3. **System Upgradability:** The system architecture should be modular and scalable to support future upgrades and enhancements.

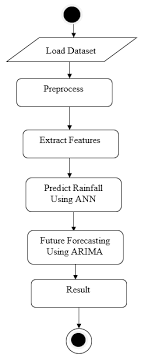
|  |  |
| --- | --- |
| 1. **Functional Requirements Specification** | **Non Functional Requirements Specification** |
| **Modal Evaluation** | **Maintainability and performance** |
| **Modal Training** | **Usability** |
| **Prediction Generation** | **Compatibility** |
| **Data Visualization** | **Security** |
| **Feature Extraction** | **Scalability** |

**Table 2. Functional and Non Functional**

## 3.2 FLOWCHARTS

A flowchart is a visual representation of a process, system, or algorithm. It uses standardized symbols to depict different steps, decisions, inputs, and outputs within a process. Flowcharts help in understanding and analyzing the flow of activities and the sequence in which they occur. **The key elements of a flowchart include:**

* 1. Oval (Terminator): Represents the start or end of the process.
  2. Rectangle (Process): Denotes a step or action in the process.
  3. Diamond (Decision): Indicates a decision point where the process can branch based on a yes/no question or condition.
  4. Arrow (Flow Line): Shows the direction of the flow of the process.
  5. Parallelogram (Input/Output): Represents data input or output.



3.2.0.1 Flow Chart

### 3.2.1 USE OF FLOWCHART IN RAINFALL PREDICTION MODEL

Flowcharts are particularly useful in developing a rainfall prediction model as they provide a clear, visual way to outline the process. Here’s how a flowchart can be used in the context of a rainfall prediction model:

Flowcharts can help define the steps involved in collecting, preprocessing, analyzing, and predicting rainfall data. This ensures that all necessary stages are included and properly ordered.

By visualizing the workflow, a flowchart helps team members understand the sequence of operations and the dependencies between different tasks. This clarity is crucial for collaborative projects.

In the model development process, decision points such as choosing the appropriate machine learning algorithm, handling missing data, or deciding when to retrain the model are crucial. Flowcharts highlight these decision points, aiding in better planning and execution.

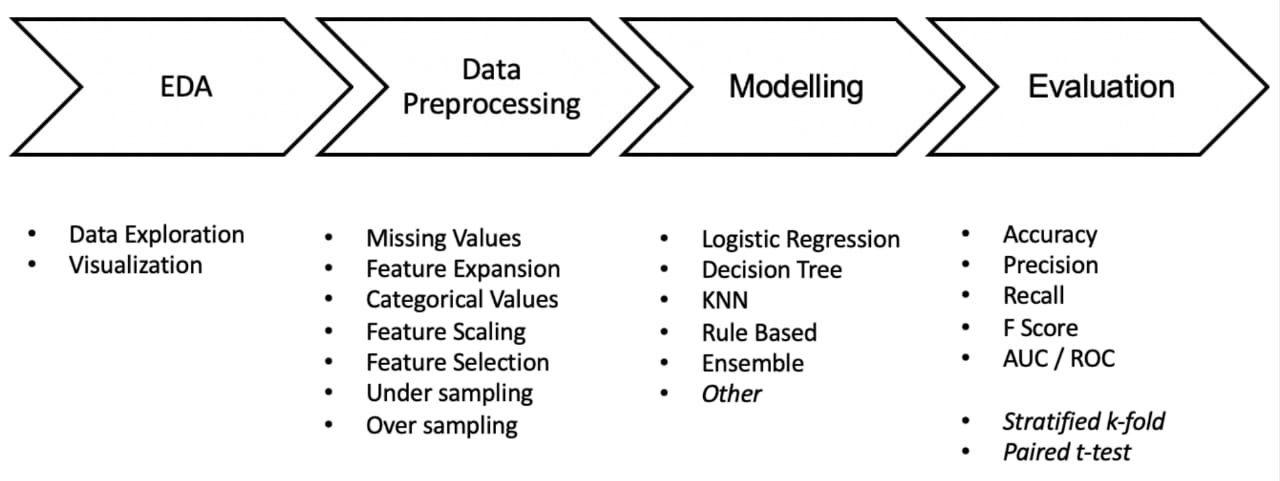
Flowcharts serve as an effective communication tool among stakeholders, including data scientists, developers, and domain experts. They provide a common language to discuss the steps and logic involved in the model.

When issues arise or optimizations are needed, flowcharts help in tracing the process flow, identifying bottlenecks, and pinpointing areas that require improvement.

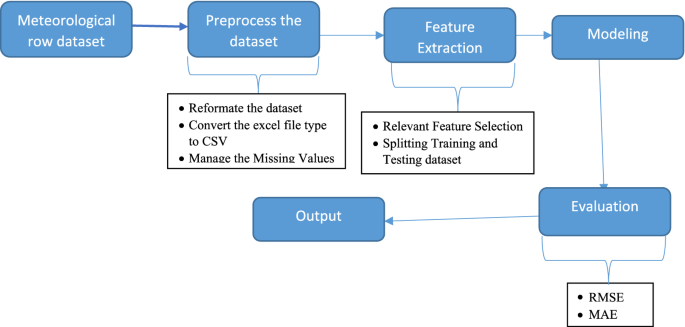
A flowchart serves as a part of the project documentation, capturing the logic and sequence of the process. This is valuable for future reference, training new team members, or transitioning the project to other teams.

This flowchart provides a simplified overview of the steps involved in developing and using a rainfall prediction model, highlighting the iterative nature of model training and evaluation.

Flowcharts for rainfall prediction with model training with different evaluation such as Logistic Regression, Random forest regression , KNN or prediction using SVM.



3.2.1.1 Basic Methodology



3.2.1.2 Methodology Digram

## 3.3 DATA FLOW DIAGRAM

A Data Flow Diagram (DFD) is a graphical representation used to illustrate the flow of data within a system. It shows how data moves from input to processing to output, providing a clear picture of the system's processes, data stores, and interactions with external entities.

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

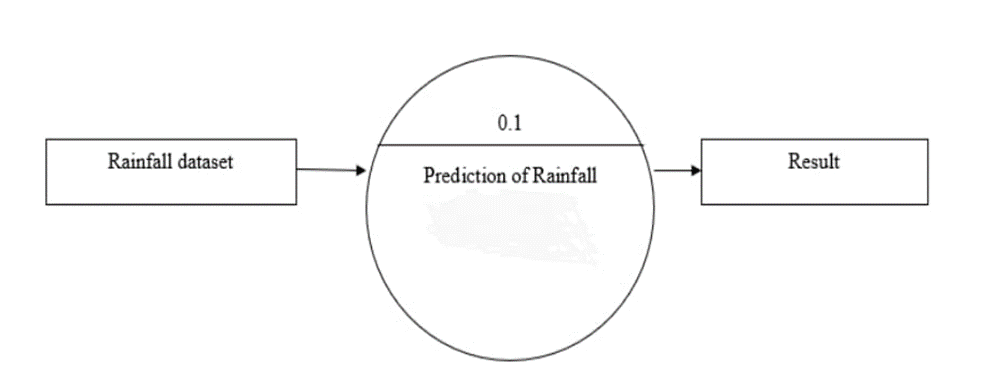
A Data Flow Diagram (DFD) is a powerful tool for visualizing the process of predicting rainfall using machine learning. It helps in breaking down the complex process into simpler, more understandable components. Here’s how a DFD can be helpful:

1. **Clarifying System Components and Processes:**
   1. Input Data: The DFD can clearly show where the input data (e.g., historical weather data, real-time data from weather stations, satellite data) comes from.
   2. Data Preprocessing: It can outline steps involved in cleaning and preparing the data, such as handling missing values, data normalization, and feature extraction.
   3. Model Training: The DFD can illustrate the process of training the machine learning model, including the algorithms used (e.g., regression, neural networks).
   4. Prediction Generation: It shows how the trained model is used to make predictions for daily and monthly rainfall.
   5. Output Data: Finally, it can depict how the predicted data is outputted, displayed, or used for further analysis.
2. **Visualizing Data Flow:**
   1. Data Sources to Storage: Shows how data flows from various sources to data storage systems.
   2. Data Storage to Processing Units: Illustrates how data moves from storage to different processing units for cleaning and preprocessing.
   3. Processing to Model Training: Depicts the flow from data processing units to model training components.
   4. Model Training to Prediction: Shows the transition from trained models to generating predictions.
3. 3. Identifying Bottlenecks and Improvements:
   1. By visualizing the entire data flow, stakeholders can easily identify bottlenecks or inefficiencies in the process.
   2. Helps in optimizing the system by understanding where improvements or additional resources are needed.
4. Enhancing Communication:
   1. A DFD provides a common visual language for all stakeholders, including data scientists, engineers, and non-technical members, facilitating better communication and understanding.
   2. Helps in explaining the complex process of rainfall prediction to stakeholders who might not be familiar with machine learning or data processing.
5. Supporting System Design and Development:
   1. Aids in the initial design phase of the rainfall prediction system by providing a clear blueprint of the entire process.
   2. Useful for developers to understand the system requirements and the flow of data through various components.

### 3.3.1 LEVEL 0

A Level 0 Data Flow Diagram (DFD) provides a high-level overview of the entire system, illustrating the main processes, external entities, data stores, and data flows involved in the rainfall prediction model.

A Level 0 flowchart for a rainfall prediction system outlines the high-level process from data acquisition to prediction output. The flow begins with the Data Collection phase, where historical and real-time weather data are gathered from various sources such as weather stations, satellites, and sensors. This data is then processed in the Data Preprocessing step, which involves cleaning, normalizing, and feature extraction to prepare it for analysis. Following this, the Model Training phase utilizes machine learning algorithms like KNN, Decision Trees, Logistic Regression, SVM, and Lasso Regression to train predictive models on the preprocessed data. Once trained, the models move to the Prediction Generation step, where they forecast rainfall based on new input data. The predictions are then sent to the Visualization and Output stage, which displays the forecasted rainfall data through graphical interfaces, maps, and dashboards. The system concludes with a Feedback Loop for continuous improvement, incorporating user feedback and updated data to refine the predictions over time. This flowchart encapsulates the core functions of a rainfall prediction system, ensuring accurate and actionable forecasts.



3.3.1.1 Level 0 DFD

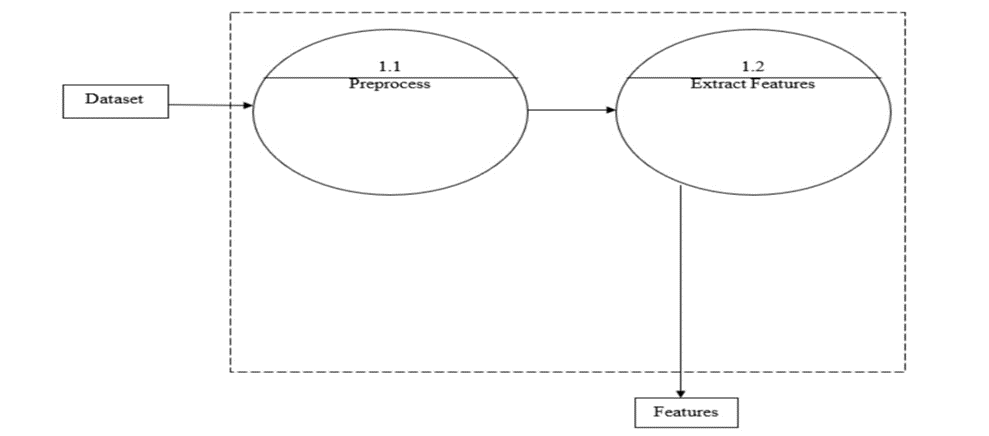
### 3.3.2 LEVEL 1

A Level 1 Data Flow Diagram (DFD) provides a more detailed breakdown of the system's main processes that were outlined in the Level 0 DFD. In the context of a rainfall prediction model using machine learning, the Level 1 DFD would elaborate on the primary functions and interactions within the system.

Creating a Level 1 flowchart for a rainfall prediction system involves outlining the primary processes from data collection to prediction output. The flowchart begins with the "Start" node, leading to "Data Collection," where historical and real-time weather data are gathered from sources such as weather stations, satellites, and sensors. Next is "Data Preprocessing," which includes cleaning, normalization, and feature extraction to prepare the data for analysis.

Following preprocessing is "Model Selection and Training," where machine learning models such as KNN, Decision Trees, Logistic Regression, SVM, and Lasso Regression are selected and trained on historical weather data. The next step, "Model Evaluation," involves validating the models using performance metrics like mean squared error and R-squared to ensure accuracy.

After evaluation, the "Prediction" node uses the best-performing model to forecast future rainfall. The final steps are "Visualization," where the predictions are graphically represented, and "End," marking the completion of the process. This flowchart ensures a structured approach to developing and deploying a rainfall prediction system.



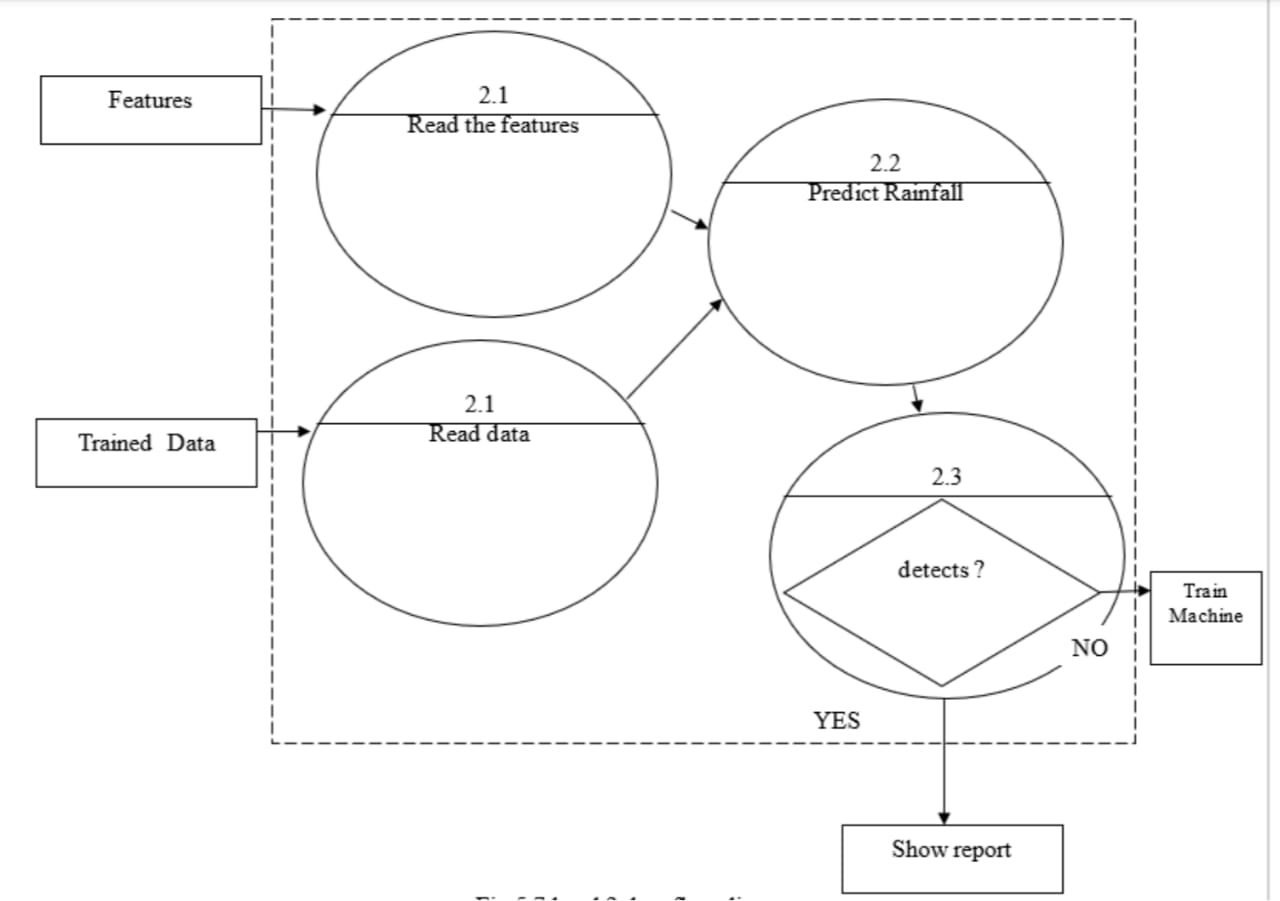
3.3.2.1 Level 1 DFD

### 3.3.3 Level 2

A Level 2 Data Flow Diagram (DFD) dives deeper into the processes detailed in the Level 1 DFD, providing an even more granular view of the system's operations. In the context of a rainfall prediction model using machine learning, the Level 2 DFD would break down each primary function into sub-processes, showing more specific steps and data interactions.

Creating a Level 2 flowchart for a rainfall prediction system involves detailing the key processes in a structured, step-by-step manner. The flowchart begins with Data Collection, where historical weather data and real-time observations are gathered from sources like weather stations and satellites. Next, the data moves to Data Preprocessing, involving steps such as data cleaning, normalization, and feature extraction to ensure quality input. This is followed by Feature Selection, where relevant variables like temperature, humidity, and wind speed are chosen.

The Model Training phase comes next, where machine learning algorithms such as KNN, Decision Trees, Logistic Regression, SVM, and Lasso Regression are trained on the preprocessed data. Model Validation follows, using techniques like cross-validation to evaluate performance metrics. Once validated, the model is used for Prediction Generation, producing rainfall forecasts for specified intervals. The final step is Visualization and Reporting, where predictions are displayed using graphs and maps for easy interpretation by users.



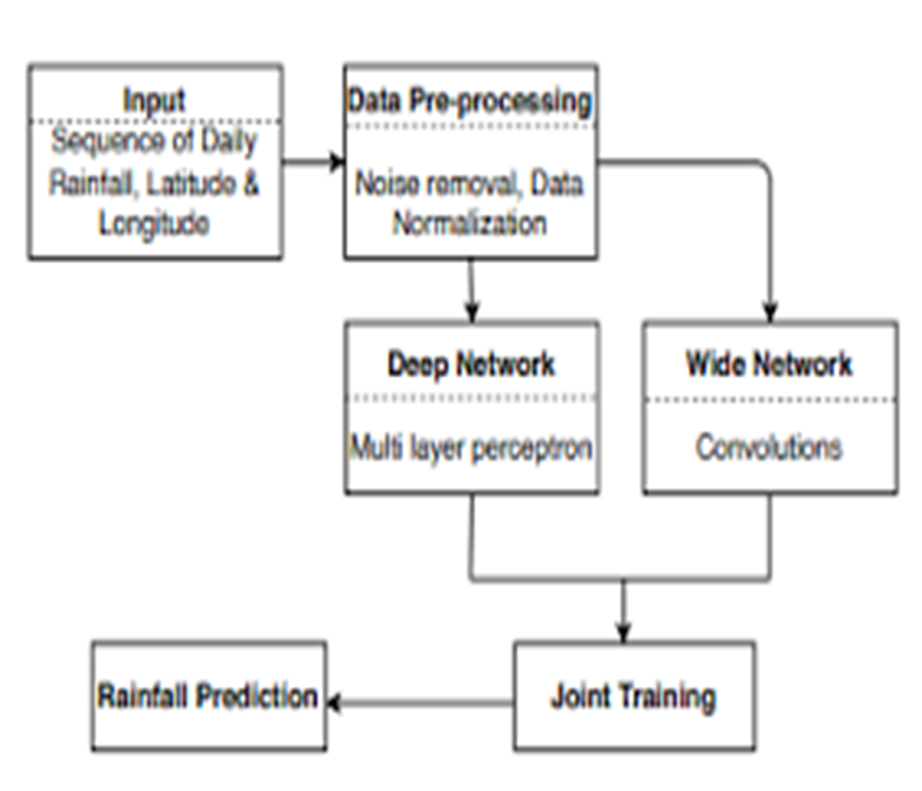
3.3.3.1 Level 2 DFD

### 3.3.4 Level 3

In a Level 3 Data Flow Diagram (DFD), we delve even deeper into the system's processes, breaking down the activities identified in the Level 2 DFD into more detailed sub-processes. For a rainfall prediction model using machine learning, the Level 3 DFD would provide a granular view of each step involved in the system's operation.

For example, in the context of data preprocessing, the Level 3 DFD might illustrate the specific tasks involved, such as data cleaning, handling missing values, encoding categorical variables, and splitting the dataset for training and testing. Similarly, for model training, the Level 3 DFD could outline the various algorithms used, parameter tuning steps, and the process of evaluating model performance.

Overall, the Level 3 DFD offers a highly detailed representation of the system's functionality, allowing for a comprehensive understanding of each component and its interactions.



3.3.4.1 Level 3 DFD

## 3.4 UML Diagrams

UML stands for Unified Modeling Language, and it's like a common language used in the field of object-oriented software engineering. It's managed by a group called the Object Management Group.

The main aim of UML is to provide a standard way to create models of software. Right now, UML has two big parts: a Meta-model and a notation. In the future, they might add some kind of method or process to UML.

Basically, UML helps with specifying, visualizing, building, and documenting software systems. It's not just for software though; it can also be used for business modeling and other types of systems.

UML is like a set of best practices for modeling complex systems. It's really important for developing object-oriented software and helps with the whole software development process.

When people use UML, they mainly use pictures and diagrams to show how they're designing software projects.

### 3.4.1 GOALS:

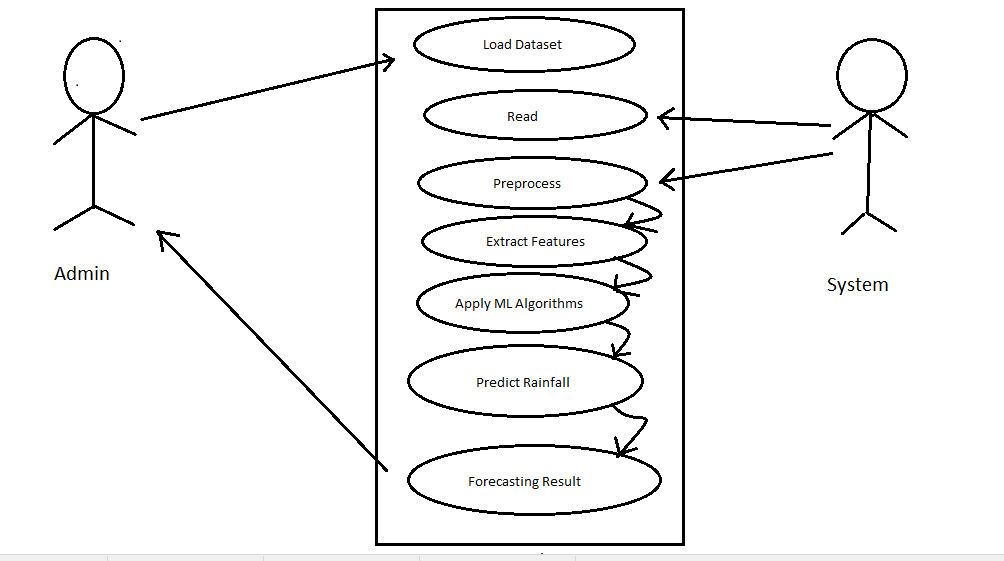
The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

### 3.4.2 USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a 21 graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases.

The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

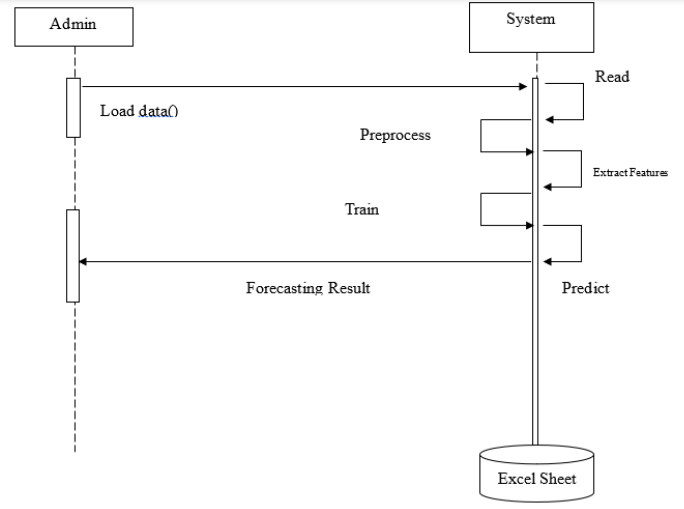


3.4.2.1 Use case diagram

### 3.4.3 SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

A sequence diagram in the context of a rainfall prediction model outlines the interaction between different components or actors within the system over time. It shows how data and control messages flow between objects to accomplish the rainfall prediction task.

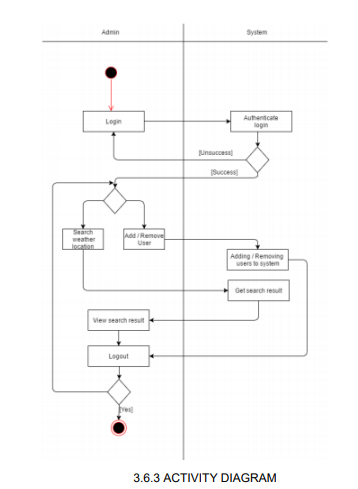


**3.4.3.1 Sequence diagram**

### 3.4.4 ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-bystep workflows of components in a system. An activity diagram shows the overall flow of control.

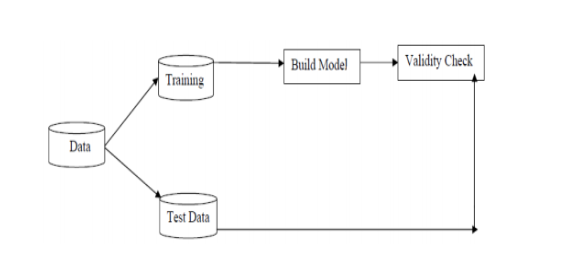
An activity diagram for a rainfall prediction model using machine learning outlines the step-by-step workflow of the system. It visually represents the sequence of activities and the flow of control from one activity to another.



3.4.4.1 Activity Diagram

## 3.5 DESIGN AND TEST STEPS

Designing a rainfall prediction system involves multiple interconnected components aimed at accurately forecasting precipitation levels. At the core of the design is the data collection mechanism, which gathers historical weather data and real-time observations from various sources such as weather stations, satellites, and sensors. This data undergoes preprocessing, including cleaning, normalization, and feature extraction, to prepare it for analysis. Feature engineering techniques may involve extracting relevant meteorological variables like temperature, humidity, and wind speed, as well as temporal features such as seasonality and trends.



3.5.0.1 Basic design

Once the data is prepared, the system employs machine learning algorithms such as linear regression, random forest, or neural networks for prediction modeling. These models are trained on historical data and validated to ensure their accuracy and reliability in forecasting rainfall. The design also incorporates techniques for model evaluation and validation, including cross-validation and performance metrics such as mean squared error and R-squared.

In addition to the predictive modeling component, the system includes visualization tools to present the forecasted rainfall data in an understandable format. Graphical representations, maps, and dashboards help stakeholders interpret the predictions and make informed decisions based on the forecasted rainfall patterns.

Start

|--- Data Collection

| |--- Gather historical weather data

| |--- Collect real-time weather data

|--- Data Preprocessing

| |--- Clean data (remove outliers, errors)

| |--- Handle missing data

| |--- Normalize or scale data

|--- Feature Engineering

| |--- Extract relevant features

| |--- Create lag features

| |--- Generate aggregated features

|--- Model Selection

| |--- Choose appropriate model

| |--- Select hyperparameters

|--- Model Training

| |--- Split data into training and validation sets

| |--- Train model on training data

|--- Prediction Generation

| |--- Provide input data for prediction period

| |--- Utilize trained model to generate forecasts

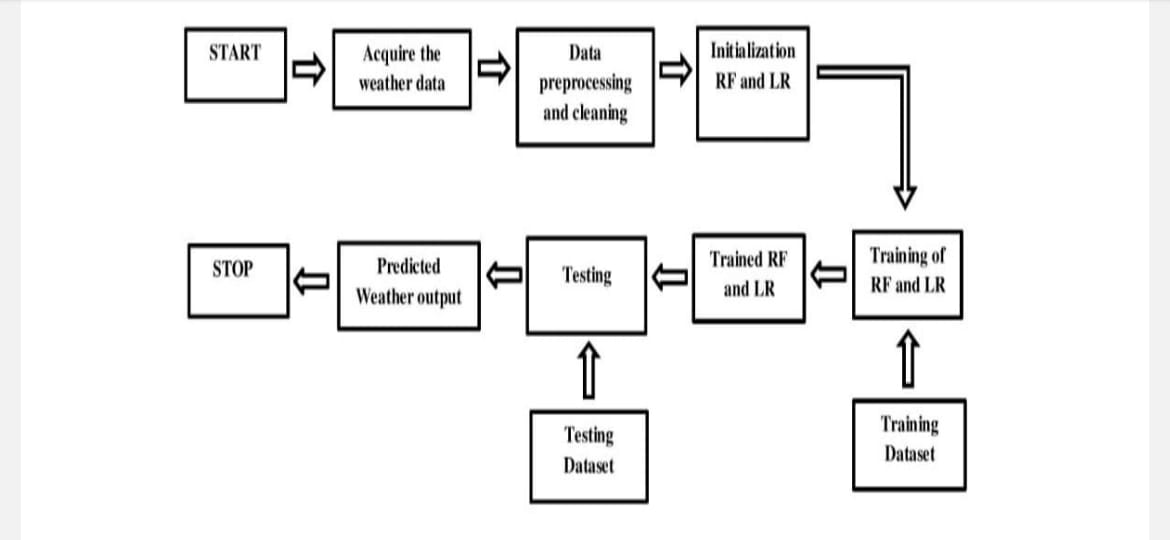
|--- Visualization and Interpretation

| |--- Visualize predicted rainfall values

| |--- Interpret predictions in context of climate and external factors

End

Furthermore, the design encompasses a feedback loop mechanism to continuously improve the accuracy of the predictions. This involves monitoring the model's performance, collecting feedback from users, and updating the model with new data to adapt to changing environmental conditions and refine its predictions over time. Overall, the design of a rainfall prediction system integrates data collection, preprocessing, modeling, visualization, and continuous improvement processes to provide accurate and actionable forecasts for various applications such as agriculture, water resource management, and disaster preparedness.



3.5.0.2 Algorithm Diagram

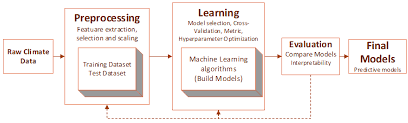
The test steps for a rainfall prediction system encompass a comprehensive approach to validating its functionality, accuracy, and reliability. Firstly, data collection mechanisms are tested to ensure the acquisition of accurate and timely weather data from various sources. Preprocessing steps, including data cleaning, normalization, and feature extraction, are then verified to confirm the preparation of data for analysis. Next, machine learning models for rainfall prediction undergo rigorous testing, including training on historical data and evaluation using validation techniques such as cross-validation. The predictive models' performance metrics, such as mean squared error and R-squared, are assessed to ensure accurate forecasting. Additionally, visualization tools for presenting forecasted rainfall data are tested for clarity and effectiveness in conveying information to users. Finally, a feedback loop mechanism is validated to ensure the system can continuously improve its predictions based on user feedback and updated data. Through these test steps, the rainfall prediction system is thoroughly evaluated to deliver reliable forecasts for various applications.

## 3.6 ALGORITHM AND PSEUDOCODE

What is an Algorithm?

An algorithm is a step-by-step procedure or formula for solving a problem. It is a finite sequence of well-defined instructions, typically used to perform a task or solve a specific problem. In computer science, algorithms are used for data processing, calculations, automated reasoning, and other tasks.

### 3.6.1 ALGORITHM FOR RAINFALL PREDICTION USING MACHINE LEARNING



3.6.1.1 Algorithm

Here's a simplified algorithm to predict rainfall amounts for a particular month or year using machine learning:

**Step-by-Step Algorithm:**

1. Data Collection:

* 1. Collect historical weather data including features such as year, month, day, temperature, humidity, wind speed, and past rainfall amounts.
  2. Ensure the dataset includes several years of data to capture patterns.

2. Data Preprocessing:

* 1. Clean the data to handle any missing values or anomalies.
  2. Normalize or standardize the features to ensure they are on a comparable scale.
  3. Encode any categorical variables, such as weather conditions.

3. Feature Selection:

Identify and select relevant features that impact rainfall prediction, such as temperature, humidity, wind speed, and previous rainfall amounts.

4. Data Splitting:

Split the dataset into training and testing sets. Typically, use 70-80% of the data for training and the remaining 20-30% for testing.

5. Model Selection:

Choose a machine learning model suitable for regression tasks, such as Linear Regression, Decision Trees, Random Forests, or Neural Networks.

6. Model Training:

Train the selected machine learning model using the training dataset.

7. Model Validation:

1. Validate the model using the testing dataset to ensure it generalizes well to unseen data.
2. Evaluate the model using metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), or R-squared.

8. Hyperparameter Tuning:

Optimize the model's hyperparameters using techniques such as Grid Search or Random earch to improve performance.

9. Prediction:

Use the trained and validated model to predict the rainfall amount for the desired month or year.

10. Visualization:

Visualize the predicted rainfall amounts along with historical data to analyze trends and patterns.

11. Deployment:

Deploy the model into a production environment where it can receive new input data and provide predictions.

### 3.6.2 PSEUDOCODE FOR RAINFALL PREDICTION ALGORITHM:

```python

# Step 1: Data Collection

def collect\_data():

# Collect historical weather data

data = load\_weather\_data()

return data

# Step 2: Data Preprocessing

def preprocess\_data(data):

# Clean data

data = clean\_missing\_values(data)

# Normalize features

data = normalize\_features(data)

# Encode categorical variables

data = encode\_categorical\_variables(data)

return data

# Step 3: Feature Selection

def select\_features(data):

features = ['temperature', 'humidity', 'wind\_speed', 'past\_rainfall']

target = 'rainfall\_amount'

return data[features], data[target]

# Step 4: Data Splitting

def split\_data(features, target):

X\_train, X\_test, y\_train, y\_test = train\_test\_split(features, target, test\_size=0.2)

return X\_train, X\_test, y\_train, y\_test

# Step 5: Model Selection

def select\_model():

model = RandomForestRegressor()

return model

# Step 6: Model Training

def train\_model(model, X\_train, y\_train):

model.fit(X\_train, y\_train)

return model

# Step 7: Model Validation

def validate\_model(model, X\_test, y\_test):

predictions = model.predict(X\_test)

mse = mean\_squared\_error(y\_test, predictions)

rmse = np.sqrt(mse)

return rmse

# Step 8: Hyperparameter Tuning

def tune\_model(model, X\_train, y\_train):

param\_grid = {'n\_estimators': [100, 200], 'max\_depth': [10, 20]}

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

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

return grid\_search.best\_estimator\_

# Step 9: Prediction

def predict\_rainfall(model, new\_data):

prediction = model.predict(new\_data)

return prediction

# Step 10: Visualization

def visualize\_predictions(predictions, actual):

plt.plot(predictions, label='Predicted')

plt.plot(actual, label='Actual')

plt.legend()

plt.show()

# Main Execution

data = collect\_data()

preprocessed\_data = preprocess\_data(data)

features, target = select\_features(preprocessed\_data)

X\_train, X\_test, y\_train, y\_test = split\_data(features, target)

model = select\_model()

trained\_model = train\_model(model, X\_train, y\_train)

rmse = validate\_model(trained\_model, X\_test, y\_test)

tuned\_model = tune\_model(trained\_model, X\_train, y\_train)

new\_data = load\_new\_data()

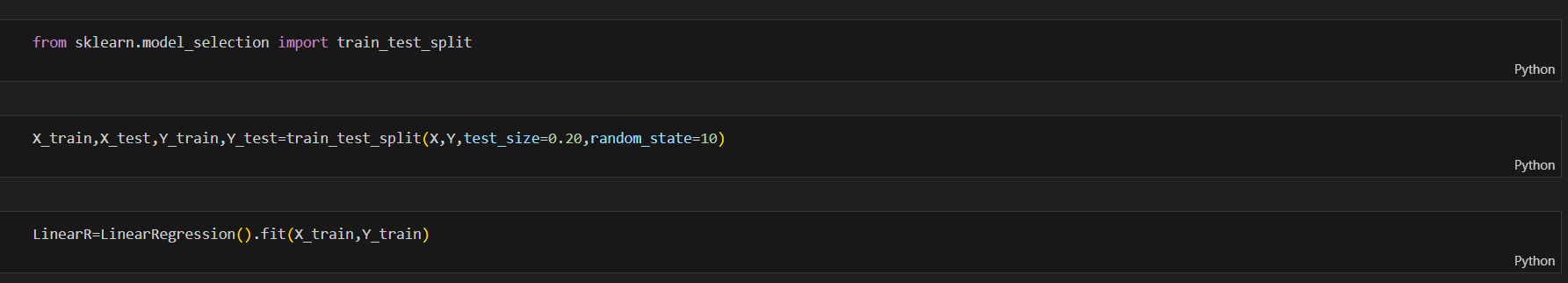
rainfall\_prediction = predict\_rainfall(tuned\_model, new\_data)

visualize\_predictions(rainfall\_prediction, actual\_rainfall\_data)

```

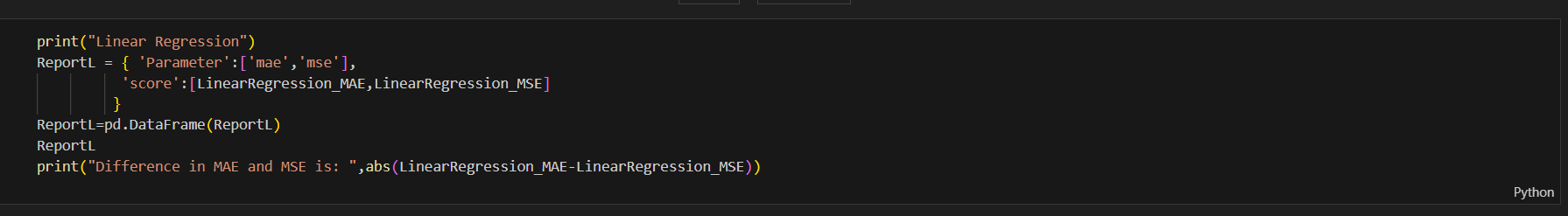
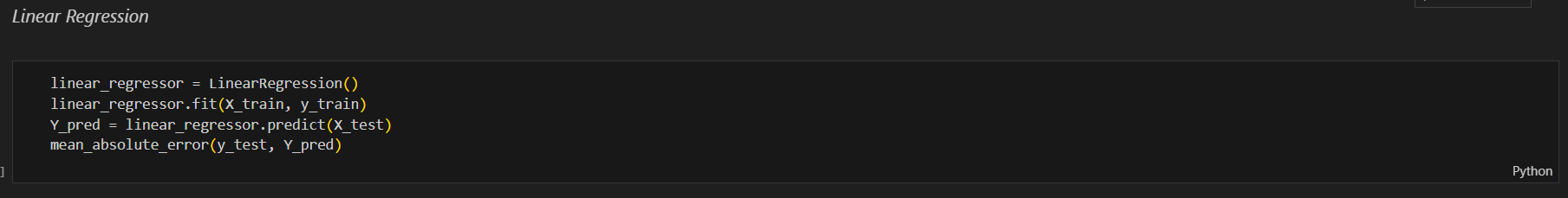
### 3.6.3 ACTUAL CODE FOR ANALYSIS

Split the data for training and accuracy for prediction.

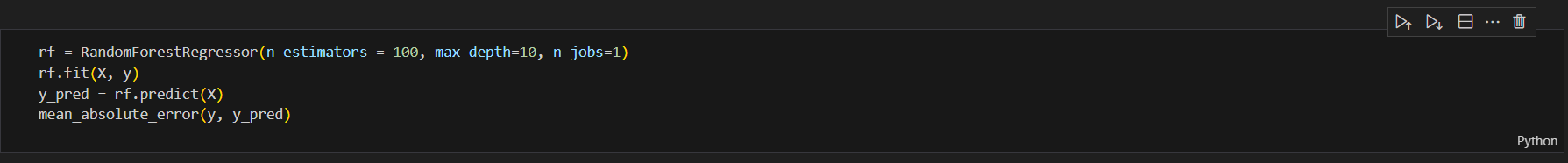
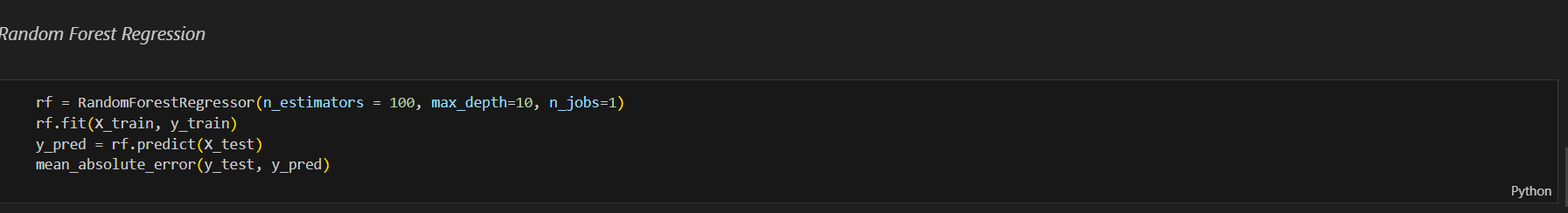


3.6.3.1 Pseudo Snippet Splitting

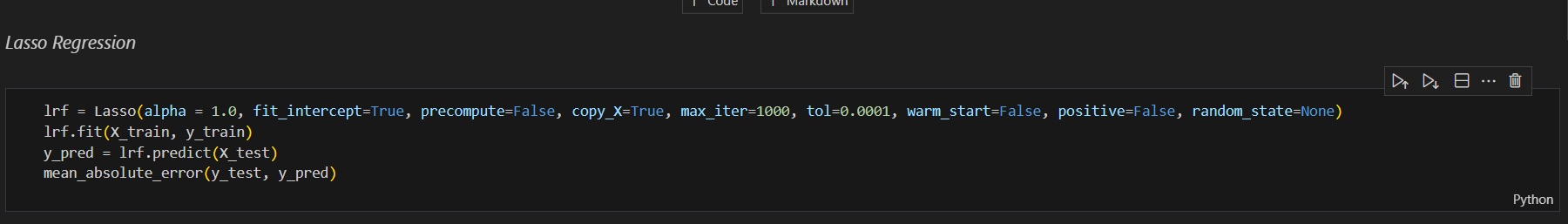
Linear regression is a statistical method used to model the relationship between a dependent variable and one or more independent variables. It assumes a linear relationship between the variables, aiming, to find the best-fit line that minimizes the difference between observed and predicted values. Linear regression is a statistical technique employed to model the relationship between a dependent variable and one or more independent variables. It assumes that this relationship can be approximated with a straight line. The primary goal of linear regression is to identify the best-fit line through the data points, which minimizes the sum of the squared differences between the observed values and the predicted values. This line is defined by the equation y=mx+b in the case of simple linear regression, where 𝑦 is the dependent variable, 𝑥 is the independent variable, 𝑚 is the slope of the line, and 𝑏 is the y-intercept. When dealing with multiple independent variables, the method extends to multiple linear regression, which uses the equation 𝑦=𝑏0+𝑏1x1+𝑏2𝑥2+...+𝑏𝑛𝑥𝑛. The coefficients 𝑏0, 𝑏1, ..., 𝑏𝑛 are calculated to minimize the prediction errors. Linear regression is widely used due to its simplicity, interpretability, and effectiveness in identifying and quantifying relationships between variables, making it a foundational tool in predictive modeling and data analysis across various fields such as economics, biology, and engineering.



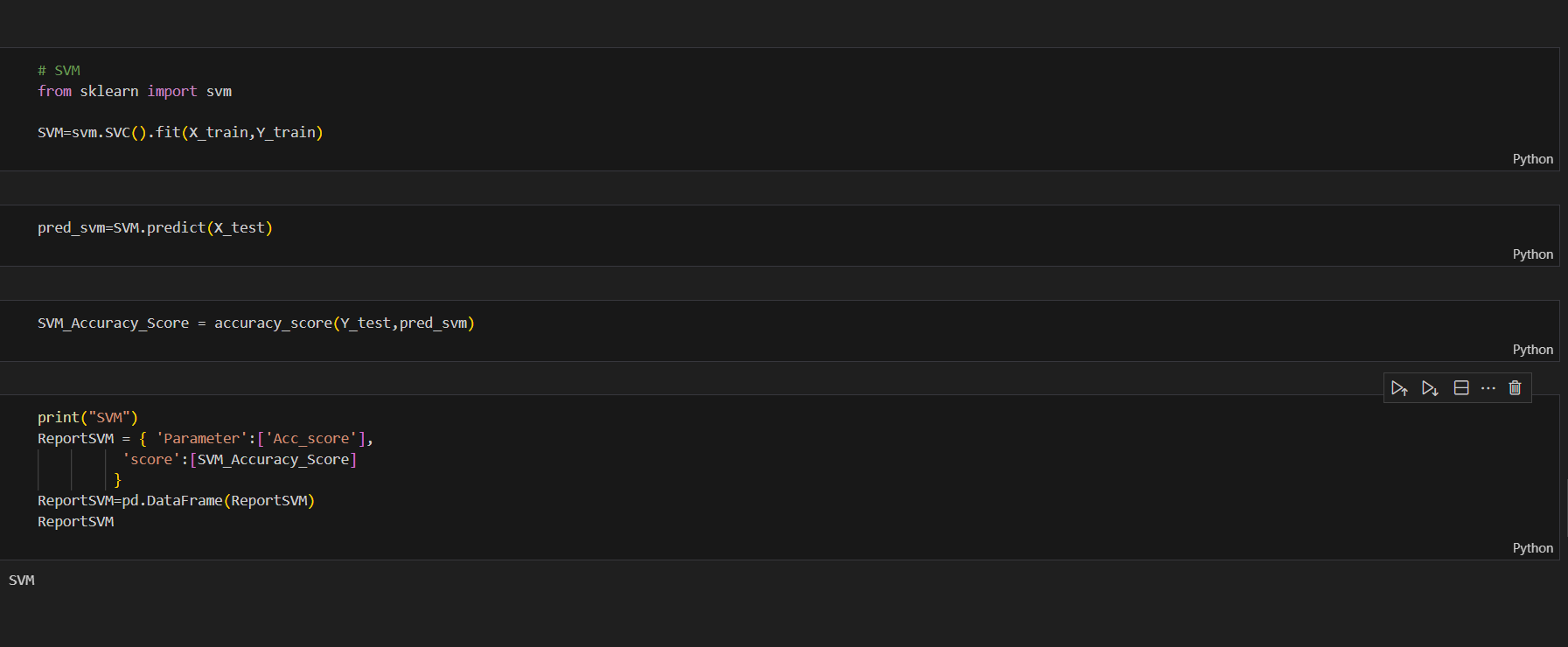
3.6.3.2 Pseudo Snippet Linear regression

Random forest regression is an ensemble learning technique designed to predict continuous outcomes by leveraging multiple decision trees. This method involves constructing a multitude of decision trees during training and generating predictions by averaging the results from each tree. The key advantage of random forest regression is its ability to improve predictive accuracy and robustness. By combining the predictions of many trees, it reduces the risk of overfitting, which is a common issue with individual decision trees. This technique also handles large datasets with higher dimensionality effectively, making it a powerful tool for various regression tasks. Random forest regression is widely used in fields like finance, healthcare, and environmental modeling, where accurate and reliable predictions are essential.

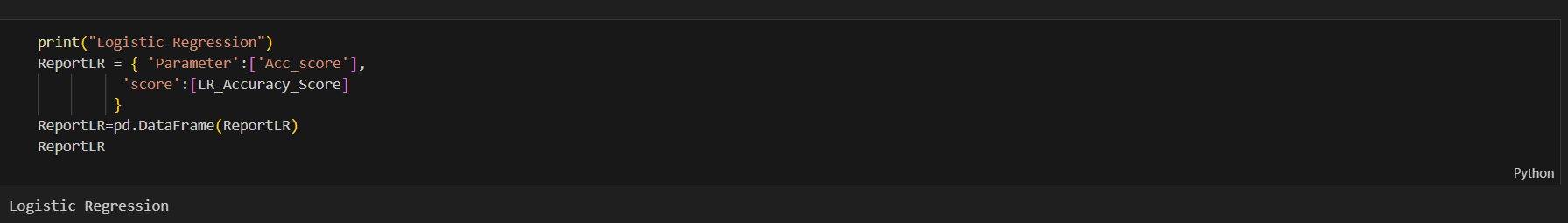
3.6.3.3 Pseudo Snippet Random Forest

Lasso regression, or Least Absolute Shrinkage and Selection Operator regression, is a linear regression method that incorporates a regularization term, known as the L1 penalty, into the model's cost function. This penalty encourages the regression coefficients to shrink towards zero, effectively imposing a constraint on their magnitude. By doing so, Lasso regression promotes sparsity in the model, leading to the selection of a subset of the most important features while discarding less relevant ones. This feature selection mechanism not only simplifies the model but also enhances interpretability by highlighting the most influential predictors. Lasso regression is particularly useful in scenarios with high-dimensional data, where identifying the most significant variables is crucial for building parsimonious and accurate predictive models. Its ability to strike a balance between model complexity and interpretability makes it a valuable tool in various fields, including economics, genetics, and machine learning.

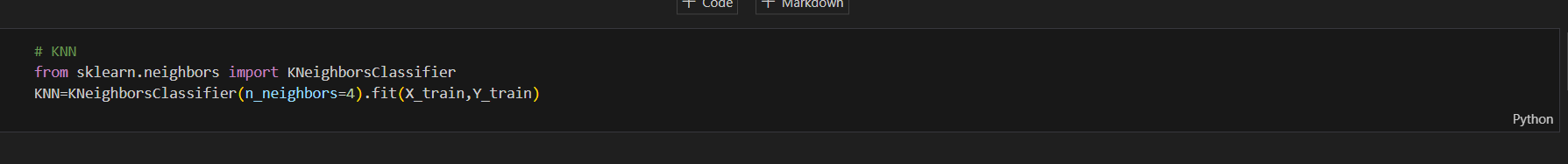
3.6.3.4 Pseudo Snippet Lasso Regression

SVM, or Support Vector Machine, is a powerful supervised learning algorithm employed for classification and regression tasks. It operates by identifying the optimal hyperplane that effectively separates data points into distinct classes while maximizing the margin between them. This hyperplane is positioned in such a way that it maximizes the distance between the nearest data points from each class, known as support vectors. By finding this optimal decision boundary, SVM aims to achieve robust classification or regression results that generalize well to unseen data. SVM is particularly effective in scenarios where the data is nonlinearly separable or high-dimensional, thanks to its ability to use kernel functions to map the input data into a higher-dimensional space. Its versatility, robustness, and ability to handle complex datasets make it a popular choice in various fields, including image recognition and, text classification.

3.6.3.5 Pseudo Snippet SVM

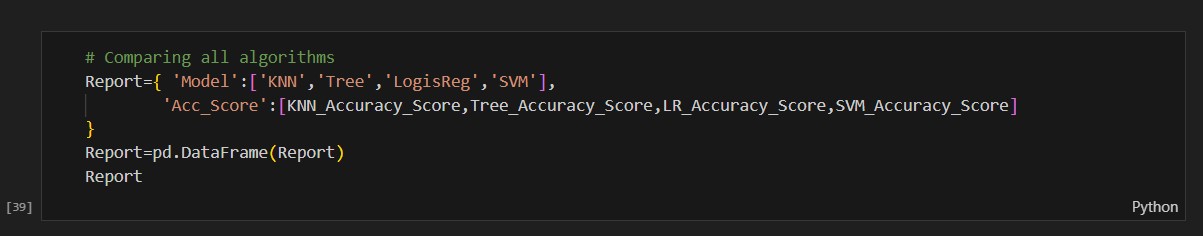
Logistic regression is a statistical technique utilized for binary classification tasks, where the outcome variable has only two possible classes. It estimates the probability of the binary outcome by modeling the relationship between the independent variables and the probability of belonging to a particular class. Unlike linear regression, logistic regression employs the logistic function, also known as the sigmoid function, to transform the output into a probability between 0 and 1. This transformation ensures that the predicted probabilities are bounded and interpretable. Logistic regression calculates the log odds of the probability of the event occurring, and its coefficients represent the change in the log odds for a unit change in the independent variable. Widely used in fields such as medicine, finance, and marketing, logistic regression is valued for its simplicity, interpretability, and effectiveness in binary classification tasks.

3.6.3.6 Pseudo Snippet Logistic Regression

K-Nearest Neighbors (KNN) is a straightforward yet powerful machine learning algorithm employed for classification and regression tasks. It operates on the principle of similarity, where the label or value of a data point is predicted based on the majority class or average of its k nearest neighbors in the feature space. KNN is non-parametric, meaning it does not make explicit assumptions about the underlying data distribution. Instead, it relies on the proximity of data points to make predictions. The choice of k, the number of neighbors considered, influences the algorithm's performance and flexibility. KNN is particularly useful for datasets with complex decision boundaries or when the underlying data distribution is unknown. While computationally intensive for large datasets, KNN is valued for its simplicity, versatility, and effectiveness in various domains, including pattern recognition, recommendation systems, and anomaly detection.

3.6.3.7 Pseudo Snippet KNN

In comparing and predicting rainfall, various machine learning models such as K-Nearest Neighbors (KNN), Decision Trees, Logistic Regression, Support Vector Machines (SVM), and Lasso Regression can be employed. KNN relies on similarity measures to predict rainfall by considering the characteristics of nearby data points. Decision Trees partition the feature space into segments, making predictions based on the majority class in each segment. Logistic Regression estimates the probability of rainfall occurrence based on input variables, while SVM identifies an optimal hyperplane to separate rainfall and non-rainfall instances. Lasso Regression adds a regularization term to promote sparsity in the model, potentially improving interpretability. Each model has its strengths and weaknesses; KNN is simple but sensitive to noise, Decision Trees are prone to overfitting, Logistic Regression assumes linearity, SVM can handle high-dimensional data, and Lasso Regression facilitates feature selection. The choice of model depends on factors such as dataset characteristics, interpretability requirements, and predictive accuracy goals.

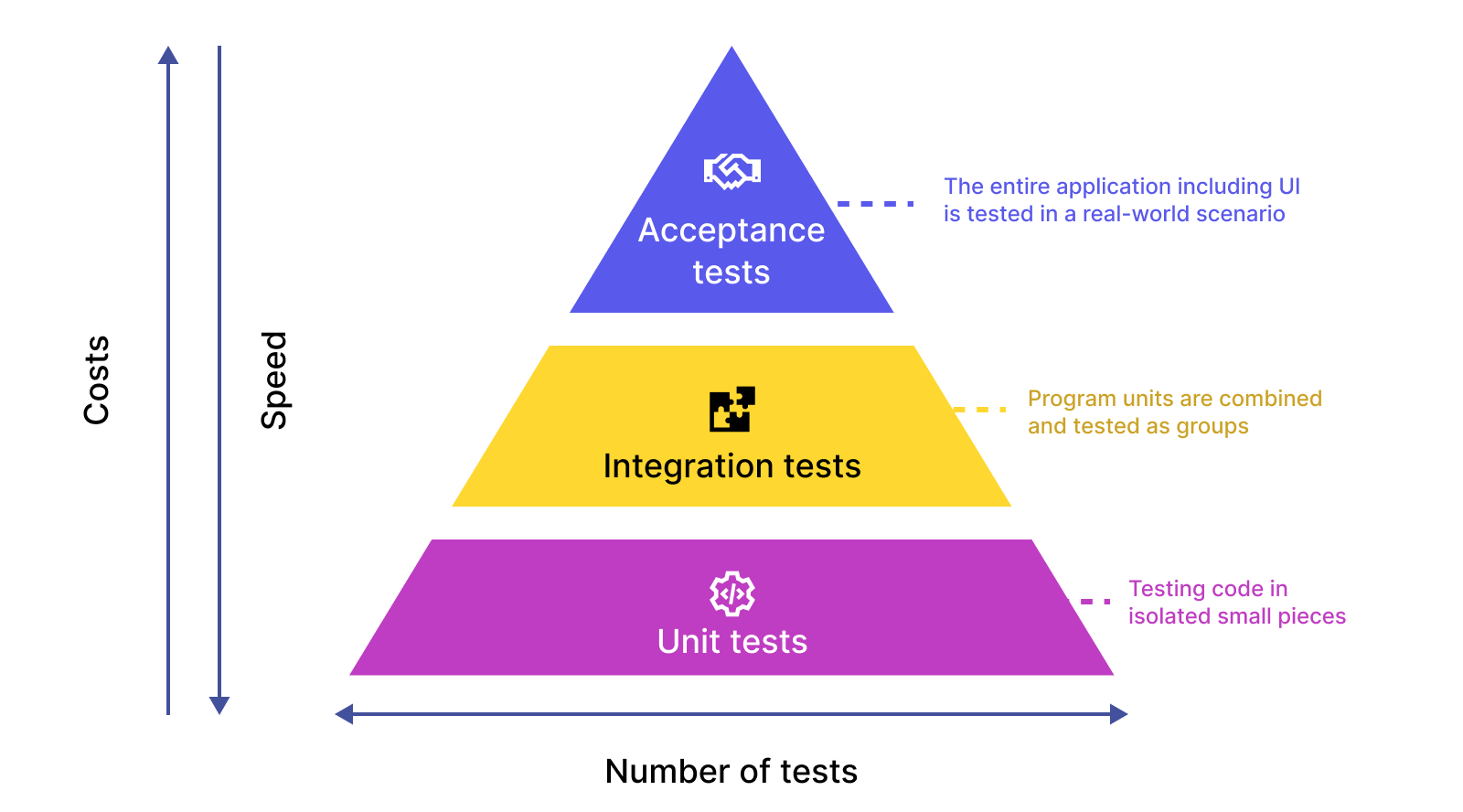


3.6.3.8 Pseudo Snippet Comparison algorithms



## 3.7 TESTING

A test plan outlines the strategy, resources, and schedule for testing a software item to identify discrepancies between actual and expected outcomes, ensuring quality. It involves verification (meeting specifications) and validation (meeting user needs) throughout development.



3.7.0.1 Testing Graph

### 3.7.1 VERIFICATION

Verification is a crucial phase in the software development lifecycle aimed at ensuring that a product aligns with its initial requirements and specifications. It involves systematically reviewing and testing the product to confirm that it behaves as intended and adheres to the defined design and implementation standards. Verification activities focus on assessing the correctness of individual components and their interactions, as well as validating that the design meets the specified requirements. Through thorough testing and inspection, verification helps identify defects early in the development process, reducing the likelihood of costly errors and rework later on. By confirming that the product meets its initial objectives, verification provides assurance to stakeholders and facilitates the delivery of high-quality software solutions that meet user expectations.

### 3.7.2 VALIDATION

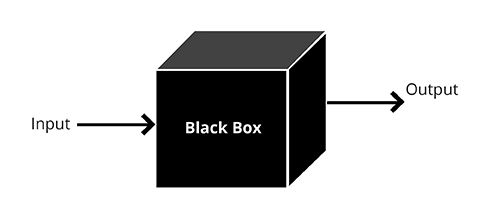
Validation is a critical stage in the software development lifecycle that ensures the final product meets the specified requirements and effectively addresses customer needs. Unlike verification, which focuses on confirming that the product conforms to initial development requirements, validation assesses whether the software functions as intended in the real-world context and delivers value to end users. This process involves testing the software in realistic environments to validate its functionality, usability, performance, and reliability. By soliciting feedback from stakeholders and end users, validation verifies that the software meets user expectations, addresses business objectives, and fulfills its intended purpose. Ultimately, validation is essential for ensuring customer satisfaction, enhancing product quality, and validating the success of the software development effort by confirming that it delivers tangible benefits and meets the needs of its intended audience.

### 3.7.3 BLACK BOX

Black box testing, also referred to as functional testing, is a method used to evaluate a system's functionality without delving into its internal structure or implementation details. This approach focuses solely on testing inputs and observing outputs to ensure that the system behaves as expected based on specified requirements. Black box testing is particularly valuable for assessing the system's adherence to user requirements and its compatibility with the intended interface. By treating the system as a "black box," testers can simulate various user interactions and scenarios to validate that the system responds correctly and produces the expected outputs. This method helps identify discrepancies between expected and actual behavior, enabling teams to uncover defects or inconsistencies in the system's functionality and address them promptly. Ultimately, black box testing plays a crucial role in verifying the system's functionality and ensuring that it meets user expectations.

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| Quick setup: Test cases can be designed as soon as the requirements are defined, potentially before any code is written. | Limited Coverage: It may not uncover all the potential issues, particularly those related to the internal implementation or structure. |
| Comprehensive testing: Test cases are based on specifications, which helps in achieving comprehensive coverage of all functional requirements. | High-level focus: It focuses more on what the system does rather than how it does it, potentially missing optimization opportunities or structural problems. |
| Non-technical testers: It allows individuals without deep technical knowledge to participate in the testing process, broadening the pool of available testers. | Internal bugs: Bugs related to the internal structure or performance issues might go undetected. |
| Improved usability: Testers can identify usability issues and ensure the interface is user-friendly. | High-level focus |

Table 3. Black Box Testing



3.7.3.1 Black Box Testing

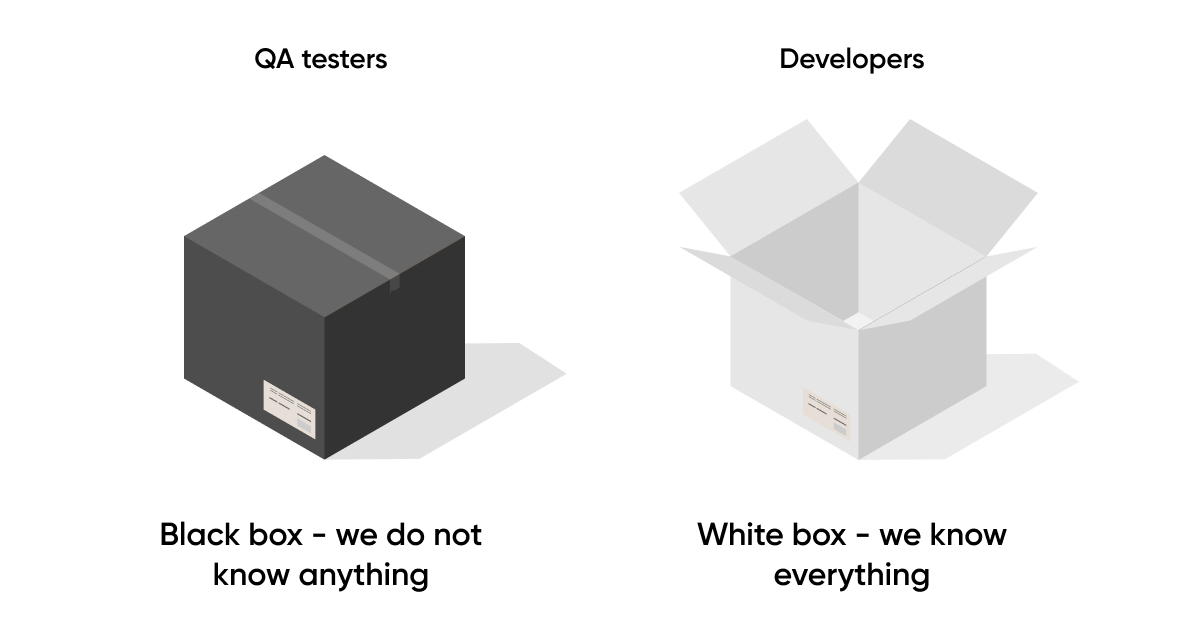
### 3.7.4 WHITE BOX

White box testing, also referred to as structural or glass box testing, scrutinizes the internal structure and workings of a system, including its code logic and pathways. Unlike black box testing, which evaluates the system's functionality without delving into its internal

|  |  |
| --- | --- |
| **Advantages** | Disadvantages |
| Code Optimization: Identifies redundant or inefficient code, leading to optimization and improvement in performance. | High Complexity: Understanding and testing complex code requires significant time and expertise. |
| Unit Testing: Facilitates unit testing, allowing developers to test individual components or modules for specific functionality. | Time-Consuming: Writing detailed test cases and achieving high code coverage can be time-consuming. |
| Early in Development: Bugs can be detected and fixed early in the development process, potentially reducing the cost and time required to address them later. | Frequent Changes: Frequent changes to the code can lead to continuous updates in test cases, making it hard to maintain. |
| Security Vulnerabilities: Can help identify security vulnerabilities within the code, such as buffer overflows, which might not be evident through black box testing. | Missed Scenarios: Tests might not cover real-world scenarios and edge cases that a user might encounter. |

Table 4. White Box Testing

mechanisms, white box testing requires knowledge of the system's internal architecture. Testers analyze the codebase, identify potential vulnerabilities, and design test cases to exercise specific code paths, conditions, and branches. By examining the system's internal workings, white box testing aims to ensure that the implemented logic behaves as intended and adheres to predefined requirements. While white box testing is primarily used for verification, validating that the system meets its specified design and development requirements, it complements black box testing, which focuses on validating that the system meets user expectations and fulfills its intended purpose in real-world scenarios.

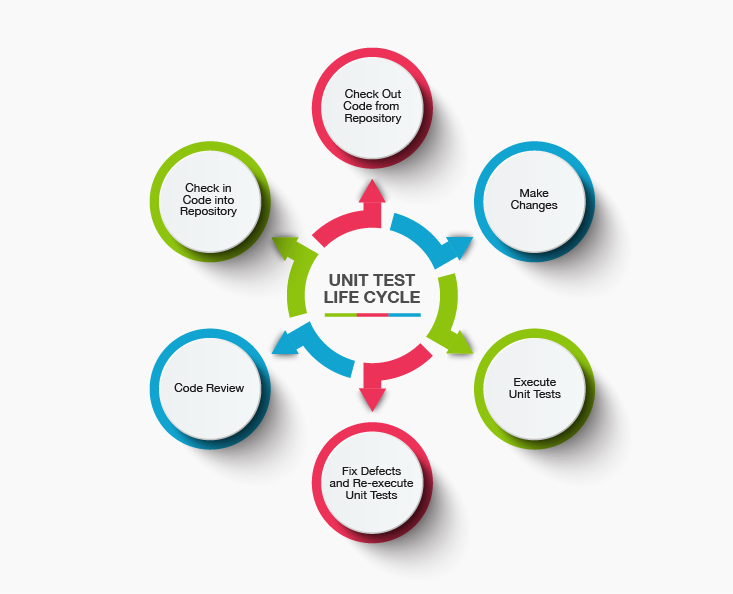


3.7.4.1 White Box Testing

### 3.7.5 TYPE OF TESTING

#### UNIT TESTING

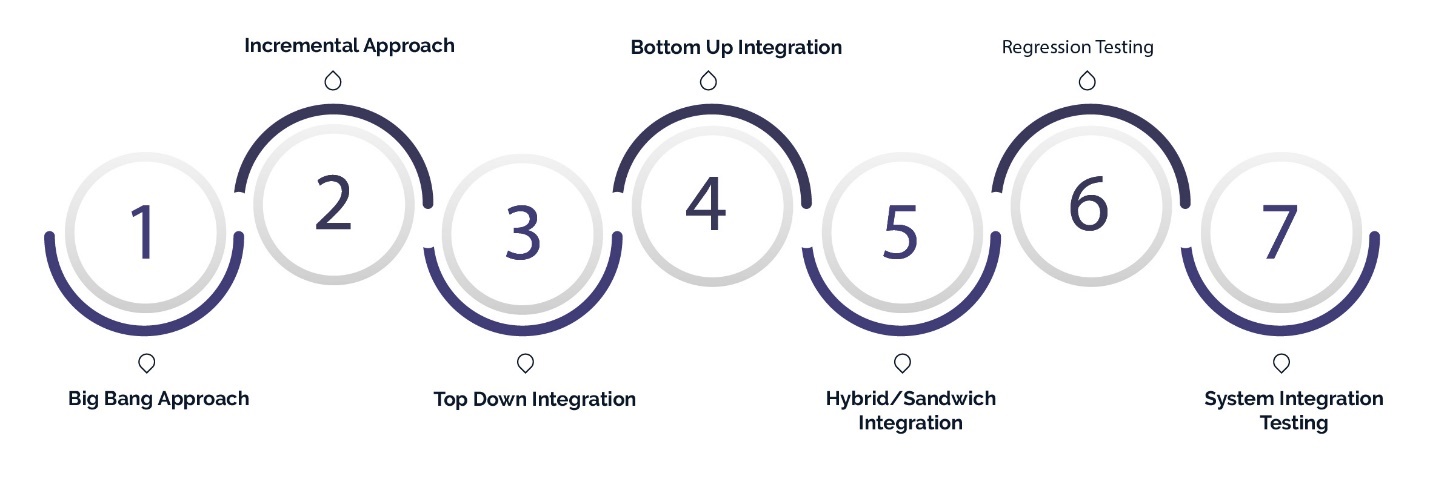
It is akin to meticulously inspecting each ingredient before baking a cake. It involves testing individual components, or units, of software in isolation to verify that they function as intended. Similar to ensuring each ingredient contributes to the overall taste and texture of the cake, unit testing confirms that each software module performs its designated task correctly. Test cases are designed to validate the behavior of specific functions, methods, or classes, exercising various input values and edge cases to assess their functionality. By identifying and fixing defects at the unit level, unit testing contributes to overall software quality and reliability. It provides developers with rapid feedback on the correctness of their code, enabling early detection of bugs and facilitating continuous integration and deployment practices. Ultimately, unit testing helps build robust and maintainable software systems by ensuring the integrity and correctness of individual code units.



3.7.5.1 Unit Testing

#### INTEGRATION TESTING

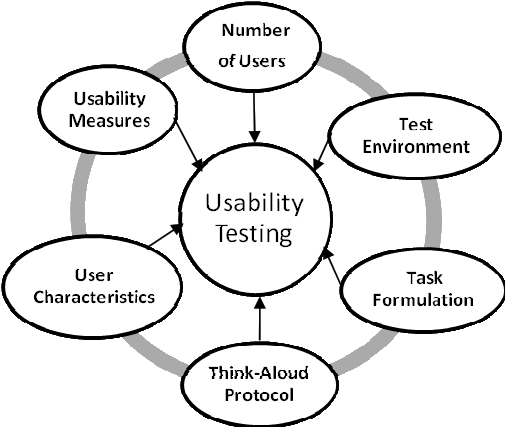
It plays a pivotal role in ensuring the seamless collaboration of individual components within a system. It focuses on verifying the interactions and interfaces between software modules and hardware components, assessing their compatibility and functionality when integrated. This phase of testing validates not only the internal connections within the software, which is akin to white box testing, but also the external interfaces, resembling black box testing. Integration testing aims to uncover any inconsistencies, communication failures, or compatibility issues that may arise when integrating various system elements. By thoroughly evaluating the interoperability of components, integration testing helps identify and resolve integration-related defects early in the development lifecycle, thereby enhancing the overall reliability, performance, and stability of the system. Ultimately, successful integration testing ensures that the system functions harmoniously as a cohesive whole, meeting the intended requirements and delivering value to end users.



3.7.5.2 Integration Testing

#### USABILITY TESTING

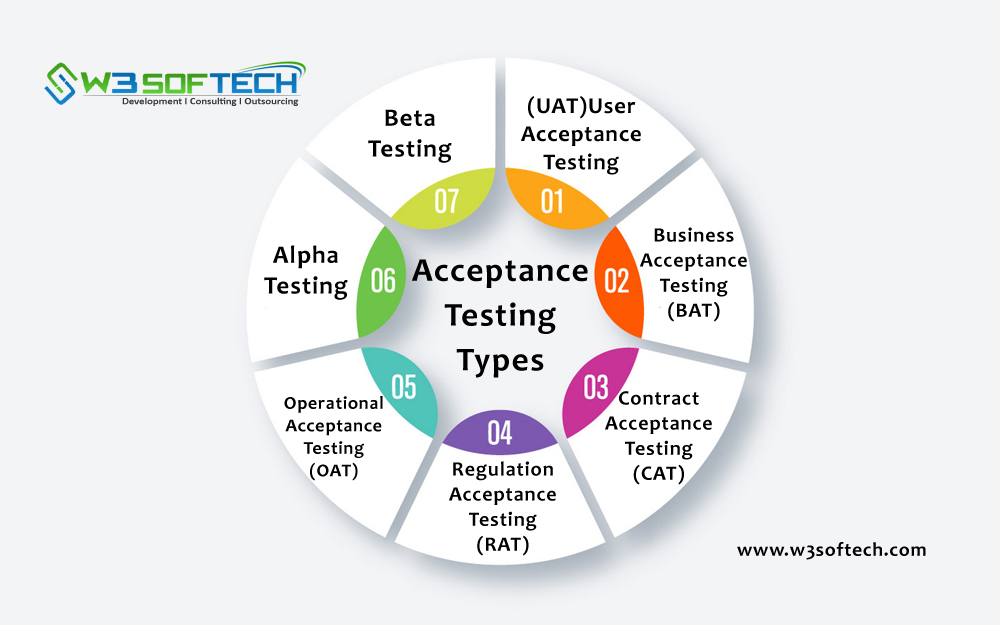
It is a crucial aspect of assessing the user-friendliness of a software's graphical interface, conducted from the client's perspective. It focuses on evaluating how easily users can learn to operate the software, how efficiently they can perform tasks, and their overall satisfaction with its design. Unlike white box testing that delves into internal mechanisms, usability testing operates as a black box testing method, examining the software's functionality without knowledge of its internal structure. By observing users interact with the interface and soliciting feedback, usability testing identifies potential usability issues, navigation challenges, and areas for improvement. This process helps developers refine the software's design, optimize user workflows, and enhance overall user experience, ultimately ensuring that the software meets the needs and expectations of its intended users.



3.7.5.3 Usability Testing

#### ACCEPTANCE TESTING

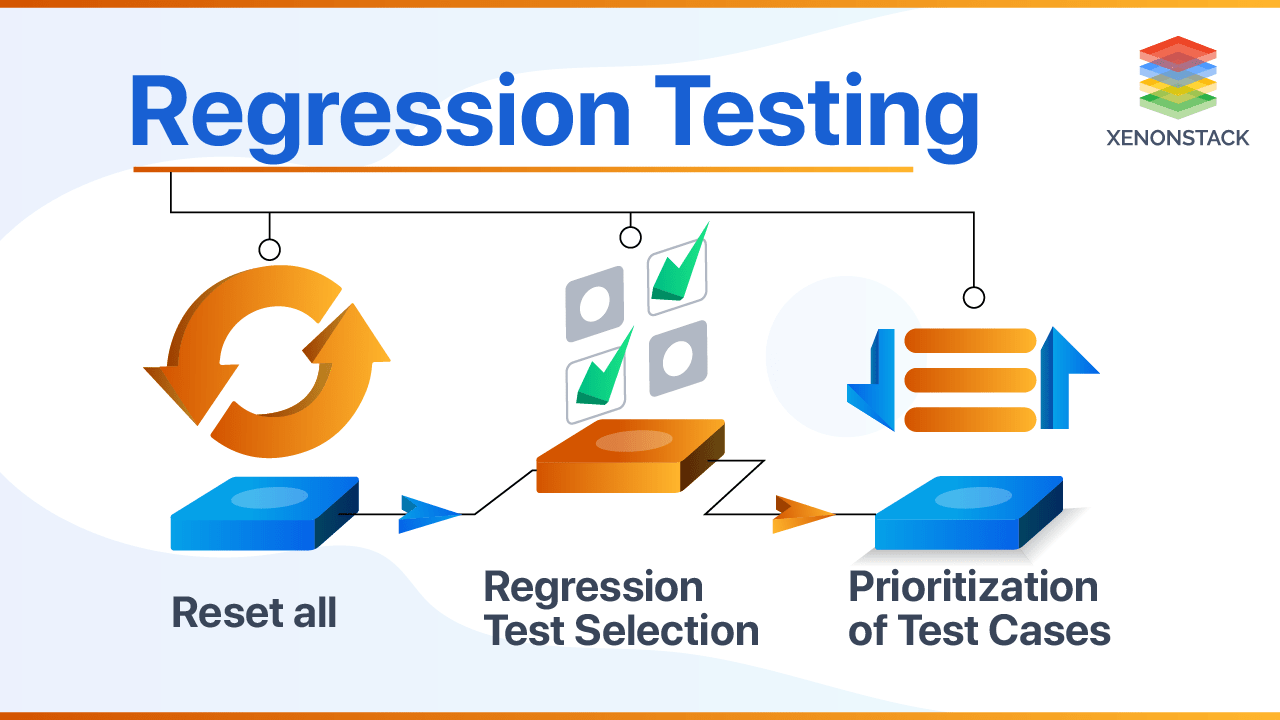
It serves as the ultimate assessment for software, akin to a final exam. It represents the customer's validation phase, where they confirm that the software aligns with their requirements and functions as anticipated. This method evaluates the system's functionality and usability from an end-user perspective, ensuring it meets predefined criteria before deployment. Similar to how a final exam assesses a student's mastery of course material, acceptance testing gauges the software's readiness for release. It provides stakeholders with confidence that the software meets their expectations and is fit for its intended purpose. By validating the software against user-defined criteria, acceptance testing helps mitigate the risk of post-deployment issues and ensures a smooth transition to production, ultimately enhancing customer satisfaction and trust in the software product.



**3.7.5.4** Acceptance Testing

#### REGRESSION TESTING

It plays a vital role in software development by verifying that modifications or updates to an application or system have not inadvertently introduced defects or caused regressions in existing functionality. It involves retesting previously implemented features to ensure their stability and integrity after changes have been made. Particularly crucial in agile development environments characterized by rapid iterations and frequent code updates, regression testing helps maintain software quality by identifying and addressing regression bugs early in the development cycle. By confirming that previously working features still operate as expected, regression testing mitigates the risk of unintended consequences and ensures the continued reliability of the software. Ultimately, it contributes to delivering a robust and dependable product that meets user expectations and maintains high standards of performance.



3.7.5.5 Regression Testing

# CHAPTER 4: RESULT AND OUTPUT

## 4.1 DATA ANALYSIS RESULTS

**Daily basis Rainfall Prediction using Machine Learning**

The main goal of this study is to find the important weather features that cause rainfall and to predict daily rainfall using machine learning techniques. Here is a summary of our findings.

We looked at environmental variables to see which ones are related to rainfall. Because the dataset is large, we focused on variables with a correlation greater than 0.20. The key variables we used to predict daily rainfall are:

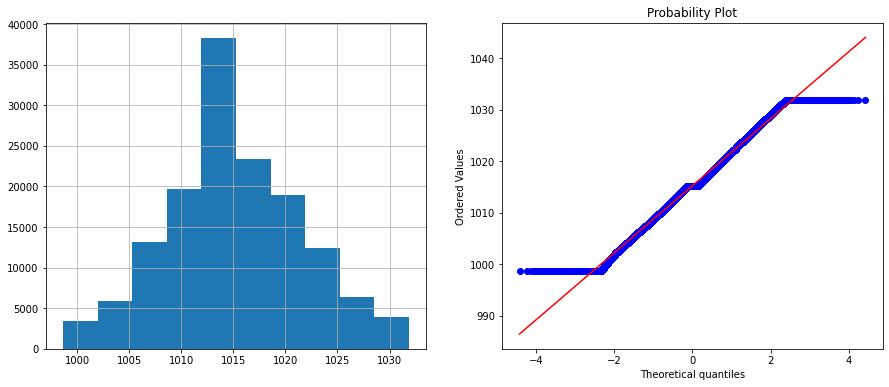
* 1. Evaporation
  2. Humidity
  3. Sunshine
  4. Wind Gust speed
  5. Maximum Daily Temperature
  6. Minimum Daily Temperature

We used several machine learning methods for prediction, including:

* 1. K-Nearest Neighbour
  2. Random Forest or decision tree
  3. Logistic Regression
  4. SVM

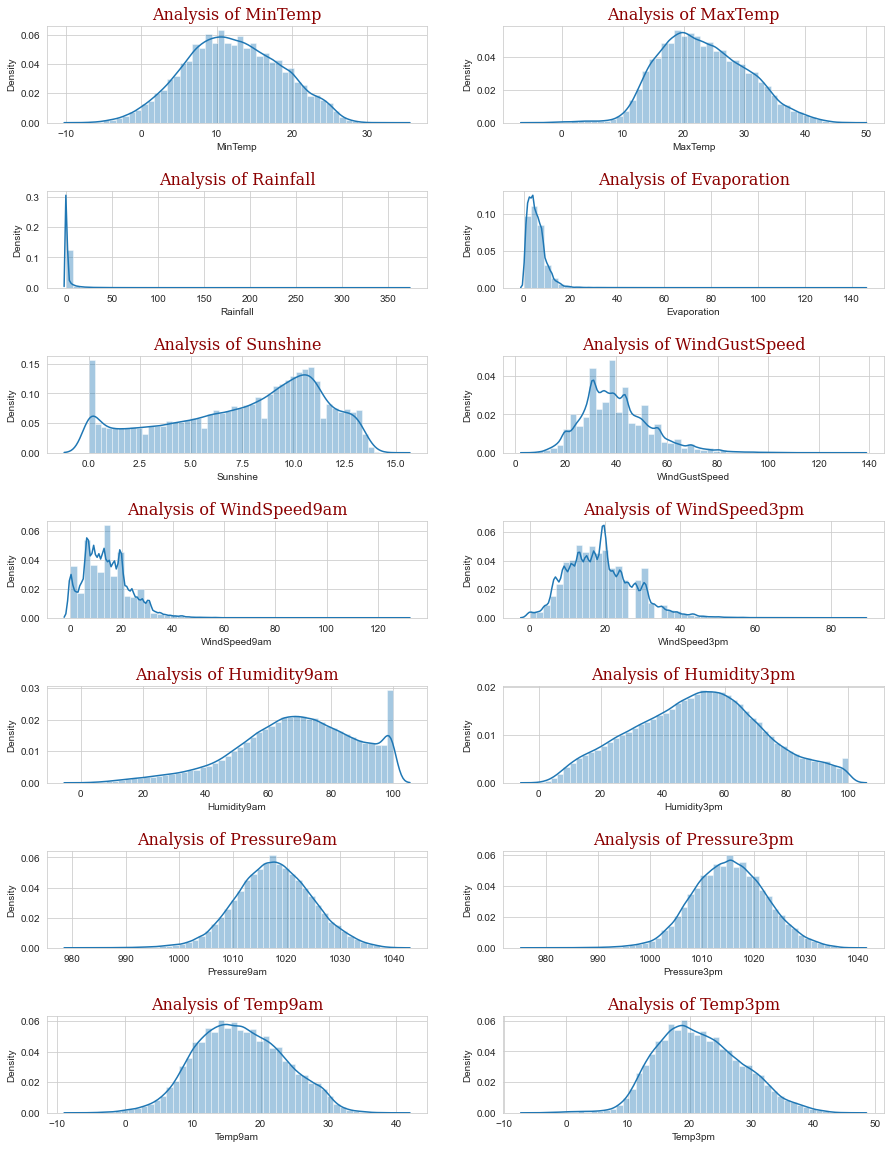
We divided the dataset into two groups: one for training the models and one for testing them. We used a Flask web application to run these algorithms. For the classification process, we carefully separated the data from the loaded dataset to ensure accurate predictions

This model predicts the rainfall using probabilities of occurring/ happening the event or not happening the event.



4.1.0.1 Data Analysis Of Feature

Here, we examined various features pattern to predict the rainfall event using the visualization tools below:



4.1.0.2 Feature EDA Diagram

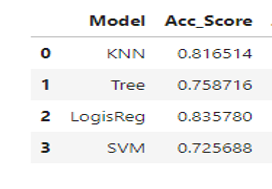
These various rainfall predicting classification algorithms have been monitored and checked using different analysis tools and techniques such as accuracy score, performance chart using plot the graph by data visualization.

Various performance measuring scores are being checked for choosing the perfect model for actual real time predicting the data of rainfall pattern such as Jaccard index score, F1 score, Accuracy score and Log loss score etc.

Here is the accuracy score for the mentioned algorithms that are used for the prediction of daily basis rainfall:

|  |  |  |
| --- | --- | --- |
| Serial No | Model | Acc\_Score |
| 1 | KNN | 0.8165 |
| 2 | Random Forest | 0.7587 |
| 3 | Logistic Regression | 0.8357 |
| 4 | SVM | 0.7256 |

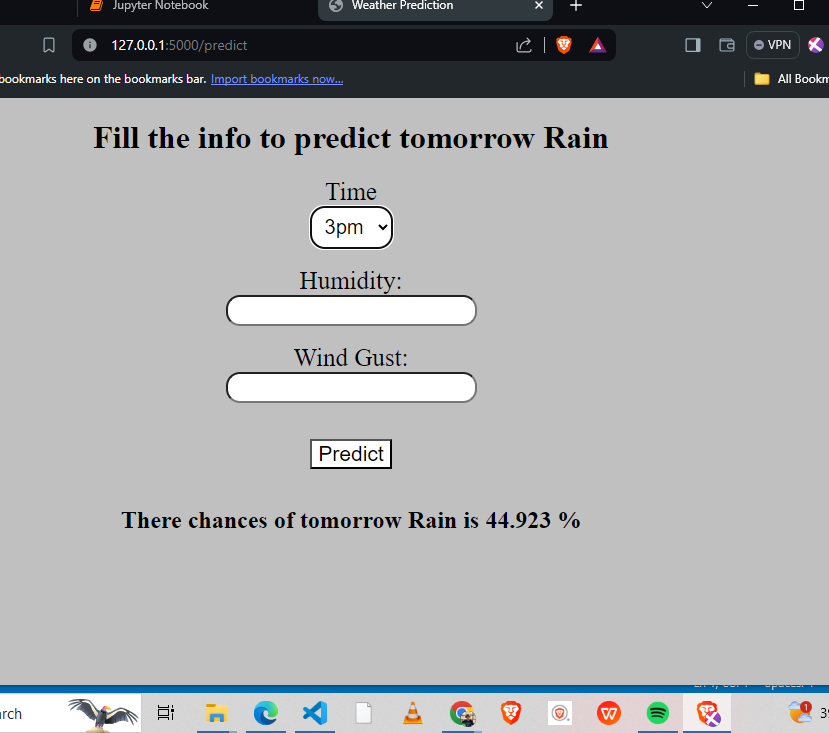
Table 5. Accuracy Score



* + - 1. Accuracy Diagram

By choosing the efficient predicting algorithm from the above mentioned we predict the happening of rainfall in percent on daily basis.

Here is the web app demo picture which shows how user interact with the GUI and predicts the daily basis rainfall in real time with prediction percent as output.



4.1.0.4 Output Daily Basis

**Monthly basis Rainfall Prediction using Machine Learning**

The goal of this project is to predict monthly rainfall using historical weather data. The dataset includes average monthly rainfall for each district in India from 1951 to 2000. We focused on key weather attributes to build an accurate prediction model.

**Handling Missing Values**

- Missing values were filled using the mean or mode of the existing data.

- For numerical attributes, the mean was used.

- For categorical attributes like events, the mode was used.

**Removing Unwanted Data**

- Irrelevant attributes were removed to focus on key features.

- Outliers and noisy data were identified and removed to ensure data quality.

**Normalization**

- All numerical features were normalized to bring them to a similar scale, which helps in improving the performance of machine learning algorithms.

**Seasonal Trends**

- Monthly trends were analyzed to observe seasonal patterns in rainfall.

- Histograms and line plots were used to visualize monthly rainfall distribution.

**Training and Testing Split**

- The dataset was split into training (80%) and testing (20%) sets.

- This ensures that the model can be evaluated on unseen data.

**Algorithms Used**

- \*\*Simple Linear Regression\*\*: For understanding linear relationships.

- \*\*Lasso Regression\*\*: For feature selection and regularization.

**Model Performance**

- Random Forest provided the highest accuracy with the lowest error rates.

- Linear and Lasso Regression also performed well, providing insights into linear relationships between variables.

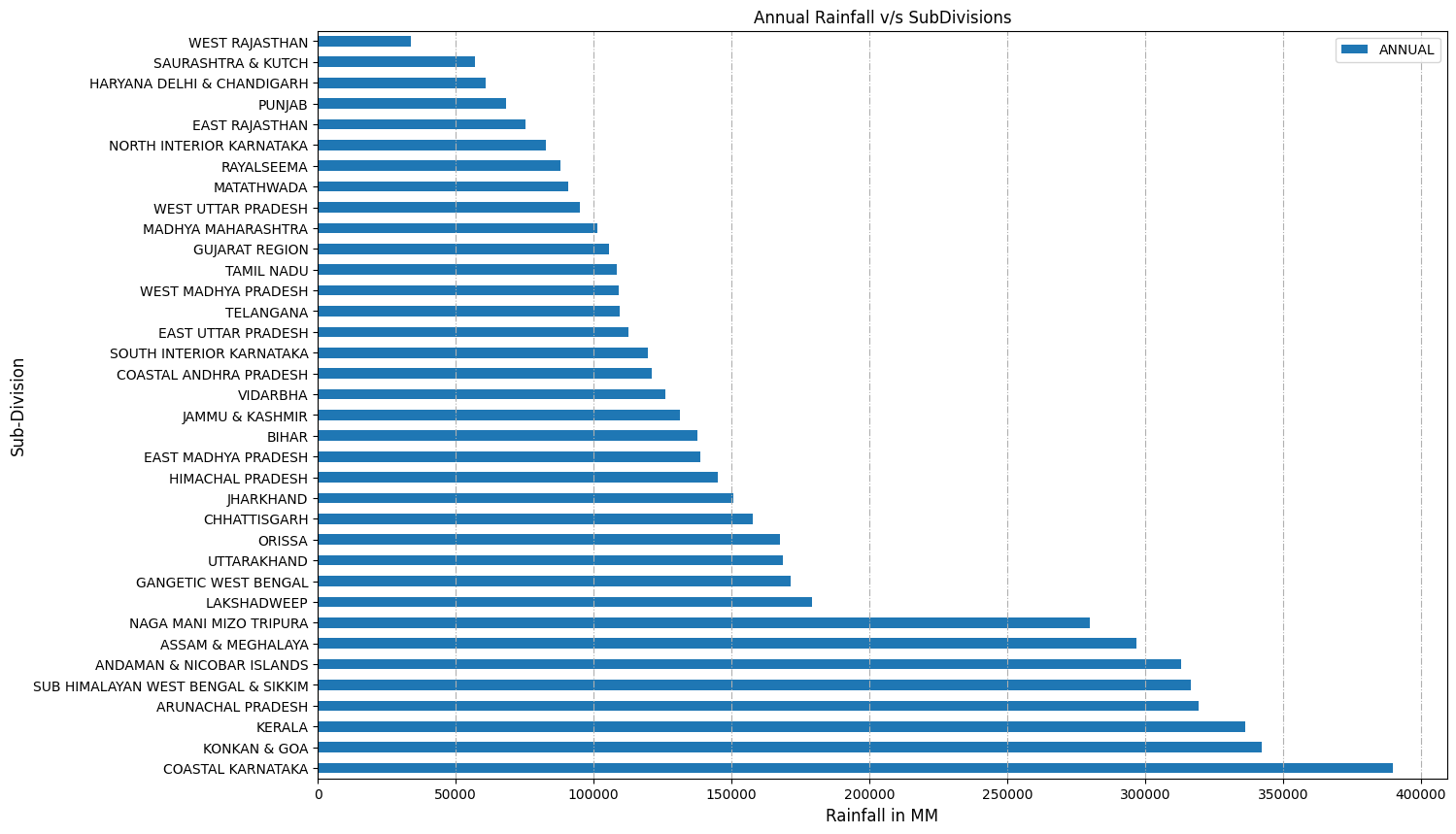
**Key Findings**

- Seasonal patterns were evident, with certain months showing consistently higher rainfall.

After all this we analysed the rainfall data for monthly basis by the help of data visualization tools here are some insights of the given data.

With the help of this visualization we are able to understand the patterns of monthly rainfall by which we understand the monthly pattern and make the machine to learn through regression algorithms to predict the amount of rainfall in particular are in a month and year.

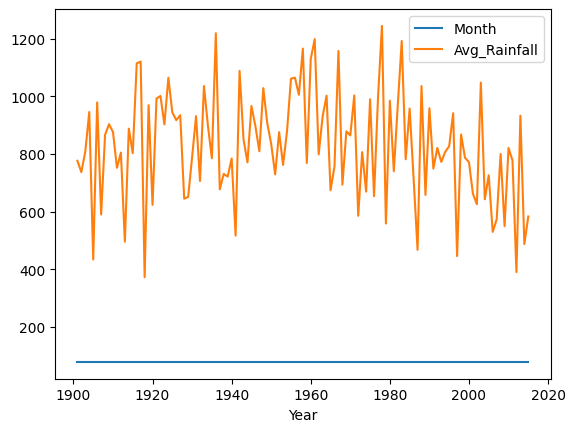
Here is the visualization of monthly rainfall prediction data:



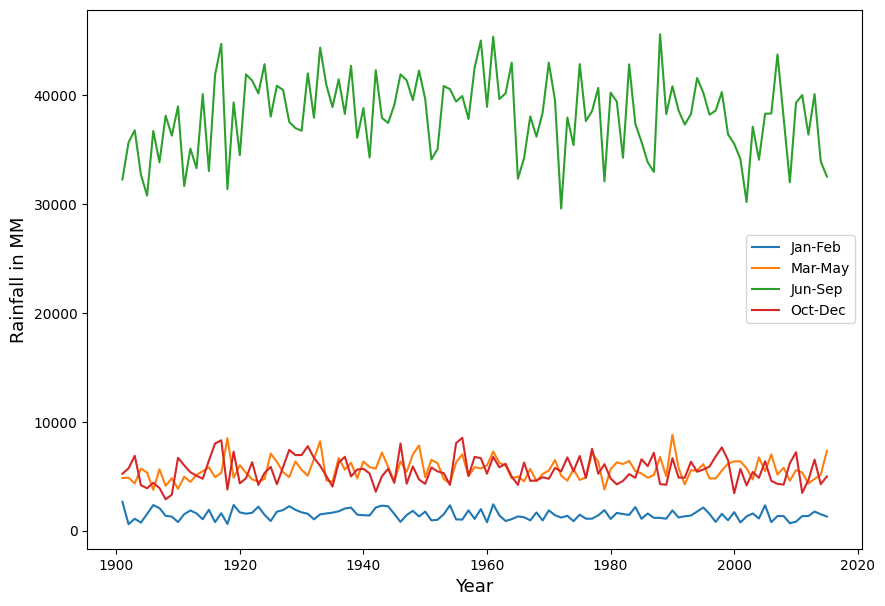
* + - 1. **A**nnual Rainfall vs SubDivisions

**Annual Rainfall in the Subdivisions**

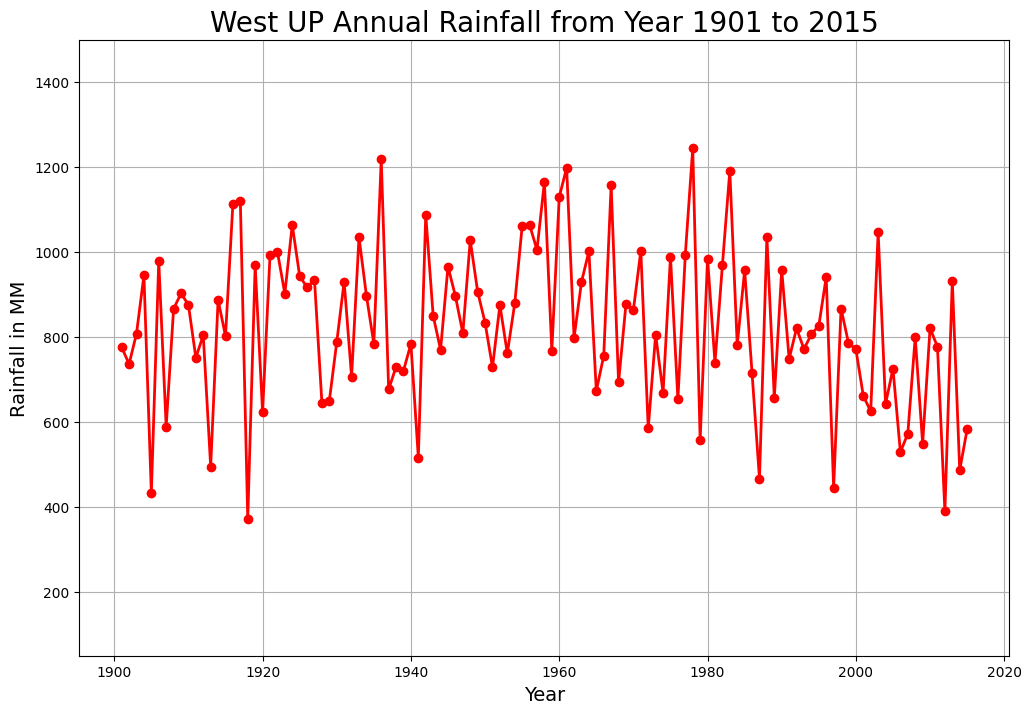
This above plot shows the information about the annual rainfall (mm) in the sub-divisions.



4.1.0.6 This plot shows the average rainfall in mm in the year.



4.1.0.7 This plot shows rainfall pattern through line graph yearly vs two months.

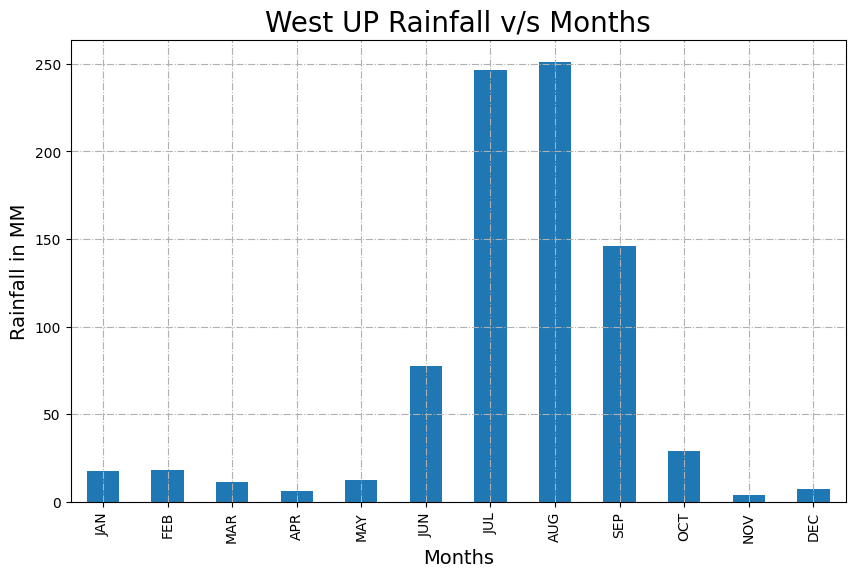


4.1.0.8 Rainfall Pattern Diagram

In this Project of monthly rainfall prediction we predict the rainfall amount in a particular region in a particular month and year.

Here we choose **west UP** as the region for which monthly rainfall predicted.

In this plot the data about the amount of rainfall in mm occur in the particular year in west UP.



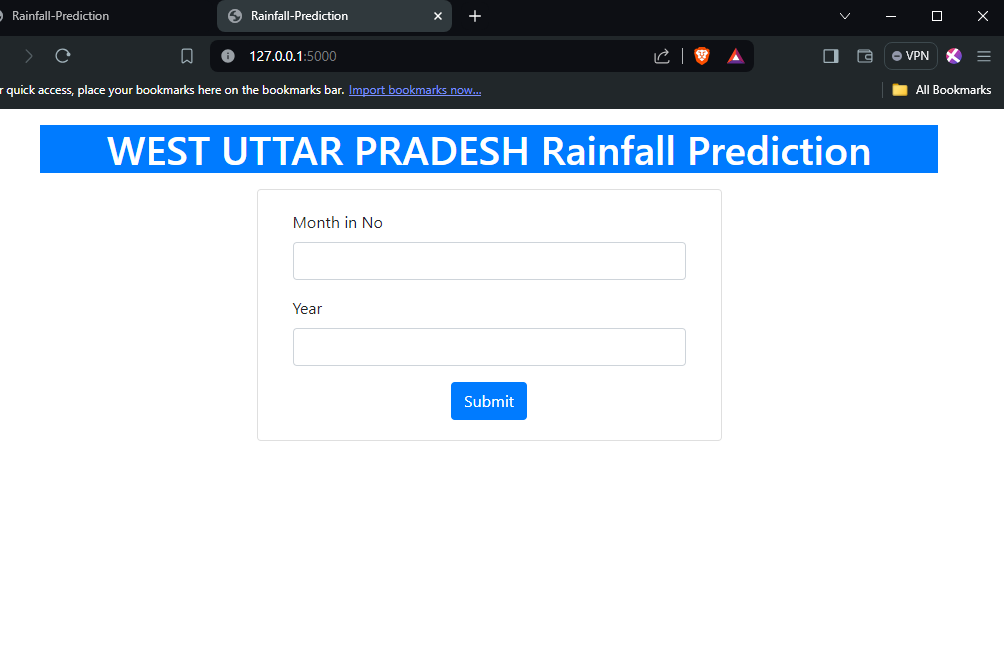
4.1.0.9 West UP Rainfall Monthly

Here we analysed the rainfall dataset on subdivision of West UP monthly basis to determine the most and least amount of rainfall occur in which month.

According to that we implement our model and predicts the rainfall amount in particular month and year in mm.

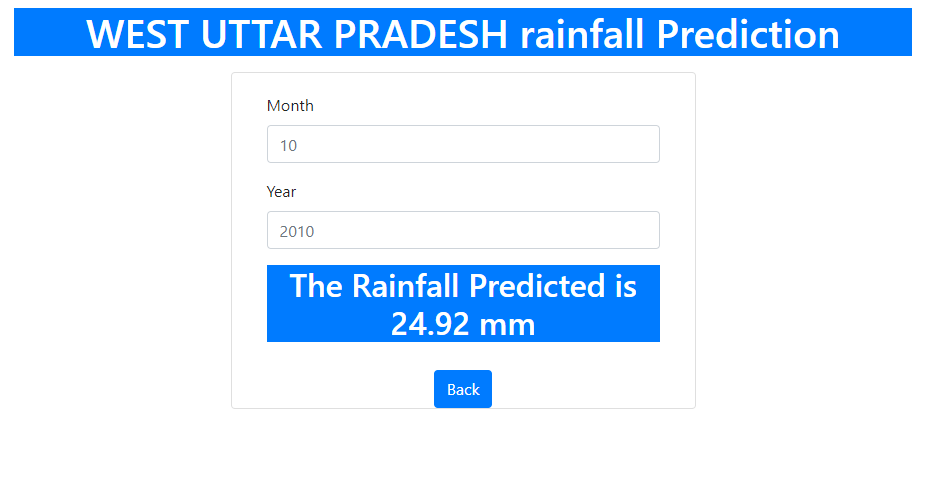
The output based on prediction is displayed through the web app using flask and python using HTML forms to take the input.

Here is the image to show GUI by which user interacts and get the predicted rainfall amount output.



4.1.0.10 Simple GUI (Graphical User Interface) by which user interacts and get the output generated by the model. Enter the month in numbers and year.

Here it is shown that how model gives the output using web application using flask and python.



4.1.0.11 Monthly Result GUI

# CHAPTER 5: CONCLUSIONS

This project discusses about the rain prediction for the daily, monthly basis. This application of web based application, the architecture for the recognition of rain prediction focused on state and previous rainfall patterns. The authors demonstrate the usage of rain prediction for rain water harvesting, for crop and natural damage like typhoon, tsunami, riverine flooding, cloud burst or floods. The authors report that data augmentation and analysis techniques have substantially improved the liner regression, SVM and random forest and logistic regression accuracy. Logistic regression is used here to acquire the derived coefficients using the relation and finds the most accurate model among them to predict the rainfall correctly.

## 5.1 OBJECTIVE SOLUTIONS

The overall objective of this study is to supplement existing manual or semi-manual solutions for rain fall prediction with smart technology.

**Accuracy Assessment**: Evaluate the accuracy of the predictions against actual rainfall data. This could involve statistical measures such as mean absolute error, root mean square error, or correlation coefficients.

**Seasonal Patterns Analyze**: If there are any discernible seasonal patterns or trends in the rainfall predictions. This could involve looking at how well the predictions align with typical rainy and dry seasons in your worse?

**Spatial Distribution**: Examine how well the predictions capture the spatial distribution of rainfall across your area of interest. Are there certain regions where the predictions consistently perform better or accurate?

**Temporal Dynamics Assess**: The temporal dynamics of the predictions. Do they accurately capture short-term rainfall events as well as longer-term trends? Are there certain times of day or year when the predictions tend to be more testing.

**Future Directions**: Lastly, consider future directions for your project are there unanswered questions or new research avenues that arise from your findings? What steps could be taken to further improve the accuracy and utility of the rainfall region.

**Impact of Input Variables**: Investigate which input variables or features have the most influence on the accuracy of the predictions. This could involve feature importance analysis or sensitivity others.

**Model Performance**: If you used machine learning or statistical models for the predictions, evaluate the overall performance of these models. Are there certain algorithms or techniques that consistently outperform scenarios?

**Uncertainty Analysis**: Consider incorporating uncertainty analysis into your conclusions. How confident are you in the predictions, and how does this confidence vary across different technologies.

**Improvement Opportunities**: Identify areas where the prediction methods could be improved. This could involve collecting additional data, refining the modeling approach, or incorporating new techniques or responders?

**Application and Use Cases**: They reflect on the potential applications and use cases for the rainfall predictions. How could this information be valuable to stakeholders such as farmers, water resource managers, or emergency predictions?

By considering these aspects, we can draw comprehensive conclusions from the project and provide valuable insights for decision-makers.



5.1.0.1 Rainfall Improvement

# CHAPTER 6: RECOMMENDATIONS:

Majority of Indian weather or rainfall has influenced their culture and season condition by the most common weather. More than 90% of the Indian weather has primary source of weather.

**Enhanced Data Collection**: Invest in more robust and diverse datasets for training your prediction models. This could include incorporating data from various sources such as satellite imagery, weather stations, ground sensors, and radar data. High-quality and comprehensive data are essential for accurate predictions.

**Feature Engineering**: Explore new features or variables that could improve the predictive power of your models. This might involve incorporating additional meteorological parameters such as humidity, wind speed, atmospheric pressure, and topographical data.

**Advanced Modeling Techniques**: Consider leveraging advanced machine learning techniques such as deep learning, ensemble methods, or hybrid models. These approaches can often capture complex patterns in the data more effectively than traditional statistical methods.

Spatial and Temporal Modeling Develop models that can effectively capture the spatial and temporal dynamics of rainfall patterns. This could involve spatial interpolation techniques for predicting rainfall across geographical areas and temporal modeling for forecasting short-term and long-term terms.

Uncertainty Quantification Integrate uncertainty quantification methods into your prediction framework to provide users with probabilistic forecasts rather than deterministic predictions. This can help stakeholders make more informed decisions, especially in uncertain or high-risk situations.

**Real-time Monitoring and Updating**: Implement real-time monitoring systems that continuously update prediction models based on the latest data. This ensures that predictions remain accurate and relevant, especially in rapidly changing weather conditions.

**Validation and Calibration**: Regularly validate and calibrate your prediction models using independent datasets and validation techniques. This helps identify any biases or inaccuracies in the models and ensures reliable performance under different conditions.

**User Feedback and Collaboration**: Engage with end-users and stakeholders to gather feedback on the usefulness and accuracy of the predictions. Collaborate with domain experts such as meteorologists, hydrologists, and agricultural specialists to refine and tailor the prediction models to specific user needs.

**Scalability and Accessibility**: Design prediction systems that are scalable and accessible to a wide range of users, including those in resource- constrained environments. This could involve developing lightweight models that can run on low-power devices or providing user-friendly interfaces for easy access to prediction results.

**Continued Research and Innovation Stay updated on the latest advancements in rainfall prediction research and technology. Continuously explore new methodologies, algorithms, and data sources to further improve the accuracy and reliability of your prediction.**

# CHAPTER 7: APPENDICES

## 7.1 DATA ANALYSIS EDA CODE:

Here are the python libraries or technologies that are used to make this rainfall prediction model based on machine learning are:

First, we import all the libraries we will be needed in the model, the libraries that we will be using in the rainfall predictiong model are:

1. Pandas

Pandas is a popular Python library used for data manipulation and analysis. It provides powerful data structures and functions to work with structured data, such as tables or spreadsheets, making it a valuable tool for tasks like data cleaning, transformation, and exploration.

1. Numpy

NumPy is a fundamental Python library used for numerical computing. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently.

1. Math

The math library in Python is a built-in module that provides access to mathematical functions and constants.

1. Sci Kit Learn

Scikit-learn, often abbreviated as sklearn, is a popular Python library for machine learning. It provides simple and efficient tools for data mining and data analysis.

1. Matplot Lib

Matplotlib is a widely used Python library for creating static, interactive, and animated visualizations in Python. It provides a comprehensive set of tools for generating plots, charts, histograms, scatterplots, and more.

1. Pybbn

PyBBN is a Python library that facilitates the construction, manipulation, and inference of Bayesian Belief Networks (BBNs) in Python. It offers a convenient way to model probabilistic relationships between variables and perform probabilistic inference based on observed evidence.

1. Pickle

In Python, "pickle" refers to a module used for serializing and deserializing Python objects. Essentially, it allows you to convert Python objects into a byte stream, which can then be saved to a file or sent over a network, and later reconstructed into Python objects.

1. Flask

Flask is a lightweight web framework for Python that makes it easy to build web applications. It provides tools and libraries for handling HTTP requests, routing URLs to Python functions, and generating HTML content to send back to the client.

1. HTML (Hyper Text Markup Language)
2. Python (a programming language)
3. Kaggle (a platform for understanding and downloading datasets)

## 7.2 WORKING ALGORITHM

1. Input historical weather data for each month:

* + - 1. Year
      2. Month
      3. Rainfall
      4. Other relevant features (temperature, humidity, wind speed, etc.)

2. Preprocess the data:

1. Handle missing values (e.g., imputation)
2. Normalize the data (if necessary)
3. Split the dataset into training and testing sets

3. Select a machine learning algorithm:

Choose an appropriate algorithm for regression tasks (e.g., Linear Regression, Random Forest, etc.)

4. Train the model:

1. Fit the selected algorithm to the training data
2. Adjust hyperparameters (if necessary) using cross-validation

5. Evaluate the model:

* + - 1. Predict rainfall values for the testing set
      2. Calculate evaluation metrics (e.g., Mean Squared Error, Root Mean Squared Error, R-squared, etc.)

6. Visualize the results:

1. Plot predicted rainfall values against actual rainfall values for each month
2. Analyze trends and patterns in the data

7. Make predictions:

1. Use the trained model to predict rainfall values for future months
2. Provide input data for the desired prediction period

8. Validate predictions:

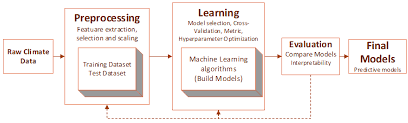
1. Compare predicted rainfall values with actual rainfall values (if available) for validation
2. Adjust the model as necessary based on validation results

9. Iterate and improve:

1. Refine the model based on feedback and validation results
2. Experiment with different algorithms, features, and hyperparameters to improve prediction accuracy

10. Finalize the model:

1. Once satisfied with the performance, finalize the model for use in predicting monthly rainfall values
2. Document the model and its parameters for future reference



7.2.0.1 Algorithm Flow Diagram

# REFERENCES

[1] Janani, B; Sebastian, P. (2014). Analysis on the weather forecasting and techniques. International Journal of Advanced Research in Computer Engineering & Technology, 3(1), 59–61. [http://ijarcet.org/wp-content/uploads/IJARCET-VOL-3- ISSUE-1-59-61.pdf](http://ijarcet.org/wp-content/uploads/IJARCET-VOL-3-%20ISSUE-1-59-61.pdf)

[2] Aftab, S., Ahmad, M., Hameed, N., Bashir, M. S., Ali, I., & Nawaz, Z. (2018). Rainfall prediction using data mining techniques: A systematic literature review. International Journal of Advanced Computer Science and Applications, 9(5), 143–150. <https://doi.org/10.14569/IJACSA.2018.090518>

[3] V.P Tharun, Ramya Prakash, S.Renuga Devi.(2019)"Prediction of Rainfall Using Data Mining Techniques" ieeexplore.ieee.org.

[4] Chowdari KK, Girisha R, Gouda KC. A study of rainfall over India using data mining. In 2015 International Conference on Emerging Research in Electronics, Computer Science and Technology (ICERECT). IEEE: New York. 2015; pp. 44–47.

[5]. Namitha K, Jayapriya A, SanthoshKumar G. Rainfall prediction using artifcial neural network on map-reduce framework. ACM. 2015. <https://doi.org/10.1145/2791405.2791468>.

[6]. Tharun VP, Prakash R, Devi SR. Prediction of Rainfall Using Data Mining Techniques. In 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT). IEEE Xplore. 2018; pp. 1507–1512.

[7] Zainudin S, Jasim DS, Bakar AA. Comparative analysis of data mining techniques for malaysian rainfall prediction. Int J Adv Sci Eng Inform Technol. 2016;6(6):1148–53.