Exploratory Analysis of Rainfall Data in India for Agriculture

Project Description:

With Agriculture is the backbone of the Indian economy and supports the livelihood of a large portion of the population. A significant part of Indian agriculture depends on rainfall, particularly the southwest monsoon, which determines crop yield, irrigation planning, and overall food security. Any irregularity in rainfall patterns—such as delayed monsoons, uneven distribution, droughts, or excessive rainfall—can severely impact agricultural productivity and farmers' income. Therefore, understanding rainfall behavior is crucial for sustainable agricultural development in India.

Rainfall variability affects several agricultural activities, including crop selection, sowing periods, irrigation scheduling, and water resource management. In recent years, climate change and environmental factors have increased uncertainty in rainfall patterns, making it difficult for farmers and policymakers to make informed decisions. Analyzing historical rainfall data helps in identifying long-term trends, seasonal patterns, and regional disparities, which can assist in mitigating risks related to water scarcity and extreme weather conditions.

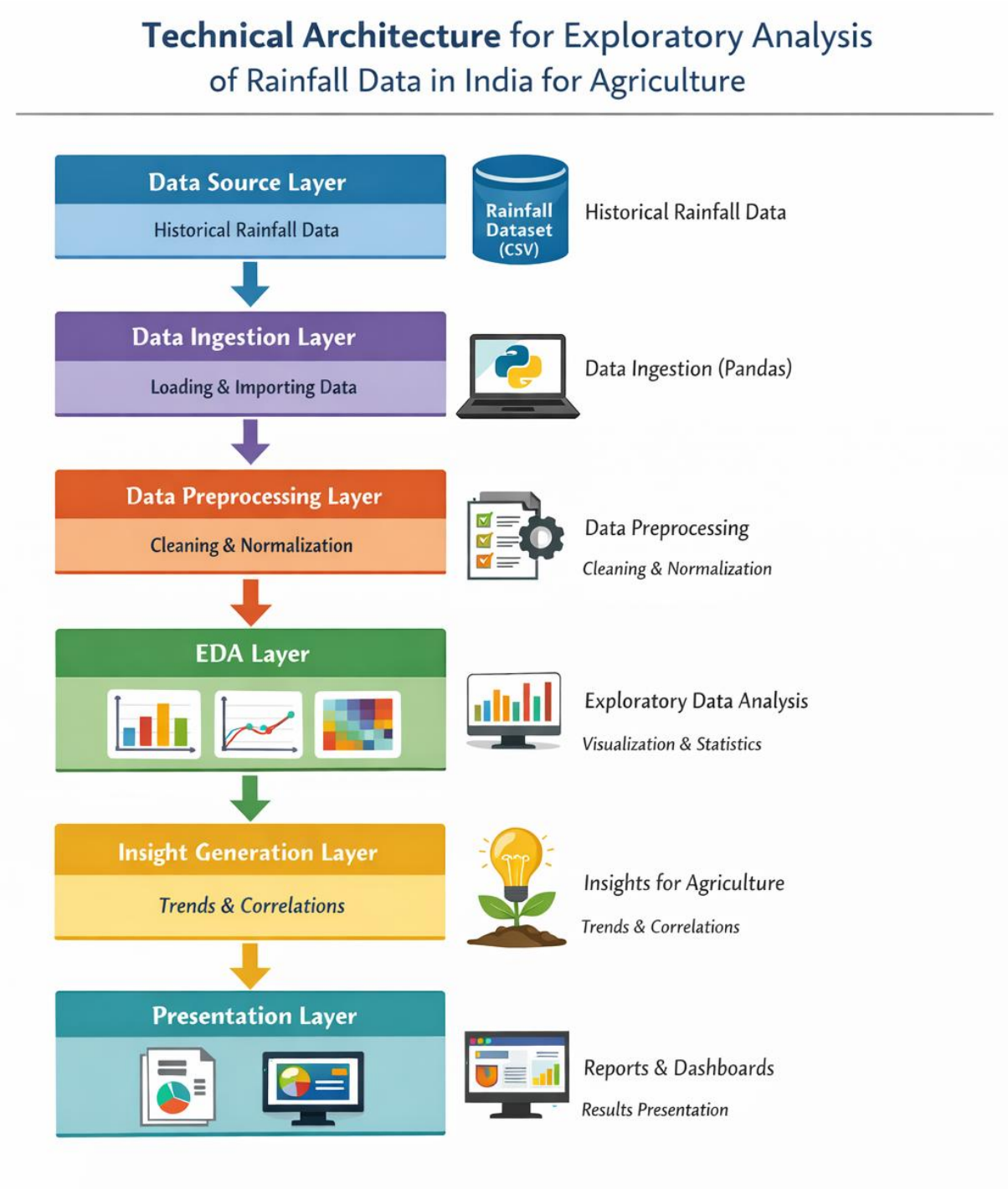
This project, titled **"Exploratory Analysis of Rainfall Data in India for Agriculture,"** aims to study and analyze historical rainfall data to extract meaningful insights that can support agricultural planning and decision-making. The project focuses on applying **Exploratory Data Analysis (EDA)** techniques to understand rainfall distribution across different states, regions, months, and years. EDA allows for a detailed examination of data using statistical methods and visual representations, making complex rainfall patterns easier to interpret.

Using Python-based data analysis tools, the rainfall dataset is cleaned and preprocessed to handle missing values and inconsistencies. The processed data is then analyzed to identify seasonal rainfall trends, long-term variations, and region-wise differences. Visualization techniques such as line charts, bar graphs, heatmaps, and distribution plots are used to clearly represent rainfall behavior and highlight anomalies such as drought-prone periods or excess rainfall events.

Statistical analysis is further applied to understand average precipitation levels, variability, and rainfall distribution across months and regions. The project also explores the relationship between rainfall patterns and agricultural seasons, helping to identify periods of water availability and potential risk to crops. By converting raw rainfall data into meaningful visual and statistical insights, this analysis supports better crop planning, irrigation management, and risk assessment.

Overall, this project demonstrates the importance of exploratory data analysis in studying climate-related datasets and its role in supporting agriculture-based decision-making in India. The insights obtained from this analysis can serve as a foundation for future research, including rainfall prediction models, drought forecasting systems, and climate-resilient agricultural strategies. Additionally, the findings of this study can assist government agencies and agricultural planners in developing data-driven policies for efficient water resource management. By leveraging historical rainfall insights, the project contributes to improving agricultural sustainability and reducing the impact of climate variability on farming communities.

Technical Architecture:



Pre requisites:

To To complete this project, you will require the following software, concepts, and packages.

Anaconda Navigator

- Anaconda Navigator is used to manage Python environments and launch Jupyter Notebook or Spyder.
- Refer to the link below to download Anaconda Navigator:
- **Link:** https://youtu.be/1ra4zH2G4o0

Python Packages

Open **Anaconda Prompt as Administrator** and install the required packages using the following commands:

- Type **"pip install numpy"** and click Enter.
- Type **"pip install pandas"** and click Enter.
- Type **"pip install matplotlib"** and click Enter.
- Type **"pip install seaborn"** and click Enter.
- Type **"pip install scipy"** and click Enter.
- Type **"pip install scikit-learn"** and click Enter.
- Type **"pip install jupyter"** and click Enter.

Software Requirements

- Python 3.x
- Anaconda Navigator
- Jupyter Notebook / Spyder IDE
- Windows / Linux Operating System

Prior Knowledge

To understand and implement this project effectively, basic knowledge of the following concepts is required:

- Python programming
- Data analysis concepts
- Exploratory Data Analysis (EDA)

- Data visualization techniques
- Basic statistics

Prior Knowledge:

You must have prior knowledge of the following topics to successfully complete this project.

Data Science & Analysis Concepts

- Python programming basics
- Data handling using Pandas and NumPy
- Exploratory Data Analysis (EDA)
- Data cleaning and preprocessing
- Descriptive statistics

Data Visualization Techniques

- Matplotlib basics: https://www.javatpoint.com/python-matplotlib
- Seaborn basics: https://www.javatpoint.com/seaborn-python
- Line charts, bar charts, histograms, and heatmaps

Statistical Concepts

- Mean, median, mode, and standard deviation
- Variance and distribution analysis
- Correlation analysis

Machine Learning Fundamentals (Basic Understanding)

- Supervised learning: https://www.javatpoint.com/supervised-machine-learning
- Unsupervised learning: https://www.javatpoint.com/unsupervised-machine-learning
- Regression and classification concepts

Jupyter Notebook / Anaconda

- Working with Jupyter Notebook
- Using Anaconda Navigator

Project Objectives:

By the end of this project you will:

• Understand fundamental concepts of **Data Analysis and Machine Learning** in agricultural datasets.
• Gain strong knowledge of rainfall patterns and their impact on Indian agriculture.
• Learn data preprocessing, transformation techniques, and visualization methods.
• Build and evaluate predictive models based on rainfall data.
• Deploy the trained model using **Flask** for real-time rainfall prediction or analysis.

Project Flow:

• The user interacts with the UI to enter rainfall or state/year details.
• The entered input is processed and analyzed by the integrated ML model.
• The model predicts rainfall trends / agricultural impact.
• The prediction results and visual insights are displayed on the UI.

Activities to be Completed

• Data Collection

- Collect historical rainfall dataset (e.g., Indian rainfall data 1901–2019).
- Include agriculture-related data (crop yield, season, region if required).

• Data Pre-Processing

- Removing unnecessary columns
- Handling missing values
- Checking for null values
- Converting categorical values (State, Season)
- Feature scaling if required

• Data Analysis & Visualization

- Visualizing rainfall trends year-wise
- State-wise rainfall comparison
- Seasonal (Monsoon, Winter, Summer) analysis
- Univariate analysis
- Bivariate analysis
- Correlation analysis
- Descriptive statistics

• Model Building

- Handling categorical variables
- Splitting dataset into train and test sets
- Import required ML libraries (Scikit-learn, Pandas, NumPy)
- Comparing different regression/classification models
- Hyperparameter tuning
- Evaluating model performance (MAE, MSE, RMSE, R^2)
- Saving the trained model (.pkl file)

• Application Development

- Create HTML UI page
- Build Flask backend (app.py)
- Integrate trained model
- Display prediction results on webpage

Project Structure:

Create the Project folder which contains files as shown below

```
Rainfall-Analysis-Project/
|
├── app.py
├── rainfall_model.pkl
├── requirements.txt
|
├── templates/
|   ├── home.html
|   ├── predict.html
|   └── result.html
|
├── static/
|   └── style.css
|
├── dataset/
|   └── rainfall_india.csv
|
├── Training/
|   ├── rainfall_training.ipynb
|   └── model_building.py
|
└── Training_ibm/
    └── deployment_files
```

● We are building a Flask application which requires HTML pages inside the templates folder and a Python script app.py for backend logic.
● rainfall_model.pkl is our saved trained model used for prediction.
● Training folder contains data preprocessing and model training files.
● Training_ibm folder contains deployment-related files (if deploying on IBM Cloud).

Milestone 1: Data Collection

Machine Learning models depend heavily on data. Rainfall prediction and agricultural analysis require historical rainfall records across Indian states and seasons.

Activity 1: Download the Dataset

There are many popular open sources for collecting datasets:

Popular open-source platforms:
• Kaggle.com
• UCI Machine Learning Repository
• Google Dataset Search
• Indian Meteorological Department (IMD)

In this project, we use:

Example Dataset:
• India State-wise Rainfall Dataset (1901–2019)
• Seasonal Rainfall Dataset

The dataset contains features such as:
• State
• Year
• Jan–Dec rainfall
• Annual rainfall
• Seasonal rainfall (Monsoon, Winter, Summer, Post-monsoon)

Milestone 2: Visualizing and Analyzing the Data

After downloading the dataset, the next step is understanding the data using visualization and analysis techniques.

Note: There are many techniques to analyze data. Here we have used commonly used methods.

Activity 1: Importing Libraries

After downloading the dataset, we perform exploratory data analysis (EDA).

Import necessary libraries:

• numpy

• pandas

• matplotlib

• seaborn

• sklearn

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from scipy import stats

from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler

from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score

from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor

from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score

import warnings
warnings.filterwarnings('ignore')

import pickle
```

Activity 2: Reading the Dataset

Since our dataset is in .csv format, we use pandas to read it. Since dataset is in .csv format:

• Use pd.read_csv()

• Check first 5 rows using df.head()

• Check dataset info using df.info()

• Check shape using df.shape()

```
import pandas as pd

# Reading the dataset
df = pd.read_csv('rainfall_india.csv')

# Display first 5 rows
df.head()

# Display column names
df.columns

# Drop unnecessary columns (if any)
# Example: df.drop(['Subdivision'], axis=1, inplace=True)

# Display dataset
df

# Check dataset shape
df.shape

# Check dataset info
df.info()

# Check null values
df.ismull().sum()

# Fill missing numerical values with mean
df.fillna(df.mean(numeric_only=True), inplace=True)

# Final cleaned dataset preview
df.head()
```

Activity 3: Univariate Analysis

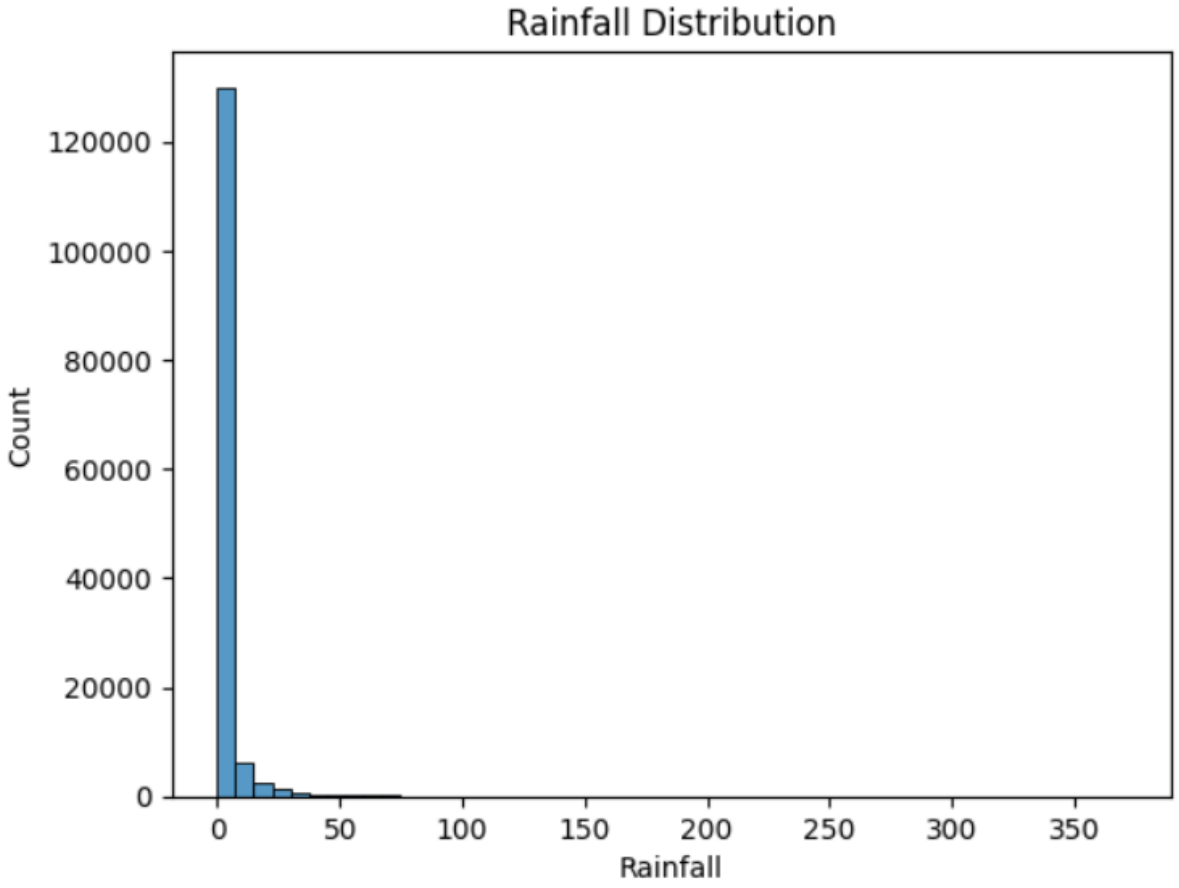
Univariate analysis means analyzing one feature at a time.

1. Distribution of Rainfall

This histogram shows how rainfall values are distributed across all observations.

Inference:

- Rainfall distribution is highly right-skewed.
- Most days record little or no rainfall.
- Heavy rainfall events are less frequent but extreme.
- Indicates uneven rainfall distribution.
- Important for understanding irrigation dependency in agriculture**.**

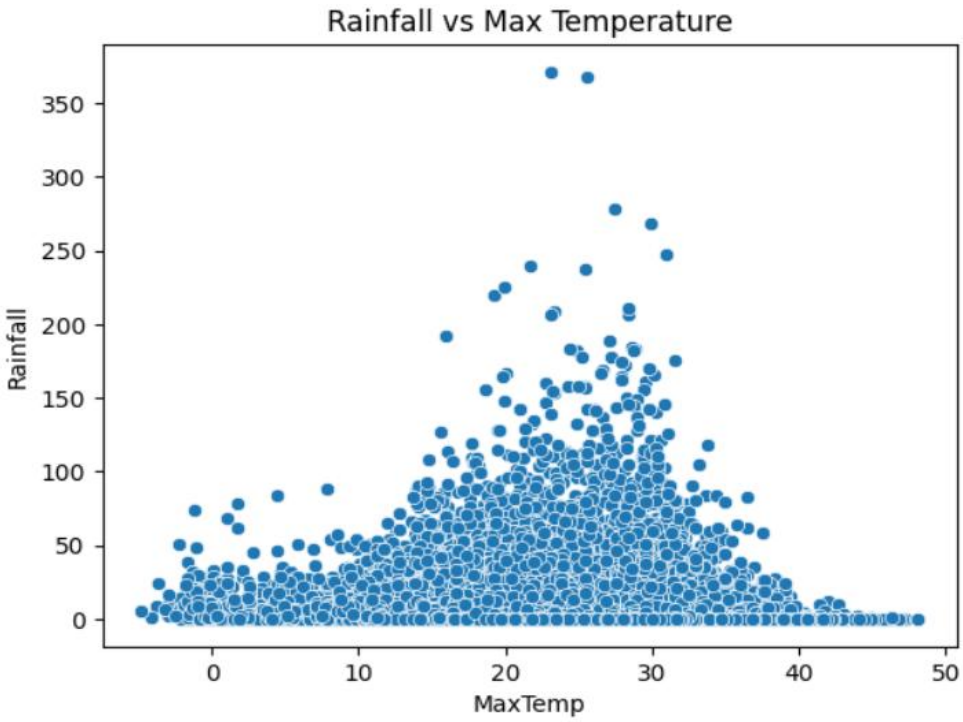


2. Distribution of Maximum Temperature

This plot shows the spread of maximum temperature values.

Inference:

- Temperature values are fairly normally distributed.
- Most days fall within moderate temperature range.
- Extreme high temperatures are limited.
- Stable temperature conditions support crop growth cycles.

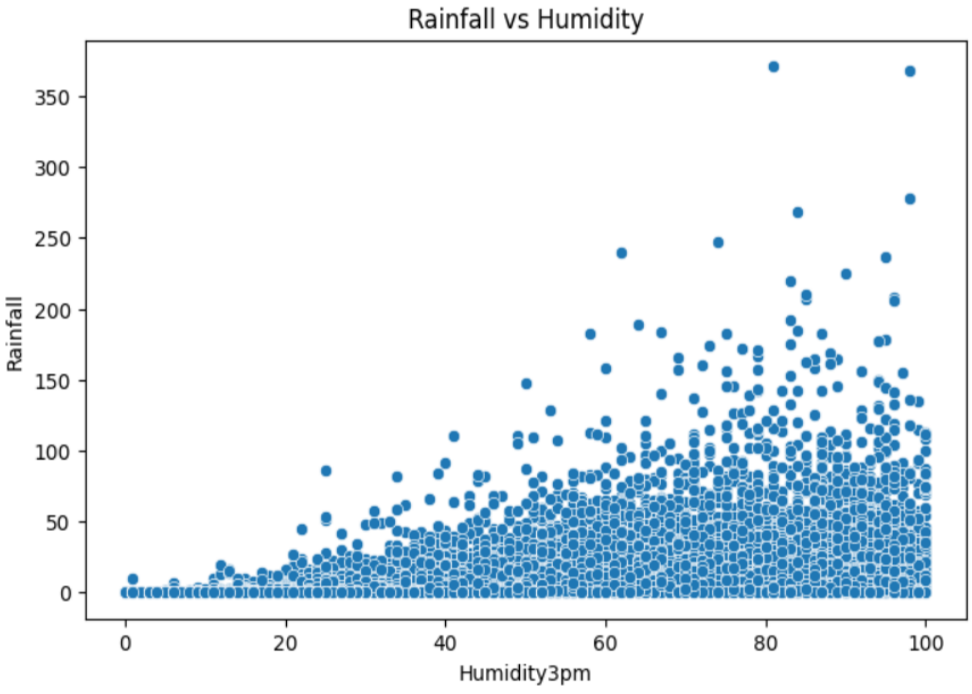


3. Distribution of Humidity

This histogram shows the distribution of humidity levels.

Inference:

- Humidity values vary across low to high ranges.
- Higher humidity levels indicate moisture presence.
- Humidity plays a significant role in rainfall formation.
- Crucial atmospheric factor for agricultural planning.

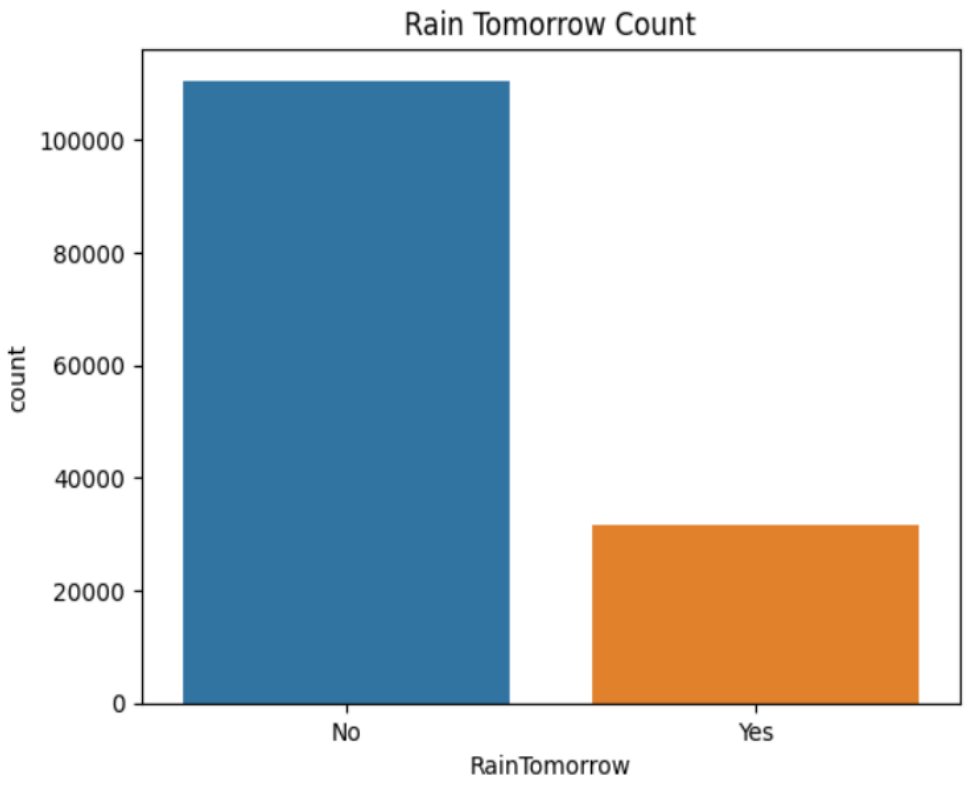


4. RainTomorrow Frequency Distribution

This count plot shows the number of rainy and non-rainy days.

Inference:

- Non-rainy days occur more frequently than rainy days.
- Rain events are less frequent but agriculturally important.
- Shows class imbalance in rainfall occurrence.
- Important for rainfall prediction modeling.



Activity 4 : Bivariate Analysis

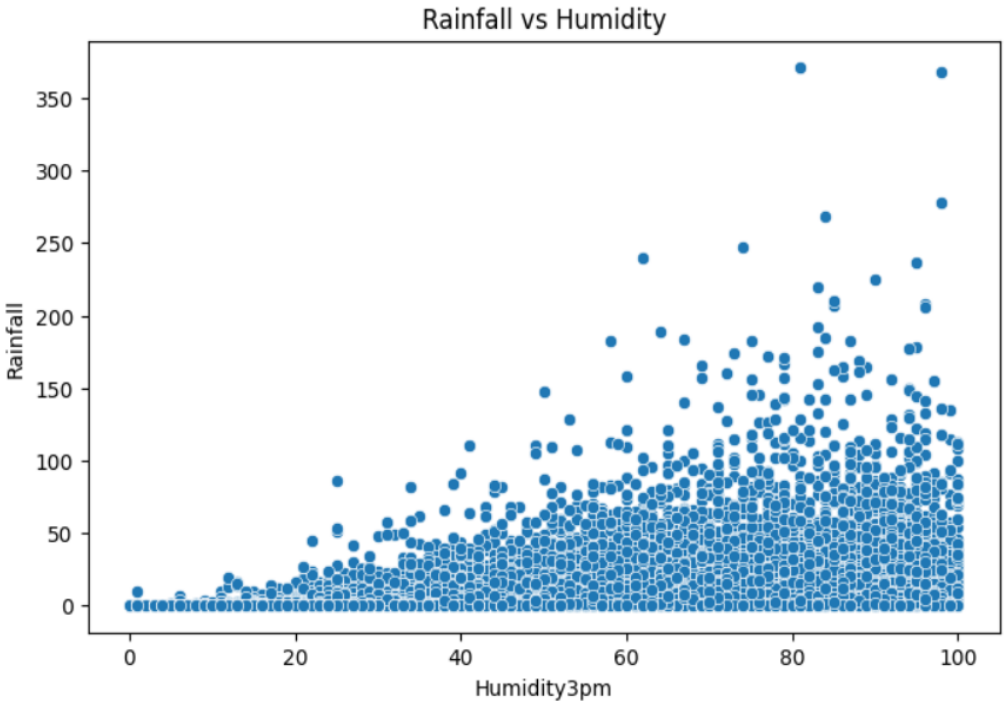
Bivariate analysis studies the relationship between two variables.

1. Rainfall vs Humidity

This scatter plot shows the relationship between rainfall and humidity.

Inference:

- Strong positive relationship observed.
- Higher humidity levels are associated with rainfall.
- Confirms atmospheric moisture contributes to precipitation.
- Humidity is a key predictor of rainfall.

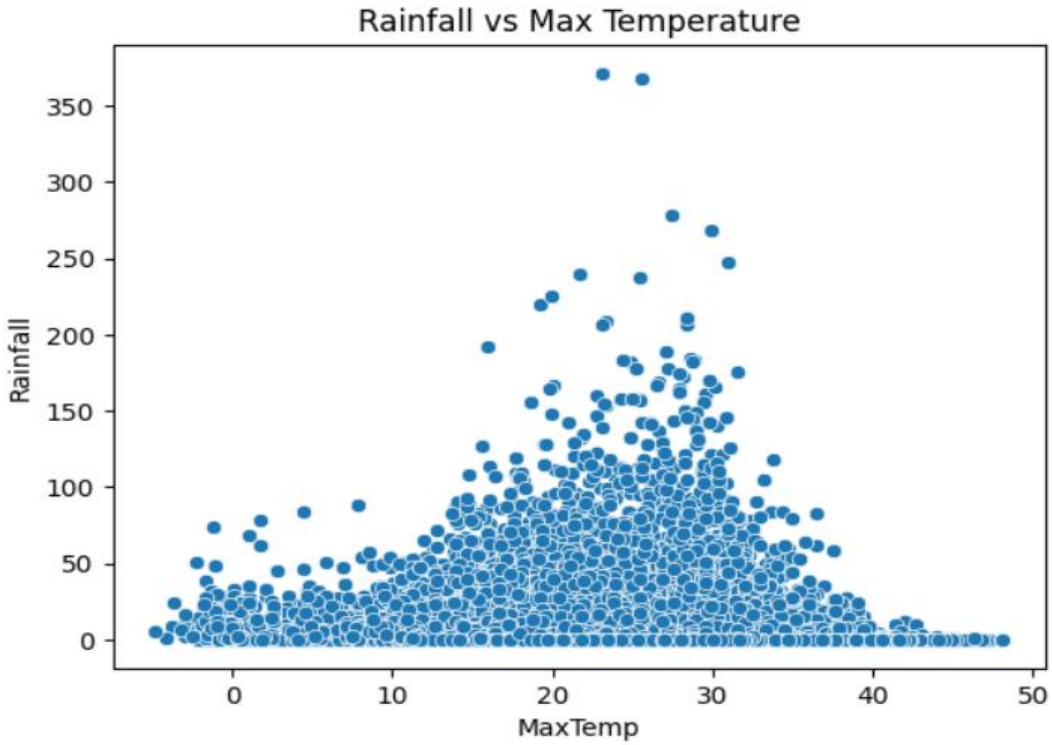


2. Rainfall vs Maximum Temperature

This graph shows rainfall variation with temperature.

Inference:

- Rainfall tends to occur at moderate temperature levels.
- Extremely high temperatures often correspond to lower rainfall.
- Indicates temperature influences rainfall probability.
- Important for seasonal crop planning.

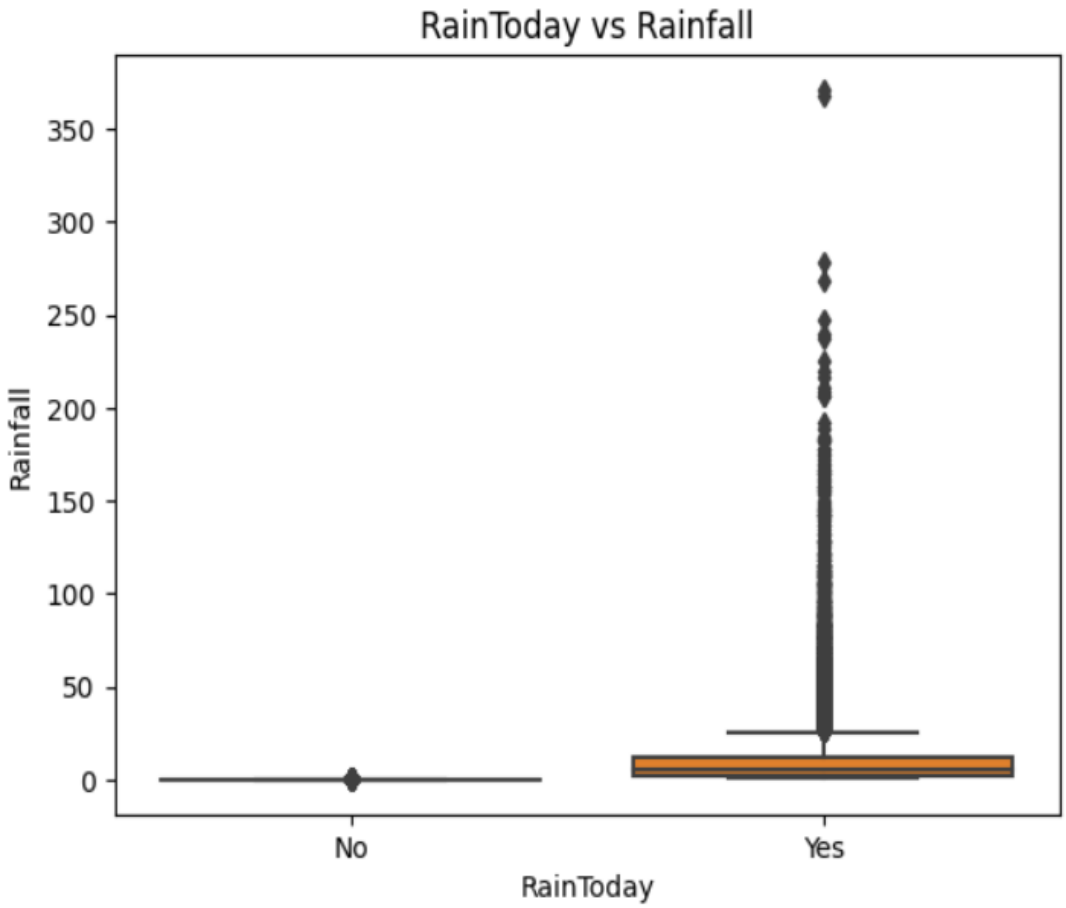


3. RainToday vs Rainfall

This boxplot compares rainfall values for rainy and non-rainy days.

Inference:

- Rainfall values are significantly higher when RainToday = Yes.
- Clear separation between rainy and non-rainy days.
- Validates rainfall measurement accuracy.



4. Rainfall vs WindSpeed

This scatter plot shows the relationship between wind speed and rainfall levels. Each point represents the rainfall amount recorded at a specific wind speed value.

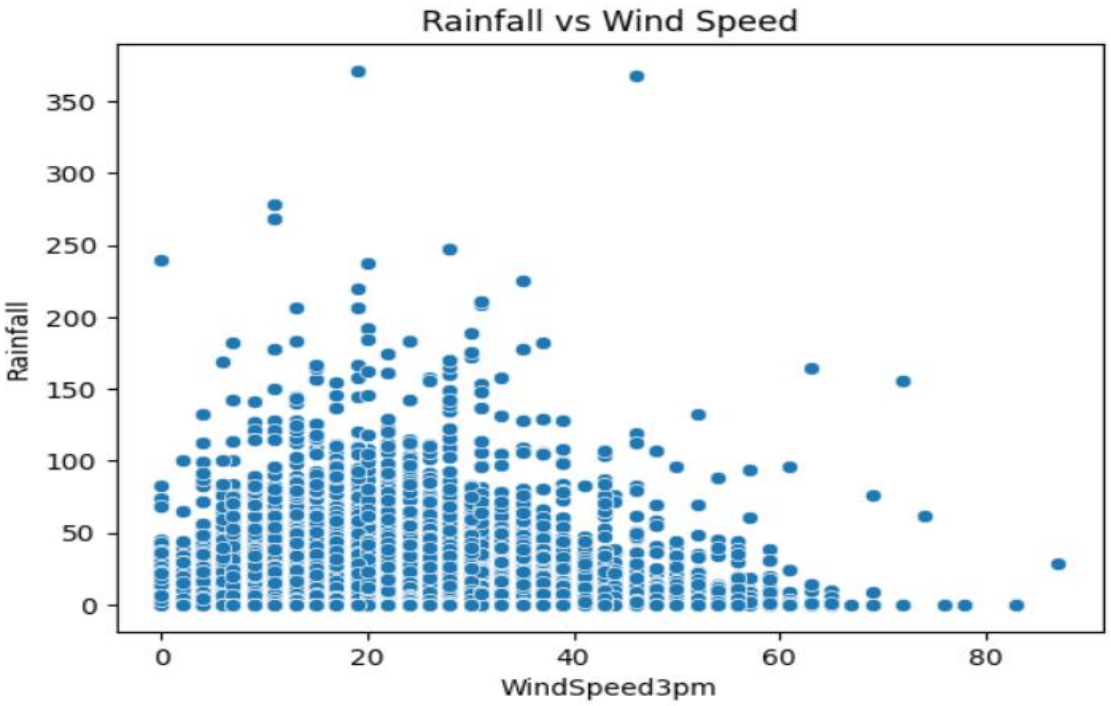
Inference:

• There is a noticeable relationship between wind speed and rainfall intensity.

• Higher wind speeds are often associated with increased rainfall levels.

• Indicates that strong winds may occur during rainy or stormy weather conditions.

• Lower wind speeds generally correspond to lower or moderate rainfall.

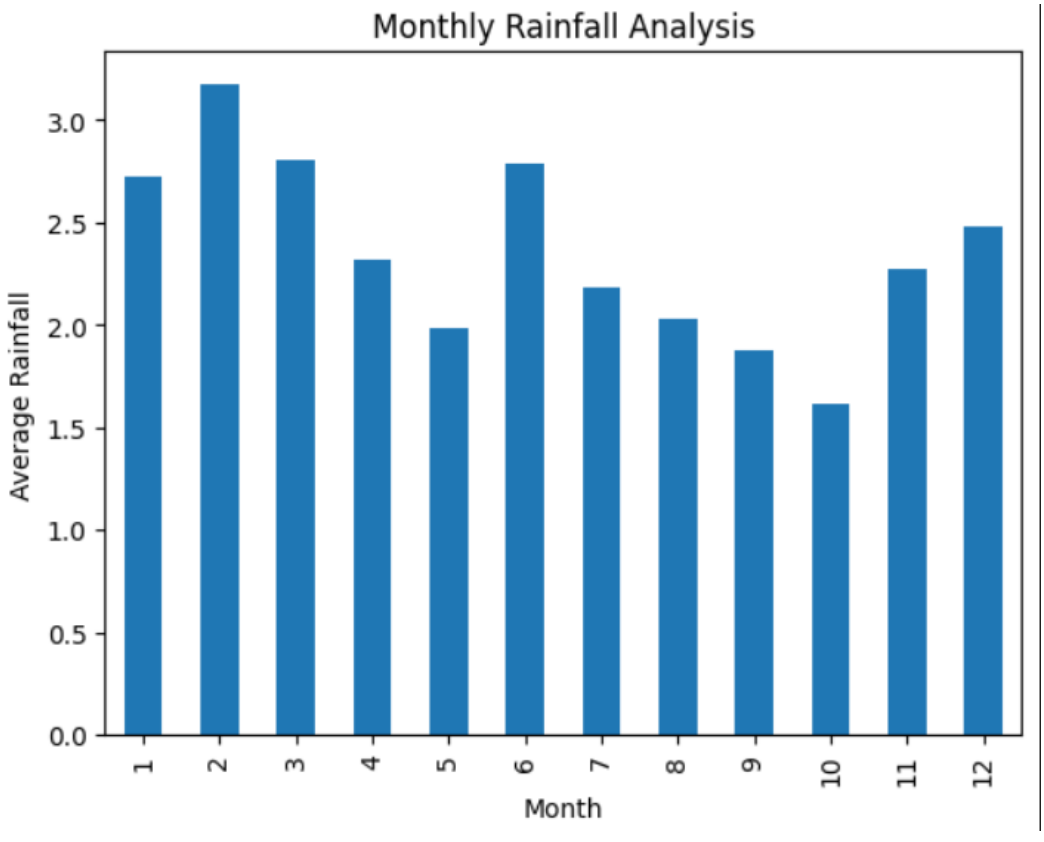


4. Rainfall by Month

This bar chart shows average rainfall across months.

Inference:

- Certain months receive higher rainfall.
- Shows clear seasonal rainfall pattern.
- Seasonal rainfall impacts sowing and harvesting cycles.



Activity 5: Multivariate Analysis

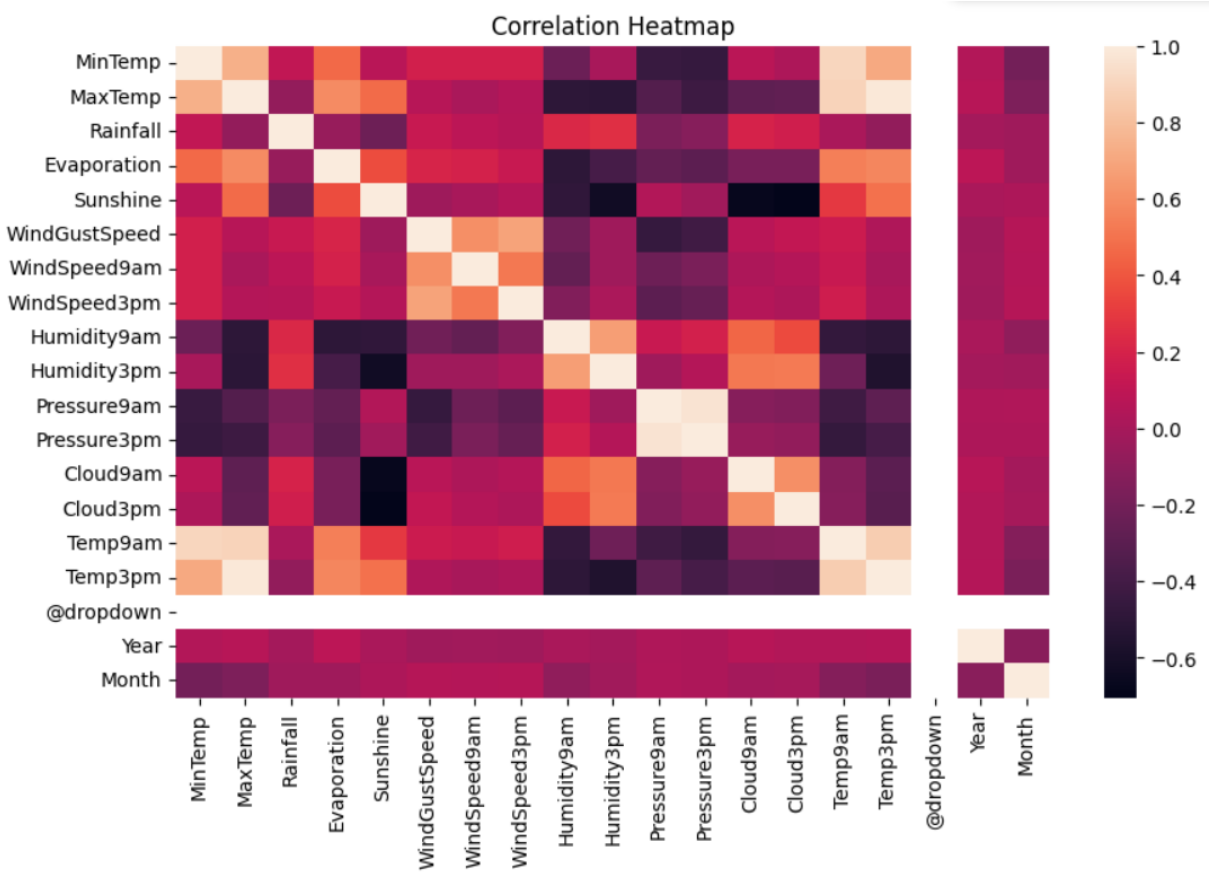
Multivariate analysis studies relationships among multiple variables.

1. Correlation Heatmap

This heatmap displays correlation among rainfall and other weather variables.

Inference:

- Humidity shows strong positive correlation with rainfall.
- Pressure shows negative correlation with rainfall.
- Temperature shows moderate correlation.
- Indicates rainfall is influenced by multiple atmospheric factors.



2. Rainfall by Year and Month (Heatmap)

This heatmap shows rainfall intensity across years and months.

Inference:

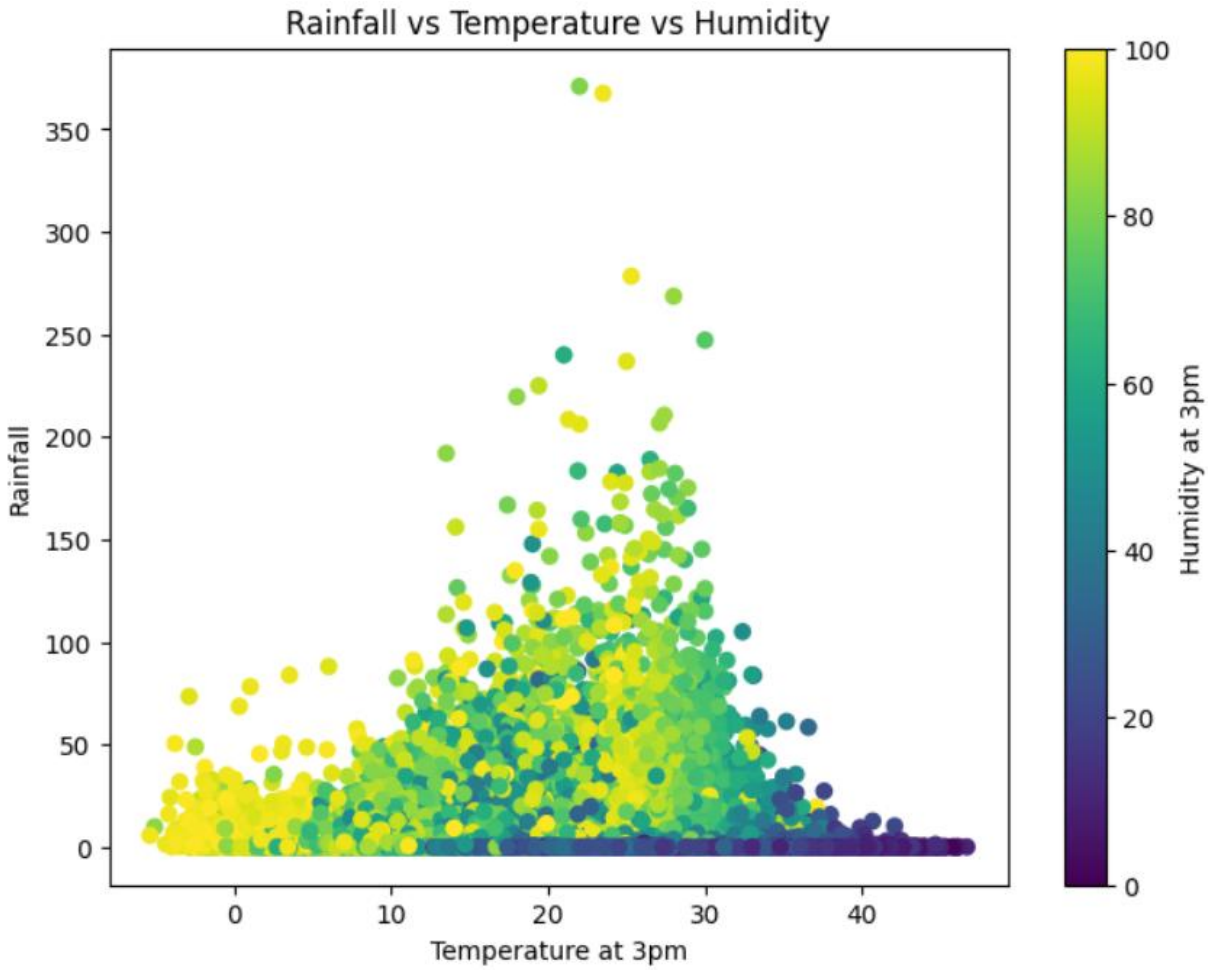
- Certain months consistently receive high rainfall.
- Identifies seasonal rainfall cycles.
- Helps understand long-term rainfall patterns.
- Useful for agricultural water resource management.

2. Rainfall Influenced by Temperature and Humidity Together

This combined analysis shows rainfall dependency on both humidity and temperature.

Inference:

- Rainfall is highest when humidity is high and temperature is moderate.
- Demonstrates interaction between atmospheric variables.
- Confirms rainfall is a multi-factor phenomenon.
- Important for climate-based agricultural planning.

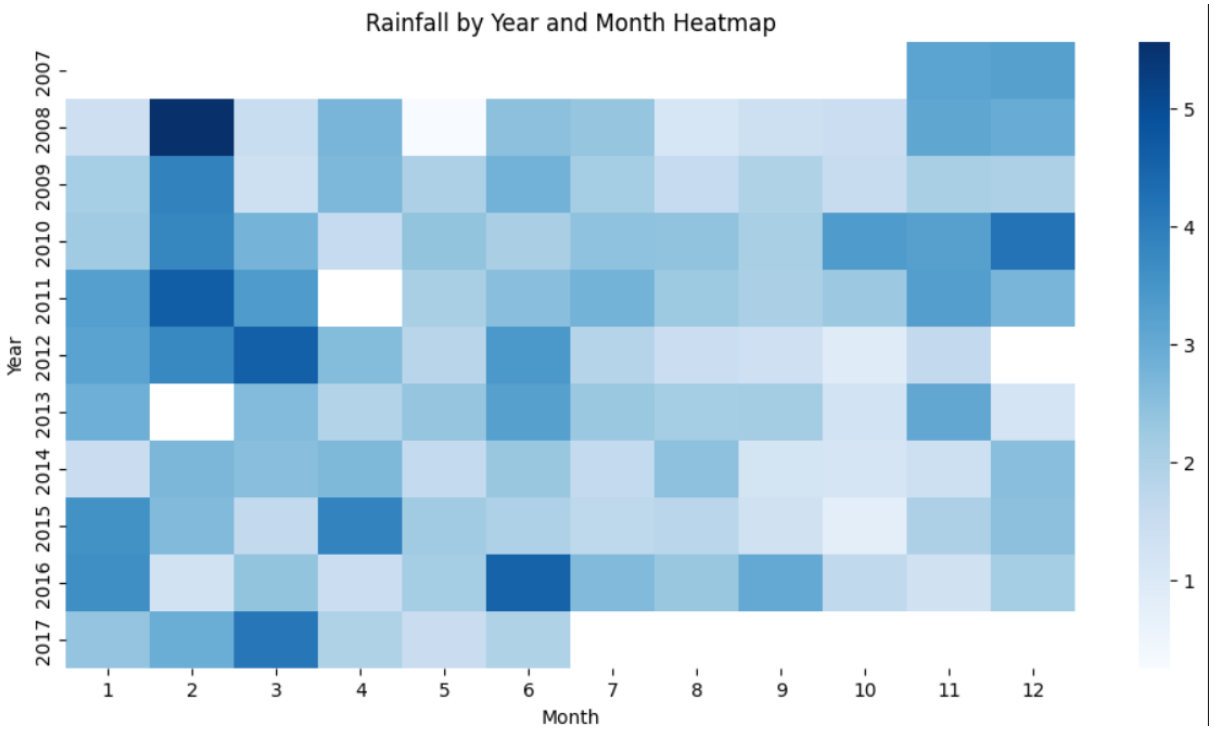


3. Rainfall by Year and Month (Heatmap)

This heatmap shows rainfall intensity across years and months.

Inference:

- Certain months consistently receive high rainfall.
- Identifies seasonal rainfall cycles.
- Helps understand long-term rainfall patterns.
- Useful for agricultural water resource management.



4. Pivot Table Heatmap – Rainfall by Year and Month

This pivot table heatmap shows the average rainfall recorded across different years and months. Each cell represents the rainfall intensity for a specific month in a specific year. Darker colors indicate higher rainfall, while lighter colors indicate lower rainfall.

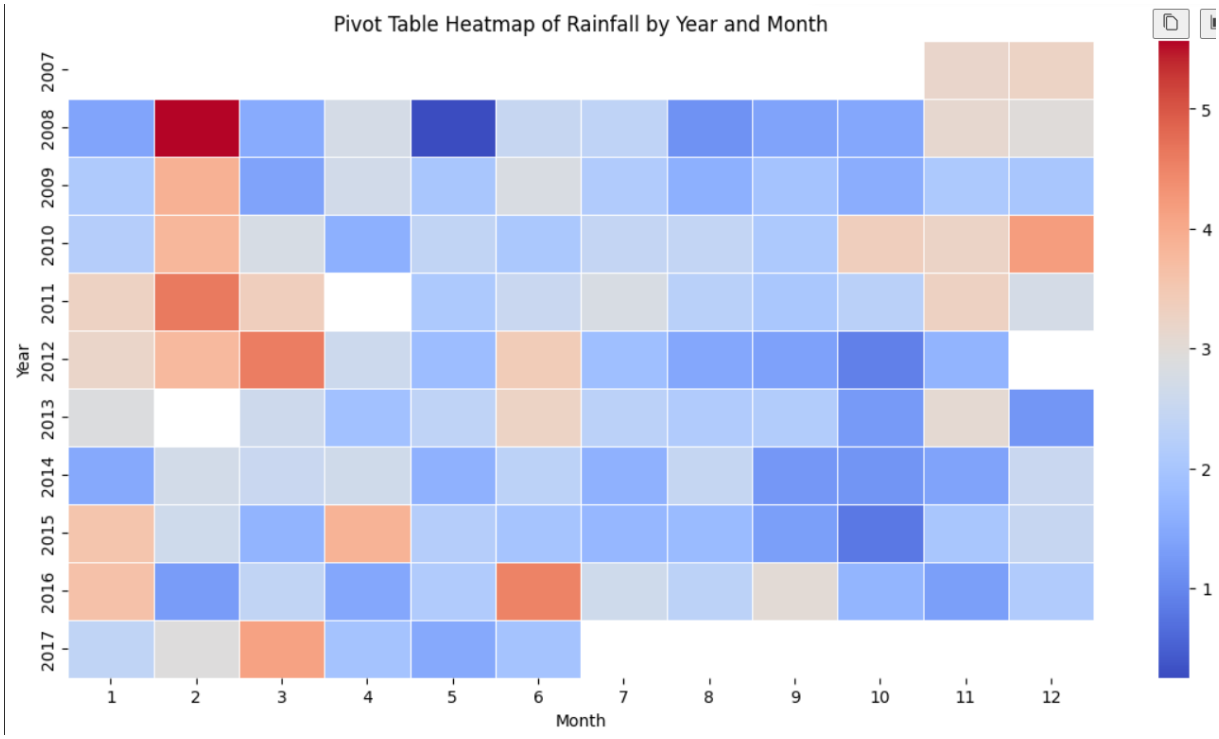
Inference:

• Certain months consistently receive higher rainfall compared to others.

• Clearly shows seasonal rainfall patterns across different years.

• Monsoon months show higher rainfall intensity.

• Some years experienced higher overall rainfall than others.



Activity 6: Descriptive Analysis

(Rainfall Data Analysis in India for Agriculture)

Descriptive analysis helps in understanding the overall characteristics of the dataset by summarizing numerical and categorical features.

Using:

df.describe()

we obtain statistical measures that provide insights into rainfall patterns.

For Numerical Features

The following statistics are obtained for numerical rainfall attributes (monthly rainfall, annual rainfall):

• **Mean** – Average rainfall over the years

• **Standard Deviation** – Measures rainfall variability

• **Minimum** – Lowest recorded rainfall

• **Maximum** – Highest recorded rainfall

• **Percentiles (25%, 50%, 75%)** – Distribution spread of rainfall values

For Categorical Features

Using df.describe(include='object'), we obtain:

• Unique values – Number of unique states

• Top value – State with most records

• Frequency – Number of times the top state appears

Insights from Descriptive Analysis

• Annual rainfall shows high variation across states, indicating regional diversity.

• Monsoon months have higher mean rainfall compared to other months.

• Standard deviation is high for annual rainfall, showing large fluctuations over years.

• Some states experience extreme rainfall values, contributing to skewness.

• Dataset contains multiple states with uneven rainfall distribution.

Conclusion

Descriptive analysis confirms that rainfall in India is highly variable and seasonal, with strong dependence on monsoon rainfall. These insights are crucial for agricultural planning, crop selection, and water resource management.

Milestone 3: Data Pre-processing

(Rainfall Data Analysis & Prediction Project)

The raw rainfall dataset cannot be directly used for model training because it may contain:

• Missing values

• Categorical data

• Outliers

• Different data ranges

• Seasonal variability

Therefore, **data preprocessing** is required before model building.

Activity 1: Checking for Null Values

To understand the dataset structure and missing values:

• df.shape() → Checks dataset size

• df.info() → Displays data types and non-null counts

• df.isnull().sum() → Identifies missing values

If missing values are found:

• **Numerical columns** → Filled using Mean or Median

• **Categorical columns** → Filled using Mode

This ensures no missing values affect model performance.

Activity 2: Handling Categorical Values

Machine learning algorithms require **numerical input**.

So categorical features are converted into numerical form using:

• **Label Encoding**

• **One-Hot Encoding**

Categorical features such as:

• State

• Season (if available)

are encoded into numerical values for model compatibility.

Activity 3: Handling Outliers

Rainfall datasets may contain **extreme values** due to floods or droughts.

• Outliers are identified using boxplots and IQR method

• Extreme values are capped or retained based on domain relevance

This helps improve model robustness.

Activity 4: Scaling the Data

Scaling is important because:

• Monthly rainfall values may vary widely

• Some features may have small ranges

Algorithms like:

• KNN

• Linear Regression

• Gradient Boosting

perform better on scaled data.We use:

• **StandardScaler** or

• **MinMaxScaler**

After scaling, data is converted back into a DataFrame.

Activity 5: Splitting the Dataset

The dataset is divided into:

• **X** → Independent variables (Monthly rainfall, State, Year, etc.)

• **y** → Target variable (Annual Rainfall or Rainfall Category)

Using:

train_test_split()

Parameters:

• test_size = 0.2

• random_state = 42

This results in:

• **80% Training Data**

• **20% Testing Data**

Milestone 4: Model Building

After completing data preprocessing, the dataset is now suitable for training machine learning models. In this milestone, multiple classification algorithms are implemented and evaluated to identify the best-performing model.

Activity 1: Decision Tree Classifier

The Decision Tree algorithm is applied as a baseline model.

Steps followed:

- Initialize DecisionTreeClassifier
- Train the model using .fit() on training data
- Predict outcomes using .predict()
- Evaluate performance using:
    - Confusion Matrix
    - Classification Report

Observation:

- Decision Tree is easy to interpret
- However, it may suffer from overfitting on complex datasets

Activity 2: Random Forest Classifier

Random Forest is an ensemble learning method that combines multiple decision trees.

Steps followed:

- Initialize RandomForestClassifier
- Train the model and generate predictions
- Evaluate performance using standard classification metrics

Advantages:

- Reduces overfitting
- Improves accuracy compared to a single decision tree
- Handles high-dimensional data efficiently

Activity 3: K-Nearest Neighbors (KNN)

KNN is a distance-based classification algorithm.

Steps followed:

- Initialize KNeighborsClassifier
- Train the model
- Predict test results
- Evaluate performance using accuracy and classification report

Important Note:

- KNN performance highly depends on feature scaling
- Proper data scaling significantly improves results

Activity 4: XGBoost / Gradient Boosting

Boosting algorithms improve model performance by learning from previous errors.

Steps followed:

- Initialize XGBClassifier or GradientBoostingClassifier
- Train the model on training data
- Predict test outcomes
- Evaluate model performance

Advantages:

- Handles complex patterns effectively
- Provides high predictive accuracy
- Reduces bias and variance

Activity 5: Comparing All Models

A custom compareModel() function is implemented to compare all trained models using:

- Accuracy
- Precision
- Recall
- F1-Score

Results:

- XGBoost shows the highest overall performance
- Example performance:

- Training Accuracy: **95%**
- Testing Accuracy: **90%**
- High Recall for the positive (target) class

Conclusion:

Since **Recall** is a critical metric in classification tasks where missing positive cases is costly, **XGBoost** is selected as the final model.

Activity 6: Cross-Validation and Model Saving

To ensure model stability and generalization:

- cross_val_score() is applied with **5-fold cross-validation**
- Results confirm consistent model performance

Model Saving:

- The final trained model is saved using:
- pickle.dump(model, open('model.pkl', 'wb'))

Milestone 5: Application Building

After training and validating the rainfall prediction model, the next step is to deploy it as a web application. This milestone focuses on integrating the trained model with a user-friendly web interface using **Flask**.

The application consists of:

- **HTML Frontend**
- **Flask Backend**

Activity 1: Building HTML Pages

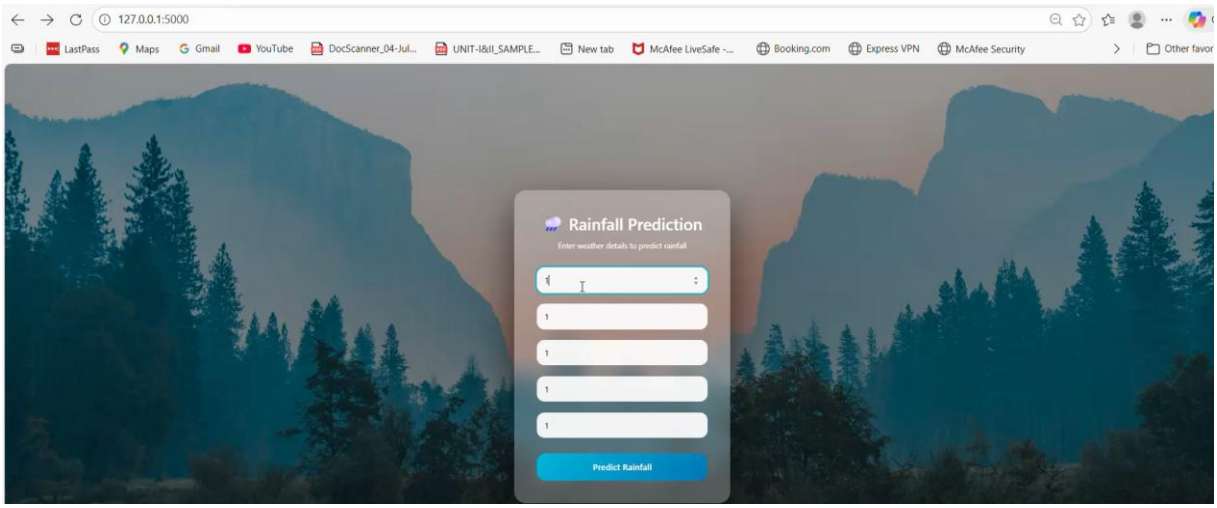
The frontend is developed using HTML, CSS, and JavaScript with weather-based animations. Three HTML files are created inside the **templates** folder.

1. index.html

This is the **home page** of the Rainfall Prediction System.

Features:

- Project Title: **Rainfall Prediction System**
- Attractive weather-themed background
- Short description of rainfall prediction
- “Predict” button to start the prediction process



Purpose:

- Acts as the landing page
- Redirects the user to the prediction process handled by Flask

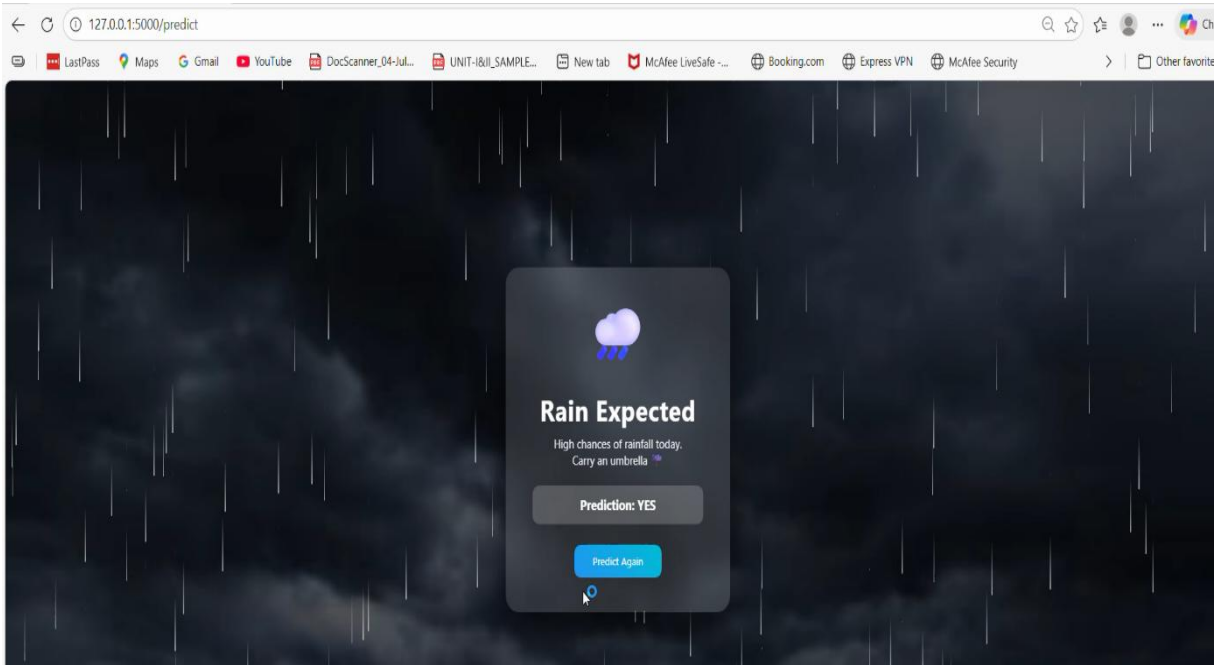
2. chance.html

This page is displayed when the model predicts **rainfall**.

Displayed Output:

- Animated rainfall background
- Message:

  "High chances of rainfall today. Carry an umbrella
- Prediction Result: **YES**
- "Predict Again" button to return to the home page



Purpose:

- Visually communicates rainfall prediction
- Enhances user understanding using animations

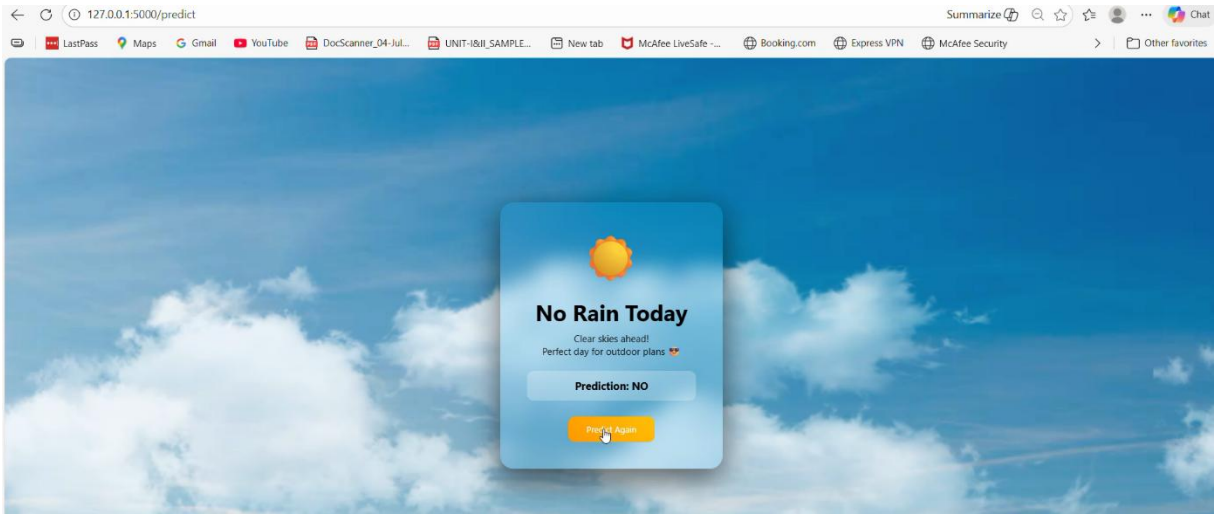
3. nochance.html

This page is displayed when the model predicts **no rainfall**.

Displayed Output:

- Sunny background with moving clouds
- Message:

  "Clear skies ahead! Perfect day for outdoor plans "
- Prediction Result: **NO**
- "Predict Again" button to return to the home page



Purpose:

- Clearly displays no-rain prediction
- Provides an engaging and informative user experience