Report on Rainfall Prediction Model

INTRODUCTION

Rainfall prediction models have a wide range of potential applications. They are essential for agricultural planning, disaster management, and water resources management. Accurate and timely rainfall predictions can help farmers plan their crop cycles, irrigation schedules, and other farming practices, which can lead to increased yields and better resource management. In areas where rainfall can cause floods, landslides, or other natural disasters, rainfall prediction models can be used to warn people and prepare them for the upcoming events. Water resources management depends on accurate rainfall predictions to help determine water supply and demand, and to make decisions on how to allocate water resources. Additionally, rainfall prediction models are also used in weather forecasting, climate research, and energy production, which can all benefit from accurate rainfall predictions. The goal of this report is to detail the development of a rainfall prediction model based on 3 years of historical data from Bangalore, using daily weather data from open-meteo.com.

DATA COLLECTION AND PREPROCESSING

The raw data required significant preprocessing,which is described in the **data\_cleaning.py** file. The initial three rows of the file were ignored as the fourth row was the header. Additionally, the columns containing information about time and weathercode were excluded from the dataset as they seemed to be irrelevant for prediction. Furthermore, the units that were specified alongside the column names were removed. Lastly, any rows that contained 'NaN' values were eliminated from the dataset.

MODEL SELECTION AND TRAINING

After the data was processed and stored in the **weather\_clean.csv** file, a list of input features was created with all the available columns from the file except, precipitation\_sum, which was the target variable to ensure that the model was only trained on input features. The data was then split into training and testing datasets using the train\_test\_split() function from Scikit-learn. The train\_test\_split() function takes the data dataframe and a test\_size parameter of 0.2, indicating that 20% of the data will be used for testing and the remaining 80% will be used for training. The random\_state parameter was set to 42 to ensure that the data is split in the same way each time the code was run.The training and testing datasets were assigned to two separate variables train\_data and test\_data. The train\_data variable contained 80% of the data, and test\_data contained the remaining 20% of the data. These datasets was be used to train and evaluate a machine learning model for predicting precipitation.

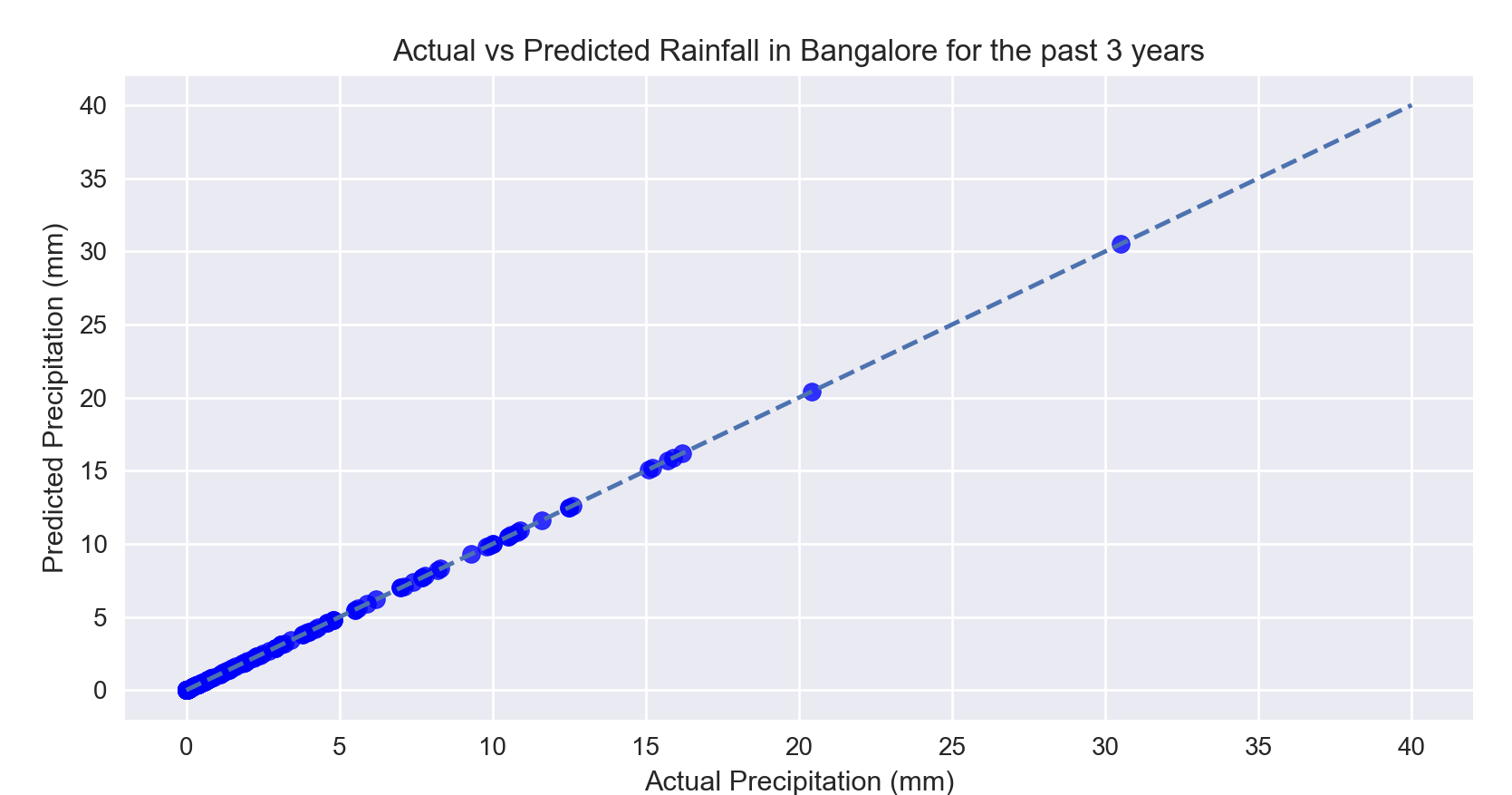
A linear regression model was instantiated and trained on the training data, and its performance was evaluated on the testing data. Specifically, the LinearRegression() function from scikit-learn was used to create an instance of the linear regression model, and then the fit() method is called on the model, passing in the input features and target variable from the training data, to train the model on the training data.

After the model is trained, the performance of the model is evaluated on the testing data. The predict() method is called on the model, passing in the input features from the testing data, to generate predictions of the target variable. These predictions are then used to calculate the mean absolute error and root mean squared error of the model's predictions compared to the actual target variable values in the testing data, using the mean\_absolute\_error() and mean\_squared\_error() functions from scikit-learn.

RESULTS AND ANALYSIS

The model achieved an R-squared score of 1.00, a mean absolute error of 0.00, and a root mean squared error of 0.00, indicating that the model perfectly predicted the target variable on the testing data. A scatter plot was created to visualize the relationship between the actual and predicted values of the target variable, precipitation\_sum, which showed that there was no difference between the predicted values and the actual values. The x-axis represented the actual precipitation values from the testing data, and the y-axis represented the predicted precipitation values from the model.

A screenshot of the scatter plot is attached below.

****