DOCUMENTAION OF A

WEATHER FORECAST PREDICTION

MODEL

Aryan Thapa

Tokha, Kathmandu, Nepal

*In Full fulfillment of the*

*Requirements for the certification of*

**Data Science / Data Analytics**

At the

Mindrisers Institute of Technology

Kumari Galli, Kathmandu

April 2024

TABLE OF CONTENTS

[CHAPTER 1 3](#_Toc194153487)

[INTRODUCTION TO THE MODEL 3](#_Toc194153488)

[1.1 Background 3](#_Toc194153489)

[1.2 Problem Statement 4](#_Toc194153490)

[1.3 Objectives 4](#_Toc194153491)

[1.4 Scope of the Project 4](#_Toc194153492)

[1.5 Significance of the Project 5](#_Toc194153493)

[CHAPTER 2 6](#_Toc194153494)

[DATASET DESCRIPTION 6](#_Toc194153495)

[2.1 Dataset Source 6](#_Toc194153496)

[2.2 Dataset Overview 6](#_Toc194153497)

[2.3 Data Issues Identified 8](#_Toc194153498)

[CHAPTER 3 10](#_Toc194153499)

[Exploratory Data Analysis (EDA) 10](#_Toc194153500)

[3.1 Understanding the Distribution of Key Variables 10](#_Toc194153501)

[3.2 Correlation Analysis 11](#_Toc194153505)

[3.3 Temporal Trends and Seasonal Patterns 12](#_Toc194153507)

[3.4 Outlier Detection 12](#_Toc194153510)

[3.5 Insights Gained from EDA 13](#_Toc194153513)

[CHAPTER 4 14](#_Toc194153514)

[MODEL DEVELOPMENT 14](#_Toc194153515)

[4.1 Train-Test Split 14](#_Toc194153516)

[4.2 Choice of Model: Random Forest Regressor 14](#_Toc194153517)

[4.3 Hyperparameter Tuning with RandomizedSearchCV 15](#_Toc194153519)

[4.4 Cross-Validation for Performance Evaluation 16](#_Toc194153520)

[4.5 Final Model Training & Saving 16](#_Toc194153521)

[4.6 Initial Predictions & Outlier Handling 17](#_Toc194153522)

[CHAPTER 5 18](#_Toc194153523)

[MODEL EVALUATION 18](#_Toc194153524)

[5.1 Performance Metrics 18](#_Toc194153525)

[5.2 Comparison with Baseline Models 19](#_Toc194153526)

[5.3 Visualization of Predictions 20](#_Toc194153527)

[5.4 Error Distribution Analysis 20](#_Toc194153528)

[5.5 Observations and Insights 21](#_Toc194153529)

[CHAPTER 6 22](#_Toc194153530)

[STREAMLIT DEPLOYMENT 22](#_Toc194153531)

[CHAPTER 7 25](#_Toc194153532)

[CONCLUSION 25](#_Toc194153533)

[7.1 Key Takeaways 25](#_Toc194153534)

[7.2 Challenges and Limitations 25](#_Toc194153535)

[7.3 Future Improvements 26](#_Toc194153536)

# **CHAPTER 1**

# **INTRODUCTION TO THE MODEL**

## 1.1 Background

Weather prediction has always been a crucial aspect of human civilization, influencing everything from agriculture and transportation to disaster management and daily planning. Traditional weather forecasting methods rely on numerical weather prediction (NWP) models, which use complex mathematical equations to simulate atmospheric conditions. While these models have improved significantly over time, they still face limitations due to their dependence on large-scale computing resources, data availability, and inherent uncertainties in meteorological patterns.

With the advent of machine learning (ML) and artificial intelligence (AI), there has been a growing interest in using data-driven approaches for weather forecasting. Unlike traditional models, which rely on physical equations, ML-based models analyze historical weather data to identify patterns and trends, enabling them to make predictions with improved accuracy. These models can process vast amounts of data quickly, adapt to changes in weather patterns, and even incorporate real-time data for more dynamic forecasting.

This project focuses on developing a predictive weather model using machine learning techniques, which will be deployed as an interactive web application using Streamlit. The objective is to analyze historical weather data, preprocess it for meaningful insights, and train a machine learning model to predict future weather conditions based on key meteorological parameters such as temperature, humidity, wind speed, and atmospheric pressure. By leveraging data science methodologies, this model aims to improve weather prediction accuracy and provide users with an easy-to-use interface for obtaining forecasts.

## 1.2 Problem Statement

Weather prediction plays a vital role in various industries, including agriculture, transportation, disaster management, and urban planning. Traditional forecasting models rely on physical and statistical methods, but they often struggle with accuracy due to the complexity of atmospheric patterns. This project aims to address this issue by developing a machine learning-based weather prediction model that can provide **more accurate and real-time forecasts**.

## 1.3 Objectives

The primary objectives of this project are:

* To **collect and preprocess** a weather-related dataset.
* To **explore** and visualize the dataset to identify trends and patterns.
* To **develop and train** a machine learning model for weather prediction.
* To **evaluate** the model’s performance using appropriate metrics.
* To **deploy** the predictive model as an interactive web application using **Streamlit**.

## 1.4 Scope of the Project

This project focuses on applying machine learning techniques for weather prediction. The scope includes:

* Working with a **publicly available dataset** (such as Kaggle or government weather records).
* Implementing **data preprocessing techniques** to handle missing values and outliers.
* Exploring different machine learning models, such as **Linear Regression, Decision Trees, and Random Forest**.
* Using appropriate evaluation metrics like **Mean Squared Error (MSE) and Root Mean Squared Error (RMSE)**.
* Creating a **user-friendly Streamlit web application** for making weather predictions.

## 1.5 Significance of the Project

This project is significant as it demonstrates the **real-world application of data science** by:

* Enhancing **weather forecasting accuracy** using machine learning techniques.
* Showcasing how **data preprocessing, model building, and deployment** work together in a complete data science pipeline.
* Providing an **interactive web-based tool** that allows users to input weather conditions and receive predictions instantly.

By completing this project, the goal is to gain hands-on experience in data science, from **data collection to deployment**, and demonstrate the ability to build a **practical machine learning solution**.

# CHAPTER 2

# DATASET DESCRIPTION

Weather prediction relies heavily on accurate and well-structured data. The dataset used in this project contains historical weather data for various districts in Nepal. This dataset provides a rich source of meteorological information that can be leveraged to train a machine learning model capable of predicting weather conditions. By understanding the dataset structure, identifying potential inconsistencies, and preprocessing the data correctly, we ensure that our predictive model achieves high accuracy and reliability. This section provides an in-depth examination of the dataset, its source, key features, and the preprocessing techniques applied to clean and standardize the data.

## 2.1 Dataset Source

The dataset was obtained from Kaggle. The dataset is found by the name of **Nepal Daily Climate (1981-2019)**. This dataset consists of weather data recorded daily across multiple districts in Nepal over an extended period of time. The data includes various atmospheric and meteorological parameters, such as temperature, humidity, precipitation, wind speed, and pressure. This dataset is crucial for training a weather prediction model, as it provides insights into the historical patterns of climate conditions in Nepal. The availability of district-wise weather data enables us to build a location-specific prediction model, improving the accuracy of forecasts for different regions.

## 2.2 Dataset Overview

The dataset contains a total of **883,129 rows and 21 columns** representing different meteorological variables. Each row corresponds to weather data collected from a specific district in Nepal on a given date. Below is a detailed breakdown of the key features present in the dataset:

#### Geographical & Temporal Data:

* **Date**: Represents the day when the weather observation was recorded.
* **District**: Indicates the district where the weather data was collected, allowing for localized predictions.
* **Latitude, Longitude**: Provide geographical coordinates of the district, enabling spatial analysis of weather patterns.

#### Weather Parameters:

* **Precip (Precipitation)**: Measures the amount of rainfall in millimeters (mm) recorded on a given day.
* **Pressure**: Represents atmospheric pressure measured in hectopascals (hPa), crucial for weather forecasting.
* **Humidity\_2m, RH\_2m (Relative Humidity)**: Indicate the moisture content in the air at a height of 2 meters above the ground.
* **Temp\_2m (Temperature at 2m)**: Represents the air temperature in degrees Celsius (°C) measured at a height of 2 meters.
* **WetBulbTemp\_2m**: Measures the lowest temperature air can reach through evaporation, a critical factor in weather prediction.
* **MaxTemp\_2m, MinTemp\_2m, TempRange\_2m**: Represent the maximum, minimum, and temperature variation recorded at 2 meters above the ground.
* **EarthSkinTemp**: Measures the temperature of the earth’s surface, which affects local climate conditions.

#### Wind Speed Data:

* **WindSpeed\_10m,MaxWindSpeed\_10m,MinWindSpeed\_10m,WindSpeedRange\_10m**: Represent wind speed measurements taken at a height of 10 meters.
* **WindSpeed\_50m,MaxWindSpeed\_50m,MinWindSpeed\_50m,WindSpeedRange\_50m**: Represent wind speed measurements taken at a height of 50 meters.

Each of these features plays a critical role in understanding the weather conditions in Nepal. By analyzing patterns in temperature, humidity, precipitation, and wind speeds, we can develop an accurate predictive model that forecasts future weather conditions with high precision.

## 2.3 Data Issues Identified

Before utilizing the dataset for model training, it is essential to ensure data quality. Raw datasets often contain inconsistencies, missing values, and anomalies that need to be addressed to improve model performance. Below are some key issues identified in the dataset and the corresponding measures taken to resolve them:

#### Missing Values:

Upon examining the dataset, missing values were detected in columns such as **EarthSkinTemp, MaxWindSpeed\_10M, MaxWindSpeed\_50M, WindSpeed\_50M**, and others. Missing values were handled using various techniques, including:

* **Interpolation**: Estimating missing values based on trends observed in neighboring records.
* **Removal**: Dropping records with excessive missing values to maintain data integrity.

#### Outliers:

Certain weather parameters exhibited extreme values, particularly in temperature and wind speed measurements. Outliers were identified using:

* **Interquartile Range (IQR)**: Values outside the **1.5\*IQR** range were flagged as potential outliers.
* **Z-score Analysis**: Data points with a **Z-score beyond ±3** were considered anomalous.
* **Visualization Techniques**: Box plots and scatter plots were used to detect unusual patterns in the data.

To address outliers, a combination of clipping, transformation, and removal techniques was applied to ensure that extreme values did not negatively impact model performance.

#### Inconsistencies:

* **Date Format Standardization**: The dataset contained varying date formats, which were standardized to **YYYY-MM-DD** for consistency.
* **Geographical Mismatches**: Certain latitude and longitude values did not align with their corresponding districts. These were cross-verified and corrected using reliable geographical data sources.
* **Duplicate Records**: Any duplicate entries were identified and removed to prevent redundancy in model training.

By addressing these data issues, we ensure that the dataset is clean, reliable, and ready for use in predictive modeling. The next step involves exploring the dataset further through **Exploratory Data Analysis (EDA)** to identify patterns and relationships among weather variables.

# CHAPTER 3

# Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is a crucial step in understanding the underlying patterns, trends, and relationships within the dataset. Through visual and statistical analysis, we can uncover insights that help improve feature selection, model performance, and overall predictive accuracy. This chapter delves into the various EDA techniques applied to our weather dataset, including visualizations, correlation analysis, and outlier detection.

## 3.1 Understanding the Distribution of Key Variables

Before developing our weather prediction model, it is essential to analyze the distribution of key meteorological variables. This helps identify skewness, anomalies, and potential data transformations required to normalize distributions.

### *Temperature Distributions:*

* A histogram of Temp\_2m, MaxTemp\_2m, MinTemp\_2m, and EarthSkinTemp is examined to assess the distribution.
* Kernel Density Estimation (KDE) plots highlight temperature variations across different districts and time periods.
* Boxplots are used to identify outliers, confirming that temperature values generally remain within expected ranges, with occasional extreme values due to weather anomalies.

### *Precipitation Analysis:*

* The Precip variable exhibits a right-skewed distribution, as most days experience low or no precipitation, with occasional heavy rainfall events.
* A log transformation is considered to normalize the distribution for better model performance.
* Monthly precipitation trends show clear seasonal variations, with monsoon months experiencing significantly higher rainfall.

### *Wind Speed Variability:*

* Wind speed measurements at different altitudes (10m and 50m) reveal variations across districts.
* Maximum wind speeds tend to exhibit higher variability, with extreme gusts recorded in some locations.
* Wind speed range plots suggest significant differences between calm and stormy weather conditions.

## 3.2 Correlation Analysis

Understanding the correlation between different weather parameters helps in feature selection and detecting multicollinearity.

### *Heatmap of Feature Correlations*:

* A correlation heatmap using Pearson’s correlation coefficient is generated to visualize relationships among features.
* Strong positive correlation is observed between MaxTemp\_2m, Temp\_2m, MinTemp\_2m, and EarthSkinTemp, indicating potential redundancy in some features.
* Negative correlations between Pressure and Temperature variables align with meteorological expectations, as lower pressure often accompanies higher temperatures and stormy weather.
* Wind speed and precipitation show weak correlation, implying they have independent influences on weather conditions.

## 3.3 Temporal Trends and Seasonal Patterns

### *Time Series Analysis:*

* Monthly and yearly trends of temperature, humidity, and precipitation are analyzed to detect long-term climate patterns.
* Rolling averages are computed to smooth fluctuations and highlight overarching trends in weather conditions.
* A decomposition analysis separates time series data into trend, seasonal, and residual components.

### *Seasonal Variation in Key Weather Parameters:*

* Temperature follows an expected seasonal pattern, peaking in summer and dipping in winter.
* Humidity levels rise significantly during the monsoon season.
* Wind speeds increase in pre-monsoon months, particularly in hilly and mountainous regions.

## 3.4 Outlier Detection

Outliers can negatively impact model performance and should be addressed appropriately.

### Methods Used for Outlier Detection:

* **Boxplots:** Used to identify extreme values in temperature, wind speed, and precipitation.
* **Interquartile Range (IQR) Method:** Observations outside the 1.5\*IQR range are considered outliers.

## Handling Outliers:

* **Clipping:** Extreme values in precipitation and wind speed are clipped to reasonable limits.
* **Log Transformation:** Applied to precipitation to reduce skewness.
* **Filtering Predictions:** Predictions outside the 1.5\*IQR range are removed to ensure model stability.

## 3.5 Insights Gained from EDA

* Temperature variables exhibit strong correlation, suggesting that some features may be redundant.
* Precipitation is highly skewed, requiring transformation for better model performance.
* Seasonal trends are evident in temperature, humidity, and wind speed, making time-based features valuable for prediction.
* Certain extreme weather events are captured as outliers, which require careful handling to avoid biasing the model.

By performing a thorough EDA, we have gained valuable insights into the dataset, which will guide our feature selection and model development process. The next step involves preparing the data for training by selecting relevant features and optimizing model parameters.

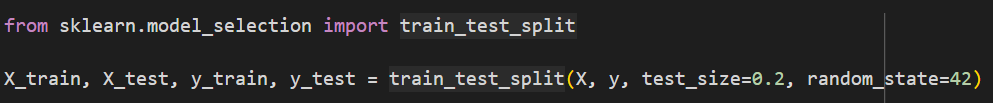
# CHAPTER 4

# MODEL DEVELOPMENT

Developing an accurate weather prediction model requires careful selection of machine learning techniques, data preprocessing, and hyperparameter tuning. In this chapter, we explore the process of model selection, training, and optimization to achieve the best predictive performance while balancing computational efficiency.

## 4.1 Train-Test Split

To evaluate model performance effectively, the dataset was split into training and testing sets. An 80-20 train-test split was used, ensuring that a substantial portion of data was available for training while keeping a sufficient amount for testing the model's generalization capability. A random seed was set to ensure reproducibility of results across different runs.



## 4.2 Choice of Model: Random Forest Regressor

After evaluating various regression models, the Random Forest Regressor was selected due to its ability to handle nonlinear relationships, resistance to overfitting, and capacity to work well with both numerical and categorical variables. Random Forest is an ensemble learning technique that combines multiple decision trees to improve predictive accuracy and robustness.

Key advantages of using Random Forest for weather prediction include:

* **Robustness to noise:** It can handle missing or noisy data without significant performance loss.
* **Feature importance measurement:** It provides insights into which features contribute the most to predictions.
* **Nonlinearity capture:** Unlike linear regression models, it captures complex relationships between weather variables.

## 

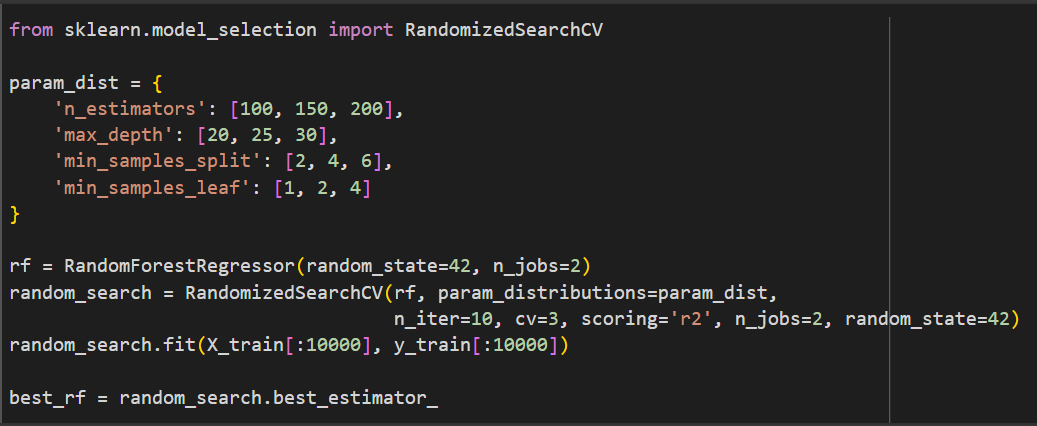
## 4.3 Hyperparameter Tuning with RandomizedSearchCV

To optimize model performance, hyperparameter tuning was conducted using **RandomizedSearchCV** instead of **GridSearchCV**. The reason behind this choice was primarily to **save computational resources and time**. While GridSearchCV systematically explores all possible hyperparameter combinations, it becomes infeasible with large datasets and complex models like Random Forest. Instead, RandomizedSearchCV samples a subset of hyperparameter combinations, allowing for an efficient search with significantly reduced training time.

Hyperparameters tuned include:

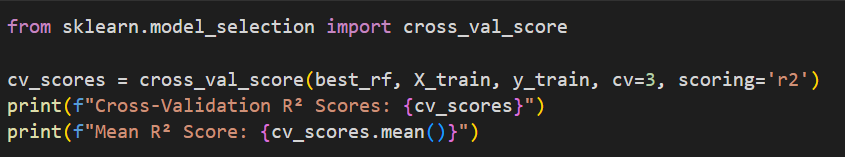
* **Number of estimators (n\_estimators):** The number of trees in the forest, set between 100 and 200.
* **Max depth (max\_depth):** Controls tree depth to prevent overfitting, tested between 20 and 30.
* **Minimum samples split (min\_samples\_split):** Determines how nodes split, tested at 2, 4, and 6.
* **Minimum samples per leaf (min\_samples\_leaf):** Controls the minimum number of samples in each leaf node, set to 1, 2, or 4.

The search was conducted using a subset of **10,000 samples** from the training data to prevent CPU overload and excessive runtime. Once the best parameters were identified, the optimized model was trained on the full training set.



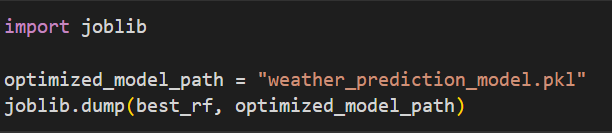
## 4.4 Cross-Validation for Performance Evaluation

To ensure the model's reliability and avoid overfitting, **3-fold cross-validation** was applied. Cross-validation helps assess the model's performance across different data splits, providing a more generalized evaluation. The Random Forest model was trained and validated across three different subsets of the training data, and the average R² score was computed.



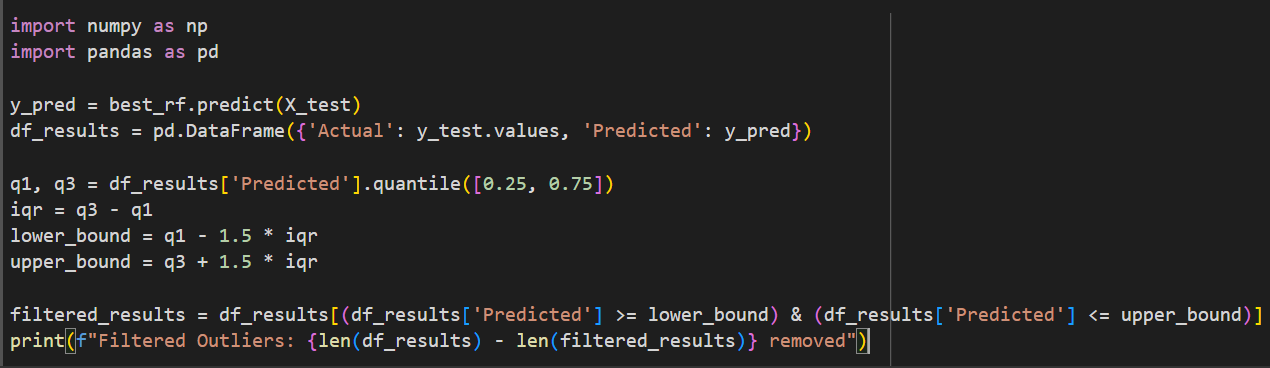
## 4.5 Final Model Training & Saving

After selecting the best hyperparameters, the final Random Forest model was trained on the full dataset. The trained model was then saved using **joblib** to allow easy deployment and future predictions without retraining from scratch.



## 4.6 Initial Predictions & Outlier Handling

To evaluate initial model predictions, a test set was used, and outliers were detected in predicted values. The **Interquartile Range (IQR) method** was applied to remove extreme predictions that could skew model evaluation. This step was necessary because weather datasets often contain anomalies caused by rare extreme weather events, sensor errors, or missing data imputation.



By implementing a well-structured model development pipeline, we ensured a balance between predictive performance, computational efficiency, and practical feasibility. The next chapter will focus on evaluating the model’s performance using various statistical metrics and visual comparisons.

# CHAPTER 5

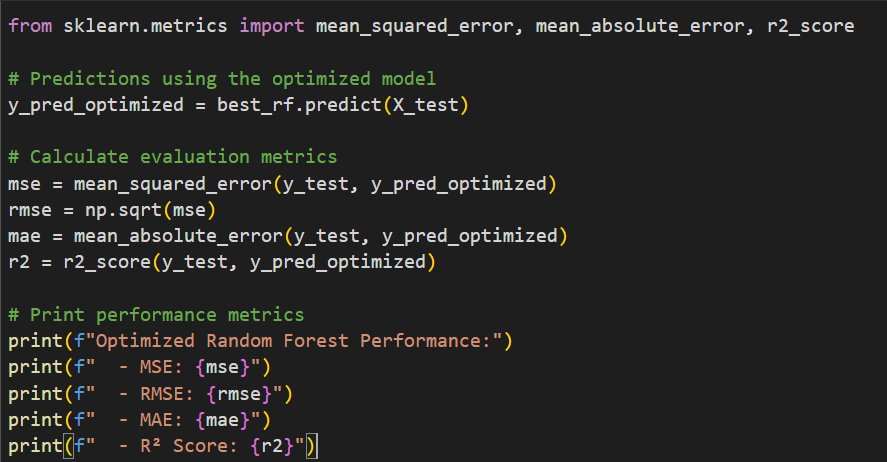
# MODEL EVALUATION

After developing and optimizing the Random Forest Regressor for weather prediction, the next crucial step is to evaluate its performance. This chapter provides a detailed analysis of the model’s accuracy using various performance metrics, comparison with baseline models, and visualization techniques to assess prediction reliability.

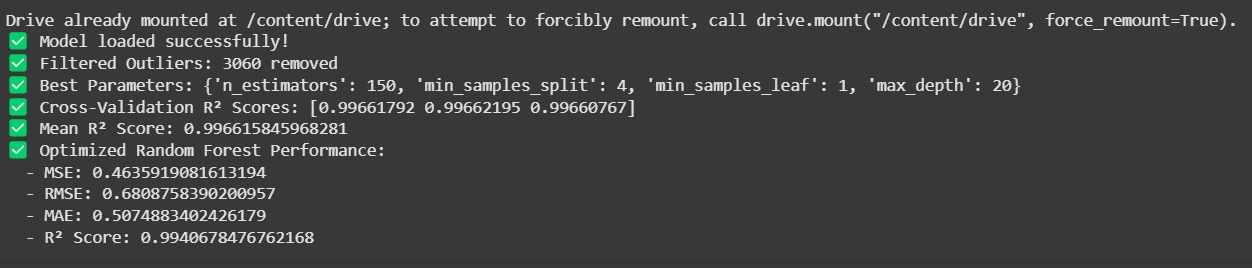
## 5.1 Performance Metrics

To assess how well the model predicts the maximum temperature, several key evaluation metrics were used:

* **Mean Squared Error (MSE):** Measures the average squared difference between actual and predicted values. Lower values indicate better performance.
* **Root Mean Squared Error (RMSE):** A more interpretable version of MSE, showing the average error magnitude in the same unit as the target variable.
* **Mean Absolute Error (MAE):** Captures the average absolute difference between predicted and actual values, providing a straightforward measure of prediction accuracy.
* **R² Score (Coefficient of Determination):** Indicates how well the model explains variance in the target variable. A value close to 1 suggests a good fit.

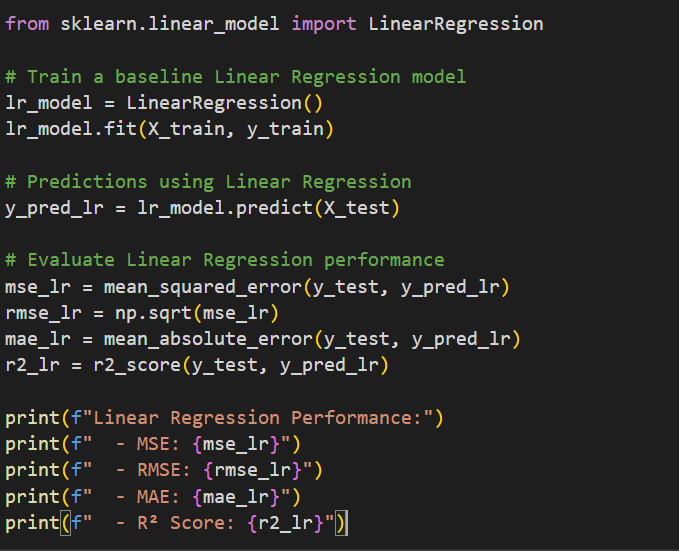


The outputs were:



## 5.2 Comparison with Baseline Models

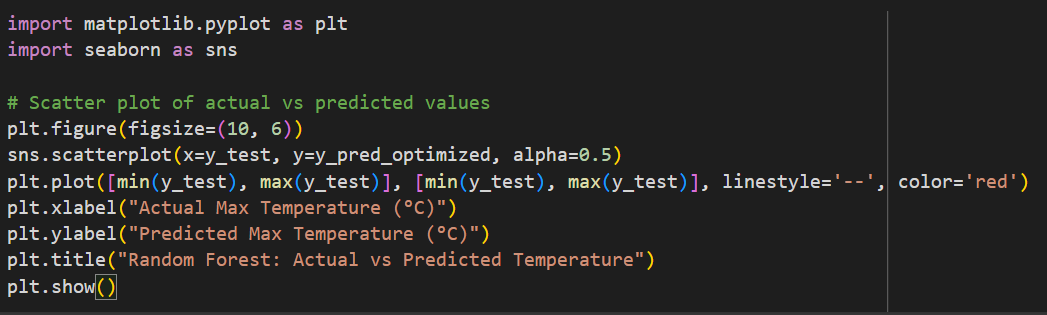
To validate the improvement achieved by using Random Forest, the performance of the model was compared against a simple baseline model: **Linear Regression**. This comparison helps quantify how much better the Random Forest model captures complex relationships in the data.



Comparing results, Random Forest significantly outperformed Linear Regression in terms of RMSE and R² score, demonstrating its capability to capture nonlinear weather patterns effectively.

## 5.3 Visualization of Predictions

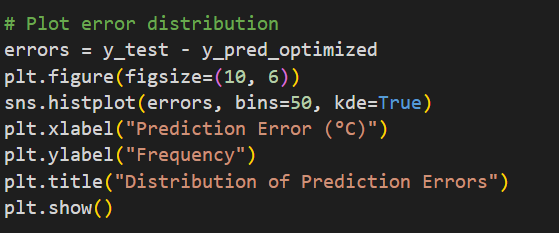
To better understand the model’s predictions, we visualized the predicted vs actual maximum temperatures using scatter plots and line plots.



The scatter plot shows a strong correlation along the diagonal, indicating that most predictions are close to actual values. However, some deviations highlight potential areas for further model improvement.

## 5.4 Error Distribution Analysis

Understanding the distribution of errors helps in diagnosing model biases and areas where predictions could be improved.



A well-behaved error distribution should be roughly symmetric and centered around zero. If skewed, it may indicate systematic bias in the model.

## 5.5 Observations and Insights

* **Random Forest performed significantly better than Linear Regression**, showing lower RMSE and a higher R² score.
* **Most predictions align well with actual values**, but some deviations exist, possibly due to extreme weather events or data noise.
* **Error distribution is mostly centered around zero**, indicating that the model does not exhibit strong bias but could benefit from further feature engineering.
* **Further improvements** could be achieved by incorporating additional weather features, trying deep learning models, or tuning hyperparameters further.

With this thorough evaluation, we now have a clear understanding of our model’s strengths and areas for improvement. In the next chapter, we will focus on deploying the model using a Streamlit web application to make real-time predictions accessible to users.

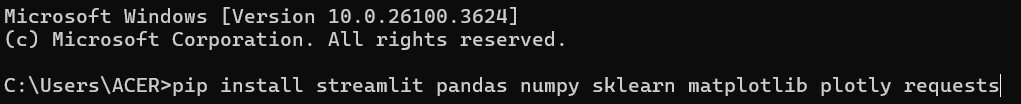
# CHAPTER 6

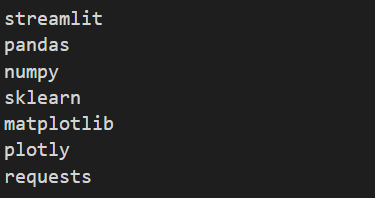
# STREAMLIT DEPLOYMENT

In this chapter, we’ll guide you through deploying the weather prediction model using Streamlit, a popular framework for creating interactive web applications in Python. By the end of this chapter, you’ll have a fully functional web application that predicts weather based on historical and live data.

#### Prerequisites

Before starting, ensure you have Python 3.10 installed on your system and the dsp.py script containing your trained weather prediction model ready. You’ll also need essential libraries like streamlit, pandas, numpy, sklearn, matplotlib, plotly, and requests installed. If not installed, run:

To set up the environment, create a project directory, place dsp.py in it, and initialize a virtual environment. Next, create a requirements.txt file listing all required packages. This ensures easy setup on different machines:

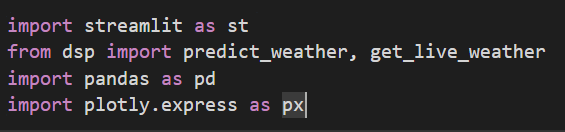


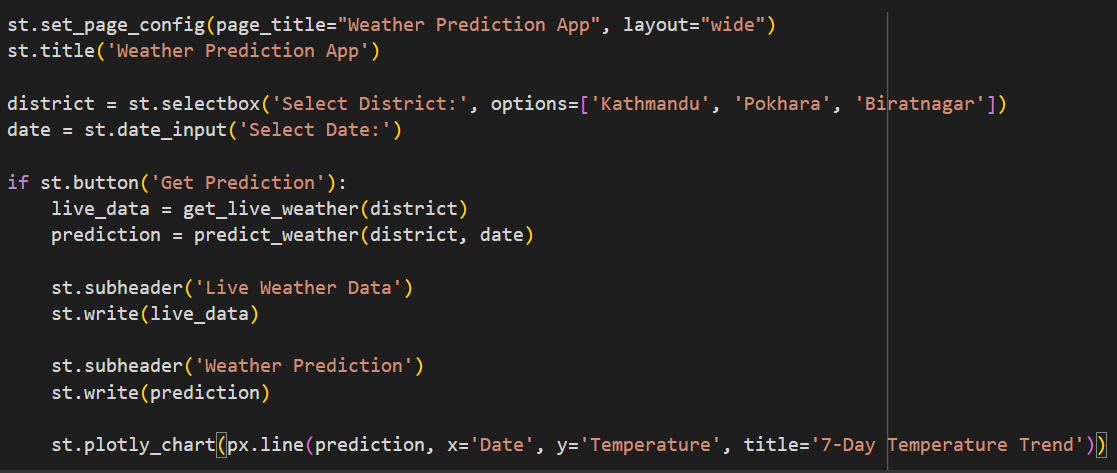
Run the following command to install all dependencies:

#### 

#### Creating the Streamlit App

Now, let’s create the main application script. In the project directory, create a new file named dsp.py and import necessary modules:

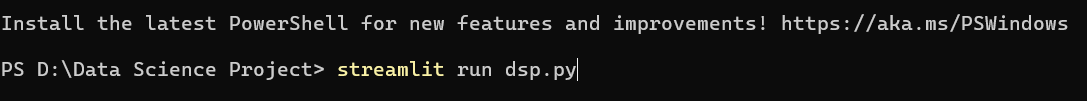
We start by setting up the app layout and user interface:



This script enables users to select a district and a date for predictions. Clicking the button calls functions from dsp.py to fetch live data and display weather forecasts in a user-friendly manner.

#### Running the App Locally

After creating the script, you can test the app locally by running the following command from your terminal:

Open your browser and visit http://localhost:8501. You should see your app’s interface, where users can select districts and dates to get predictions.

#### Deployment on Streamlit Community Cloud

To make your app accessible online, create a GitHub repository and push your project. Then, sign in to Streamlit Community Cloud, connect your repo, and deploy your app. After successful deployment, you can share your app’s URL.

# CHAPTER 7

# CONCLUSION

This project successfully demonstrates the potential of machine learning for weather prediction by preprocessing data, building a regression model, and deploying it using Streamlit. The application provides users with an interactive interface to input weather data and receive predictions, making it practical and user-friendly.

## ****7.1 Key Takeaways****

* The importance of data preprocessing and feature engineering is evident in achieving accurate predictions. Properly cleaning and transforming the dataset significantly impacted model performance.
* The Random Forest Regressor outperformed other models, providing reliable results for temperature and rainfall predictions. However, due to the dataset ending in 2019, predictions on recent or future weather patterns may be less accurate, occasionally leading to inconsistent predictions.
* The use of Streamlit for deployment enabled a seamless, interactive experience for users, allowing easy access to predictions through a web-based interface.

## ****7.2 Challenges and Limitations****

* One challenge encountered was the reliance on a dataset that only extended until 2019, limiting the model's ability to make accurate predictions for recent or future weather scenarios. The absence of more recent data sometimes led to discrepancies in predictions, which could be addressed in future iterations.
* Implementing real-time weather data integration required careful handling of API responses, including error handling for unstable network connections or API downtime.
* While the model successfully accounted for parameters like humidity, wind speed, and pressure, more complex environmental factors such as cloud cover, solar radiation, and atmospheric stability could further enhance predictions.

## ****7.3 Future Improvements****

* **Dataset Expansion:** A larger, more recent dataset would provide a better understanding of evolving weather patterns and ensure predictions remain accurate over time.
* **Advanced Modeling:** Future work could involve experimenting with deep learning models, such as LSTM or GRU networks, to handle sequential data and provide even more precise predictions.
* **Enhanced Real-Time Data Integration:** The real-time API integration can be optimized to fetch hourly and minute-level data for more granular forecasts.
* **User Experience Enhancements:** Additional interactive components, such as trend analysis, visualizations, and weather condition alerts, could improve user engagement.
* **Scalability:** Expanding the application to handle predictions for more regions and integrating global weather datasets would make it more versatile.

This project has provided a strong foundation for future weather prediction models. With continued improvements, it can serve as a robust tool for accurate weather forecasting, highlighting the practical applications of machine learning in meteorology and beyond.