PROJECTREPORT

2025

Weather Prediction
&
Analysis Web Application

PREPARED FOR:

DR. MD. RASHID AL ASIF

ASSISTANT PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIVERSITY OF BARISHAL

PREPARED BY:

MD. RASEL MOLLA

ROLL NO: 09-006-07

BATCH: DS-06

COURSE: DATA SCIENCE WITH PYTHON

Abstract

Summary

Weather prediction and analysis have become increasingly important in various sectors, including agriculture, disaster management, and urban planning. This project, developed as part of the "Data Science with Python" course, aims to provide an interactive platform where users can analyze historical weather data and make predictions about future weather conditions. The system allows users to upload weather datasets, visualize trends, and utilize machine learning models for temperature and rainfall predictions.

The project is built using Django for the backend and Jupyter Notebook for model development. The core dataset, collected from Kaggle, spans from 1901 to 2023, providing a comprehensive historical perspective. Using scikit-learn’s Linear Regression and K-Means clustering algorithms, this system identifies key weather trends, such as the months with the highest and lowest rainfall or temperature variations.

In addition to predictions, the platform includes a weather visualization feature that generates various graphical representations, including scatter plots, line graphs, and bar charts. The weather analysis section offers insights into long-term patterns, helping users understand how weather conditions fluctuate over different time periods. By training machine learning models on historical data, the system can predict future rainfall based on month, year, and temperature or future temperature based on month, year, and rainfall.

This project combines multiple functionalities into a single platform, making it useful for both researchers and general users. The use of Python-based data science libraries, such as Pandas, Matplotlib, Seaborn, and scikit-learn, ensures accurate analysis and visualization of weather patterns.

Purpose

The primary objective of this project is to leverage data science and machine learning to improve weather analysis and forecasting capabilities. By utilizing historical weather data, the system provides users with a powerful tool to:

Predict Weather Conditions: Forecast rainfall or temperature based on historical trends.

Analyze Trends: Identify high and low rainfall/temperature periods using machine learning techniques.

Visualize Data: Generate graphical representations for easy interpretation of weather patterns.

Enable User Interaction: Allow users to upload datasets and gain insights through statistical models.

This project not only demonstrates data-driven decision-making but also provides a practical implementation of machine learning algorithms in meteorological analysis. The results can be used for agricultural planning, urban development, and climate change studies, making this system an essential tool for both research and real-world applications.

Table of Contents

Introduction

Weather prediction and analysis play a critical role in various sectors, including agriculture, disaster management, transportation, and environmental research. Understanding past weather patterns and forecasting future trends can help mitigate the risks associated with extreme weather events, improve agricultural planning, and enhance decision-making processes in weather-dependent industries.

This project, developed as part of the "Data Science with Python" course, focuses on creating an interactive web-based platform that enables users to analyze and predict weather conditions using machine learning techniques. The system allows users to upload weather datasets, visualize trends, and perform predictive analysis based on historical weather data. The dataset used in this project, collected from Kaggle, contains weather records from 1901 to 2023, including temperature, rainfall, month, and year data. By utilizing Python-based data science libraries and machine learning models, the system provides valuable insights into long-term weather patterns.

Project Scope and Objectives

The primary goal of this project is to implement machine learning algorithms to analyze historical weather data and predict future weather conditions. The system includes the following key features:

Weather Prediction: Using Linear Regression, the platform predicts rainfall based on temperature, month, and year or temperature based on rainfall, month, and year.

Weather Visualization: Users can generate various graphical representations (scatter plots, line graphs, bar charts) to understand seasonal trends in temperature and rainfall.

Weather Analysis: Through techniques like K-Means clustering, the system

identifies periods of high and low rainfall/temperature over different years and months.

User-Interactive Dataset Upload: Users can upload a dataset in CSV format, which is then processed for visualization and analysis.

Significance of the Project

This project is significant because it applies data science and machine learning to meteorological data, making weather forecasting more accessible and data-driven. Traditional weather forecasting methods rely on complex atmospheric models, but this project demonstrates how historical data analysis can provide reliable insights with minimal computational resources. By integrating machine learning models, the system allows users to make data-driven predictions without requiring in-depth meteorological knowledge.

Moreover, climate change has increased the variability of weather patterns, making historical data analysis more crucial than ever. This project provides a tool that can help researchers, policymakers, and farmers make informed decisions based on past and present climate trends. It also serves as a practical demonstration of Python-based data science applications, reinforcing concepts such as data preprocessing, regression analysis, and clustering techniques.

Methodology

Overview

This project follows a structured methodology that integrates data collection, preprocessing, analysis, model training, and web-based deployment to create a weather prediction and analysis system. The approach involves using historical weather data from Kaggle, applying machine learning models for prediction, and developing an interactive web interface for visualization and analysis.

Tools and Technologies

The project is implemented using a combination of programming languages, libraries, web frameworks, and data processing tools. Below are the core technologies used:

1. Programming Language

Python – Used for data preprocessing, machine learning model development, and backend processing in Django.

2. Required Libraries

Pandas – For data handling and preprocessing.

NumPy – For numerical computations.

Matplotlib & Seaborn – For data visualization.

Scikit-learn – For machine learning models (Linear Regression and K-Means Clustering).

Joblib – For saving and loading trained machine learning models.

3. Web Framework

Django – Used for building the web-based interface that allows users to upload datasets, visualize weather trends, and make predictions.

4. Frontend Technologies

HTML, CSS, JavaScript – Used to design the interactive web interface.

5. Database

SQLite – The built-in database of Django, used for managing user-uploaded files and storing processed data.

6. Codding Environment

VS Studio & Jupyter Notebook – Used for initial data exploration, model training, and testing before integrating models into the Django web app.

7. Data Source

Kaggle – Used as the historical weather data source. The dataset includes four essential columns: Year, Month, Rainfall, and Temperature.

Project Steps

The development of this project follows a structured pipeline to ensure the accuracy of predictions, meaningful analysis, and a user-friendly interface. The project is divided into six major steps, from data collection to deployment.

Step 1: Data Collection & Preprocessing

- The dataset is collected from Kaggle, covering weather data from 1901 to 2023.
- It is stored in the static/dataset folder for access during model training and prediction.

The dataset contains four essential columns:

Year: Represents the year of observation (repeated for each month).

Month: Represents the month (values from 1 to 12).

Rainfall: Monthly recorded rainfall (in mm).

Temperature: Monthly average temperature (in °C).

Data Cleaning:

- Missing values are handled using mean imputation (if any).
- Data is checked for inconsistencies, such as negative rainfall or temperature values.

Step 2: Model Training

Machine learning models are trained using scikit-learn's Linear Regression to predict temperature and rainfall.

Temperature Prediction Model (temp_model.joblib)

Input: *Year, Month, Rainfall*

Output: *Predicted Temperature*

Rainfall Prediction Model (rain_model.joblib)

Input: *Year, Month, Temperature*

Output: *Predicted Rainfall*

The models are trained once and stored using Joblib, allowing the system to use them without retraining.

Code for Model Training:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import joblib

# Load dataset
df = pd.read_csv('static/dataset/weather_dataset.csv')

# Train Rainfall Prediction Model
X_rain = df[['Year', 'Month', 'Temperature']]
y_rain = df['Rainfall']
```

```
X_train_r, X_test_r, y_train_r, y_test_r = train_test_split(X_rain, y_rain,
test_size=0.2, random_state=42)
rain_model = LinearRegression()
rain_model.fit(X_train_r, y_train_r)

joblib.dump(rain_model, 'rain_model.joblib')

# Train Temperature Prediction Model
X_temp = df[['Year', 'Month', 'Rainfall']]
y_temp = df['Temperature']

X_train_t, X_test_t, y_train_t, y_test_t = train_test_split(X_temp, y_temp,
test_size=0.2, random_state=42)
temp_model = LinearRegression()
temp_model.fit(X_train_t, y_train_t)

joblib.dump(temp_model, 'temp_model.joblib')
```

Step 3: Dataset Upload & Visualization

Users upload a dataset through the Django web interface.

The system validates the dataset for required columns: Year, Month, Rainfall, Temperature.

The following visualizations are generated:

- **Rainfall vs. Temperature scatter plots**
- **Monthly temperature trends over different years**
- **Annual average rainfall trends**

Code for Data Visualization:

```
import matplotlib.pyplot as plt
import seaborn as sns

def generate_visualizations(df):
    plt.figure(figsize=(8, 5))
    sns.scatterplot(x=df['Temperature'], y=df['Rainfall'])
    plt.title("Rainfall vs Temperature")
    plt.xlabel("Temperature (°C)")
    plt.ylabel("Rainfall (mm)")
    plt.savefig("static/graphs/rain_vs_temp.png")
```

Step 4: Predictive Analysis

Users can enter the Year, Month, and either Rainfall or Temperature to predict the missing value using the trained models.

If Rainfall Prediction is selected → The system takes Year, Month, Temperature and predicts Rainfall.

If Temperature Prediction is selected → The system takes Year, Month, Rainfall and predicts Temperature.

Code for Making Predictions:

```
import joblib

rain_model = joblib.load('rain_model.joblib')
temp_model = joblib.load('temp_model.joblib')

def predict_weather(year, month, input_value, prediction_type):
    if prediction_type == 'rainfall':
        predicted_value = rain_model.predict([[year, month, input_value]])[0]
        return f"Predicted Rainfall: {predicted_value:.2f} mm"
    else:
        predicted_value = temp_model.predict([[year, month, input_value]])[0]
        return f"Predicted Temperature: {predicted_value:.2f} °C"
```

Step 5: Weather Analysis using Machine Learning

K-Means Clustering is used to group months based on rainfall and temperature levels.

Seasonal Trend Analysis: Identifies which months have the highest or lowest rainfall and temperature.

Regression Analysis: Examines the long-term trend in temperature and rainfall over the years.

Code for K-Means Clustering:

```
from sklearn.cluster import KMeans

def cluster_weather_data(df):
    kmeans = KMeans(n_clusters=3, random_state=42)
    df['Cluster'] = kmeans.fit_predict(df[['Rainfall', 'Temperature']])
    return df
```

Step 6: Web Deployment & User Interaction

The Django web app integrates all functionalities, allowing users to:

Upload datasets for analysis.

View visualizations of past weather trends.

Predict temperature or rainfall for a given month and year.

The web app is designed for ease of use, making it accessible to researchers, students, and policymakers.

Conclusion

This project successfully implements weather data analysis and prediction using machine learning techniques and data visualization. By leveraging historical weather data from 1901 to 2023, we developed predictive models that estimate temperature and rainfall for any given year and month. The project also provides detailed statistical insights into long-term weather patterns through trend analysis and clustering methods.

One of the key strengths of this project is its data-driven approach to weather forecasting. By applying Linear Regression, we were able to build models that learn from past weather trends and make accurate predictions. Additionally, K-Means Clustering allows for the identification of seasonal patterns, highlighting which months experience the highest or lowest temperatures and rainfall.

Challenges and Limitations

Despite the project's success, some challenges were encountered:

Data quality: The accuracy of predictions heavily depends on the quality and completeness of the dataset.

Model generalization: While Linear Regression provides a simple and interpretable model, it may not capture complex non-linear weather patterns as effectively as advanced deep learning models.

External factors: Weather patterns are influenced by external variables like global climate change, which are not included in our dataset.

References / Sources

1. Dataset Source

The historical weather dataset used in this project was collected from Kaggle.

Kaggle: https://www.kaggle.com/

2. Machine Learning Techniques

Linear Regression:

Scikit-Learn Documentation:

https://scikit-learn.org/stable/modules/linear_model.html

K-Means Clustering:

Scikit-Learn Documentation:

https://scikit-learn.org/stable/modules/clustering.html#k-means

3. Python Libraries Used

Pandas: Data processing and manipulation – https://pandas.pydata.org/

Matplotlib & Seaborn: Data visualization – https://matplotlib.org/, https://seaborn.pydata.org/

Scikit-Learn: Machine learning models – https://scikit-learn.org/

4. Web Framework

Django Framework: Used for building the web application – https://www.djangoproject.com/

5. Development Environment

Jupyter Notebook: Used for model training and data analysis – https://jupyter.org/

Python 3: Primary programming language – https://www.python.org/