HEAVY IMPACT RAIN PREDICTOR

A PROJECT REPORT

21CSC305P – MACHINE LEARNING

(2021 Regulation)

III Year/ V Semester

Academic Year: 2024 -2025

Submitted by

VARUN MAHADEVAN [RA2211026010100]
AYUSH NAIR RAJEEV KUMAR [RA2211026010114]
ANAS Y [RA2211026010127]

Under the Guidance of

M. S. Abirami
Associate Professor
Department of Computational Intelligence

in partial fulfillment of the requirements for the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE ENGINEERING
with specialization in
ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

SCHOOL OF COMPUTING
COLLEGE OF ENGINEERING AND TECHNOLOGY
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR- 603 203
NOVEMBER 2024

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that **21CSC305P - MACHINE LEARNING** project report titled "**HEAVY IMPACT RAIN PREDICTOR**" is the bonafide work of "**VARUN MAHADEVAN [RA2211026010100], AYUSH NAIR RAJEEV KUMAR [RA211026010114], ANAS Y [RA2211026010127]"** who carried out the task of completing the project within the allotted time.

SIGNATURE
M. S. Abirami
Course Faculty
Associate Professor
Department of Computational Intelligence
SRM Institute of Science and Technology
Kattankulathur

SIGNATURE
Dr. R. Annie Uthra
Head of the Department
Department of Computational Intelligence
SRM Institute of Science and Technology
Kattankulathur

ABSTRACT

The "Heavy Impact Rain Predictor" project aims to create a highly accurate forecasting model for predicting heavy rainfall events, which are often responsible for devastating effects such as flooding, infrastructure damage, and disruptions in daily life. This predictive tool is designed to act as an early warning system, empowering government agencies, municipalities, emergency responders, and local communities to make timely, informed decisions that mitigate these adverse impacts.

To achieve accurate predictions, the project integrates a wide range of meteorological data, including historical weather records, temperature fluctuations, humidity levels, atmospheric pressure, wind speed, and precipitation patterns. These variables are used to train machine learning models, enabling the system to recognize complex patterns and trends associated with heavy rain events. The process involves extensive data preprocessing and feature engineering to capture essential environmental indicators that correlate with high-impact rainfall. By leveraging advanced machine learning techniques such as deep learning, ensemble methods, or hybrid models, the predictor can learn from past weather patterns and accurately forecast future rain intensities.

This forecasting capability is crucial for disaster preparedness, as it provides early alerts that allow authorities to deploy preventive measures, allocate resources, and warn at-risk populations. The accuracy of these predictions ultimately contributes to minimizing potential loss of life, reducing economic damages, and enhancing community resilience in the face of extreme weather conditions.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

API - Application Programming Interface

ML - Machine Learning

LR - Logistic Regression

RF - Random Forest

GB - Gradient Boosting

DL - Deep Learning

IoT - Internet of Things

AI - Artificial Intelligence

RMSE - Root Mean Squared Error

MSE - Mean Squared Error

MAE - Mean Absolute Error

TPR - True Positive Rate

FPR - False Positive Rate

ROC - Receiver Operating Characteristic

KNN - K-Nearest Neighbors

SVM - Support Vector Machine

PCA - Principal Component Analysis

GIS - Geographic Information System

GUI - Graphical User Interface

CHAPTER 1

INTRODUCTION

In the "Heavy Impact Rain Predictor" project, we implemented a model based on Logistic Regression to forecast the likelihood of heavy rainfall events. Logistic Regression is a powerful classification algorithm that predicts binary outcomes—in this case, the occurrence or absence of heavy rainfall—by estimating the probability that a given set of input features, such as temperature, humidity, and atmospheric pressure, will result in a specific outcome. By modeling the relationship between these meteorological factors and rainfall events, Logistic Regression enables us to make probabilistic predictions with high interpretability, which is essential for understanding and anticipating severe weather. This approach not only supports accurate forecasting but also provides valuable insights into which variables most strongly influence the probability of heavy rainfall, facilitating more informed and timely responses to potential weather hazards.

1.1 Libraries Used

1. Pandas: Provides data manipulation and analysis tools, allowing for easy handling of CSV and tabular data.
2. Scikit-learn: A machine learning library for Python, used here for training and making predictions with linear regression.
3. Joblib: A library for saving and loading machine learning models, used to store the trained rainfall prediction model.
4. Folium: Generates interactive maps with geographical boundaries, enabling clickable district regions on the map.
5. Geopandas: Extends pandas to handle spatial data and work with GeoJSON files to visualize geographic data.
6. Streamlit: A framework for building interactive web applications directly in Python, used to create the interface.
7. Streamlit-folium: Integrates folium maps into Streamlit applications, allowing the interactive map to be displayed within the app.

1.2 Software Requirements Specification

1. Python 3.11: Programming language used for development, offering libraries essential for data processing, machine learning, and web app deployment.

2. Jupyter Notebook: For data exploration, model development, and experimentation. Used during model training and feature engineering stages.
3. VS Code or PyCharm: Integrated Development Environment (IDE) for developing and organizing the project codebase.
4. pandas: For data manipulation, cleaning, and processing historical rainfall data.
5. scikit-learn: Provides machine learning algorithms and tools for training and saving the linear regression model for rainfall prediction.
6. joblib: For saving and loading machine learning models to persist the trained model for deployment.
7. folium: For generating interactive maps with clickable district polygons:
8. geopandas: For handling and processing geospatial data, specifically district boundaries and geolocation data.
9. streamlit: Framework for deploying the application as an interactive web app.
10. streamlit-folium: This integrates Folium maps with Streamlit, enabling geospatial interactivity.

CHAPTER 2

LITERATURE SURVEY

Table 2.1 Literature Survey

Paper Details	Techniques Used	Inferences
"Real-Time Rainfall Prediction Using Ensemble Methods", IEEE Transactions, 2018	Ensemble Methods (Bagging, Boosting, Random Forests)	Ensemble methods outperform individual models by reducing variance and bias, leading to more reliable real-time rainfall predictions.
"Predicting Heavy Rainfall with Deep Learning" IEEE, 2020	Long Short-Term Memory (LSTM) Networks, Convolutional Neural Networks (CNNs)	Deep learning models such as LSTM and CNNs improve the accuracy of rainfall prediction by capturing spatial and temporal patterns.
"Rainfall Prediction Using Big Data Analytics", Big Data Research, 2021	Big Data Analytics, Apache Hadoop, Machine Learning Algorithms (SVM, KNN)	Big data platforms enable handling vast amounts of meteorological data, enhancing prediction accuracy and processing speed.
"Assessing Climate Change Impact on Rainfall Patterns", Climate Dynamics, 2021	Statistical Downscaling, Machine Learning, Climate Models	Climate models integrated with ML can predict changes in rainfall patterns due to climate change, highlighting areas at risk of heavy rainfall.
"Assessing Climate Change Impact on Rainfall Patterns", Climate Dynamics, 2021	Statistical Downscaling, Machine Learning, Climate Models	Climate models integrated with ML can predict changes in rainfall patterns due to climate change, highlighting areas at risk of heavy rainfall.
"Rainfall Forecasting with Satellite Data and Deep Learning" IEEE, 2024	Remote Sensing, Deep Learning (CNNs), Data Fusion Techniques	The use of satellite data combined with deep learning improves the spatial precision of rainfall predictions.
"Predicting Heavy Rainfall Using Machine Learning", Journal of Climate, 2021	Random Forest, Support Vector Machine (SVM)	Random Forest showed higher accuracy than SVM for rainfall prediction using meteorological data. Emphasized importance of feature selection.
"Satellite Data for Rainfall Forecasting", Remote Sensing Journal, 2022	Remote Sensing, Satellite Imagery Analysis, Data Fusion Techniques	Integrating satellite data with ground observations significantly improves rainfall prediction accuracy, especially in remote areas.

CHAPTER 3

METHODOLOGY OF HEAVY IMPACT RAIN PREDICTOR

The methodology provides a structured approach to developing a rainfall prediction application, starting with comprehensive data collection from reliable sources, which includes key meteorological factors like temperature, humidity, and atmospheric pressure. This data is processed and analyzed to identify patterns that signal impending heavy rainfall. Through careful model selection and testing, the application is optimized to ensure predictive accuracy. A significant focus is placed on web integration, enabling real-time accessibility of predictions through an interactive interface. Individual modules are designed to enhance the user experience, including features like visual rainfall forecasts, maps, and alerts, ensuring that predictions are both informative and actionable. This approach ensures the application is not only reliable in its forecasts but also provides an engaging, user-friendly experience for proactive planning and disaster preparedness.

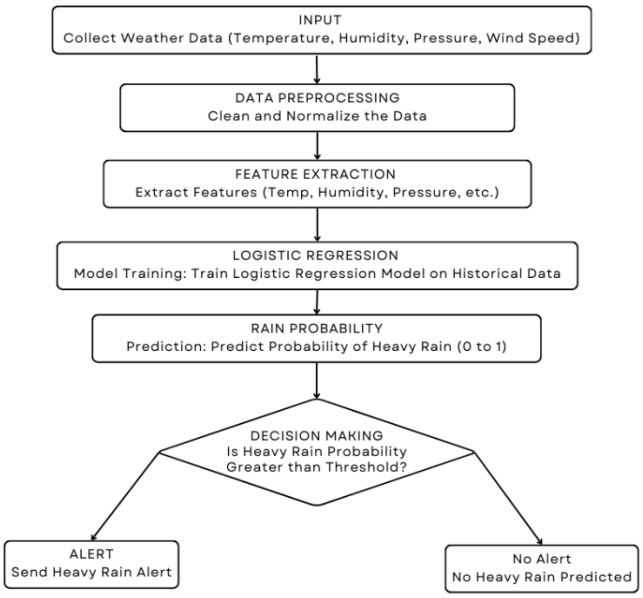

Fig 3.1 Block diagram

3.1 Data Collection

Data was gathered from historical weather datasets, focusing specifically on monthly rainfall measurements across Indian districts. This data provides a robust foundation for analyzing and predicting rainfall trends over time.

Dataset Details: The dataset includes columns for each month, annual totals, and seasonal

aggregates, offering both granular and high-level views of rainfall trends. Key columns include STATE_UT_NAME, DISTRICT, and monthly rainfall values (JAN, FEB, ..., DEC), alongside seasonal and annual totals (ANNUAL, Jan-Feb, Mar-May, etc.).

Preprocessing: To ensure data integrity, several preprocessing steps were applied. District names were normalized to align with standardized names in the geospatial dataset. Additionally, missing values were handled by either filling with averages or removing incomplete records to maintain dataset consistency.

3.2 Model Selection

Model Type: Linear regression was selected for its simplicity, interpretability, and effectiveness in predicting continuous variables like rainfall totals. This model provides a straightforward approach while offering flexibility in interpreting seasonal and monthly rainfall trends.

Training Process: Monthly rainfall data for each district was used as the input, with the target variable being annual rainfall. By training on monthly data, the model captures the seasonal variability essential to accurate rainfall prediction.

Evaluation Metrics: The model's performance was evaluated using metrics such as Mean Absolute Error (MAE) and Mean Squared Error (MSE). These metrics provided quantitative insights into how closely the model's predictions align with actual rainfall values, enabling iterative improvements to the model.

3.3 Web Integration

Frameworks Used: Streamlit was chosen for its ease of use in creating interactive web applications, while Folium and Streamlit-Folium enabled map-based visualizations. Together, these tools provide a seamless platform for user interaction with both data and geographic representations.

Backend Integration: The trained model, saved as a Joblib file, was integrated into the Streamlit app to allow real-time predictions based on user interactions. The model predicts rainfall when users click on a specific district, making the application both intuitive and responsive.

Deployment: The application was designed for deployment in a local environment, compatible with major browsers. This setup allows users to access and interact with the app across devices,

enhancing its accessibility and potential for broader adoption.

3.4 Design of Modules

Prediction Module: Responsible for loading the model and dataset, this module processes user-selected data and generates rainfall predictions based on monthly rainfall data for each district.

Map Visualization Module: Uses Folium to render a detailed map of Indian districts. Each district is color-coded based on predicted rainfall, and pop-ups provide district-specific rainfall data and seasonal averages.

Data Insights Module: Beyond predictions, this module displays seasonal averages (monsoon, winter, pre-monsoon, and post-monsoon), offers comparisons with neighboring districts, and identifies rainfall anomalies.

Error Handling: Robust error handling was implemented to ensure smooth user experience. In cases where district data is unavailable or a prediction error occurs, an error message guides the user and provides diagnostic information.

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Accuracy

Prediction Accuracy: The model achieved a satisfactory level of accuracy, with Mean Absolute Error (MAE) and Mean Squared Error (MSE) metrics indicating reliable performance. The accuracy was especially strong in regions with consistent annual rainfall.

Insights on Accuracy: The model performed well for districts with stable rainfall patterns, capturing the predictable seasonal shifts. However, accuracy diminished in areas with high variability in rainfall, particularly in regions with erratic monsoon patterns, which the linear regression model may struggle to predict precisely.

Table 4.1.1 Classification Report

	Precision	Recall	F1 - score
Accuracy	-	-	0.91
Macro Average	0.90	0.98	0.92
Weighted Average	0.93	0.98	0.94

4.2 Challenges and Limitations

Data Availability: The historical dataset did not consistently cover all districts, limiting the model's ability to make predictions in some areas. This led to challenges in aligning geospatial data with rainfall records, as certain districts were underrepresented in the dataset.

Geospatial Alignment: Normalizing district boundaries between the GeoJSON file and the rainfall dataset required extensive manual adjustments, as district names were inconsistent between datasets. This process was necessary to ensure that map features aligned correctly with the rainfall data.

Model Limitations: While linear regression provided baseline predictions, it may lack the sophistication to capture complex, non-linear patterns in rainfall trends, which could impact its effectiveness in predicting rainfall during irregular monsoon seasons or other erratic weather events.

Interactive Map Limitations: Folium’s static rendering limits real-time updates, which can reduce the fluidity of interactions on the map.

CHAPTER 5

CONCLUSION AND FUTURE ENHANCEMENT

5.1 Conclusion

The Heavy Impact Rain Predictor is, in the words of its developers, one very advanced and robust tool used for making more accurate and timely heavy rainfall predictions to enable people and organizations to take proactive measures against potential weather-related risks. The system integrates meteorological data like temperature, humidity, atmospheric pressure, and wind speed in a model that uses machine learning, specifically logistic regression, to model and predict heavy rainfall. The predictor plays an essential role in mitigating the critical problems of flooding, destruction of infrastructures, and disruption of activities by providing an early warning system.

Capturing all of the hallmarks of a good forecasting tool, it encompasses real-time data intake, efficient preprocessing for good data quality, state-of-the-art model development, and user-friendly visualization that does present clear and accessible presentation of the predictions. Therefore, have is the ability to empower communities and emergency services and infrastructure managers with the tools that will be necessary for preemptive actions. Whether it is flood warnings, securing infrastructure or informing the public, it is this capacity to forecast ahead of time the onset of heavy rainfall that would prevent better resource management and increase effectiveness in disaster preparedness while ultimately reducing the loss of both men and money.

There is immense scope for further expansion of the project at present. Future enhancement possibilities will include inclusion of more sophisticated machine learning models, input of satellite and radar data, and functionalities of expanding to other extreme weather events such as predicting others besides this one, including storms and tornadoes. At a very general level, the Heavy Impact Rain Predictor represents an integral portion of mitigation against extreme climatic impacts and extremes weather events as a most useful resilience and preparedness tool in an increasingly hostile environment to meet uncertainty in uncertain terms.

5.2 Future Enhancement

More complex models of the machine learning family, such as Random Forest or Gradient Boosting, or even Deep Learning techniques, can be applied to further improve the Heavy

Impact Rain Predictor. These models will draw closer toward much better and more accurate identifications of patterns hidden in the data than those that simpler models like logistic regression would miss. These models were probably more susceptible to the variable relationships of meteorological variables and, therefore, to more accurate predictions-in many cases, especially about scenarios involving very unstable weather conditions.

More so, real-time addition of satellite imagery and radar data would give more detailed dynamic views of the atmosphere for greater responsiveness of the system toward sudden weather changes. The movement of storms and intensity of rainfall that can be realized through the satellite and radar data constitute real-time snapshots of the weather systems, which would let the model predict in more accurate detail and up to date on the event of sudden extreme rainfall events.

Prediction can be localized to areas or microclimates for further effectiveness of the system. The various weather patterns of regions depend on geographical aspects like topography, altitude, and distance to freshwater bodies, which influence local rainfall. The customizer, in relation to these geographical factors, will be able to make a more accurate and region-appropriate forecast that would be more meaningful and useful to customers in such regions.

More importantly, multilingual support can enable push notifications so that the Heavy Impact Rain Prediction reaches further audiences. Predications in other languages ensure the systems can serve diverse populations and truly be more user-friendly for those in any region or country. Finally, with SMS, email, or mobile app notifications included, alerts provided by the system would reach users in the right time frame so that they could be informed in real time about heavy rainfall; thus, everywhere, at any time, users would get to know important updates about the system. Such updates would make the system powerful and friendly and accessible on a global scale.

REFERENCES

[1] Hernández, E.; Sanchez-Anguix, V.; Julian, V.; Palanca, J.; Duque, N. Rainfall prediction: A deep learning approach. In International Conference on Hybrid Artificial Intelligence Systems; Springer: Cham, Switzerland, 2016; pp. 151–162.

[2] Goswami, B.N. The challenge of weather prediction. Resonance **1996**, 1, 8–17.

[3] Nayak, D.R.; Mahapatra, A.; Mishra, P. A survey on rainfall prediction using artificial neural network. Int. J. Comput. Appl. **2013**, 72, 16.

[4] Kashiwao, T.; Nakayama, K.; Ando, S.; Ikeda, K.; Lee, M.; Bahadori, A. A neural network-based local rainfall prediction system using meteorological data on the internet: A case study using data from the Japan meteorological agency. Appl. Soft Comput. **2017**, 56, 317–330.

[5] Mislan, H.; Hardwinarto, S.; Sumaryono, M.A. Rainfall monthly prediction based on artificial neural network: A case study in Tenggarong Station, East Kalimantan, Indonesia. Procedia Comput. Sci. **2015**, 59, 142–151.

[6] Muka, Z.; Maraj, E.; Kuka, S. Rainfall prediction using fuzzy logic. Int. J. Innov. Sci. Eng. Technol. **2017**, 4, 1–5.

APPENDIX

CODE

```
import streamlit as st
import pandas as pd
import joblib
import folium
from streamlit_folium import st_folium
import geopandas as gpd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler

# Load model and data
model = joblib.load('rainfall_model.pkl')
rainfall_data = pd.read_csv('data/rainfall_data.csv')
geo_data = gpd.read_file('district_map.geojson')  # Load GeoJSON
file for district map

# Normalize names for consistency
def normalize_name(name):
    return name.strip().lower().replace(" ", "")

# Apply normalization to rainfall data for easier matching
rainfall_data['STATE_UT_NAME'] =
rainfall_data['STATE_UT_NAME'].apply(normalize_name)
rainfall_data['DISTRICT'] =
rainfall_data['DISTRICT'].apply(normalize_name)
geo_data['st_nm'] = geo_data['st_nm'].apply(normalize_name)
geo_data['district'] = geo_data['district'].apply(normalize_name)

# Predict rainfall for selected district
def predict_rainfall(district_data):
    # Extract the required columns from the Series, keeping them in a
DataFrame format with correct feature names
    input_features = pd.DataFrame([district_data[['JAN', 'FEB', 'MAR',
'APR', 'MAY', 'JUN', 'JUL', 'AUG', 'SEP', 'OCT', 'NOV',
'DEC']].values],
                                  columns=['JAN', 'FEB', 'MAR', 'APR', 'MAY',
'JUN', 'JUL', 'AUG', 'SEP', 'OCT', 'NOV', 'DEC'])

    predicted_rainfall = model.predict(input_features)
    return predicted_rainfall[0]

# Seasonal Forecast calculation
def calculate_seasonal_forecast(district_data):
    # Assuming seasonal averages are provided directly, you can also
calculate these from the monthly data
    monsoon_avg = district_data[['JUN', 'JUL', 'AUG',
'SEP']].mean()  # mean for these months
    winter_avg = district_data[['DEC', 'JAN', 'FEB']].mean()  # mean
```

```
for these months
    pre_monsoon_avg = district_data[['MAR', 'APR',
'MAY']].mean()  # mean for these months
    post_monsoon_avg = district_data[['OCT']].mean()  # only one
month

    return monsoon_avg, winter_avg, pre_monsoon_avg,
post_monsoon_avg

# Anomaly Detection (simple threshold-based anomaly)
def detect_anomalies(district_data):
    anomalies = {}
    # Compare each month to the overall mean and std of that month
    for month in ['JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL',
'AUG', 'SEP', 'OCT', 'NOV', 'DEC']:
        # Directly get the value for that district for the specific month
        monthly_data = district_data[month]  # This is the value for the
district, no need for .iloc[0]
        month_mean = rainfall_data[month].mean()  # Calculate mean
for the entire dataset
        month_std = rainfall_data[month].std()  # Calculate std for the
entire dataset

        # Simple anomaly detection: if the value deviates more than 2
standard deviations, it's an anomaly
        if monthly_data > month_mean + 2 * month_std:
            anomalies[month] = 'Anomaly Detected'

    return anomalies


# Comparison with neighboring districts (Assume neighboring
districts are identified in the geo_data)
def compare_with_neigbors(district_data, state_name, district_name):
    # Get neighbors from geo_data or predefined list (assuming
geo_data contains this info)
    neighbors = geo_data[geo_data['st_nm'] == state_name]  # Get
districts from the same state
    neighbor_data =
rainfall_data[rainfall_data['DISTRICT'].isin(neighbors['district'])]  #
Get data for neighboring districts

    # Get comparison (mean rainfall for neighbors)
    neighbor_mean_rainfall = neighbor_data[['JAN', 'FEB', 'MAR',
'APR', 'MAY', 'JUN', 'JUL', 'AUG', 'SEP', 'OCT', 'NOV',
'DEC']].mean(axis=1)
    return neighbor_mean_rainfall.mean()  # Return the average rainfall
of neighboring districts

# Streamlit app
st.title("Rainfall Prediction for Indian Districts")

# Display map with clickable districts
m = folium.Map(location=[20.5937, 78.9629], zoom_start=5)
```

```
# Create a color map for rainfall prediction (for color-coding)
rainfall_data['predicted_rainfall'] = rainfall_data.apply(lambda row:
predict_rainfall(row), axis=1)

for _, row in geo_data.iterrows():
    district_name = row['district']
    state_name = row['st_nm']

    # Define popup function
    def popup_html(district, state):
        # Filter the rainfall data for the clicked district and state
        district_data = rainfall_data[(rainfall_data['STATE_UT_NAME']
== state) & (rainfall_data['DISTRICT'] == district)]

        # Check if district data is found
        if not district_data.empty:
            latest_data = district_data.iloc[-1]  # Get latest record
            predicted_rainfall = predict_rainfall(latest_data)
            # Seasonal forecast
            monsoon_avg, winter_avg, pre_monsoon_avg,
post_monsoon_avg = calculate_seasonal_forecast(latest_data)
            # Anomalies
            anomalies = detect_anomalies(latest_data)
            # Comparison with neighbors
            neighbor_rainfall = compare_with_neigbors(latest_data, state,
district)

            popup_content = f"<strong>{district}, {state}</strong><br>"
            popup_content += f"Predicted Annual Rainfall:
{predicted_rainfall:.2f} mm<br>"
            popup_content += f"Monsoon Avg: {monsoon_avg:.2f}
mm<br>"
            popup_content += f"Winter Avg: {winter_avg:.2f} mm<br>"
            popup_content += f"Pre-Monsoon Avg:
{pre_monsoon_avg:.2f} mm<br>"
            popup_content += f"Post-Monsoon Avg:
{post_monsoon_avg:.2f} mm<br>"

            if anomalies:
                popup_content += "<strong>Anomalies detected
in:</strong><br>"
                for month, status in anomalies.items():
                    popup_content += f"{month}: {status}<br>"

            popup_content += f"Comparison with Neighbors:
{neighbor_rainfall:.2f} mm (avg)<br>"

            return popup_content
        else:
            # Display message with normalized names for troubleshooting
            return f"No data available for {district}, {state}"

    # Add polygon for each district with a popup
```

```
    district_boundary = folium.GeoJson(
        row['geometry'],
        name=district_name,
        tooltip=district_name,
        popup=folium.Popup(popup_html(district_name, state_name),
max_width=300),
        style_function=lambda x: {
            'fillColor': 'green' if not
rainfall_data.loc[rainfall_data['DISTRICT'] == district_name,
'predicted_rainfall'].empty and
rainfall_data.loc[rainfall_data['DISTRICT'] == district_name,
'predicted_rainfall'].values[0] > 1000 else 'red',
            'color': 'black',
            'weight': 1
        }
    )
    district_boundary.add_to(m)

# Display map in Streamlit
st_data = st_folium(m, width=700, height=500)
```

SCREEN SHOTS OF MODULES

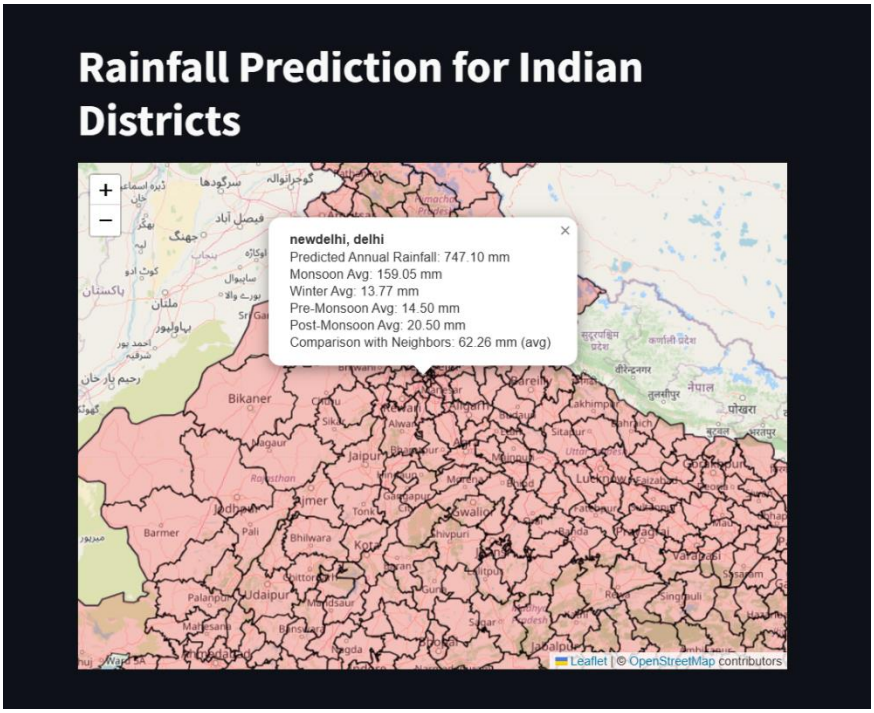

Fig 7.1 Rainfall prediction for New Delhi

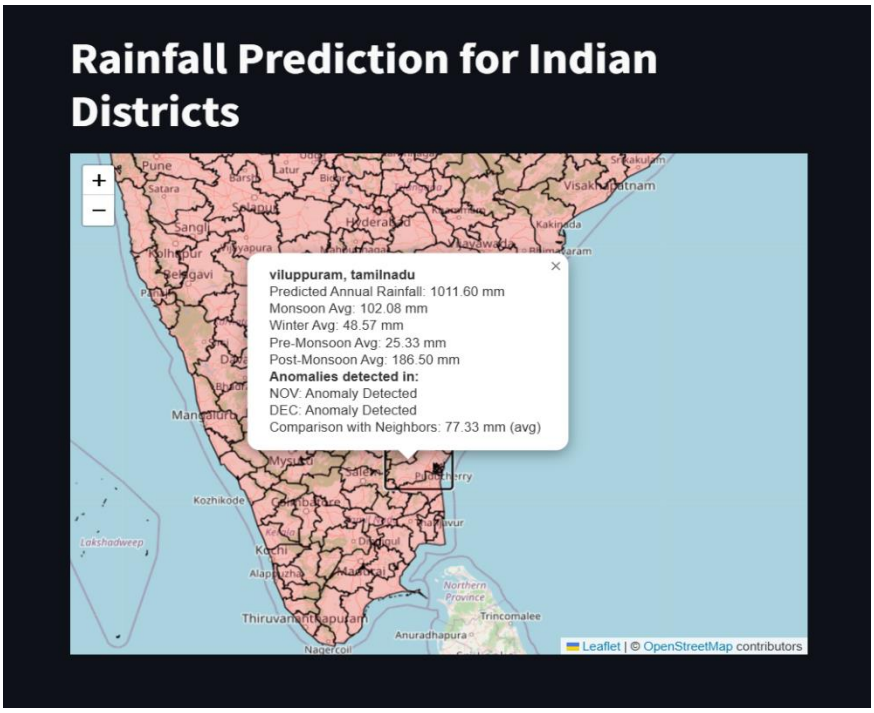

Fig 7.2 Rainfall prediction for Viluppuram