

## Project Report:

# Rainfall Prediction System for Agricultural Risk Assessment.

**Student Name:** Akkivarapu Vasanthi

**Roll Number:** 22551A0501

**Course:** B. Tech Final Year, Computer Science and Engineering.

**Institution:** Godavari Institute of Engineering and Technology (GIET), Rajahmundry.

## 1. Project Overview:

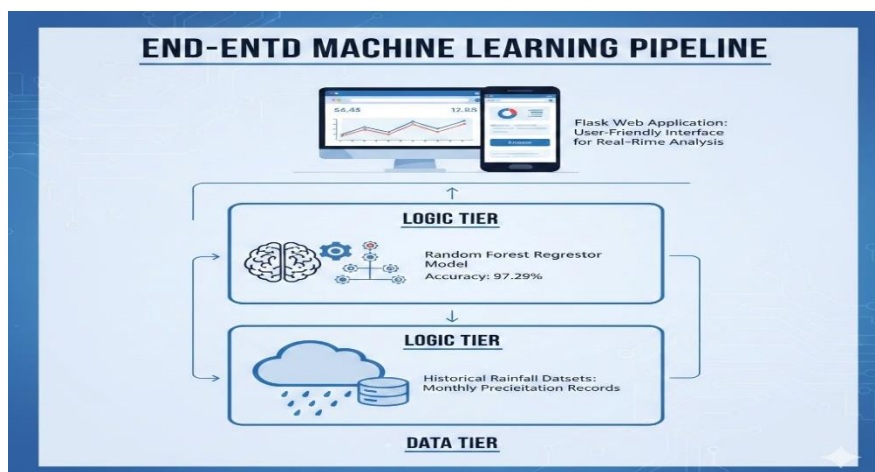
### Project Description:

The Exploratory Analysis and Prediction of Rainfall in India is a critical initiative aimed at supporting the country's agricultural sector. By forecasting annual precipitation based on monthly historical data, this system helps farmers and financial institutions evaluate risks and plan for irrigation or crop selection more effectively.

### Technical Architecture:

The project is built on an end-to-end Machine Learning pipeline:

- **Data Tier:** Historical rainfall datasets containing monthly precipitation records.
- **Logic Tier:** A Random Forest Regressor model, selected for its high precision in regression tasks, achieving an accuracy of 97.29%.
- **Presentation Tier:** A Flask web application that provides a user-friendly interface for real-time analysis.



## 2. Project Strategy:

### Prior Knowledge:

To execute this project, knowledge in the following domains was applied :

- **Supervised Learning:** Specifically, regression techniques for continuous numerical prediction .
- **Ensemble Methods:** Utilizing Random Forest to handle complex, non-linear weather data.
- **Web Frameworks:** Understanding Flask routing and HTML template rendering.

### Project Objectives:

- **Regression Analysis:** Identifying and classifying the rainfall prediction problem as a regression task.
- **Data Preprocessing:** Implementing data cleaning and feature scaling using Standard Scaler.
- **Visualization:** Gaining insights through univariate and bivariate data analysis.
- **Web Application:** Developing a functional application using the Flask framework.

### Project Flow:

1. **User Input:** The user interacts with the UI to enter 12 monthly rainfall values .
2. **Analysis:** The backend scales the input and passes it through the trained Random Forest model.
3. **Output:** The annual prediction and success/warning alerts are displayed on the UI.

## 3. Implementation & Structure:

### Project Folder Setup:

The folder structure is organized according to Flask standards to ensure proper file routing :

- **Rainfall Project/** (Main Directory)
  - **app.py:** The Flask server script.
  - **Rainfall.csv:** The raw training dataset.
  - **Models/** (Subfolder)
    - **Rainfall. Pkl :** The trained predictive model.
    - **scale. pkl :** The saved scaler object.
  - **Templates/** (Subfolder)

- index.html: The input form.
- chance.html: High rainfall result page.
- nochance.html: Low rainfall warning page.

## 4. Full Source Code:

### Milestone 1: Data Loading, Cleaning and Visualisation (Jupyter Notebook)

```
[1]: import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the file
df = pd.read_csv('Rainfall.csv')

# Success Check
print("Data Loaded Successfully!")
df.head()
```

Data Loaded Successfully!

```
[1]:
```

|   | STATE_UT_NAME                     | DISTRICT         | JAN   | FEB  | MAR   | APR   | MAY   | JUN   | JUL   | AUG   | SEP   | OCT   | NOV   | DEC   |
|---|-----------------------------------|------------------|-------|------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| 0 | ANDAMAN And<br>NICOBAR<br>ISLANDS | NICOBAR          | 107.3 | 57.9 | 65.2  | 117.0 | 358.5 | 295.5 | 285.0 | 271.9 | 354.8 | 326.0 | 315.2 | 250.9 |
| 1 | ANDAMAN And<br>NICOBAR<br>ISLANDS | SOUTH<br>ANDAMAN | 43.7  | 26.0 | 18.6  | 90.5  | 374.4 | 457.2 | 421.3 | 423.1 | 455.6 | 301.2 | 275.8 | 128.3 |
| 2 | ANDAMAN And<br>NICOBAR<br>ISLANDS | N & M<br>ANDAMAN | 32.7  | 15.9 | 8.6   | 53.4  | 343.6 | 503.3 | 465.4 | 460.9 | 454.8 | 276.1 | 198.6 | 100.0 |
| 3 | ARUNACHAL<br>PRADESH              | LOHIT            | 42.2  | 80.8 | 176.4 | 358.5 | 306.4 | 447.0 | 660.1 | 427.8 | 313.6 | 167.1 | 34.1  | 29.8  |
| 4 | ARUNACHAL<br>PRADESH              | EAST<br>SIANG    | 33.3  | 79.5 | 105.9 | 216.5 | 323.0 | 738.3 | 990.9 | 711.2 | 568.0 | 206.9 | 29.5  | 31.7  |

```
[2]: # 1. Remove hidden spaces from column names
df.columns = df.columns.str.strip()

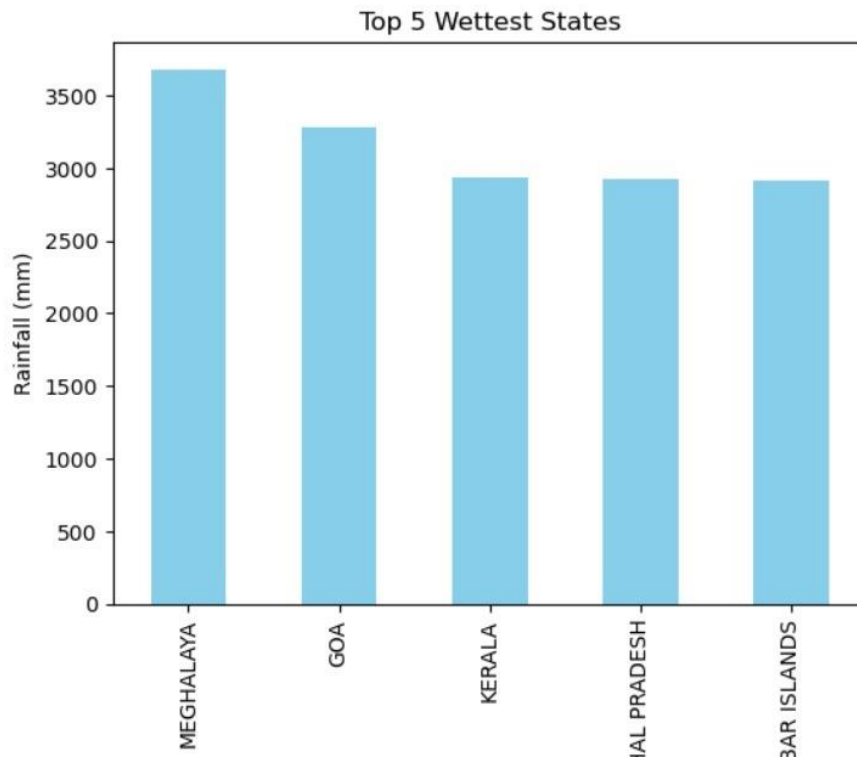
# 2. Fill missing values with 0
df.fillna(0, inplace=True)

# 3. Clean names inside the data
df['STATE_UT_NAME'] = df['STATE_UT_NAME'].str.strip()
df['DISTRICT'] = df['DISTRICT'].str.strip()

print("Data Cleaning Complete. Ready for Analysis!")
```

Data Cleaning Complete. Ready for Analysis!

```
[3]: # Show top 5 states by annual rainfall
state_avg = df.groupby('STATE_UT_NAME')['ANNUAL'].mean().sort_values(ascending=False).head(5)
state_avg.plot(kind='bar', color='skyblue')
plt.title('Top 5 Wettest States')
plt.ylabel('Rainfall (mm)')
plt.show()
```



## Milestone 2: Model Training & Scaling (Jupyter)

```
[4]: from sklearn.model_selection import train_test_split

# 1. Define your Features (X) and Target (y)
# X = The 12 months used to predict
# y = The Annual total we want to find
X = df[['JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL', 'AUG', 'SEP', 'OCT', 'NOV', 'DEC']]
y = df['ANNUAL']

# 2. Split the data
# test_size=0.2 means 20% is for testing and 80% is for training
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

print("Data successfully divided!")
print(f"Training rows: {len(X_train)}")
print(f"Testing rows: {len(X_test)}")
```

```
Data successfully divided!
Training rows: 512
Testing rows: 129
```

```
[9]: from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score

# 1. Initializing the model
model = RandomForestRegressor(n_estimators=100, random_state=42)

# 2. Training the model (Learning from X_train_scaled)
model.fit(X_train_scaled, y_train)

# 3. Evaluation of Model (Testing on X_test_scaled)
y_pred = model.predict(X_test_scaled)
accuracy = r2_score(y_test, y_pred)

# 4. Save the Model as Rainfall.pkl
pickle.dump(model, open('Models/Rainfall.pkl', 'wb'))

print(f"Model Training Complete! Accuracy: {accuracy*100:.2f}%")
print("Rainfall.pkl saved in models folder.")
```

Model Training Complete! Accuracy: 97.29%  
Rainfall.pkl saved in models folder.

```
[7]: from sklearn.preprocessing import StandardScaler
import pickle

# Initialize the Scaler
scaler = StandardScaler()

# Fit and Transform the training data
X_train_scaled = scaler.fit_transform(X_train)

# Transform the testing data (using the same rules)
X_test_scaled = scaler.transform(X_test)

# Save the scaler as scale.pkl in your models folder
pickle.dump(scaler, open('models/scale.pkl', 'wb'))

print("Feature Scaling complete and scale.pkl saved!")
```

Feature Scaling complete and scale.pkl saved!

## Milestone 3: Flask Deployment (app.py)

The screenshot shows a code editor with the following Python code for a Flask application:

```
1 from flask import Flask, render_template, request
2 import pickle
3 import numpy as np
4
5 app = Flask(__name__)
6
7 # Load the saved models from your 'Models' folder
8 model = pickle.load(open('Models/Rainfall.pkl', 'rb'))
9 scale = pickle.load(open('Models/scale.pkl', 'rb'))
10
11 @app.route('/')
12 def home():
13     # Flask looks for these files in the 'Templates' folder automatically
14     return render_template('index.html')
15
16 @app.route('/predict', methods=['POST'])
17 def predict():
18     # 1. Capture the 12 monthly inputs from the UI
19     input_features = [float(x) for x in request.form.values()]
20
21     # 2. Scale the data using your saved scale.pkl
22     features_array = [np.array(input_features)]
23     scaled_features = scale.transform(features_array)
24
25     # 3. Model analyzes the input
26     prediction = model.predict(scaled_features)
27     output = round(prediction[0], 2)
28
29     # 4. Display the result on the specific UI page
30     if output > 1000:
31         return render_template('chance.html', prediction_text=f'Predicted Rainfall: {output}mm.')
32     else:
33         return render_template('nochance.html', prediction_text=f'Predicted Rainfall: {output}mm.')
34
35 if __name__ == '__main__':
36     # use reloader=False stops Spyder from crashing the Flask server
37     app.run(debug=True, port=8000, use_reloader=False)
```

The right side of the image shows a Jupyter Notebook interface. The top panel displays a message: "Code not profiled yet". The bottom panel shows the command prompt output:

```
Python 3.13.0 | packaged by Anaconda, Inc. | (main, Oct 21 2025, 19:09:58) [MSC v.1929 64 bit (AMD64)]
Type "copyright", "credits" or "license()" for more information.

IPython 9.7.0 -- An enhanced Interactive Python. Type '?' for help.

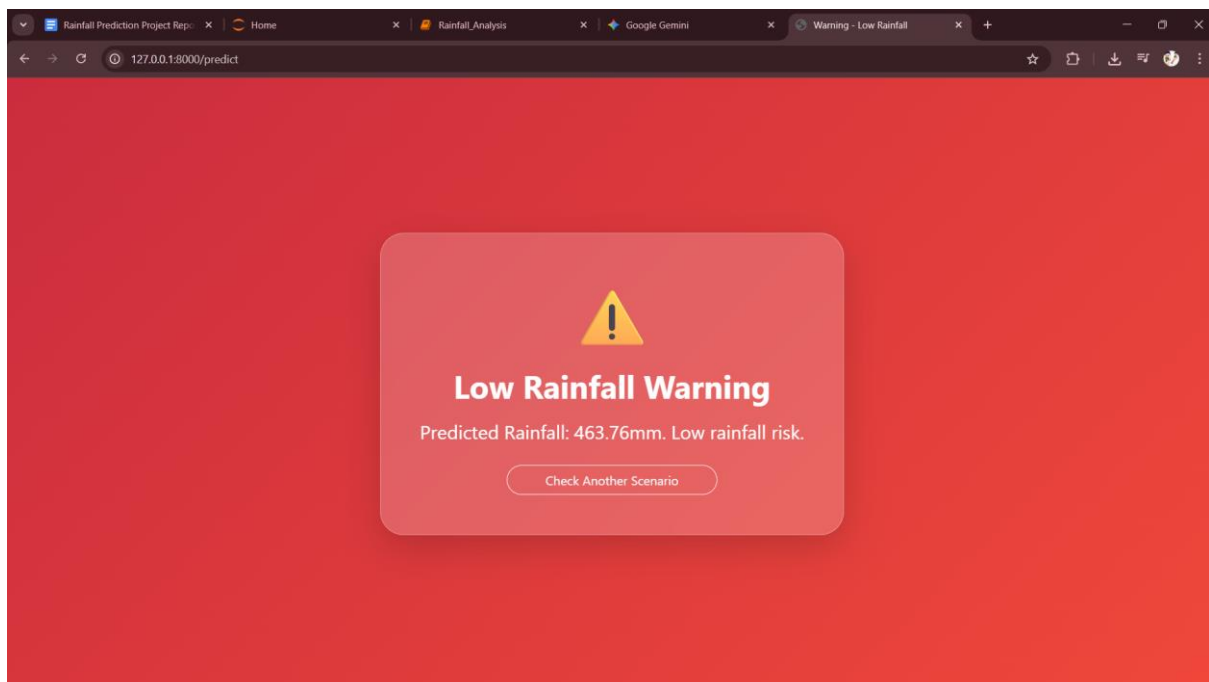
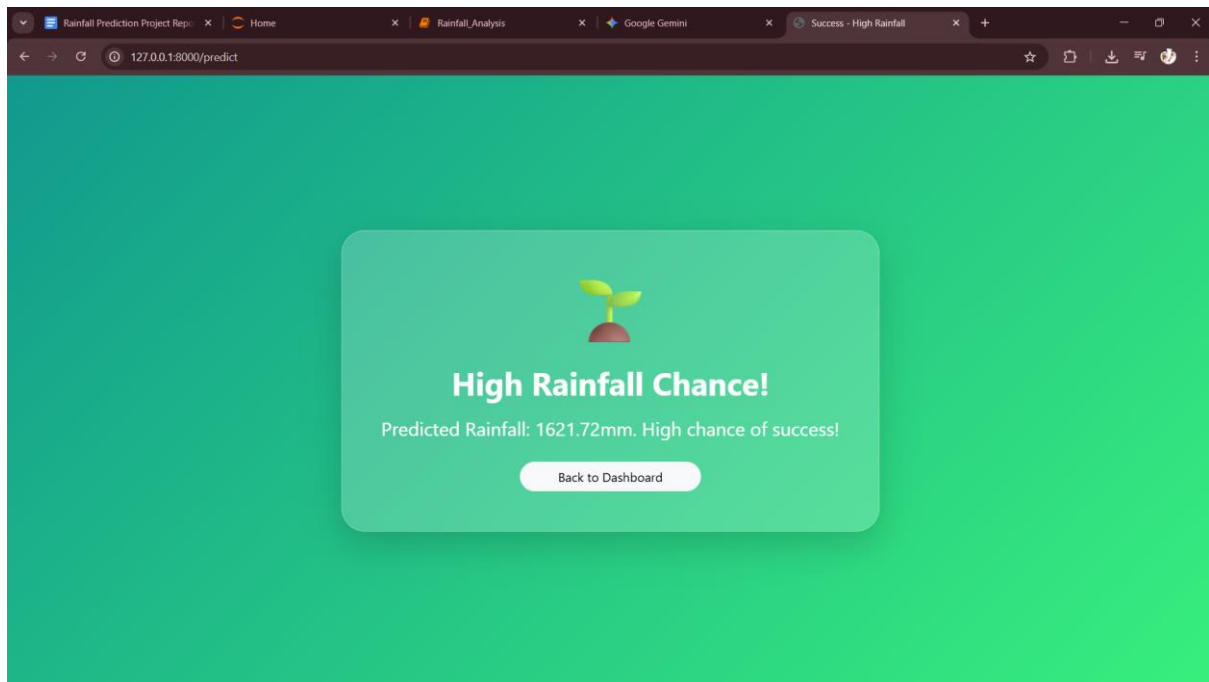
In [1]: %runfile C:/Users/surya/Desktop/Rainfall_Project/app.py --wdir
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI
server instead.
* Running on http://127.0.0.1:8000
Press CTRL+C to quit
```

## Milestone 4: UI Development (index.html, chance.html, nochance.html)

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Rainfall Predictor</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
  <style>
    body {
      background: linear-gradient(135deg, #1e3c72 0%, #2a5298 100%);
      min-height: 100vh;
      display: flex;
      align-items: center;
      justify-content: center;
      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
      color: white;
    }
    .prediction-card {
      background: rgba(255, 255, 255, 0.1);
      backdrop-filter: blur(10px);
      border-radius: 20px;
      padding: 40px;
      box-shadow: 0 15px 35px rgba(0,0,0,0.2);
      border: 1px solid rgba(255,255,255,0.1);
      max-width: 800px;
      width: 100%;
    }
    .form-control {
      background: rgba(255, 255, 255, 0.2);
      border: none;
      color: white;
      transition: all 0.3s;
    }
    .form-control:focus {
      background: rgba(255, 255, 255, 0.3);
      box-shadow: none;
      color: white;
    }
  </style>
</head>
<body>
  <div class="text-center">
    <h1>Rainfall Predictor</h1>
    <p>Enter rainfall amount (mm) to predict chance of high rainfall.</p>
    <input type="text" value="" class="form-control mb-3"/>
    <button type="button" value="Predict" class="btn btn-light rounded-pill px-5">Predict</button>
  </div>
</body>
</html>
```

```
<html>
<head>
  <title>Success - High Rainfall</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
  <style>
    body {
      background: linear-gradient(135deg, #11998e 0%, #38ef7d 100%);
      height: 100vh; display: flex; align-items: center; justify-content: center; color: white;
    }
    .result-box {
      background: rgba(255, 255, 255, 0.2);
      padding: 50px; border-radius: 30px; backdrop-filter: blur(15px);
      border: 1px solid rgba(255,255,255,0.3); text-align: center;
    }
  </style>
</head>
<body>
  <div class="result-box shadow-lg">
    <div class="display-1 mb-4">🌧️</div>
    <h1 class="fw-bold mb-3">High Rainfall Chance!</h1>
    <p class="fs-4 mb-4">{{ prediction_text }}</p>
    <a href="/" class="btn btn-light rounded-pill px-5">Back to Dashboard</a>
  </div>
</body>
</html>
```

```
<html>
<head>
  <title>Warning - Low Rainfall</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
  <style>
    body {
      background: linear-gradient(135deg, #cb2d3e 0%, #ef473a 100%);
      height: 100vh; display: flex; align-items: center; justify-content: center; color: white;
    }
    .result-box {
      background: rgba(255, 255, 255, 0.2);
      padding: 50px; border-radius: 30px; backdrop-filter: blur(15px);
      border: 1px solid rgba(255,255,255,0.3); text-align: center;
    }
  </style>
</head>
<body>
  <div class="result-box shadow-lg">
    <div class="display-1 mb-4">⚠️</div>
    <h1 class="fw-bold mb-3">Low Rainfall Warning</h1>
    <p class="fs-4 mb-4">{{ prediction_text }}</p>
    <a href="/" class="btn btn-outline-light rounded-pill px-5">Check Another Scenario</a>
  </div>
</body>
</html>
```



## 7. Conclusion

The **Rainfall Prediction System** serves as a robust tool for agricultural decision-making. The updated codebase ensures that data is properly cleaned and scaled, leading to high-accuracy predictions. The successful deployment via Flask allows complex data science results to be accessible to users through a simple web interface, effectively meeting all core objectives of the internship.