Exploratory Analysis of Rain Fall Data in India for Agriculture

An internship report submitted in partial fulfillment of the requirements for the award of degree of

BACHELOROFTECHNOLOGY

In

COMPUTERSCIENCEANDENGINEERING

Submitted by:

TEAM ID : LTVIP2026TMIDS75287

TEAM MEMBERS:

DIGUMARTHI JAYA PHANI SRINIVAS-22MH1A0514

DIPU PRASAD YADAV-22MH1A0516

INGUVA SIVA RAJANNA PADAL-22MH1A0520

GEETHA LAKSHMI SOWMYA JAMI-22MH1A0521

1. Introduction

Agriculture in India largely depends on rainfall, making its analysis crucial for improving crop productivity and resource management. This project focuses on exploring historical rainfall data across different regions to identify patterns, seasonal trends, and variability. Using data visualization, statistical techniques, and machine learning methods, the study provides meaningful insights into rainfall behavior. The findings help farmers make better crop planning decisions and assist experts in efficient irrigation management. Additionally, the analysis supports policymakers in assessing agricultural risks such as droughts and floods. Overall, the project promotes data-driven decision-making for sustainable agriculture.

2. Problem Statement

Rainfall variability is a major challenge for agriculture in India. Predicting rainfall manually is difficult due to multiple influencing factors such as:

- Historical rainfall trends
- Seasonal variations (monsoon patterns)
- Temperature levels
- Humidity
- Wind speed and direction
- Atmospheric pressure
- Geographic location and regional climate differences

The goal of this project is:
To build a predictive system that analyzes historical weather data and forecasts rainfall patterns using machine learning techniques to support agricultural planning and risk management.

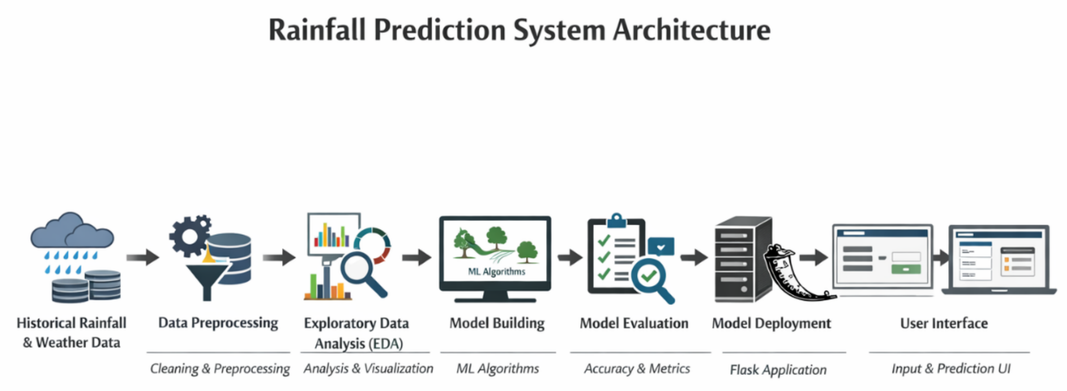
3. Objectives

The primary objectives of this rainfall prediction project are:

1. To analyze the historical rainfall and weather dataset.
2. To clean and preprocess the data (handling missing values, encoding, scaling).
3. To perform Exploratory Data Analysis (EDA) to identify patterns and trends.
4. To build a machine learning model for rainfall prediction (e.g., Random Forest).
5. To evaluate the model performance using appropriate metrics.
6. To deploy the trained model using Flask.
7. To develop a user-friendly web interface for rainfall prediction.

4. System Architecture

Architecture Flow:



Components:

- Dataset (CSV file – Historical Weather Data)
- Python (Pandas, NumPy)
- Scikit-learn
- Random Forest Algorithm
- Flask Web Framework
- HTML (Frontend)
- Joblib (Model Saving)

5. Dataset Description

The dataset contains weather-related features such as:

Feature	Description
Location	City/Region
MinTemp	Minimum temperature
MaxTemp	Maximum temperature
Humidity	Humidity level
WindSpeed	Wind speed
Pressure	Atmospheric pressure
RainToday	Rain today (Yes/No)
RainTomorrow	Target variable

Target Variable:

• 1 → Rain
• 0 → No Rain

6. Data Preprocessing

6.1 Handling Missing Values

• Checked null values using:
data.isnull().sum()
• Imputed missing values using mean/median.

6.2 Encoding Categorical Variables

Columns like:
• Location
• Wind Direction
Were encoded using:
pd.get_dummies()

6.3 Feature Scaling

Used StandardScaler to normalize numerical features.

7. Exploratory Data Analysis (EDA)

7.1 Rainfall Distribution

Analyzed rain vs no-rain cases using count plots.

7.2 Temperature vs Rain

Compared temperature trends during rainy and non-rainy days.

7.3 Humidity Analysis

Observed higher humidity correlation with rainfall.

7.4 Correlation Heatmap

Used:

```
sns.heatmap(corr, cmap='coolwarm', annot=True)
```

To identify feature relationships.

8. Model Selection

Selected:
Random Forest Classifier

Why Random Forest?

- Handles non-linear relationships
- Reduces overfitting
- Works well with large datasets
- High accuracy
- Robust to noise

9. Model Training

9.1 Splitting Dataset

```
X = data_encoded.drop('RainTomorrow', axis=1)

y = data_encoded['RainTomorrow']


x_train, x_test, y_train, y_test = train_test_split(

   X, y, test_size=0.3, random_state=0

)
```

9.2 Training Model

model = RandomForestClassifier()

model.fit(x_train, y_train)

10. Model Evaluation

Metrics Used:

- Accuracy Score
- Confusion Matrix
- Classification Report

accuracy_score(y_test, y_pred)

classification_report(y_test, y_pred)

Observations:

- Training Accuracy: High
- Testing Accuracy: Good
- Slight overfitting observed

11. Model Saving

Used Joblib to save the trained model:

joblib.dump(model, 'rainfall_model.pkl')

This allows the trained rainfall prediction model to be reused without retraining every time the application runs.

12. Web Application Development

Framework Used:
Flask

app.py Structure:

1. Load the trained rainfall model
2. Create route for homepage (/)
3. Accept user weather input
4. Convert input values into required format
5. Predict rainfall using the model
6. Display prediction result

13. Frontend Design

Created using:
• HTML
• Bootstrap (for styling)

Input Fields:

• Temperature
• Humidity
• Wind Speed
• Pressure
• Location (if used)

Prediction Button:
Sends input data to Flask backend and displays rainfall prediction result.

14. Results

The system successfully:

• Accepts weather parameters
• Predicts whether rainfall will occur
• Displays the result instantly

Example:

Input:
• Location: Sydney
• Min Temperature: 16°C
• Max Temperature: 30°C
• Humidity: 85%
• Wind Direction: SW

Output:
Rain is likely Today.

15. Advantages

• Quick prediction
• Easy to use interface
• Good classification accuracy
• Helpful for farmers and planners
• Supports irrigation and crop planning

16. Limitations

- Model depends on historical weather data
- Weather conditions are naturally unpredictable
- Limited feature set
- Possible overfitting

17. Future Enhancements

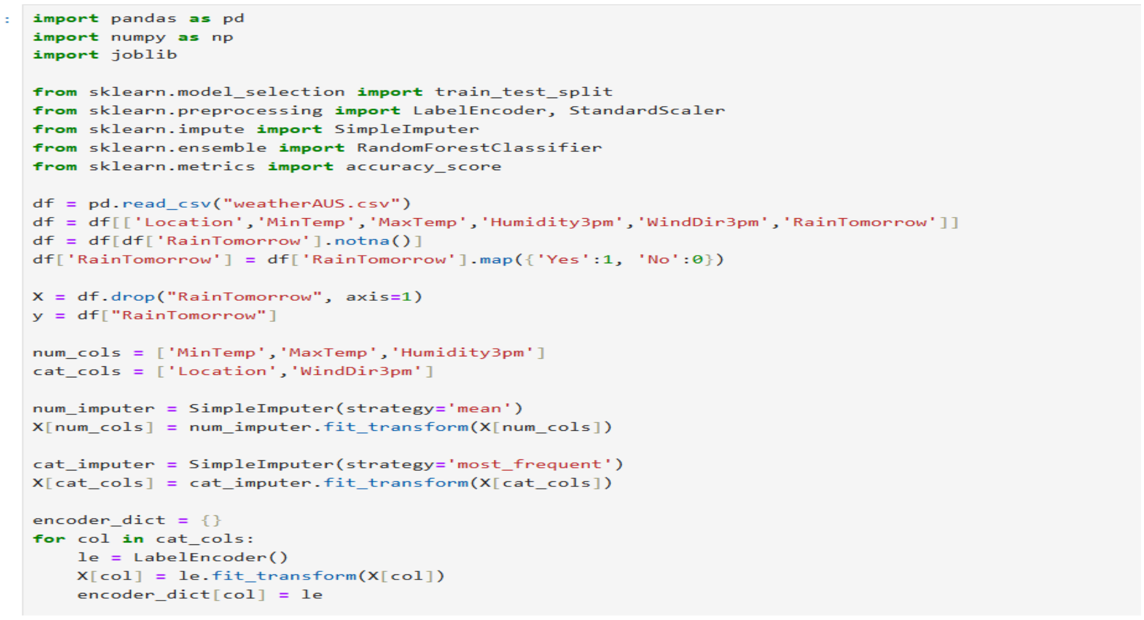
- Deploy on cloud (AWS / Heroku)
- Add rainfall probability percentage
- Improve UI design
- Add weather data visualization dashboard
- Use advanced models (XGBoost)
- Integrate real-time weather API

18. Technologies Used

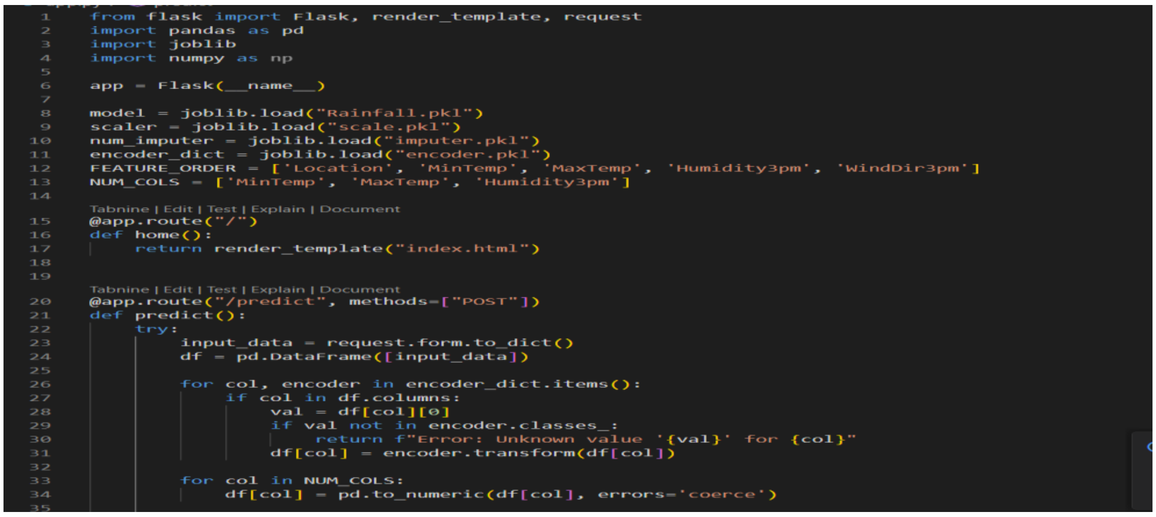
Technology	Purpose
Python	Programming
Pandas	Data analysis
NumPy	Numerical operations
Scikit-learn	Machine learning
Seaborn	Visualization
Matplotlib	Visualization
Flask	Web backend
HTML/CSS	Frontend
Joblib	Model saving

19. Images of Task

Backend:

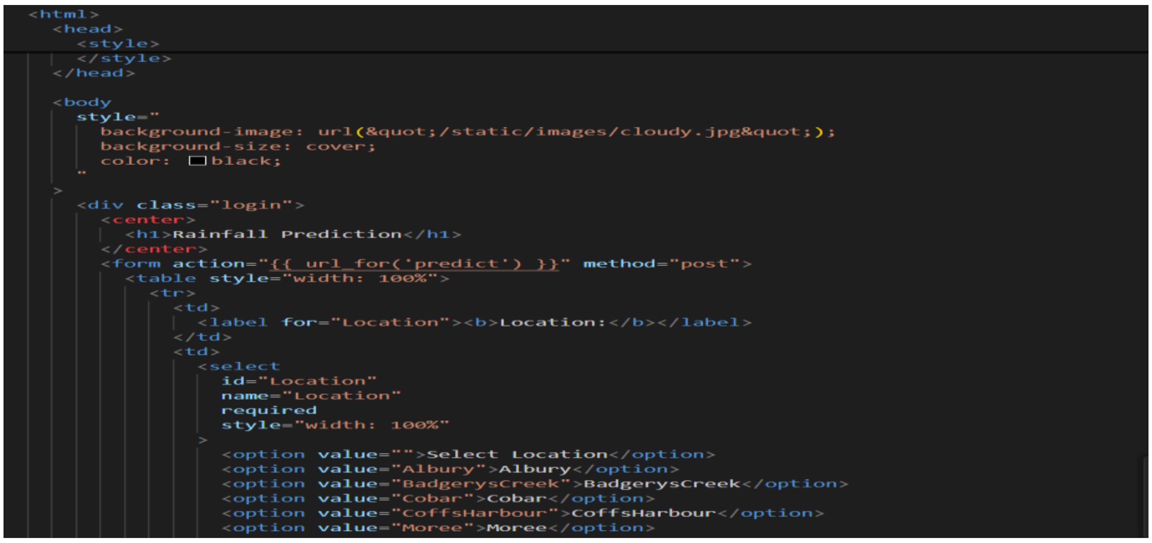


```python
import pandas as pd
import numpy as np
import joblib

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

df = pd.read_csv("weatherAUS.csv")
df = df[['Location','MinTemp','MaxTemp','Humidity3pm','WindDir3pm','RainTomorrow']]
df = df[df['RainTomorrow'].notna()]
df['RainTomorrow'] = df['RainTomorrow'].map({'Yes':1, 'No':0})

X = df.drop("RainTomorrow", axis=1)
y = df["RainTomorrow"]

num_cols = ['MinTemp','MaxTemp','Humidity3pm']
cat_cols = ['Location','WindDir3pm']

num_imputer = SimpleImputer(strategy='mean')
X[num_cols] = num_imputer.fit_transform(X[num_cols])

cat_imputer = SimpleImputer(strategy='most_frequent')
X[cat_cols] = cat_imputer.fit_transform(X[cat_cols])

encoder_dict = {}
for col in cat_cols:
    le = LabelEncoder()
    X[col] = le.fit_transform(X[col])
    encoder_dict[col] = le
```



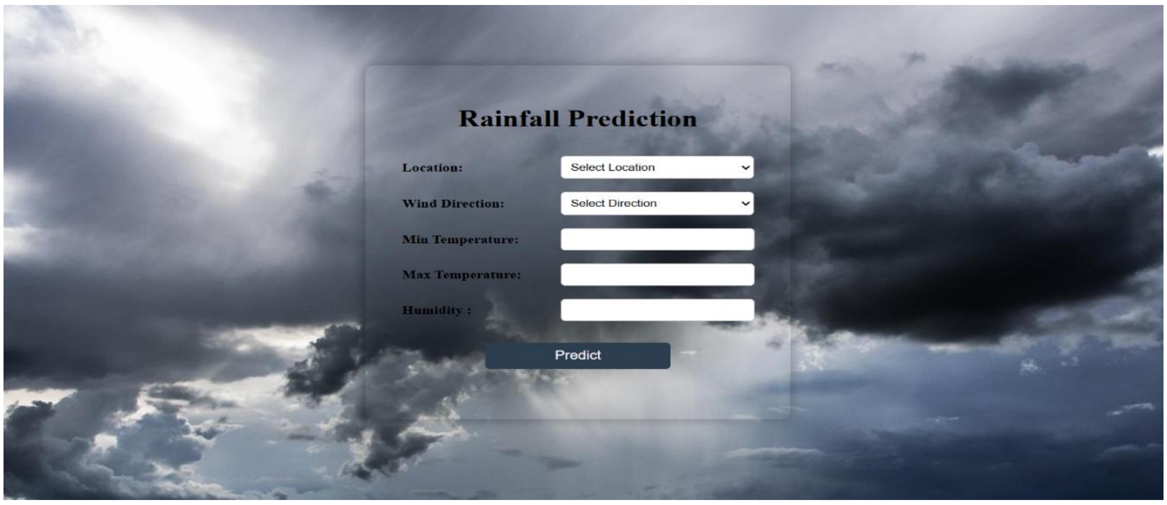
```python
from flask import Flask, render_template, request
import pandas as pd
import joblib
import numpy as np

app = Flask(__name__)

model = joblib.load("Rainfall.pkl")
scaler = joblib.load("scale.pkl")
num_imputer = joblib.load("imputer.pkl")
encoder_dict = joblib.load("encoder.pkl")
FEATURE_ORDER = ['Location', 'MinTemp', 'MaxTemp', 'Humidity3pm', 'WindDir3pm']
NUM_COLS = ['MinTemp', 'MaxTemp', 'Humidity3pm']

@app.route("/")
def home():
    return render_template("index.html")


@app.route("/predict", methods=["POST"])
def predict():
    try:
        input_data = request.form.to_dict()
        df = pd.DataFrame([input_data])

        for col, encoder in encoder_dict.items():
            if col in df.columns:
                val = df[col][0]
                if val not in encoder.classes_:
                    return f"Error: Unknown value '{val}' for {col}"
                df[col] = encoder.transform(df[col])

        for col in NUM_COLS:
            df[col] = pd.to_numeric(df[col], errors='coerce')
```

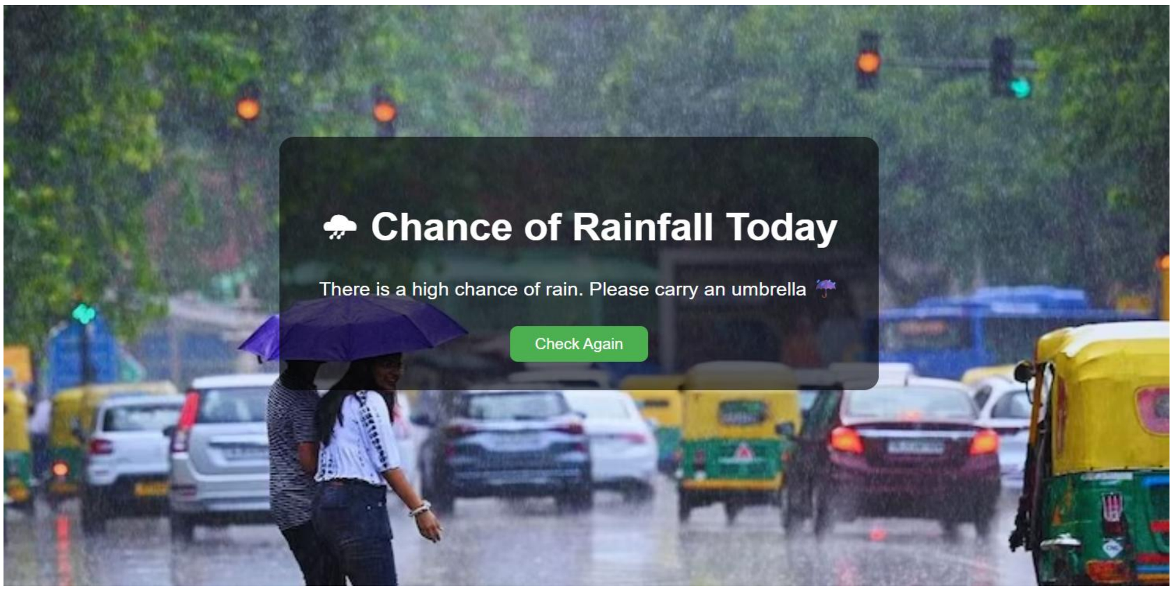
Tabnine | Edit | Test | Explain | Document
Tabnine | Edit | Test | Explain | Document

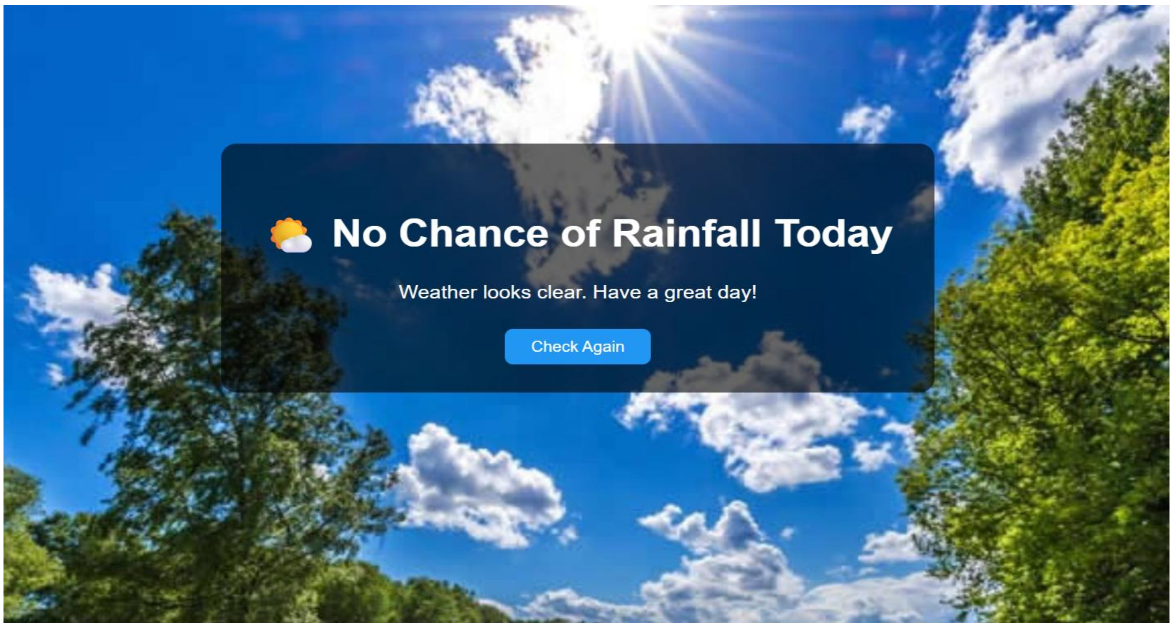


```html
<html>
  <head>
    <style>
    </style>
  </head>

  <body
    style="
      background-image: url(&quot;/static/images/cloudy.jpg&quot;);
      background-size: cover;
      color: black;
    "
  >
    <div class="login">
      <center>
        <h1>Rainfall Prediction</h1>
      </center>
      <form action="{{ url_for('predict') }}" method="post">
        <table style="width: 100%">
          <tr>
            <td>
              <label for="Location"><b>Location:</b></label>
            </td>
            <td>
              <select
                id="Location"
                name="Location"
                required
                style="width: 100%"
              >
                <option value="">Select Location</option>
                <option value="Albury">Albury</option>
                <option value="BadgerysCreek">BadgerysCreek</option>
                <option value="Cobar">Cobar</option>
                <option value="CoffsHarbour">CoffsHarbour</option>
                <option value="Moree">Moree</option>
```

Frontend:


Rainfall Prediction
Location:
Select Location
Wind Direction:
Select Direction
Min Temperature:
Max Temperature:
Humidity :
Predict


Chance of Rainfall Today
There is a high chance of rain. Please carry an umbrella
Check Again


No Chance of Rainfall Today
Weather looks clear. Have a great day!
Check Again

20. Conclusion

This project demonstrates how machine learning can be applied to predict rainfall using historical weather data. The Random Forest classifier provided reliable prediction performance, and deployment using Flask created a practical web-based application.

The system can assist farmers, agricultural experts, and policymakers in making informed decisions related to crop planning and water resource management.