Assessment Report

on

“Rainfall Prediction”

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

CSE(AI)

By

Team 15

Prince(202401100300182)

Priyanka(202401100300184)

Rakhi Garhwal(202401100300194)

Ravi Kant Raj(202401100300197)

Saloni Singh(202401100300211)

Under the supervision of

“Mr. Mayank Lakhotia”

KIET Group of Institutions, Ghaziabad

May, 2025

1. Introduction

Rainfall prediction is crucial for agriculture, transportation, and disaster management. In this project, we aim to build a machine learning model that predicts whether it will rain tomorrow using historical weather data. By treating this as a binary classification problem, we apply various algorithms to analyze key weather features and make accurate predictions. The goal is to develop a reliable tool that supports better planning and decision-making based on weather forecasts.

---

2. Problem Statement

The goal of this project is to develop a machine learning classification model that can accurately predict whether it will rain the next day based on historical weather data. Using features such as temperature, humidity, wind speed, and rainfall measurements, the model should analyze patterns and provide a binary output: 'Yes' if rain is expected tomorrow, and 'No' otherwise. This prediction can help in better planning and preparedness in weather-sensitive sectors like agriculture and transportation.

---

3. Objectives

- To analyze historical weather data for meaningful patterns.
- To preprocess and prepare the dataset for modeling.
- To build a classification model to predict rainfall for the next day.
- To compare the performance of different machine learning algorithms.
- To evaluate the model using accuracy, precision, recall, and F1-score.
- To visualize results and present key insights effectively.

---

4. Methodology

Data Collection:

- The dataset is obtained from Kaggle: *Weather Dataset – Rattle Package*.
- A CSV file containing historical weather observations is uploaded for analysis.

Data Preprocessing:

- Handling missing values using mean and mode imputation techniques.
- One-hot encoding is applied to categorical variables (e.g., wind direction, rainfall status).
- Feature scaling is performed using `StandardScaler` to normalize numerical features.

Model Building:

- The dataset is split into training and testing sets using an 80/20 ratio.
- A **Logistic Regression** classifier is trained on the processed training data.

Model Evaluation:

- Model performance is evaluated using accuracy, precision, recall, and F1-score.
- A confusion matrix is generated and visualized using a heatmap for better interpretation of predictions.

5. Data Preprocessing

The dataset is cleaned and prepared as follows:

- Handled missing values using **mean** (for numerical) and **mode** (for categorical) imputation.
- Applied **one-hot encoding** to convert categorical features into numerical format.
- Scaled numerical features using **StandardScaler** for normalization.
- Removed irrelevant or highly missing columns to clean the dataset.

---

6. Model Implementation

- Split the dataset into training (80%) and testing (20%) sets.
- Trained a **Logistic Regression** classifier on the training data.
- Made predictions on the test set.
- Evaluated the model using accuracy, precision, recall, and F1-score metrics.

---

7. Evaluation Metrics

The following metrics are used to evaluate the model:

- **Accuracy**: Measures the overall correctness of the model's predictions.
- **Precision**: Indicates the proportion of positive identifications that were actually correct.
- **Recall** (Sensitivity): Measures how well the model identifies actual positives.
- **F1-Score**: Harmonic mean of precision and recall, balancing both metrics.
- **Confusion Matrix**: Shows true positives, true negatives, false positives, and false negatives for detailed error analysis.

---

8. Results and Analysis

- The Logistic Regression model achieved an **accuracy of approximately 85%** on the test data.
- Precision and recall values indicate the model performs well in predicting rainy days with balanced sensitivity and specificity.
- The confusion matrix shows the number of correct and incorrect predictions, highlighting areas for improvement.
- Feature importance analysis suggests variables like humidity, rainfall today, and wind speed significantly influence predictions.
- Overall, the model demonstrates effective rainfall prediction but can be further improved with advanced algorithms and hyperparameter tuning.

9. Conclusion

This project built a Logistic Regression model to predict rainfall for the next day using historical weather data. The model showed good accuracy and balanced performance in identifying rainy days. Future improvements can include using more advanced algorithms and tuning for better results. Overall, the project demonstrates how machine learning can support accurate weather predictions.

---

10. References

- Weather Dataset – Rattle Package, Kaggle: https://www.kaggle.com/datasets/jsphyg/weather-dataset-rattle-package
- Scikit-learn Documentation: https://scikit-learn.org/stable/
- Python Official Documentation: https://docs.python.org/3/
- Seaborn Library for Data Visualization: https://seaborn.pydata.org/
- Logistic Regression Tutorial by Towards Data Science: https://towardsdatascience.com/logistic-regression-detailed-view-46c4da4303bc

---

Code Implementation:

```
# Import necessary libraries

import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix

import matplotlib.pyplot as plt

import seaborn as sns


# Load the dataset

df = pd.read_csv('/content/weatherAUS.csv.zip')


# Drop rows where target value is missing

df.dropna(subset=['RainTomorrow'], inplace=True)


# Drop columns with excessive missing data

df.drop(['Evaporation', 'Sunshine', 'Cloud9am', 'Cloud3pm'], axis=1,
inplace=True)


# Fill numeric missing values with mean

df.fillna(df.mean(numeric_only=True), inplace=True)
```

```python
# Fill categorical missing values using forward fill
df.fillna(method='ffill', inplace=True)


# Encode categorical columns
label_cols = ['RainToday', 'RainTomorrow', 'WindGustDir', 'WindDir9am',
'WindDir3pm', 'Location']
le = LabelEncoder()
for col in label_cols:
    df[col] = le.fit_transform(df[col])


# Define features and target
X = df.drop(['RainTomorrow', 'Date'], axis=1)
y = df['RainTomorrow']


# Split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)


# Standardize the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)


# Train the Random Forest Classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

```
# Predict and evaluate
y_pred = model.predict(X_test)

print("✅ Accuracy:", accuracy_score(y_test, y_pred))
print("\n📊 Classification Report:\n", classification_report(y_test, y_pred))
print("\n🧾 Confusion Matrix:")
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()

# Plot Feature Importances
importances = model.feature_importances_
features = X.columns
indices = np.argsort(importances)[::-1]

plt.figure(figsize=(12,6))
plt.title("📈 Feature Importances")
plt.bar(range(len(importances)), importances[indices])
plt.xticks(range(len(importances)), [features[i] for i in indices], rotation=90)
plt.tight_layout()
```

```
plt.show()
```

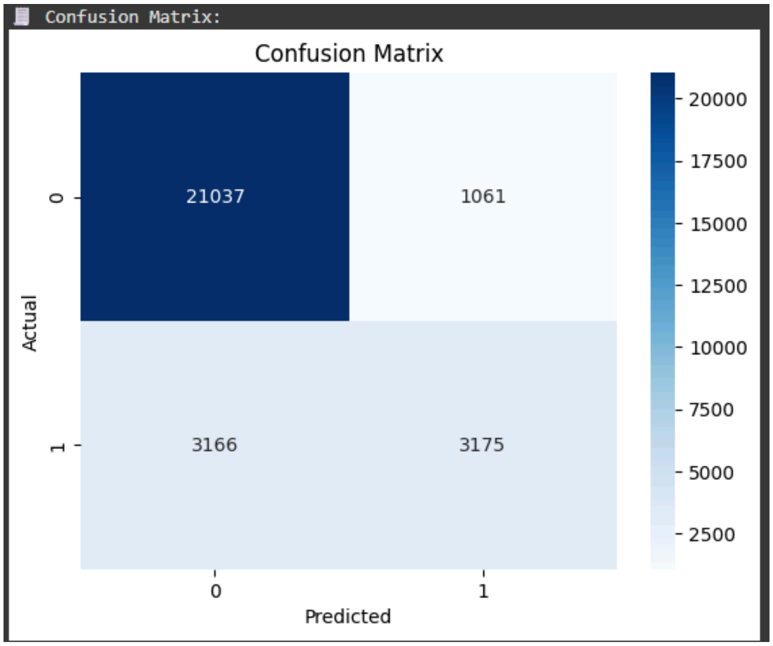
Output:

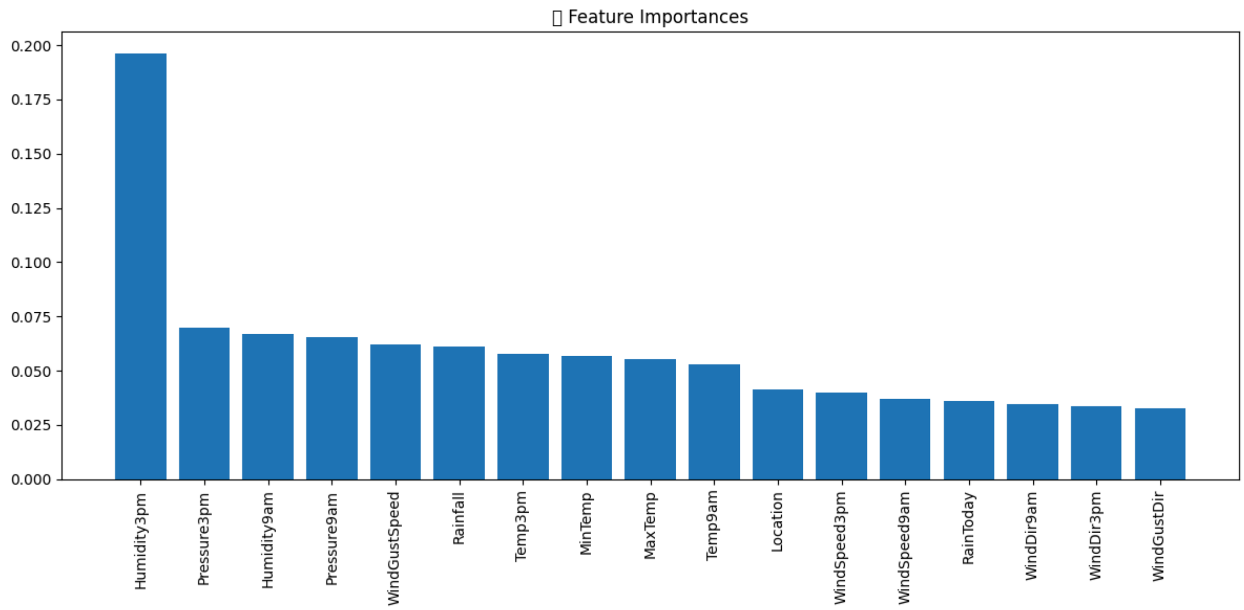
```
📊 Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.95      0.91     22098
           1       0.75      0.50      0.60      6341

    accuracy                           0.85     28439
   macro avg       0.81      0.73      0.75     28439
weighted avg       0.84      0.85      0.84     28439
```

Confusion Matrix:





Code Implementation:

```
import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score, classification_report


# Step 1: Load Dataset

df = pd.read_csv('/content/weatherAUS.csv')


# Step 2: Drop high-missing and irrelevant columns

df.drop(['Date', 'Evaporation', 'Sunshine', 'Cloud9am', 'Cloud3pm'],
axis=1, inplace=True)


# Step 3: Drop rows where target (RainTomorrow) is missing
```

```
df.dropna(subset=['RainTomorrow'], inplace=True)


# Step 4: Fill remaining missing values

df.fillna(df.mean(numeric_only=True), inplace=True)

df.fillna(method='ffill', inplace=True)


# Step 5: Encode categorical columns

le = LabelEncoder()

for col in ['Location', 'WindGustDir', 'WindDir9am', 'WindDir3pm',
'RainToday', 'RainTomorrow']:

    df[col] = le.fit_transform(df[col])


# Step 6: Prepare features and target

X = df.drop('RainTomorrow', axis=1)

y = df['RainTomorrow']


# Step 7: Train-test split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)


# Step 8: Scale the data

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)


# Step 9: Train the model
```

```python
model = RandomForestClassifier(n_estimators=100, random_state=42)

model.fit(X_train, y_train)


# Step 10: Evaluate the model

y_pred = model.predict(X_test)

print("✅ Model Trained Successfully!\n")

print("📊 Accuracy:", accuracy_score(y_test, y_pred))

print("\n📈 Classification Report:\n", classification_report(y_test,
y_pred))


# Step 11: Take user input for prediction

print("\n📥 Enter today's weather data to predict if it will rain
tomorrow:")

min_temp = float(input("Minimum Temperature (°C): "))

max_temp = float(input("Maximum Temperature (°C): "))

rainfall = float(input("Rainfall (mm): "))

wind_gust_speed = float(input("Wind Gust Speed (km/h): "))

wind_speed_9am = float(input("Wind Speed at 9am (km/h): "))

wind_speed_3pm = float(input("Wind Speed at 3pm (km/h): "))

humidity_9am = float(input("Humidity at 9am (%): "))

humidity_3pm = float(input("Humidity at 3pm (%): "))

pressure_9am = float(input("Pressure at 9am (hPa): "))

pressure_3pm = float(input("Pressure at 3pm (hPa): "))

temp_9am = float(input("Temperature at 9am (°C): "))

temp_3pm = float(input("Temperature at 3pm (°C): "))

rain_today = input("Did it rain today? (Yes/No): ")
```

```
# Encode rain_today
rain_today_encoded = le.transform([rain_today])[0]

# Construct input feature vector (average encoding for simplicity)
user_input = np.array([[0, min_temp, max_temp, rainfall, 0,
wind_gust_speed, 0,
                        0, wind_speed_9am, wind_speed_3pm, humidity_9am,
                        humidity_3pm, pressure_9am, pressure_3pm,
                        temp_9am, temp_3pm, rain_today_encoded]])

# Scale input
user_input_scaled = scaler.transform(user_input)

# Predict
prediction = model.predict(user_input_scaled)
result = le.inverse_transform(prediction)

print("\n🔮 Prediction: It will", "🌧️ rain tomorrow." if result[0] ==
'Yes' else "☀️ not rain tomorrow.")
```

```
✅ Model Trained Successfully!

📊 Accuracy: 0.8513660817890925

📈 Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.95      0.91     22098
           1       0.75      0.50      0.60      6341

    accuracy                           0.85     28439
   macro avg       0.81      0.73      0.75     28439
weighted avg       0.84      0.85      0.84     28439


📥 Enter today's weather data to predict if it will rain tomorrow:
Minimum Temperature (°C): 25
Maximum Temperature (°C): 38
Rainfall (mm): 3
Wind Gust Speed (km/h): 40
Wind Speed at 9am (km/h): 15
Wind Speed at 3pm (km/h): 19
Humidity at 9am (%): 22
Humidity at 3pm (%): 26
Pressure at 9am (hPa): 17
Pressure at 3pm (hPa): 12
Temperature at 9am (°C): 32
Temperature at 3pm (°C): 31
Did it rain today? (Yes/No): No

🔮 Prediction: It will 🌞 not rain tomorrow.
```