Assessment Report

on

"Model to predict whether it will rain tomorrow using classification algorithms and weather data."

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY

DEGREE

SESSION 2024-25

in

CSE(AI)-B

By

Ayush Gupta – 202401100300084

Ashish Yadav-202401100300076

Ayush Prasad – 202401100300086

Himanshu Singh – 202401100300125

Under the supervision of

"Shivansh Prasad"

KIET Group of Institutions, Ghaziabad

May, 2025

1. Introduction

With the increasing impact of weather on agriculture, travel, and daily planning, accurate rainfall prediction has become a vital component of meteorology. This project focuses on using **supervised machine learning** to build a model that predicts whether it will rain tomorrow. By leveraging historical weather data such as temperature, humidity, wind, and rainfall patterns, we aim to help users and organizations make informed decisions.

---

2. Problem Statement

To **predict whether it will rain tomorrow** (Yes/No) using past weather data. The classification model can assist in preparing for weather-related risks and enhance decision-making in agriculture, event planning, and infrastructure management.

---

3. Objectives

- Preprocess the weather dataset for use in a machine learning pipeline.
- Train a **Logistic Regression** model to predict rainfall occurrence.
- Evaluate the model's performance using standard classification metrics.
- Visualize the results with a **confusion matrix heatmap** for better understanding.

---

4. Methodology

4.1 Data Collection

- The dataset used includes weather observations from various locations in Australia. It contains features like temperature, humidity, wind speed, pressure, and rainfall measurements.

4.2 Data Preprocessing

- **Handling Missing Values**: Numerical values imputed with **mean**; categorical with **mode**.
- **Encoding**: Used **one-hot encoding** for categorical variables (e.g., WindDirection, Location).
- **Scaling**: Applied **StandardScaler** to normalize features.
- **Splitting**: The dataset is divided into **80% training** and **20% testing** sets.

4.3 Model Building

- A **Logistic Regression** classifier is trained on the preprocessed data to predict the binary outcome: **RainTomorrow** (Yes/No).

4.4 Model Evaluation

- Evaluated using the following metrics:
    - Accuracy
    - Precision
    - Recall
    - F1-Score
- A **confusion matrix** is created and visualized using **Seaborn heatmap**.

---

5. Data Preprocessing

- **Numerical Features**: Missing values filled with **column mean**.
- **Categorical Features**: Missing values filled with **mode**, then **one-hot encoded**.
- **Standardization**: All numerical features scaled to have zero mean and unit variance.
- **Train-Test Split**: Dataset split into 80% training and 20% testing.

---

6. Model Implementation

A **Logistic Regression** model is selected due to its effectiveness in binary classification and ease of implementation. The model is trained on the processed training set and then used to predict the **RainTomorrow** label on the test set.

---

7. Evaluation Metrics

- **Accuracy**: Proportion of total correct predictions.
- **Precision**: Correctly predicted rain days out of all predicted rain days.
- **Recall**: Actual rain days correctly predicted.
- **F1 Score**: Harmonic mean of precision and recall.
- **Confusion Matrix**: Heatmap visualization to interpret prediction outcomes.

---

8. Results and Analysis

- The model showed reasonable prediction capability on the test dataset.
- The **confusion matrix** indicated the trade-off between predicting rain (true positives) and missing actual rain days (false negatives).
- **Precision and recall** revealed the effectiveness of the model in identifying rainy days with minimized false alarms.

---

9. Conclusion

The Logistic Regression model effectively predicted the likelihood of rain on the next day using historical weather data. This project highlights the potential of machine learning in **weather forecasting** and supports automation in climate-sensitive domains.

Future Improvements:

- Handle **class imbalance** (e.g., SMOTE or resampling).
- Experiment with advanced models like **Random Forest**, **Logistic Regression.**
- Incorporate **time-series components** or real-time weather feeds for enhanced accuracy.

---

10. References

- Kaggle Weather Dataset
- Scikit-learn Documentation
- Pandas Documentation
- Seaborn Visualization Library
- Research papers on weather prediction using machine learning

[1]:
```python
import pandas as pd
from sklearn.model_selection import train_test_split as tts
```

[2]:
```python
df=pd.read_csv('weather.csv',low_memory=True)
```

[3]:
```python
df.head()
```

[3]:

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	...	Humidity9am	Humidity3pm	Pressure9am
0	2008-12-01	Albury	13.4	22.9	0.6	NaN	NaN	W	44.0	W	...	71.0	22.0	1007.7
1	2008-12-02	Albury	7.4	25.1	0.0	NaN	NaN	WNW	44.0	NNW	...	44.0	25.0	1010.6
2	2008-12-03	Albury	12.9	25.7	0.0	NaN	NaN	WSW	46.0	W	...	38.0	30.0	1007.6
3	2008-12-04	Albury	9.2	28.0	0.0	NaN	NaN	NE	24.0	SE	...	45.0	16.0	1017.6
4	2008-12-05	Albury	17.5	32.3	1.0	NaN	NaN	W	41.0	ENE	...	82.0	33.0	1010.8

5 rows × 23 columns

[4]:
```python
df.shape
```

[4]:
```
(145460, 23)
```

[5]:
```python
df.isnull().sum()
```

[5]:
```
Date                 0
Location             0
MinTemp           1485
MaxTemp           1261
Rainfall          3261
Evaporation      62790
Sunshine         69835
WindGustDir      10326
WindGustSpeed    10263
WindDir9am       10566
WindDir3pm        4228
WindSpeed9am      1767
WindSpeed3pm      3062
Humidity9am       2654
Humidity3pm       4507
Pressure9am      15065
Pressure3pm      15028
Cloud9am         55888
Cloud3pm         59358
Temp9am           1767
Temp3pm           3609
```

[10]:
```
from sklearn.preprocessing import LabelEncoder
```

[11]:
```
df.dtypes
```

[11]:
```
Location          object
MinTemp          float64
MaxTemp          float64
Rainfall         float64
WindGustDir       object
WindGustSpeed    float64
WindDir9am        object
WindDir3pm        object
WindSpeed9am     float64
WindSpeed3pm     float64
Humidity9am      float64
Humidity3pm      float64
Pressure9am      float64
Pressure3pm      float64
Temp9am          float64
Temp3pm          float64
RainToday         object
RainTomorrow      object
dtype: object
```

[12]:
```
le1=LabelEncoder()
le2=LabelEncoder()
le3=LabelEncoder()
le4=LabelEncoder()
le5=LabelEncoder()
le6=LabelEncoder()

df['Location']=le1.fit_transform(df['Location'])
df['WindGustDir']=le2.fit_transform(df['WindGustDir'])
df['WindDir9am']=le3.fit_transform(df['WindDir9am'])
df['WindDir3pm']=le4.fit_transform(df['WindDir3pm'])
df['RainToday']=le5.fit_transform(df['RainToday'])
df['RainTomorrow']=le6.fit_transform(df['RainTomorrow'])
```

[13]:
```
df.head()
```

[13]:

	Location	MinTemp	MaxTemp	Rainfall	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm
0	2	13.4	22.9	0.6	13	44.0	13	14	20.0	24.0	71.0	22.0
1	2	7.4	25.1	0.0	14	44.0	6	15	4.0	22.0	44.0	25.0
2	2	12.9	25.7	0.0	15	46.0	13	15	19.0	26.0	38.0	30.0
3	2	9.2	28.0	0.0	4	24.0	9	0	11.0	9.0	45.0	16.0
4	2	17.5	32.3	1.0	13	41.0	1	7	7.0	20.0	82.0	33.0

[61]:
```
accuracy_score(Y_test,y_pred)*100
```

[61]: 84.09413813648655

[63]:
```
user_input_dict = {
    'Date': '2023-01-01',
    'Location': 'Sydney',
    'MinTemp': 15.0,
    'MaxTemp': 28.0,
    'Rainfall': 2.0,
    'Evaporation': 5.0,
    'Sunshine': 8.0,
    'WindGustDir': 'N',
    'WindGustSpeed': 40.0,
    'WindDir9am': 'NE',
    'WindDir3pm': 'E',
    'WindSpeed9am': 15.0,
    'WindSpeed3pm': 20.0,
    'Humidity9am': 65.0,
    'Humidity3pm': 55.0,
    'Pressure9am': 1012.0,
    'Pressure3pm': 1010.0,
    'Cloud9am': 3.0,
    'Cloud3pm': 4.0,
    'Temp9am': 20.0,
    'Temp3pm': 27.0,
    'RainToday': 'No'
}
```

[65]:
```
user_df = pd.DataFrame([user_input_dict])
user_df=user_df.drop(columns=['Date','Evaporation','Sunshine','Cloud9am','Cloud3pm'])
user_df['Location']=le1.fit_transform(user_df['Location'])
user_df['WindGustDir']=le2.fit_transform(user_df['WindGustDir'])
user_df['WindDir9am']=le3.fit_transform(user_df['WindDir9am'])
user_df['WindDir3pm']=le4.fit_transform(user_df['WindDir3pm'])
user_df['RainToday']=le5.fit_transform(user_df['RainToday'])
```

[67]:
```
user_df.head()
```

[67]:

	Location	MinTemp	MaxTemp	Rainfall	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm
0	0	15.0	28.0	2.0	0	40.0	0	0	15.0	20.0	65.0	55.0

[69]:
```
ans= lr.predict(user_df)
if ans==1 :
    print('Yes')
else:
    print('No')
```

No

[6]:
```
df=df.drop(columns=['Date','Evaporation','Sunshine','Cloud9am','Cloud3pm'])
df.head()
```

[6]:

laxTemp	Rainfall	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm
22.9	0.6	W	44.0	W	WNW	20.0	24.0	71.0	22.0	1007.7	1007.1
25.1	0.0	WNW	44.0	NNW	WSW	4.0	22.0	44.0	25.0	1010.6	1007.8
25.7	0.0	WSW	46.0	W	WSW	19.0	26.0	38.0	30.0	1007.6	1008.7
28.0	0.0	NE	24.0	SE	E	11.0	9.0	45.0	16.0	1017.6	1012.8
32.3	1.0	W	41.0	ENE	NW	7.0	20.0	82.0	33.0	1010.8	1006.0

[7]:
```
for i in df.columns:
    # print(i)
    if df[i].dtypes=='object':
        df[i] = df[i].fillna(df[i].mode()[0])
    else:
        df[i]=df[i].fillna(df[i].median())
```

[8]:
```
df.isnull().sum()
```

[8]:
```
Location         0
MinTemp          0
MaxTemp          0
Rainfall         0
WindGustDir      0
WindGustSpeed    0
WindDir9am       0
WindDir3pm       0
WindSpeed9am     0
WindSpeed3pm     0
Humidity9am      0
Humidity3pm      0
Pressure9am      0
Pressure3pm      0
Temp9am          0
Temp3pm          0
RainToday        0
RainTomorrow     0
dtype: int64
```

[9]:
```
df.head()
```

[9]:

	Location	MinTemp	MaxTemp	Rainfall	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm
0	Albury	13.4	22.9	0.6	W	44.0	W	WNW	20.0	24.0	71.0	22.0
1	Albury	7.4	25.1	0.0	WNW	44.0	NNW	WSW	4.0	22.0	44.0	25.0
2	Albury	12.9	25.7	0.0	WSW	46.0	W	WSW	19.0	26.0	38.0	30.0
3	Albury	9.2	28.0	0.0	NE	24.0	SE	E	11.0	9.0	45.0	16.0