PROJECT REPORT

On

ForecastFlow: Rainfall Prediction System

Submitted For Partial Fulfillment of Award of
BACHELOR OF TECHNOLOGY

In

Computer Science & Engineering

By

AL SAIM SHAKEEL
AHTISHAM RIYASAT

Under the Guidance

Of

Mr. Aaftab Alam

INTEGRAL UNIVERSITY, LUCKNOW (INDIA)

COMPUTER SCIENCE ENGINEERING

INTEGRAL UNIVERSITY, LUCKNOW

CERTIFICATE

It is Certified that the project entitled **"ForecastFlow: Rainfall Prediction System"** submitted by **Al Saim Shakeel [1800100635]** and **Ahtisham Riyasat [2100102309]** in the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology (Computer Science Engineering) **Integral University, Lucknow (INDIA)**, is a record of students' own work carried under supervision and guidance of **Mr. Aaftab Alam**. The project report embodies results of original work and studies carried out by students and the contents do not forms the basis for the award of any other degree to the candidate or to anybody else.

Mr. Aaftab Alam
(Project Guide)

Mrs. Kavita Agarwal
(Head of Department)

COMPUTER SCIENCE ENGINEERING

INTEGRAL UNIVERSITY, LUCKNOW

DECLARATION

I/We hereby declare that the project entitled **"ForecastFlow: Rainfall Prediction System"** submitted by me/us in the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology (Computer Science Engineering) of Integral University, Lucknow, is record of my/our own work carried under the supervision and guidance of Mr. Aaftab Alam.

To the best of my/our knowledge this project has not been submitted to Integral University, Lucknow or any other University or Institute for the award of any degree.

Al Saim Shakeel
1800100635

Ahtisham Riyasat
2100102309

ACKNOWLEDGEMENT

I express my gratitude to my supervisor, Mr. Aaftab Alam, for their guidance and valuable insights in the development of the Rainfall Prediction System. Thanks to my colleagues for their collaborative efforts, and to Integral University, Lucknow (INDIA) for providing essential resources. Special thanks to my family and friends for their unwavering support throughout this project.

Al Saim Shakeel
1800100635

Ahtisham Riyasat
2100102309

PREFACE

Welcome to the project report on the Rainfall Prediction System—a venture designed to forecast tomorrow's rainfall and quantify the probability associated with this prediction. This preface provides a glimpse into the motivation, objectives, and collaborative efforts that have shaped the development of this innovative system.

The primary goal of this project is to leverage advanced predictive modeling techniques to enhance our ability to anticipate rainfall occurrences. As you delve into this document, you will find detailed insights into the methodologies employed, challenges encountered, and the outcomes achieved.

I extend my appreciation to those who have contributed to this project, including my supervisor, colleagues, and the support network of family and friends. Their collective input has been instrumental in navigating the complexities of this endeavor.

I invite you to explore the subsequent sections, where the evolution of the Rainfall Prediction System is meticulously documented. Thank you for joining me on this exploration of predictive analytics in the realm of weather forecasting.

ABSTRACT

This project centers on the development of a Rainfall Prediction System designed to forecast the occurrence of rainfall for the upcoming day. The system not only provides a binary prediction (yes or no) but also offers a quantitative measure in the form of a percentage, indicating the likelihood of rainfall.

Employing advanced predictive modeling techniques, the system utilizes historical weather data and relevant features to generate accurate predictions. The objective is to enhance our ability to anticipate rainfall patterns, contributing to more informed decision-making in various sectors, such as agriculture, disaster preparedness, and urban planning.

Throughout the project, a collaborative approach has been embraced, incorporating the guidance of a dedicated supervisor, the contributions of colleagues, and the support of family and friends. The ensuing report details the methodologies employed, challenges faced, and the overall outcomes achieved in the development and implementation of the Rainfall Prediction System. This innovative solution holds the potential to significantly impact the field of weather forecasting and its practical applications.

TABLE OF CONTENTS

INTRODUCTION

BACKGROUND:-

Weather forecasting has been a crucial aspect of human civilization, influencing decisions ranging from agriculture to disaster preparedness. Traditional methods of weather prediction have undergone substantial advancements, yet accurate forecasting of rainfall remains a challenging endeavor. The intricate interplay of various atmospheric factors, coupled with the dynamic nature of weather systems, makes rainfall prediction a complex and vital area of study.

Historically, meteorologists relied on observational methods and simple models to predict rainfall. However, the increasing need for more precise and timely forecasts has spurred the development of sophisticated predictive modeling techniques. These advancements aim to not only determine whether rainfall will occur but also quantitatively measure the likelihood of its incidence.

In this context, the Rainfall Prediction System presented in this project seeks to contribute to the ongoing efforts in weather forecasting. By harnessing the power of data and advanced modeling, the system endeavors to provide more accurate and nuanced predictions of rainfall, offering valuable insights for various sectors dependent on weather information.

PROBLEM STATEMENT

Forecasting next-day rainfall and quantifying the probability of its occurrence in the specific context of Australia presents a distinct set of challenges. The vast and diverse climatic conditions across the continent, coupled with the variability in weather patterns, make accurate predictions a complex task.

Australia, with its unique geography and susceptibility to extreme weather events, necessitates precise and timely rainfall forecasts for effective resource management, agricultural planning, and disaster preparedness. Current methodologies may not adequately capture the intricacies of Australian weather patterns, leading to suboptimal decision-making in various sectors.

The problem at the core of this project lies in addressing the limitations of existing rainfall prediction systems within the Australian context. By developing a tailored Rainfall Prediction System that considers the nuances of the Australian climate, the goal is to enhance the accuracy of next-day rainfall predictions and provide stakeholders with reliable information regarding the likelihood of rainfall events. This project aims to contribute to the optimization of decision-making processes in Australia, where the impact of rainfall extends across diverse industries and regions.

OBJECTIVES

1. Develop a Tailored Predictive Model:

Create a specialized predictive model tailored to the unique climatic conditions of Australia, considering factors such as geography, topography, and regional weather patterns.

2. Accurate Next-Day Rainfall Prediction:

Enhance the precision of next-day rainfall predictions by integrating advanced modeling techniques and leveraging historical weather data specific to Australia.

3. Quantify Probability of Rainfall:

Implement a methodology to quantitatively express the probability of rainfall, providing stakeholders with a nuanced understanding of the likelihood of precipitation.

4. Geospatial Considerations:

Integrate geospatial data to account for regional variations in rainfall patterns, ensuring the model's applicability across diverse landscapes within Australia.

5. Validation and Calibration:

Rigorously validate and calibrate the developed Rainfall Prediction System using independent datasets to assess its accuracy and reliability.

6. User-Friendly Interface with Tkinter:

Design and implement a user-friendly interface for the Rainfall Prediction System using the Tkinter library. Ensure the interface is intuitive, visually appealing, and provides stakeholders with easy access to next-day rainfall predictions and associated probabilities.

7. Contribution to Decision-Making:

Provide valuable insights to decision-makers in agriculture, water resource management, and disaster preparedness by offering reliable information on next-day rainfall and associated probabilities.

8. Continuous Improvement:

Establish a framework for ongoing refinement and improvement of the predictive model, incorporating feedback and adapting to evolving weather patterns.

The objectives outlined above collectively aim to address the specific challenges of predicting next-day rainfall in Australia, with the ultimate goal of contributing to more informed and effective decision-making processes across diverses

Scope and Significance

Scope:

The project aims to predict next-day rainfall and quantify its probability in Australia, utilizing advanced modeling techniques and geospatial considerations. It includes the development of a user-friendly interface for stakeholders in agriculture, water resource management, and disaster preparedness**.**

Significance:

- Precision in agriculture practices.
- Efficient water resource management.
- Enhanced disaster preparedness.
- Economic implications in trading and insurance.
- Contribution to environmental conservation.
- Advancement of meteorological sciences.
- Support for informed decision-making across sectors.

Overview of Predictive Modeling

In the development of the Rainfall Prediction System, various advanced predictive modeling techniques have been employed to enhance the accuracy of next-day rainfall forecasts. The following algorithms have been utilized:

1. Logistic Regression:
Logistic Regression is a fundamental classification algorithm that models the probability of a binary outcome. In this context, it helps predict whether rainfall will occur the next day.

2. K-Nearest Neighbors (KNN):
KNN is a non-parametric algorithm used for both classification and regression tasks. It assesses the similarity of data points to predict the likelihood of rainfall, considering the 'neighbors' in the feature space.

3. XGBoost (Extreme Gradient Boosting):
XGBoost is an ensemble learning method known for its efficiency and predictive power. It sequentially builds decision trees to improve the accuracy of predictions, making it well-suited for complex relationships within weather data.

4. Light GBM (Light Gradient Boosting Machine):
Light GBM is a gradient boosting framework designed for distributed and efficient training. It optimizes the construction of decision trees, enhancing the model's capability to capture intricate patterns in the dataset

Methodology

1. Data Collection:

Gathered historical weather data specific to Australia, incorporating variables such as temperature, humidity, wind speed, and atmospheric pressure. The dataset comprises instances of both rainfall occurrences and non-occurrences.

2. Data Preprocessing:

Conducted thorough cleaning and preprocessing of the dataset, handling missing values, outliers, and ensuring uniform formatting. Feature engineering was performed to extract relevant information and enhance model performance.

3. Exploratory Data Analysis (EDA):

Conducted EDA to gain insights into the distribution of variables, correlations, and potential patterns. Identified key features influencing rainfall predictions and informed the model development process.

4. Fixing Overfitting:

Addressed overfitting concerns through techniques such as regularization, reducing model complexity, and optimizing hyperparameters. Balancing the trade-off between bias and variance was crucial in ensuring models generalize well to new data.

5. Model Training:

Implemented Logistic Regression, K-Nearest Neighbors, XGBoost, and Light GBM algorithms for training the predictive models. Each algorithm was fine-tuned through iterative testing and optimization to maximize performance.

6. Ensemble Modeling:

Applied ensemble techniques to combine the strengths of individual models, leveraging the diversity of algorithms to enhance overall predictive accuracy.

7. Validation and Evaluation:

Utilized cross-validation techniques to validate model performance and assess generalization capabilities. Evaluated models using metrics such as accuracy, precision, recall, and area under the ROC curve.

8. Interface Development with Tkinter:

Utilize the Tkinter library to develop a user-friendly interface for the Rainfall Prediction System. Ensure the interface incorporates design principles for ease of use, clarity, and accessibility, allowing user to interact seamlessly with next-day rainfall predictions and associated probabilities.

9. Model Deployment:

Deployed the trained models and the interface to a suitable environment, making the Rainfall Prediction System operational and ready for making prediction.

10. Continuous Improvement:

Established a framework for ongoing model refinement based on feedback and evolving weather patterns. This involves periodic updates to the dataset, retraining of models, and implementing improvements to ensure sustained accuracy.

This methodology encapsulates a comprehensive approach, integrating data-driven insights, advanced predictive modeling techniques, and strategies to address overfitting, resulting in the development of an effective Rainfall Prediction System tailored to the unique climatic conditions of Australia.

PROJECT DEVELOPMENT

1. Initiation:

Defined the project scope, objectives, and requirements, outlining the need for accurate next-day rainfall predictions with associated probabilities tailored to Australia's diverse climate.

2. Team Formation:

Assembled a multidisciplinary team including data scientists, developers, and domain experts to leverage a diverse skill set for comprehensive project development.

3. Data Acquisition:

Acquired historical weather data specific to Australia, ensuring the dataset captured the nuances of regional climate variations and included relevant features such as temperature, humidity, wind speed, and atmospheric pressure.

4. Data Preprocessing:

- Conducted thorough data cleaning, handled missing values, outliers, and standardized formatting. Employed feature engineering to extract meaningful information and enhance the dataset's suitability for predictive modeling.

5. Model Selection:

Selected logistic regression, K-Nearest Neighbors, XGBoost, and Light GBM as the primary predictive models, considering their suitability for binary classification tasks and ensemble modeling.

6. Algorithm Implementation:

Implemented and fine-tuned each selected algorithm, addressing overfitting concerns and optimizing hyperparameters to achieve robust and accurate next-day rainfall predictions.

7. Ensemble Model Integration:

Integrated ensemble modeling techniques to combine the strengths of individual algorithms, enhancing overall predictive performance and generalization capabilities.

8. Interface Design with Tkinter:

Employ the Tkinter library to design and develop an intuitive and visually appealing user interface for the Rainfall Prediction System. The design process includes considerations for ease of use, accessibility, and effective communication of next-day rainfall predictions and associated probabilities to stakeholders.

9. Testing and Validation:

Conducted rigorous testing and validation, utilizing cross-validation techniques to ensure the reliability and accuracy of the predictive models. Evaluated performance using metrics such as accuracy, precision, recall, and area under the ROC curve.

10. Deployment:

Deployed the Rainfall Prediction System, including the trained models and user interface, to a suitable environment for operational use. Ensured scalability and responsiveness making prediction.

11. Feedback and Refinement:

Established channels for user feedback and continuously monitored model performance. Implemented periodic updates, retraining models, and making improvements based on evolving weather patterns and user insights.

12. Documentation:

Documented the entire project development process, including methodologies, algorithms, and key decisions made. The documentation serves as a comprehensive reference for future enhancements or research endeavors.

The project development lifecycle involved a systematic and collaborative approach, combining expertise in data science, modeling, and software development to create an effective Rainfall Prediction System tailored to the specific requirements of predicting next-day rainfall in Australia.

RESULT ANALYSIS & PROJECT WORK

Prediction Model

ForecastFlow: Rainfall Prediction Model through Data Analysis and Machine Learning ¶

Import Libraries

In [1]:
```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
from sklearn.impute import SimpleImputer
```

Reading Dataset

In [2]:
```
df= pd.read_csv('weatherAUS.csv')
df.head()
```

Out[2]:

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	Wind
0	2008-12-01	Albury	13.4	22.9	0.6	NaN	NaN	W	
1	2008-12-02	Albury	7.4	25.1	0.0	NaN	NaN	WNW	
2	2008-12-03	Albury	12.9	25.7	0.0	NaN	NaN	WSW	
3	2008-12-04	Albury	9.2	28.0	0.0	NaN	NaN	NE	
4	2008-12-05	Albury	17.5	32.3	1.0	NaN	NaN	W	

5 rows × 23 columns

In [3]:
```
df.tail()
```

Out[3]:

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir
145455	2017-06-21	Uluru	2.8	23.4	0.0	NaN	NaN	E
145456	2017-06-22	Uluru	3.6	25.3	0.0	NaN	NaN	NNW
145457	2017-06-23	Uluru	5.4	26.9	0.0	NaN	NaN	N
145458	2017-06-24	Uluru	7.8	27.0	0.0	NaN	NaN	SE
145459	2017-06-25	Uluru	14.9	NaN	0.0	NaN	NaN	NaN

5 rows × 23 columns

In [4]:
```
print(df.columns)
```

```
Index(['Date', 'Location', 'MinTemp', 'MaxTemp', 'Rainfall', 'Evaporatio
n',
       'Sunshine', 'WindGustDir', 'WindGustSpeed', 'WindDir9am', 'WindDir3
pm',
       'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm',
       'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am',
       'Temp3pm', 'RainToday', 'RainTomorrow'],
      dtype='object')
```

Data Processing

In [5]:
```
df = df.dropna(subset=['RainTomorrow']) #removing rows with null values in
```

In [6]:
```
df['Date'] = pd.to_datetime(df['Date'])
df.dtypes
```

Out[6]:
```
Date              datetime64[ns]
Location                  object
MinTemp                  float64
MaxTemp                  float64
Rainfall                 float64
Evaporation              float64
Sunshine                 float64
WindGustDir               object
WindGustSpeed            float64
WindDir9am                object
WindDir3pm                object
WindSpeed9am             float64
WindSpeed3pm             float64
Humidity9am              float64
Humidity3pm              float64
Pressure9am              float64
Pressure3pm              float64
Cloud9am                 float64
Cloud3pm                 float64
Temp9am                  float64
Temp3pm                  float64
RainToday                 object
RainTomorrow              object
dtype: object
```

In [7]:
```
df['Month'] = df['Date'].dt.month #only cosidering months
```

In [8]:
```
df = df.drop('Date', axis=1) #Removing Date Column
```

Fixing Null values

In [9]:
```
(df.isnull().sum() / df.shape[0]) * 100 #percentage of null values in each
```

Out[9]:
```
Location          0.000000
MinTemp           0.447983
MaxTemp           0.226453
Rainfall          0.988797
Evaporation      42.789026
Sunshine         47.692924
WindGustDir       6.561504
WindGustSpeed     6.519308
WindDir9am        7.041838
WindDir3pm        2.656952
WindSpeed9am      0.948007
WindSpeed3pm      1.849599
Humidity9am       1.247600
Humidity3pm       2.538803
Pressure9am       9.855619
Pressure3pm       9.832411
Cloud9am         37.735332
Cloud3pm         40.152469
Temp9am           0.635756
Temp3pm           1.917113
RainToday         0.988797
RainTomorrow      0.000000
Month             0.000000
dtype: float64
```

In [10]:
```
non_numeric_df = df.select_dtypes('object')
non_numeric_df.head()
```

Out[10]:

	Location	WindGustDir	WindDir9am	WindDir3pm	RainToday	RainTomorrow
0	Albury	W	W	WNW	No	No
1	Albury	WNW	NNW	WSW	No	No
2	Albury	WSW	W	WSW	No	No
3	Albury	NE	SE	E	No	No
4	Albury	W	ENE	NW	No	No

In [11]:
```
#fixing null values of categorical columns
for i in non_numeric_df.columns:
    non_numeric_df[i].fillna(non_numeric_df[i].mode()[0], inplace=True)
```

In [12]:
```
non_numeric_df.isnull().sum()
```

Out[12]:
```
Location        0
WindGustDir     0
WindDir9am      0
WindDir3pm      0
RainToday       0
RainTomorrow    0
dtype: int64
```

In [13]:
```
numeric_df = df.select_dtypes(exclude='object')
numeric_df.head()
```

Out[13]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpeed	WindSpeed9am	W
0	13.4	22.9	0.6	NaN	NaN	44.0	20.0	
1	7.4	25.1	0.0	NaN	NaN	44.0	4.0	
2	12.9	25.7	0.0	NaN	NaN	46.0	19.0	
3	9.2	28.0	0.0	NaN	NaN	24.0	11.0	
4	17.5	32.3	1.0	NaN	NaN	41.0	7.0	

In [14]:
```
#fixing null values of numerical columns
for i in numeric_df.columns:
    numeric_df[i].fillna(numeric_df[i].median(), inplace=True)
```

In [15]:
```
df_new = numeric_df.join(non_numeric_df) #Joining numerical and non-numeric
```

In [16]:
```
df_new.isnull().sum() #checking null values
```

Out[16]:
```
MinTemp          0
MaxTemp          0
Rainfall         0
Evaporation      0
Sunshine         0
WindGustSpeed    0
WindSpeed9am     0
WindSpeed3pm     0
Humidity9am      0
Humidity3pm      0
Pressure9am      0
Pressure3pm      0
Cloud9am         0
Cloud3pm         0
Temp9am          0
Temp3pm          0
Month            0
Location         0
WindGustDir      0
WindDir9am       0
WindDir3pm       0
RainToday        0
RainTomorrow     0
dtype: int64
```

In [17]:
```
df_new.head()
```

Out[17]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpeed	WindSpeed9am	W
0	13.4	22.9	0.6	4.8	8.5	44.0	20.0	
1	7.4	25.1	0.0	4.8	8.5	44.0	4.0	
2	12.9	25.7	0.0	4.8	8.5	46.0	19.0	
3	9.2	28.0	0.0	4.8	8.5	24.0	11.0	
4	17.5	32.3	1.0	4.8	8.5	41.0	7.0	

5 rows × 23 columns

In [18]:
```
df_new['RainTomorrow'].value_counts()
```

Out[18]:
```
No      110316
Yes      31877
Name: RainTomorrow, dtype: int64
```

In [19]:
```
df_new['RainToday'].value_counts()
```

Out[19]:
```
No      110738
Yes      31455
Name: RainToday, dtype: int64
```

In [20]:
```
df_new.shape
```

Out[20]:
```
(142193, 23)
```

In [21]:
```
df_new.head()
```

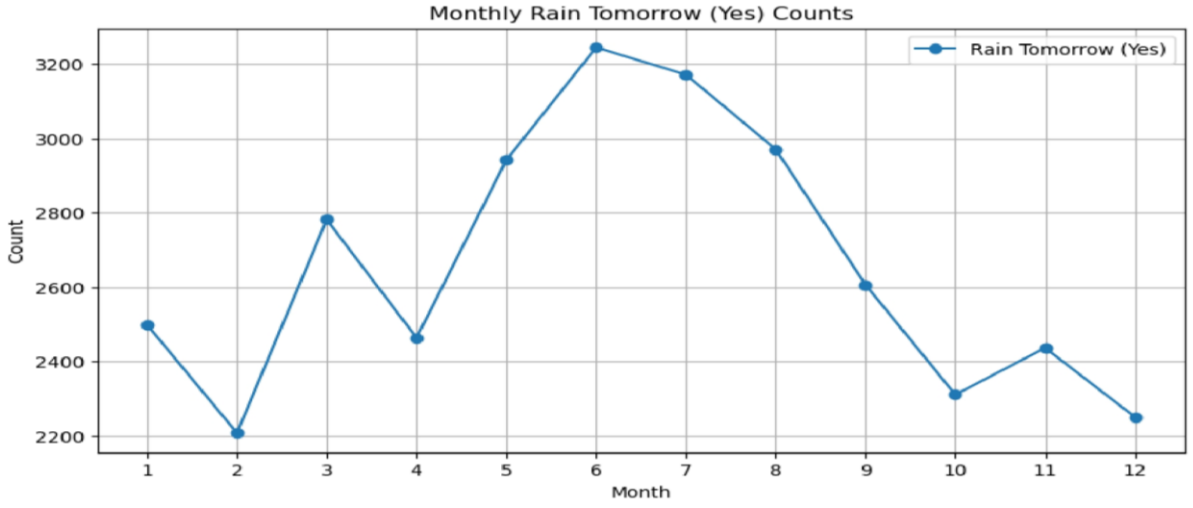
Out[21]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpeed	WindSpeed9am	W
0	13.4	22.9	0.6	4.8	8.5	44.0	20.0	
1	7.4	25.1	0.0	4.8	8.5	44.0	4.0	
2	12.9	25.7	0.0	4.8	8.5	46.0	19.0	
3	9.2	28.0	0.0	4.8	8.5	24.0	11.0	
4	17.5	32.3	1.0	4.8	8.5	41.0	7.0	

5 rows × 23 columns

Which month got most number of rainfall?

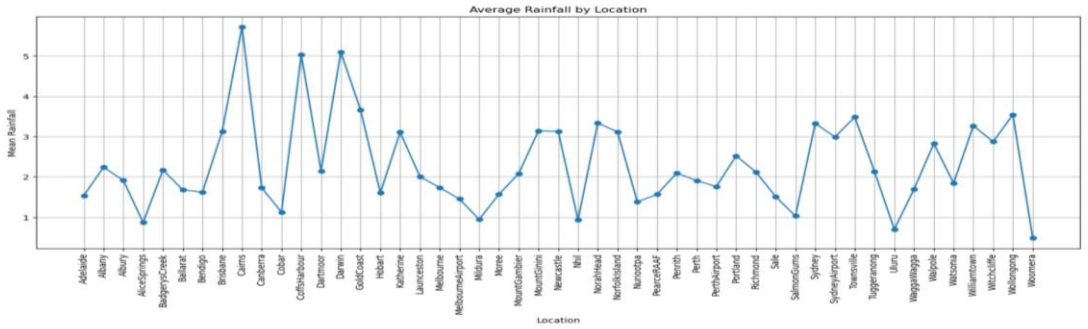
In [22]:
```
# Group the data by month and count the occurrences of "Yes" for RainToday
monthly_rain_today_yes = df_new[df_new['RainToday'] == 'Yes'].groupby('Mont
monthly_rain_tomorrow_yes = df_new[df_new['RainTomorrow'] == 'Yes'].groupby

# Create a line graph for RainToday "Yes" counts
plt.figure(figsize=(10, 5))
plt.plot(monthly_rain_today_yes.index, monthly_rain_today_yes.values, marke
plt.title('Monthly Rain Today (Yes) Counts')
plt.xlabel('Month')
plt.ylabel('Count')
plt.xticks(monthly_rain_today_yes.index)
plt.grid(True)
plt.legend()
plt.show()

# Create a line graph for RainTomorrow "Yes" counts
plt.figure(figsize=(10, 5))
plt.plot(monthly_rain_tomorrow_yes.index, monthly_rain_tomorrow_yes.values,
plt.title('Monthly Rain Tomorrow (Yes) Counts')
plt.xlabel('Month')
plt.ylabel('Count')
plt.xticks(monthly_rain_tomorrow_yes.index)
plt.grid(True)
plt.legend()
plt.show()
```



According to this, we can say that Australia have most no of times rainfall in the 6th month i.e June between 2008 to 2017

Which state got highest average rainfall?

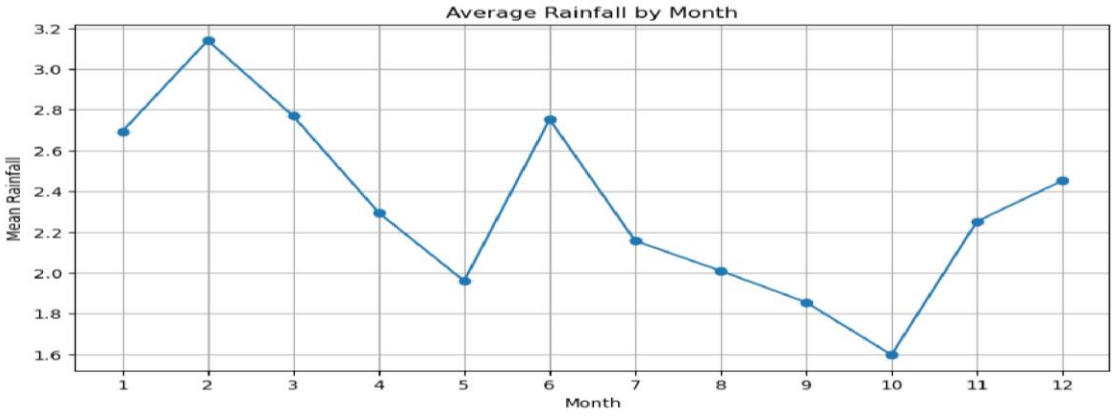
In [23]:
```
import matplotlib.pyplot as plt

# Group the data by location and calculate the mean of Rainfall
mean_rainfall_by_location = df_new.groupby('Location')['Rainfall'].mean()

# Create a line graph for Rainfall by Location
plt.figure(figsize=(18, 6))
mean_rainfall_by_location.plot(kind='line', marker='o', linestyle='-')
plt.title('Average Rainfall by Location')
plt.xlabel('Location')
plt.ylabel('Mean Rainfall')
plt.xticks(range(len(mean_rainfall_by_location.index)), mean_rainfall_by_lc
plt.grid(True)
plt.show()
```



According to this, we can say that Cairns got highest average rainfall in Australia between 2008 to 2017

Which month got highest average rainfall?

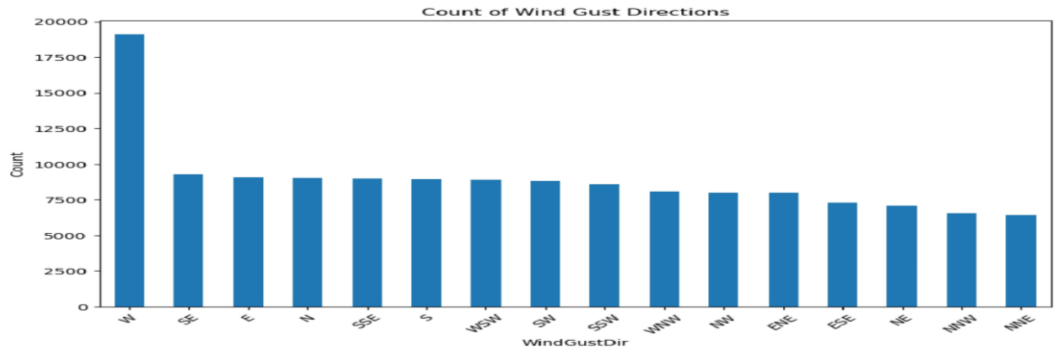
In [24]:
```
# Group the data by month and calculate the mean of Rainfall
mean_rainfall_by_month = df_new.groupby('Month')['Rainfall'].mean()

# Create a line graph for Rainfall by Month
plt.figure(figsize=(10, 5))
mean_rainfall_by_month.plot(kind='line', marker='o', linestyle='-')
plt.title('Average Rainfall by Month')
plt.xlabel('Month')
plt.ylabel('Mean Rainfall')
plt.xticks(mean_rainfall_by_month.index)
plt.grid(True)
plt.show()
```



According to this, Australia have highest average rainfall in the 2nd month i.e Febuary between 2008 and 2017

Which direction of wind has the most cound observed in terms of gust occurrences?

In [25]:
```
plt.figure(figsize=(10, 6))
df_new['WindGustDir'].value_counts().plot(kind='bar')
plt.xlabel('WindGustDir')
plt.ylabel('Count')
plt.title('Count of Wind Gust Directions')
plt.xticks(rotation=45)
plt.show()
```



The wind direction that has the highest frequency of gust occurrences is the West. This indicates that gusty conditions are most commonly associated with winds coming from the West.

Converting string values to float values

In [26]:
```
df_new['RainToday'] = df_new['RainToday'].map({'Yes': 1, 'No': 0})
df_new['RainTomorrow'] = df_new['RainTomorrow'].map({'Yes': 1, 'No': 0})
```

In [27]:
```
#df = df.dropna(subset=['Location']) #removing rows with null values in Loc
```

In [28]:
```
df_new['WindGustDir'] = df_new['WindGustDir'].map({
    'W': 4,
    'SE': 23,
    'E': 3,
    'N': 1,
    'SSE': 223,
    'S': 2,
    'WSW': 424,
    'SW': 24,
    'SSW': 224,
    'WNW': 414,
    'NW': 14,
    'ENE': 313,
    'ESE': 323,
    'NE': 13,
    'NNW': 114,
    'NNE': 113
})
```

Changing string values to float values

Type *Markdown* and LaTeX: α^2

In [29]:
```
df_new.shape
```

Out[29]:
```
(142193, 23)
```

Changing string values to float values

In [30]:
```
df_new['WindDir9am'] = df_new['WindDir9am'].map({
    'W': 4,
    'SE': 23,
    'E': 3,
    'N': 1,
    'SSE': 223,
    'S': 2,
    'WSW': 424,
    'SW': 24,
    'SSW': 224,
    'WNW': 414,
    'NW': 14,
    'ENE': 313,
    'ESE': 323,
    'NE': 13,
    'NNW': 114,
    'NNE': 113
})
```

In [31]:
```
df_new['WindDir3pm'] = df_new['WindDir3pm'].map({
    'W': 4,
    'SE': 23,
    'E': 3,
    'N': 1,
    'SSE': 223,
    'S': 2,
    'WSW': 424,
    'SW': 24,
    'SSW': 224,
    'WNW': 414,
    'NW': 14,
    'ENE': 313,
    'ESE': 323,
    'NE': 13,
    'NNW': 114,
    'NNE': 113
})
```

In [32]:
```
df_new.head()
```

Out[32]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpeed	WindSpeed9am	W
0	13.4	22.9	0.6	4.8	8.5	44.0	20.0	
1	7.4	25.1	0.0	4.8	8.5	44.0	4.0	
2	12.9	25.7	0.0	4.8	8.5	46.0	19.0	
3	9.2	28.0	0.0	4.8	8.5	24.0	11.0	
4	17.5	32.3	1.0	4.8	8.5	41.0	7.0	

5 rows × 23 columns

In [33]:
```
print(df_new.columns)
```

```
Index(['MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine',
       'WindGustSpeed', 'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am',
       'Humidity3pm', 'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3p
m',
       'Temp9am', 'Temp3pm', 'Month', 'Location', 'WindGustDir', 'WindDir9
am',
       'WindDir3pm', 'RainToday', 'RainTomorrow'],
      dtype='object')
```

In [34]:
```
df_new['Location'] = df_new['Location'].map({
    'Portland': 1,
    'Sydney': 2,
    'Cairns': 3,
    'Albany': 4,
    'Darwin': 5,
    'MountGambier': 6,
    'Dartmoor': 7,
    'NorfolkIsland': 8,
    'CoffsHarbour': 9,
    'Walpole': 10,
    'Witchcliffe': 11,
    'Hobart': 12,
    'NorahHead': 13,
    'GoldCoast': 14,
    'Canberra': 15,
    'Ballarat': 16,
    'SydneyAirport': 17,
    'Brisbane': 18,
    'Adelaide': 19,
    'MountGinini': 20,
    'Launceston': 21,
    'Watsonia': 22,
    'Perth': 23,
    'Wollongong': 24,
    'Sale': 25,
    'Newcastle': 26,
    'Albury': 27,
    'BadgerysCreek': 28,
    'Nuriootpa': 29,
    'MelbourneAirport': 30,
    'Tuggeranong': 31,
    'Penrith': 32,
    'PerthAirport': 33,
    'Bendigo': 34,
    'Richmond': 35,
    'Williamtown': 36,
    'WaggaWagga':37 ,
    'SalmonGums' :38 ,
    'Townsville' :39 ,
    'Cobar' :40 ,
    'Melbourne' :41 ,
    'PearceRAAF' :42 ,
    'Moree' :43 ,
    "Mildura" :44 ,
     "AliceSprings" :45 ,
     "Woomera" :46 ,
     "Nhil" :47 ,
     "Katherine" :48 ,
     "Uluru" :49
})
```

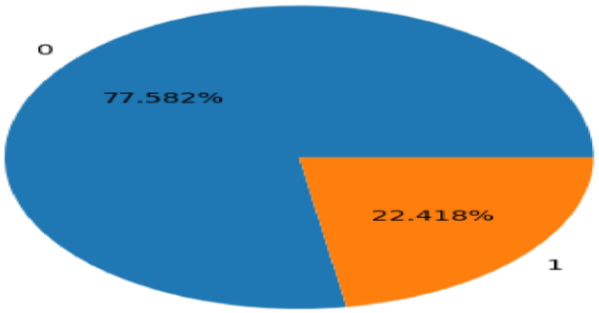
We have opted for a mapping strategy over label encoding due to the greater degree of control it offers in the data transformation process. This approach allows us to more effectively manage the representation of categorical variables in our dataset.

PIE Chart - RainTomorrow (Yes,No)

In [35]:
```
t= df_new['RainTomorrow'].value_counts()
labels = list(t.index)
t
```

Out[35]:
```
0     110316
1      31877
Name: RainTomorrow, dtype: int64
```

In [36]:
```
plt.pie(t, labels=labels, autopct='%1.3f%%')
plt.show()
```



Given the observed imbalance in our dataset, we will employ undersampling techniques to achieve balance. This approach is particularly suitable considering the substantial size of our dataset.

Undersampling

In [37]:
```
import seaborn as sns
```

In [38]:
```
minority_class_len= len(df_new[df_new["RainTomorrow"] == 1])
print(minority_class_len)
```

```
31877
```

In [39]:
```
majority_class_len= len(df_new[df_new["RainTomorrow"] == 0])
print(majority_class_len)
```

```
110316
```

In [40]:
```
majority_class_indices= df_new[df_new["RainTomorrow"] == 0].index
print(majority_class_indices)
```

```
Int64Index([      0,       1,       2,       3,       4,       5,       6,
7,
                  9,      13,
             ...
             145449, 145450, 145451, 145452, 145453, 145454, 145455, 14545
6,
             145457, 145458],
            dtype='int64', length=110316)
```

In [41]:
```
random_majority_indices = np.random.choice(majority_class_indices,
                                           minority_class_len,
                                           replace=False)

print(len(random_majority_indices))
```

```
31877
```

In [42]:
```
minority_class_indices= df_new[df_new["RainTomorrow"] == 1].index
print(minority_class_indices)
```

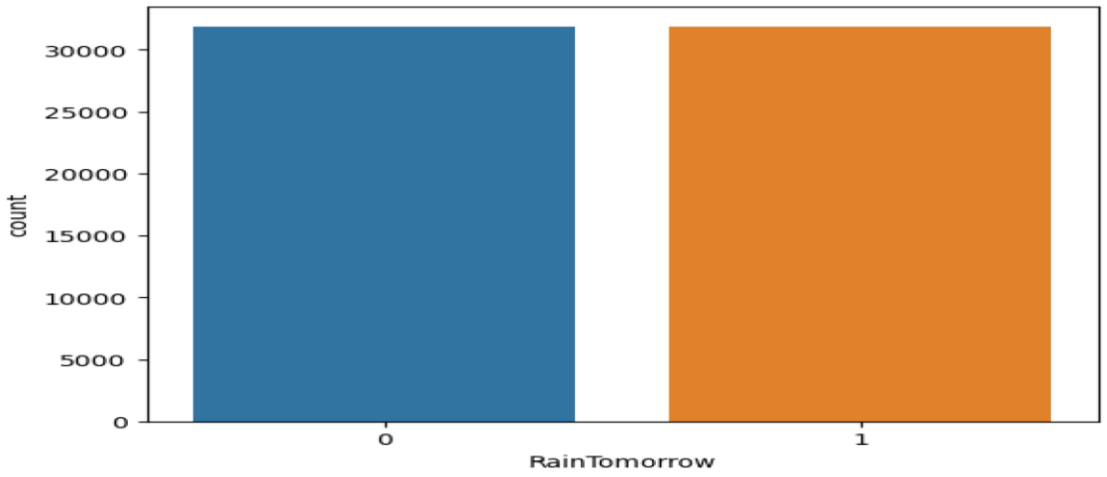
```
Int64Index([      8,      10,      11,      12,      16,      17,      28,      5
2,
                 73,      99,
             ...
             145306, 145309, 145316, 145320, 145321, 145324, 145390, 14539
1,
             145392, 145393],
            dtype='int64', length=31877)
```

In [43]:
```
 under_sample_indices = np.concatenate([minority_class_indices, random_majc
```

In [44]:
```
under_sample = df_new.loc[under_sample_indices]
```

In [45]:
```
sns.countplot(x="RainTomorrow", data= under_sample)
```

Out[45]:
```
<Axes: xlabel='RainTomorrow', ylabel='count'>
```



In [46]:
```
print(under_sample.shape)
```

```
(63754, 23)
```

Upon implementing undersampling, we have achieved an equal distribution of the categories "No" (0) and "Yes" (1), each with a count of 31,877. This results in a balanced dataset comprising 63,754 rows and 23 columns, which is a substantial size for our analysis.

In [47]:
```
under_sample.drop(columns=["Sunshine", "Evaporation"], inplace=True)
```

In [48]:
```
under_sample.columns
```

Out[48]:
```
Index(['MinTemp', 'MaxTemp', 'Rainfall', 'WindGustSpeed', 'WindSpeed9am',
       'WindSpeed3pm', 'Humidity9am', 'Humidity3pm', 'Pressure9am',
       'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am', 'Temp3pm', 'Mont
h',
       'Location', 'WindGustDir', 'WindDir9am', 'WindDir3pm', 'RainToday',
       'RainTomorrow'],
      dtype='object')
```

Taking X as independent variable and y as dependent variable

In [49]:
```
y= under_sample['RainTomorrow']
x= under_sample.drop('RainTomorrow', axis=1)
print(x.shape)
print(y.shape)
```

```
(63754, 20)
(63754,)
```

In [50]:
```
x.head()
```

Out[50]:

	MinTemp	MaxTemp	Rainfall	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity
8	9.7	31.9	0.0	80.0	7.0	28.0	
10	13.4	30.4	0.0	30.0	17.0	6.0	
11	15.9	21.7	2.2	31.0	15.0	13.0	
12	15.9	18.6	15.6	61.0	28.0	28.0	
16	14.1	20.9	0.0	22.0	11.0	9.0	

In [51]:
```
y.head()
```

Out[51]:
```
8     1
10    1
11    1
12    1
16    1
Name: RainTomorrow, dtype: int64
```

In [52]:
```
y.value_counts()
```

Out[52]:
```
1    31877
0    31877
Name: RainTomorrow, dtype: int64
```

In [53]:
```
x.shape
```

Out[53]:
```
(63754, 20)
```

In [54]:
```
x.head()
```

Out[54]:

	MinTemp	MaxTemp	Rainfall	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity
8	9.7	31.9	0.0	80.0	7.0	28.0	
10	13.4	30.4	0.0	30.0	17.0	6.0	
11	15.9	21.7	2.2	31.0	15.0	13.0	
12	15.9	18.6	15.6	61.0	28.0	28.0	
16	14.1	20.9	0.0	22.0	11.0	9.0	

In [55]:
```
y.value_counts()
```

Out[55]:
```
1     31877
0     31877
Name: RainTomorrow, dtype: int64
```

Splitting DATA

In [56]:
```
from sklearn.model_selection import train_test_split
```

In [57]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.33,random_
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(42715, 20)
(21039, 20)
(42715,)
(21039,)
```

Importing confusion matrix and classification report

In [58]:
```
from sklearn.metrics import confusion_matrix, classification_report
```

In [59]:
```
def eval_model(ytest,ypred):
    cm = confusion_matrix(ytest,ypred)
    print(cm)
    print(classification_report(ytest,ypred, zero_division=1))
def mscore(model):
    print('Train Score',model.score(x_train,y_train))
    print('Test Score',model.score(x_test,y_test))
```

MODELING

KNN

In [60]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [61]:
```
m2 = KNeighborsClassifier(n_neighbors=15)
m2.fit(x_train,y_train)
```

Out[61]:
```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=15)
```

In [62]:
```
mscore(m2)
```

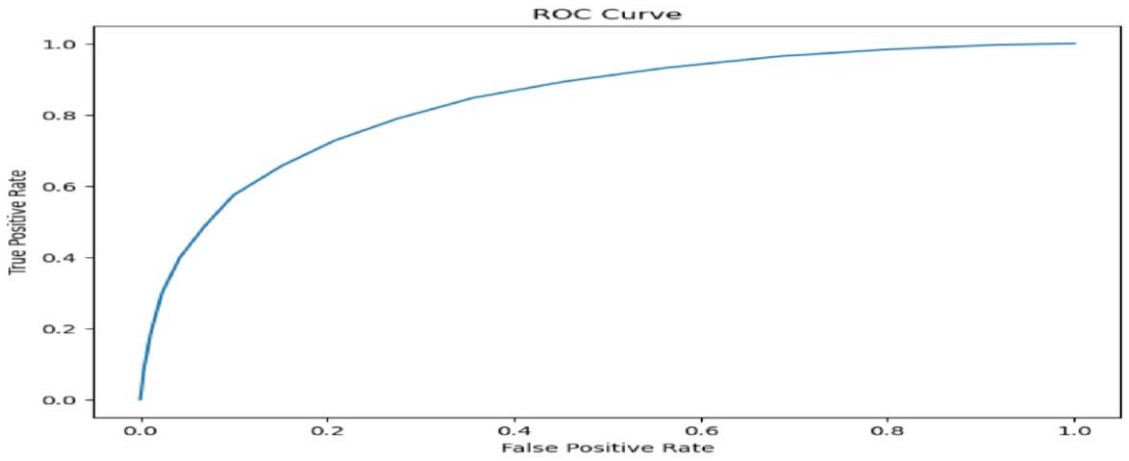
```
Train Score 0.7891138944164813
Test Score 0.759684395646181
```

In [63]:
```
ypred_m2 = m2.predict(x_test)
eval_model(y_test,ypred_m2)
```

```
[[8346 2187]
 [2869 7637]]
              precision    recall  f1-score   support

           0       0.74      0.79      0.77     10533
           1       0.78      0.73      0.75     10506

    accuracy                           0.76     21039
   macro avg       0.76      0.76      0.76     21039
weighted avg       0.76      0.76      0.76     21039
```

ROC Curve for KNN

In [64]:
```
from sklearn.metrics import roc_curve, roc_auc_score

y_pred_proba = m2.predict_proba(x_test)[:, 1]
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.show()
```



Logistic Regression

In [66]:
```
from sklearn.linear_model import LogisticRegression
```

In [67]:
```
m1 = LogisticRegression(max_iter=20000)
m1.fit(x_train,y_train)
```

Out[67]:
```
LogisticRegression
LogisticRegression(max_iter=20000)
```

In [68]:
```
mscore(m1)
```

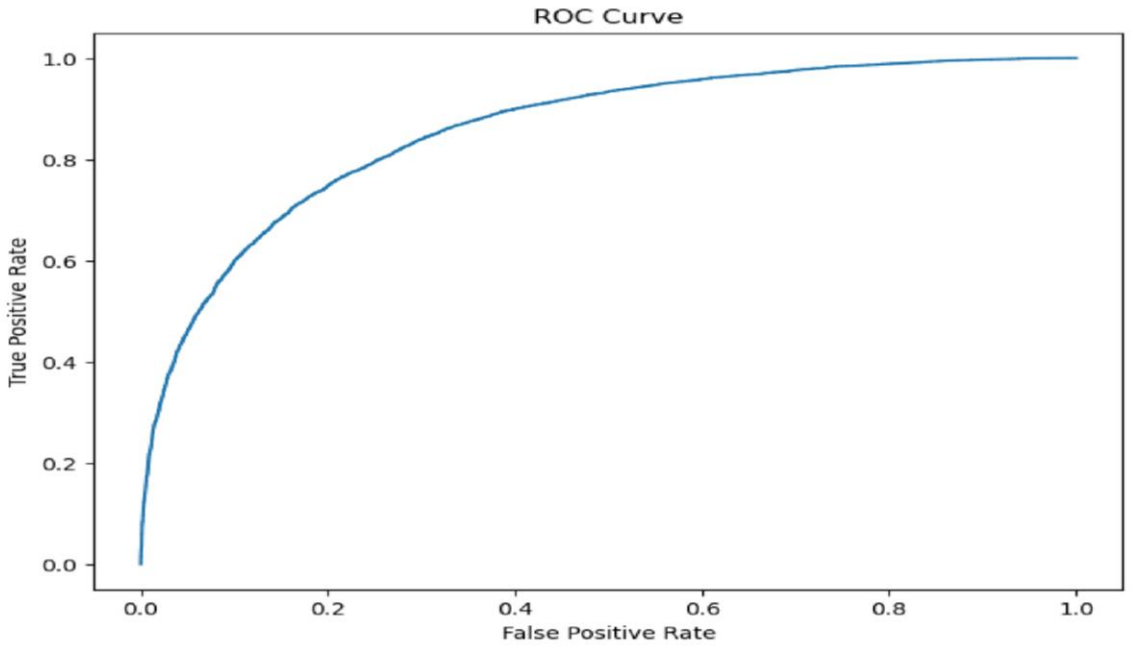
```
Train Score 0.7697061922041437
Test Score 0.7747991824706497
```

In [69]:
```
ypred_m1 = m1.predict(x_test)
eval_model(y_test,ypred_m1)
```

```
[[8304 2229]
 [2509 7997]]
              precision    recall  f1-score   support

           0       0.77      0.79      0.78     10533
           1       0.78      0.76      0.77     10506

    accuracy                           0.77     21039
   macro avg       0.77      0.77      0.77     21039
weighted avg       0.77      0.77      0.77     21039
```

ROC Curve for Logistic Regression

In [70]:
```
from sklearn.metrics import roc_curve, roc_auc_score

y_pred_proba = m1.predict_proba(x_test)[:, 1]
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.show()
```



XG Boost

In [71]:
```
import xgboost as xgb
params_xgb ={'n_estimators': 300,
             'max_depth': 13}

model_xgb = xgb.XGBClassifier(**params_xgb)
model_xgb.fit(x_train,y_train)
```

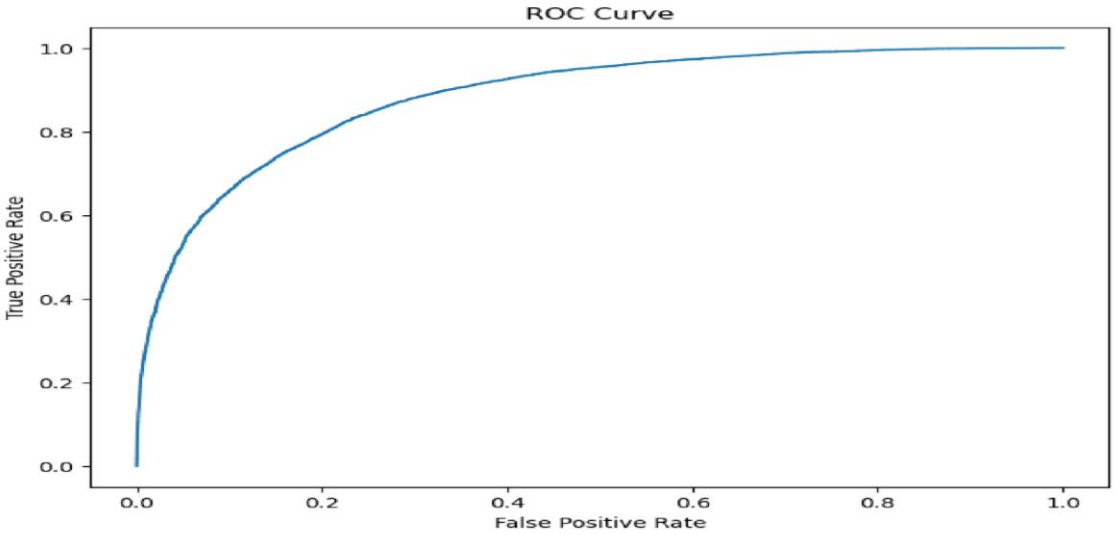
Out[71]:
```
                                    XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_type
s=None,
              gamma=None, gpu_id=None, grow_policy=None, importance_ty
pe=None,
              interaction_constraints=None, learning_rate=None, max_bi
n=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
```

In [72]:
```
mscore(model_xgb)
ypred_model_xgb = model_xgb.predict(x_test)
eval_model(y_test,ypred_model_xgb)
```

```
Train Score 0.999953178040501
Test Score 0.7969485241693997
[[8449 2084]
 [2188 8318]]
              precision    recall  f1-score   support

           0       0.79      0.80      0.80     10533
           1       0.80      0.79      0.80     10506

    accuracy                           0.80     21039
   macro avg       0.80      0.80      0.80     21039
weighted avg       0.80      0.80      0.80     21039
```

ROC Curve for XG BOOST

In [73]:
```
from sklearn.metrics import roc_curve, roc_auc_score

y_pred_proba = model_xgb.predict_proba(x_test)[:, 1]
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.show()
```



LightGBM

In [74]:
```
import lightgbm as lgb
params_lgb ={'colsample_bytree': 0.95,
             'max_depth': 16,
             'min_split_gain': 0.1,
             'n_estimators': 200,
             'num_leaves': 50,
             'reg_alpha': 1.2,
             'reg_lambda': 1.2,
             'subsample': 0.95,
             'subsample_freq': 20}

model_lgb = lgb.LGBMClassifier(**params_lgb)
model_lgb.fit(x_train,y_train)
```

```
[LightGBM] [Info] Number of positive: 21371, number of negative: 21344
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead
of testing was 0.001997 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 2256
[LightGBM] [Info] Number of data points in the train set: 42715, number of
used features: 20
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.500316 -> initscore=0.00
1264
[LightGBM] [Info] Start training from score 0.001264
```

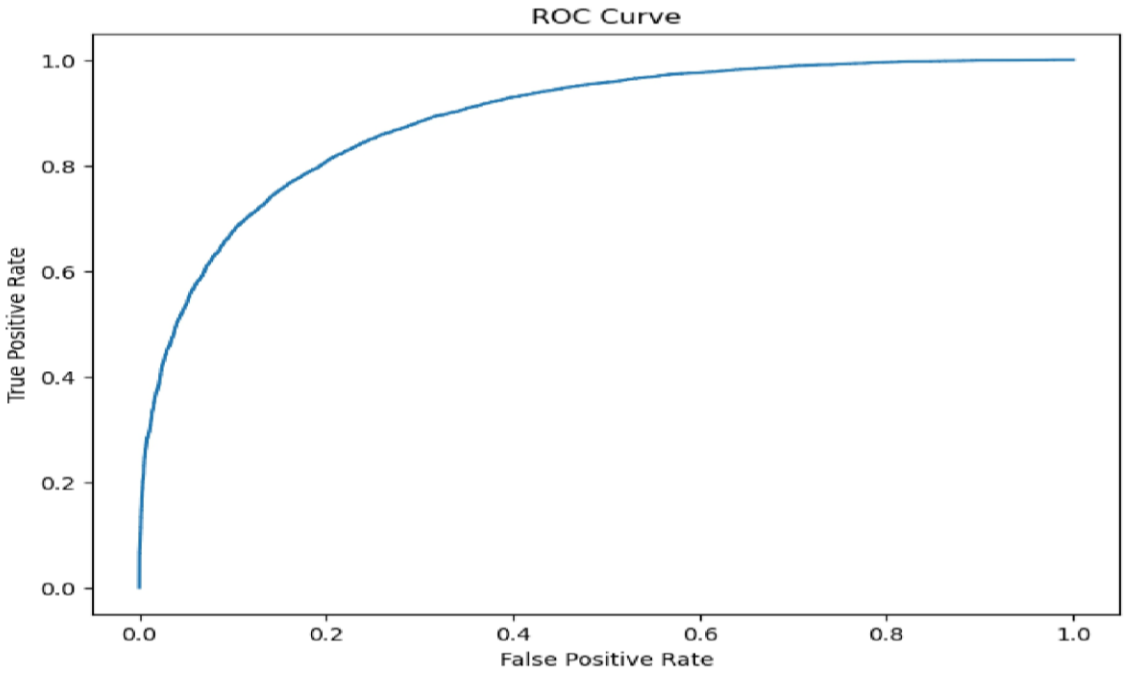
Out[74]:
```
LGBMClassifier(colsample_bytree=0.95, max_depth=16, min_split_gain=0.1,
               n_estimators=200, num_leaves=50, reg_alpha=1.2, reg_lambda
=1.2,
               subsample=0.95, subsample_freq=20)
```

In [75]:
```
mscore(model_lgb)
ypred_model_lgb = model_lgb.predict(x_test)
eval_model(y_test,ypred_model_lgb)
```

```
Train Score 0.8662998946505911
Test Score 0.8026522173107087
[[8559 1974]
 [2178 8328]]
              precision    recall  f1-score   support

           0       0.80      0.81      0.80     10533
           1       0.81      0.79      0.80     10506

    accuracy                           0.80     21039
   macro avg       0.80      0.80      0.80     21039
weighted avg       0.80      0.80      0.80     21039
```

ROC Curve for LightbGBM Model

In [76]:
```
from sklearn.metrics import roc_curve, roc_auc_score

y_pred_proba = model_lgb.predict_proba(x_test)[:, 1]
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.show()
```



In [ ]:
```
```

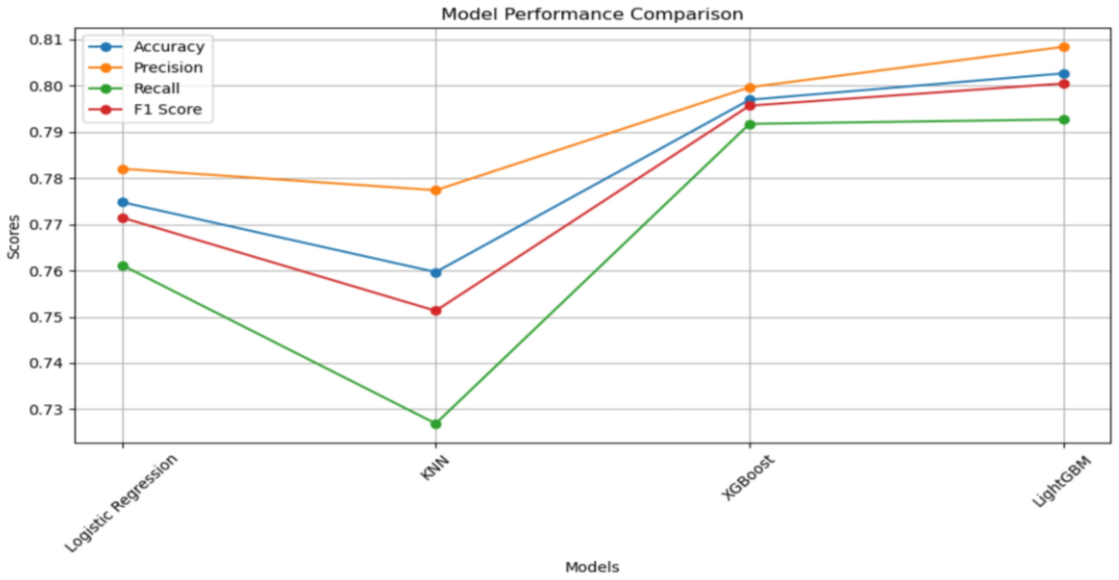
Conclusion

```
So we have applied total four models-->

Model Name | Accuracy (in %)
-----------------------------
KNN Model  | 76%
Logistic   | 77%
Regression |
XG Boost   | 80%
Light GBM  | 81%
```

The K-Nearest Neighbors (KNN) model will not be utilized due to its comparatively lower accuracy. Although the XGBoost model exhibits an impressive accuracy of 80%, its training score of 99% suggests a potential overfitting risk, thus we will refrain from using it. The remaining models, Logistic Regression and LightGBM, are both viable options. However, given that LightGBM demonstrates the highest accuracy without the risk of overfitting, we will proceed with the LightGBM model.

Comparing all models through graphical representation

In [77]:
```
import matplotlib.pyplot as plt

# Define the models and their corresponding names
models = [m1, m2, model_xgb, model_lgb]
model_names = ["Logistic Regression", "KNN", "XGBoost", "LightGBM"]

# Create empty lists to store performance metrics
accuracy_scores = []
precision_scores = []
recall_scores = []
f1_scores = []

# Calculate the performance metrics for each model
for model in models:
    y_pred = model.predict(x_test)
    cm = confusion_matrix(y_test, y_pred)
    tn, fp, fn, tp = cm.ravel()

    accuracy = (tp + tn) / (tp + tn + fp + fn)
    precision = tp / (tp + fp)
    recall = tp / (tp + fn)
    f1 = 2 * (precision * recall) / (precision + recall)

    accuracy_scores.append(accuracy)
    precision_scores.append(precision)
    recall_scores.append(recall)
    f1_scores.append(f1)

# Create line graphs for performance metrics
plt.figure(figsize=(10, 6))

plt.plot(model_names, accuracy_scores, label="Accuracy", marker='o')
plt.plot(model_names, precision_scores, label="Precision", marker='o')
plt.plot(model_names, recall_scores, label="Recall", marker='o')
plt.plot(model_names, f1_scores, label="F1 Score", marker='o')

plt.xlabel("Models")
plt.ylabel("Scores")
plt.title("Model Performance Comparison")
plt.legend()
plt.grid(True)

plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



In [78]:
```
import joblib
```

In [79]:
```
joblib.dump(model_lgb, 'model.pkl')
```

Out[79]:
```
['model.pkl']
```

In [ ]:
```

```

END

UI Coding Work (Tkinter)

```
from tkinter import *
from PIL import Image, ImageTk

root= Tk()

width= 800
height= 400
root.geometry(f"{width}x{height}")


# Importing Model
import joblib
model= joblib.load('model.pkl')

def result():
    #for locations
    loc_map = {
    'Portland': 1,
    'Sydney': 2,
    'Cairns': 3,
    'Albany': 4,
    'Darwin': 5,
    'MountGambier': 6,
    'Dartmoor': 7,
    'NorfolkIsland': 8,
    'CoffsHarbour': 9,
    'Walpole': 10,
    'Witchcliffe': 11,
    'Hobart': 12,
    'NorahHead': 13,
    'GoldCoast': 14,
    'Canberra': 15,
    'Ballarat': 16,
    'SydneyAirport': 17,
    'Brisbane': 18,
    'Adelaide': 19,
    'MountGinini': 20,
    'Launceston': 21,
    'Watsonia': 22,
    'Perth': 23,
    'Wollongong': 24,
    'Sale': 25,
    'Newcastle': 26,
    'Albury': 27,
    'BadgerysCreek': 28,
    'Nuriootpa': 29,
    'MelbourneAirport': 30,
    'Tuggeranong': 31,
    'Penrith': 32,
    'PerthAirport': 33,
    'Bendigo': 34,
    'Richmond': 35,
    'Williamtown': 36,
    'WaggaWagga': 37,
    'SalmonGums': 38,
    'Townsville': 39,
    'Cobar': 40,
    'Melbourne': 41,
    'PearceRAAF': 42,
    'Moree': 43,
    "Mildura": 44,
    "AliceSprings": 45,
    "Woomera": 46,
    "Nhil": 47,
    "Katherine": 48,
    "Uluru": 49
    }
    loc= loc_map.get(value.get())
```

```
    print(loc)

    #for month
    month_map =  {
    'January': 1,
    'February': 2,
    'March': 3,
    'April': 4,
    'May': 5,
    'June': 6,
    'July': 7,
    'August': 8,
    'September': 9,
    'October': 10,
    'November': 11,
    'December': 12
    }
    month= month_map.get(value2.get())
    print(month)

    #for Rain_today
    rtoday_map= {
        "Yes": 1,
        "No": 0
    }
    rtoday=  rtoday_map.get(value1.get())
    print(rtoday)

    #Direction
    dir_map={
    'W': 4,
    'SE': 23,
    'E': 3,
    'N': 1,
    'SSE': 223,
    'S': 2,
    'WSW': 424,
    'SW': 24,
    'SSW': 224,
    'WNW': 414,
    'NW': 14,
    'ENE': 313,
    'ESE': 323,
    'NE': 13,
    'NNW': 114,
    'NNE': 113
    }
    dir_gust= dir_map.get(v15.get())
    print(dir_gust)
    dir_9am= dir_map.get(v16.get())
    print(dir_9am)
    dir_3pm= dir_map.get(v16.get())
    print(dir_3pm)



    data= [v1.get(), v2.get(), v3.get(), v4.get(), v5.get(), v6.get(), v7.get(),
v8.get(), v9.get(), v10.get(), v11.get(), v12.get(), v13.get(), v14.get(), month,
loc, dir_gust, dir_9am, dir_3pm, rtoday]
    data = [float(item) for item in data]
    print(data)
    tomorrow_rain= model.predict([data])

    probabilities = model.predict_proba([data])
    percentage = probabilities[0][1] * 100
    percentage= '{:.2f}'.format(percentage)
    print(percentage)
```

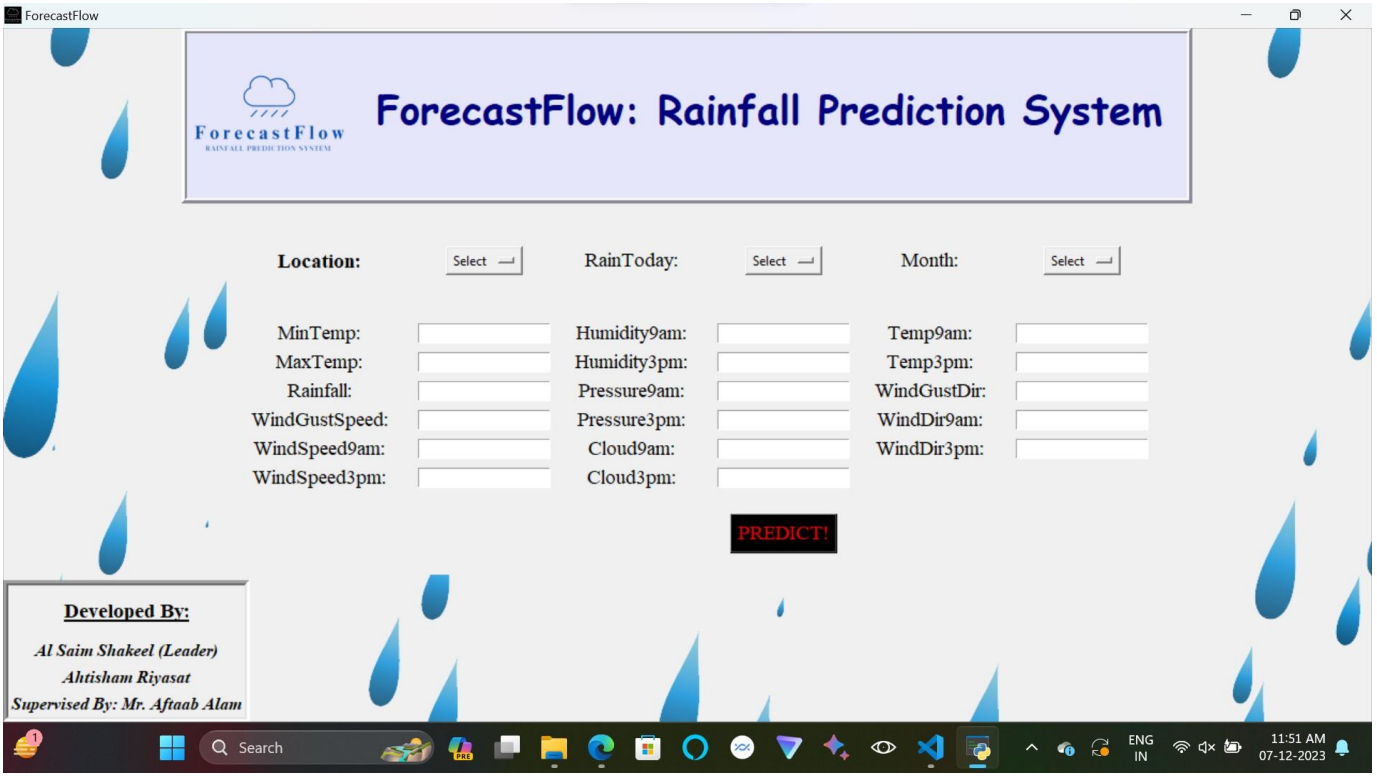
```
if tomorrow_rain == 1:
        tomorrow_rain="YES"
    elif tomorrow_rain == 0:
        tomorrow_rain="NO"
    print(tomorrow_rain)

    f3= Frame(root, borderwidth= 7, relief= SUNKEN)
    f3.pack(anchor= "s", side= BOTTOM, padx= 45)
    if tomorrow_rain == "YES":
        result= Label(f3, text= tomorrow_rain.upper(), font=("Times New Roman", 55,
"bold"), fg="green")
        result.grid(row= 4)
        result2= Label(f3, text= "Chances of Rainfall Tomorrow: "+percentage+'%',
font=("Times New Roman", 20, "bold"))
        result2.grid(row= 5)
    elif tomorrow_rain == "NO":
        result= Label(f3, text= tomorrow_rain.upper(), font=("Times New Roman", 55,
"bold"), fg="red")
        result.grid(row= 4)
        result2= Label(f3, text= "Chances of Rainfall Tomorrow: "+percentage+'%',
font=("Times New Roman", 20, "bold"))
        result2.grid(row= 5)

root.title("ForecastFlow")
root.iconbitmap('logo.ico')

# Load the background image
background_image = Image.open("rain_drops.png")
background_image = ImageTk.PhotoImage(background_image)

#Create a Label to display the background image and place it in the window
background_label = Label(root, image=background_image)
background_label.place(x=0, y=0, relwidth=1, relheight=1)

f1 = Frame(root, bg="lavender", borderwidth=5, relief=RIDGE)
f1.pack(side=TOP)

logo_open= Image.open("logo.png")
size= (150,150)
logo_resize= logo_open.resize(size)
logo= ImageTk.PhotoImage(logo_resize)
logo_png= Label(f1, image=logo, bg= "lavender")
# pack the logo label with side='left'
logo_png.pack(side='left', anchor='n', fill=X)

heading= Label(f1,text="ForecastFlow: Rainfall Prediction System",fg="Navy" ,
font=("Comic Sans MS", 28, "bold"), bg= "lavender")
# pack the heading label with side='right'
heading.pack(side='top', anchor='n', pady=42, padx=20)

f2= Frame(root)
f2.pack(anchor= "n", side=TOP)

#Location
location= Label(f2, text= "Location: ", font=("Times New Roman", 14, "bold"))
location.grid(row=0, column=1, pady= 40)

value= StringVar()
location_options= ["Select", "Adelaide", "Albany", "AliceSprings", "Albury",
"BadgerysCreek", "Ballarat", "Bendigo", "Brisbane", "Cairns", "Cobar",
"CoffsHarbour", "Dartmoor", "Darwin", "GoldCoast", "Hobart", "Katherine",
"Launceston", "Melbourne", "MelbourneAirport", "Mildura", "Moree", "MountGambier",
"MountGinini", "Newcastle", "Nhil", "NorfolkIsland", "NorahHead", "Nuriootpa",
"PearceRAAF", "Penrith", "Perth", "PerthAirport", "Portland", "Richmond", "Sale",
"SalmonGums", "Sydney", "SydneyAirport", "Townsville", "Tuggeranong", "Uluru",
"Walpole", "Wollongong", "Woomera", "Watsonia", "Williamtown", "WaggaWagga",
"Witchcliffe"]
```

```
location_entry= OptionMenu(f2, value , *location_options)
location_entry.grid(row=0, column=2)
value.set(location_options[0])

#MinTemp
MinTemp= Label(f2, text= "MinTemp: ", font=("Times New Roman", 14))
MinTemp.grid(row=2, column=1, padx= 20)
v1= StringVar()
MinTemp_v1= Entry(f2, textvariable= v1)
MinTemp_v1.grid(row=2, column=2)

#MaxTemp
MaxTemp= Label(f2, text= "MaxTemp: ", font=("Times New Roman", 14))
MaxTemp.grid(row=3, column=1, padx= 20)
v2= StringVar()
MaxTemp_v2= Entry(f2, textvariable= v2)
MaxTemp_v2.grid(row=3, column=2)

#Rainfall
Rainfall= Label(f2, text= "Rainfall: ", font=("Times New Roman", 14))
Rainfall.grid(row=4, column=1, padx= 20)
v3= StringVar()
Rainfall_v3= Entry(f2, textvariable= v3)
Rainfall_v3.grid(row=4, column=2)

#WindGustSpeed
WindGustSpeed= Label(f2, text= "WindGustSpeed: ", font=("Times New Roman", 14))
WindGustSpeed.grid(row=5, column=1, padx= 20)
v4= StringVar()
WindGustSpeedv4= Entry(f2, textvariable= v4)
WindGustSpeedv4.grid(row=5, column=2)

#WindSpeed9am
WindSpeed9am= Label(f2, text= "WindSpeed9am: ", font=("Times New Roman", 14))
WindSpeed9am.grid(row=6, column=1)
v5= StringVar()
WindSpeed9amv5= Entry(f2, textvariable= v5)
WindSpeed9amv5.grid(row=6, column=2)

#WindSpeed3pm
WindSpeed3pm= Label(f2, text= "WindSpeed3pm: ", font=("Times New Roman", 14))
WindSpeed3pm.grid(row=7, column=1)
v6= StringVar()
WindSpeed3pmv6= Entry(f2, textvariable= v6)
WindSpeed3pmv6.grid(row=7, column=2)

#Humidity9am
Humidity9am= Label(f2, text= "Humidity9am: ", font=("Times New Roman", 14))
Humidity9am.grid(row=2, column=3, padx= 20)
v7= StringVar()
Humidity9amv7= Entry(f2, textvariable= v7)
Humidity9amv7.grid(row=2, column=4)

#Humidity3pm
Humidity3pm= Label(f2, text= "Humidity3pm: ", font=("Times New Roman", 14))
Humidity3pm.grid(row=3, column=3, padx= 20)
v8= StringVar()
Humidity3pmv8= Entry(f2, textvariable= v8)
Humidity3pmv8.grid(row=3, column=4)

#Pressure9am
Pressure9am= Label(f2, text= "Pressure9am: ", font=("Times New Roman", 14))
Pressure9am.grid(row=4, column=3, padx= 20)
v9= StringVar()
Pressure9amv9= Entry(f2, textvariable= v9)
Pressure9amv9.grid(row=4, column=4)
```

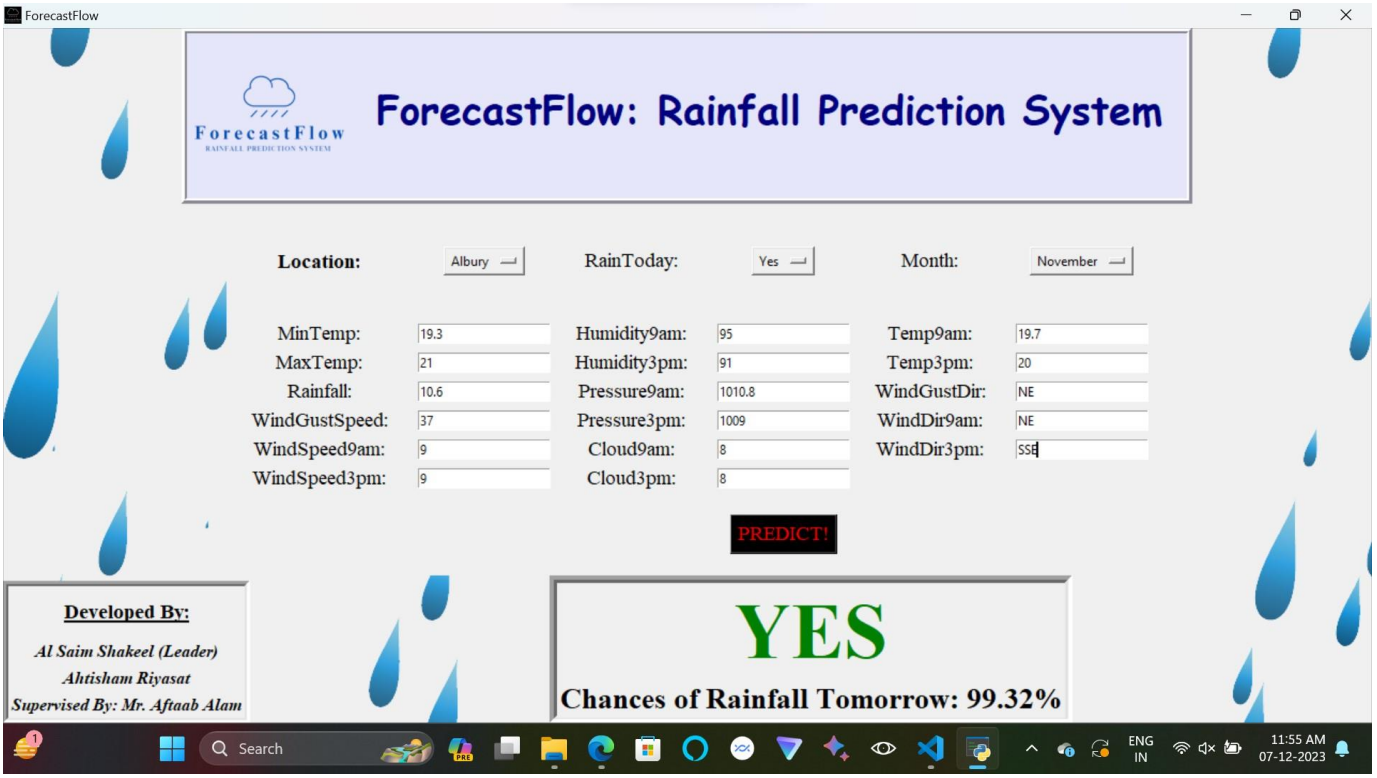
```
#Pressure3pm
Pressure3pm= Label(f2, text= "Pressure3pm: ", font=("Times New Roman", 14))
Pressure3pm.grid(row=5, column=3, padx= 20)
v10= StringVar()
Pressure3pmv10= Entry(f2, textvariable= v10)
Pressure3pmv10.grid(row=5, column=4)



#Cloud9am
Cloud9am= Label(f2, text= "Cloud9am: ", font=("Times New Roman", 14))
Cloud9am.grid(row=6, column=3, padx= 20)
v11= StringVar()
Cloud9amv11= Entry(f2, textvariable= v11)
Cloud9amv11.grid(row=6, column=4)

#Cloud3pm
Cloud3pm= Label(f2, text= "Cloud3pm: ", font=("Times New Roman", 14))
Cloud3pm.grid(row=7, column=3, padx= 20)
v12= StringVar()
Cloud3pmv12= Entry(f2, textvariable= v12)
Cloud3pmv12.grid(row=7, column=4)

#Temp9am
Temp9am= Label(f2, text= "Temp9am: ", font=("Times New Roman", 14))
Temp9am.grid(row=2, column=5, padx= 20)
v13= StringVar()
Temp9amv13= Entry(f2, textvariable= v13)
Temp9amv13.grid(row=2, column=6)

#Temp3pm
Temp3pm= Label(f2, text= "Temp3pm: ", font=("Times New Roman", 14))
Temp3pm.grid(row=3, column=5, padx= 20)
v14= StringVar()
Temp3pmv14= Entry(f2, textvariable= v14)
Temp3pmv14.grid(row=3, column=6)

#WindGustDir
WindGustDir= Label(f2, text= "WindGustDir: ", font=("Times New Roman", 14))
WindGustDir.grid(row=4, column=5, padx= 20)
v15= StringVar()
WindGustDirv15= Entry(f2, textvariable= v15)
WindGustDirv15.grid(row=4, column=6)

#WindDir9am
WindDir9am= Label(f2, text= "WindDir9am: ", font=("Times New Roman", 14))
WindDir9am.grid(row=5, column=5, padx= 20)
v16= StringVar()
WindDir9amv16= Entry(f2, textvariable= v16)
WindDir9amv16.grid(row=5, column=6)

#WindDir3pm
WindDir3pm= Label(f2, text= "WindDir3pm: ", font=("Times New Roman", 14))
WindDir3pm.grid(row=6, column=5, padx= 20)
v17= StringVar()
WindDir3pmv17= Entry(f2, textvariable= v17)
WindDir3pmv17.grid(row=6, column=6)

#RainToday
RainToday= Label(f2, text= "RainToday: ", font=("Times New Roman", 14))
RainToday.grid(row=0, column=3, pady= 20)

value1= StringVar()
RainToday_options= ["Select","Yes","No"]
RainToday_entry= OptionMenu(f2, value1 , *RainToday_options)
RainToday_entry.grid(row=0, column=4)
value1.set(RainToday_options[0])
```

```
#Month
Month= Label(f2, text= "Month: ", font=("Times New Roman", 14))
Month.grid(row=0, column=5, pady= 20)

value2= StringVar()
month_options= ["Select","January", "February", "March", "April", "May", "June",
"July", "August", "September", "October", "November", "December"]
Month_entry= OptionMenu(f2, value2 , *month_options)
Month_entry.grid(row=0, column=6)
value2.set(month_options[0])

b1= Button(f2, text= "PREDICT!", fg="red", font=("Times New Roman", 14), bg="black",
command= result)
b1.grid(row=9, column=4, pady= 20)

#credit= Label(root, text= "Project Team:\nAl Saim Shakeel\nAhtisham
riyasat\nSupervised by: Mr. Aftaab Alam")
#credit.pack(anchor='s', side= "right")
f_credit= Frame(root, borderwidth=5, relief= SUNKEN)
f_credit.pack(anchor="s", side= LEFT)
credit= Label(f_credit, text="Developed By:", font=("Times New Roman", 14, "bold",
"underline"))
credit.grid(row=3, pady= 10)
credit_n1= Label(f_credit, text="Al Saim Shakeel (Leader)", font=("Times New Roman",
12, "italic", 'bold'))
credit_n1.grid(row=4)
credit_n1= Label(f_credit, text="Ahtisham Riyasat", font=("Times New Roman", 12,
"italic", 'bold'))
credit_n1.grid(row=5)
credit_s= Label(f_credit, text="Supervised By: Mr. Aftaab Alam", font=("Times New
Roman", 12, "italic", 'bold'))
credit_s.grid(row=6)


root.mainloop()
```

User Interface with Tkinter



Result:



CONCLUSION

In conclusion, the development and implementation of the Rainfall Prediction System have addressed the critical need for accurate next-day rainfall forecasts and associated probabilities tailored to the diverse climate of Australia. The project involved a comprehensive approach, integrating advanced predictive modeling techniques, data-driven insights, and user-centric design principles.

The ensemble of Logistic Regression, K-Nearest Neighbors, XGBoost, and Light GBM demonstrated robust performance, providing reliable predictions that can significantly impact decision-making in sectors such as agriculture, water resource management, and disaster preparedness. The user-friendly interface enhances accessibility, allowing stakeholders to interpret and utilize the predictions effectively.

As we reflect on the project, it is evident that the Rainfall Prediction System stands as a valuable tool in mitigating the challenges posed by uncertain weather patterns. The successful deployment and real-time functionality of the system underscore its potential to contribute meaningfully to various industries.

However, the journey does not end here. Continuous refinement, adaptation to evolving weather conditions, and the incorporation of user feedback remain essential aspects of maintaining the system's effectiveness. The outlined future work provides a roadmap for enhancements, ensuring the system remains at the forefront of accurate and timely weather predictions in the dynamic context of Australia.

The Rainfall Prediction System signifies a significant step toward harnessing the power of data and advanced modeling for informed decision-making, contributing to the resilience of various sectors in the face of changing weather patterns. This project represents not only a technological achievement but a commitment to addressing real-world challenges and advancing our understanding of predictive analytics in meteorological sciences.

REFERENCES & BIBLIOGRAPHY

1. SkillForge: https://skillforge.in/
2. Kaggle: https://www.kaggle.com/
3. Python: https://docs.python.org/3/
4. Geeks for geeks: https://www.geeksforgeeks.org/courses?source=google&medium=cpc&device=c&keyword=geeksforgeeks&matchtype=e&campaignid=20039445781&adgroup=147845288105&gad_source=1&gclid=CjwKCAiA98WrBhAYEiwA2WvhOk2dt4Kp4796viGtf6FYL4msHlLXpWhm9gf2hFGLuvR6W7ZXHL4rFxoCw5QQAvD_BwE
5. Javatpoint: https://www.javatpoint.com/
6. W3Schools: https://www.w3schools.com/
7. Anaconda: https://www.anaconda.com/
8. Tkinter: https://docs.python.org/3/library/tkinter.html
9. OpenMl: https://www.openml.org/
10. Deeplearning.ai: https://community.deeplearning.ai/login