PROFESSIONAL TRAINING REPORT
entitled

RAINFALL PREDICTION USING MACHINE LEARNING TECHNIQUES

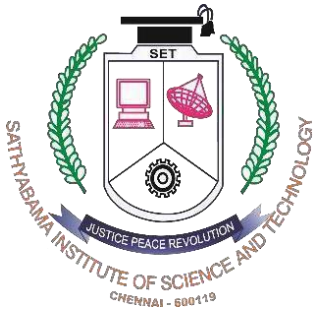
Submitted in partial fulfillment of the requirements for the reward of

Bachelor of Engineering degree in Computer Science and Engineering

In Artificial Intelligence and Machine Learning

By

INDUKURI.H.P.RAKESH VARMA (41611066)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHISALAI,
CHENNAI – 600119

I

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(Established under Section 3 of UGC Act, 1956)
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai - 600119
www.sathyabama.ac.in

SCHOOL OF COMPUTING

BONAFIDE CERTIFICATE

This is to certify that this Professional Training Report is the bonafide work of **INDUKURI H.P.RAKESH VARMA(41611066).** who underwent the professional training in **" RAINFALL PREDICTION USING MACHINE LEARNING TECNIQUES"** under our supervision from June 2023 to October 2023.

Internal Guide

DR. MINU SUSAN JACOB

Head of the Department

Dr.S.VIGNESHWARI.,M.E.,(Ph.D)

Submitted for Viva voce Examination held on_____

Internal Examiner **External Examiner**

DECLARATION

INDUKURI H.P.RAKESH VARMA hereby declare that the Professional Training Report on "**RAINFALL PREDICTION USING MACHINE LEARNING TECHNIQUES"** done by me under the guidance of **DR. MINU SUSAN JACOB** at Sathyabama university is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in computer science and engineering.

DATE:

PLACE: **SIGNATURE OF THE CANDIDATE**

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks Board of Management of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr .T. SASIKALA.,M.E.,Ph.D., Dean, School of Computing** and **Dr.S.VIGNESHWARI.,M.E.,(Ph.D) Head of the Department, Dept of Computer Science and Engineering** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **DR. MINU SUSAN JACOB** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-Teaching staff members of the Department of Computer science and Engineering who were helpful in many ways for the completion of the project.

SAMPLE COURSE CERTIFICATE

ABSTRACT

TITLE: RAINFALL PREDICTION

Rainfall prediction is important as heavy rainfall can lead to many disasters. The prediction helps people to take preventive measures and moreover the prediction should be accurate. There are two types of prediction short term rainfall prediction and long term rainfall. Prediction mostly short term prediction can gives us the accurate result. The main challenge is to build a model for long term rainfall prediction. Heavy precipitation prediction could be a major drawback for earth science department because it is closely associated with the economy and lifetime of human. It's a cause for natural disasters like flood and drought that square measure encountered by individuals across the world each year. Accuracy of rainfall statement has nice importance for countries like India whose economy is basically dependent on agriculture. The dynamic nature of atmosphere, applied mathematics techniques fail to provide sensible accuracy for precipitation statement. The prediction of precipitation using machine learning techniques may use regression. Intention of this project is to offer non-experts easy access to the techniques, approaches utilized in the sector of precipitation prediction and provide a comparative study among the various machine learning techniques

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION

BACKGROUND AND BASICS

Rainfall Prediction is one of the most challenging tasks. Though already many algorithms have being proposed but still accurate prediction of rainfall is very difficult. In an agricultural country like India, the success or failure of the crops and water scarcity in any year is always viewed with greatest concern. A small fluctuation in the seasonal rainfall can have devastating impacts on agriculture sector. Accurate rainfall prediction has a potential benefit of preventing causalities and damages caused by natural disasters. Under certain circumstances such as flood and drought, highly accurate rainfall prediction is useful for agriculture management and disaster prevention. In this paper, various algorithms have been analyzed. Data mining techniques are efficiently used in rainfall prediction.

EXISTING SYSTEM

Agriculture is the strength of our Indian economy. Farmer only depends upon monsoon to be their cultivation.. Weather forecasting is the very important requirement of the each farmer. Due to the sudden changes in climate/weather, The people are suffered economically and physically. Weather prediction is one of the challenging problems in current state. The main motivation of this paper to predict the weather using various data mining techniques. Such as classification, clustering, decision tree and also neural networks. Weather related information is also called the meteorological data. In this paper the most commonly used weather parameters are rainfall, wind speed, temperature and cold.

DISADVANTAGES OF EXISTING SYSTEM

1.Classification

2.Clustering

3.Decision Tree

PROPOSED SYSTEM

Rainfall is important for food production plan, water resource management and all activity plans in the nature. The occurrence of prolonged dry period or heavy rain at the critical stages of the crop growth and development may lead to significant reduce crop yield. India is an agricultural country and its economy is largely based upon crop productivity. Thus rainfall prediction becomes a significant factor in agricultural countries like India. Rainfall forecasting has been one of the most scientifically and technologically challenging problems around the world in the last century.

ADVANTAGES OF PROPOSED SYSTEM

1.Numerical Weather Pediction

2.Statistical Weather Prediction

3.Synoptic Weather Prediction

CHAPTER 2
METHODOLOGY

System Requirement Specification

HARDWARE REQUIREMENTS:

System - Windows10/11
Speed - 2.4GHZ
Hard disk - 40GB
Monitor - 15VGA Color
Ram - 4GB

SOFTWARE REQUIREMENTS:

Coding Language - PYTHON
IDE - GOOGLE COLABORATORY

SOFTWARE ENVIRONMENT

Python:

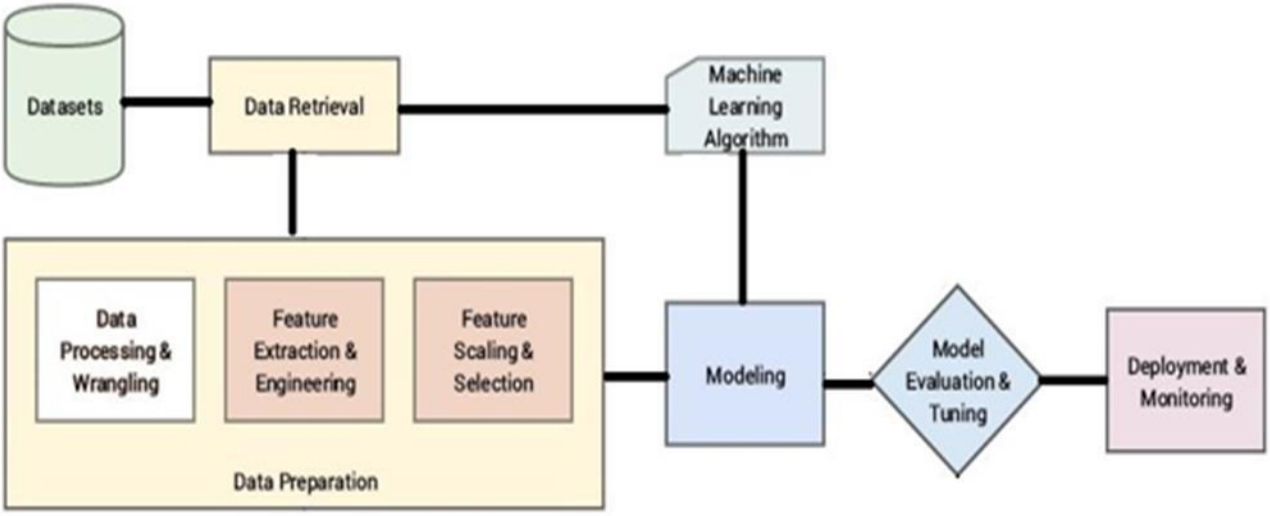
Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses Eng10/11lish keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

ARCHITECTURE



ARCHITETURE FIGURE

MODULES

- Data Collection
- Data Cleaning
- Data Selection
- Data Transformation
- Data Mining Stage

Data Collection

The data used for this work was collected from meteorologist‟s centre. The case data covered the period of 2012 to 2015. The following procedures were adopted at this stage of the research: Data Cleaning, Data Selection, Data Transformation and Data Mining.

Data Cleaning

In this stage, a consistent format for the data model was developed which is search missing data, finding duplicated data, and weeding out of bad data. Finally system cleaned data were transformed into a format suitable for data mining.

Data Selection

At this stage, data relevant to the analysis like decision tree was decided on and retrieved from the dataset. The Meteorological dataset had ten attributes in that were using two attributes for future prediction.

Data Transformation

“This is also known as data consolidation”. It is the stage in which the selected data is transformed into forms appropriate for data mining. The data file was saved in Commas Separated Value (CVS) file format and the datasets were normalized to reduce the effect of scaling on the data.

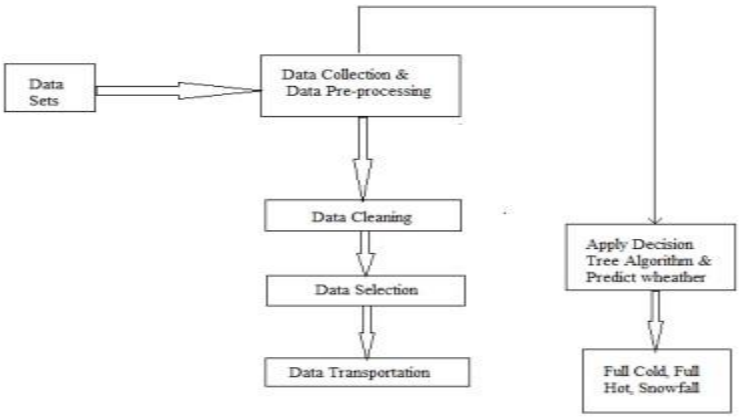
Data Mining Stage

The data mining stage was divided into three phases. At each phase all the algorithms were used to analyse the meteorological datasets. The testing method adopted for this research was percentage split that train on a percentage of the dataset, cross validate on it and test on the remaining percentage. There after interesting patterns representing knowledge were identified.

DATA FLOW DIAGRAM

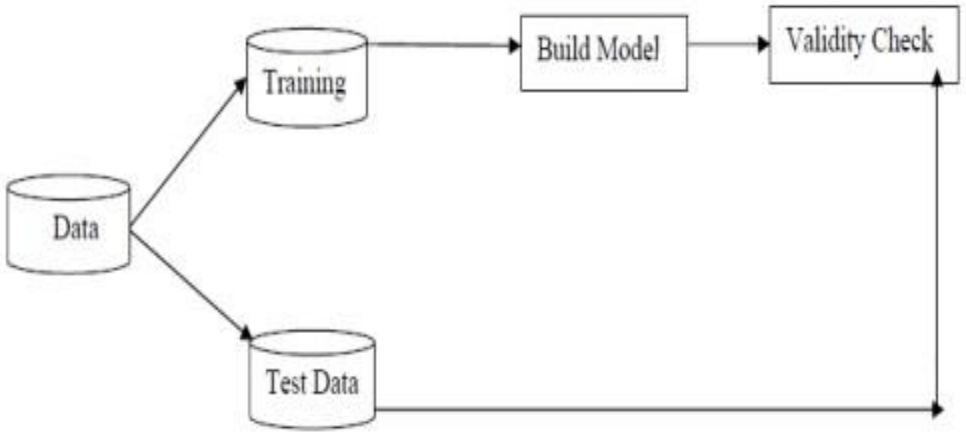
1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

LEVEL - 0



DATAFLOW DIAGRAM 1

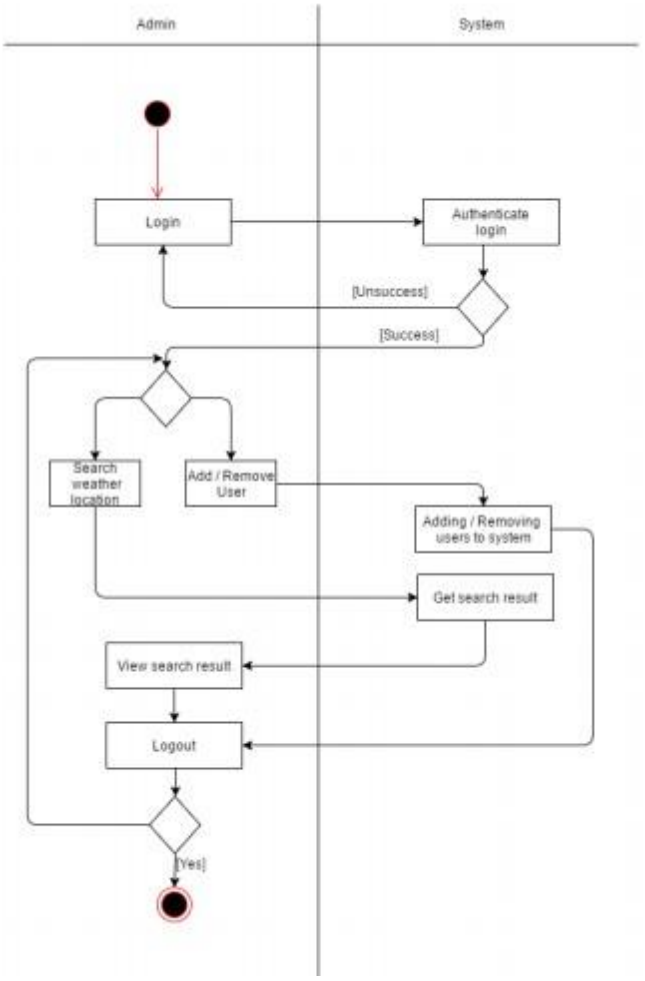
LEVEL - 1



SEQUENCE DIAGRAM

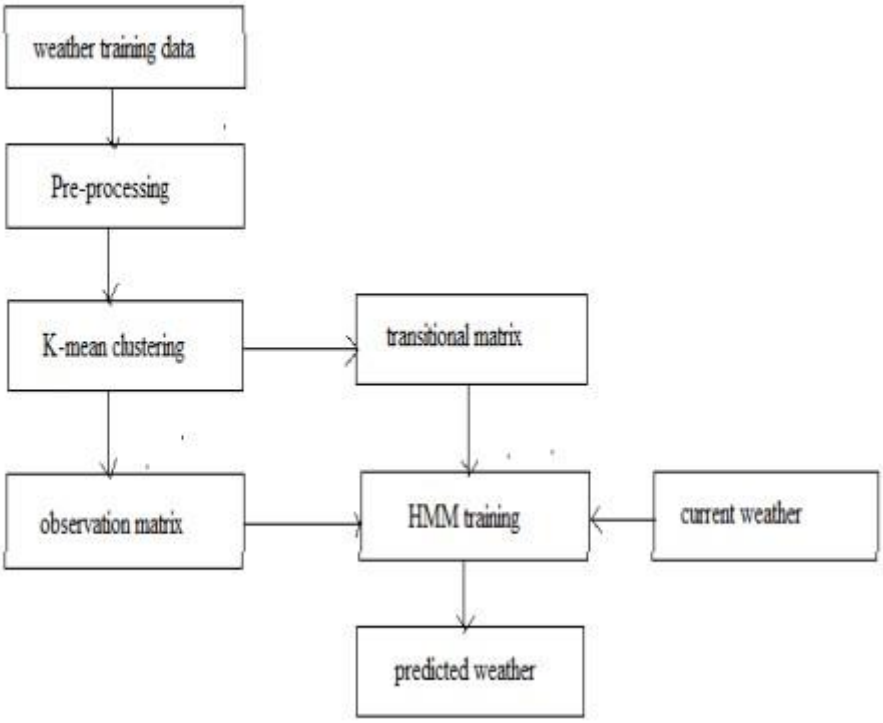
ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-bystep workflows of components in a system. An activity diagram shows the overall flow of control.



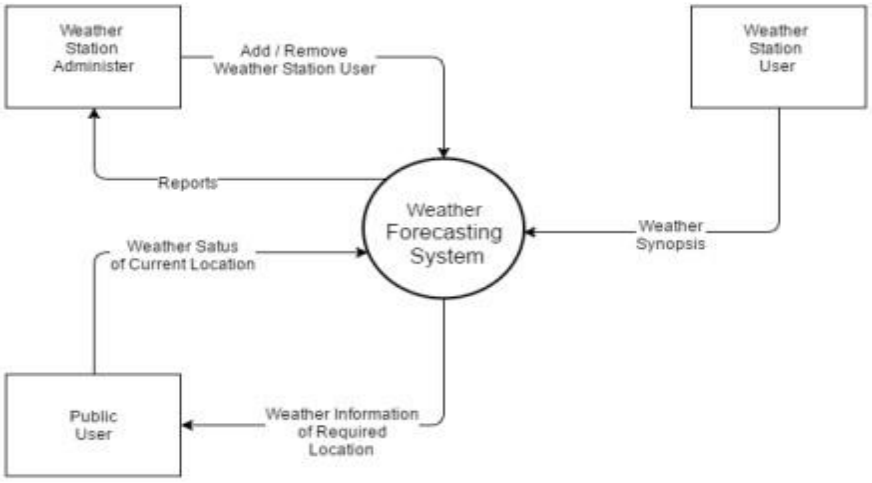
DATAFLOW DIAGRAM 2

LEVEL - 2



DATAFLOW DIAGRAM 3

LEVEL – 3



DATAFLOW DIAGRAM 4

UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

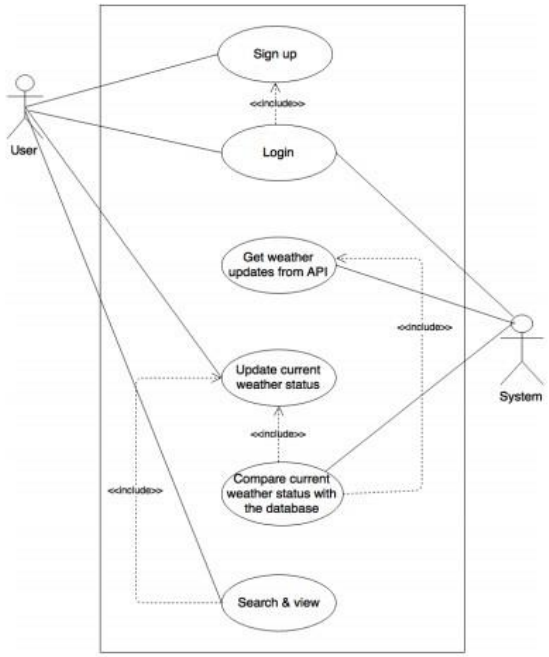
GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

USE CASE DIAGRAM:

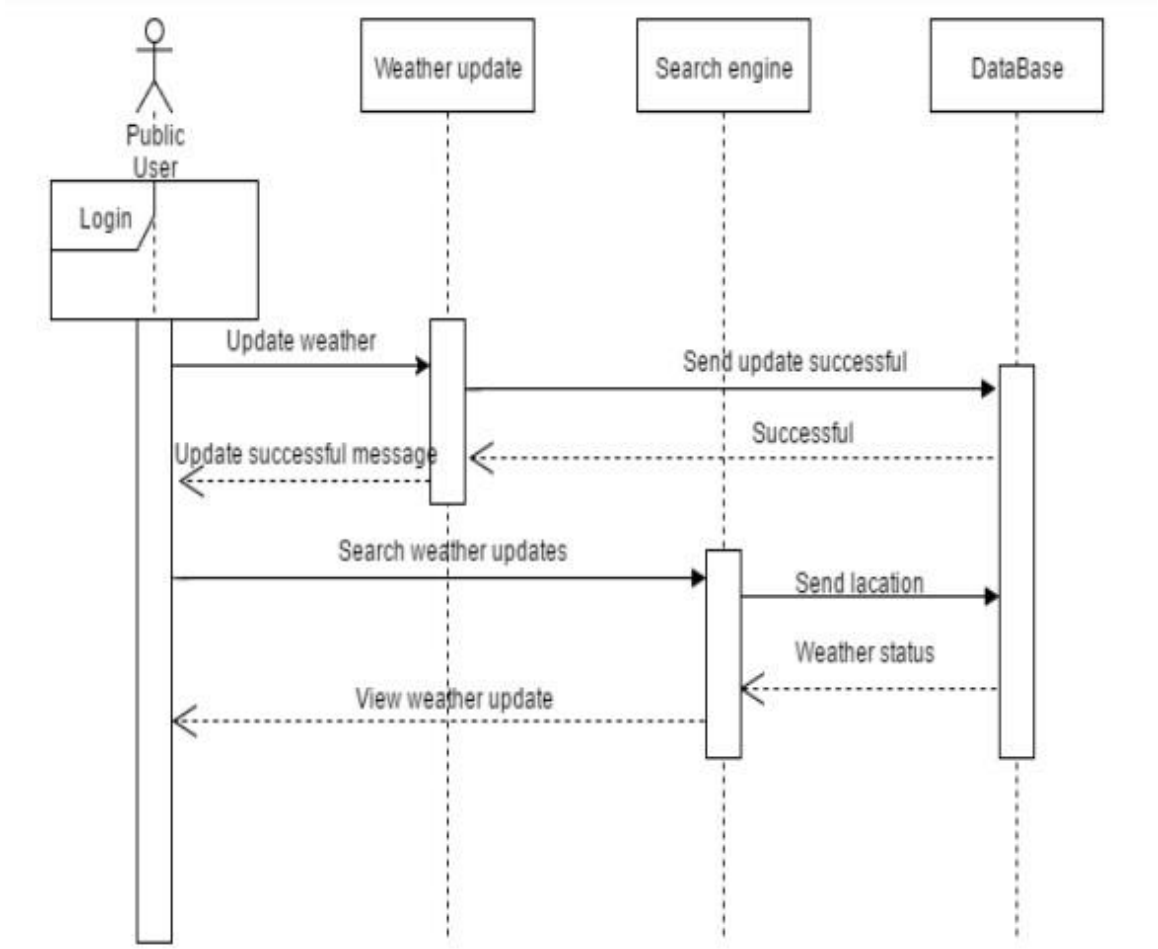
A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



USECASE DIAGRAM

SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



ACTIVITY DIAGRAM

PREDICTION

Prediction is nothing but the output of an algorithm after being trained on a dataset and applied to new data and predicts the output. Finally our model will predict the rainfall price based on user inputs.

+ Code + Text

```
[ ] import pandas as pd
    full_data = pd.read_csv('Dataset2.csv')
    full_data.head()
```

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	...	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	Cloud3
0	39783	Delhi	13.4	22.9	0.6	NaN	NaN	W	44.0	W	...	71.0	22.0	1007.7	1007.1	8.0	N
1	39784	Delhi	7.4	25.1	0.0	NaN	NaN	WNW	44.0	NNW	...	44.0	25.0	1010.6	1007.8	NaN	N
2	39785	Delhi	12.9	25.7	0.0	NaN	NaN	WSW	46.0	W	...	38.0	30.0	1007.6	1008.7	NaN	
3	39786	Delhi	9.2	28.0	0.0	NaN	NaN	NE	24.0	SE	...	45.0	16.0	1017.6	1012.8	NaN	N
4	39787	Delhi	17.5	32.3	1.0	NaN	NaN	W	41.0	ENE	...	82.0	33.0	1010.8	1006.0	7.0	

5 rows × 23 columns

Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm	Temp9am	Temp3pm	RainToday	RainTomorrow
22.0	1007.7	1007.1	8.0	NaN	16.9	21.8	No	No
25.0	1010.6	1007.8	NaN	NaN	17.2	24.3	No	No
30.0	1007.6	1008.7	NaN	2.0	21.0	23.2	No	No
16.0	1017.6	1012.8	NaN	NaN	18.1	26.5	No	No
33.0	1010.8	1006.0	7.0	8.0	17.8	29.7	No	No

```
full_data.shape
```

```
(145460, 23)
```

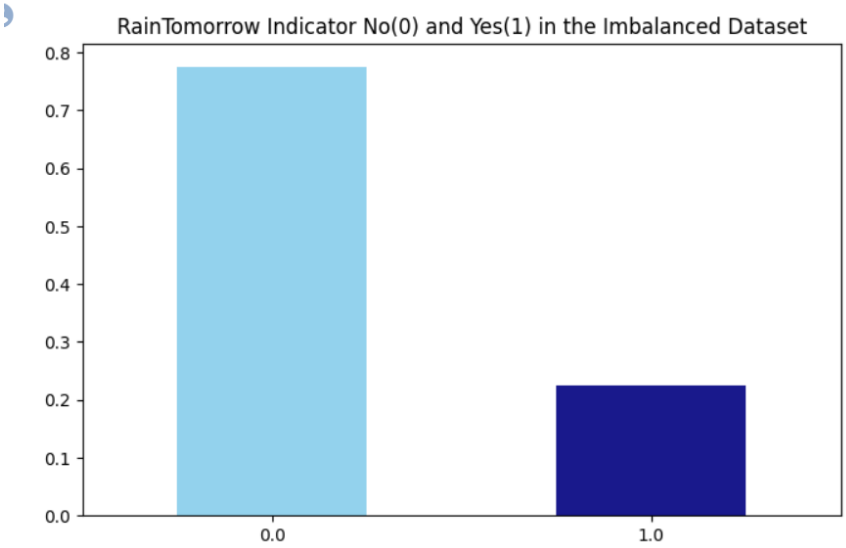
```
full_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 145460 entries, 0 to 145459
Data columns (total 23 columns):
 #   Column         Non-Null Count   Dtype  
---  ------         --------------   -----  
 0   Date           145460 non-null  int64  
 1   Location       145460 non-null  object 
 2   MinTemp        143975 non-null  float64
 3   MaxTemp        144199 non-null  float64
 4   Rainfall       142199 non-null  float64
 5   Evaporation    82670 non-null   float64
 6   Sunshine       75625 non-null   float64
 7   WindGustDir    135134 non-null  object 
 8   WindGustSpeed  135197 non-null  float64
 9   WindDir9am     134894 non-null  object 
 10  WindDir3pm     141232 non-null  object 
 11  WindSpeed9am   143693 non-null  float64
 12  WindSpeed3pm   142398 non-null  float64
 13  Humidity9am    142806 non-null  float64
 14  Humidity3pm    140953 non-null  float64
 15  Pressure9am    130395 non-null  float64
 16  Pressure3pm    130432 non-null  float64
 17  Cloud9am       89572 non-null   float64
 18  Cloud3pm       86102 non-null   float64
 19  Temp9am        143693 non-null  float64
 20  Temp3pm        141851 non-null  float64
 21  RainToday      142199 non-null  object 
 22  RainTomorrow   142193 non-null  object 
dtypes: float64(16), int64(1), object(6)
memory usage: 25.5+ MB
```

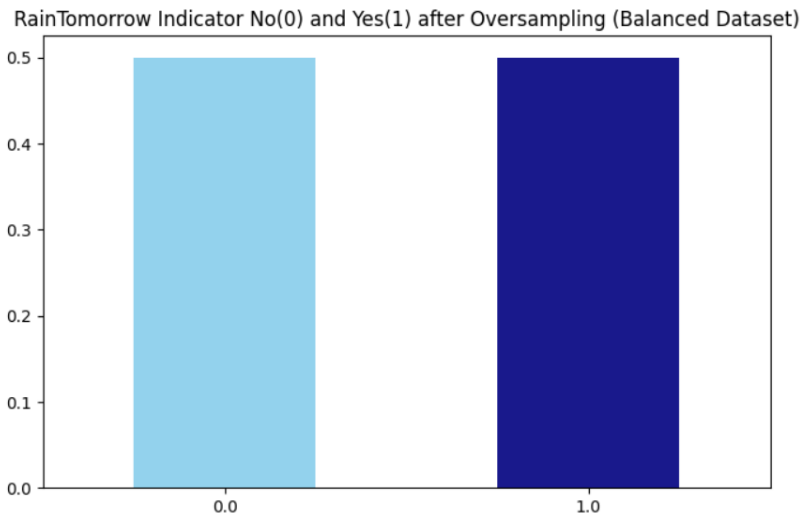
```
[ ] full_data['RainToday'].replace({'No': 0, 'Yes': 1},inplace = True)
    full_data['RainTomorrow'].replace({'No': 0, 'Yes': 1},inplace = True)
```

```
[ ] import matplotlib.pyplot as plt
    fig = plt.figure(figsize = (8,5))
    full_data.RainTomorrow.value_counts(normalize = True).plot(kind='bar', color= ['skyblue','navy'], alpha = 0.9, rot=0)
    plt.title('RainTomorrow Indicator No(0) and Yes(1) in the Imbalanced Dataset')
    plt.show()
```

```
import matplotlib.pyplot as plt
fig = plt.figure(figsize = (8,5))
full_data.RainTomorrow.value_counts(normalize = True).plot(kind='bar', color= ['skyblue','navy'], alpha = 0.9, rot=0)
plt.title('RainTomorrow Indicator No(0) and Yes(1) in the Imbalanced Dataset')
plt.show()
```

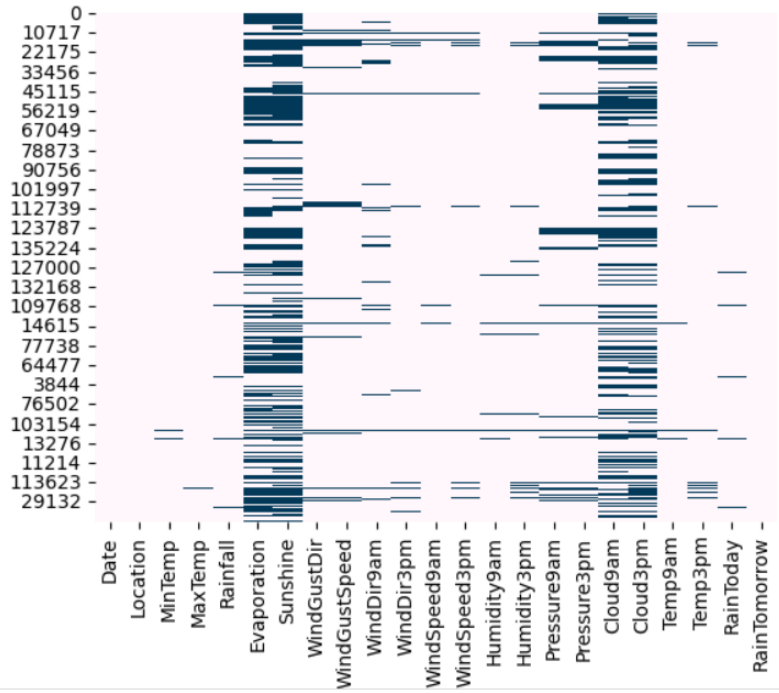



```
[ ] from sklearn.utils import resample

    no = full_data[full_data.RainTomorrow == 0]
    yes = full_data[full_data.RainTomorrow == 1]
    yes_oversampled = resample(yes, replace=True, n_samples=len(no), random_state=123)
    oversampled = pd.concat([no, yes_oversampled])

    fig = plt.figure(figsize = (8,5))
    oversampled.RainTomorrow.value_counts(normalize = True).plot(kind='bar', color= ['skyblue','navy'], alpha = 0.9, rot=0)
    plt.title('RainTomorrow Indicator No(0) and Yes(1) after Oversampling (Balanced Dataset)')
    plt.show()
```



```
[ ] import seaborn as sns
    sns.heatmap(oversampled.isnull(), cbar=False, cmap='PuBu')
```

```
<Axes: >
```




```
total = oversampled.isnull().sum().sort_values(ascending=False)
percent = (oversampled.isnull().sum()/oversampled.isnull().count()).sort_values(ascending=False)
missing = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
missing.head(4)
```

	Total	Percent
Sunshine	104831	0.475140
Evaporation	95411	0.432444
Cloud3pm	85614	0.388040
Cloud9am	81339	0.368664

```
[ ] oversampled.select_dtypes(include=['object']).columns
```

```
Index(['Location', 'WindGustDir', 'WindDir9am', 'WindDir3pm'], dtype='object')
```

```
[ ] # Impute categorical var with Mode
    oversampled['Date'] = oversampled['Date'].fillna(oversampled['Date'].mode()[0])
    oversampled['Location'] = oversampled['Location'].fillna(oversampled['Location'].mode()[0])
    oversampled['WindGustDir'] = oversampled['WindGustDir'].fillna(oversampled['WindGustDir'].mode()[0])
    oversampled['WindDir9am'] = oversampled['WindDir9am'].fillna(oversampled['WindDir9am'].mode()[0])
    oversampled['WindDir3pm'] = oversampled['WindDir3pm'].fillna(oversampled['WindDir3pm'].mode()[0])
```

```
# Convert categorical features to continuous features with Label Encoding
from sklearn.preprocessing import LabelEncoder
lencoders = {}
for col in oversampled.select_dtypes(include=['object']).columns:
    lencoders[col] = LabelEncoder()
    oversampled[col] = lencoders[col].fit_transform(oversampled[col])
```

```
import warnings
warnings.filterwarnings("ignore")
```

```
# Multiple Imputation by Chained Equations
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
MiceImputed = oversampled.copy(deep=True)
mice_imputer = IterativeImputer()
MiceImputed.iloc[:, :] = mice_imputer.fit_transform(oversampled)
```

```
MiceImputed.head()
```

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	...	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	Clo
0	39783.0	14.0	13.4	22.9	0.6	5.998853	7.370055	13.0	44.0	13.0	...	71.0	22.0	1007.7	1007.1	8.000000	4.8
1	39784.0	14.0	7.4	25.1	0.0	5.890191	11.275041	14.0	44.0	6.0	...	44.0	25.0	1010.6	1007.8	1.894968	2.7
2	39785.0	14.0	12.9	25.7	0.0	8.015809	12.271518	15.0	46.0	13.0	...	38.0	30.0	1007.6	1008.7	1.806505	2.0
3	39786.0	14.0	9.2	28.0	0.0	6.227639	11.554111	4.0	24.0	9.0	...	45.0	16.0	1017.6	1012.8	1.407042	2.2
4	39787.0	14.0	17.5	32.3	1.0	7.119435	5.641643	13.0	41.0	1.0	...	82.0	33.0	1010.8	1006.0	7.000000	8.0

5 rows × 23 columns

:y9am	Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm	Temp9am	Temp3pm	RainToday	RainTomorrow
71.0	22.0	1007.7	1007.1	8.000000	4.830339	16.9	21.8	0.0	0.0
44.0	25.0	1010.6	1007.8	1.894968	2.779984	17.2	24.3	0.0	0.0
38.0	30.0	1007.6	1008.7	1.806505	2.000000	21.0	23.2	0.0	0.0
45.0	16.0	1017.6	1012.8	1.407042	2.282628	18.1	26.5	0.0	0.0
82.0	33.0	1010.8	1006.0	7.000000	8.000000	17.8	29.7	0.0	0.0

```
MiceImputed.isna()
```

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	...	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	C
0	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
133585	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	
117307	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	
87307	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	
56427	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	
27220	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	

220632 rows × 23 columns

```
MiceImputed.isna()
```

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	...	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	C
0	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
133585	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	
117307	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	
87307	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	
56427	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	
27220	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	

220632 rows × 23 columns

midity9am	Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm	Temp9am	Temp3pm	RainToday	RainTomorrow
False	False	False	False	False	False	False	False	False	False
False	False	False	False	False	False	False	False	False	False
False	False	False	False	False	False	False	False	False	False
False	False	False	False	False	False	False	False	False	False
False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...
False	False	False	False	False	False	False	False	False	False
False	False	False	False	False	False	False	False	False	False
False	False	False	False	False	False	False	False	False	False
False	False	False	False	False	False	False	False	False	False
False	False	False	False	False	False	False	False	False	False

```
[ ] # Detecting outliers with IQR
    Q1 = MiceImputed.quantile(0.25)
    Q3 = MiceImputed.quantile(0.75)
    IQR = Q3 - Q1
    print(IQR)
```

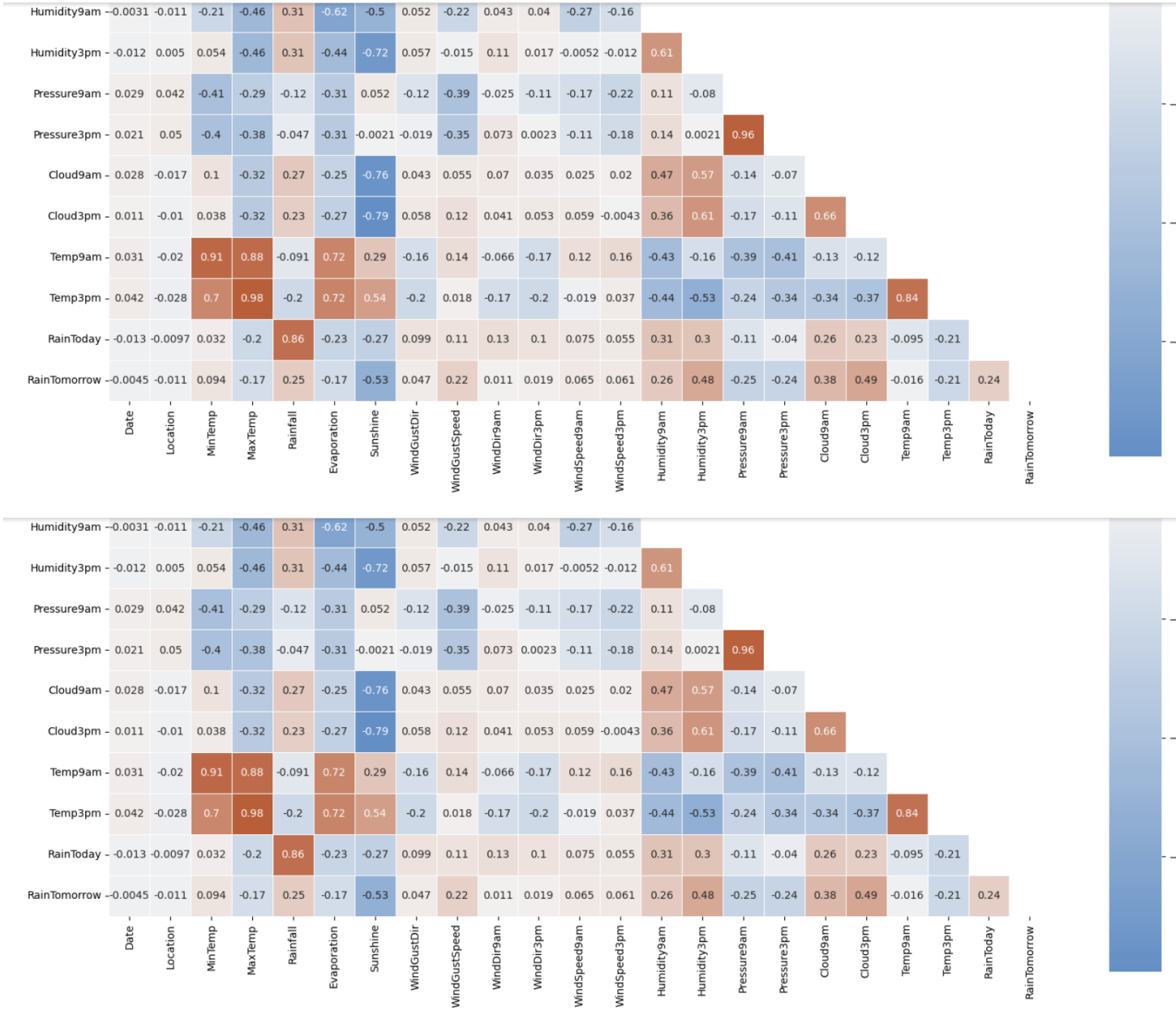
```
Date             1624.000000
Location           26.000000
MinTemp             9.300000
MaxTemp            10.200000
Rainfall            2.400000
Evaporation         4.114554
Sunshine            5.842051
WindGustDir         9.000000
WindGustSpeed      19.000000
WindDir9am          8.000000
WindDir3pm          8.000000
WindSpeed9am       13.000000
WindSpeed3pm       11.000000
Humidity9am        26.000000
Humidity3pm        30.000000
Pressure9am         8.500000
Pressure3pm         8.600000
Cloud9am            4.000000
Cloud3pm            3.583194
Temp9am             9.300000
Temp3pm             9.800000
RainToday           1.000000
RainTomorrow        1.000000
dtype: float64
```

```
[ ] # Removing outliers from dataset
    MiceImputed = MiceImputed[~((MiceImputed < (Q1 - 1.5 * IQR)) |(MiceImputed > (Q3 + 1.5 * IQR))).any(axis=1)]
    MiceImputed.shape
```

```
(170183, 23)
```

```
[ ] # Correlation Heatmap
    import numpy as np
    import matplotlib.pyplot as plt
    import seaborn as sns
    corr = MiceImputed.corr()
    mask = np.triu(np.ones_like(corr, dtype=np.bool))
    f, ax = plt.subplots(figsize=(20, 20))
    cmap = sns.diverging_palette(250, 25, as_cmap=True)
    sns.heatmap(corr, mask=mask, cmap=cmap, vmax=None, center=0,square=True, annot=True, linewidths=.5, cbar_kws={"shrink": .9})
```

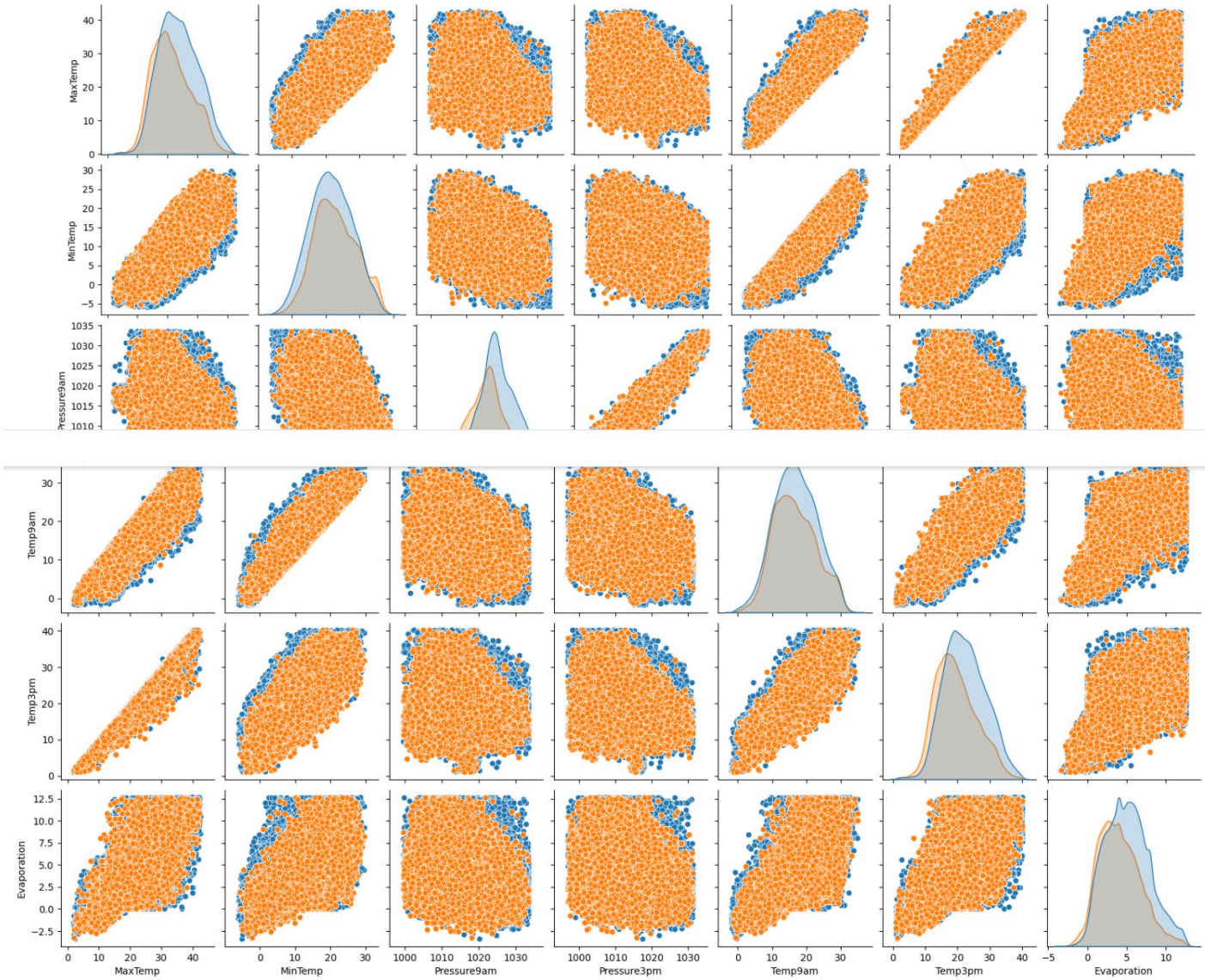
```
<Axes: >
```


Humidity9am -0.0031 -0.011 -0.21 -0.46 0.31 -0.62 -0.5 0.052 -0.22 0.043 0.04 -0.27 -0.16
Humidity3pm -0.012 0.005 0.054 -0.46 0.31 -0.44 -0.72 0.057 -0.015 0.11 0.017 -0.0052 -0.012 0.61
Pressure9am 0.029 0.042 -0.41 -0.29 -0.12 -0.31 0.052 -0.12 -0.39 -0.025 -0.11 -0.17 -0.22 0.11 -0.08
Pressure3pm 0.021 0.05 -0.4 -0.38 -0.047 -0.31 -0.0021 -0.019 -0.35 0.073 0.0023 -0.11 -0.18 0.14 0.0021 0.96
Cloud9am 0.028 -0.017 0.1 -0.32 0.27 -0.25 -0.76 0.043 0.055 0.07 0.035 0.025 0.02 0.47 0.57 -0.14 -0.07
Cloud3pm 0.011 -0.01 0.038 -0.32 0.23 -0.27 -0.79 0.058 0.12 0.041 0.053 0.059 -0.0043 0.36 0.61 -0.17 -0.11 0.66
Temp9am 0.031 -0.02 0.91 0.88 -0.091 0.72 0.29 -0.16 0.14 -0.066 -0.17 0.12 0.16 -0.43 -0.16 -0.39 -0.41 -0.13 -0.12
Temp3pm 0.042 -0.028 0.7 0.98 -0.2 0.72 0.54 -0.2 0.018 -0.17 -0.2 -0.019 0.037 -0.44 -0.53 -0.24 -0.34 -0.34 -0.37 0.84
RainToday -0.013 -0.0097 0.032 -0.2 0.86 -0.23 -0.27 0.099 0.11 0.13 0.1 0.075 0.055 0.31 0.3 -0.11 -0.04 0.26 0.23 -0.095 -0.21
RainTomorrow -0.0045 -0.011 0.094 -0.17 0.25 -0.17 -0.53 0.047 0.22 0.011 0.019 0.065 0.061 0.26 0.48 -0.25 -0.24 0.38 0.49 -0.016 -0.21 0.24
Date
Location
MinTemp
MaxTemp
Rainfall
Evaporation
Sunshine
WindGustDir
WindGustSpeed
WindDir9am
WindDir3pm
WindSpeed9am
WindSpeed3pm
Humidity9am
Humidity3pm
Pressure9am
Pressure3pm
Cloud9am
Cloud3pm
Temp9am
Temp3pm
RainToday
RainTomorrow

Humidity9am -0.0031 -0.011 -0.21 -0.46 0.31 -0.62 -0.5 0.052 -0.22 0.043 0.04 -0.27 -0.16
Humidity3pm -0.012 0.005 0.054 -0.46 0.31 -0.44 -0.72 0.057 -0.015 0.11 0.017 -0.0052 -0.012 0.61
Pressure9am 0.029 0.042 -0.41 -0.29 -0.12 -0.31 0.052 -0.12 -0.39 -0.025 -0.11 -0.17 -0.22 0.11 -0.08
Pressure3pm 0.021 0.05 -0.4 -0.38 -0.047 -0.31 -0.0021 -0.019 -0.35 0.073 0.0023 -0.11 -0.18 0.14 0.0021 0.96
Cloud9am 0.028 -0.017 0.1 -0.32 0.27 -0.25 -0.76 0.043 0.055 0.07 0.035 0.025 0.02 0.47 0.57 -0.14 -0.07
Cloud3pm 0.011 -0.01 0.038 -0.32 0.23 -0.27 -0.79 0.058 0.12 0.041 0.053 0.059 -0.0043 0.36 0.61 -0.17 -0.11 0.66
Temp9am 0.031 -0.02 0.91 0.88 -0.091 0.72 0.29 -0.16 0.14 -0.066 -0.17 0.12 0.16 -0.43 -0.16 -0.39 -0.41 -0.13 -0.12
Temp3pm 0.042 -0.028 0.7 0.98 -0.2 0.72 0.54 -0.2 0.018 -0.17 -0.2 -0.019 0.037 -0.44 -0.53 -0.24 -0.34 -0.34 -0.37 0.84
RainToday -0.013 -0.0097 0.032 -0.2 0.86 -0.23 -0.27 0.099 0.11 0.13 0.1 0.075 0.055 0.31 0.3 -0.11 -0.04 0.26 0.23 -0.095 -0.21
RainTomorrow -0.0045 -0.011 0.094 -0.17 0.25 -0.17 -0.53 0.047 0.22 0.011 0.019 0.065 0.061 0.26 0.48 -0.25 -0.24 0.38 0.49 -0.016 -0.21 0.24
Date
Location
MinTemp
MaxTemp
Rainfall
Evaporation
Sunshine
WindGustDir
WindGustSpeed
WindDir9am
WindDir3pm
WindSpeed9am
WindSpeed3pm
Humidity9am
Humidity3pm
Pressure9am
Pressure3pm
Cloud9am
Cloud3pm
Temp9am
Temp3pm
RainToday
RainTomorrow

```
[ ] sns.pairplot( data=MiceImputed, vars=('MaxTemp','MinTemp','Pressure9am','Pressure3pm', 'Temp9am', 'Temp3pm', 'Evaporation'), hue='RainTomorrow' )
```

```
<seaborn.axisgrid.PairGrid at 0x7fc1fe5969e0>
```




```
] # Standardizing data
  from sklearn import preprocessing
  r_scaler = preprocessing.MinMaxScaler()
  r_scaler.fit(MiceImputed)
  modified_data = pd.DataFrame(r_scaler.transform(MiceImputed), index=MiceImputed.index, columns=MiceImputed.columns)
  modified_data.head()
```

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	...	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	Cl
0	0.112372	0.285714	0.543417	0.514778	0.436785	0.584873	0.510633	0.866667	0.521127	0.866667	...	0.552255	0.223795	0.238938	0.293605	0.864222	0.
1	0.112656	0.285714	0.375350	0.568966	0.374206	0.578126	0.742548	0.933333	0.521127	0.400000	...	0.247951	0.253649	0.324484	0.313953	0.331893	0.
2	0.112940	0.285714	0.529412	0.583744	0.374206	0.710102	0.801728	1.000000	0.549296	0.866667	...	0.180328	0.303406	0.235988	0.340116	0.324179	0.
3	0.113224	0.285714	0.425770	0.640394	0.374206	0.599078	0.759122	0.266667	0.239437	0.600000	...	0.259222	0.164087	0.530973	0.459302	0.289348	0.
4	0.113507	0.285714	0.658263	0.746305	0.478505	0.654448	0.407984	0.866667	0.478873	0.066667	...	0.676230	0.333260	0.330383	0.261628	0.777027	0.

5 rows × 23 columns

dity9am	Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm	Temp9am	Temp3pm	RainToday	RainTomorrow
.552255	0.223795	0.238938	0.293605	0.864222	0.418757	0.506775	0.530612	0.222848	0.0
.247951	0.253649	0.324484	0.313953	0.331893	0.241005	0.514905	0.594388	0.222848	0.0
.180328	0.303406	0.235988	0.340116	0.324179	0.173386	0.617886	0.566327	0.222848	0.0
.259222	0.164087	0.530973	0.459302	0.289348	0.197888	0.539295	0.650510	0.222848	0.0
.676230	0.333260	0.330383	0.261628	0.777027	0.693545	0.531165	0.732143	0.222848	0.0

[ ]
```
# Feature Importance using Filter Method (Chi-Square)
from sklearn.feature_selection import SelectKBest, chi2
X = modified_data.loc[:,modified_data.columns!='RainTomorrow']
y = modified_data[['RainTomorrow']]
selector = SelectKBest(chi2, k=10)
selector.fit(X, y)
X_new = selector.transform(X)
print(X.columns[selector.get_support(indices=True)])
```

```
Index(['Rainfall', 'Sunshine', 'WindGustSpeed', 'Humidity9am', 'Humidity3pm',
       'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'RainToday'],
      dtype='object')
```

[ ]
```
from sklearn.feature_selection import SelectFromModel
from sklearn.ensemble import RandomForestClassifier as rf

X = MiceImputed.drop('RainTomorrow', axis=1)
y = MiceImputed['RainTomorrow']
selector = SelectFromModel(rf(n_estimators=100, random_state=0))
selector.fit(X, y)
support = selector.get_support()
features = X.loc[:,support].columns.tolist()
print(features)
print(rf(n_estimators=100, random_state=0).fit(X,y).feature_importances_)
```

```
['Sunshine', 'Humidity3pm', 'Pressure3pm', 'Cloud9am', 'Cloud3pm']
[0.03252805 0.0281657  0.03338748 0.03263243 0.02110761 0.03257039
 0.13228203 0.0206113  0.04250231 0.02149348 0.02218968 0.02166525
 0.02365058 0.03480388 0.10995365 0.04504083 0.06165314 0.05669448
 0.14838131 0.03182158 0.03544359 0.01142126]
```

[ ]
```
import warnings
warnings.filterwarnings("ignore")
```

[ ]
```
pip install eli5
```

```
Requirement already satisfied: eli5 in /usr/local/lib/python3.10/dist-packages (0.13.0)
Requirement already satisfied: attrs>17.1.0 in /usr/local/lib/python3.10/dist-packages (from eli5) (23.1.0)
Requirement already satisfied: jinja2>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from eli5) (3.1.2)
Requirement already satisfied: numpy>=1.9.0 in /usr/local/lib/python3.10/dist-packages (from eli5) (1.23.5)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from eli5) (1.11.2)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from eli5) (1.16.0)
Requirement already satisfied: scikit-learn>=0.20 in /usr/local/lib/python3.10/dist-packages (from eli5) (1.2.2)
Requirement already satisfied: graphviz in /usr/local/lib/python3.10/dist-packages (from eli5) (0.20.1)
Requirement already satisfied: tabulate>=0.7.7 in /usr/local/lib/python3.10/dist-packages (from eli5) (0.9.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2>=3.0.0->eli5) (2.1.3)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.20->eli5) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.20->eli5) (3.2.0)
```

```
import eli5
from eli5.sklearn import PermutationImportance

perm = PermutationImportance(rf(n_estimators=100, random_state=0).fit(X,y),random_state=1).fit(X,y)
eli5.show_weights(perm, feature_names = X.columns.tolist())
```

Weight	Feature
0.1562 ± 0.0012	Sunshine
0.1356 ± 0.0005	Humidity3pm
0.1230 ± 0.0012	Cloud3pm
0.0730 ± 0.0007	Pressure3pm
0.0347 ± 0.0008	WindGustSpeed
0.0266 ± 0.0004	Cloud9am
0.0108 ± 0.0003	Pressure9am
0.0036 ± 0.0002	Rainfall
0.0032 ± 0.0002	Humidity9am
0.0032 ± 0.0003	Evaporation
0.0020 ± 0.0002	MinTemp
0.0014 ± 0.0001	WindDir9am
0.0013 ± 0.0001	Location
0.0011 ± 0.0001	Temp3pm
0.0010 ± 0.0001	Temp9am
0.0009 ± 0.0001	Date
0.0009 ± 0.0001	RainToday
0.0008 ± 0.0001	WindSpeed3pm
0.0007 ± 0.0001	WindDir3pm
0.0006 ± 0.0001	MaxTemp
… 2 more …	

```
[ ] features = MiceImputed[['Location', 'MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine', 'WindGustDir',
                            'WindGustSpeed', 'WindDir9am', 'WindDir3pm', 'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am',
                            'Humidity3pm', 'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am', 'Temp3pm',
                            'RainToday']]
    target = MiceImputed['RainTomorrow']

    # Split into test and train
    from sklearn.model_selection import train_test_split
    X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.25, random_state=12345)

    # Normalize Features
    from sklearn.preprocessing import StandardScaler
    scaler = StandardScaler()
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.fit_transform(X_test)
```

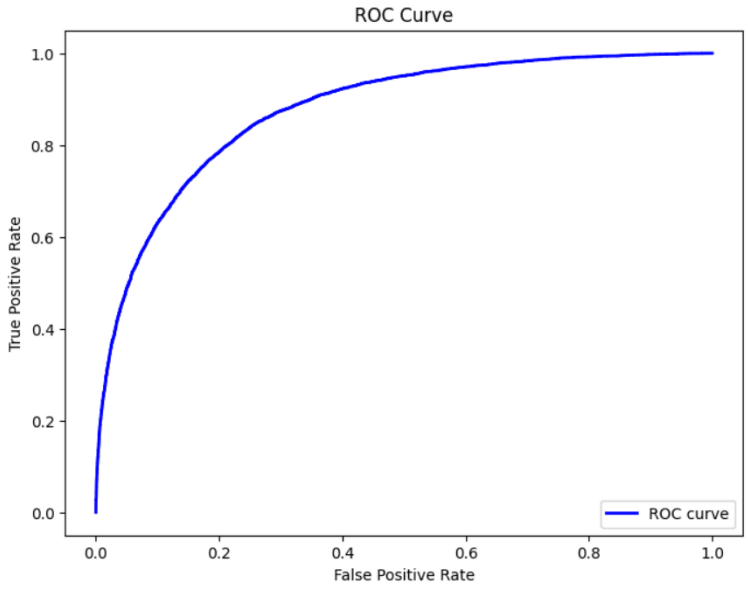
```
[ ] def plot_roc_cur(fper, tper):
        plt.plot(fper, tper, color='orange', label='ROC')
        plt.plot([0, 1], [0, 1], color='darkblue', linestyle='--')
        plt.xlabel('False Positive Rate')
        plt.ylabel('True Positive Rate')
        plt.title('Receiver Operating Characteristic (ROC) Curve')
        plt.legend()
        plt.show()
```

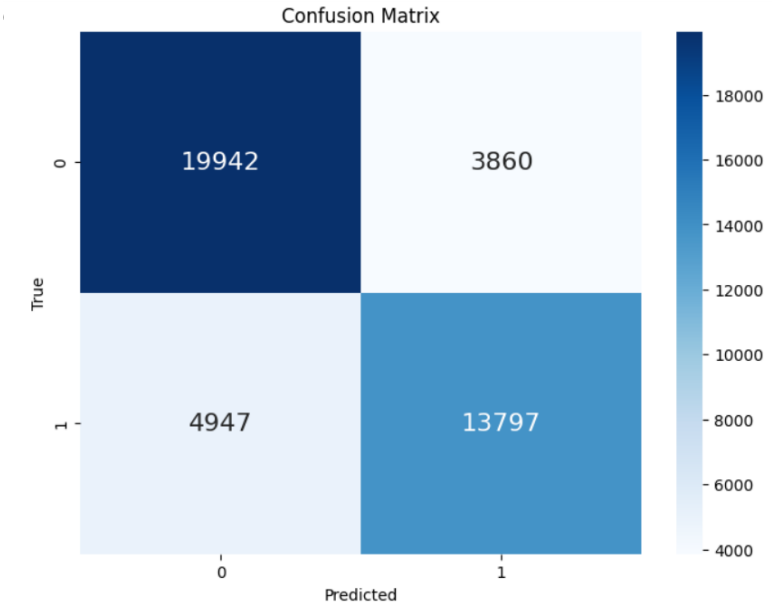
```
import time
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import (
    accuracy_score,
    roc_auc_score,
    cohen_kappa_score,
    confusion_matrix,
    roc_curve,
    classification_report,
)

def run_model(model, X_train, y_train, X_test, y_test, verbose=True):
    t0 = time.time()
    model.fit(X_train, y_train) if verbose else model.fit(X_train, y_train, verbose=0)
    y_pred = model.predict(X_test)

    accuracy = accuracy_score(y_test, y_pred)
    roc_auc = roc_auc_score(y_test, y_pred)
    coh_kap = cohen_kappa_score(y_test, y_pred)
    time_taken = time.time() - t0

    print("Accuracy = {:.5f}".format(accuracy))
    print("ROC Area under Curve = {:.5f}".format(roc_auc))
    print("Cohen's Kappa = {:.5f}".format(coh_kap))
    print("Time taken = {:.2f} seconds".format(time_taken))
    print(classification_report(y_test, y_pred, digits=5))

    probs = model.predict_proba(X_test)[:, 1]
    fper, tper, thresholds = roc_curve(y_test, probs)
```

```
    print("Time taken = {:.2f} seconds".format(time_taken))
    print(classification_report(y_test, y_pred, digits=5))

    probs = model.predict_proba(X_test)[:, 1]
    fper, tper, thresholds = roc_curve(y_test, probs)

    plt.figure(figsize=(8, 6))
    plt.plot(fper, tper, color='blue', lw=2, label='ROC curve')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('ROC Curve')
    plt.legend(loc='lower right')
    plt.show()

    # Calculate and display confusion matrix
    cm = confusion_matrix(y_test, y_pred)
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", annot_kws={"size": 16})
    plt.xlabel('Predicted')
    plt.ylabel('True')
    plt.title('Confusion Matrix')
    plt.show()

    return model, accuracy, roc_auc, coh_kap, time_taken

# Usage example:
# model, accuracy, roc_auc, coh_kap, time_taken = run_model(your_model, X_train, y_train, X_test, y_test, verbose=True)
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split


params_lr = {'penalty': 'l1', 'solver':'liblinear'}

model_lr = LogisticRegression(**params_lr)
model_lr, accuracy_lr, roc_auc_lr, coh_kap_lr, tt_lr = run_model(model_lr, X_train, y_train, X_test, y_test)
```

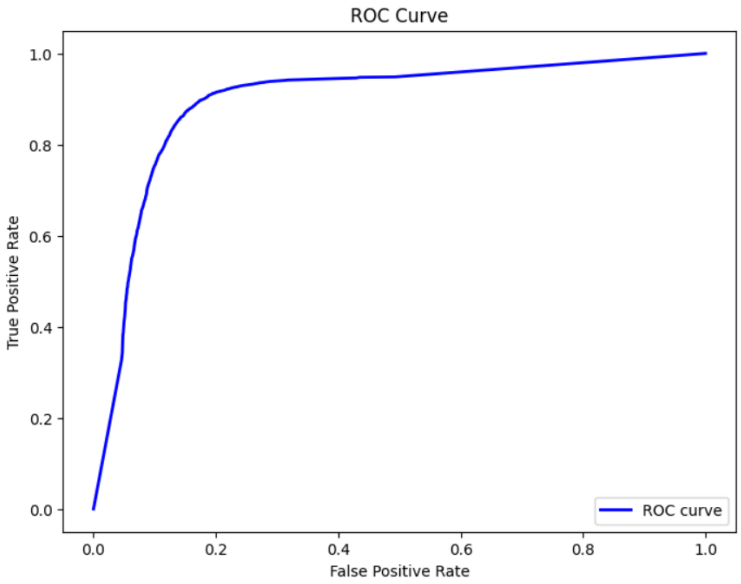
```
Accuracy = 0.79300
ROC Area under Curve = 0.78695
Cohen's Kappa = 0.57746
Time taken = 6.73 seconds
              precision    recall  f1-score   support

         0.0    0.80124   0.83783   0.81912     23802
         1.0    0.78139   0.73608   0.75806     18744

    accuracy                        0.79300     42546
   macro avg    0.79131   0.78695   0.78859     42546
weighted avg    0.79249   0.79300   0.79222     42546
```

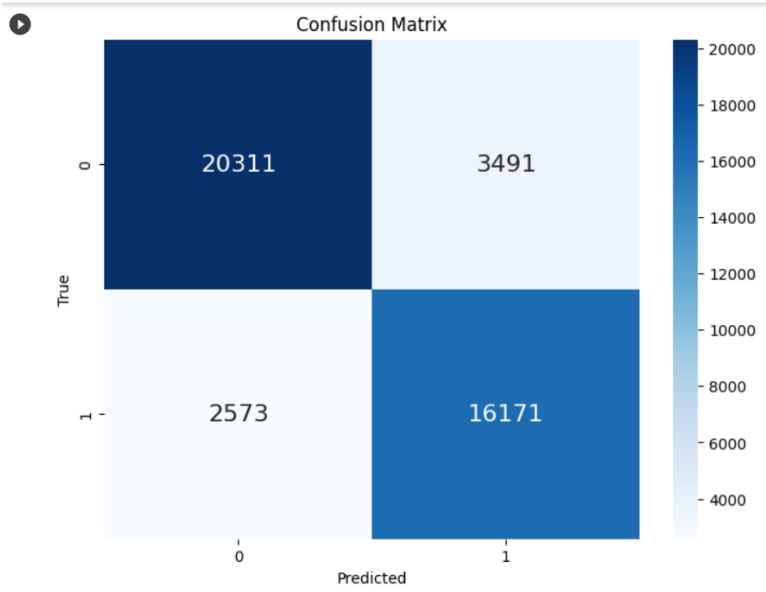





```
from sklearn.tree import DecisionTreeClassifier

params_dt = {'max_depth': 16,
             'max_features': "sqrt"}

model_dt = DecisionTreeClassifier(**params_dt)
model_dt, accuracy_dt, roc_auc_dt, coh_kap_dt, tt_dt = run_model(model_dt, X_train, y_train, X_test, y_test)
```

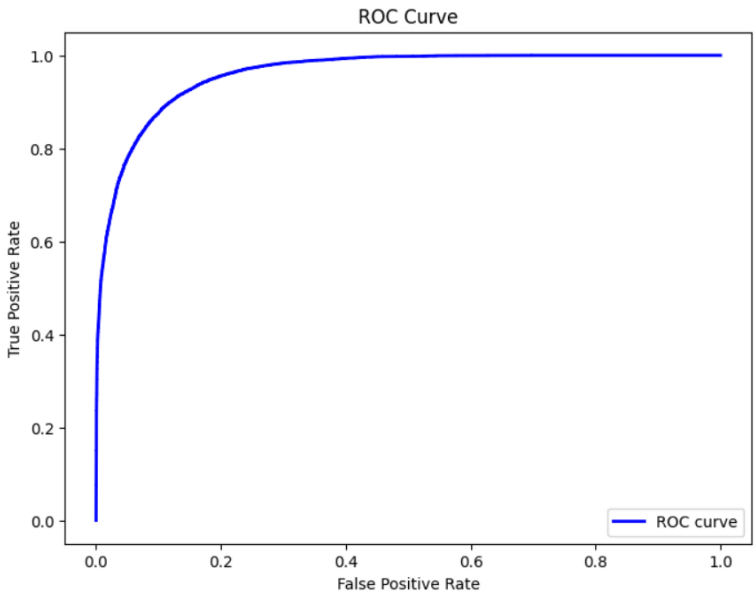
```
Accuracy = 0.85747
ROC Area under Curve = 0.85803
Cohen's Kappa = 0.71235
Time taken = 0.56 seconds
              precision    recall  f1-score   support

         0.0    0.88756   0.85333   0.87011     23802
         1.0    0.82245   0.86273   0.84211     18744

    accuracy                        0.85747     42546
   macro avg    0.85501   0.85803   0.85611     42546
weighted avg    0.85888   0.85747   0.85777     42546
```

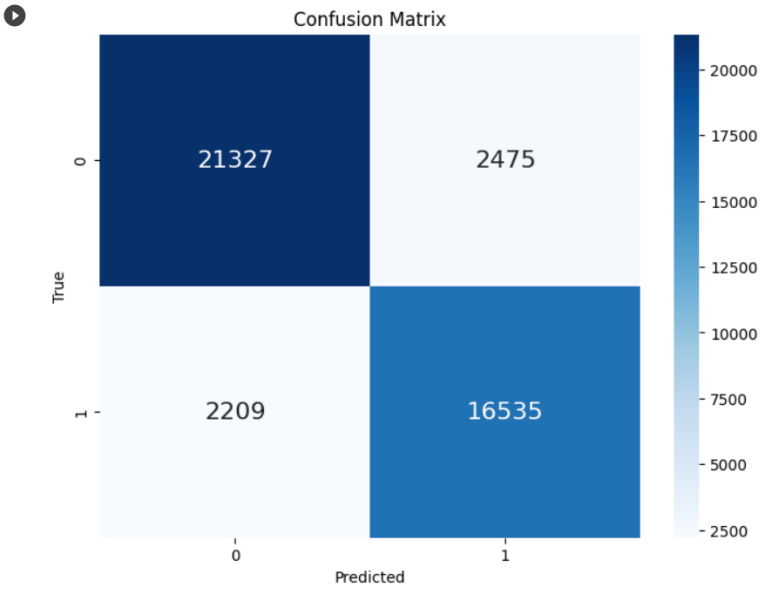





```
from sklearn.neural_network import MLPClassifier

params_nn = {'hidden_layer_sizes': (30,30,30),
             'activation': 'logistic',
             'solver': 'lbfgs',
             'max_iter': 500}

model_nn = MLPClassifier(**params_nn)
model_nn, accuracy_nn, roc_auc_nn, coh_kap_nn, tt_nn = run_model(model_nn, X_train, y_train, X_test, y_test)
```

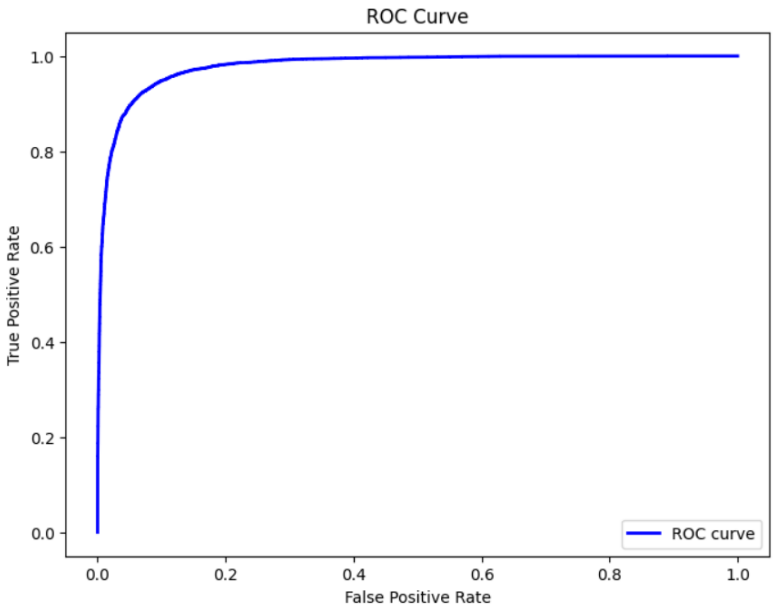
```
Accuracy = 0.88991
ROC Area under Curve = 0.88908
Cohen's Kappa = 0.77699
Time taken = 345.19 seconds
              precision    recall  f1-score   support

         0.0    0.90614   0.89602   0.90105     23802
         1.0    0.86981   0.88215   0.87593     18744

    accuracy                        0.88991     42546
   macro avg    0.88797   0.88908   0.88849     42546
weighted avg    0.89013   0.88991   0.88999     42546
```

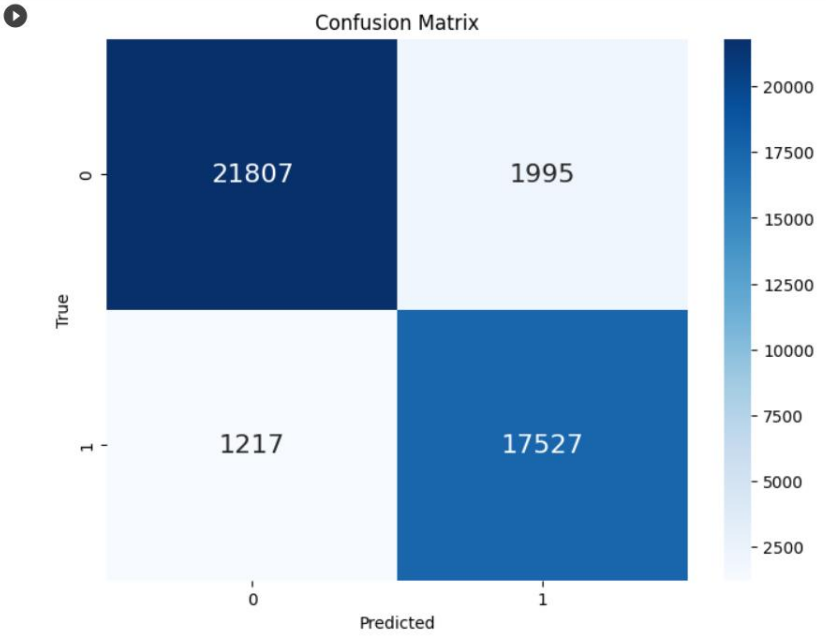

ROC Curve
True Positive Rate
False Positive Rate
ROC curve
0.0
0.2
0.4
0.6
0.8
1.0


Confusion Matrix
21327
2475
2209
16535
True
Predicted
0
1
20000
17500
15000
12500
10000
7500
5000
2500

```
from sklearn.ensemble import RandomForestClassifier

params_rf = {'max_depth': 16,
             'min_samples_leaf': 1,
             'min_samples_split': 2,
             'n_estimators': 100,
             'random_state': 12345}

model_rf = RandomForestClassifier(**params_rf)
model_rf, accuracy_rf, roc_auc_rf, coh_kap_rf, tt_rf = run_model(model_rf, X_train, y_train, X_test, y_test)
```

```
Accuracy = 0.92451
ROC Area under Curve = 0.92563
Cohen's Kappa = 0.84752
Time taken = 37.74 seconds
              precision    recall  f1-score   support

         0.0    0.94714   0.91618   0.93141     23802
         1.0    0.89781   0.93507   0.91606     18744

    accuracy                        0.92451     42546
   macro avg    0.92247   0.92563   0.92373     42546
weighted avg    0.92541   0.92451   0.92465     42546
```






```
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, accuracy_lgb, roc_auc_lgb, coh_kap_lgb, tt_lgb = run_model(model_lgb, X_train, y_train, X_test, y_test)
```

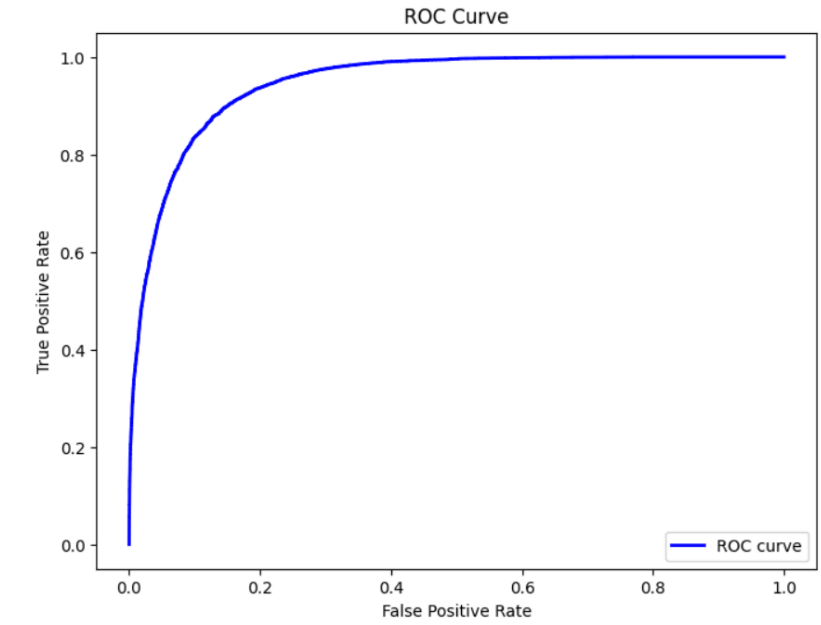
```
[LightGBM] [Info] Number of positive: 56098, number of negative: 71539
[LightGBM] [Warning] Auto-choosing col-wise multi-threading, the overhead of testing was 0.026642 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 4322
[LightGBM] [Info] Number of data points in the train set: 127637, number of used features: 21
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.439512 -> initscore=-0.243143
[LightGBM] [Info] Start training from score -0.243143
Accuracy = 0.87334
ROC Area under Curve = 0.87329
Cohen's Kappa = 0.74395
Time taken = 7.46 seconds
              precision    recall  f1-score   support

         0.0    0.89719   0.87371   0.88529     23802
         1.0    0.84479   0.87287   0.85860     18744

    accuracy                        0.87334     42546
```

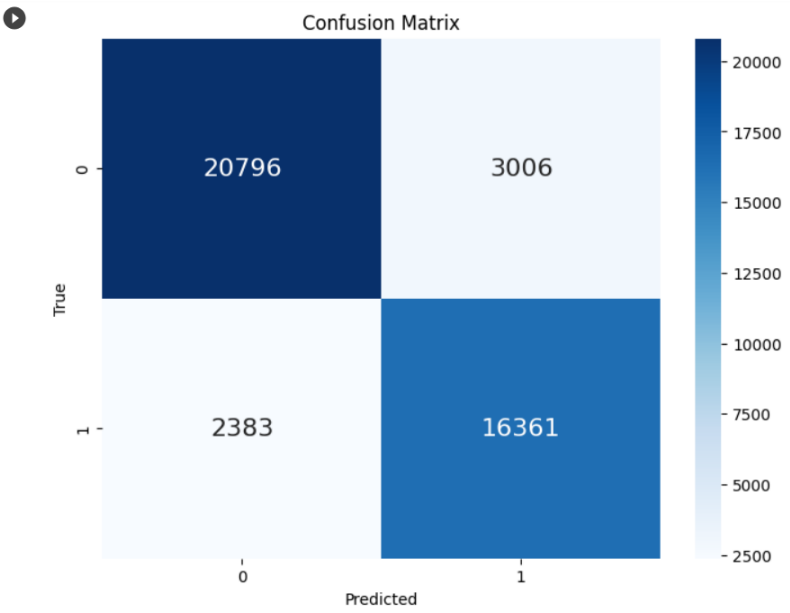
```
model_lgb = lgb.LGBMClassifier(**params_lgb)
model_lgb, accuracy_lgb, roc_auc_lgb, coh_kap_lgb, tt_lgb = run_model(model_lgb, X_train, y_train, X_test, y_test)
```

```
[LightGBM] [Info] Number of positive: 56098, number of negative: 71539
[LightGBM] [Warning] Auto-choosing col-wise multi-threading, the overhead of testing was 0.026642 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 4322
[LightGBM] [Info] Number of data points in the train set: 127637, number of used features: 21
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.439512 -> initscore=-0.243143
[LightGBM] [Info] Start training from score -0.243143
Accuracy = 0.87334
ROC Area under Curve = 0.87329
Cohen's Kappa = 0.74395
Time taken = 7.46 seconds
              precision    recall  f1-score   support

         0.0    0.89719   0.87371   0.88529     23802
         1.0    0.84479   0.87287   0.85860     18744

    accuracy                        0.87334     42546
   macro avg    0.87099   0.87329   0.87195     42546
weighted avg    0.87410   0.87334   0.87353     42546
```

```
model_lgb = lgb.LGBMClassifier(**params_lgb)
model_lgb, accuracy_lgb, roc_auc_lgb, coh_kap_lgb, tt_lgb = run_model(model_lgb, X_train, y_train, X_test, y_test)
```

```
[LightGBM] [Info] Number of positive: 56098, number of negative: 71539
[LightGBM] [Warning] Auto-choosing col-wise multi-threading, the overhead of testing was 0.026642 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 4322
[LightGBM] [Info] Number of data points in the train set: 127637, number of used features: 21
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.439512 -> initscore=-0.243143
[LightGBM] [Info] Start training from score -0.243143
Accuracy = 0.87334
ROC Area under Curve = 0.87329
Cohen's Kappa = 0.74395
Time taken = 7.46 seconds
              precision    recall  f1-score   support

         0.0    0.89719   0.87371   0.88529     23802
         1.0    0.84479   0.87287   0.85860     18744

    accuracy                        0.87334     42546
   macro avg    0.87099   0.87329   0.87195     42546
weighted avg    0.87410   0.87334   0.87353     42546
```



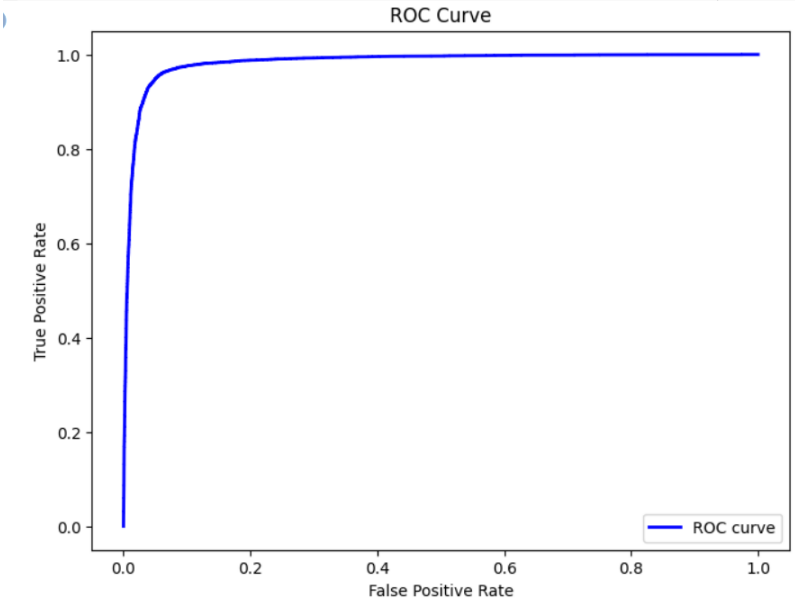


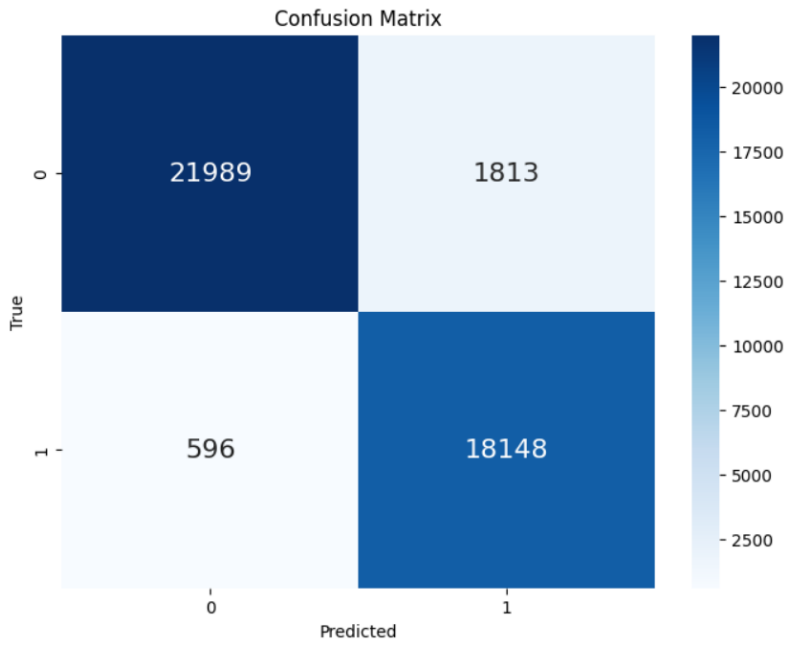
```
pip install catboost
```

```
Collecting catboost
  Downloading catboost-1.2.2-cp310-cp310-manylinux2014_x86_64.whl (98.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 98.7/98.7 MB 6.0 MB/s eta 0:00:00
Requirement already satisfied: graphviz in /usr/local/lib/python3.10/dist-packages (from catboost) (0.20.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from catboost) (3.7.1)
Requirement already satisfied: numpy>=1.16.0 in /usr/local/lib/python3.10/dist-packages (from catboost) (1.23.5)
Requirement already satisfied: pandas>=0.24 in /usr/local/lib/python3.10/dist-packages (from catboost) (1.5.3)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from catboost) (1.11.2)
Requirement already satisfied: plotly in /usr/local/lib/python3.10/dist-packages (from catboost) (5.15.0)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from catboost) (1.16.0)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24->catboost) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.24->catboost) (2023.3.post1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (1.1.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (4.42.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (23.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (3.1.1)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from plotly->catboost) (8.2.3)
Installing collected packages: catboost
Successfully installed catboost-1.2.2
```

```
import catboost as cb
params_cb ={'iterations': 50,
            'max_depth': 16}

model_cb = cb.CatBoostClassifier(**params_cb)
model_cb, accuracy_cb, roc_auc_cb, coh_kap_cb, tt_cb = run_model(model_cb, X_train, y_train, X_test, y_test, verbose=False)
```

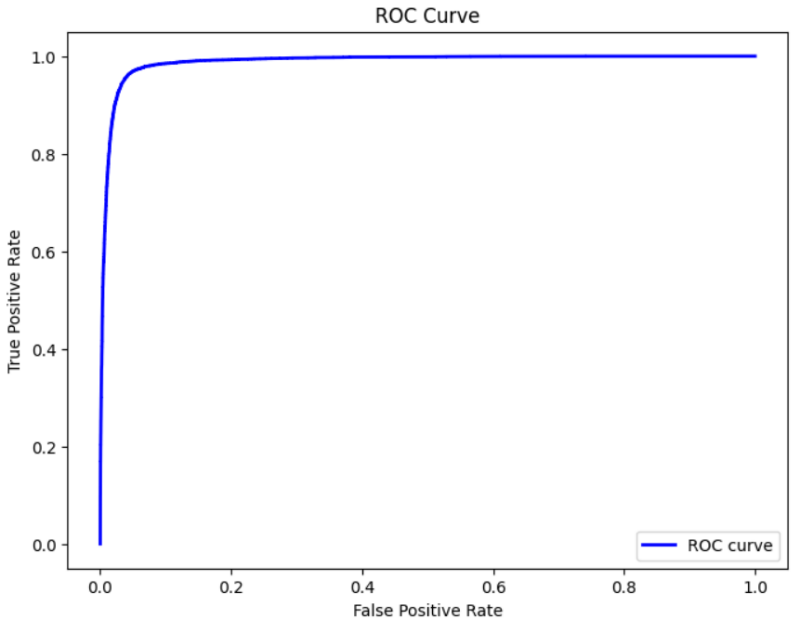
```
Accuracy = 0.94338
ROC Area under Curve = 0.94602
Cohen's Kappa = 0.88592
Time taken = 448.02 seconds
              precision    recall  f1-score   support

         0.0    0.97361   0.92383   0.94807     23802
         1.0    0.90917   0.96820   0.93776     18744

    accuracy                        0.94338     42546
   macro avg    0.94139   0.94602   0.94291     42546
weighted avg    0.94522   0.94338   0.94353     42546
```

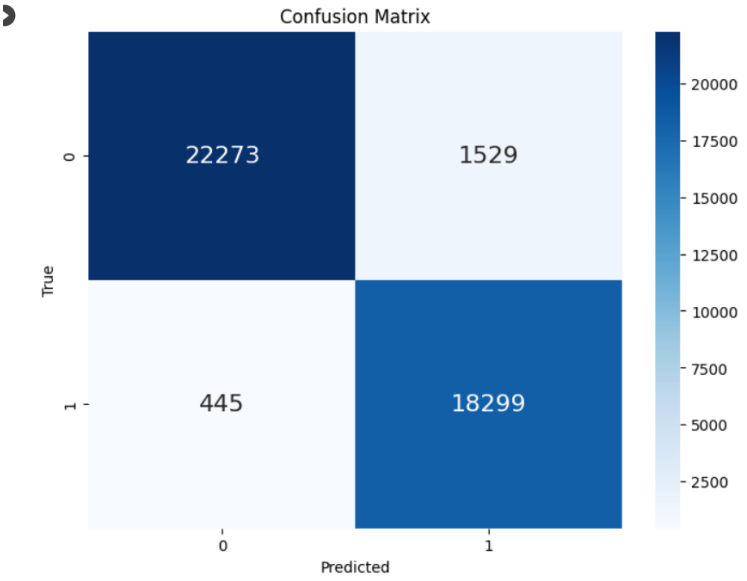





```
[ ] import xgboost as xgb
    params_xgb ={'n_estimators': 500,
                 'max_depth': 16}

    model_xgb = xgb.XGBClassifier(**params_xgb)
    model_xgb, accuracy_xgb, roc_auc_xgb, coh_kap_xgb, tt_xgb = run_model(model_xgb, X_train, y_train, X_test, y_test)
```

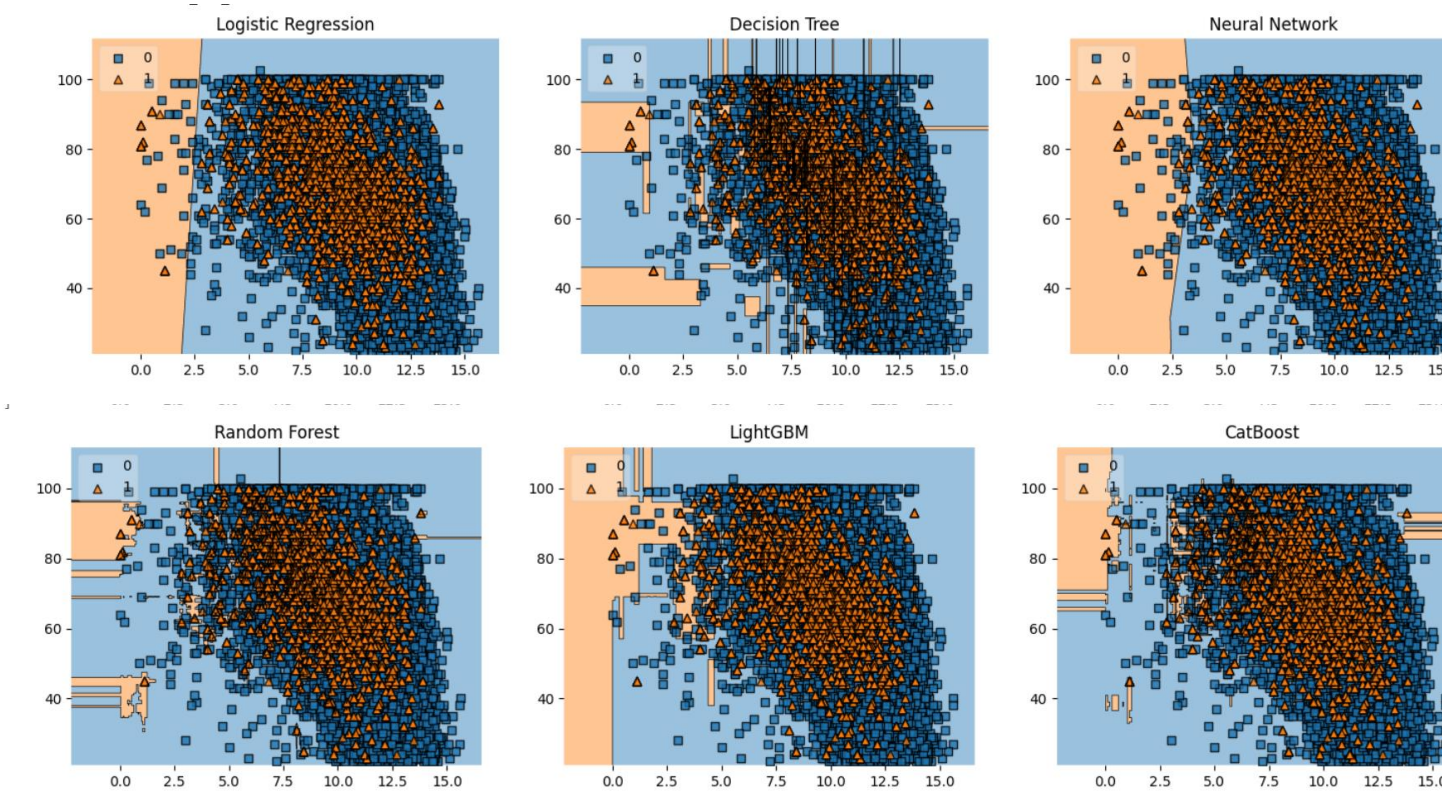
```
Accuracy = 0.95360
ROC Area under Curve = 0.95601
Cohen's Kappa = 0.90645
Time taken = 348.38 seconds
              precision    recall  f1-score   support

         0.0    0.98041   0.93576   0.95757     23802
         1.0    0.92289   0.97626   0.94882     18744

    accuracy                        0.95360     42546
   macro avg    0.95165   0.95601   0.95319     42546
weighted avg    0.95507   0.95360   0.95371     42546
```

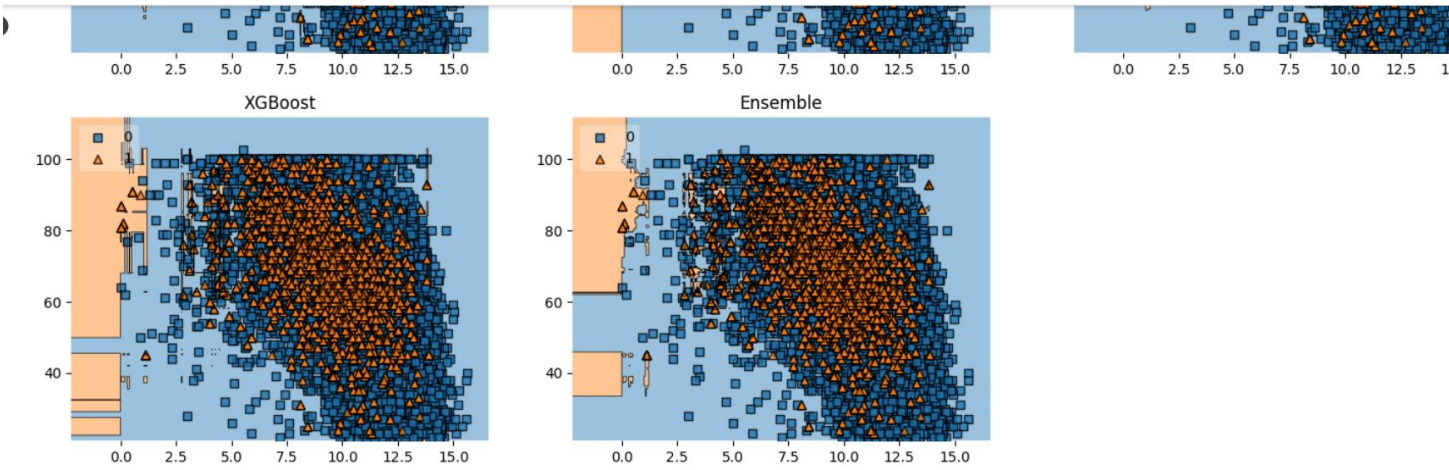




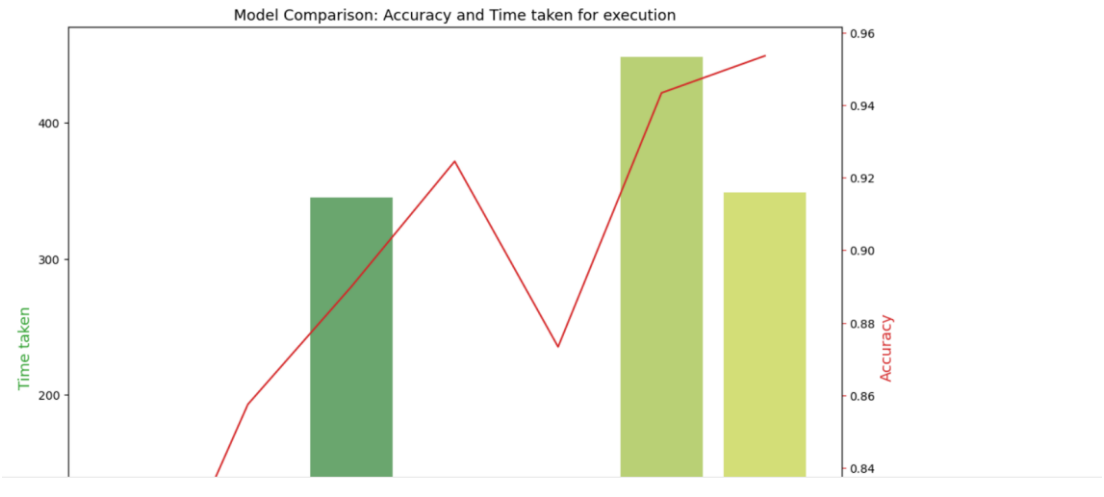

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
import itertools
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier
import lightgbm as lgb
import catboost as cb
import xgboost as xgb
from mlxtend.classifier import EnsembleVoteClassifier
from mlxtend.plotting import plot_decision_regions

value = 1.80
width = 0.90

clf1 = LogisticRegression(random_state=12345)
clf2 = DecisionTreeClassifier(random_state=12345)
clf3 = MLPClassifier(random_state=12345, verbose = 0)
clf4 = RandomForestClassifier(random_state=12345)
clf5 = lgb.LGBMClassifier(random_state=12345, verbose = 0)
clf6 = cb.CatBoostClassifier(random_state=12345, verbose = 0)
clf7 = xgb.XGBClassifier(random_state=12345)
eclf = EnsembleVoteClassifier(clfs=[clf4, clf5, clf6, clf7], weights=[1, 1, 1, 1], voting='soft')
```

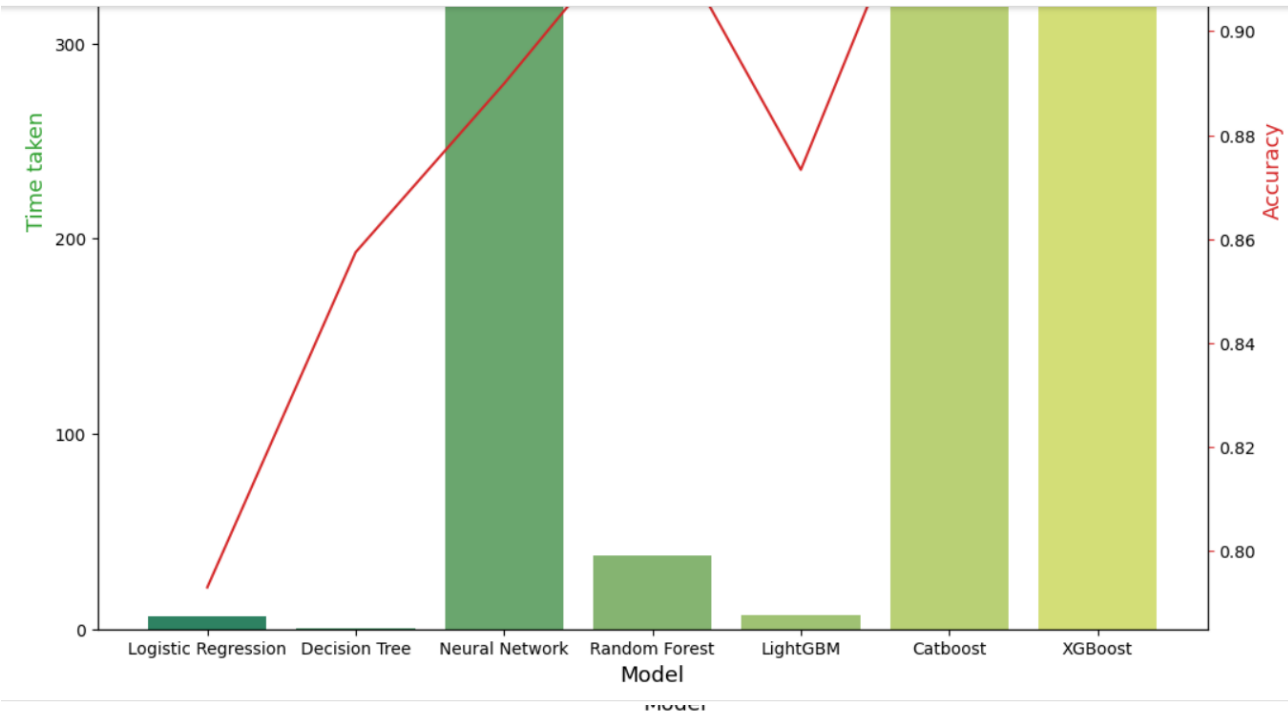
```
X_list = MiceImputed[["Sunshine", "Humidity9am", "Cloud3pm"]] #took only really important features
X = np.asarray(X_list, dtype=np.float32)
y_list = MiceImputed["RainTomorrow"]
y = np.asarray(y_list, dtype=np.int32)

# Plotting Decision Regions
gs = gridspec.GridSpec(3,3)
fig = plt.figure(figsize=(18, 14))

labels = ['Logistic Regression',
          'Decision Tree',
          'Neural Network',
          'Random Forest',
          'LightGBM',
          'CatBoost',
          'XGBoost',
          'Ensemble']

for clf, lab, grd in zip([clf1, clf2, clf3, clf4, clf5, clf6, clf7, eclf],
                         labels,
                         itertools.product([0, 1, 2],
                         repeat=2)):
    clf.fit(X, y)
    ax = plt.subplot(gs[grd[0], grd[1]])
    fig = plot_decision_regions(X=X, y=y, clf=clf,
                                filler_feature_values={2: value},
                                filler_feature_ranges={2: width},
                                legend=2)
    plt.title(lab)
```



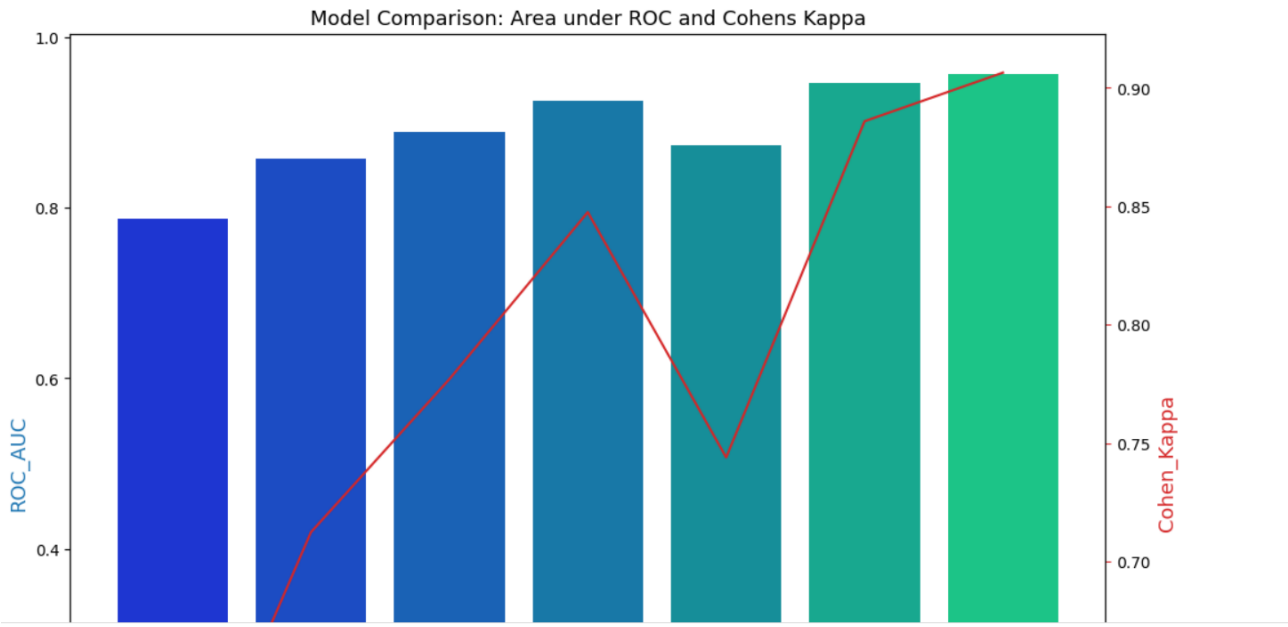


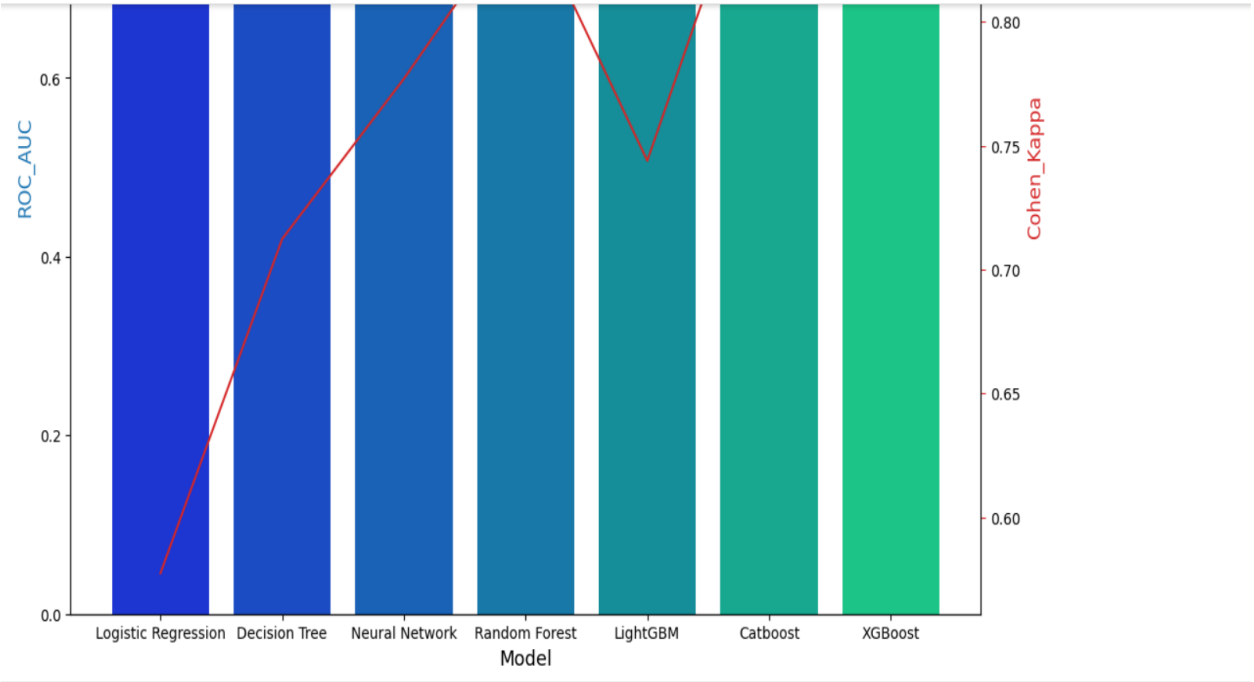

```
accuracy_scores = [accuracy_lr, accuracy_dt, accuracy_nn, accuracy_rf, accuracy_lgb, accuracy_cb, accuracy_xgb]
roc_auc_scores = [roc_auc_lr, roc_auc_dt, roc_auc_nn, roc_auc_rf, roc_auc_lgb, roc_auc_cb, roc_auc_xgb]
coh_kap_scores = [coh_kap_lr, coh_kap_dt, coh_kap_nn, coh_kap_rf, coh_kap_lgb, coh_kap_cb, coh_kap_xgb]
tt = [tt_lr, tt_dt, tt_nn, tt_rf, tt_lgb, tt_cb, tt_xgb]

model_data = {'Model': ['Logistic Regression','Decision Tree','Neural Network','Random Forest','LightGBM','Catboost','XGBoost'],
              'Accuracy': accuracy_scores,
              'ROC_AUC': roc_auc_scores,
              'Cohen_Kappa': coh_kap_scores,
              'Time taken': tt}
data = pd.DataFrame(model_data)

fig, ax1 = plt.subplots(figsize=(12,10))
ax1.set_title('Model Comparison: Accuracy and Time taken for execution', fontsize=13)
color = 'tab:green'
ax1.set_xlabel('Model', fontsize=13)
ax1.set_ylabel('Time taken', fontsize=13, color=color)
ax2 = sns.barplot(x='Model', y='Time taken', data = data, palette='summer')
ax1.tick_params(axis='y')
ax2 = ax1.twinx()
color = 'tab:red'
ax2.set_ylabel('Accuracy', fontsize=13, color=color)
ax2 = sns.lineplot(x='Model', y='Accuracy', data = data, sort=False, color=color)
ax2.tick_params(axis='y', color=color)
```






```
[ ]  fig, ax3 = plt.subplots(figsize=(12,10))
     ax3.set_title('Model Comparison: Area under ROC and Cohens Kappa', fontsize=13)
     color = 'tab:blue'
     ax3.set_xlabel('Model', fontsize=13)
     ax3.set_ylabel('ROC_AUC', fontsize=13, color=color)
     ax4 = sns.barplot(x='Model', y='ROC_AUC', data = data, palette='winter')
     ax3.tick_params(axis='y')
     ax4 = ax3.twinx()
     color = 'tab:red'
     ax4.set_ylabel('Cohen_Kappa', fontsize=13, color=color)
     ax4 = sns.lineplot(x='Model', y='Cohen_Kappa', data = data, sort=False, color=color)
     ax4.tick_params(axis='y', color=color)
     plt.show()
```




ROC_AUC
Cohen_Kappa
0.0
0.2
0.4
0.6
0.60
0.65
0.70
0.75
0.80
Logistic Regression
Decision Tree
Neural Network
Random Forest
LightGBM
Catboost
XGBoost
Model

CHAPTER 4 RESULTS AND DISCUSSION

Decision Tree, RandomForest, Xgboost,catboost ,etc..are the classification method used for time series predict in this research work. Two group are separated from the data set for training and for testing the algorithms of classification. To execute the classification algorithms, the tool used is flask webapp data examination. For classification procedure no more than a separation of data is particular from the loaded data. To choose a subset from innovative data, “Select attribute” are utilised by the operative. The preferred subset is then subjected to “X-Validation” operator. It develop the classification representation which is validated by the test data.

CHAPTER 5
CONCLUSION

The overall aim is to define various ML techniques that are useful in predicting rainfall. The goal of this research is to design accurate and efficient model by applying lesser number of attributes and tests. Firstly, the data is pre-processed and then it is used in the model. Random Forest classifier with approximately 88% are the most efficient classification algorithms. However, Decision Tree classifier gives the least accuracy with 73%. We can further expand this research covering other ML techniques such as time series, clustering and association rules and other ensemble techniques. Taking into consideration the limitations of this study, there is a need to build more complex and combination of models to get higher accuracy for rainfall prediction system. Study can also be formulated using greater articulate monitoring for particular area and create this kind of model for enormous dataset so that calculation rate can be increased with better precision and with more accuracy.TWeather forecasting is a meteorological work that easy to modify researcher work by applying the numerical weather prediction method. Weather forecasted by using various data mining techniques especially classification clustering and neural network, decision tree. The key aim for improving the classification and prediction performance for the traditional; weather prediction model is designed and developed in this work. But some limitation of the model is also observed, thus in near future need to be review before use of the proposed technique. And also soil there are some issues and challenges in which better implement of data mining technique should be implemented in field of weather forecasting.The performance was evaluated by comparing the predicted results with the observed (actual) measures. Information retrieval metrics and statistical techniques were used for performance analysis of proposed techniques in comparison with other methods from the literature. We can observe that **XGBoost, CatBoost and Random Forest** have performed better compared to other models. However, if speed is an important thing to consider, we can stick to Random Forest instead of XGBoost or CatBoost. 'Logistic Regression','Decision Tree','Neural Network','Random Forest','LightGBM','CatBoost','XGBoost','Ensemble' are the

Classifiers used in this Program

REFERENCES:

[1] Xiong, Lihua, and Kieran M. OConnor. “An empirical method to improve the prediction limits of the GLUE methodology in rainfallrunoff modeling.” Journal of Hydrology 349.1-2 (2008): 115-124.

[2] Schmitz, G. H., and J. Cullmann. “PAI-OFF: A new proposal for online flood forecasting in flash flood prone catchments.” Journal of hydrology 360.1-4 (2008): 1-14.

[3] Riordan, Denis, and Bjarne K. Hansen. “A fuzzy casebased system for weather prediction.” Engineering Intelligent Systems for Electrical Engineering and Communications 10.3 (2002): 139-146.

[4] Guhathakurta, P. “Long-range monsoon rainfall prediction of 2005 for the districts and sub-division Kerala with artificial neural network.” Current Science 90.6 (2006): 773-779. [5] Pilgrim, D. H., T. G. Chapman, and D. G. Doran. “Problems of rainfall-runoff modelling in arid and semiarid regions.” Hydrological Sciences Journal 33.4 (1988): 379-400. [6] Lee, Sunyoung, Sungzoon Cho, and Patrick M. Wong. “Rainfall prediction using artificial neural networks.” journal of geographic information and Decision Analysis 2.2 (1998): 233242..

[7] French, Mark N., Witold F. Krajewski, and Robert R. Cuykendall. “Rainfall forecasting in space and time using a neural network.” Journal of hydrology 137.1-4 (1992): 1-31. [8] Charaniya, Nizar Ali, and Sanjay V. Dudul. “Committee of artificial neural networks for monthly rainfall prediction using wavelet transform.” Business, Engineering and Industrial Applications (ICBEIA), 2011 International Conference on. IEEE, 2011.

[9] Noone, David, and Harvey Stern. “Verification of rainfall forecasts from the Australian Bureau of Meteorology"s Global Assimilation and Prognosis(GASP) system.” Australian Meteorological Magazine 44.4 (1995): 275-286.

[10] Hornik, Kurt, Maxwell Stinchcombe, and Halbert White. “Multilayer feedforward networks are universal approximators.” Neural networks 2.5 (1989): 359-366.

[11] Haykin, Simon. Neural networks: a comprehensive foundation. Prentice Hall PTR, 1994.

[12] Rajeevan, M., Pulak Guhathakurta, and V. Thapliyal. “New models for long range forecasts of summer monsoon rainfall over North West and Peninsular India.” Meteorology and Atmospheric Physics 73.3-4 (2000): 211-225.