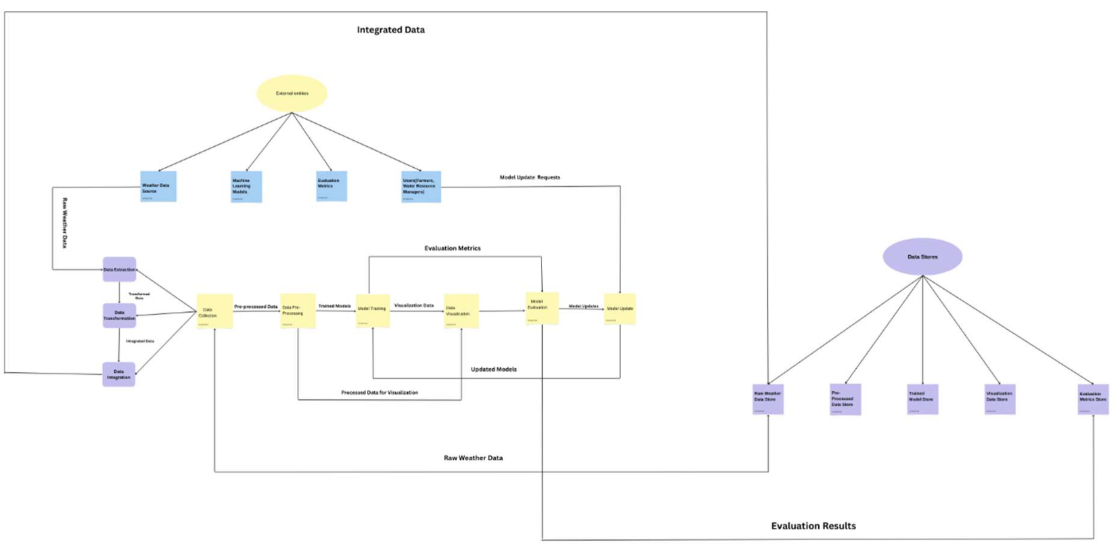
PROJECT MANUAL

Date	01 NOVEMBER 2023
Team ID	Team-591871
Project Name	Prediction of rain fall
Maximum Marks	4 Marks

Project Description:

Particularly during the torrential rainfall event. Moreover, one of the major focuses of Climate change study is to understand whether there are extreme changes in the occurrence and frequency of heavy rainfall events. The accuracy level of the ML models used in predicting rainfall based on historical data has been one of the most critical concerns in hydrological studies. An accurate ML model could give early alerts of severe weather to help prevent natural disasters and destruction. Hence, there is needs to develop ML algorithms capable in predicting rainfall with acceptable level of precision and in reducing the error in the dataset of the projected rainfall from climate change model with the expected observable rainfall.

Technical Architecture:



Activity 1: Import Necessary Libraries

o It is important to import all the necessary libraries such as pandas, numpy, matplotlib.

o Numpy- It is an open-source numerical Python library. It contains a multi-dimensional array and matrix data structures. It can be used to perform mathematical operations on arrays such as trigonometric, statistical, and algebraic routines.

o Pandas- It is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

o Seaborn- Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

o Matplotlib- Visualisation with python. It is a comprehensive library for creating static,animated, and interactive visualizations in Python

o Sklearn – which contains all the modules required for model building.

```
In [59]: # Libraries required
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn import model_selection
from sklearn import metrics
from sklearn import linear_model
from sklearn import ensemble
from sklearn import tree
from sklearn import svm
import xgboost
```

Activity 2: Importing the Dataset

You might have your data in .csv files, .excel files

Let's load a .csv data file into pandas using read_csv() function.We will need to locate the directory of the CSV file at first (it's more efficient to keep the dataset in the same directory as your program).

If your dataset is in some other location ,Then Data=pd.read_csv(r"File_location/datasetname.csv") Note:r stands for "raw" and will cause backslashes in the string to be interpreted as actual backslashes rather than special characters.

If the dataset in same directory of your program, you can directly read it, without giving raw as r.

- Our Dataset weatherAus.csv contains following Columns
- Location, MinTemp, MaxTemp, Rainfall, WindGustSpeed,
- WindSpeed9am, WindSpeed3pm, Humidity9am, Humidity3pm
- Pressure9am, Pressure3pm, Temp9am, Temp3pm, RainToday

, ● WindGustDir, WindDir9am, WindDir3pm,date

● Raintommorrow – output column

The output column to be predicted is RainTommorow .Based on the input variables we predict the chance of rain. The predicted output gives them a fair idea about it will rain or not.

Activity 3: Analyse the data

● head() method is used to return top n (5 by default) rows of a DataFrame or series

```
In [60]: data = pd.read_csv(r"C:\Users\anumo\Downloads\Dataset - Dataset.csv")

In [61]: data.head()
Out[61]:
```

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	...	Humidity9am	Humidity3pm	Pressure9
0	2008-12-01	Delhi	13.4	22.9	0.6	NaN	NaN	W	44.0	W	...	71.0	22.0	100
1	2008-12-02	Delhi	7.4	25.1	0.0	NaN	NaN	WNW	44.0	NNW	...	44.0	25.0	101
2	2008-12-03	Delhi	12.9	25.7	0.0	NaN	NaN	WSW	46.0	W	...	38.0	30.0	100
3	2008-12-04	Delhi	9.2	28.0	0.0	NaN	NaN	NE	24.0	SE	...	45.0	16.0	101
4	2008-12-05	Delhi	17.5	32.3	1.0	NaN	NaN	W	41.0	ENE	...	82.0	33.0	101

5 rows × 23 columns

● describe() method computes a summary of statistics like count, mean, standard deviation, min, max and quartile values.

```
data.describe()
```

The output is as shown below

```
In [62]: data.describe()
Out[62]:
```

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3p
count	143975.000000	144199.000000	142199.000000	82670.000000	75625.000000	135197.000000	143693.000000	142398.000000	142806.000000	140953.0000
mean	12.194034	23.221348	2.360918	5.468232	7.611178	40.035230	14.043426	18.662657	68.880831	51.5391
std	6.398495	7.119049	8.478060	4.193704	3.785483	13.607062	8.915375	8.809800	19.029164	20.7959
min	-8.500000	-4.800000	0.000000	0.000000	0.000000	6.000000	0.000000	0.000000	0.000000	0.0000
25%	7.600000	17.900000	0.000000	2.600000	4.800000	31.000000	7.000000	13.000000	57.000000	37.0000
50%	12.000000	22.600000	0.000000	4.800000	8.400000	39.000000	13.000000	19.000000	70.000000	52.0000
75%	16.900000	28.200000	0.800000	7.400000	10.600000	48.000000	19.000000	24.000000	83.000000	66.0000
max	33.900000	48.100000	371.000000	145.000000	14.500000	135.000000	130.000000	87.000000	100.000000	100.0000

From the data we infer that there are only decimal values and no categorical values ● info() gives information about the data

```
In [63]: 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  object 
 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), object(7)
memory usage: 25.5+ MB
```

Activity 4: Handling Missing Values

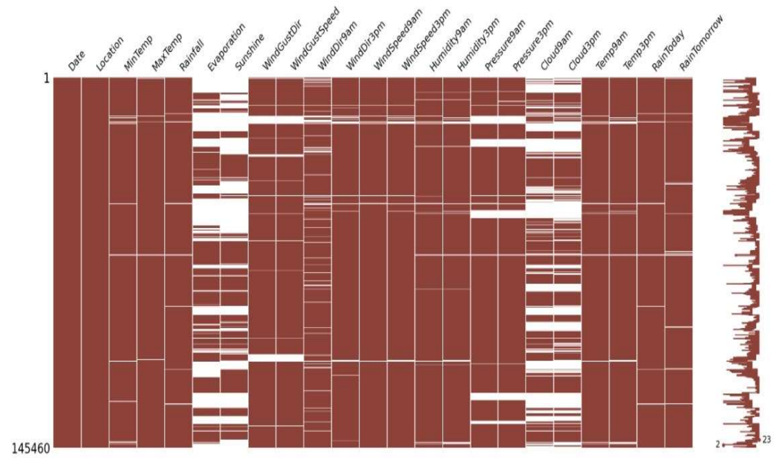
1. After loading it is important to check the complete information of data as it can indication many of the hidden information such as null values in a column or a row 2.Check whether any null values are there or not. if it is present then following can be done

```
In [65]: !pip install missingno
```

```
Requirement already satisfied: missingno in c:\users\anumo\anaconda3\lib\site-packages (0.5.2)
Requirement already satisfied: numpy in c:\users\anumo\anaconda3\lib\site-packages (from missingno) (1.24.3)
Requirement already satisfied: matplotlib in c:\users\anumo\anaconda3\lib\site-packages (from missingno) (3.7.2)
Requirement already satisfied: scipy in c:\users\anumo\anaconda3\lib\site-packages (from missingno) (1.11.1)
Requirement already satisfied: seaborn in c:\users\anumo\anaconda3\lib\site-packages (from missingno) (0.12.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\anumo\anaconda3\lib\site-packages (from matplotlib->missingno) (1.
0.5)
Requirement already satisfied: cycler>=0.10 in c:\users\anumo\anaconda3\lib\site-packages (from matplotlib->missingno) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\anumo\anaconda3\lib\site-packages (from matplotlib->missingno) (4.
25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\anumo\anaconda3\lib\site-packages (from matplotlib->missingno) (1.
4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\anumo\anaconda3\lib\site-packages (from matplotlib->missingno) (23.
1)
Requirement already satisfied: pillow>=6.2.0 in c:\users\anumo\anaconda3\lib\site-packages (from matplotlib->missingno) (9.4.0)
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in c:\users\anumo\anaconda3\lib\site-packages (from matplotlib->missingno)
(3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\anumo\anaconda3\lib\site-packages (from matplotlib->missingno)
(2.8.2)
Requirement already satisfied: pandas>=0.25 in c:\users\anumo\anaconda3\lib\site-packages (from seaborn->missingno) (2.0.3)
Requirement already satisfied: pytz>=2020.1 in c:\users\anumo\anaconda3\lib\site-packages (from pandas>=0.25->seaborn->missingn
o) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\anumo\anaconda3\lib\site-packages (from pandas>=0.25->seaborn->missin
gno) (2023.3)
Requirement already satisfied: six>=1.5 in c:\users\anumo\anaconda3\lib\site-packages (from python-dateutil>=2.7->matplotlib->m
issingno) (1.16.0)
```

2. Missing matrix: It is way of representing the data in 2-D form. It gives coloured visual summary of the data

```
In [66]: import missingno as msno
         msno.matrix(data, color=(0.55, 0.255, 0.225), fontsize=16)

Out[66]: <Axes: >
```



3.

4.Imputing data using Imputation method in sklearn.Simpleimputer a.Filling NaN values with mean, median and mode using fillna() method.

```
In [69]: # Check for missing values and calculate the percentage
         missing_percent = data.isnull().mean() * 100

         # Identify columns with more than 20% missing values
         columns_to_drop = missing_percent[missing_percent > 20].index

         # Drop columns with more than 20% missing values
         data.drop(columns=columns_to_drop, inplace=True)

         # Segregate categorical and numerical variables
         data_cat = data[['RainToday', 'WindGustDir', 'WindDir9am', 'WindDir3pm']]
         data_num = data.select_dtypes(include=['float64', 'int64'])

         # Drop the categorical variables from the main dataset
         data.drop(columns=data_cat.columns, inplace=True)
```

–

```
In [70]: data['MinTemp'].fillna(data['MinTemp'].mean(), inplace=True)
         data['MaxTemp'].fillna(data['MaxTemp'].mean(), inplace=True)
         data['Rainfall'].fillna(data['Rainfall'].mean(), inplace=True)
         data['WindGustSpeed'].fillna(data['WindGustSpeed'].mean(), inplace=True)
         data['WindSpeed9am'].fillna(data['WindSpeed9am'].mean(), inplace=True)
         data['WindSpeed3pm'].fillna(data['WindSpeed3pm'].mean(), inplace=True)
         data['Humidity9am'].fillna(data['Humidity9am'].mean(), inplace=True)
         data['Humidity3pm'].fillna(data['Humidity3pm'].mean(), inplace=True)
         data['Pressure9am'].fillna(data['Pressure9am'].mean(), inplace=True)
         data['Pressure3pm'].fillna(data['Pressure3pm'].mean(), inplace=True)
         data['Temp9am'].fillna(data['Temp9am'].mean(), inplace=True)
         data['Temp3pm'].fillna(data['Temp3pm'].mean(), inplace=True)
```

```
In [71]: cat_names = data_cat.columns
```

```
In [72]: import numpy as np
         from sklearn.impute import SimpleImputer

         # Initializing SimpleImputer for missing categorical values
         imp_mode = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
```

```
In [73]: # Filling and transforming the missing data for categorical columns
         data_cat_imputed = imp_mode.fit_transform(data_cat)

         # Converting array to DataFrame
         data_cat_imputed = pd.DataFrame(data_cat_imputed, columns=cat_names)

         # Concatenating the imputed categorical data with the original DataFrame
         data = pd.concat([data, data_cat_imputed], axis=1)
```

Activity 5: Data Visualisation

● Data visualization is where a given data set is presented in a graphical format. It helps the detection of patterns, trends and correlations that might go undetected in text-based data

. ● Understanding your data and the relationship present within it is just as important as any algorithm used to train your machine learning model. In fact, even the most sophisticated machine learning models will perform poorly on data that wasn't visualized and understood properly.

● To visualize the dataset we need libraries called Matplotlib and Seaborn.

● The Matplotlib library is a Python 2D plotting library which allows you to generate plots, scatter plots, histograms, bar charts etc. Let's visualize our data using Matplotlib and searborn library. Before diving into the code, let's look at some of the basic properties we will be using when plotting.

xlabel: Set the label for the x-axis. ylabel: Set the label for the y-axis. title: Set a title for the axes. Legend: Place a legend on the axes.

1. data.corr() gives the correlation between the column

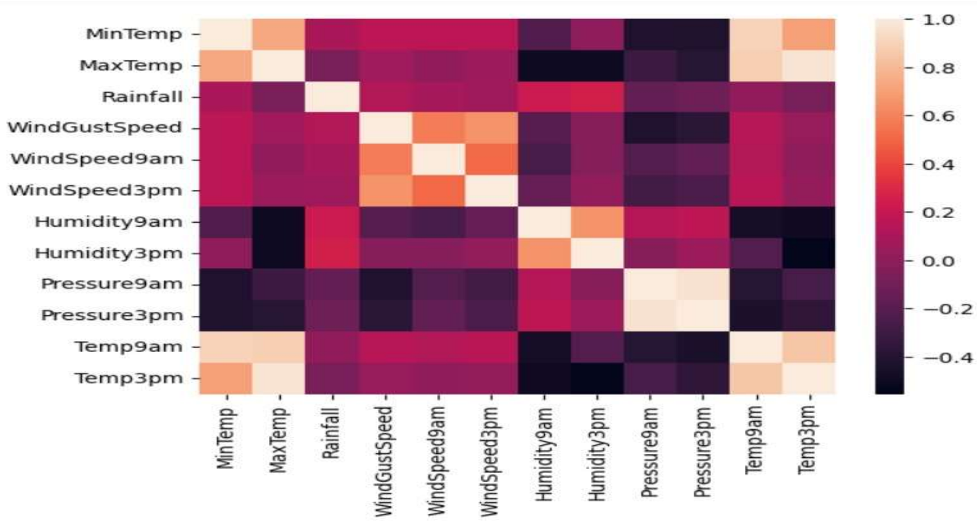
```
In [79]: numeric_data.corr()
```

Out[79]:

	MinTemp	MaxTemp	Rainfall	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Tem
MinTemp	1.000000	0.733400	0.102706	0.172553	0.173404	0.173058	-0.230970	0.005995	-0.423584	-0.433147	0.8
MaxTemp	0.733400	1.000000	-0.074040	0.065895	0.014294	0.049717	-0.497927	-0.498760	-0.308309	-0.396622	0.8
Rainfall	0.102706	-0.074040	1.000000	0.126446	0.085925	0.056527	0.221380	0.248905	-0.159055	-0.119541	0.0
WindGustSpeed	0.172553	0.065895	0.126446	1.000000	0.577319	0.657243	-0.207964	-0.025355	-0.425760	-0.383938	0.1
WindSpeed9am	0.173404	0.014294	0.085925	0.577319	1.000000	0.512427	-0.268271	-0.030887	-0.215339	-0.165388	0.1
WindSpeed3pm	0.173058	0.049717	0.056527	0.657243	0.512427	1.000000	-0.143458	0.016275	-0.277604	-0.239659	0.1
Humidity9am	-0.230970	-0.497927	0.221380	-0.207964	-0.268271	-0.143458	1.000000	0.659072	0.131503	0.176009	-0.4
Humidity3pm	0.005995	-0.498760	0.248905	-0.025355	-0.030887	0.016275	0.659072	1.000000	-0.025848	0.048695	-0.2
Pressure9am	-0.423584	-0.308309	-0.159055	-0.425760	-0.215339	-0.277604	0.131503	-0.025848	1.000000	0.959662	-0.3
Pressure3pm	-0.433147	-0.396622	-0.119541	-0.383938	-0.165388	-0.239659	0.176009	0.048695	0.959662	1.000000	-0.4
Temp9am	0.897692	0.879170	0.011069	0.145904	0.127592	0.161060	-0.469641	-0.216964	-0.397131	-0.441459	1.0
Temp3pm	0.699211	0.968713	-0.077684	0.031884	0.004476	0.027587	-0.490709	-0.555608	-0.265532	-0.360707	0.8

```
In [80]: cor = numeric_data.corr()
```

```
In [83]: import seaborn as sns
import matplotlib.pyplot as plt

# Assuming 'correlation_matrix' is your computed correlation matrix
sns.heatmap(data=correlation_matrix, xticklabels=correlation_matrix.columns.values, yticklabels=correlation_matrix.columns.values
plt.show()
```



● Correlation strength varies based on colour, lighter the colour between two variables, more the strength between the variables, darker the colour displays the weaker correlation

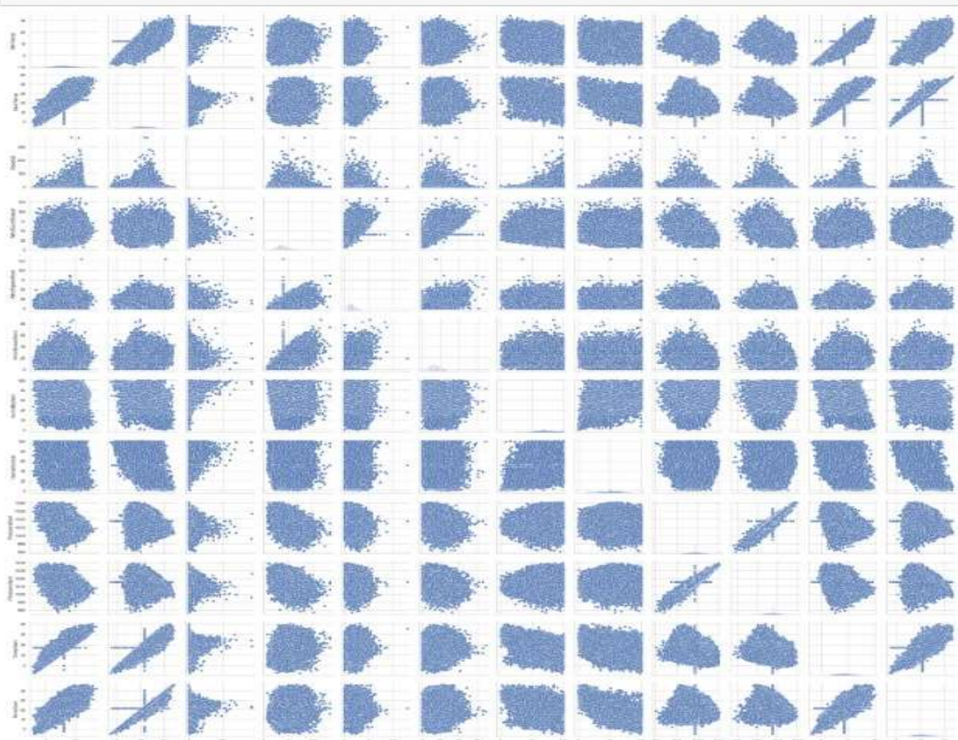
● We can see the correlation scale values on left side of the above image

Code:- sns.pairplot(data)

The output is as shown below

```
In [88]: import warnings

         # Ignore the specific UserWarning related to figure layout changes in Seaborn
         warnings.filterwarnings("ignore", category=UserWarning, module="seaborn")

         import seaborn as sns
         import matplotlib.pyplot as plt

         # Set Seaborn style or context
         sns.set(style="whitegrid")

         # Assuming 'data' is your DataFrame
         sns.pairplot(data)
         plt.tight_layout()
         plt.show()
```

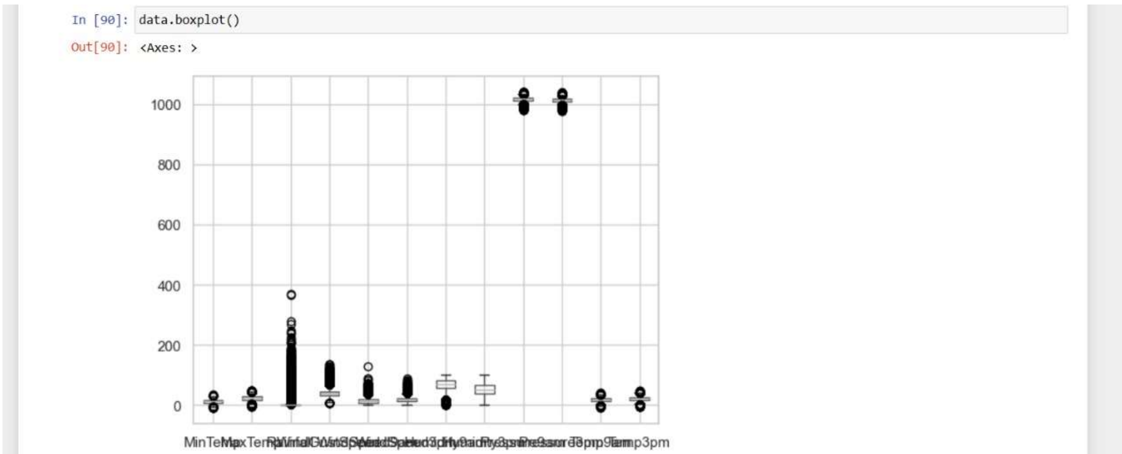

```
In [88]: import warnings

         # Ignore the specific UserWarning related to figure layout changes in Seaborn
         warnings.filterwarnings("ignore", category=UserWarning, module="seaborn")

         import seaborn as sns
         import matplotlib.pyplot as plt

         # Set Seaborn style or context
         sns.set(style="whitegrid")

         # Assuming 'data' is your DataFrame
         sns.pairplot(data)
         plt.tight_layout()
         plt.show()
```

Pair plot usually gives pair wise relationships of the columns in the dataset From the above pairplot we infer that

1.from the above plot we can draw inferences such as linearity and strength between the variables
2.how features are correlated(positive, neutral and negative)

3.Box Plot: jupyter has a built-in function to create boxplot called boxplot(). A boxplot plot is a type of plot that shows the spread of data in all the quartiles.



Activity 6: Splitting the Dataset into Dependent and Independent variable

● In machine learning, the concept of dependent variable (y) and independent variables(x) is important to understand. Here, Dependent variable is nothing but output in dataset and independent variable is all inputs in the dataset.

● With this in mind, we need to split our dataset into the matrix of independent variables and the vector or dependent variable. Mathematically, Vector is defined as a matrix that has just one column. To read the columns, we will use iloc of pandas (used to fix the indexes for selection) which takes two parameters — [row selection, column selection]. Let's split our dataset into independent and dependent variables.

y = data['RainTomorrow'] – independent

x = data.drop('RainTomorrow',axis=1

Activity 7: Feature Scaling There is huge disparity between the x values so let us use feature scaling. Feature scaling is a method used to normalize the range of independent variables or features of data.

In [95]:
```
sc = StandardScaler()
```

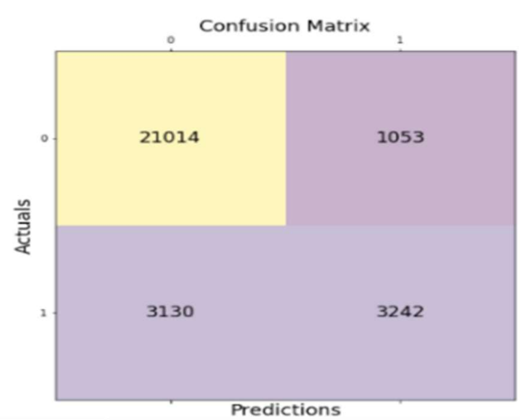
In [98]:
```
from sklearn.preprocessing import StandardScaler
import pandas as pd

# Assuming 'data' is your DataFrame
numeric_columns = data.select_dtypes(include=['float64', 'int64']).columns

# Extract only numeric columns
x = data[numeric_columns]

# Initialize the StandardScaler
sc = StandardScaler()

# Fit and transform only on numeric data
x_scaled = sc.fit_transform(x)
```

In [100]:
```
x = pd.DataFrame(x, columns=names)
```

In [116]:
```
import xgboost
import sklearn.ensemble
import sklearn.svm
import sklearn.tree
import sklearn.ensemble
import sklearn.linear_model

# Models initialization
XGBoost = xgboost.XGBRFClassifier()
Rand_forest = sklearn.ensemble.RandomForestClassifier()
svm = sklearn.svm.SVC()
Dtree = sklearn.tree.DecisionTreeClassifier()
GBM = sklearn.ensemble.GradientBoostingClassifier()
log = sklearn.linear_model.LogisticRegression()
```

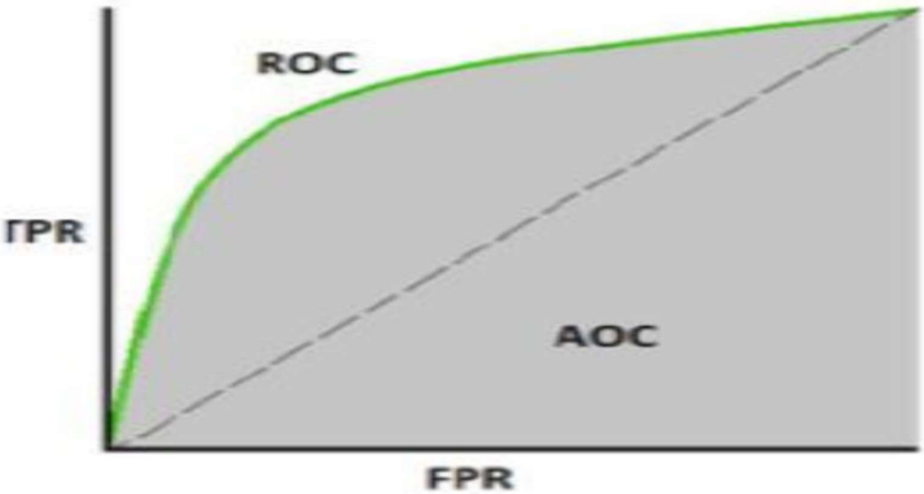
In [124]:
```
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)
```

In [127]:
```
from xgboost import XGBClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression

# Assuming you have imported the necessary libraries and created instances of the models
XGBoost = XGBClassifier()
Rand_forest = RandomForestClassifier()
svm = SVC()
Dtree = DecisionTreeClassifier()
GBM = GradientBoostingClassifier()
log = LogisticRegression()

# Assuming x_train and y_train are your training data
XGBoost.fit(x_train, y_train)
Rand_forest.fit(x_train, y_train)
svm.fit(x_train, y_train)
Dtree.fit(x_train, y_train)
GBM.fit(x_train, y_train)
log.fit(x_train, y_train)
```



```
]: print(conf_matrix)
   print("Accuracy:",Accuracy)
   print("Precesion:",Precesion)
   print("Recall:",Recall)
   print("F1-score:",F1_score)
```

```
]: auc = metrics.roc_auc_score(y_test,y_pred)

   fpr, tpr, thresolds = metrics.roc_curve(y_test,y_pred)

   plt.figure(figsize=(12, 10), dpi=80)
   plt.axis('scaled')
   plt.xlim([0, 1])
   plt.ylim([0, 1])
   plt.title("AUC & ROC Curve")
   plt.plot(fpr, tpr, 'v')
   plt.fill_between(fpr, tpr, facecolor='blue', alpha=0.8)
   plt.text(1, 0.05, 'AUC = %0.4f' % auc, ha='right', fontsize=10, weight='bold', color='black')
   plt.xlabel("False Positive Rate")
   plt.ylabel("True Positive Rate")
   plt.show()
```



Activity 3: Save the Model

After building the model we have to save the model.

Pickle in **Python** is primarily **used** in serializing and deserializing a **Python** object structure. In other words, it's the process of converting a **Python** object into a byte stream to store it in a file/database, maintain program state across sessions, or transport data over the network. wb indicates write method and rd indicates read method.

This is done by the below code

saving the model

```
[29]: import pickle

[ ]: pickle.dump(model,open('rainfall.pkl','wb')) # model
     pickle.dump(le,open('encoder.pkl','wb'))      # encoder saving
     pickle.dump(imp_mode,open('impter.pkl','wb'))# imputer saving
     pickle.dump(sc,open('scale.pkl','wb'))        # scaling the data
```

Milestone 4 : Application Building

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

- Building HTML Pages
- Building server side script

Activity 1: Build HTML Code

o In this HTML page, we will create the front end part of the web page. In this page we will accept input from the user and Predict the values

. For more information regarding HTML https://www.w3schools.com/html/ In our project we have 3 HTML files ,they are

1.inex.html

2.chance.html

3.noChance.htm

index.html

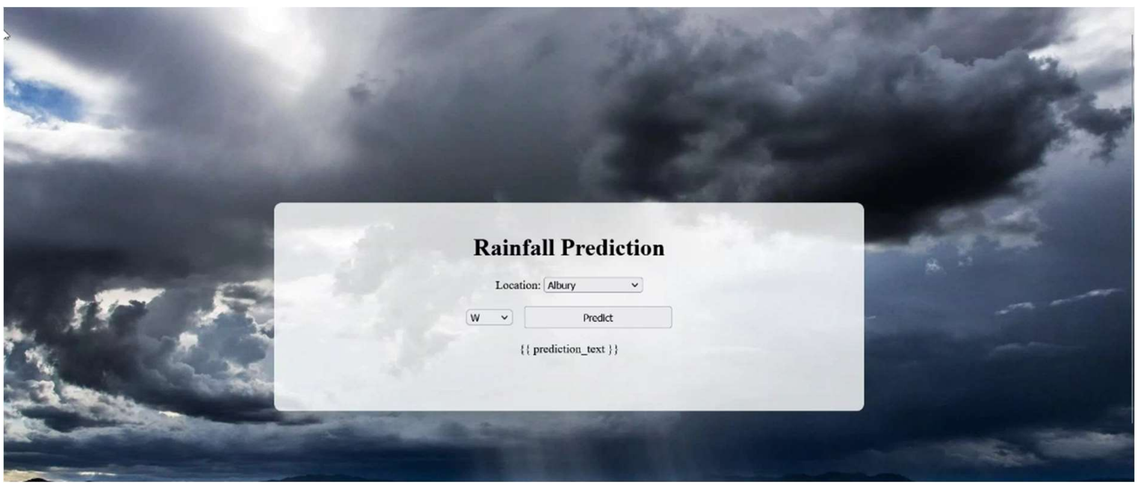
```
index > html > head > style > .login
<!DOCTYPE html>
<html>

<head>
    <meta charset="UTF-8">
    <title>Rainfall Prediction</title>
    <style>
        body {
            background: url('https://wallpaperaccess.com/full/701614.jpg') no-repeat center center fixed;
            color: black;
            background-size: cover;
        }

        .login {
            text-align: center;
            padding: 20px;
            background-color: rgba(255, 255, 255, 0.8);
            margin: 20% auto;
            width: 50%;
            border-radius: 10px;
        }

        /* Add any additional styles as needed */
    </style>
</head>

<body>

    <div class="login">
        <h1>Rainfall Prediction</h1>
```

```html
<option value=21>Moree</option>
<option value=24>Newcastle</option>

<option value=26>NorahHead</option>

<option value=27>NorfolkIsland</option>

<option value=30>Penrith</option>

<option value=34>Richmond</option>
 <option value=37>Sydney</option>

<option value=38>Sydney Airport</option>
< option value-42>Waggallagga</option>

<option value=45>Williamtown</option>

<option value=47>Wollongong</option>
<option value=9>Canberra</option>

<option value=40>Tuggeranong</option>

<option value-23>MountGinini</option>
<option value=5>Ballarat</options

<option value=6>Bendigo</option> <option value-35>Sale</option>

<option value=19>Helbourne Airport</option>

<option value=18>Melbourne</options <option value=20>Mildura</option>

<option value=25>Nhil</option>

<option value=33>Portland</option>
<option value=44>Watsonia</option>
```

```html
            </select>
            <br><br>

            <select id="WindDirection" name="WindDirection">
                <option value="14">W</option>
                <option value="15">WNW</option>
                <option value="0">WSW</option>
                <option value="7">NE</option>
                <option value="13">NNW</option>
                <option value="10">N</option>
                <option value="2">NNE</option>
                <option value="1">SW</option>
                <option value="6">ENE</option>
                <option value="11">SSE</option>
                <option value="12">S</option>
                <option value="9">MW</option>
                <option value="3">SE</option>
                <option value="8">ESE</option>
                <option value="5">E</option>
                <option value="4">SSM</option>
            </select>
              

            <button type="submit" style="height: 30px; width: 200px;">Predict</button>
        </form>

        <br>

        <br><br>

        <img src="data:image/png;base64, {{ url_3 }}" alt="Prediction Image 1" height="180" width="233"
            onerror="this.style.display='none'" />
        <img src="data:image/png;base64, {{ url_1 }}" alt="Prediction Image 2" height="180" width="233"
            onerror="this.style.display='none'" />
        <img src="data:image/png;base64, {{ url_4 }}" alt="Prediction Image 3" height="180" width="233"
            onerror="this.style.display='none'" />
```

```
            <button type="submit" style="height: 30px; width: 200px;">Predict</button>
        </form>

        <br>

        <br><br>

        <img src="data:image/png;base64, {{ url_3 }}" alt="Prediction Image 1" height="180" width="233"
            onerror="this.style.display='none'" />
        <img src="data:image/png;base64, {{ url_1 }}" alt="Prediction Image 2" height="180" width="233"
            onerror="this.style.display='none'" />
        <img src="data:image/png;base64, {{ url_4 }}" alt="Prediction Image 3" height="180" width="233"
            onerror="this.style.display='none'" />

        <br><br>
        <img src="data:image/png;base64, {{ url_2 }}" alt="Prediction Image 4" height="150" width="711"
            onerror="this.style.display='none'" />
    </div>

</body>

</html>
```

The html page looks likes



No chance html:

```
chance.html > html
<!DOCTYPE html>

<html >

<head>

<meta charset="UTF-8">

<title>Rainfall prediction</title>

</head>

<body background="https://wnavprd.blob.core.windows.net/images/guide/chris-seufert-cape-cod-rain-beach-1400-110-1.jpg" text="black">

<div class="login">

</body>

</html>
```

The html page looks likes

Chance.html:

```html
<!DOCTYPE html>

<html >

<head>

<meta charset="UTF-8">

<title>Rainfall prediction</title>

</head>

<body background="https://wnavprd.blob.core.windows.net/images/guide/chris-seufert-cape-cod-rain-beach-1400-110-1.jpg" text="black">

<div class="login">

</body>

</html>
```

The html page looks likes

Activity 2: Main Python Script

Let us build app.py flask file which is a web framework written in python for server-side scripting. Let's see step by step procedure for building the backend application. In order to develop web api with respect to our model, we basically use Flask framework which is written in python. Line 1-3 We are importing necessary libraries like Flask to host our model request Line 4 Initialise the Flask application Line 5 Loading the model using pickle Line 7 Routes the api url Line 9 Rendering the template. This helps to redirect to home page. In this home page ,we give our input and ask the model to predict Line 19 we are taking the inputs from the form Line 21-23 Feature Scaling the inputs Line 24 Predicting the values given by the user Line 27-30 If output is false render noChance template If output is True render chance template Line 31 The value of __name__ is set to __main__ when module run as main program other wise it is set to name of the module

```python
1   import numpy as np
2   import pickle
3   from flask import Flask, request, render_template
4
5   app = Flask(__name__)
6
7   model = pickle.load(open("C:\Users\anumo\Downloads/Dataset  -  Dataset.csv", 'rb'))
8   scale = pickle.load(open("C:/Users/SmartbridgePC/Desktop/AIML/Guided projects/rainfall_prediction/scale.pkl", 'rb'))
9
10  @app.route('/')
11  def home():
12      return render_template('index.html')
13
14  @app.route('/predict', methods=["POST", "GET"])
15  def predict():
16      if request.method == "POST":
17          input_features = [float(x) for x in request.form.values()]  # assuming form values are numeric
18
19          # Use the input_features to make predictions
20          features_values = [np.array(input_features)]
21          names = ['Location', 'MinTemp', 'MaxTemp', 'Rainfall', 'WindGustSpeed',
22                   'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm', 'Pressure9am', 'Pressure3pm',
23                   'Temp9am', 'Temp3pm', 'RainToday', 'WindGustDir', 'WindDir9am', 'WindDir3pm',
24                   'year', 'month', 'day']
25
26          data = pd.DataFrame(features_values, columns=names)
27          data = scale.transform(data)
28          data = pd.DataFrame(data, columns=names)
29
30          # predictions using the loaded model file
31          prediction = model.predict(data)
32          pred_prob = model.predict_proba(data)
33
34          print(prediction)
```

```python
        # Use the input_features to make predictions
        features_values = [np.array(input_features)]
        names = ['Location', 'MinTemp', 'MaxTemp', 'Rainfall', 'WindGustSpeed',
                 'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm', 'Pressure9am', 'Pressure3pm',
                 'Temp9am', 'Temp3pm', 'RainToday', 'WindGustDir', 'WindDir9am', 'WindDir3pm',
                 'year', 'month', 'day']

        data = pd.DataFrame(features_values, columns=names)
        data = scale.transform(data)
        data = pd.DataFrame(data, columns=names)

        # predictions using the loaded model file
        prediction = model.predict(data)
        pred_prob = model.predict_proba(data)

        print(prediction)

        if prediction[0] == "Yes":
            return render_template("chance.html")
        else:
            return render_template("nochance.html")

if __name__ == "__main__":
    app.run(debug=True)
```

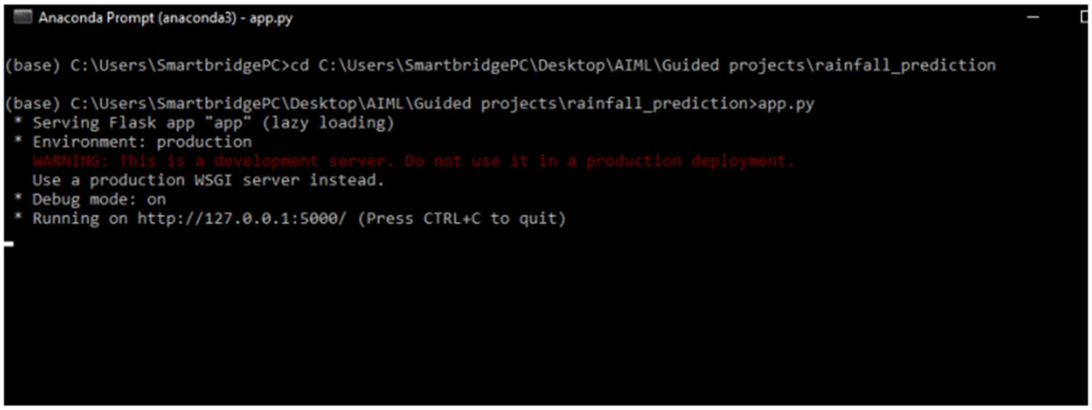
Activity 3:

Run the App

o Open anaconda prompt from the start menu

o Navigate to the folder where your python script is.

o Now type “python app.py” command Navigate to the localhost where you can view your web page,Then it will run on local host:5000



Activity 4:

● Copy the http link and paste it in google link tab,it will display the form page

● Enter the values as per the form and click on predict buttion

● It will redirect to the page based on prediction outpu