EXPLORATORY ANALYSIS OF RAINFALL DATA IN INDIA FOR AGRICULTURE

MENTOR NAME :- **SRI L. LANKA LAKSHMI NARAYANA**

TEAM ID :- **LTVIP2026TMIDS46425**

TEAM LEADER

NAME : M.CHINNA REDDY
MAIL : youthreddy93@gmail.com
ID NO : SBAP0040482

TEAM MEMBERS

MEMBER - 1

NAME : CH.LAKSHMI GANAPATHI
MAIL : ganapathichelluriganapathi@gmail.com
ID NO : SBAP0040446

MEMBER - 2

MEMBER - 2

NAME : U.S.V SATYANARAYANA
MAIL : saiuppalapati068@gmail.com
ID NO : SBAP0040414

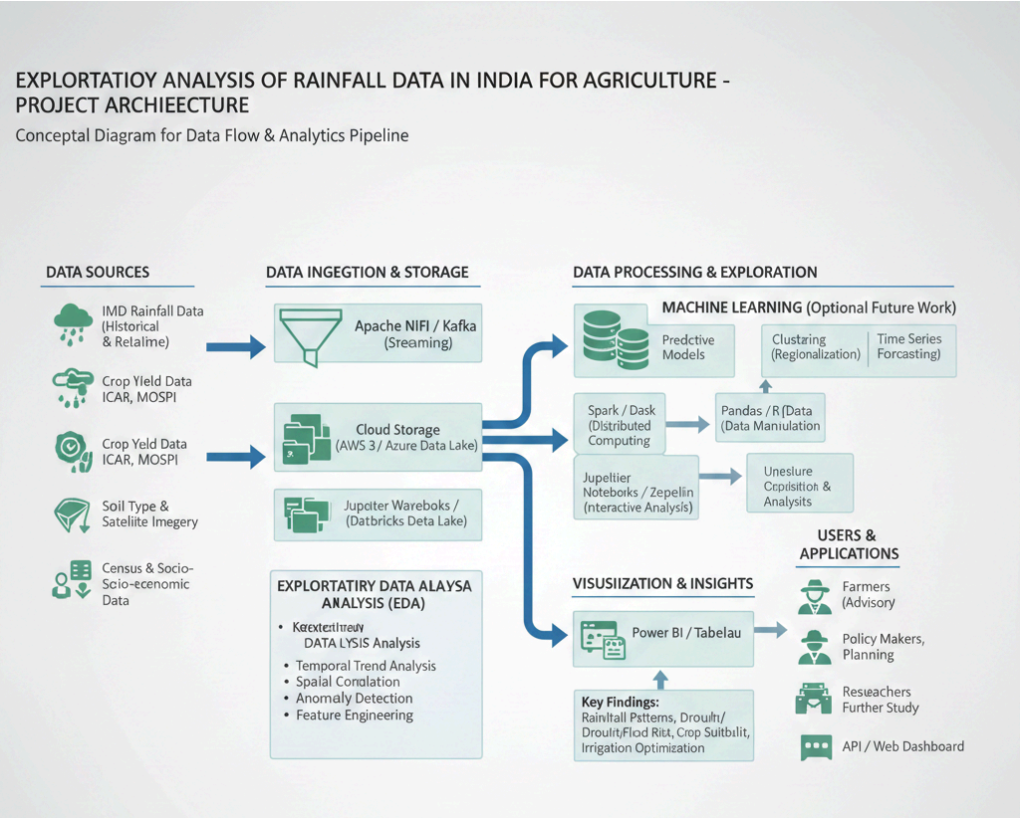
MEMBER - 3

NAME : K.N.S.V SOMESWAR RAO
MAIL : someshkundeti@gmail.com
ID NO : SBAP0040453

Introduction

Agriculture is the backbone of the Indian economy, with a significant portion of the population relying on it for livelihood. However, Indian agriculture is heavily dependent on rainfall, making it vulnerable to the erratic and unpredictable nature of monsoons. Understanding rainfall patterns and their variability is crucial for informed agricultural planning, risk mitigation, and ensuring food security. This project, "Exploratory Analysis of Rainfall Data in India for Agriculture," aims to delve into historical rainfall data across various regions of India to identify trends, anomalies, and spatial-temporal relationships relevant to agricultural practices. By leveraging data analysis techniques, we seek to uncover insights that can help farmers, policymakers, and researchers make better decisions regarding crop selection, irrigation strategies, and disaster preparedness.

The project will involve cleaning, transforming, and visualizing large datasets of rainfall measurements, correlating them with agricultural output (where feasible), and identifying regions most susceptible to rainfall variability. Ultimately, this exploratory analysis will lay the groundwork for more advanced predictive modeling and decision support systems to enhance the resilience and productivity of Indian agriculture.



Project Objectives

By the end of this project:

- You'll be able to understand the problem to classify if it is a regression or a classification kind of problem.
- You will be able to know how to pre-process/clean the data using different data preprocessing techniques.
- You will be able to analyze or get insights into data through visualization.
- Applying different algorithms according to the dataset and based on visualization.
- You will be able to know how to find the accuracy of the model.
- You will be able to know how to build a web application using the Flask framework.

Project Flow

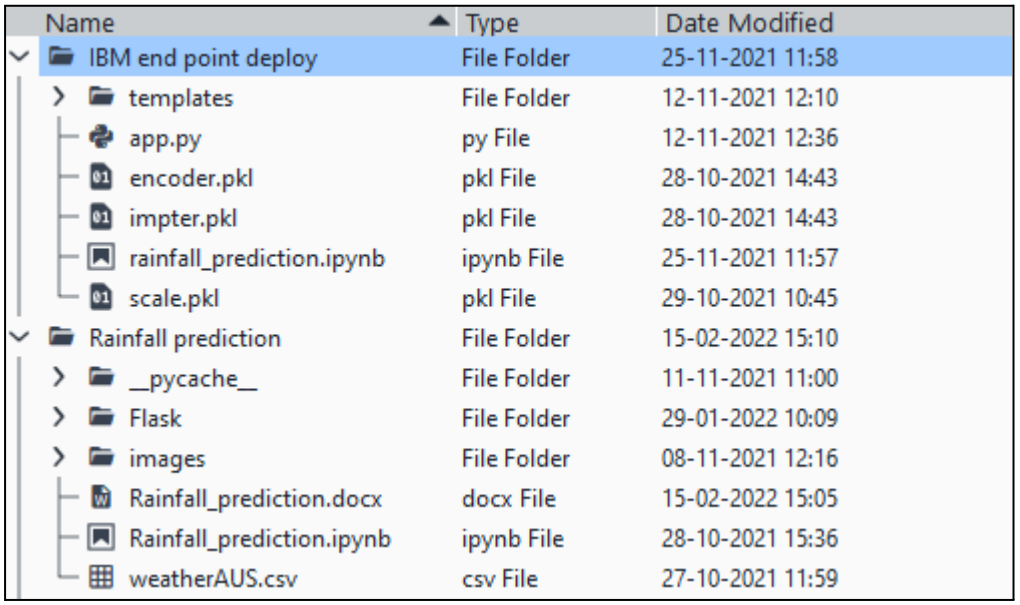
- User interacts with the UI (User Interface) to enter the input values
- Entered input values are analyzed by the model which is integrated
- Once the model analyses the input the prediction is showcased on the UI

To accomplish this, we have to complete all the activities and tasks listed below

- Data Collection.
  - Collect the dataset or Create the dataset
- Data Pre-processing.
  - Import the Libraries.
  - Importing the dataset.
  - Checking for Null Values.
  - Data Visualization.
  - Taking care of Missing Data.
  - Feature Scaling.
  - Splitting Data into Train and Test.
- Model Building
  - Import the model building Libraries
  - Initializing the model
  - Training and testing the model
  - Evaluation of Model
  - Save the Model
- Application Building
  - Create an HTML file
  - Build a Python Code

Project Structure

The Project folder that contains files as shown below



- IBM endpoint deploy which contains template files, app.py, and .pkl files which are used for application building.
- Rainfall_prediction.ipynb is a model training code file of IBM Watson.
- Rainfall prediction consists of flask files for application building in local system and .ipynb model training code file.
- We need the model which is saved and the saved model in this content is Rainfall.pkl.
- Templates folder which contains index.HTML file, chance.HTML file, noChance.HTML file.
- Scale.pkl for scaling,encoder.pkl file for encoding the categorical data,imputer.pkl file for filling out the missing values.

Data Collection

ML depends heavily on data, without data, it is impossible for an "AI" to learn. It is the most crucial aspect that makes algorithm training possible. In Machine Learning projects, we need a training data set. It is the actual data set used to train the model for performing various actions.

Download the dataset

- You can collect datasets from different open sources like kaggle.com, data.gov, UCI machine learning repository etc.
- Please refer to the link given below to download the data set and to know about the dataset

Data Pre-processing

Data Pre-processing includes the following main tasks

- Import the Libraries.
- Importing the dataset.
- Checking for Null Values.
- Data Visualization.
- Feature Scaling.
- Splitting Data into Train and Test.

Importing 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
paste the image

```
# 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
```

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")

```
data = pd.read_csv("C:\\Users\\SmartbridgePC\\Desktop\\AIML\\Guided projects\\rainfall_prediction\\weatherAUS.csv")
```

Analyse the data

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

```
[2]: data = pd.read_csv("C:\\Users\\SmartbridgePC\\Desktop\\AIML\\Guided projects\\rainfall_prediction\\weatherAUS.csv")

[3]: data.head()
```

Out[3]:

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

The output is as shown below

```
]: data.describe() # gives the descriptive statistics of the data
```

]:

	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-

```
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
```

```
data.shape  # gives the dimensions of the data
```

```
(145460, 19)
```

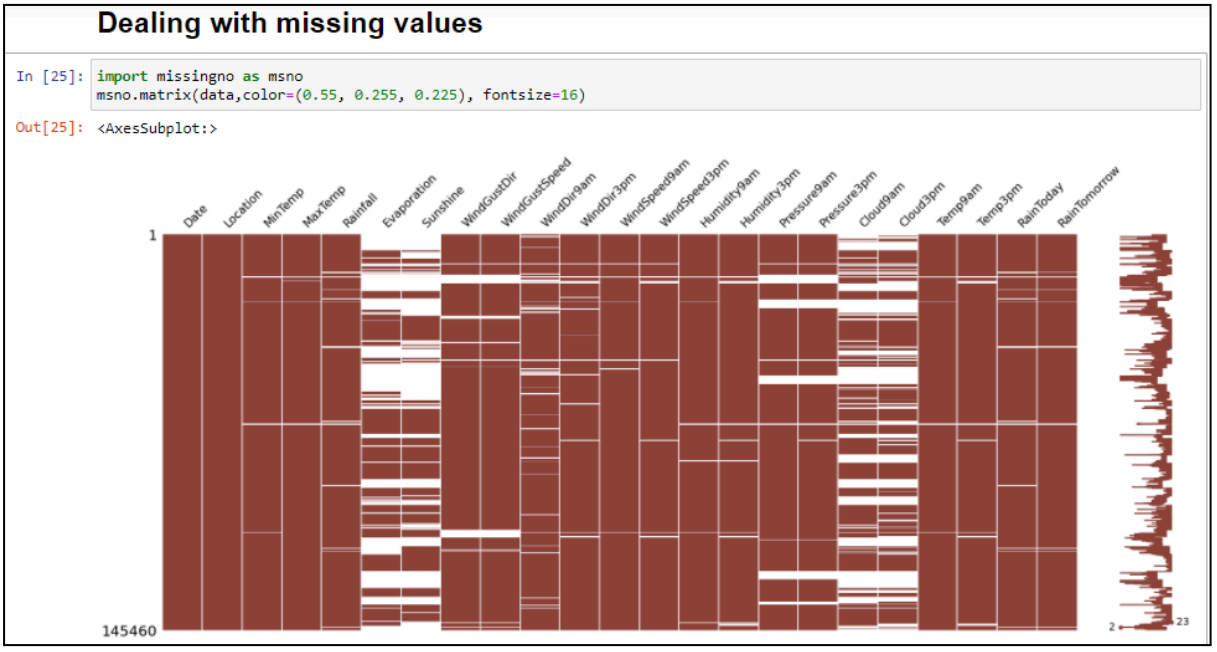
data. shape: give the information about the dimension of the dataset, such as no. of rows and no. of columns

Handling Missing Values

1. After loading it is important to check the complete information of data as it can indicate 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 the following can be done,

```
5]: data.isnull().sum()

5]: Date                0
    Location            0
    MinTemp          1485
    MaxTemp          1261
    Rainfall         3261
    Evaporation     62790
    Sunshine        69835
    WindGustDir     10326
    WindGustSpeed   10263
    WindDir9am      10566
    WindDir3pm       4228
    WindSpeed9am     1767
    WindSpeed3pm     3062
    Humidity9am      2654
    Humidity3pm      4507
    Pressure9am     15065
    Pressure3pm     15028
    Cloud9am        55888
    Cloud3pm        59358
    Temp9am          1767
    Temp3pm          3609
    RainToday        3261
    RainTomorrow     3267
    dtype: int64
```

3. **Missing matrix**: It is a way of representing the data in the 2-D form. It gives a colored visual summary of the data, Missingno matrix is used to identify the missing data in the dataset, which is represented as white color gaps



4. Imputing data using the Imputation method in sklearn.Simpleimputer

- Filling NaN values with mean, median, and mode using fillna() method.
- We are using the mean method to fill the missing data, where the data type of the columns are numeric,

- We are using the simple imputer method, which is in sklearn library from the module impute.

In impute method, for filling missing values of categorical data type we are using the most frequent method

```
[5]: # removing columns with more than 20% missing values and segregatting cat and num variables
     data_cat = data[['RainToday','WindGustDir','WindDir9am','WindDir3pm']]
     data.drop(columns=['Evaporation','Sunshine','Cloud9am','Cloud3pm'],axis=1,inplace=True)
     data.drop(columns=['RainToday','WindGustDir','WindDir9am','WindDir3pm'],axis=1,inplace=True)

[6]: # filling the missing data of numeric variables with mean
     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)

[7]: # loading the names of categorical columns
     cat_names = data_cat.columns

[8]: # intializing the simple imputer for missing categorical values
     import numpy as np
     from sklearn.impute import SimpleImputer
     imp_mode = SimpleImputer(missing_values=np.nan, strategy='most_frequent')

[9]: # fitting and transforming the missing data
     data_cat = imp_mode.fit_transform(data_cat)

10]: # converting array to dataframe
     data_cat = pd.DataFrame(data_cat,columns=cat_names)

11]: # concatinating the categorical and numeric data
     data = pd.concat([data,data_cat],axis=1)
```

From the heat map, we see that there are missing values in the dataset

Data Visualization

- 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 that allows you to generate plots, scatter plots, histograms, bar charts etc.

Let's visualize our data using Matplotlib and seaborn 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 columns

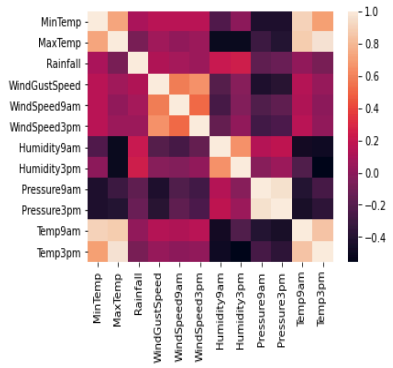
Correlation is a statistical term describing the degree to which two variables move in coordination with one another. If the two variables move in the same direction, then those variables are said to have a positive correlation. If they move in opposite directions, then they have a negative correlation.

```
data.corr()
```

	Location	MinTemp	MaxTemp	Rainfall	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am	Pressur
Location	1.000000	-0.006195	-0.020490	-0.003456	0.069158	0.077025	0.064150	-0.002063	0.011040	0.036480	0.0
MinTemp	-0.006195	1.000000	0.733919	0.103315	0.173361	0.174938	0.174187	-0.232382	0.005930	-0.424336	-0.4
MaxTemp	-0.020490	0.733919	1.000000	-0.074204	0.066298	0.014580	0.050374	-0.499789	-0.499714	-0.309077	-0.3
Rainfall	-0.003456	0.103315	-0.074204	1.000000	0.127263	0.085981	0.056766	0.221395	0.249617	-0.159677	-0.1
WindGustSpeed	0.069158	0.173361	0.066298	0.127263	1.000000	0.577790	0.658334	-0.209226	-0.025751	-0.426586	-0.3
WindSpeed9am	0.077025	0.174938	0.014580	0.085981	0.577790	1.000000	0.513067	-0.269031	-0.031001	-0.215159	-0.1
WindSpeed3pm	0.064150	0.174187	0.050374	0.056766	0.658334	0.513067	1.000000	-0.144264	0.015782	-0.277469	-0.2
Humidity9am	-0.002063	-0.232382	-0.499789	0.221395	-0.209226	-0.269031	-0.144264	1.000000	0.659865	0.131593	0.1
Humidity3pm	0.011040	0.005930	-0.499714	0.249617	-0.025751	-0.031001	0.015782	0.659865	1.000000	-0.025781	0.0
Pressure9am	0.036480	-0.424336	-0.309077	-0.159677	-0.426586	-0.215159	-0.277469	0.131593	-0.025781	1.000000	0.9
Pressure3pm	0.046338	-0.434021	-0.397412	-0.120366	-0.384622	-0.165038	-0.239725	0.176164	0.048554	0.959878	1.0
Temp9am	-0.015597	0.897998	0.880085	0.011385	0.146761	0.128769	0.162143	-0.471141	-0.217573	-0.397746	-0.4
Temp3pm	-0.022712	0.699824	0.969733	-0.077555	0.032245	0.005023	0.028444	-0.492441	-0.555778	-0.266291	-0.3
RainToday	-0.004911	0.055644	-0.226474	0.500279	0.148264	0.100563	0.078361	0.348843	0.370561	-0.179226	-0.1
WindGustDir	-0.005055	-0.136317	-0.212210	0.044855	0.137914	0.009762	0.084012	0.067956	0.063964	-0.120630	-0.0
WindDir9am	-0.004434	-0.029638	-0.212589	0.085140	0.074638	0.109916	0.111072	0.088842	0.148893	-0.050314	0.0
WindDir3pm	0.008325	-0.158958	-0.181344	0.047905	0.136674	0.050200	0.089824	0.026077	-0.007182	-0.133633	-0.0

2.Heat-map: A heatmap is a graphical representation of data that uses a system of color-coding to represent different values.it is used to identify the correlation between the columns using a visual manner.

```
In [12]: cor = data.corr()

In [13]: sns.heatmap(data=cor,xticklabels=cor.columns.values,yticklabels=cor.columns.values)
Out[13]: <AxesSubplot:>
```



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

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

2.Pair Plot: Plot pairwise relationships in a dataset.

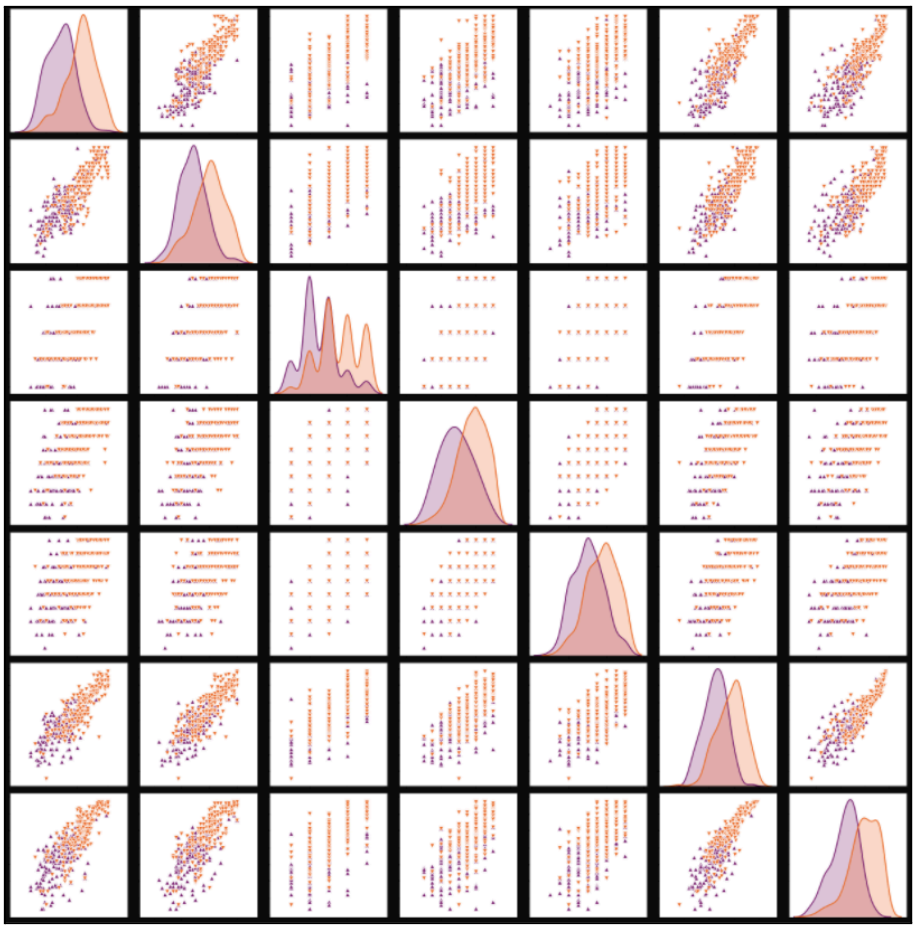
Pair plot is used to understand the best set of features to explain a relationship between two variables or to form the most separated clusters. It also helps to form some simple classification models by drawing some simple lines or making a linear separation in our data-set.

- By default, this function will create a grid of Axes such that each numeric variable in data will be shared across the y-axes across a single row and the x-axes across a single column. The diagonal plots are treated differently: a univariate distribution plot is drawn to show the marginal distribution of the data in each column.

- We implement this using the below code

Code:- sns.pairplot(data)

The output is as shown below-



Pair plot usually gives pair wise relationships of the columns in the dataset

From the above pairplot we infer that

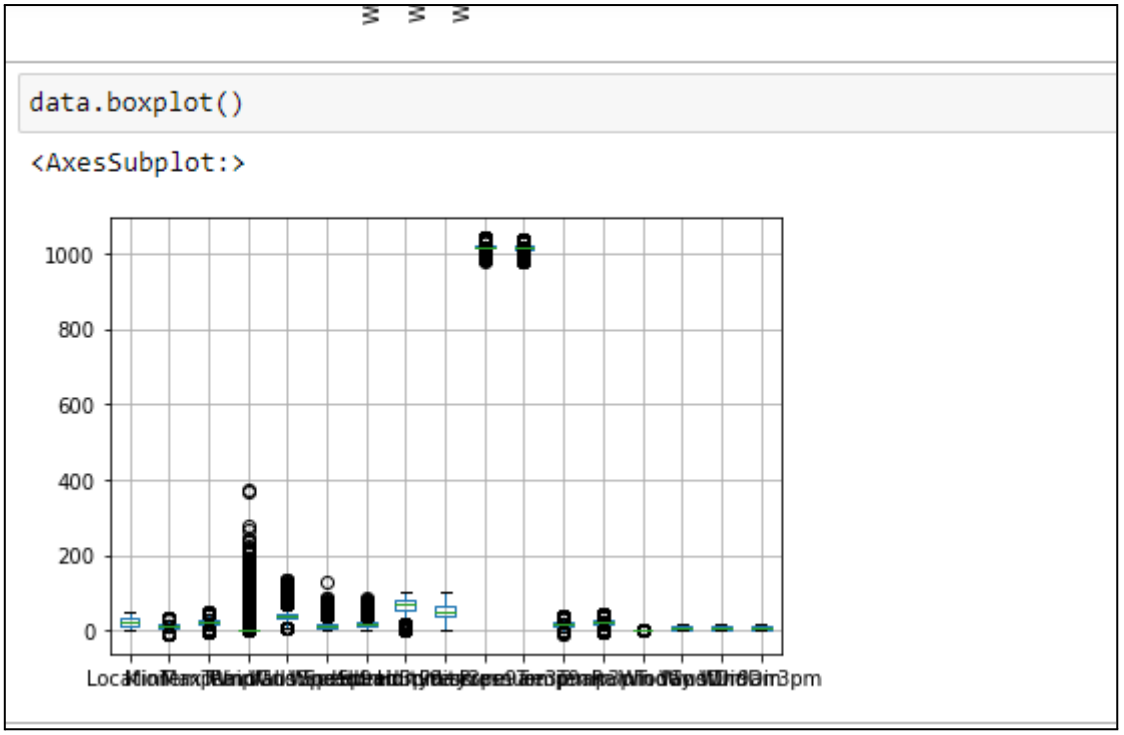
1. from the above plot we can draw inferences such as linearity and strength between the variables

2. how features are correlated(positive, neutral and negative)

3. Box Plot:

Box-plot is a type of chart often used in explanatory data analysis. Box plots visually show the distribution of numerical data and skewness by displaying the data quartiles (or percentiles) and averages.

Box plots are useful as they show the average score of a data set. The median is the average value from a set of data and is shown by the line that divides the box into two parts. Half the scores are greater than or equal to this value and half are less.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



From the above box plot, we infer how the data points are spread and the existence of the outliers

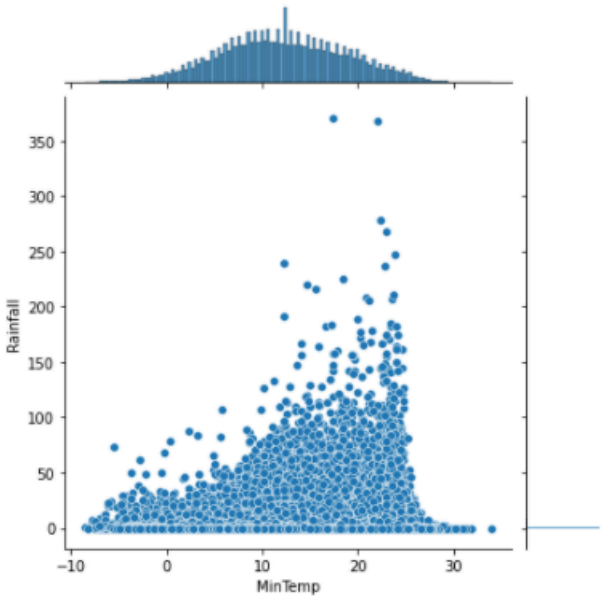
4. Joint plot:

A Joint plot comprises two plots. Out of the three, one plot displays a bivariate graph that shows how the dependent variable(Y) varies with the independent variable(X). Another plot is placed horizontally at the top of the bivariate graph and it shows the distribution of the independent variable(X).

Here we are comparing the variables such as “Min Temp” and “Rainfall”

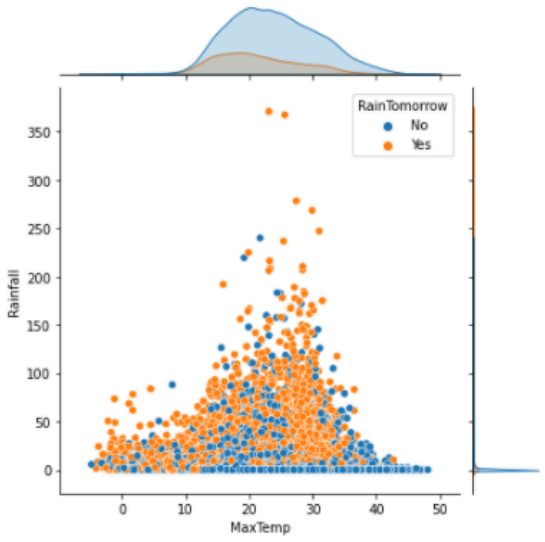
The column “MinTemp” is following the normal distribution.

```
In [17]: sns.jointplot(data["MinTemp"],data['Rainfall'])

C:\Users\SmartbridgePC\anaconda4\lib\site-packages\seaborn\_decorators.py:36: FutureWarn
yword args: x, y. From version 0.12, the only valid positional argument will be `data`,
explicit keyword will result in an error or misinterpretation.
  warnings.warn(

Out[17]: <seaborn.axisgrid.JointGrid at 0x20b3aa3bfa0>
```



Joinplot with hue function:-

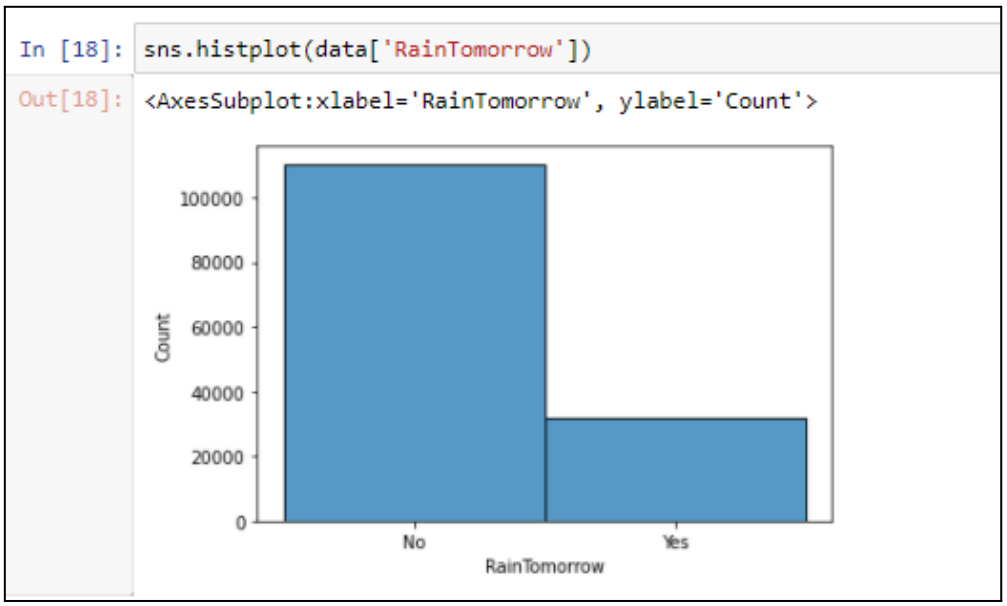
```
In [23]: sns.jointplot(data['MaxTemp'],data['Rainfall'],hue=data['RainTomorrow'])

C:\Users\SmartbridgePC\anaconda4\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass t
yword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing
explicit keyword will result in an error or misinterpretation.
  warnings.warn(

Out[23]: <seaborn.axisgrid.JointGrid at 0x20b41fd9fd0>
```



This plot is able to generate the relationship between 3 variables, in the above plot we can see the relation between MaxTemp,Rainfall and class segregation of the data

5 .Hist plot:-

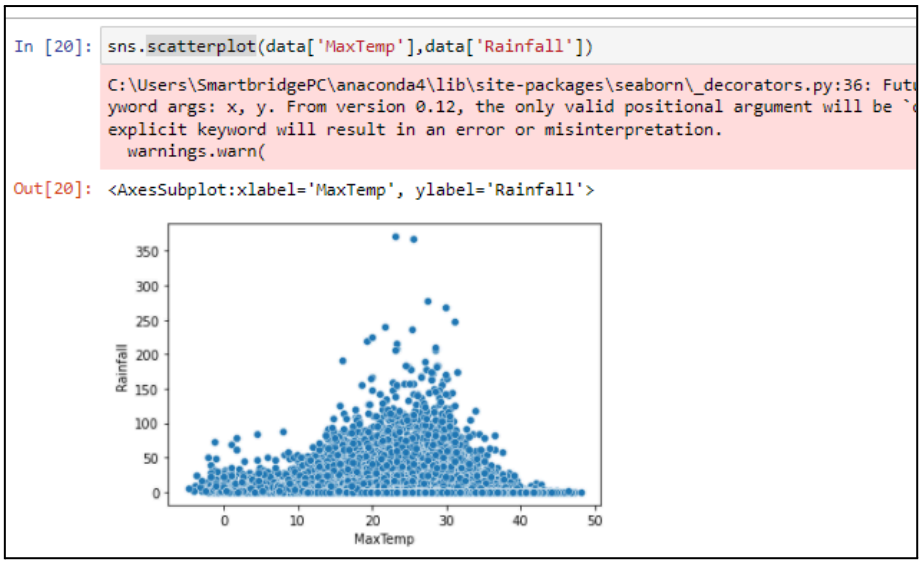
Plot univariate or bivariate histograms to show distributions of datasets. A histogram is a classic visualization tool that represents the distribution of one or more variables by counting the number of observations that fall within discrete bins.



From the above graph we can infer that,RainTomorrow column has more no.of NO's and less no.fo Yes categories.

6.Scatter-plot:-

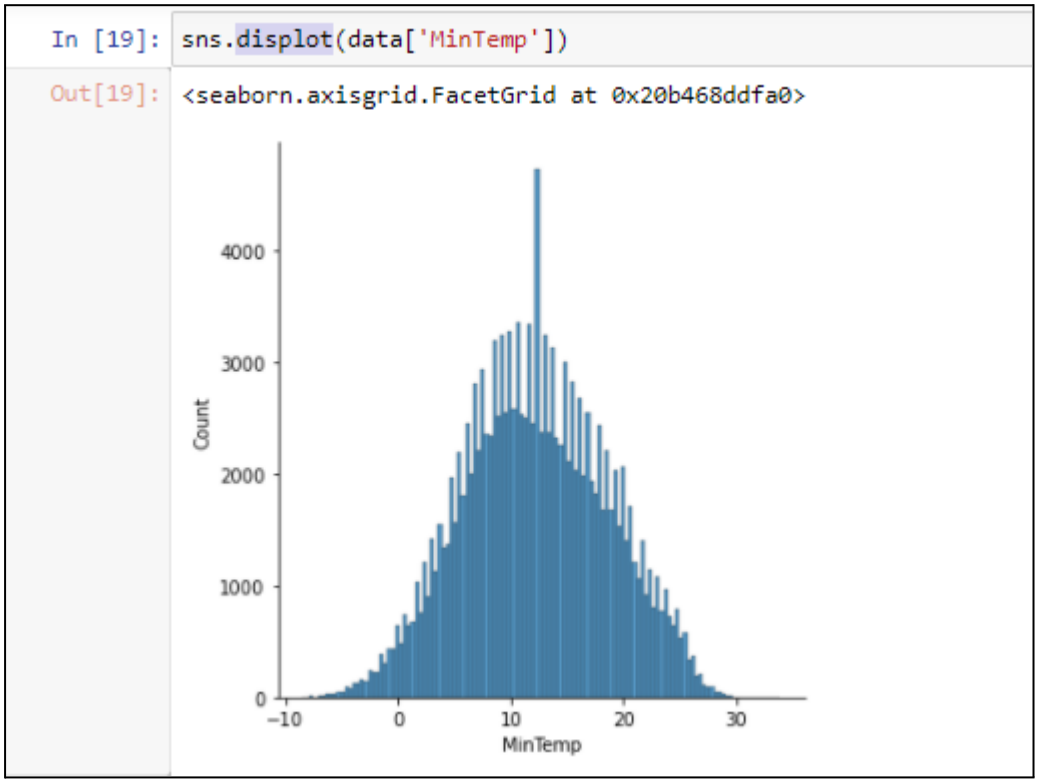
A scatter plot (also called a scatterplot, scatter graph, scatter chart, scattergram, or scatter diagram) is a type of plot or mathematical diagram using Cartesian coordinates to display values for typically two variables for a set of data.



From the above graph, we can infer that there is some positive relation between the two variables

7. A Distplot or distribution plot:-

depicts the variation in the data distribution. Seaborn Distplot represents the overall distribution of continuous data variables. The Seaborn module along with the Matplotlib module is used to depict the distplot with different variations in it.



From the above graph, we can infer that the column MinTemp follows the normal distribution

Splitting the Dateset into Dependent and Independent variable

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

```
#splitting x and y values
y = data['RainTomorrow']
x = data.drop('RainTomorrow',axis=1)
```

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.

standardizing the data

```
from sklearn.preprocessing import StandardScaler
```

```
#splitting x and y values
y = data['RainTomorrow']
x = data.drop('RainTomorrow',axis=1)
```

```
names = x.columns # loading the names of the x_features
```

```
names
```

```
Index(['Location', 'MinTemp', 'MaxTemp', 'Rainfall', 'WindGustSpeed',
       'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm',
       'Pressure9am', 'Pressure3pm', 'Temp9am', 'Temp3pm', 'RainToday',
       'WindGustDir', 'WindDir9am', 'WindDir3pm', 'year', 'month', 'day'],
      dtype='object')
```

```
sc = StandardScaler() # initializing the standardscaler
```

```
x = sc.fit_transform(x) # fitting an transforming the data into standard
```

```
x = pd.DataFrame(x,columns=names) #scaled data turns into array format,converting them into dataframe
```

- After scaling the data will be converted into an array form.
- Loading the feature names before scaling and converting them back to data frame after standard scaling is applied.

Splitting the data into Train and Test

· When you are working on a model and you want to train it, you obviously have a dataset. But after training, we have to test the model on some test datasets. For this, you will a dataset which is different from the training set you used earlier. But it might not always be possible to have so much data during the development phase. In such cases, the solution is to split the dataset into two sets, one for training and the other for testing.

- The train-test split is a technique for evaluating the performance of a machine learning algorithm.
- **Train Dataset**: Used to fit the machine learning model.
- **Test Dataset:** Used to evaluate the fit machine learning model.

- In general you can allocate 80% of the dataset to the training set and the remaining 20% to the test set.
- Now split our dataset into train set and test using train_test_split class from scikit learn library.

```
from sklearn import model_selection

x_train,x_test,y_train,y_test = model_selection.train_test_split(x,y,test_size =0.2,random_state =0)
```

Model Building

The model building includes the following main tasks

- Import the model building Libraries
- Initializing the model
- Training and testing the model
- Evaluation of Model
- Save the Model

Training and Testing the Model

Once after splitting the data into train and test, the data should be fed to an algorithm to build a model.

There are several Machine learning algorithms to be used depending on the data you are going to process such as images, sound, text, and numerical values. The algorithms that you can choose according to the objective that you might have it may be Classification algorithms are Regression algorithms.

1.Logistic Regression

2. Decision Tree Classifier

3. Random Forest Classifier

4.KNN

5.svm

5.xgboost

Steps in Building the model:-

Initialize the model.

Fit the models with x_train and y_train.

Predict the y_train values and calculate the accuracy.

Predict the y_test values and calculate the accuracy.

initializing all the models and predicting for better accuracy

```
[ ]: #Models intilization of the models
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()
```

```
[ ]: # fitting the model
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)
```

```
[ ]: # predicting the train  values
p1 = XGBoost.predict(x_train)
p2 = Rand_forest.predict(x_train)
p3 = svm.predict(x_train)
p4 = Dtree.predict(x_train)
p5 = GBM.predict(x_train)
p6 = log.predict(x_train)
```

```
[ ]: #checking the accuraccy score
print("xgboost:",metrics.accuracy_score(y_train,p1))
print("Rand_forest:",metrics.accuracy_score(y_train,p2))
print("svm:",metrics.accuracy_score(y_train,p3))
print("Dtree:",metrics.accuracy_score(y_train,p4))
print("GBM:",metrics.accuracy_score(y_train,p5))
print("log:",metrics.accuracy_score(y_train,p6))
```

We're going to use x_train and y_train obtained above in the train_test_split section to train our decision tree regression model. We're using the fit method and passing the parameters as shown below.

We are using the algorithm from Scikit learn library to build the model as shown below,

Once the model is trained, it's ready to make predictions. We can use the predict method on the model and pass x_test as a parameter to get the output as y_pred.

Notice that the prediction output is an array of real numbers corresponding to the input array.

Model Evaluation

After training the model, the model should be tested by using the test data which is been separated while splitting the data for checking the functionality of the model.

Regression Evaluation Metrics:

These model evaluation techniques are used to find out the accuracy of models built in classification type of machine learning models. We have three types of evaluation methods.

- Accuracy_score
- Confusion matrix
- Roc- Auc Curve

1. Accuracy_score

It is the ratio of the number of correct predictions to the total number of input samples.

$$Accuracy = \frac{Number\ of\ Correct\ predictions}{Total\ number\ of\ predictions\ made}$$

```
[40]: print("xgboost:",metrics.accuracy_score(y_test,t1))
      print("Rand_forest:",metrics.accuracy_score(y_test,t2))
      print("svm:",metrics.accuracy_score(y_test,t3))
      print("Dtree:",metrics.accuracy_score(y_test,t4))
      print("GBM:",metrics.accuracy_score(y_test,t5))
      print("log:",metrics.accuracy_score(y_test,t6))

      xgboost: 0.8420478919793242
      Rand_forest: 0.8569218326945391
      svm: 0.8525967861035901
      Dtree: 0.7837476704525476
      GBM: 0.8499947255529379
      log: 0.8418369140968388
```

Select the model, which gives the best accuracy of all, generate predictions and find the accuracy with training and testing data

2. Confusion Matrix

It is a matrix representation of the results of any binary testing.

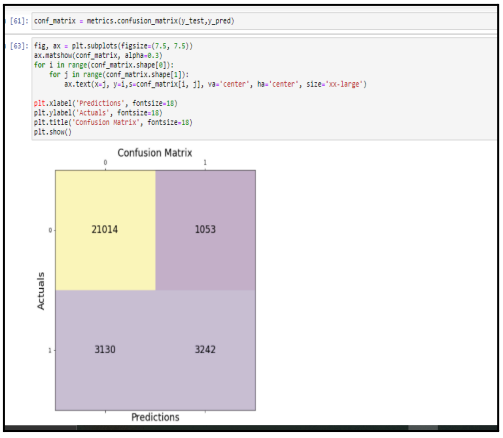


Fig: Confusion Matrix of prediction of rainfall

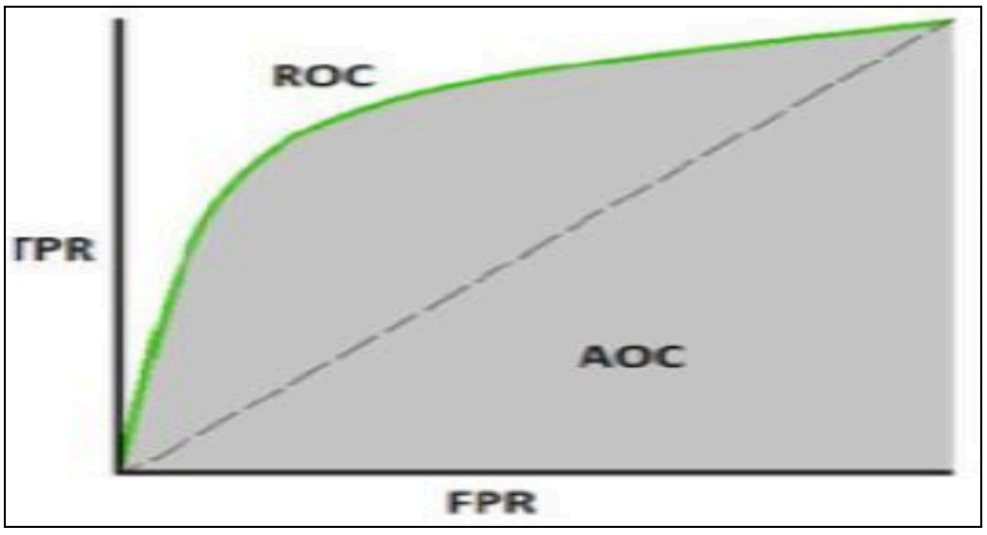
1. True Positive: 3242 (You have predicted the positive case correctly!)
2. True Negative: 21014 (You have predicted negative case correctly!)
3. False Positive: 1053 (You have predicted it will rain, but in actuality, it will not rain)
4. False Negative: 3130 (Wrong predictions)

3. Roc-Auc Curve

- AUC is the area under the ROC curve. AUC ROC indicates how well the probabilities from the positive classes are separated from the negative classes.
- AUC - ROC curve is a performance measurement for classification problems at various thresholds settings
- ROC is a probability curve and AUC represents the degree or measure of separability.
- Higher the AUC, better the model is at predicting 0s as 0s and 1s as 1sThe ROC curve is plotted with TPR against the FPR where TPR is on the y-axis and FPR is on the x-axis.

Code for Roc and performance

```
]: 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()
```



Save the Model

After building the model we have to save the model.

Pickle in Python is primarily used in serializing and de serializing 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

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

Build HTML Code

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

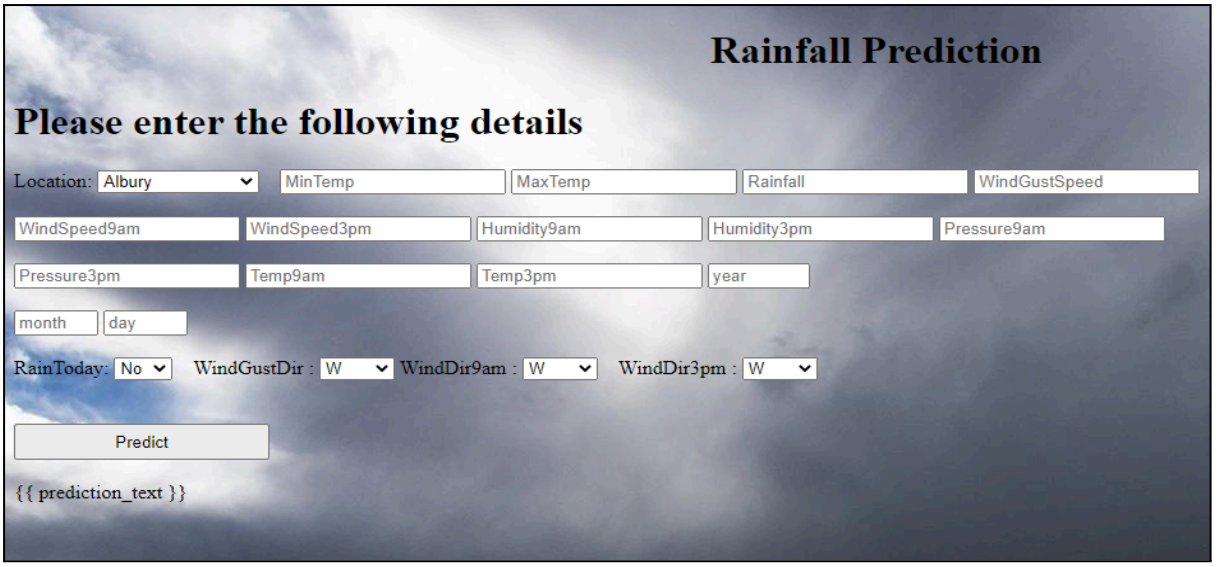
In our project we have 3 HTML files,they are

1.index.html

```
<!DOCTYPE html>
<html >
<head>
  <meta charset="UTF-8">
  <title>Rainfall prediction</title>
</head>


<body background="https://wallpaperaccess.com/full/701614.jpg" text="black">

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

     <!-- Main Input For Receiving Query to our ML -->
    <form action="{{ url_for('predict')}}"method="post">


</style></head>

      <label for="Location">Location:</label>
        <select id="Location" name="Location">
            <option value=2>Albury</option>
            <option value=4>BadgerysCreek</option>
            <option value=10>Cobar</option>
            <option value=11>CoffsHarbour</option>
            <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>SydneyAirport</option>
            <option value=42>WaggaWagga</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</option>
            <option value=6>Bendigo</option>
            <option value=35>Sale</option>
            <option value=19>MelbourneAirport</option>
            <option value=18>Melbourne</option>
            <option value=20>Mildura</option>
            <option value=25>Nhil</option>
            <option value=33>Portland</option>
            <option value=44>Watsonia</option>
```

```
        <select id= WindDir3pm  name= WindDir3pm >
            <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>NW<option>
            <option value=3>SE<option>
            <option value=8>ESE<option>
            <option value=5>E<option>
            <option value=4>SSW<option>
        </select>   


<br>

<br><br>


<button type="submit" class="btn btn-primary btn-block btn-large" style="height:30px;width:200px">Predict</button>

    </form>
<br>


   {{ prediction_text }}


  <br>
  <br>
  <img src="{{url_3}}" alt="Submit Form" height="180" width="233" onerror="this.style.display='none'"/>
  <img src="{{url_1}}" alt="Submit Form" height="180" width="233" onerror="this.style.display='none'"/>
  <img src="{{url_4}}" alt="Submit Form" height="180" width="233" onerror="this.style.display='none'"/>
  <br>
  <br>
  <img src="{{url_2}}" alt="Submit Form" height="150" width="711" onerror="this.style.display='none'"/>

 </div>


</body>
</html>
```

The html page looks like



2.chance.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">
  <center><h1>chances of rain today.</h1></center>
</body>
</html>
```

3. noChance.html

```
<!DOCTYPE html>
<html >
<head>
  <meta charset="UTF-8">
  <title>Rainfall prediction</title>
</head>


<body background="https://cdn.tourradar.com/s3/tour/1500x800/139566_d97baf8c.jpg" text="black">

 <div class="login">
  <center><h1>No chances of rain today,enjoy your outing.</h1></center>
</body>
</html>
```

Main Python Script

Let us build an 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 the Flask framework which is written in python.

Importing the necessary libraries for building a flask application and integrating model and HTML pages

- Initializing the flask app
- Calling the pkl models and saving into a variable
- Routing and rending to the HTML page
- Calling the inputs from the HTML page and saving into the variable
- Creating the data labels
- Forming the data frame with labels and the data
- Scaling the data
- Predicting the values,by passing the data into the model
- Rendering the results onto the HTML pages based on the output

If the output is class-0,it means a page that displays non-potential customer will be rendered, if the output is 1 , a page with the potential customers will be displayed and the output is 2 a page with highly potential customer will be rendered.

The value of __name__ is set to __main__ when the module run as the main program otherwise it is set to the name of the module

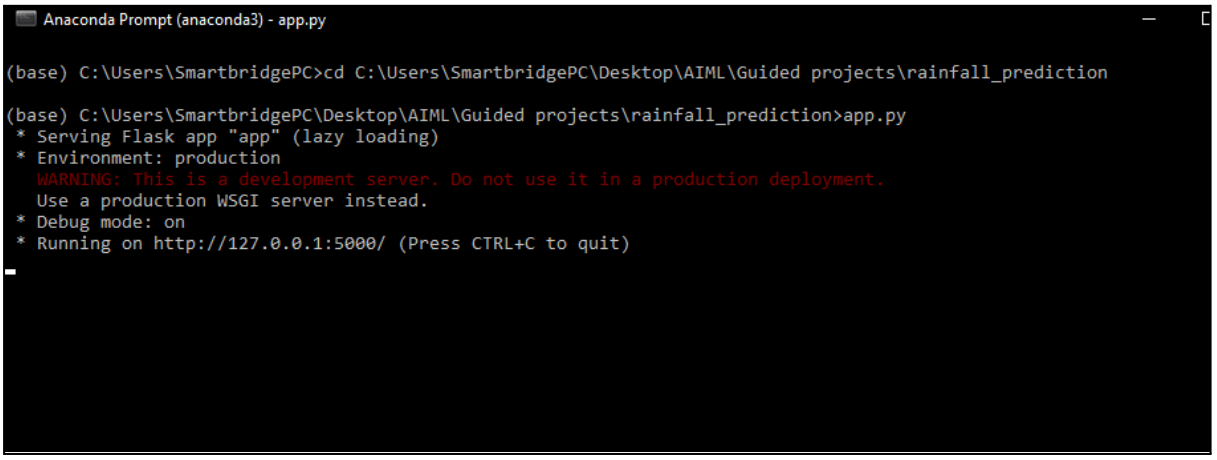
app.py

```
import numpy as np
import pickle
import joblib
import matplotlib
import matplotlib.pyplot as plt
import time
import pandas
import os
from flask import Flask, request, jsonify, render_template


app = Flask(__name__)
model = pickle.load(open('C:/Users/SmartbridgePC/Desktop/AIML/Guided projects/rainfall_prediction/Rainfall.pkl', 'rb'))
scale = pickle.load(open('C:/Users/SmartbridgePC/Desktop/AIML/Guided projects/rainfall_prediction/scale.pkl','rb'))

@app.route('/')# route to display the home page
def home():
    return render_template('index.html') #rendering the home page

@app.route('/predict',methods=["POST","GET"])# route to show the predictions in a web UI
def predict():
    #  reading the inputs given by the user
    input_feature=[x for x in request.form.values() ]
    features_values=[np.array(input_feature)]
    names = [['Location', 'MinTemp', 'MaxTemp', 'Rainfall', 'WindGustSpeed',
       'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm',
       'Pressure9am', 'Pressure3pm', 'Temp9am', 'Temp3pm', 'RainToday',
       'WindGustDir', 'WindDir9am', 'WindDir3pm', 'year', 'month', 'day']]
    data = pandas.DataFrame(features_values,columns=names)
    data = scale.fit_transform(data)
    data = pandas.DataFrame(data,columns = names)
     # predictions using the loaded model file
    prediction=model.predict(data)
    pred_prob = model.predict_proba(data)
    print(prediction)
    if prediction == "Yes":
        return render_template("chance.html")
    else:
        return render_template("nochance.html")
     # showing the prediction results in a UI
if __name__=="__main__":
```

Run the App

- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- 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



Output

- 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 button
- It will redirect to the page based on prediction output
- If the output predicted, the rain will not fall, this page will be displayed on the user interface

If the prediction is the occurrence of rain, this page will be get displayed on the user interface

COMMENTS

FIRST AND FOREMOST, I SINCERELY GRATITUDE TO OUR ESTEEMED INSTITUTE SRI VASAVI DEGREE COLLEGE, FOR GIVING ME THIS OPPORTUNITY TO FULFILL OUR WARM DREAM TO BECOME A GRADUATE. OUR SINCERE GRADITUDE TO OUR LONG-TERM INTERNSHIP GUIDE **SRI L LAKSHMI NARAYANA**, LECTURER DEPARTMENT OF COMPUTER SCIENCE FOR TIMELY COOPERATION AND VALUABLE SUGGESTIONS WHILE CARRYING OUT THIS INTERNSHIP.

I EXPRESS MY SINCERE THANKS AND HEARTFUL GRATITUDE TO **SRI L LAKSHMI NARAYANA**, HOD IN COMPUTER SCIENCE FOR PERMITTING ME TO DO MY PROJECT INTERNSHIP. I EXPRESS MY SINCERE THANKS AND HEARTFUL GRATITUDE TO **SRI M RAMA KRISHNA**, PRINCIPAL FOR PROVIDING A FAVOURABLE ENVIRONMENT AND SUPPORTING ME DURING THE DEVELOPMENT OF THIS INTERNSHIP.

THANK YOU,SMART BRIDGE

----MEDAPATI CHINNA REDDY
TEAM LEADER

THE END

SIGNATURE OF THE HOD

SIGNATURE OF THE PRINCIPAL