Exploratory Analysis of Rain Fall Data in India for Agriculture

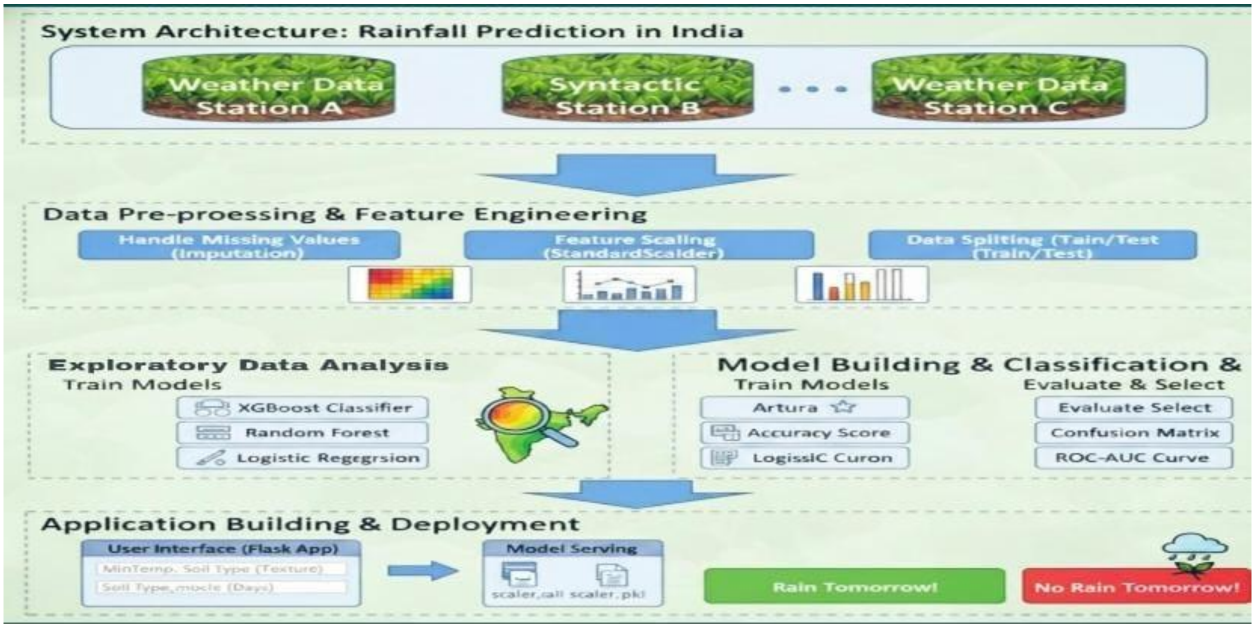
Project Description:

Rainfall plays a crucial role in agriculture, water resource management, and climate planning. Predicting rainfall accurately helps farmers, agricultural experts, and policymakers make informed decisions regarding crop planning, irrigation scheduling, and disaster management.

Rainfall prediction is a complex task due to the dynamic nature of weather conditions. By leveraging Machine Learning techniques, we can analyze historical weather data and predict whether it will rain tomorrow.

In this project, we use classification algorithms such as Decision Tree, Random Forest, Support Vector Machine (SVM), Gradient Boosting, and XGBoost. The best-performing model is selected based on accuracy and evaluation metrics and saved in .pkl format. The model is integrated into a Flask web application and deployed for real-time prediction.

Technical Architecture:


System Architecture: Rainfall Prediction in India
Weather Data Station A
Syntactic Station B
Weather Data Station C
Data Pre-proessing & Feature Engineering
Handle Missing Values (Imputation)
Feature Scaling (StandardScalder)
Data Spliting (Tain/Test (Train/Test)
Exploratory Data Analysis
Train Models
XGBoost Classifier
Random Forest
Logistic Regegrsion
Model Building & Classification &
Train Models
Artura
Accuracy Score
LogissiC Curon
Evaluate & Select
Evaluate Select
Confusion Matrix
ROC-AUC Curve
Application Building & Deployment
User Interface (Flask App)
MinTemp, Soil Type (Texture)
Soil Type, mocle (Days)
Model Serving
scaler,call scaler.pkl
Rain Tomorrow!
No Rain Tomorrow!

Pre requisites:

To complete this project, you must required following software’s, concepts and packages

- **Anaconda navigator and pycharm:**
  - Refer the link below to download anaconda navigator
  - Link : https://youtu.be/1ra4zH2G4o0
- **Python packages:**
  - Open anaconda prompt as administrator
  - Type “pip install numpy” and click enter.
  - Type “pip install pandas” and click enter.
  - Type “pip install scikit-learn” and click enter.
  - Type ”pip install matplotlib” and click enter.
  - Type ”pip install scipy” and click enter.
  - Type ”pip install pickle-mixin” and click enter.
  - Type ”pip install seaborn” and click enter.
  - Type “pip install Flask” and click enter.
  - Type “pip install xgboost” and click enter.

Prior Knowledge:

You must have prior knowledge of following topics to complete this project.

- **ML Concepts**
  - Supervised learning: https://www.javatpoint.com/supervised-machine-learning
  - Unsupervised learning: https://www.javatpoint.com/unsupervised-machine-learning
  - Regression and classification
  - Decision tree: https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm
  - Random forest: https://www.javatpoint.com/machine-learning-random-forest-algorithm
  - KNN: https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning
  - Xgboost: https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/
  - Evaluation metrics: https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/
- **Flask Basics** : https://www.youtube.com/watch?v=lj4I_CvBnt0

Project Objectives:

By the end of this project you will:

- Understand data preprocessing techniques
- Perform exploratory data analysis
- Apply classification algorithms
- Compare multiple ML models

- Deploy ML model using Flask
- Understand real-world ML deployment workflow

Project Flow:

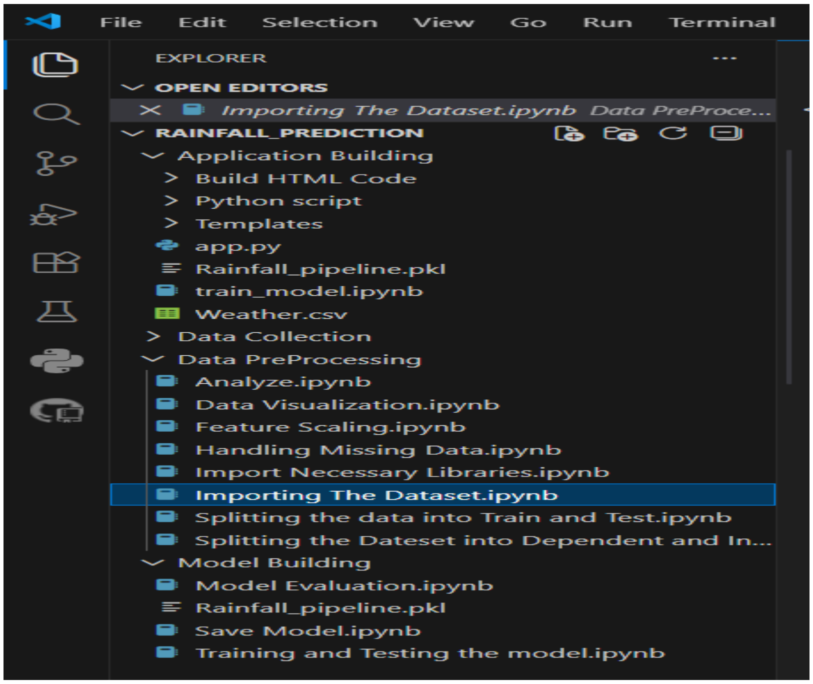
- User interacts with the UI and enters weather parameters.
- The input values are sent to the Flask backend.
- The backend preprocesses the input (encoding + scaling).
- The trained model predicts rainfall.
- Prediction result is displayed on UI.

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

- Data collection
  - Collect the dataset or create the dataset
- Visualizing and analyzing data
  - Univariate analysis
  - Bivariate analysis
  - Multivariate analysis
  - Descriptive analysis
- Data pre-processing
  - Checking for null values
  - Handling outlier
  - Handling categorical data
  - Splitting data into train and test
- Model building
  - Import the model building libraries
  - Initializing the model
  - Training and testing the model
  - Evaluating performance of model
  - Save the model
- Application Building
  - Create an HTML file
  - Build python code

Project Structure:

Create the Project folder which contains files as shown below



```
OPEN EDITORS
  × Importing The Dataset.ipynb Data PreProce...
RAINFALL_PREDICTION
  Application Building
    > Build HTML Code
    > Python script
    > Templates
    app.py
    Rainfall_pipeline.pkl
    train_model.ipynb
    Weather.csv
  > Data Collection
  Data PreProcessing
    Analyze.ipynb
    Data Visualization.ipynb
    Feature Scaling.ipynb
    Handling Missing Data.ipynb
    Import Necessary Libraries.ipynb
    Importing The Dataset.ipynb
    Splitting the data into Train and Test.ipynb
    Splitting the Dateset into Dependent and In...
  Model Building
    Model Evaluation.ipynb
    Rainfall_pipeline.pkl
    Save Model.ipynb
    Training and Testing the model.ipynb
```

- We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.
- rainfall.pkl is our saved model. Further we will use this model for flask integration.
- Training folder contains model training files and templates folder contains IBM deployment files.

Milestone 1: Data Collection

ML depends heavily on data, It is most crucial aspect that makes algorithm training possible. So this section allows you to download the required dataset.

Activity 1: Download the dataset

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

In this project we have used WeatherAUS.csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

Link:

https://docs.google.com/spreadsheets/d/1RA2OO0LZTeQykI_mvnensAjp6LM4YzWI1Tz0SUG5-Ao/edit?gid=121883362#gid=121883362

The dataset contains historical weather observations including:

- Location
- Temperature
- Rainfall
- Humidity
- Pressure
- Wind Speed
- Rain Tomorrow (Target Variable)

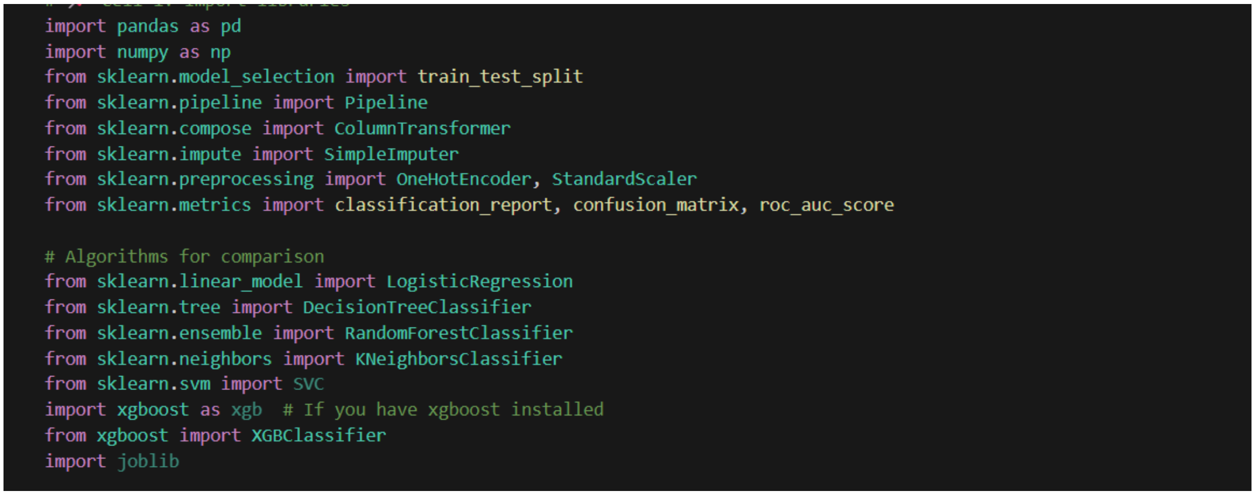
Milestone 2: Data Visualizing and analysis

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualization techniques and some analysing techniques.

Note: There is n number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

Activity 1: Importing the libraries

Import the necessary libraries as shown in the image. (optional) Here we have used visualization style as fivethirtyeight.

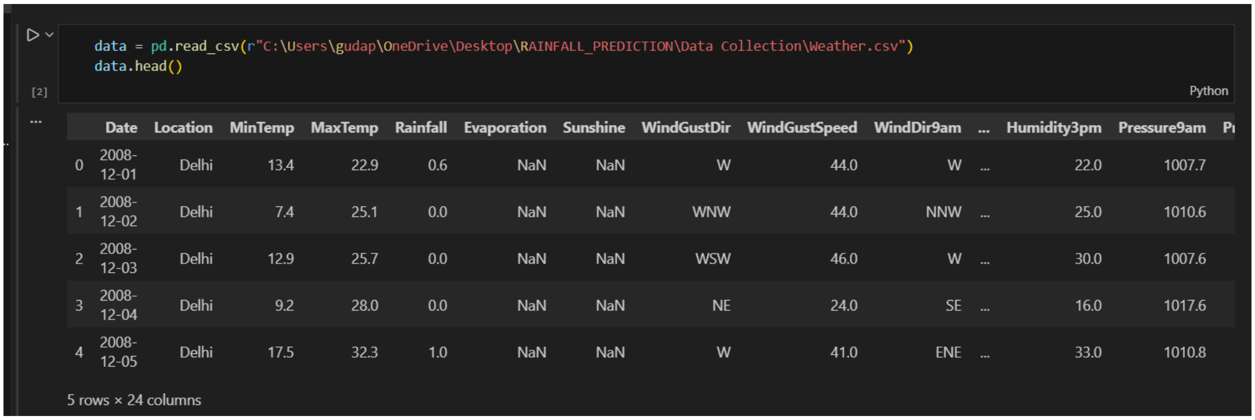


```
# %% Cell 1: Import libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score

# Algorithms for comparison
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
import xgboost as xgb  # If you have xgboost installed
from xgboost import XGBClassifier
import joblib
```

Activity 2: Read the Dataset

Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas.

In pandas we have a function called read_csv() to read the dataset. As a parameter we have to give the directory of csv file.



```
data = pd.read_csv(r"C:\Users\gudap\OneDrive\Desktop\RAINFALL_PREDICTION\Data Collection\Weather.csv")
data.head()
```

[2] Python

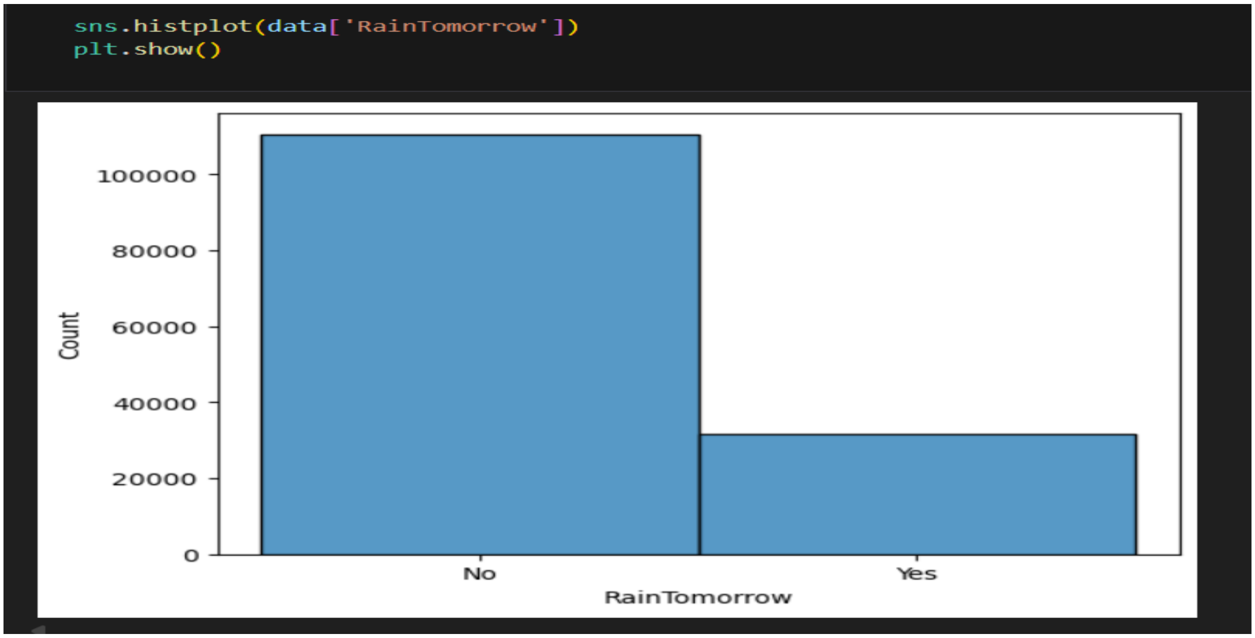
	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	...	Humidity3pm	Pressure9am	Pı
0	2008-12-01	Delhi	13.4	22.9	0.6	NaN	NaN	W	44.0	W	...	22.0	1007.7	
1	2008-12-02	Delhi	7.4	25.1	0.0	NaN	NaN	WNW	44.0	NNW	...	25.0	1010.6	
2	2008-12-03	Delhi	12.9	25.7	0.0	NaN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

5 rows × 24 columns

Activity 3: Univariate analysis

In simple words, univariate analysis is understanding the data with single feature. Here we have displayed two different graphs such as distribution plot.

- Seaborn package provides a wonderful function boxplot. With the help of count plot, we can find the distribution of the feature. To make multiple graphs in a single plot, we use subplot.

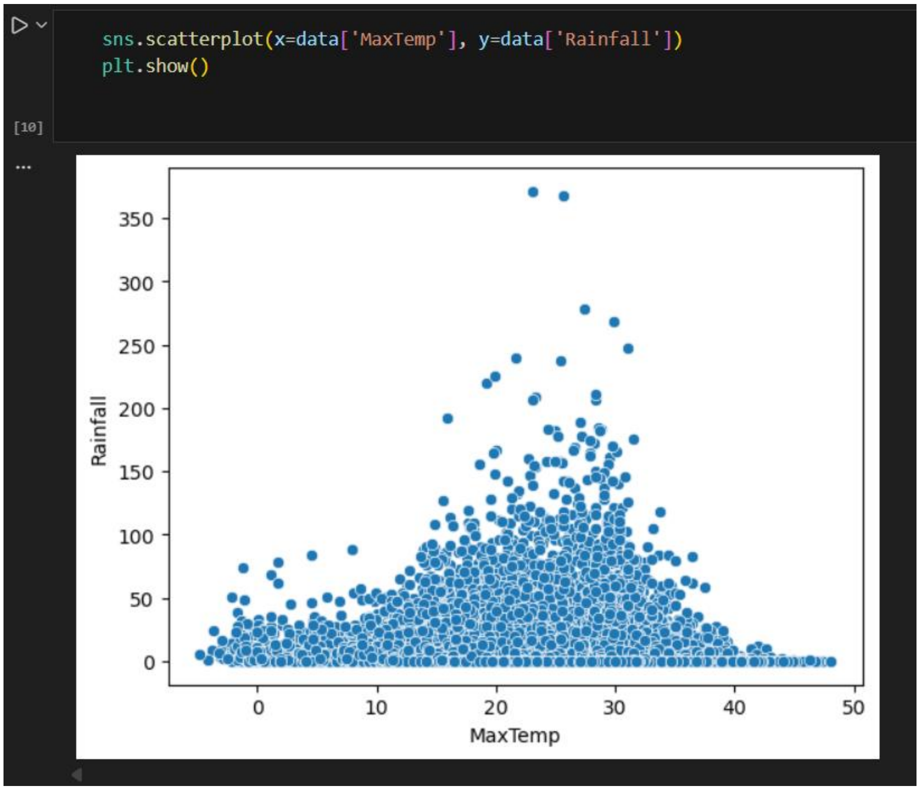


```
sns.histplot(data['RainTomorrow'])
plt.show()
```

- In our dataset we have some categorical features. With the count plot function, we are going to count the unique category in those features. We have created a dummy data frame with categorical features. With for loop and subplot we have plotted this below graph.
- From the plot we came to know, Applicants income is skewed towards left side, where as credit history is categorical with 1.0 and 0.0
-

Activity 4: Bivariate analysis

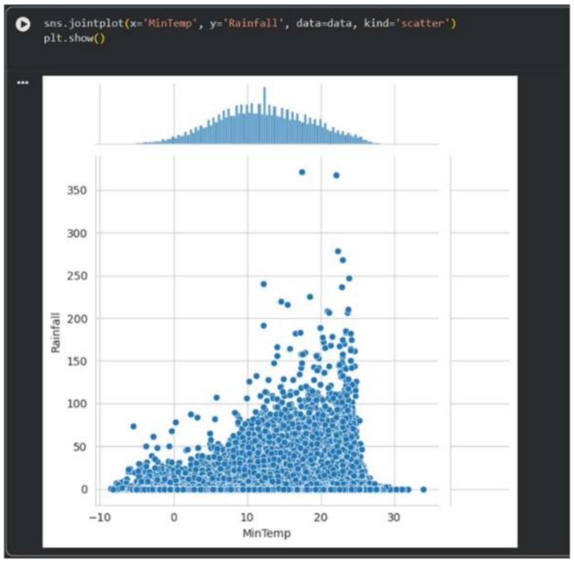
Scatter plot:-



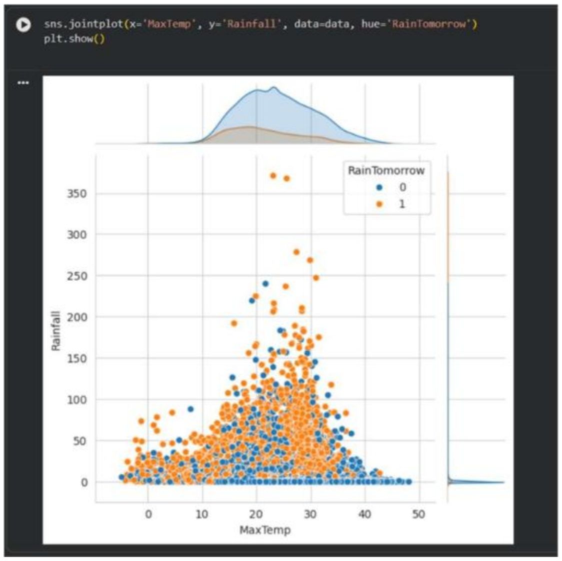
```
sns.scatterplot(x=data['MaxTemp'], y=data['Rainfall'])
plt.show()
```

Joint Plot:

A joint plot can be thought of as a stogram across a categorical, instead of quantitative, variable. The basic API and options are identical to those for barplot() , so you can compare counts across nested variables.



```
sns.jointplot(x='MinTemp', y='Rainfall', data=data, kind='scatter')
plt.show()
```



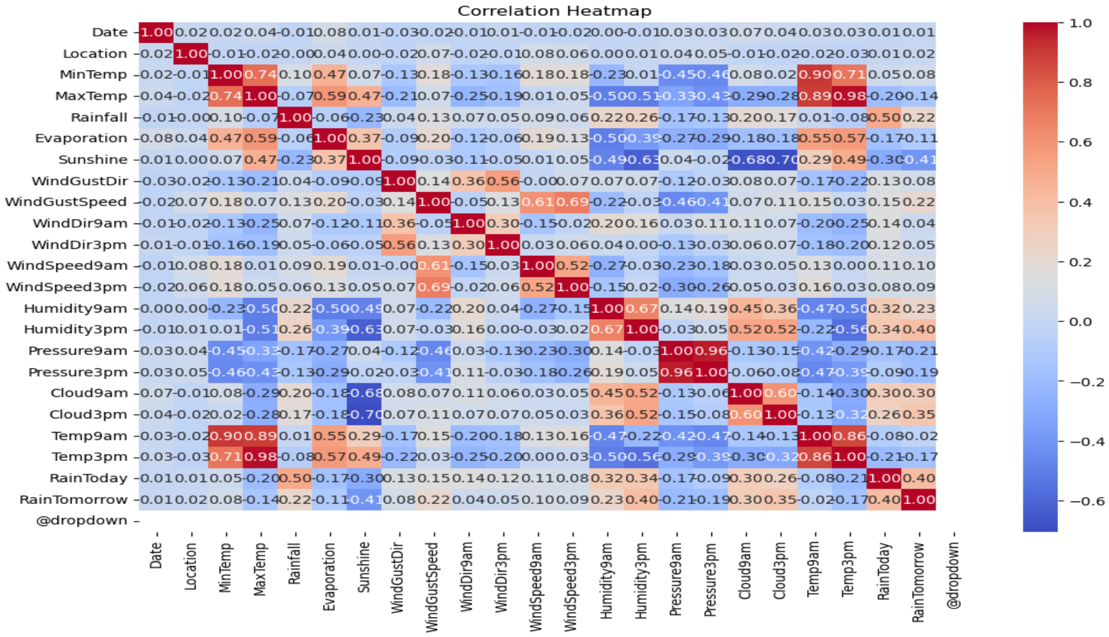
```
sns.jointplot(x='MaxTemp', y='Rainfall', data=data, hue='RainTomorrow')
plt.show()
```

From the above graph we can infer the analysis such as

- Segmenting the whether column and temparature column based on bar graphs
- Segmenting the Rainy day and No rain tommorow based on bar graphs ,for drawing insights such as Rainy or not.

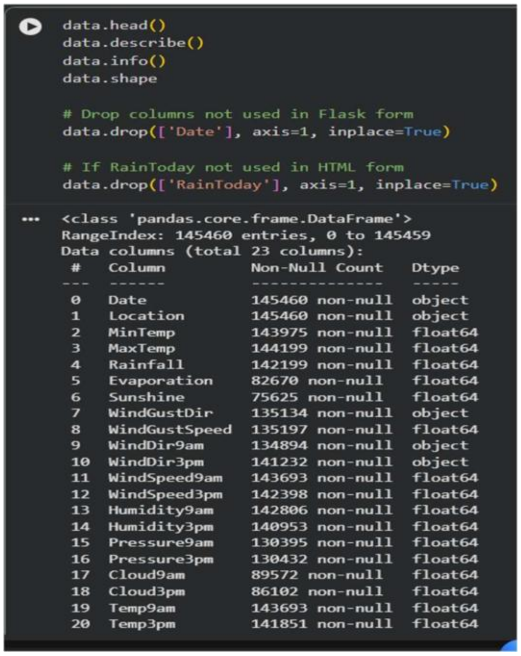
Activity 5: Multivariate analysis

In simple words, multivariate analysis is to find the relation between multiple features. Here we have used heatmap from seaborn package.



Activity 6: Descriptive analysis

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has a worthy function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features.



```
data.head()
data.describe()
data.info()
data.shape

# Drop columns not used in Flask form
data.drop(['Date'], axis=1, inplace=True)

# If RainToday not used in HTML form
data.drop(['RainToday'], axis=1, inplace=True)
```

```
<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
```

Milestone 3: Data Pre-processing

As we have understood how the data is lets pre-process the collected data.

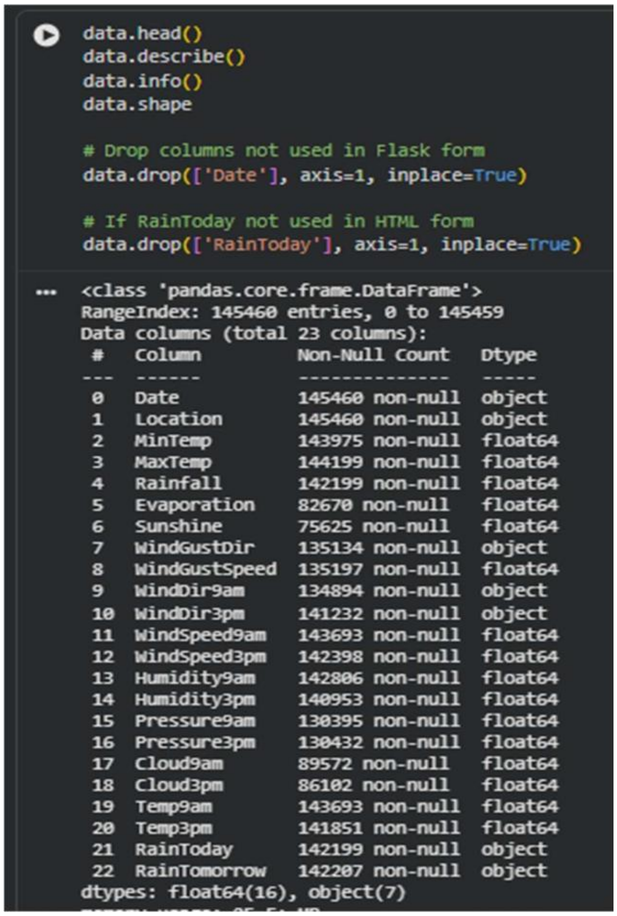
The download data set is not suitable for training the machine learning model as it might have so much of randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

- Handling missing values
- Encoding categorical variables
- Feature scaling
- Splitting dataset into training and test set

Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps.

Activity 1: Checking for null values

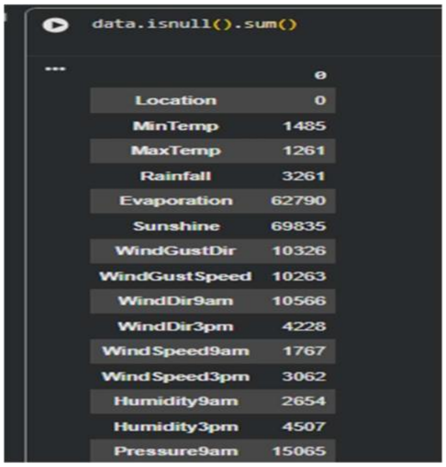
- Let's find the shape of our dataset first, To find the shape of our data, df.shape method is used. To find the data type, df.info() function is used.



```
data.head()
data.describe()
data.info()
data.shape

# Drop columns not used in Flask form
data.drop(['Date'], axis=1, inplace=True)

# If RainToday not used in HTML form
data.drop(['RainToday'], axis=1, inplace=True)
```

```
<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   142207 non-null  object
dtypes: float64(16), object(7)
```

- For checking the null values, df.isnull() function is used. To sum those null values we use .sum() function to it. From the below image we found that there are no null values present in our dataset. So we can skip handling of missing values step.



```
data.isnull().sum()
```

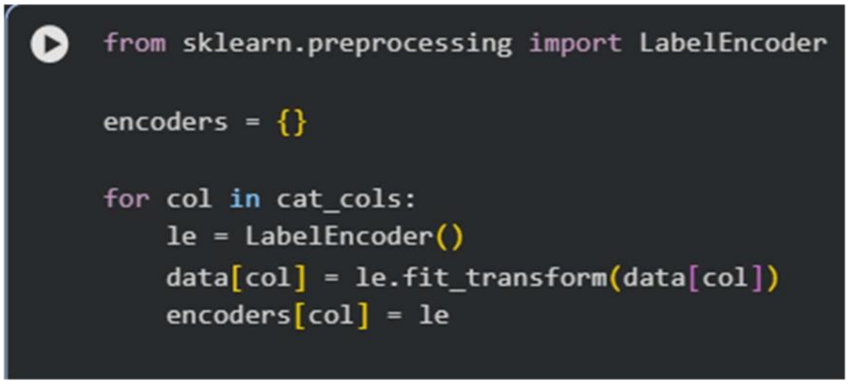
```
                    0
Location            0
MinTemp          1485
MaxTemp          1261
Rainfall         3261
Evaporation     62790
Sunshine        69835
WindGustDir     10326
WindGustSpeed   10263
WindDir9am      10566
WindDir3pm       4228
WindSpeed9am     1767
WindSpeed3pm     3062
Humidity9am      2654
Humidity3pm      4507
Pressure9am     15065
```

Activity 2: Handling Categorical Values

As we can see our dataset has categorical data we must convert the categorical data to integer encoding or binary encoding.

To convert the categorical features into numerical features we use encoding techniques. There are several techniques but in our project we are using manual encoding with the help of list comprehension.

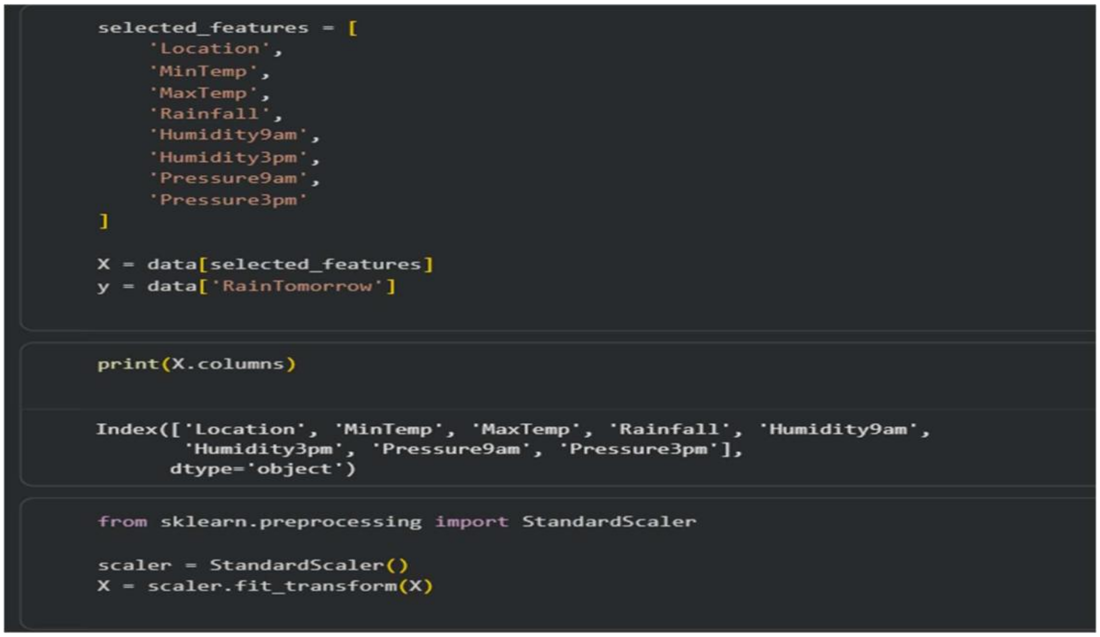
- In our project, so many data Location With list comprehension encoding is done.



```
from sklearn.preprocessing import LabelEncoder

encoders = {}

for col in cat_cols:
    le = LabelEncoder()
    data[col] = le.fit_transform(data[col])
    encoders[col] = le
```

Activity 3: Balancing the dataset:

Data Balancing is one of the most important step, which need to be performed for classification models, because when we train our model on imbalanced dataset , we will get biased results, which means our model is able to predict only one class element



```
selected_features = [
    'Location',
    'MinTemp',
    'MaxTemp',
    'Rainfall',
    'Humidity9am',
    'Humidity3pm',
    'Pressure9am',
    'Pressure3pm'
]

X = data[selected_features]
y = data['RainTomorrow']
```

```
print(X.columns)
```

```
Index(['Location', 'MinTemp', 'MaxTemp', 'Rainfall', 'Humidity9am',
       'Humidity3pm', 'Pressure9am', 'Pressure3pm'],
      dtype='object')
```

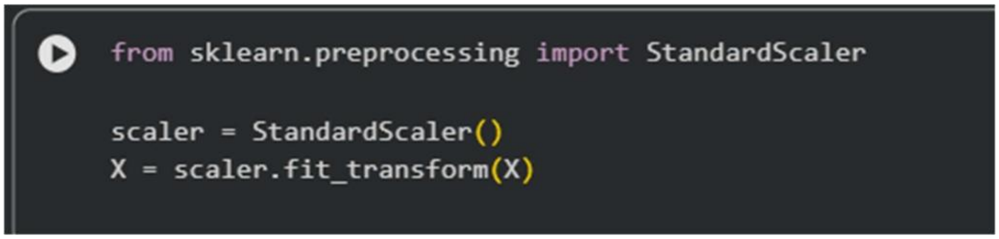
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X = scaler.fit_transform(X)
```

From the above picture, we can infer that ,previously our dataset is having many classes now we get some class items, after applying smote technique on the dataset the size has been changed for temperature class.

Activity 4: Scaling the Data

Scaling is one the important process, we have to perform on the dataset, because of data measures in different ranges can leads to mislead in prediction

Models such as KNN, Logistic regression need scaled data, as they follow distance based method and Gradient Descent concept.



```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X = scaler.fit_transform(X)
```

We will perform scaling only on the input values

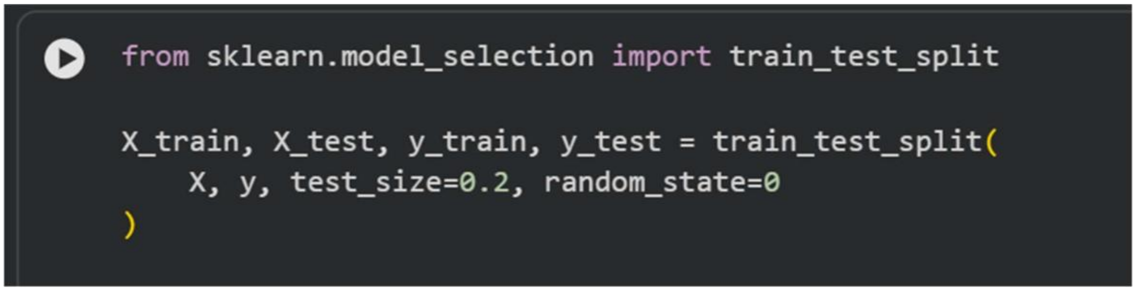
Once the dataset is scaled, it will be converted into array and we need to convert it back to dataframe.

Activity : Splitting data into train and test

Now let’s split the Dataset into train and test sets

Changes: first split the dataset into x and y and then split the data set

Here x and y variables are created. On x variable, df is passed with dropping the target variable. And on y target variable is passed. For splitting training and testing data we are using train_test_split() function from sklearn. As parameters, we are passing x, y, test_size, random_state.



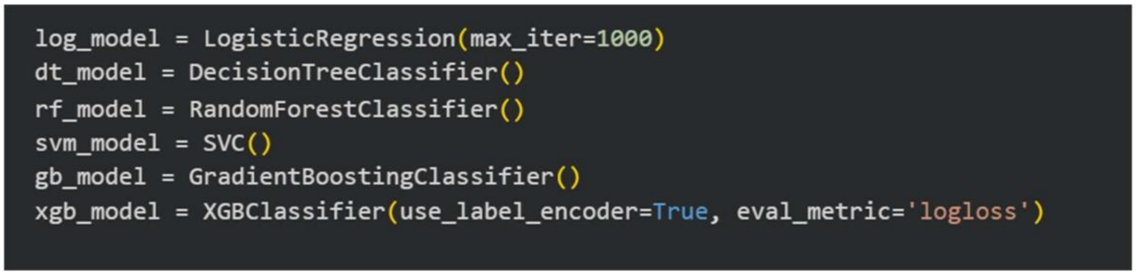
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=0
)
```

Milestone 4: Model Building

Now our data is cleaned and it’s time to build the model. We can train our data on different algorithms. for this project we are applying four classification algorithms. The best model is saved based on its performance.

Activity 1: Decision tree model

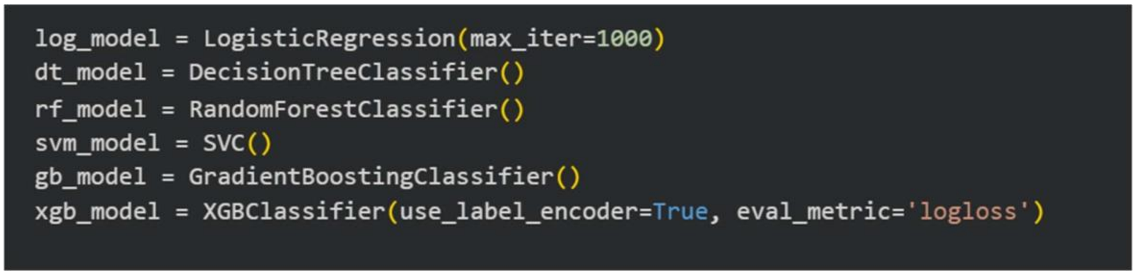
A function named decisionTree is created and train and test data are passed as the parameters. Inside the function, DecisionTreeClassifier algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classification report is done.



```
log_model = LogisticRegression(max_iter=1000)
dt_model = DecisionTreeClassifier()
rf_model = RandomForestClassifier()
svm_model = SVC()
gb_model = GradientBoostingClassifier()
xgb_model = XGBClassifier(use_label_encoder=True, eval_metric='logloss')
```

Activity 2: Random forest model

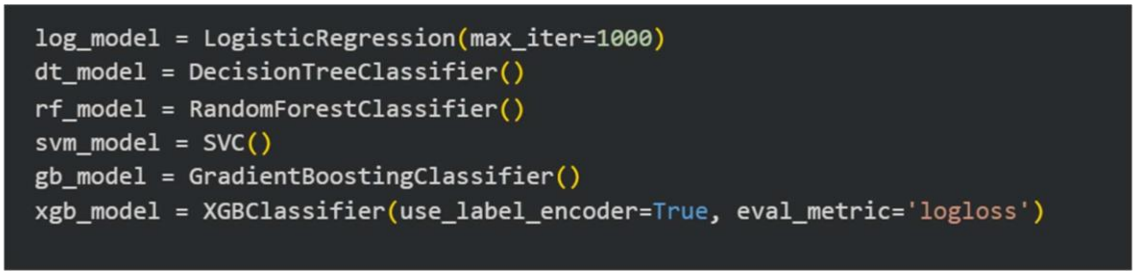
A function named randomForest is created and train and test data are passed as the parameters. Inside the function, RandomForestClassifier algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classification report is done.



```
log_model = LogisticRegression(max_iter=1000)
dt_model = DecisionTreeClassifier()
rf_model = RandomForestClassifier()
svm_model = SVC()
gb_model = GradientBoostingClassifier()
xgb_model = XGBClassifier(use_label_encoder=True, eval_metric='logloss')
```

Activity 3: Xgboost model

A function named xgboost is created and train and test data are passed as the parameters. Inside the function, GradientBoostingClassifier algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classification report is done.

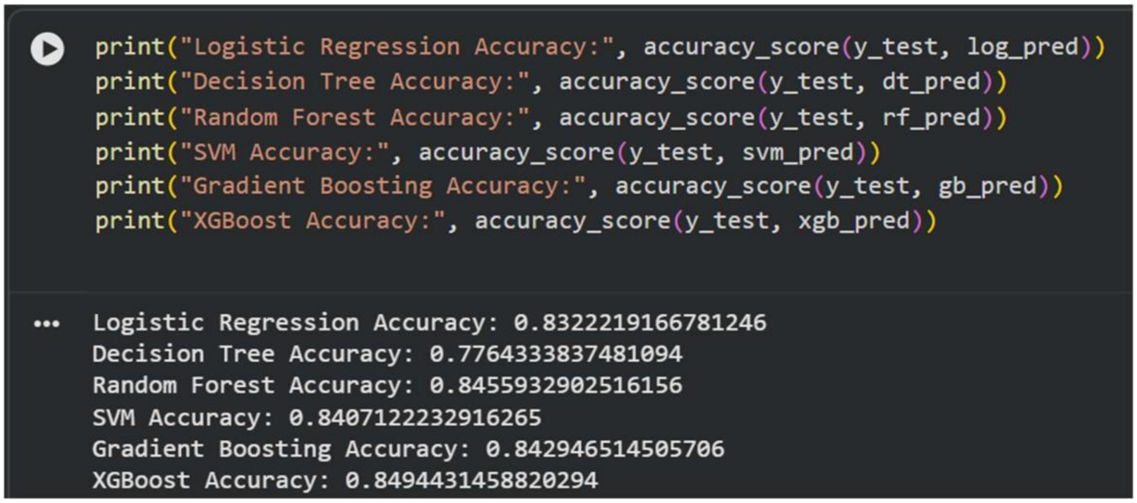


```
log_model = LogisticRegression(max_iter=1000)
dt_model = DecisionTreeClassifier()
rf_model = RandomForestClassifier()
svm_model = SVC()
gb_model = GradientBoostingClassifier()
xgb_model = XGBClassifier(use_label_encoder=True, eval_metric='logloss')
```

Now let’s see the performance of all the models and save the best model

Activity 4: Compare the model

For comparing the above models compare Model function is defined.



```
print("Logistic Regression Accuracy:", accuracy_score(y_test, log_pred))
print("Decision Tree Accuracy:", accuracy_score(y_test, dt_pred))
print("Random Forest Accuracy:", accuracy_score(y_test, rf_pred))
print("SVM Accuracy:", accuracy_score(y_test, svm_pred))
print("Gradient Boosting Accuracy:", accuracy_score(y_test, gb_pred))
print("XGBoost Accuracy:", accuracy_score(y_test, xgb_pred))
```

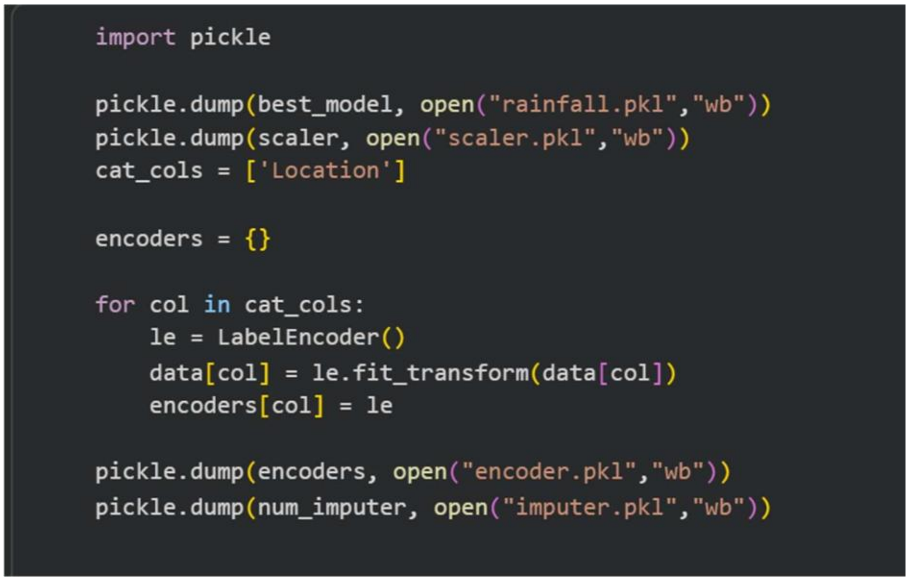
```
Logistic Regression Accuracy: 0.8322219166781246
Decision Tree Accuracy: 0.7764333837481094
Random Forest Accuracy: 0.8455932902516156
SVM Accuracy: 0.8407122232916265
Gradient Boosting Accuracy: 0.842946514505706
XGBoost Accuracy: 0.8494431458820294
```

After calling the function, the results of models are displayed as output. From the models Xgboost is performing well. From the below image, We can see the accuracy of the model. Xgboost is giving the accuracy of 84.9% with training data , 81.1% accuracy for the testing data.so we considering xgboost and deploying this model.

Activity 5: Evaluating performance of the model and saving the model

From sklearn, cross_val_score is used to evaluate the score of the model. On the parameters, we have given rf (model name), x, y, cv (as 5 folds). Our model is performing well. So, we are saving the model by pickle.dump().

Note: To understand cross validation, refer this link. https://towardsdatascience.com/cross-validation-explained-evaluating-estimator-performance-e51e5430ff85.



```
import pickle

pickle.dump(best_model, open("rainfall.pkl","wb"))
pickle.dump(scaler, open("scaler.pkl","wb"))
cat_cols = ['Location']

encoders = {}

for col in cat_cols:
    le = LabelEncoder()
    data[col] = le.fit_transform(data[col])
    encoders[col] = le

pickle.dump(encoders, open("encoder.pkl","wb"))
pickle.dump(num_imputer, open("imputer.pkl","wb"))
```

Milestone 5: 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 app.py

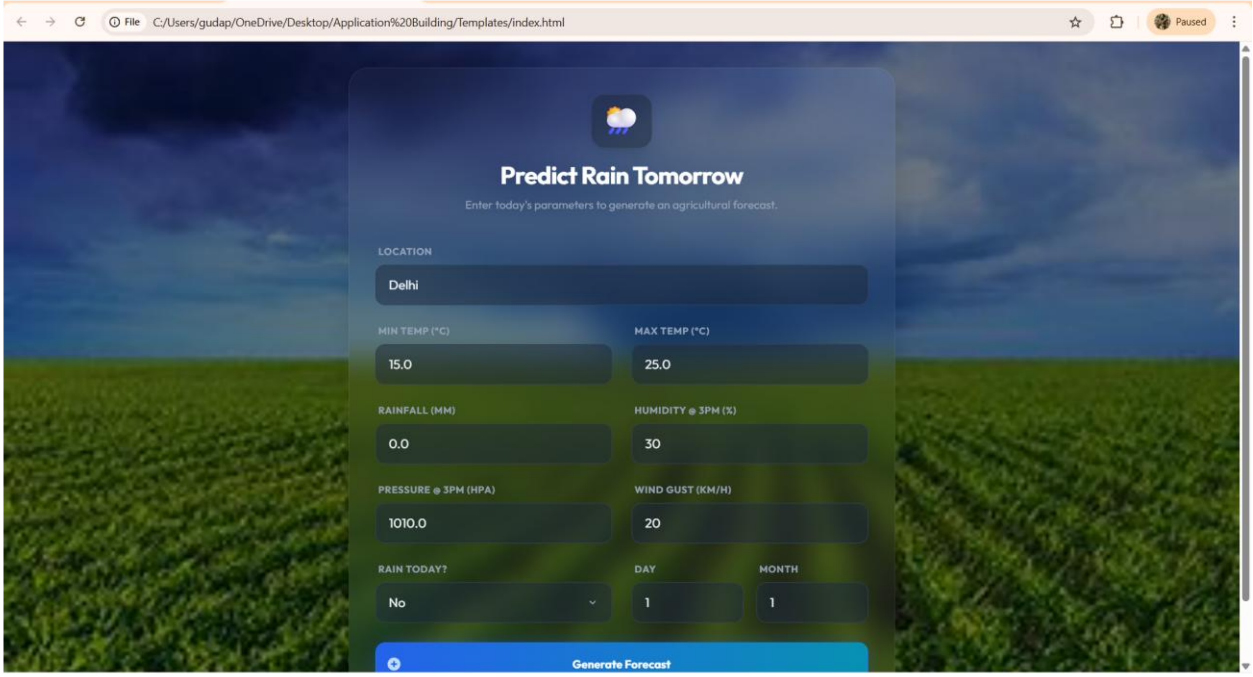
Activity1: Building Html Pages:

For this project create three HTML files namely

- Index.html
- Chance.html
- nochance.html

and save them in templates folder

Let’s see how our Index.html page looks like this after input temperature humidity values and rain values then press the Predict Button to check the Result :



File
C:/Users/gudap/OneDrive/Desktop/Application%20Building/Templates/index.html
Paused

Predict Rain Tomorrow

Enter today's parameters to generate an agricultural forecast.

LOCATION: Delhi

MIN TEMP (°C): 15.0

MAX TEMP (°C): 25.0

RAINFALL (MM): 0.0

HUMIDITY @ 3PM (%): 30

PRESSURE @ 3PM (HPA): 1010.0

WIND GUST (KM/H): 20

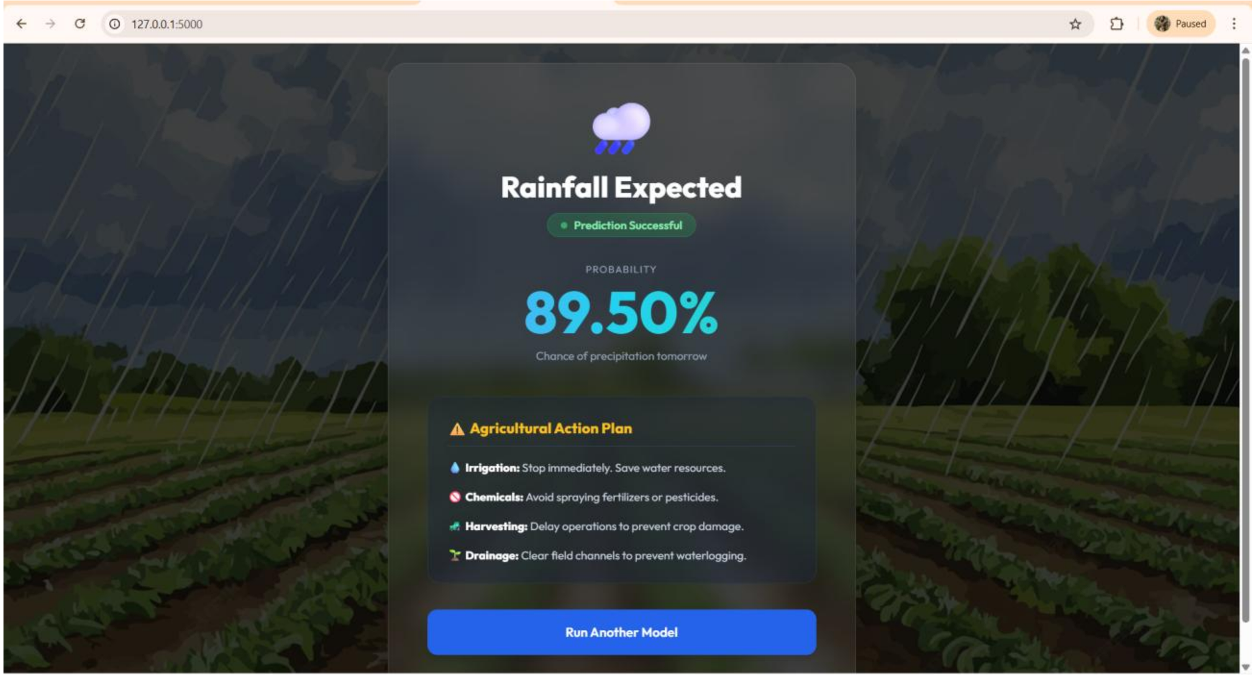
RAIN TODAY?: No

DAY: 1

MONTH: 1

Now when you click on predict button bottom you will get redirected to html file result,

Lets look how our chance.html file looks like:



Rainfall Expected

● Prediction Successful

PROBABILITY

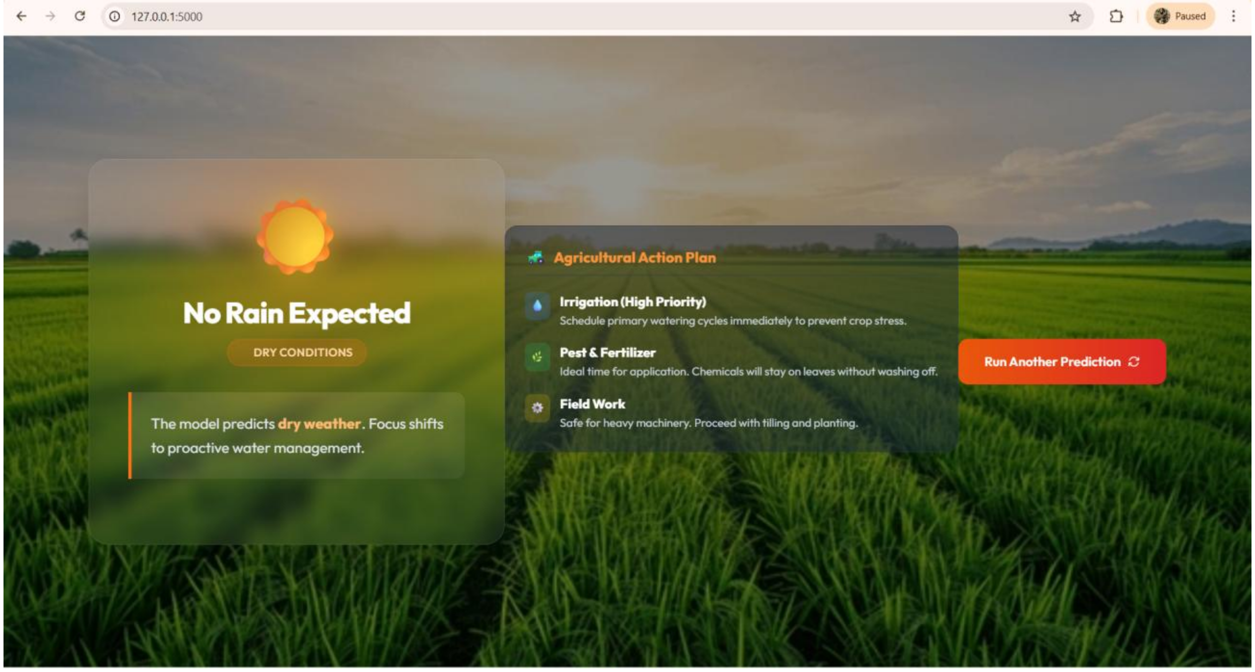
89.50%

Chance of precipitation tomorrow

⚠ Agricultural Action Plan

- 💧 **Irrigation:** Stop immediately. Save water resources.
- 🚫 **Chemicals:** Avoid spraying fertilizers or pesticides.
- 🚜 **Harvesting:** Delay operations to prevent crop damage.
- 🌱 **Drainage:** Clear field channels to prevent waterlogging.

Lets look how our no chance.html file looks like:



No Rain Expected

DRY CONDITIONS

The model predicts dry weather. Focus shifts to proactive water management.

Agricultural Action Plan

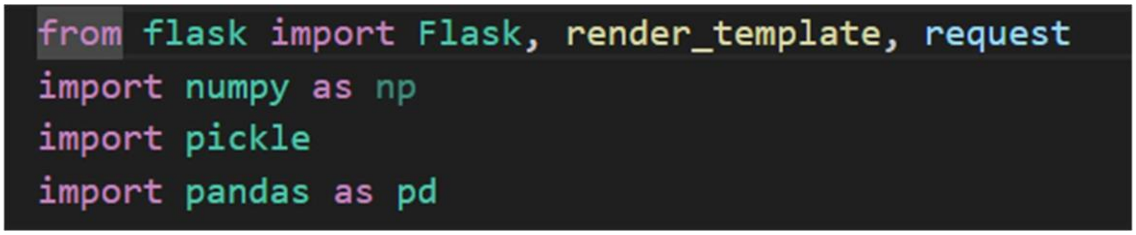
Irrigation (High Priority)
Schedule primary watering cycles immediately to prevent crop stress.

Pest & Fertilizer
Ideal time for application. Chemicals will stay on leaves without washing off.

Field Work
Safe for heavy machinery. Proceed with tilling and planting.

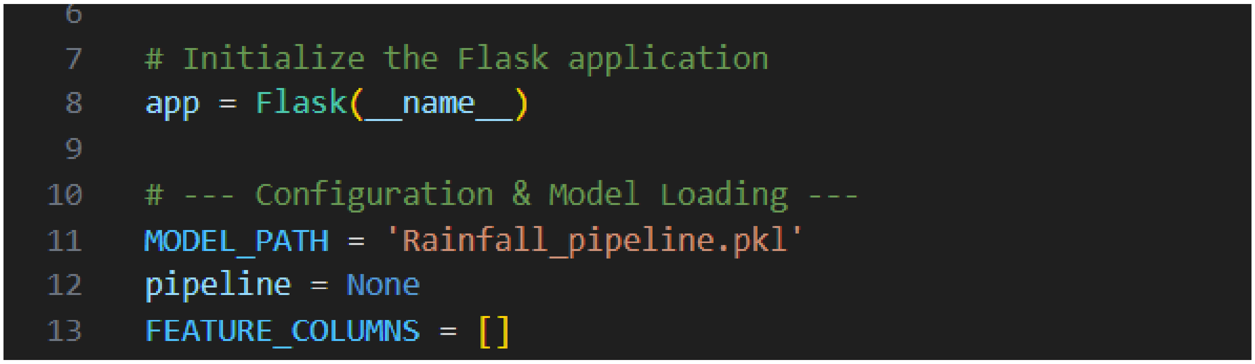
Activity 2: Build Python code:

Import the libraries



```python
from flask import Flask, render_template, request
import numpy as np
import pickle
import pandas as pd
```

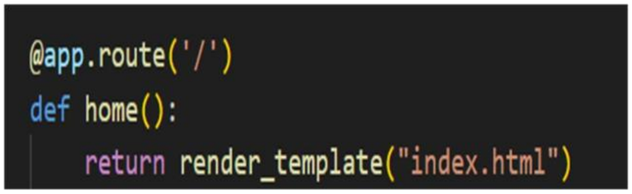
Load the saved model. Importing flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (__name__) as argument.



```python

# Initialize the Flask application
app = Flask(__name__)

# --- Configuration & Model Loading ---
MODEL_PATH = 'Rainfall_pipeline.pkl'
pipeline = None
FEATURE_COLUMNS = []
```

Render HTML page:

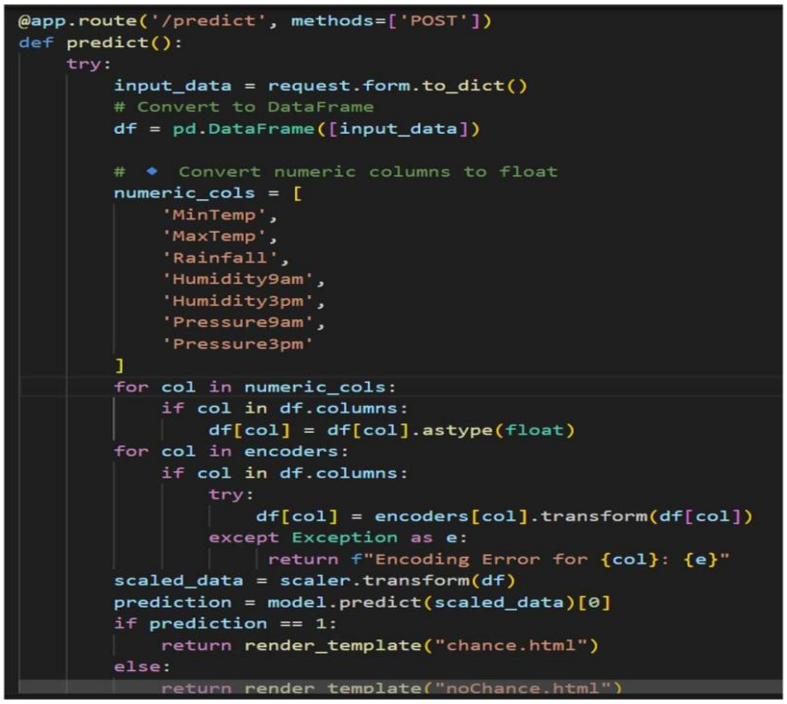
Here we will be using declared constructor to route to the HTML page which we have created earlier.



```
@app.route('/')
def home():
    return render_template("index.html")
```

In the above example, '/' URL is bound with home.html function. Hence, when the home page of the web server is opened in browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

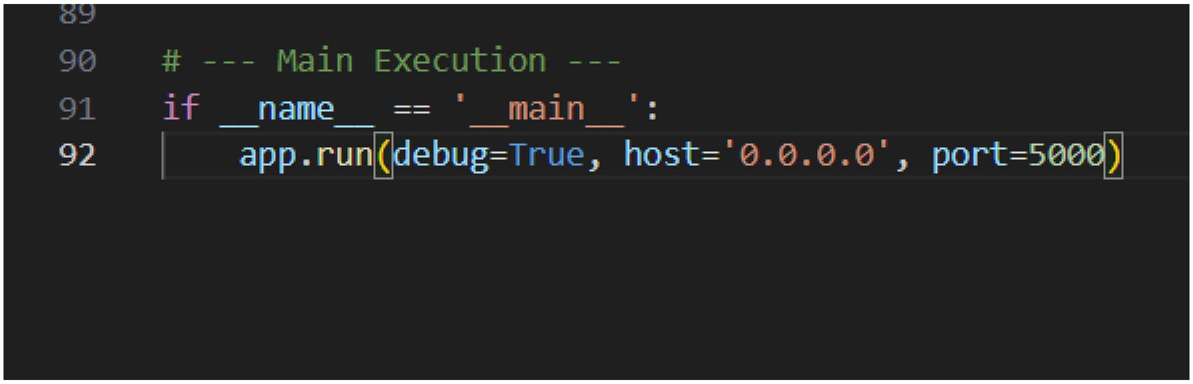
Retrieves the value from UI:



```
@app.route('/predict', methods=['POST'])
def predict():
    try:
        input_data = request.form.to_dict()
        # Convert to DataFrame
        df = pd.DataFrame([input_data])

        #  ◆  Convert numeric columns to float
        numeric_cols = [
            'MinTemp',
            'MaxTemp',
            'Rainfall',
            'Humidity9am',
            'Humidity3pm',
            'Pressure9am',
            'Pressure3pm'
        ]
        for col in numeric_cols:
            if col in df.columns:
                df[col] = df[col].astype(float)
        for col in encoders:
            if col in df.columns:
                try:
                    df[col] = encoders[col].transform(df[col])
                except Exception as e:
                    return f"Encoding Error for {col}: {e}"
        scaled_data = scaler.transform(df)
        prediction = model.predict(scaled_data)[0]
        if prediction == 1:
            return render_template("chance.html")
        else:
            return render_template("noChance.html")
```

Here we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will rendered to the text that we have mentioned in the submit.html page earlier.

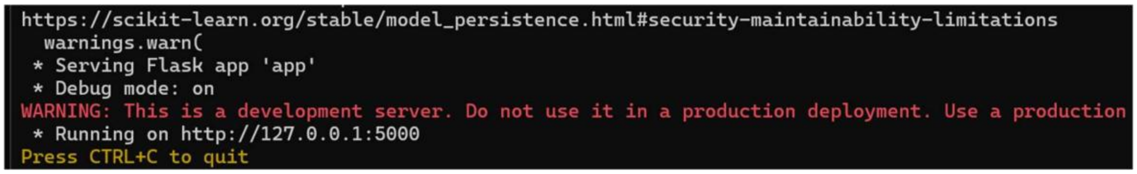
Main Function:



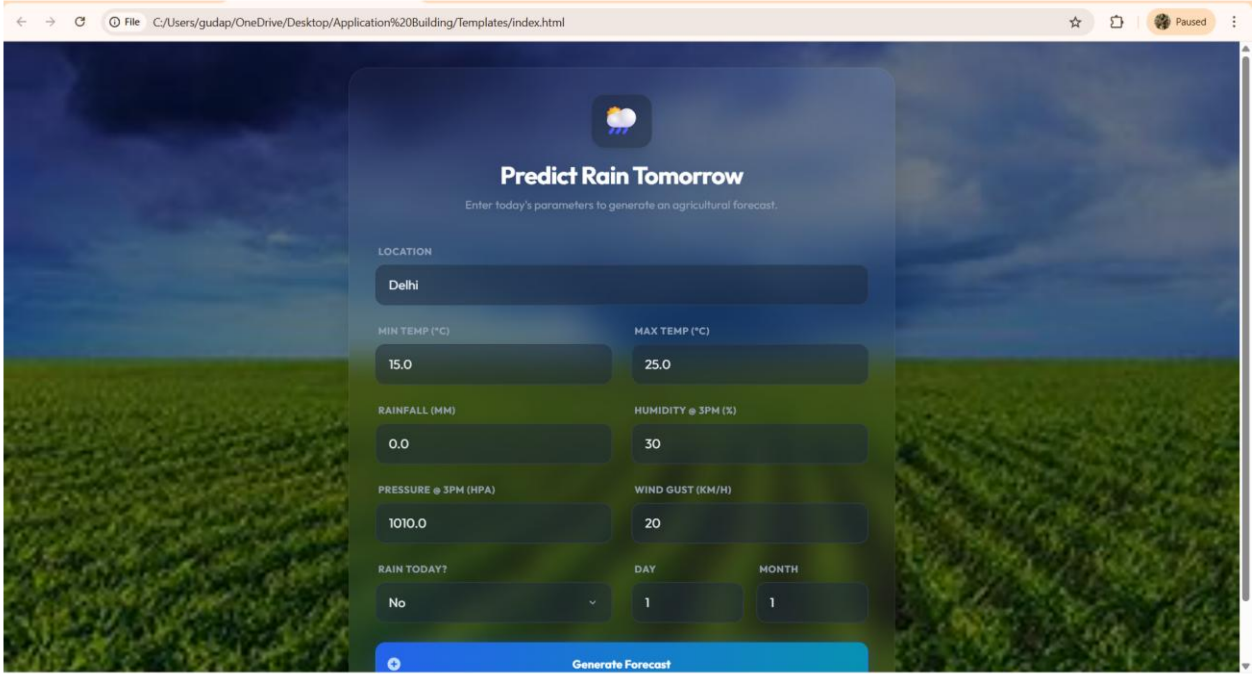
```
# --- Main Execution ---
if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0', port=5000)
```

Activity 3: Run the application

- 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.
- Click on the predict button from bottom center, enter the inputs, click on the predict button, and see the result/prediction on the web.



```
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```



File
C:/Users/gudap/OneDrive/Desktop/Application%20Building/Templates/index.html
Paused

Predict Rain Tomorrow

Enter today's parameters to generate an agricultural forecast.

LOCATION: Delhi

MIN TEMP (°C): 15.0

MAX TEMP (°C): 25.0

RAINFALL (MM): 0.0

HUMIDITY @ 3PM (%): 30

PRESSURE @ 3PM (HPA): 1010.0

WIND GUST (KM/H): 20

RAIN TODAY?: No

DAY: 1

MONTH: 1