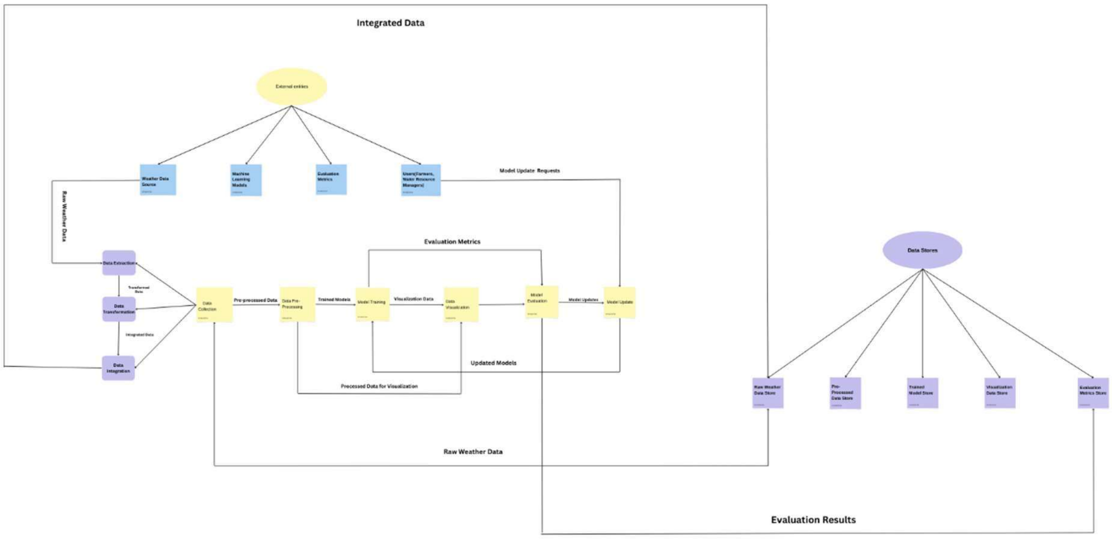
# PROJECT MANUAL

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

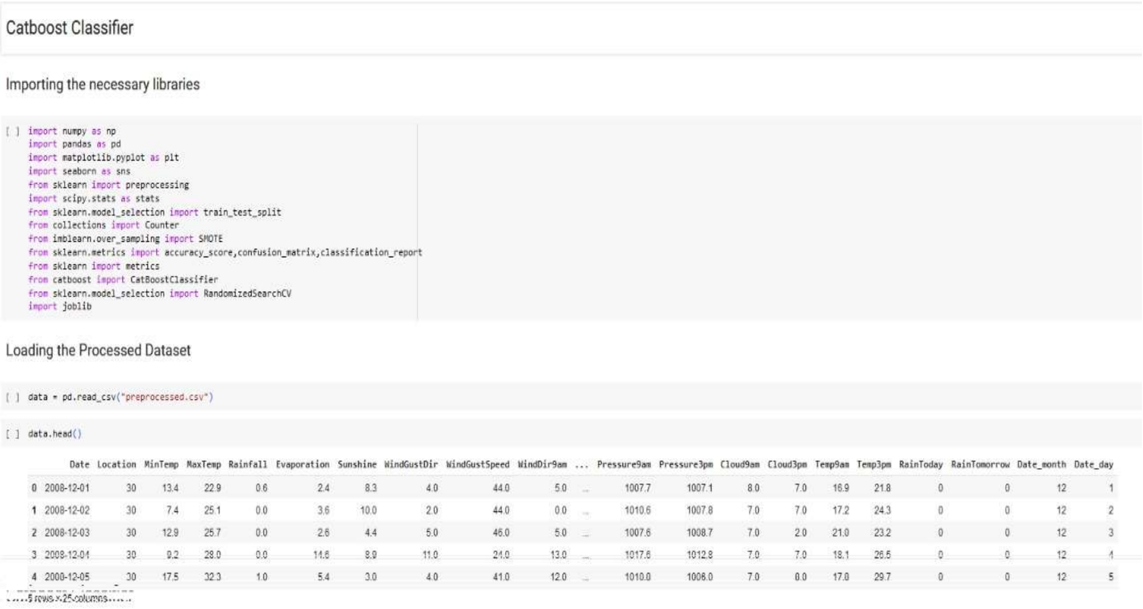
## Project Description:

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:



## CATBOOST CLASIFIER:



Catboost Classifier

Importing the necessary libraries

```
[ ] import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as sns
    from sklearn import preprocessing
    import scipy.stats as stats
    from sklearn.model_selection import train_test_split
    from collections import Counter
    from imblearn.over_sampling import SMOTE
    from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
    from sklearn import metrics
    from catboost import CatBoostClassifier
    from sklearn.model_selection import RandomizedSearchCV
    import joblib
```

Loading the Processed Dataset

```
[ ] data = pd.read_csv("preprocessed.csv")
```

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

```
         Date  Location  MinTemp  MaxTemp  Rainfall  Evaporation  Sunshine  WindGustDir  WindGustSpeed  WindDir9am  ...  Pressure9am  Pressure3pm  Cloud9am  Cloud3pm  Temp9am  Temp3pm  RainToday  RainTomorrow  Date_month  Date_day
0  2008-12-01        30     13.4     22.9       0.6          2.4       8.3          4.0           44.0         5.0  ...       1007.7       1007.1       8.0       7.0     16.9     21.8          0             0          12         1
1  2008-12-02        30      7.4     25.1       0.0          3.6      10.0          2.0           44.0         0.0  ...       1010.6       1007.8       7.0       7.0     17.2     24.3          0             0          12         2
2  2008-12-03        30     12.9     25.7       0.0          2.6       4.4          5.0           46.0         5.0  ...       1007.6       1008.7       7.0       2.0     21.0     23.2          0             0          12         3
3  2008-12-04        30      9.2     28.0       0.0         14.6       8.9         11.0           24.0        13.0  ...       1017.6       1012.8       7.0       7.0     18.1     26.5          0             0          12         4
4  2008-12-05        30     17.5     32.3       1.0          5.4       3.0          4.0           41.0        12.0  ...       1010.0       1006.0       7.0       8.0     17.0     29.7          0             0          12         5
5 rows × 25 columns
```

ti

## Classification in Machine Learning:

ti

Classification is a supervised learning task where the goal is to assign predefined labels to input data. Key concepts include:

### Features and Labels:

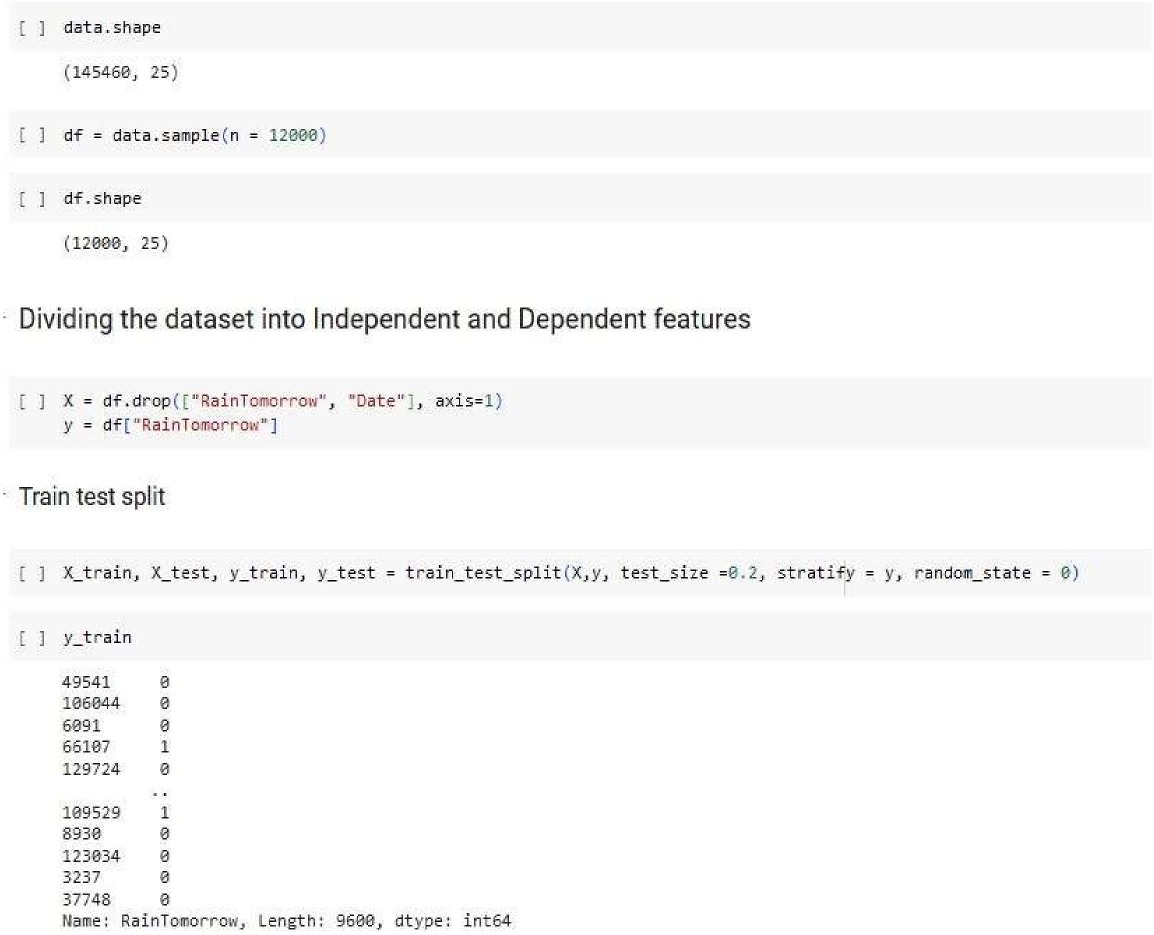
Features are the input variables used to make predictions, and labels are the output variables to be predicted.

### Training and Testing:

Datasets are typically split into training and testing sets to train the model and evaluate its performance.

### Evaluation Metrics:

Common metrics for classification tasks include accuracy, precision, recall, F1 score, and area under the receiver operating characteristic (ROC-AUC) curve.



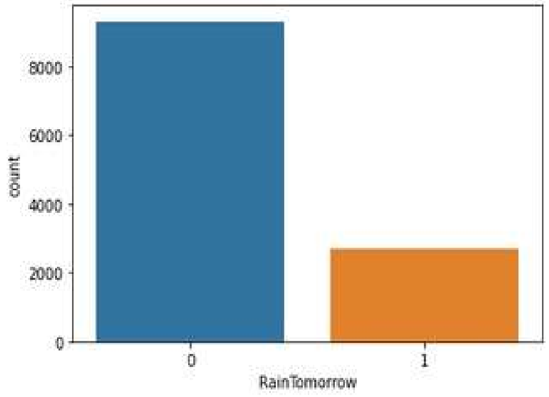
```
[ ] data.shape

    (145460, 25)

[ ] df = data.sample(n = 12000)

[ ] df.shape

    (12000, 25)
```

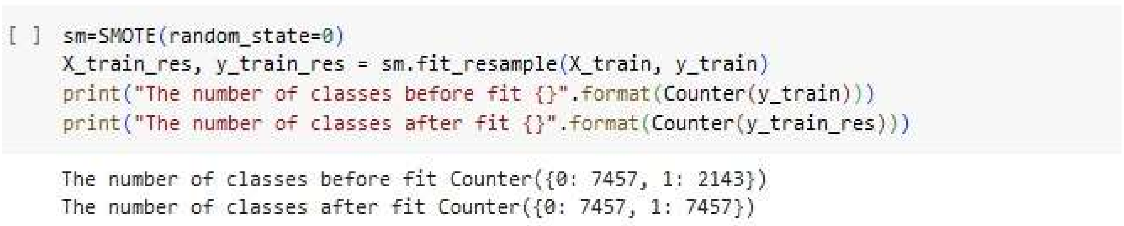
## Dividing the dataset into Independent and Dependent features

```
[ ] X = df.drop(["RainTomorrow", "Date"], axis=1)
    y = df["RainTomorrow"]
```

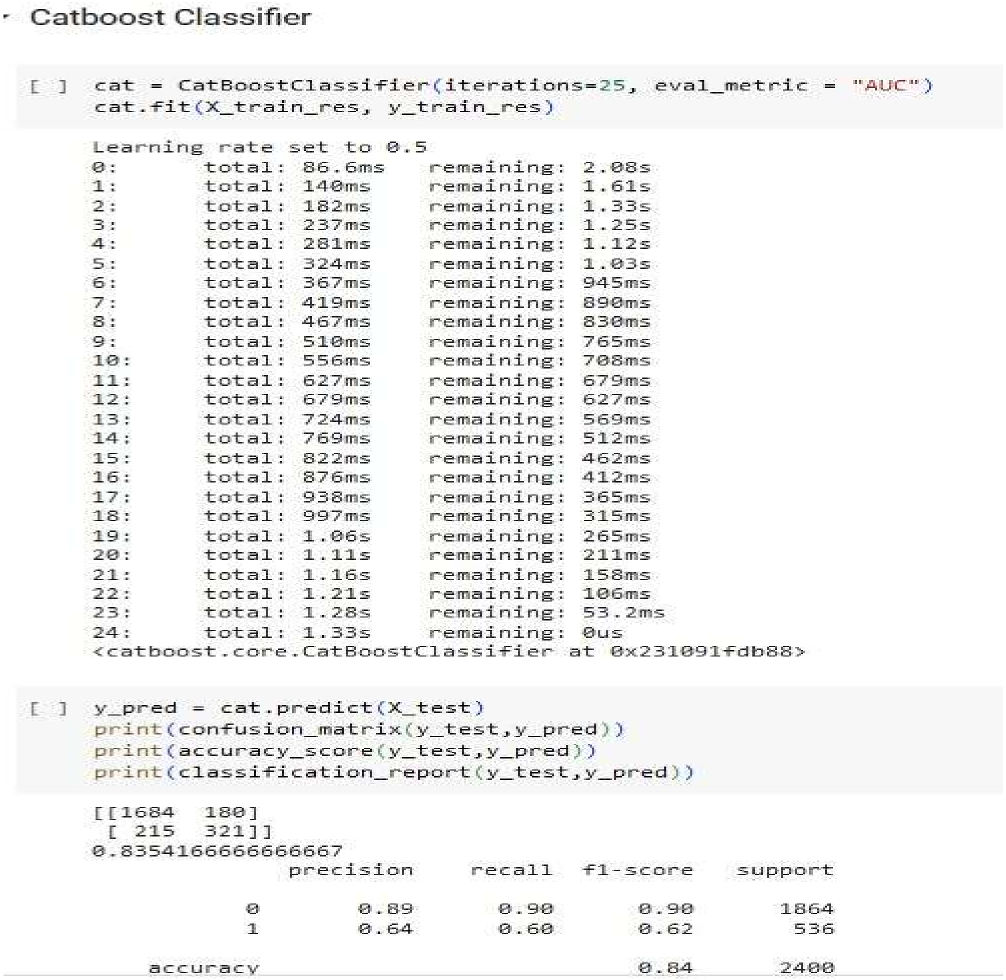
## Train test split

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

[ ] y_train

    49541     0
    106044    0
    6091      0
    66107     1
    129724    0
             ..
    109529    1
    8930      0
    123034    0
    3237      0
    37748     0
    Name: RainTomorrow, Length: 9600, dtype: int64
```





```
[ ] sm=SMOTE(random_state=0)
    X_train_res, y_train_res = sm.fit_resample(X_train, y_train)
    print("The number of classes before fit {}".format(Counter(y_train)))
    print("The number of classes after fit {}".format(Counter(y_train_res)))

    The number of classes before fit Counter({0: 7457, 1: 2143})
    The number of classes after fit Counter({0: 7457, 1: 7457})
```



## Catboost Classifier

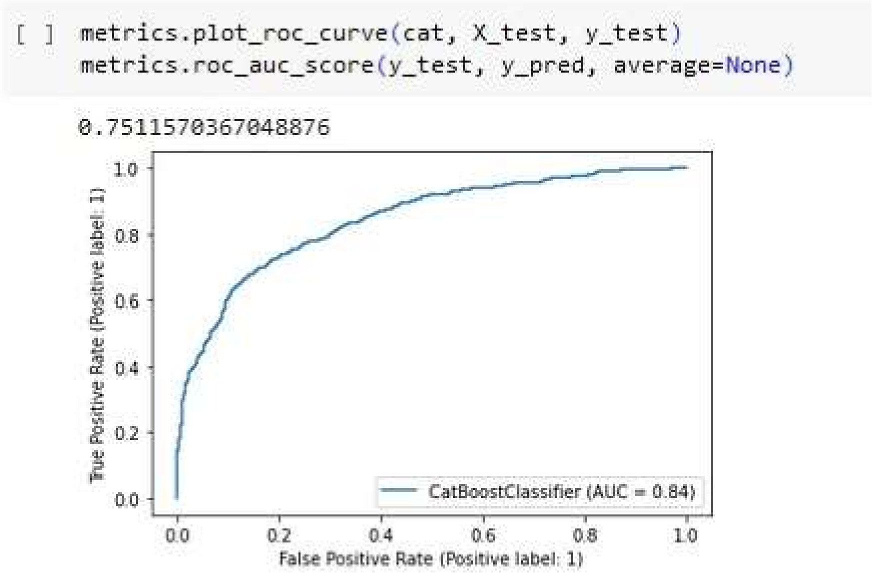
```
[ ]  cat = CatBoostClassifier(iterations=25, eval_metric = "AUC")
     cat.fit(X_train_res, y_train_res)
```

```
Learning rate set to 0.5
0:      total: 86.6ms   remaining: 2.08s
1:      total: 140ms    remaining: 1.61s
2:      total: 182ms    remaining: 1.33s
3:      total: 237ms    remaining: 1.25s
4:      total: 281ms    remaining: 1.12s
5:      total: 324ms    remaining: 1.03s
6:      total: 367ms    remaining: 945ms
7:      total: 419ms    remaining: 890ms
8:      total: 467ms    remaining: 830ms
9:      total: 510ms    remaining: 765ms
10:     total: 556ms    remaining: 708ms
11:     total: 627ms    remaining: 679ms
12:     total: 679ms    remaining: 627ms
13:     total: 724ms    remaining: 569ms
14:     total: 769ms    remaining: 512ms
15:     total: 822ms    remaining: 462ms
16:     total: 876ms    remaining: 412ms
17:     total: 938ms    remaining: 365ms
18:     total: 997ms    remaining: 315ms
19:     total: 1.06s    remaining: 265ms
20:     total: 1.11s    remaining: 211ms
21:     total: 1.16s    remaining: 158ms
22:     total: 1.21s    remaining: 106ms
23:     total: 1.28s    remaining: 53.2ms
24:     total: 1.33s    remaining: 0us
<catboost.core.CatBoostClassifier at 0x231091fdb88>
```

```
[ ]  y_pred = cat.predict(X_test)
     print(confusion_matrix(y_test,y_pred))
     print(accuracy_score(y_test,y_pred))
     print(classification_report(y_test,y_pred))
```

```
[[1684  180]
 [ 215  321]]
0.8354166666666667
              precision    recall  f1-score   support

           0       0.89      0.90      0.90      1864
           1       0.64      0.60      0.62       536

    accuracy                           0.84      2400
```

ti

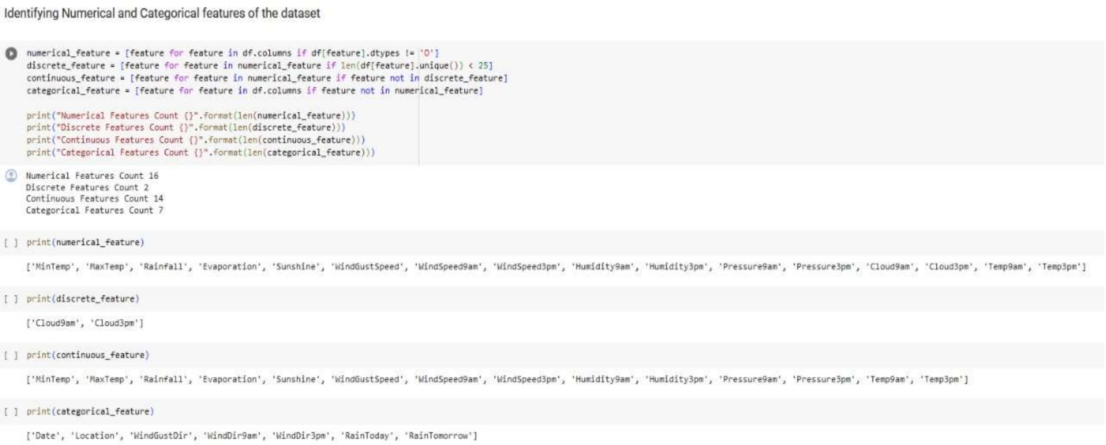


```
[ ]  metrics.plot_roc_curve(cat, X_test, y_test)
     metrics.roc_auc_score(y_test, y_pred, average=None)
```

```
0.7511570367048876
```

1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate (Positive label: 1)
CatBoostClassifier (AUC = 0.84)
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate (Positive label: 1)

## DATA PROCESSING:



Identifying Numerical and Categorical features of the dataset

```
numerical_feature = [feature for feature in df.columns if df[feature].dtypes != 'O']
discrete_feature = [feature for feature in numerical_feature if len(df[feature].unique()) < 25]
continuous_feature = [feature for feature in numerical_feature if feature not in discrete_feature]
categorical_feature = [feature for feature in df.columns if feature not in numerical_feature]

print("Numerical Features Count {}".format(len(numerical_feature)))
print("Discrete Features Count {}".format(len(discrete_feature)))
print("Continuous Features Count {}".format(len(continuous_feature)))
print("Categorical Features Count {}".format(len(categorical_feature)))
```

```
Numerical Features Count 16
Discrete Features Count 2
Continuous Features Count 14
Categorical Features Count 7
```

```
[ ] print(numerical_feature)
```

```
['MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine', 'WindGustSpeed', 'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm', 'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am', 'Temp3pm']
```

```
[ ] print(discrete_feature)
```

```
['Cloud9am', 'Cloud3pm']
```

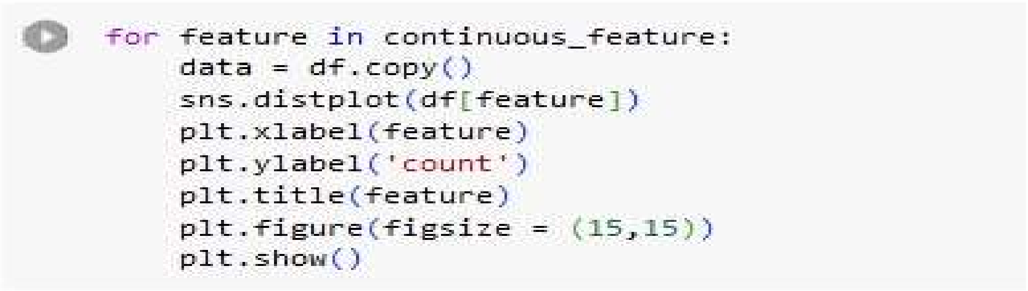
```
[ ] print(continuous_feature)
```

```
['MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine', 'WindGustSpeed', 'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm', 'Pressure9am', 'Pressure3pm', 'Temp9am', 'Temp3pm']
```

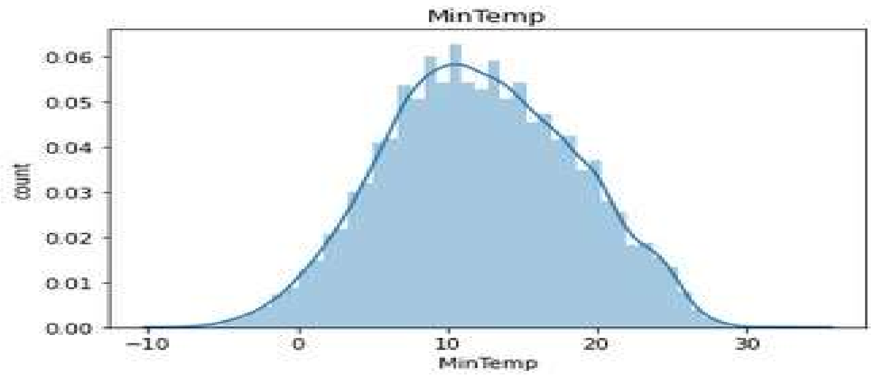
```
[ ] print(categorical_feature)
```

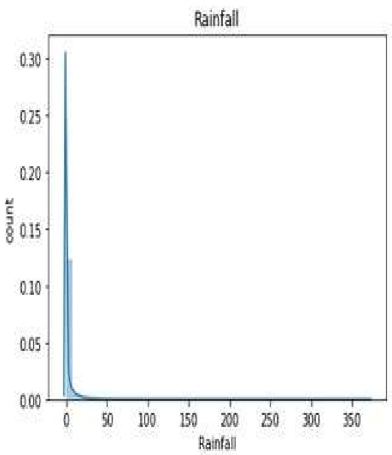
```
['Date', 'Location', 'WindGustDir', 'WindDir9am', 'WindDir3pm', 'RainToday', 'RainTomorrow']
```

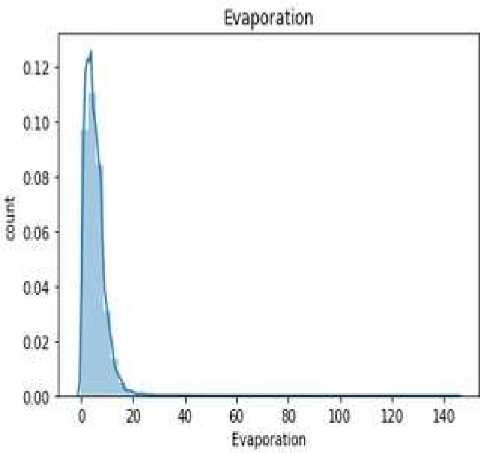
- This line creates a list called numerical_feature using a list comprehension. It iterates over all column names in the DataFrame (df). For each column, it checks if the datatype (dtypes) is not 'O' (not an object, usually indicating non-string data). If the datatype is not 'O', the column name is added to the numerical_feature list. So, numerical_feature contains the names of columns that are considered numerical.
- Here, it creates a list called discrete_feature using another list comprehension. It filters the numerical_feature list to include only those columns where the number of unique values is less than 25. This is a heuristic to identify discrete features. If a numerical feature has a small number of unique values, it's likely discrete.
- This line creates a list called continuous_feature using a list comprehension. It includes numerical features that are not in the discrete_feature list. The assumption here is that features not classified as discrete are continuous.

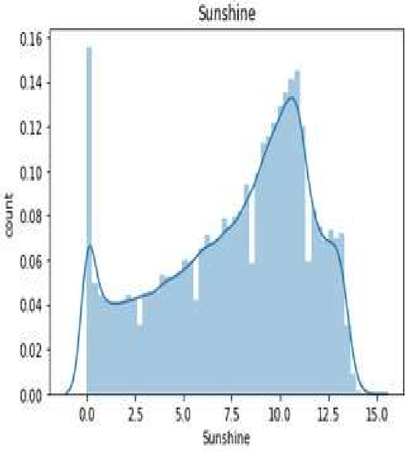


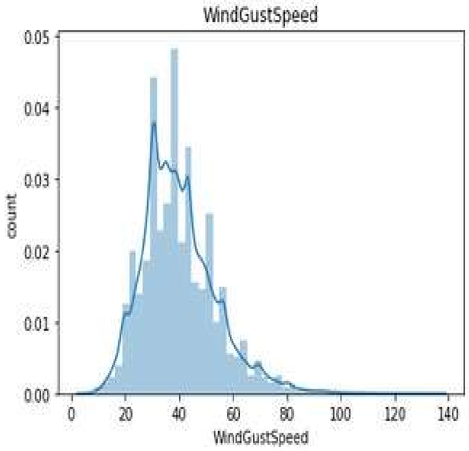
```
for feature in continuous_feature:
    data = df.copy()
    sns.distplot(df[feature])
    plt.xlabel(feature)
    plt.ylabel('count')
    plt.title(feature)
    plt.figure(figsize = (15,15))
    plt.show()
```

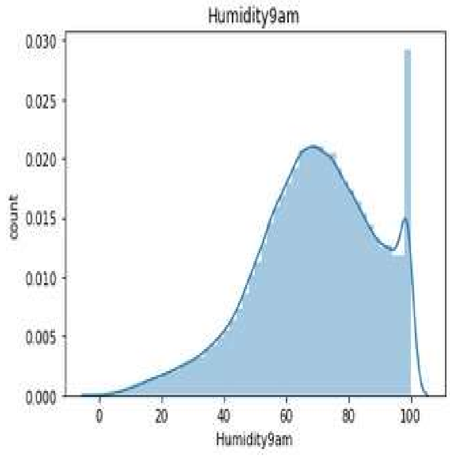


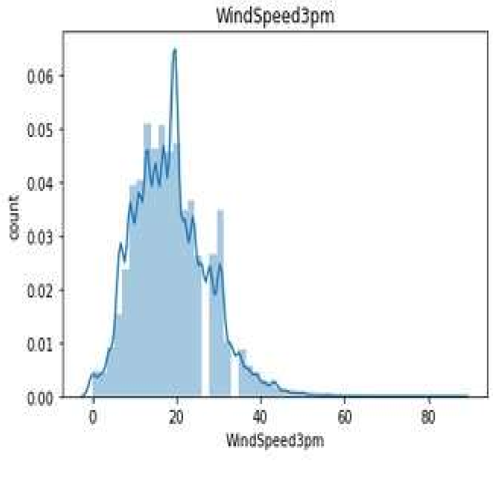


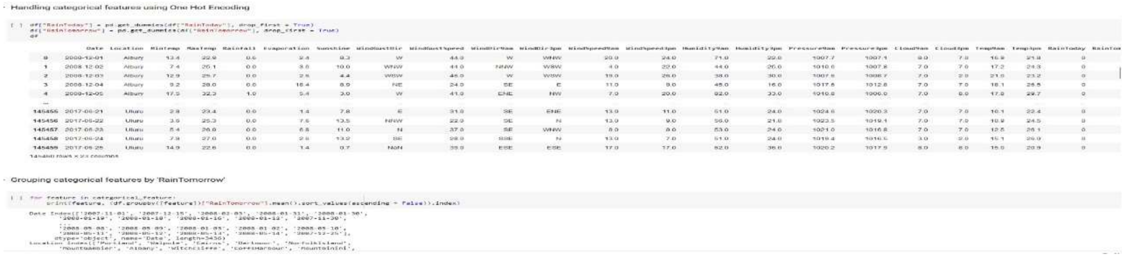












Handling categorical features using One Hot Encoding

```
[ ] df["RainToday"] = pd.get_dummies(df["RainToday"], drop_first = True)
    df["RainTomorrow"] = pd.get_dummies(df["RainTomorrow"], drop_first = True)
    df
```

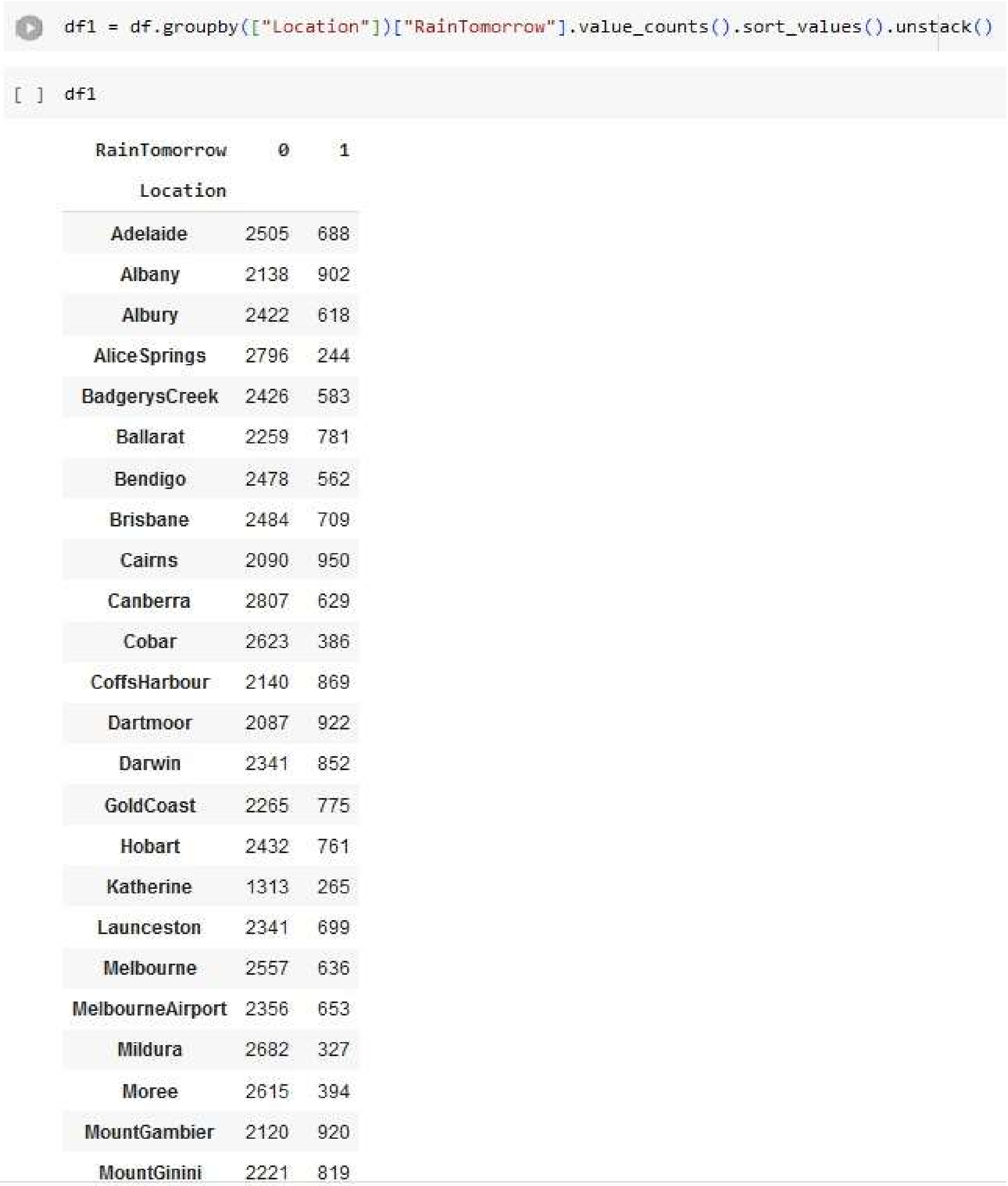
| | Date | Location | MinTemp | MaxTemp | Rainfall | Evaporation | Sunshine | WindGustDir | WindGustSpeed | WindDir9am | WindDir3pm | WindSpeed9am | WindSpeed3pm | Humidity9am | Humidity3pm | Pressure9am | Pressure3pm | Cloud9am | Cloud3pm | Temp9am | Temp3pm | RainToday | RainTom |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2008-12-01 | Albury | 13.4 | 22.9 | 0.6 | 2.4 | 8.3 | W | 44.0 | W | WNW | 20.0 | 24.0 | 71.0 | 22.0 | 1007.7 | 1007.1 | 8.0 | 7.0 | 16.9 | 21.8 | 0 | |
| 1 | 2008-12-02 | Albury | 7.4 | 25.1 | 0.0 | 3.6 | 10.0 | WNW | 44.0 | NNW | WSW | 4.0 | 22.0 | 44.0 | 25.0 | 1010.6 | 1007.8 | 7.0 | 7.0 | 17.2 | 24.3 | 0 | |
| 2 | 2008-12-03 | Albury | 12.9 | 25.7 | 0.0 | 2.6 | 4.4 | WSW | 46.0 | W | WSW | 19.0 | 26.0 | 38.0 | 30.0 | 1007.6 | 1008.7 | 7.0 | 2.0 | 21.0 | 23.2 | 0 | |
| 3 | 2008-12-04 | Albury | 9.2 | 28.0 | 0.0 | 18.4 | 8.9 | NE | 24.0 | SE | E | 11.0 | 9.0 | 45.0 | 16.0 | 1017.6 | 1012.8 | 7.0 | 7.0 | 18.1 | 26.5 | 0 | |
| 4 | 2008-12-05 | Albury | 17.5 | 32.3 | 1.0 | 5.4 | 3.0 | W | 41.0 | ENE | NW | 7.0 | 20.0 | 82.0 | 33.0 | 1010.8 | 1006.0 | 7.0 | 8.0 | 17.8 | 29.7 | 0 | |
| ... | | | | | | | | | | | | | | | | | | | | | | | |
| 145455 | 2017-06-21 | Uluru | 2.8 | 23.4 | 0.0 | 1.4 | 7.8 | E | 31.0 | SE | ENE | 13.0 | 11.0 | 51.0 | 24.0 | 1024.6 | 1020.3 | 7.0 | 7.0 | 10.1 | 22.4 | 0 | |
| 145456 | 2017-06-22 | Uluru | 3.6 | 25.3 | 0.0 | 7.6 | 13.5 | NNW | 22.0 | SE | N | 13.0 | 9.0 | 56.0 | 21.0 | 1023.5 | 1019.1 | 7.0 | 7.0 | 10.9 | 24.5 | 0 | |
| 145457 | 2017-06-23 | Uluru | 5.4 | 26.9 | 0.0 | 6.8 | 11.0 | N | 37.0 | SE | WNW | 9.0 | 9.0 | 53.0 | 24.0 | 1021.0 | 1016.8 | 7.0 | 7.0 | 12.5 | 26.1 | 0 | |
| 145458 | 2017-06-24 | Uluru | 7.8 | 27.0 | 0.0 | 2.6 | 13.2 | SE | 28.0 | SSE | N | 13.0 | 7.0 | 51.0 | 24.0 | 1019.4 | 1016.5 | 3.0 | 2.0 | 15.1 | 26.0 | 0 | |
| 145459 | 2017-06-25 | Uluru | 14.9 | 22.6 | 0.0 | 1.4 | 0.7 | NaN | 39.0 | ESE | ESE | 17.0 | 17.0 | 62.0 | 36.0 | 1020.2 | 1017.9 | 8.0 | 8.0 | 15.0 | 20.9 | 0 | |

145460 rows × 23 columns

Grouping categorical features by 'RainTomorrow'

```
[ ] for feature in categorical_feature:
        print(feature, (df.groupby([feature])["RainTomorrow"].mean().sort_values(ascending = False)).index)
```

```
Date Index(['2007-11-01', '2007-12-15', '2008-02-03', '2008-01-31', '2008-01-30',
            '2008-01-19', '2008-01-10', '2008-01-16', '2008-01-12', '2007-11-30',
            ...
            '2008-05-08', '2008-05-09', '2008-01-05', '2008-01-02', '2008-05-10',
            '2008-05-11', '2008-05-12', '2008-05-13', '2008-05-14', '2007-12-25'],
           dtype='object', name='Date', length=3436)
Location Index(['Portland', 'Walpole', 'Cairns', 'Dartmoor', 'NorfolkIsland',
            'MountGambier', 'Albany', 'Witchcliffe', 'CoffsHarbour', 'MountGinini',
```

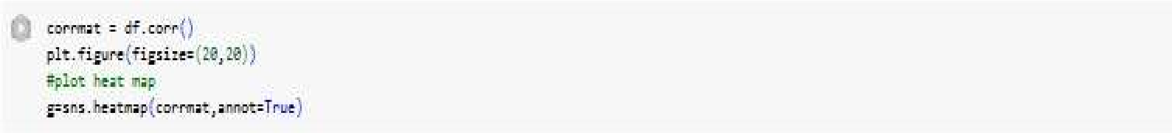


```
df1 = df.groupby(["Location"])["RainTomorrow"].value_counts().sort_values().unstack()
```

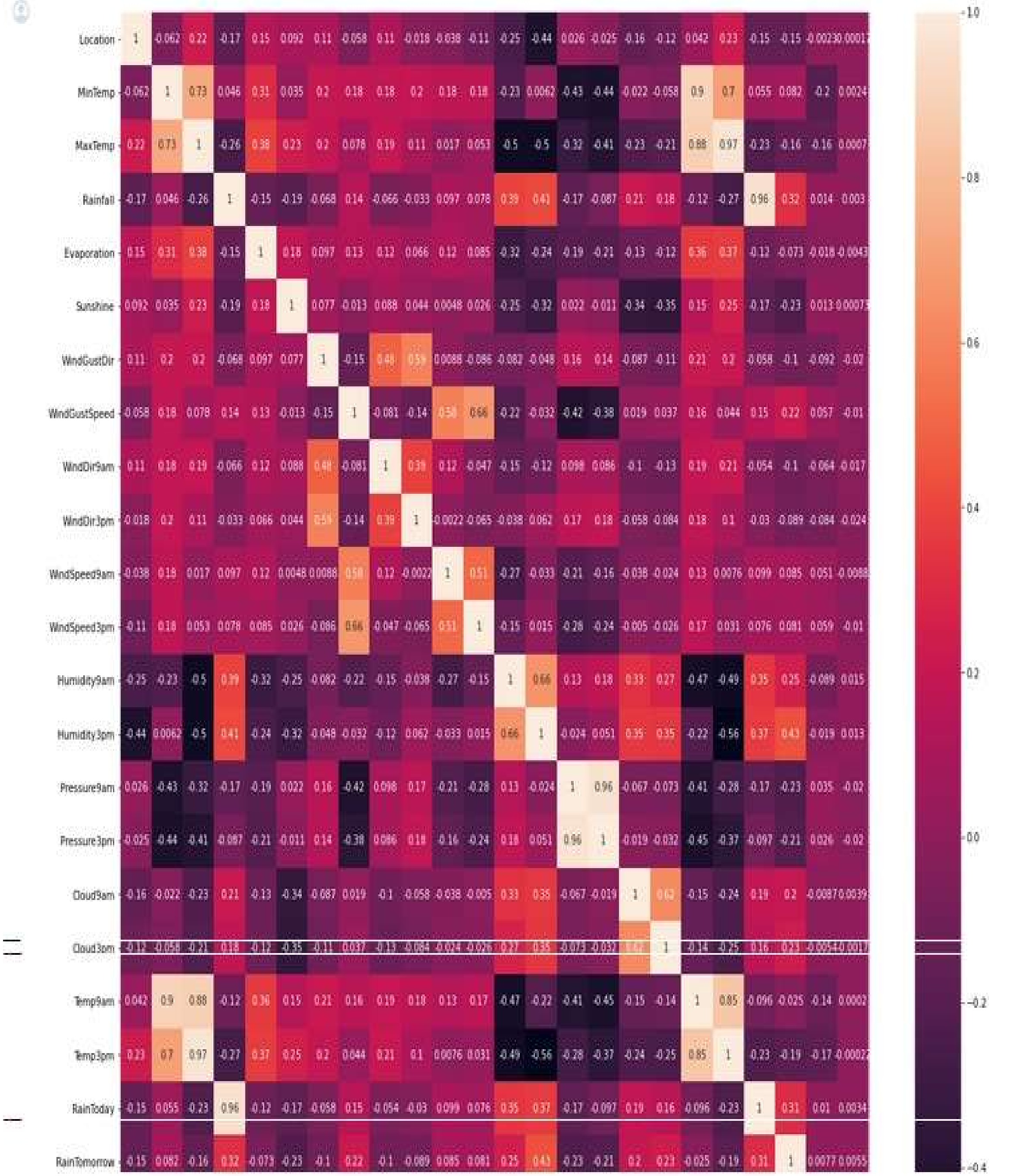
```
[ ] df1
```

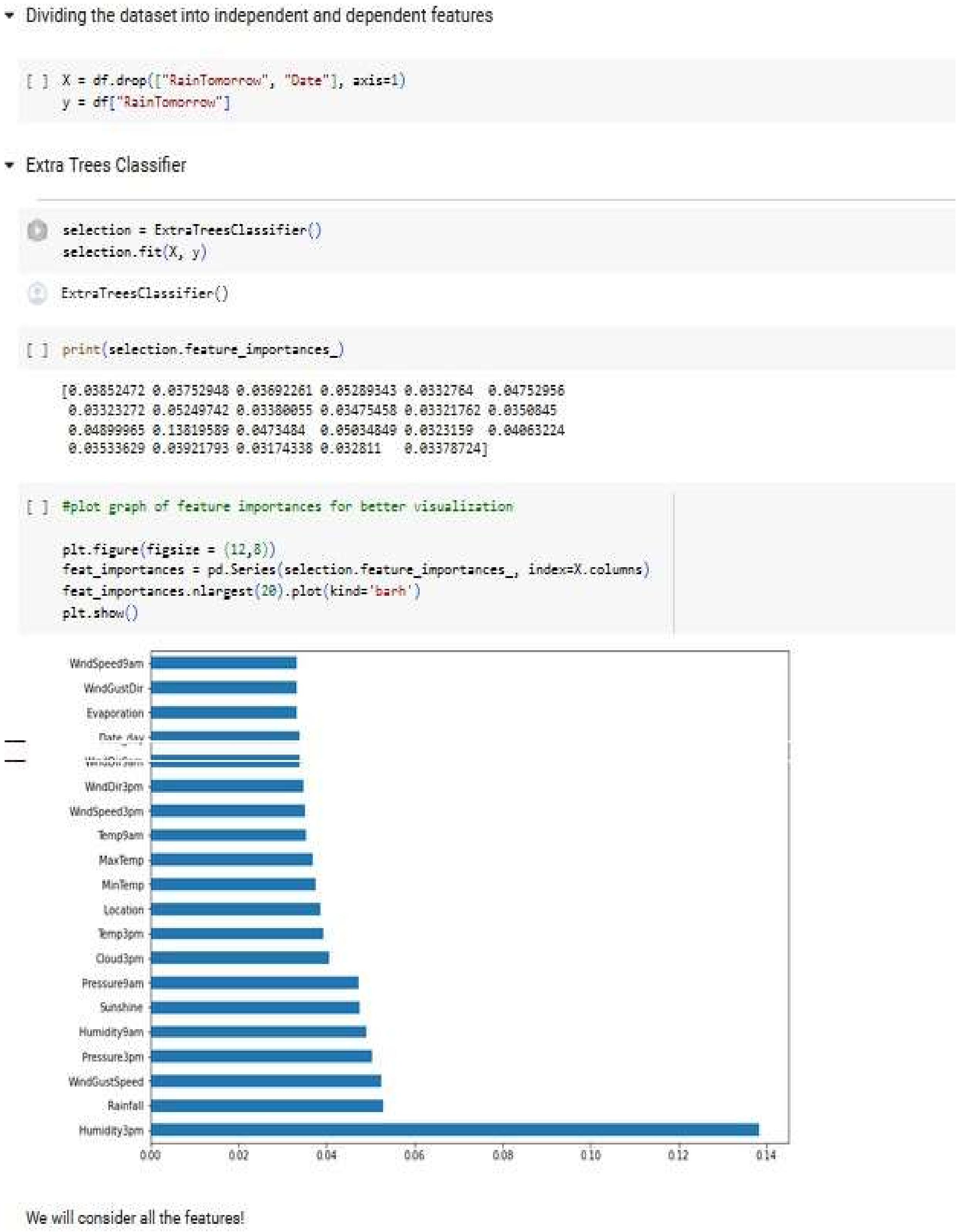
```
    RainTomorrow     0    1
        Location
        Adelaide  2505  688
          Albany  2138  902
          Albury  2422  618
    AliceSprings  2796  244
   BadgerysCreek  2426  583
        Ballarat  2259  781
         Bendigo  2478  562
        Brisbane  2484  709
          Cairns  2090  950
        Canberra  2807  629
           Cobar  2623  386
    CoffsHarbour  2140  869
        Dartmoor  2087  922
          Darwin  2341  852
       GoldCoast  2265  775
          Hobart  2432  761
       Katherine  1313  265
      Launceston  2341  699
       Melbourne  2557  636
MelbourneAirport  2356  653
         Mildura  2682  327
           Moree  2615  394
    MountGambier  2120  920
     MountGinini  2221  819
```

## **FUTURE SELECTION**:



```
corrmat = df.corr()
plt.figure(figsize=(20,20))
#plot heat map
g=sns.heatmap(corrmat,annot=True)
```





## Dividing the dataset into independent and dependent features

```
[ ] X = df.drop(["RainTomorrow", "Date"], axis=1)
    y = df["RainTomorrow"]
```

## Extra Trees Classifier

```
selection = ExtraTreesClassifier()
selection.fit(X, y)
```

```
ExtraTreesClassifier()
```

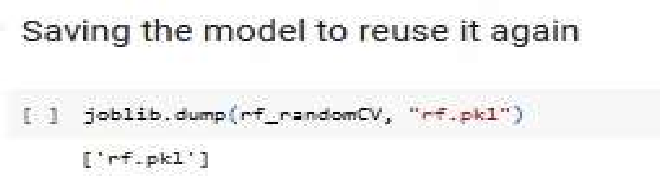
```
[ ] print(selection.feature_importances_)
```

```
[0.03852472 0.03752948 0.03692261 0.05289343 0.0332764  0.04752956
 0.03323272 0.05249742 0.03380055 0.03475458 0.03321762 0.0350845
 0.04899965 0.13819589 0.0473484  0.05034849 0.0323159  0.04063224
 0.03533629 0.03921793 0.03174338 0.032811   0.03378724]
```

```
[ ] #plot graph of feature importances for better visualization

    plt.figure(figsize = (12,8))
    feat_importances = pd.Series(selection.feature_importances_, index=X.columns)
    feat_importances.nlargest(20).plot(kind='barh')
    plt.show()
```

We will consider all the features!

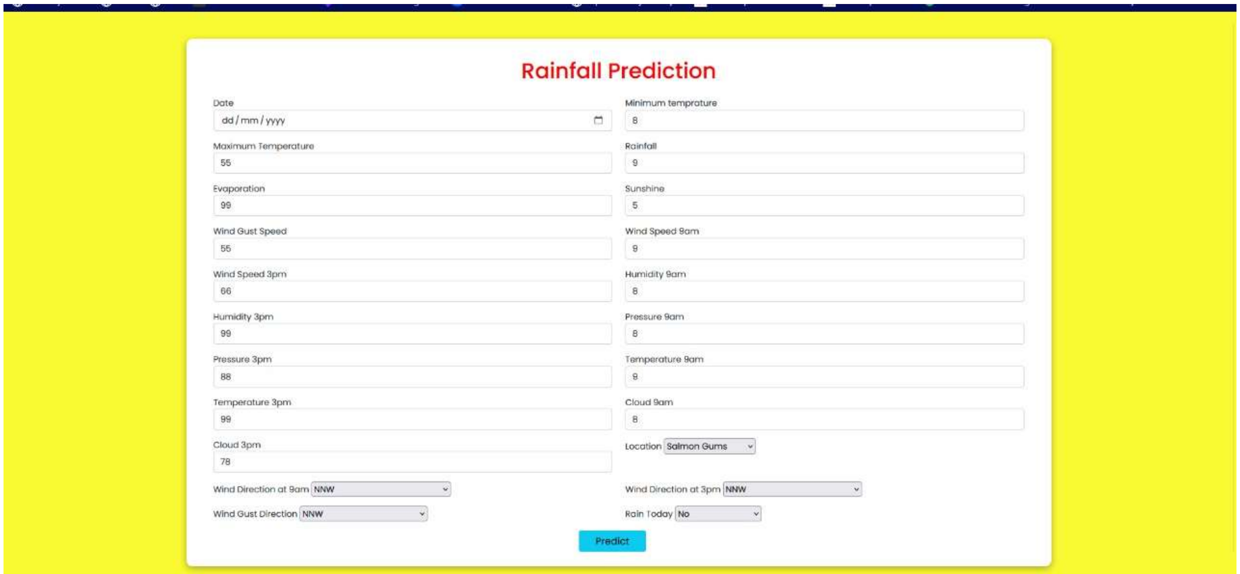
**<u>SAVING MODULE</u>**:



Saving the model to reuse it again

```
[ ]  joblib.dump(rf_randomCV, "rf.pkl")

     ['rf.pkl']
```

**<u>WEBSITE LOOKS LIKE</u>**:



Rainfall Prediction

Date
dd / mm / yyyy

Maximum Temperature
55

Evaporation
99

Wind Gust Speed
55

Wind Speed 3pm
66

Humidity 3pm
99

Pressure 3pm
88

Temperature 3pm
99

Cloud 3pm
78

Wind Direction at 9am NNW

Wind Gust Direction NNW

Minimum temprature
8

Rainfall
9

Sunshine
5

Wind Speed 9am
9

Humidity 9am
8

Pressure 9am
8

Temperature 9am
9

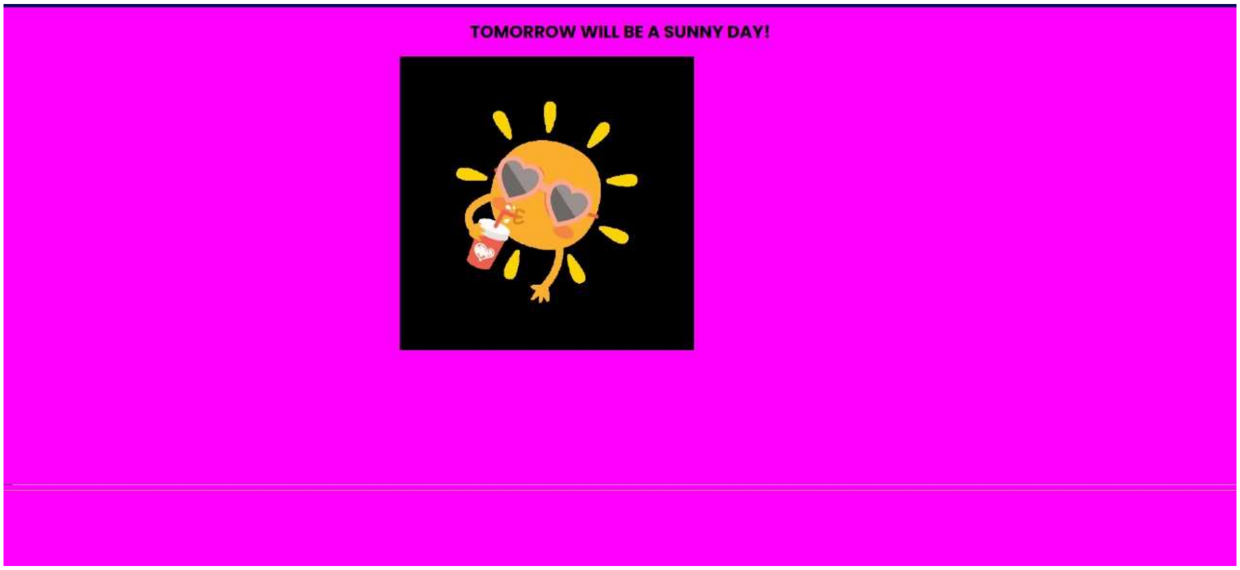
Cloud 9am
8

Location Salmon Gums

Wind Direction at 3pm NNW

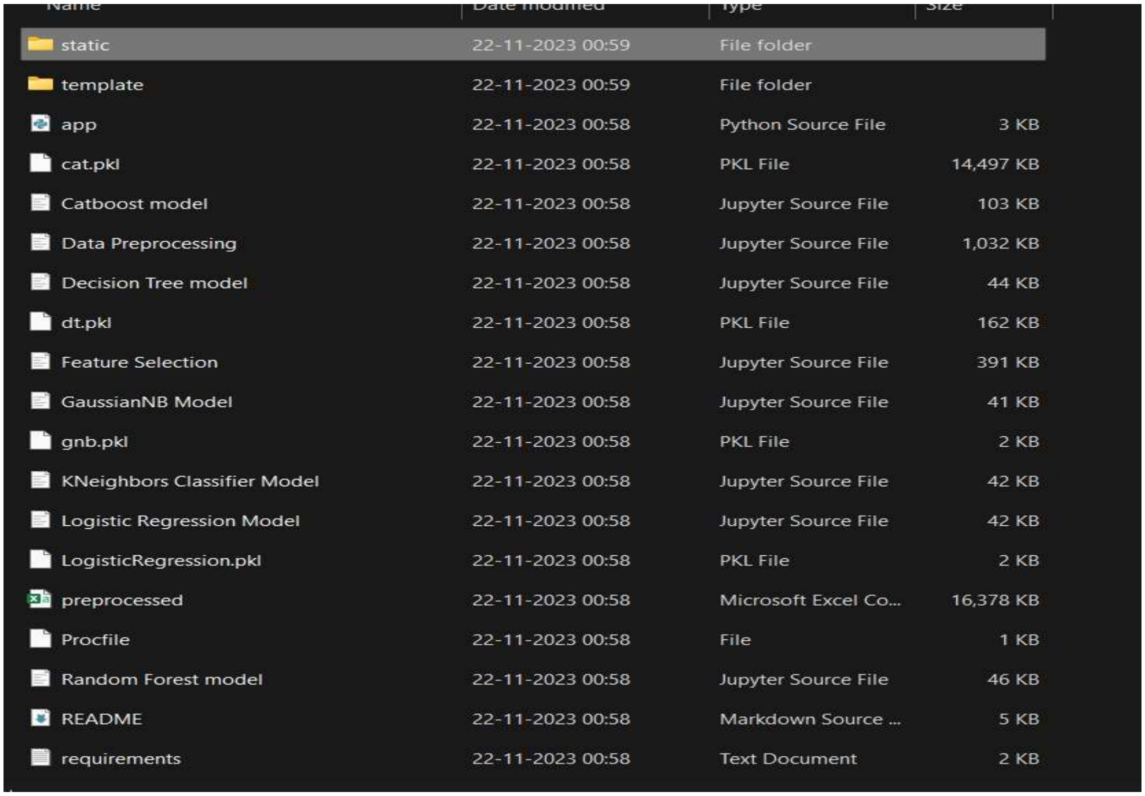
Rain Today No

**<u>OUTPUT BE LIKE</u>**:



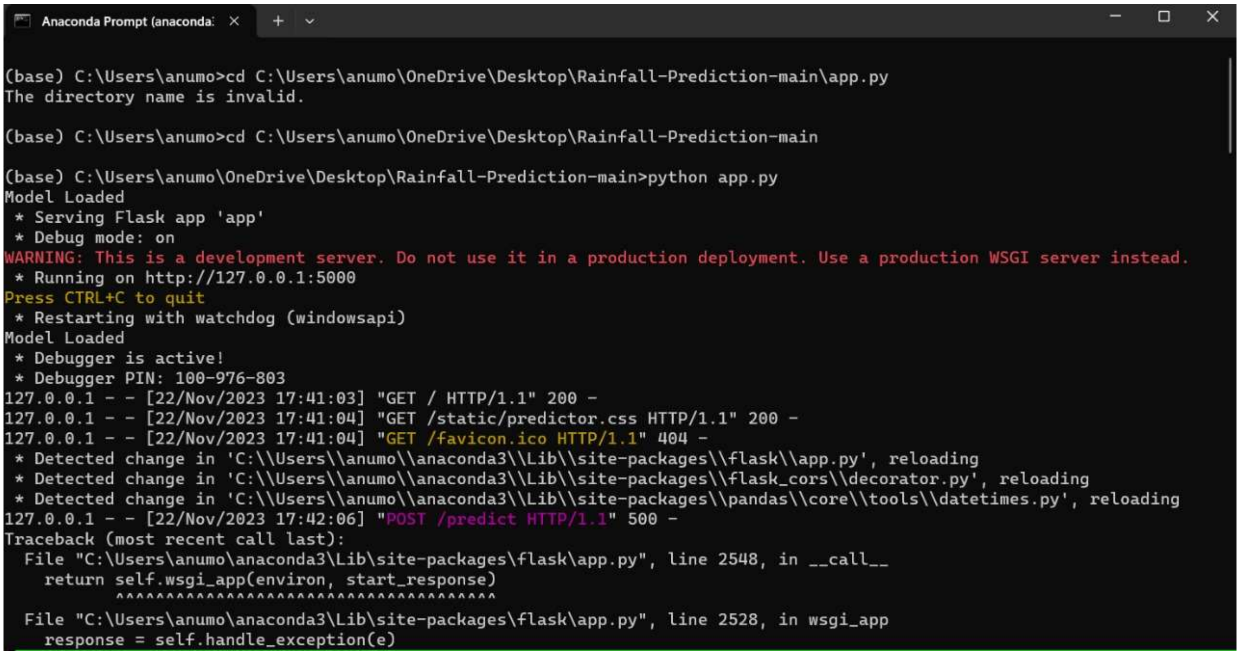
TOMORROW WILL BE A SUNNY DAY!

**ALL FILES**:



| Name | Date modified | Type | Size |
|---|---|---|---|
| static | 22-11-2023 00:59 | File folder | |
| template | 22-11-2023 00:59 | File folder | |
| app | 22-11-2023 00:58 | Python Source File | 3 KB |
| cat.pkl | 22-11-2023 00:58 | PKL File | 14,497 KB |
| Catboost model | 22-11-2023 00:58 | Jupyter Source File | 103 KB |
| Data Preprocessing | 22-11-2023 00:58 | Jupyter Source File | 1,032 KB |
| Decision Tree model | 22-11-2023 00:58 | Jupyter Source File | 44 KB |
| dt.pkl | 22-11-2023 00:58 | PKL File | 162 KB |
| Feature Selection | 22-11-2023 00:58 | Jupyter Source File | 391 KB |
| GaussianNB Model | 22-11-2023 00:58 | Jupyter Source File | 41 KB |
| gnb.pkl | 22-11-2023 00:58 | PKL File | 2 KB |
| KNeighbors Classifier Model | 22-11-2023 00:58 | Jupyter Source File | 42 KB |
| Logistic Regression Model | 22-11-2023 00:58 | Jupyter Source File | 42 KB |
| LogisticRegression.pkl | 22-11-2023 00:58 | PKL File | 2 KB |
| preprocessed | 22-11-2023 00:58 | Microsoft Excel Co... | 16,378 KB |
| Procfile | 22-11-2023 00:58 | File | 1 KB |
| Random Forest model | 22-11-2023 00:58 | Jupyter Source File | 46 KB |
| README | 22-11-2023 00:58 | Markdown Source ... | 5 KB |
| requirements | 22-11-2023 00:58 | Text Document | 2 KB |

**RUNING app.py IN ANNACONDA PROMT**:



```
(base) C:\Users\anumo>cd C:\Users\anumo\OneDrive\Desktop\Rainfall-Prediction-main\app.py
The directory name is invalid.

(base) C:\Users\anumo>cd C:\Users\anumo\OneDrive\Desktop\Rainfall-Prediction-main

(base) C:\Users\anumo\OneDrive\Desktop\Rainfall-Prediction-main>python app.py
Model Loaded
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with watchdog (windowsapi)
Model Loaded
 * Debugger is active!
 * Debugger PIN: 100-976-803
127.0.0.1 - - [22/Nov/2023 17:41:03] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [22/Nov/2023 17:41:04] "GET /static/predictor.css HTTP/1.1" 200 -
127.0.0.1 - - [22/Nov/2023 17:41:04] "GET /favicon.ico HTTP/1.1" 404 -
 * Detected change in 'C:\\Users\\anumo\\anaconda3\\Lib\\site-packages\\flask\\app.py', reloading
 * Detected change in 'C:\\Users\\anumo\\anaconda3\\Lib\\site-packages\\flask_cors\\decorator.py', reloading
 * Detected change in 'C:\\Users\\anumo\\anaconda3\\Lib\\site-packages\\pandas\\core\\tools\\datetimes.py', reloading
127.0.0.1 - - [22/Nov/2023 17:42:06] "POST /predict HTTP/1.1" 500 -
Traceback (most recent call last):
  File "C:\Users\anumo\anaconda3\Lib\site-packages\flask\app.py", line 2548, in __call__
    return self.wsgi_app(environ, start_response)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\anumo\anaconda3\Lib\site-packages\flask\app.py", line 2528, in wsgi_app
    response = self.handle_exception(e)
```