CONTENTS

EXPLORATORY ANALYSIS OF RAINFALL DATA IN INDIA FOR AGRICULTURE

1. INTRODUCTION

1.1. Project Overview

India is an agricultural country and secondary agri-based market will be steady with a good monsoon. The economic growth of each year depends on the amount of duration of monsoon rain, bad monsoon can lead to destruction of some crops, which may result in scarcity of some agricultural products which in turn can cause food inflation, insecurity and public unrest. In our analysis we are trying to understand the behavior of rainfall in India over the years, by
months and different subdivisions.

Agriculture is the backbone of the Indian economy. For agriculture, the most important thing is water source, i.e., rainfall. The prediction of the amount of rainfall gives alertness to farmers by knowing early they can protect their crops from rain. So, it is important to predict the rainfall accurately as much as possible. Exploration and analysis of data on rainfall over various regions of India and especially the regions where agricultural works have been done persistently in a wide range. With the help of analysis and the resultant data, future rainfall prediction for those regions using various machine learning techniques such as Logistic Regression, Linear Regression, Cat boost Classifier etc.

1.2. Purpose

The main aim of objective is to find the

- Rainfall Prediction is the application of science and technology to predict the amount of rainfall over a region.
- It is important to exactly determine the rainfall for effective use of water resources, crop productivity and pre-planning of water structures.

2. LITERATURE SURVEY

2.1. Existing Problem

Climate is important aspect of human life. So, the Prediction should accurate as much as possible. In this paper we try to deal with the prediction of the rainfall which is also a major aspect of human life, and which provide the major resource of human life which is Fresh Water. Fresh water is always a crucial resource of human survival – not only for the drinking purposes but also for farming, washing and many other purposes. Making a good prediction of climate is always a major task because of the climate change.

Now climate change is the biggest issue all over the world. Peoples are working on to detect the patterns in climate change as it affects the economy in production to infrastructure. So as in rainfall also making prediction of rainfall is a challenging task with a good accuracy rate. Making prediction on rainfall cannot be done by the traditional way, so scientist is using machine learning and deep learning to find out the pattern for rainfall prediction.

A bad rainfall prediction can affect the agriculture mostly framers as their whole crop is dependent on the rainfall and agriculture. It is always an important part of every economy. So, making an accurate prediction on the rainfall. There are number of techniques are used of machine learning, but accuracy is always a matter of concern in prediction made in rainfall.

There are number of causes made by rainfall affecting the world ex. Drought, Flood, and intense summer heat etc. And it will also affect water resources around the world.

2.2.References

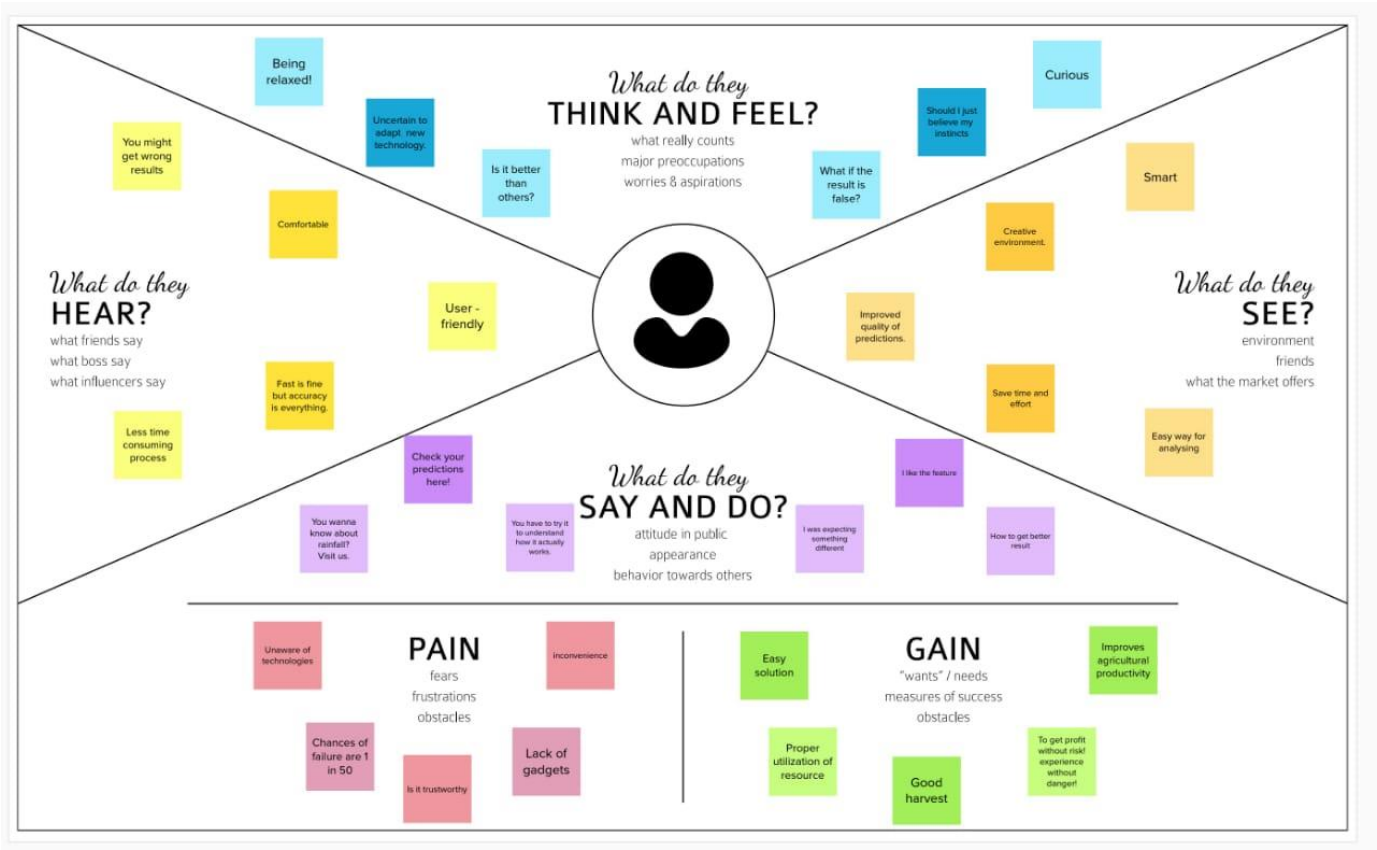
PROJECT TITLE	AUTHOR	OBJECTIVE/OUTCOME
Spatial analysis of Indian Summer monsoon Rainfall (Mar 26,2014)	Markand Oza C.M.Kishtawal	Understanding the variability in rainfall, analysis of IndianSummer monsoon rainfall using Spatial resolution.
Climate impacts on Indian Agriculture. (16 June,2004)	K.Krishna Kumar K.Rupa Kumar R.G.Ashrit N.R.Deshpande J.W.Hansen	Presents about the analysis of Crop-climate relationships for India, using historical predictions.
Exploratory data Analysis of Indian Rainfall Data	Anusha Gajinkar	This Study shows that, India has two monsoon rainfall season one is northwest monsoon and second one is southeast monsoon.

2.3. Problem Statement Definition

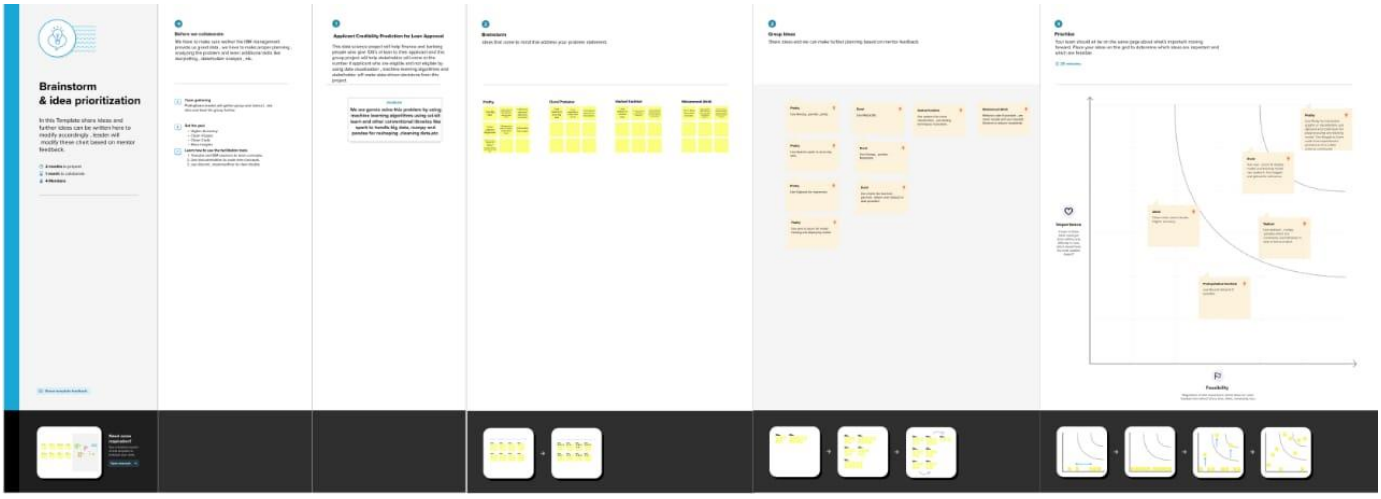
Ramu is a farmer who wants to know the rainfall rate for a certain period because he couldn't afford any further loss.

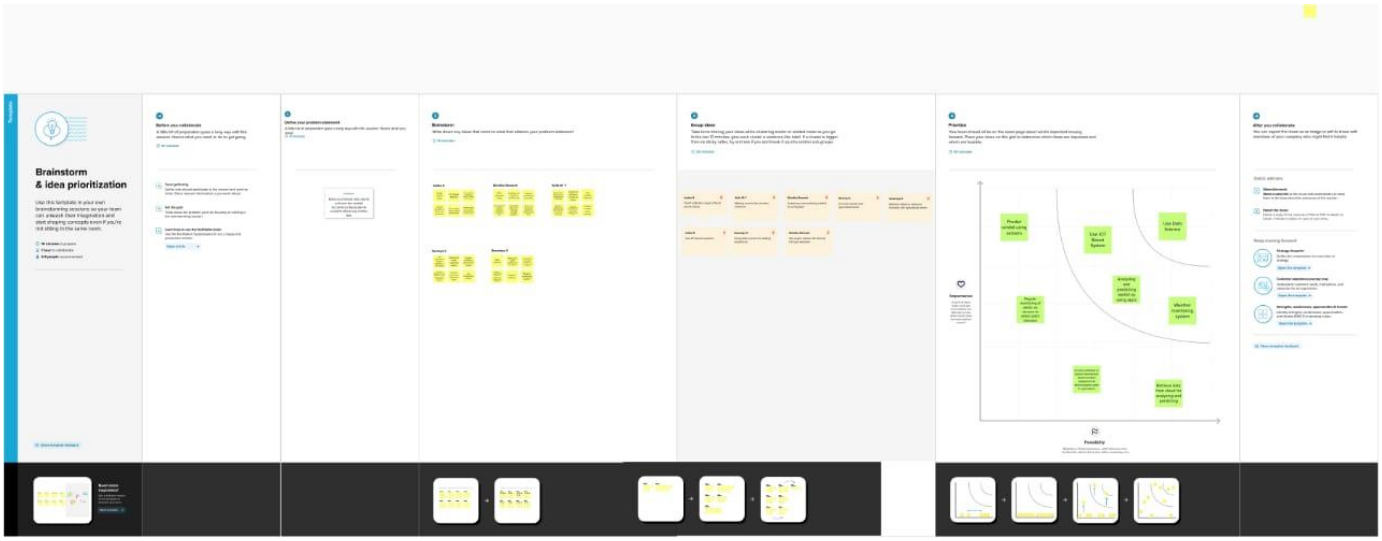
3. IDEATION AND PROPOSED SOLUTION

3.1. Empathy Map Canvas



3.2.Ideation and Brainstorming

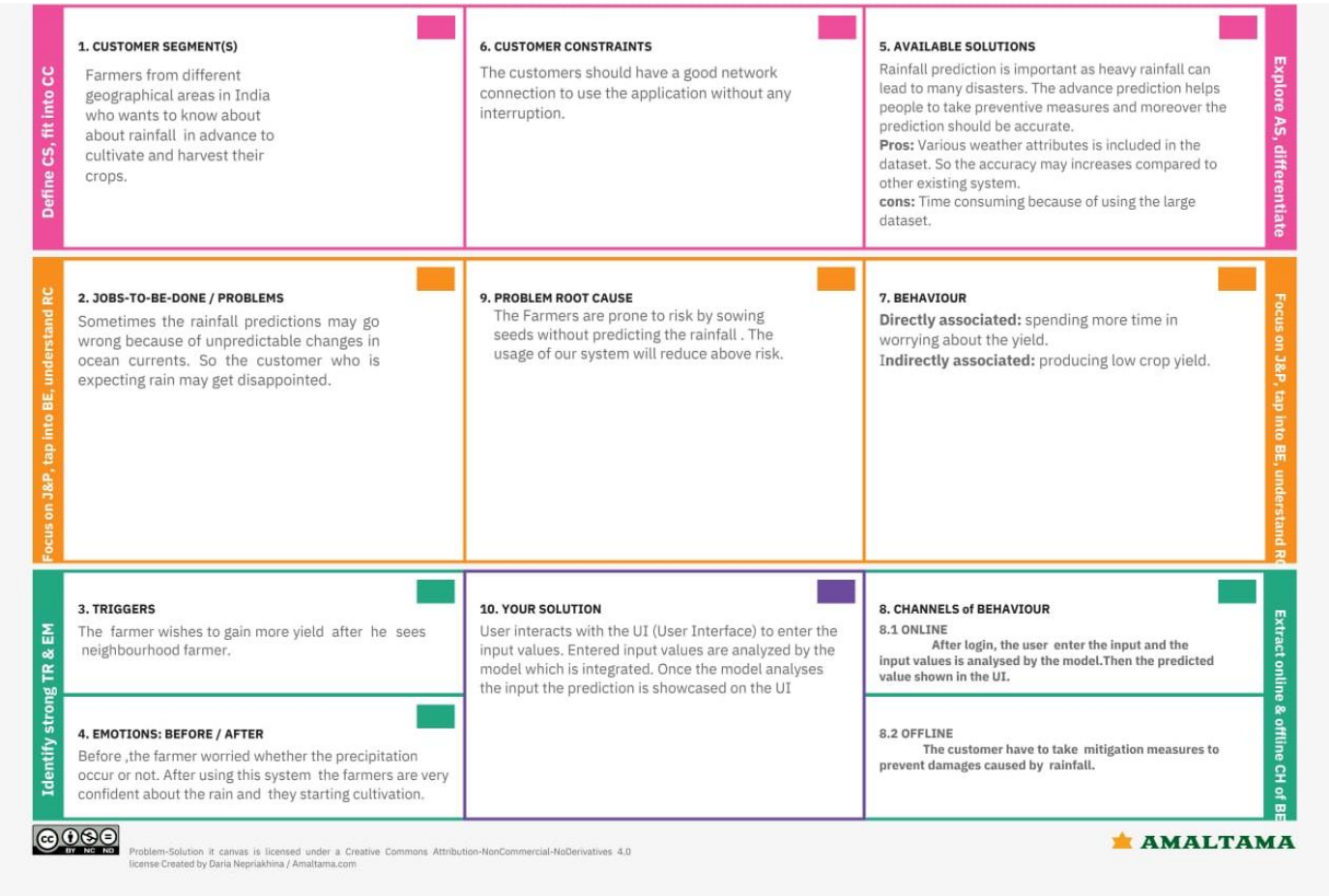




3.3. Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	How can we predict the Rainfall and the weather for better crop yield?
2.	Idea / Solution description	Ensures correct prediction of rainfall by analysing the weather which helps the farmers to protect crops from rainfall.
3.	Novelty / Uniqueness	1)Predict rainfall using data science. 2)Analyse the weather condition 3)Ensure the safety of crops.
4.	Social Impact / Customer Satisfaction	Serves as farmer's friend by helping them to take precautionary steps to minimize the economic losses.
5.	Business Model (Revenue Model)	1)Collaboration with agriculture-sector. 2)Providing technical solutions. 3)Increase in crop productivity.
6.	Scalability of the Solution	1)Analyse the climate data from meteorological stations and from food and Agricultural organizations. 2)Estimating climate change impacts on Indian agriculture.

3.4.Proposed Solution Fit



4. REQUIREMENT ANALYSIS

4.1. Functional Requirements

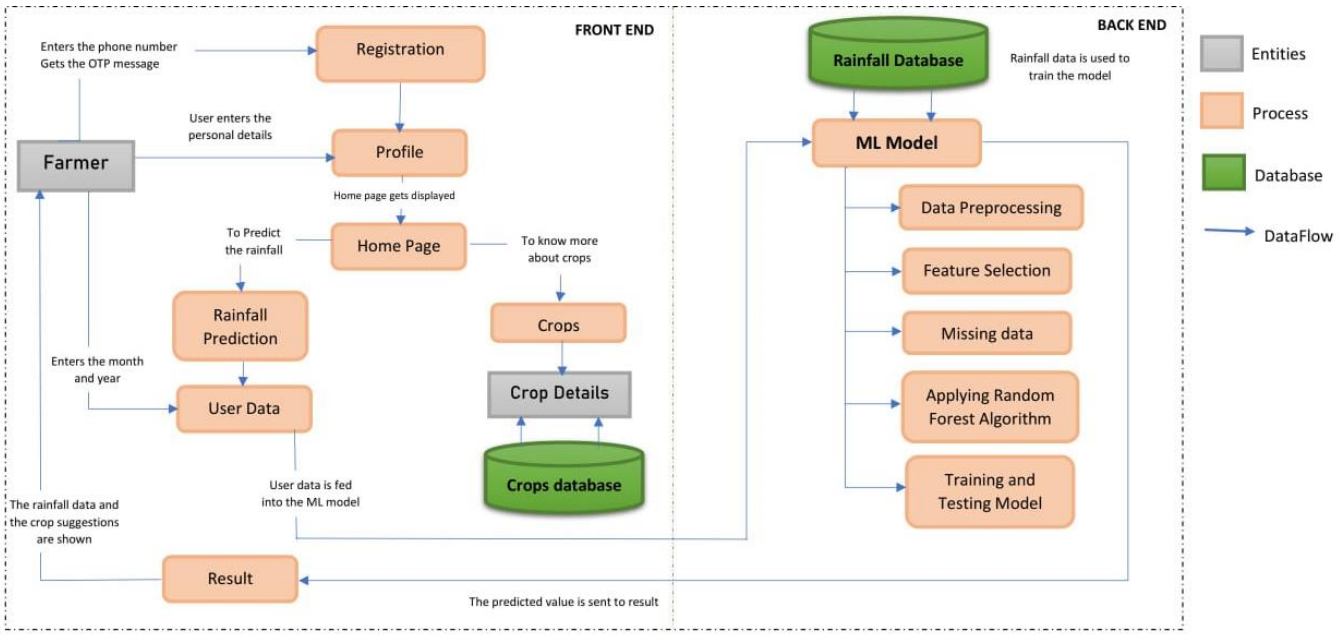
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through LinkedIn
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Prediction details	User should enter the current location to get the predicted result.
FR-4	Forecasting Accuracy	Retrieve the forecasted weather conditions and measure the accuracy.
FR-5	Forecast	Forecasted flood probability from the rainfall amount is displayed on the webpage.
FR-6	Snapshots	The web page will display the condition as a report and pictures.

4.2. Non-Functional Requirements

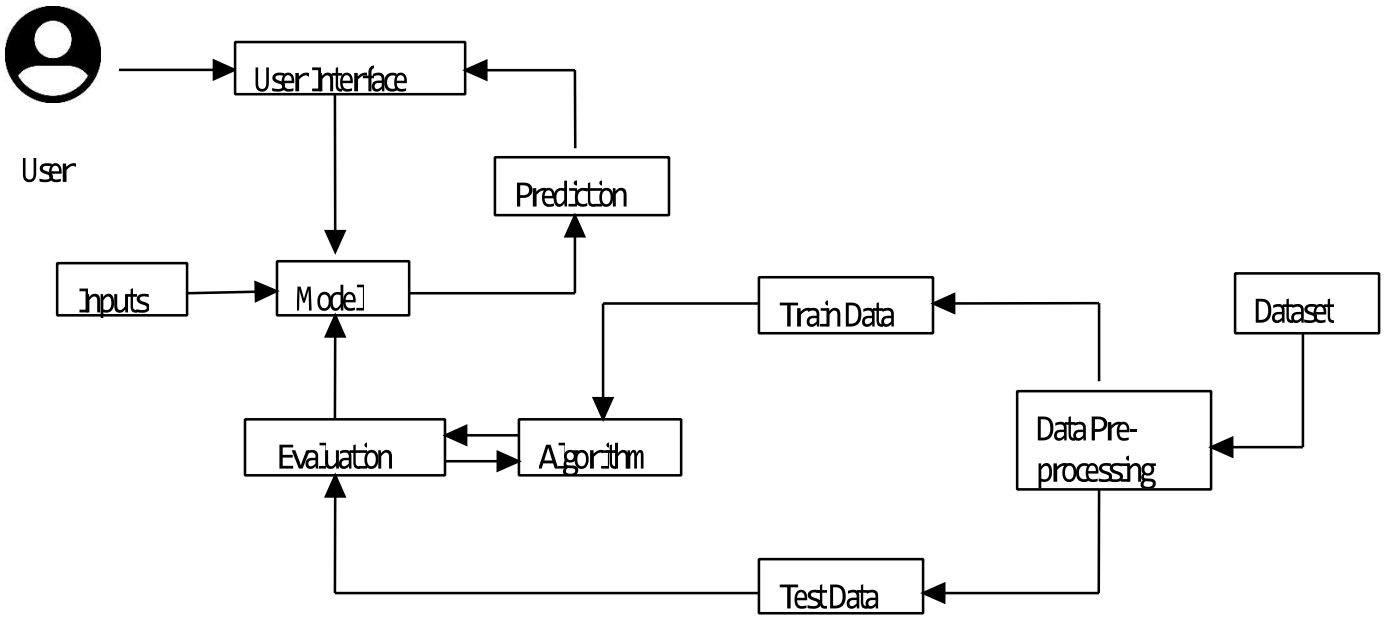
FR No.	Non-Functional Requirement	Description
NFR-1	Usability	The usability of the website is to make all users will be satisfied with our requirements of the product. The user should reach the summarized text or result with one button press if possible
NFR-2	Security	The security of the project is to develop the website that prevents SQL injection attack, XSS attack and DOS attack
NFR-3	Reliability	The reliability of the system is to make sure the website does not go offline. The users can be reach and use program at any time, so maintenance should not be big issue.
NFR-4	Performance	The performance of the website isto provide data to all users without unnecessary delay and provide 24*7 availability.
NFR-5	Availability	The availability of the website is that the website will be active on The Internet and people will be able to browse to it.
NFR-6	Scalability	The scalability of the system is we have limited our project to Indian cities We have plans to scale it to continent's level in coming updates.

5. PROJECT DESIGN

5.1. Data Flow Diagrams



5.2. Solution and Technical Architecture



5.3. User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Farmers (Mobile user)	Registration	USN-1	As a user, I can register for the application by using phone number	I can access the profile and home page	High	Sprint-1
		USN-2	As a user, I will receive OTP message, Once I have registered for the application	I can receive confirmation message	High	Sprint-1
	Profile	USN-3	As a user, I have to enter my personal details	I can access my home page	High	Sprint-1
	Home page	USN-4	As a user, I can either click on rainfall prediction or crops button	I can go to the desired page	Medium	Sprint-2
		USN-5	As a user, I can click on the "more action" button	I can view my personal details	Medium	Sprint-2
	Rainfall Prediction	USN-6	As a user, I have to enter the desired month and the year	To know the rainfall on the given month	High	Sprint-3
	Crops	USN-7	As a user, I can view the details of the crops	To know more about the crop cultivations	Medium	Sprint-3

6. PROJECT PLANNING AND SCHEDULING

6.1. Sprint Planning and Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Rainfall Prediction ML Model (Dataset)	USN-1	Weather Dataset Collection, Data pre-processing, Data Visualization.	5	High	Indira S, Saranya L, Sowmiya S , Arifa M Y, Brindha Ramesh
Sprint-1		USN-2	Train Model using Different machine learning Algorithms	5	High	Indira S, Sowmiya S , Brindha Ramesh
Sprint-1		USN-3	Test the model and give best	10	High	Indira S, Sowmiya S , Brindha Ramesh
Sprint-2	Registration	USN-4	As a user, they can register for the application through Gmail. Password is set up.	5	Medium	Saranya L,, Arifa M Y, Brindha Ramesh
Sprint-2	Login	USN-5	As a user, they can log into the application by entering email & password	5	Medium	Saranya L, Sowmiya S , Arifa M Y
Sprint-2		USN-6	Credentials should be used for multiple systems and verified	4	Medium	Indira S, Arifa M Y, Brindha Ramesh
Sprint-2	Dashboard	USN-7	Attractive dashboard forecasting live weather	6	Low	Indira S, Saranya L, Arifa M Y, Brindha Ramesh
Sprint-3	Rainfall Prediction	USN-8	User enter the location, temperature, humidity	10	High	Indira S , Sowmiya S , Arifa M Y, Brindha Ramesh
Sprint-3		USN-9	Predict the rainfall and display the result	10	High	Saranya L, Sowmiya S , Arifa M Y

6.2. Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

7. CODING AND SOLUTIONING

7.1. Feature-1: Model Building

For this feature we have made use of Jupyter notebook which uses Python programming language. To use Jupyter Notebook install Anaconda, which is a desktop graphical user interface (GUI) included in Anaconda® Distribution that allows you to launch applications and manage conda packages, environments, and channels without using command line interface (CLI) commands. Navigator can search for packages on Anaconda.org or in a local Anaconda Repository. It is available for Windows, macOS, and Linux. It provides all basic necessary python libraries which are needed for Data Analysis and Visualizations.

Below images are source code for this feature:

Data Pre-Processing

1. Importing Neccessary Libraries

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns

        from sklearn import preprocessing
        from sklearn.model_selection import train_test_split
        from sklearn.preprocessing import RobustScaler
        import scipy.stats as stats
        from collections import Counter
        from imblearn.over_sampling import SMOTE

        from sklearn import metrics
        from sklearn.ensemble import RandomForestClassifier
        from sklearn.svm import SVC
        from sklearn.linear_model import LogisticRegression
        from sklearn.naive_bayes import GaussianNB
        from sklearn.neighbors import KNeighborsClassifier
        from catboost import CatBoostClassifier
        from xgboost import XGBClassifier

        from sklearn.metrics import accuracy_score,confusion_matrix,classification_report

        import warnings
        warnings.filterwarnings("ignore", category=FutureWarning)
        warnings.filterwarnings("ignore", category=UserWarning)

        import joblib
```

In the above image, we import all necessary libraries needed for data exploration, preprocessing, model building and saving it. The below image specifies the values present in the dataset.

2. Exploratory Data Analysis

In [2]:
```
df = pd.read_csv("weatherAUS.csv")
pd.set_option("display.max_columns", None)
df
```

Out[2]:

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	WindSp
0	01-12-2008	Albury	13.4	22.9	0.6	NaN	NaN	W	44.0	W	WNW	20.0	
1	02-12-2008	Albury	7.4	25.1	0.0	NaN	NaN	WNW	44.0	NNW	WSW	4.0	
2	03-12-2008	Albury	12.9	25.7	0.0	NaN	NaN	WSW	46.0	W	WSW	19.0	
3	04-12-2008	Albury	9.2	28.0	0.0	NaN	NaN	NE	24.0	SE	E	11.0	
4	05-12-2008	Albury	17.5	32.3	1.0	NaN	NaN	W	41.0	ENE	NW	7.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
145455	21-06-2017	Uluru	2.8	23.4	0.0	NaN	NaN	E	31.0	SE	ENE	13.0	
145456	22-06-2017	Uluru	3.6	25.3	0.0	NaN	NaN	NNW	22.0	SE	N	13.0	
145457	23-06-2017	Uluru	5.4	26.9	0.0	NaN	NaN	N	37.0	SE	WNW	9.0	
145458	24-06-2017	Uluru	7.8	27.0	0.0	NaN	NaN	SE	28.0	SSE	N	13.0	
145459	25-06-2017	Uluru	14.9	NaN	0.0	NaN	NaN	NaN	NaN	ESE	ESE	17.0	

145460 rows × 23 columns

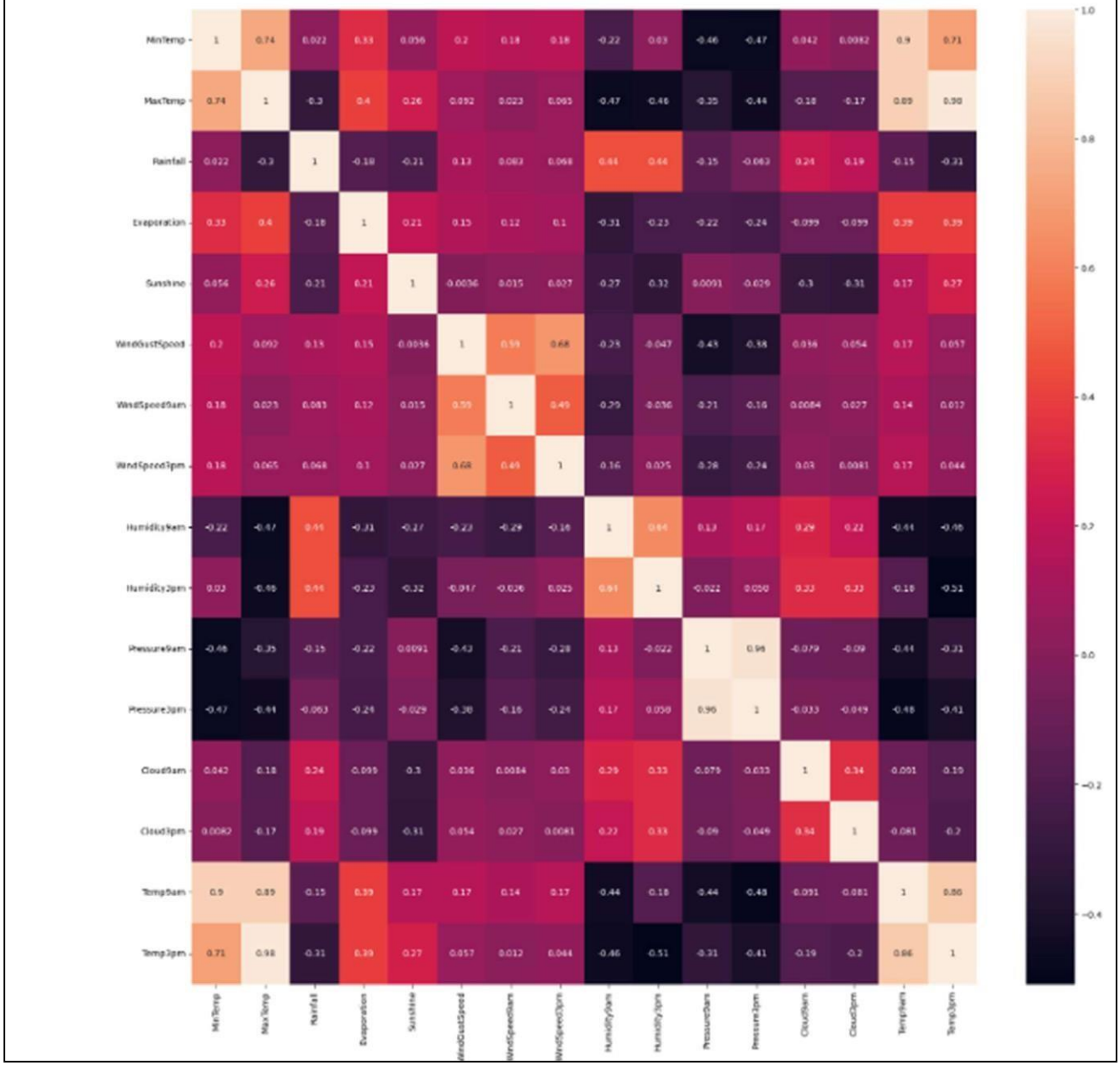
The below image specifies types of features and its count along with number of missing values in the dataset.

```
In [3]: 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 feature Count {}".format(len(discrete_feature)))
        print("Continuous feature Count {}".format(len(continuous_feature)))
        print("Categorical feature Count {}".format(len(categorical_feature)))

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

In [4]: # Handle Missing Values
        df.isnull().sum()*100/len(df)

Out[4]: Date             0.000000
        Location         0.000000
        MinTemp          1.020899
        MaxTemp          0.866905
        Rainfall         2.241853
        Evaporation     43.166506
        Sunshine        48.009762
        WindGustDir      7.098859
        WindGustSpeed    7.055548
        WindDir9am       7.263853
        WindDir3pm       2.906641
        WindSpeed9am     1.214767
        WindSpeed3pm     2.105046
        Humidity9am      1.824557
        Humidity3pm      3.098446
        Pressure9am     10.356799
        Pressure3pm     10.331363
        Cloud9am        38.421559
        Cloud3pm        40.807095
        Temp9am          1.214767
        Temp3pm          2.481094
        RainToday        2.241853
        RainTomorrow     2.245978
        dtype: float64
```

```
In [5]: print(numerical_feature)

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

In [6]: def randomsampleimputation(df, variable):
            df[variable]=df[variable]
            random_sample=df[variable].dropna().sample(df[variable].isnull().sum(),random_state=0)
            random_sample.index=df[df[variable].isnull()].index
            df.loc[df[variable].isnull(),variable]=random_sample

In [7]: randomsampleimputation(df, "Cloud9am")
        randomsampleimputation(df, "Cloud3pm")
        randomsampleimputation(df, "Evaporation")
        randomsampleimputation(df, "Sunshine")
```

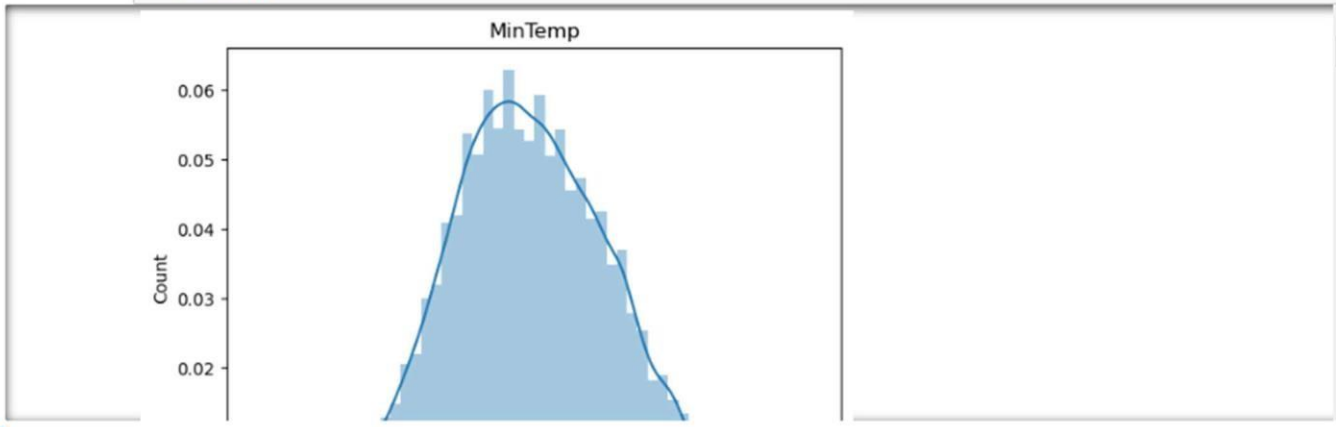
The lines 6 is used to drop rows which have high count missing values.

```
In [9]: corrmat = df.corr(method = "spearman")
        plt.figure(figsize=(20,20))
        #plot heat map
        g=sns.heatmap(corrmat,annot=True)
```

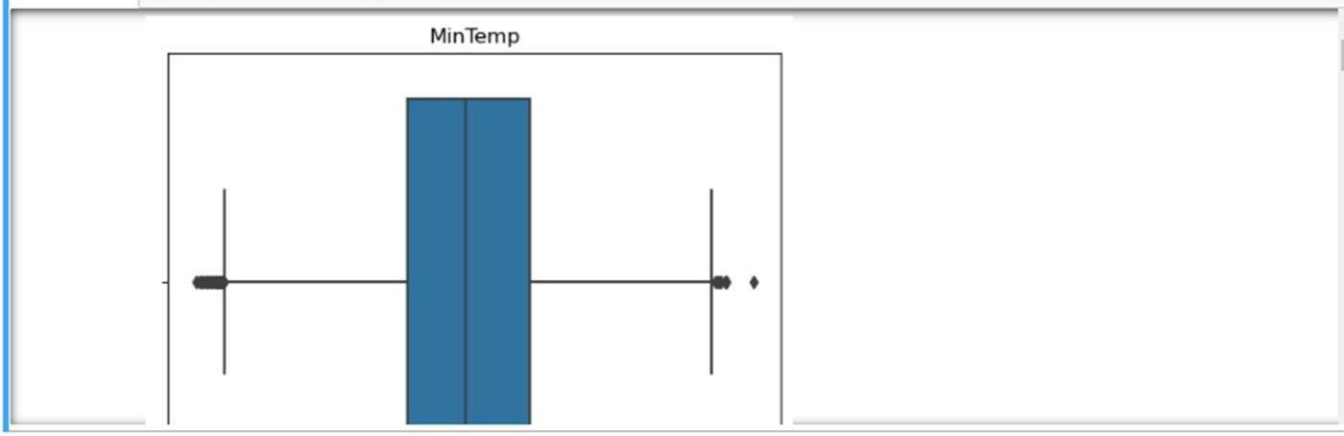


The above code displays the correlation between the columns present in the dataset.

```
In [10]: 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()
```




```
In [11]: for feature in continuous_feature:
             data=df.copy()
             sns.boxplot(data[feature])
             plt.title(feature)
             plt.figure(figsize=(15,15))
```



The above code shows the distance plot and box plot of continuous features.

```
In [12]: for feature in continuous_feature:
             if(df[feature].isnull().sum()*100/len(df))>0:
                 df[feature] = df[feature].fillna(df[feature].median())

In [13]: df.isnull().sum()*100/len(df)

Out[13]: Date             0.000000
         Location         0.000000
         MinTemp          0.000000
         MaxTemp          0.000000
         Rainfall         0.000000
         Evaporation      0.000000
         Sunshine         0.000000
         WindGustDir      7.098859
         WindGustSpeed    0.000000
         WindDir9am       7.263853
         WindDir3pm       2.906641
         WindSpeed9am     0.000000
         WindSpeed3pm     0.000000
         Humidity9am      0.000000
         Humidity3pm      0.000000
         Pressure9am      0.000000
         Pressure3pm      0.000000
         Cloud9am         0.000000
         Cloud3pm         0.000000
         Temp9am          0.000000
         Temp3pm          0.000000
         RainToday        2.241853
         RainTomorrow     2.245978
         dtype: float64
```

The above code removes null values from continuous features.

```
In [14]: discrete_feature

Out[14]: ['Cloud9am', 'Cloud3pm']

In [15]: def mode_nan(df,variable):
             mode=df[variable].value_counts().index[0]
             df[variable].fillna(mode,inplace=True)
         mode_nan(df,"Cloud9am")
         mode_nan(df,"Cloud3pm")
```

The above code removes null values by replacing it with Mode value.

```
In [16]: df["RainToday"] = pd.get_dummies(df["RainToday"], drop_first = True)
         df["RainTomorrow"] = pd.get_dummies(df["RainTomorrow"], drop_first = True)
         df
```

Out[16]:

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	WindS
0	01-12-2008	Albury	13.4	22.9	0.6	2.4	8.3	W	44.0	W	WNW	20.0	
1	02-12-2008	Albury	7.4	25.1	0.0	3.6	10.0	WNW	44.0	NNW	WSW	4.0	
2	03-12-2008	Albury	12.9	25.7	0.0	2.6	4.4	WSW	46.0	W	WSW	19.0	
3	04-12-2008	Albury	9.2	28.0	0.0	18.4	8.9	NE	24.0	SE	E	11.0	
4	05-12-2008	Albury	17.5	32.3	1.0	5.4	3.0	W	41.0	ENE	NW	7.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
145455	21-06-2017	Uluru	2.8	23.4	0.0	1.4	7.8	E	31.0	SE	ENE	13.0	
145456	22-06-2017	Uluru	3.6	25.3	0.0	7.6	13.5	NNW	22.0	SE	N	13.0	
145457	23-06-2017	Uluru	5.4	26.9	0.0	6.8	11.0	N	37.0	SE	WNW	9.0	
145458	24-06-2017	Uluru	7.8	27.0	0.0	2.6	13.2	SE	28.0	SSE	N	13.0	
145459	25-06-2017	Uluru	14.9	22.6	0.0	1.4	0.7	NaN	39.0	ESE	ESE	17.0	

145460 rows × 23 columns

The above code makes use of Label Encoding technique, which is used to convert labels into machine readable numeric values.

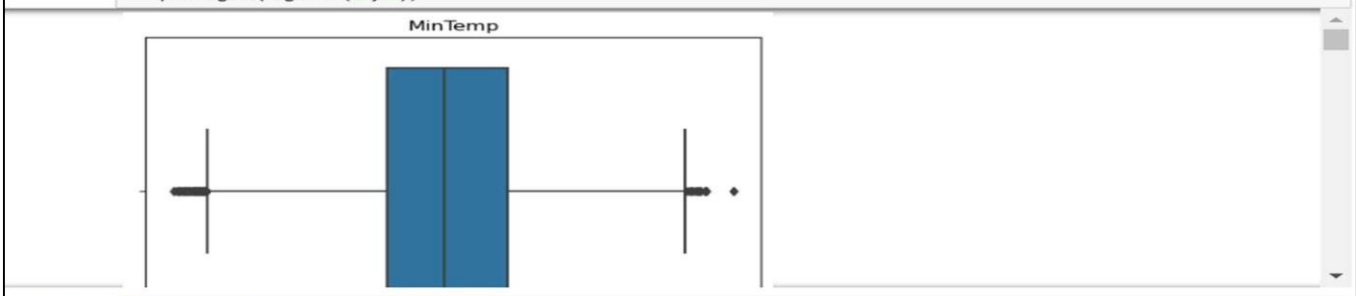
```
In [17]: for feature in categorical_feature:
             print(feature, (df.groupby([feature])["RainTomorrow"].mean().sort_values(ascending = False)).index)
```

```
Date Index(['19-12-2007', '30-01-2008', '24-12-2007', '13-04-2008', '19-06-2008',
       '02-11-2007', '03-11-2007', '20-12-2007', '03-12-2007', '21-12-2007',
       ...
       '29-04-2008', '25-04-2008', '14-01-2008', '14-02-2008', '19-08-2008',
       '29-03-2008', '29-02-2008', '08-03-2008', '19-07-2008', '01-01-2008'],
      dtype='object', name='Date', length=3436)
Location Index(['Portland', 'Walpole', 'Cairns', 'Dartmoor', 'NorfolkIsland',
       'MountGambier', 'Albany', 'Witchcliffe', 'CoffsHarbour', 'MountGinini',
       'NorahHead', 'Darwin', 'Sydney', 'SydneyAirport', 'Ballarat',
       'GoldCoast', 'Watsonia', 'Newcastle', 'Hobart', 'Wollongong',
       'Williamtown', 'Launceston', 'Brisbane', 'MelbourneAirport', 'Adelaide',
       'Sale', 'Albury', 'Perth', 'Melbourne', 'Nuriootpa', 'Penrith',
       'BadgerysCreek', 'PerthAirport', 'Tuggeranong', 'Richmond', 'Bendigo',
       'Canberra', 'WaggaWagga', 'Townsville', 'Katherine', 'PearceRAAF',
       'SalmonGums', 'Nhil', 'Moree', 'Cobar', 'Mildura', 'AliceSprings',
       'Uluru', 'Woomera'],
      dtype='object', name='Location')
WindGustDir Index(['NNW', 'NW', 'WNW', 'N', 'W', 'WSW', 'NNE', 'S', 'SSW', 'SW', 'SSE',
       'NE', 'SE', 'ESE', 'ENE', 'E'],
      dtype='object', name='WindGustDir')
WindDir9am Index(['NNW', 'N', 'NW', 'NNE', 'WNW', 'W', 'WSW', 'SW', 'SSW', 'NE', 'S',
       'SSE', 'ENE', 'SE', 'ESE', 'E'],
      dtype='object', name='WindDir9am')
WindDir3pm Index(['NW', 'NNW', 'N', 'WNW', 'W', 'NNE', 'WSW', 'SSW', 'S', 'SW', 'SE',
       'NE', 'SSE', 'ENE', 'E', 'ESE'],
      dtype='object', name='WindDir3pm')
RainToday UInt64Index([1, 0], dtype='uint64', name='RainToday')
RainTomorrow UInt64Index([1, 0], dtype='uint64', name='RainTomorrow')
```

```
In [18]: windgustdir = {'NNW':0, 'NW':1, 'WNW':2, 'N':3, 'W':4, 'WSW':5, 'NNE':6, 'S':7, 'SSW':8, 'SW':9, 'SSE':10,
             'NE':11, 'SE':12, 'ESE':13, 'ENE':14, 'E':15}
         winddir9am = {'NNW':0, 'N':1, 'NW':2, 'NNE':3, 'WNW':4, 'W':5, 'WSW':6, 'SW':7, 'SSW':8, 'NE':9, 'S':10,
             'SSE':11, 'ENE':12, 'SE':13, 'ESE':14, 'E':15}
         winddir3pm = {'NW':0, 'NNW':1, 'N':2, 'WNW':3, 'W':4, 'NNE':5, 'WSW':6, 'SSW':7, 'S':8, 'SW':9, 'SE':10,
             'NE':11, 'SSE':12, 'ENE':13, 'E':14, 'ESE':15}
         df["WindGustDir"] = df["WindGustDir"].map(windgustdir)
         df["WindDir9am"] = df["WindDir9am"].map(winddir9am)
         df["WindDir3pm"] = df["WindDir3pm"].map(winddir3pm)

In [19]: df["WindGustDir"] = df["WindGustDir"].fillna(df["WindGustDir"].value_counts().index[0])
         df["WindDir9am"] = df["WindDir9am"].fillna(df["WindDir9am"].value_counts().index[0])
         df["WindDir3pm"] = df["WindDir3pm"].fillna(df["WindDir3pm"].value_counts().index[0])
```

The above image is used to remove the remaining null values.

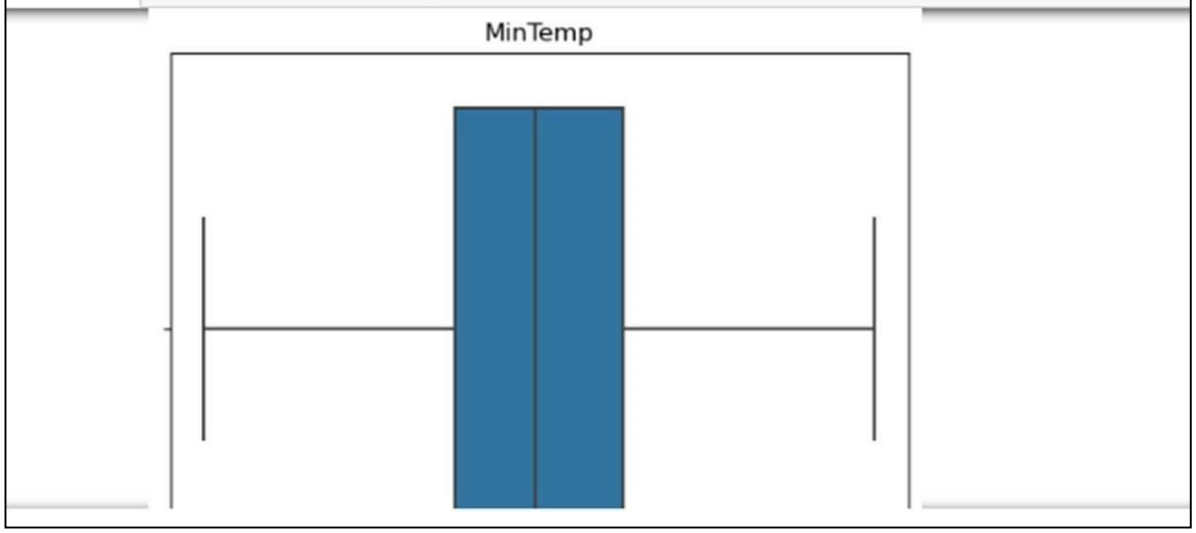
```
In [33]: for feature in continuous_feature:
             data=df.copy()
             sns.boxplot(data[feature])
             plt.title(feature)
             plt.figure(figsize=(15,15))
```




```
In [34]: for feature in continuous_feature:
             print(feature)

MinTemp
MaxTemp
Rainfall
Evaporation
Sunshine
WindGustSpeed
WindSpeed9am
WindSpeed3pm
Humidity9am
Humidity3pm
Pressure9am
Pressure3pm
Temp9am
Temp3pm
```

The above image is used to find values which lies outside the Inter-Quartile Range of each continuous feature. After finding the lower and higher bound, we remove the outliers from each continuous feature.

```
In [59]: for feature in continuous_feature:
             data=df.copy()
             sns.boxplot(data[feature])
             plt.title(feature)
             plt.figure(figsize=(15,15))
```



The above image shows the boxplot of each continuous feature after removing the outliers.

3. Splitting Dataset into Independent and Dependent Variables

```
In [64]: X = df.drop(["RainTomorrow", "Date", "Date_month", "Date_day"], axis=1)
         Y = df["RainTomorrow"]
```

4. Feature Scaling

```
In [65]: scaler = RobustScaler()
         X_scaled = scaler.fit_transform(X)
```

We split the dataset into independent and dependent variables. Here we must predict 'RainTomorrow', hence it will be the dependent variable and Date columns are unnecessary columns hence we drop it. And all other columns are independent variables. Using RobustScaler, we perform feature scaling to normalize the independent variables such that the standard distribution results to zero and standard deviation to one. This also removes remaining outliers in the independent variables.

Now using 'train_test_split', we split the variables into train and test variables for each variable .

5. Splitting The Data Into Train And Test

```
In [66]: X_train, X_test, y_train, y_test = train_test_split(X_scaled,Y, test_size =0.2, stratify = Y, random_state = 0)

In [67]: X_train.shape
         X_test.shape

Out[67]: (29092, 21)

In [68]: y_train.shape
         y_test.shape

Out[68]: (29092,)
```

6. Balancing the Data

```
In [69]: 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: 90866, 1: 25502})
         The number of classes after fit Counter({0: 90866, 1: 90866})
```

SMOTE(Synthetic Minority Oversampling Technique)is used to increase the number of test cases in a balanced way to avoid overfit cases

Model Building

```
In [70]: cat = CatBoostClassifier(iterations=2000, eval_metric = "AUC")
         cat.fit(X_train_res, y_train_res)

         Learning rate set to 0.050311
         0:      total: 470ms    remaining: 15m 38s
         1:      total: 513ms    remaining: 8m 32s
         2:      total: 555ms    remaining: 6m 9s
         3:      total: 597ms    remaining: 4m 57s
         4:      total: 634ms    remaining: 4m 12s
         5:      total: 670ms    remaining: 3m 42s
         6:      total: 707ms    remaining: 3m 21s
         7:      total: 750ms    remaining: 3m 6s
         8:      total: 790ms    remaining: 2m 54s
         9:      total: 829ms    remaining: 2m 44s
         10:     total: 869ms    remaining: 2m 37s
         11:     total: 910ms    remaining: 2m 30s
         12:     total: 951ms    remaining: 2m 25s
         13:     total: 990ms    remaining: 2m 20s
         14:     total: 1.03s    remaining: 2m 16s
         15:     total: 1.07s    remaining: 2m 12s
         16:     total: 1.11s    remaining: 2m 9s
         17:     total: 1.15s    remaining: 2m 6s
```

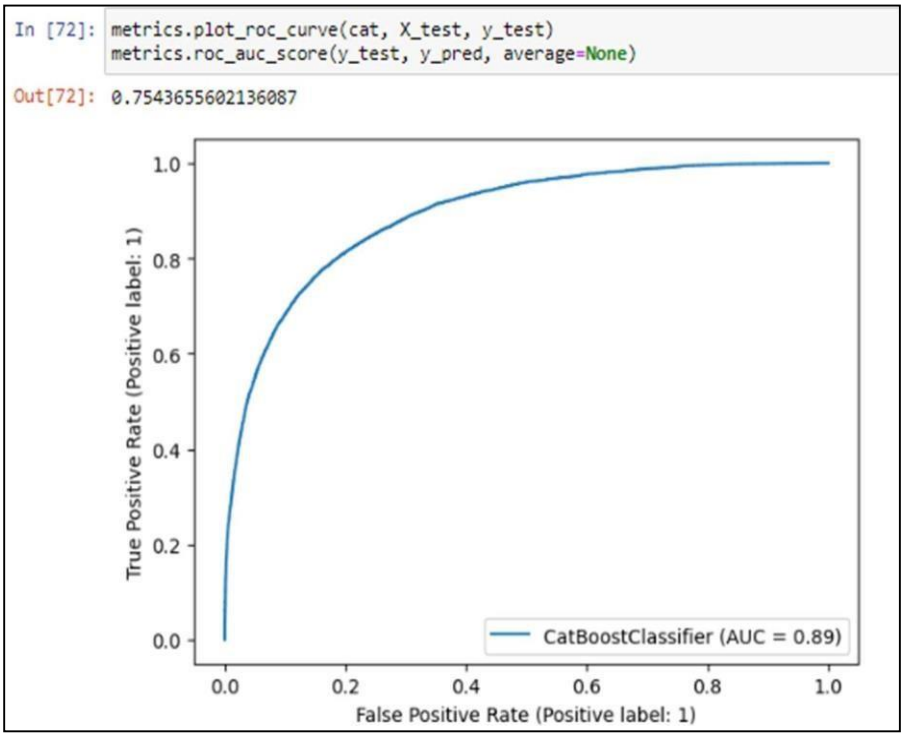
.

The algorithm chosen here to build the model is CatBoostClassifier. CatBoost is based o n gradient boosted decision trees. During training, a set of decision trees is built consecutively. Each successive tree is built with reduced loss compared to the previous trees. The number of trees is controlled by the starting parameters.

```
In [71]: 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))

         [[21506  1211]
          [ 2792  3583]]
         0.8624020349236904
                       precision    recall  f1-score   support

                    0       0.89      0.95      0.91     22717
                    1       0.75      0.56      0.64      6375

             accuracy                           0.86     29092
            macro avg       0.82      0.75      0.78     29092
         weighted avg       0.85      0.86      0.85     29092
```

The above image shows the Confusion Matrix, Accuracy Score and Classification report.

```
In [72]: metrics.plot_roc_curve(cat, X_test, y_test)
         metrics.roc_auc_score(y_test, y_pred, average=None)

Out[72]: 0.7543655602136087
```



The above image shows the roc curve and roc accuracy score for the built model.

Hyperparameter Tuning

```
In [74]: from sklearn.model_selection import RandomizedSearchCV
         from scipy.stats import randint
         param_dist = { "learning_rate": np.linspace(0,0.2,5),"max_depth": randint(3, 10)}
         rscv = RandomizedSearchCV( CatBoostClassifier(), param_dist, scoring='accuracy', cv = 5)
         rscv.fit(X_train_res, y_train_res)
         print(rscv.best_params_)
         print(rscv.best_score_)
```

```
983:    learn: 0.1411624        total: 54.3s    remaining: 883ms
984:    learn: 0.1410823        total: 54.3s    remaining: 828ms
985:    learn: 0.1410310        total: 54.4s    remaining: 772ms
986:    learn: 0.1409701        total: 54.5s    remaining: 717ms
987:    learn: 0.1409060        total: 54.5s    remaining: 662ms
988:    learn: 0.1408196        total: 54.6s    remaining: 607ms
989:    learn: 0.1407667        total: 54.6s    remaining: 552ms
990:    learn: 0.1406785        total: 54.7s    remaining: 497ms
991:    learn: 0.1406161        total: 54.8s    remaining: 442ms
992:    learn: 0.1405794        total: 54.8s    remaining: 386ms
993:    learn: 0.1405091        total: 54.9s    remaining: 331ms
994:    learn: 0.1404368        total: 54.9s    remaining: 276ms
995:    learn: 0.1403839        total: 55s      remaining: 221ms
996:    learn: 0.1402899        total: 55.1s    remaining: 166ms
997:    learn: 0.1402249        total: 55.1s    remaining: 110ms
998:    learn: 0.1401474        total: 55.2s    remaining: 55.2ms
999:    learn: 0.1400710        total: 55.2s    remaining: 0us
{'learning_rate': 0.1, 'max_depth': 8}
0.8892227301457538
```

Cross Validation

```
In [73]: from sklearn.model_selection import cross_val_score
         accuracies = cross_val_score(estimator = CatBoostClassifier(), X = X_train_res, y = y_train_res, cv = 3)
         print("Accuracy:{:.2f} %".format(accuracies.mean()*100))
         print("Standard Deviation:{:.2f} %".format(accuracies.std()*100))
```

```
983:    learn: 0.2312273        total: 25.2s    remaining: 409ms
984:    learn: 0.2311698        total: 25.2s    remaining: 384ms
985:    learn: 0.2311267        total: 25.2s    remaining: 358ms
986:    learn: 0.2310880        total: 25.2s    remaining: 333ms
987:    learn: 0.2310416        total: 25.3s    remaining: 307ms
988:    learn: 0.2310012        total: 25.3s    remaining: 281ms
989:    learn: 0.2309517        total: 25.3s    remaining: 256ms
990:    learn: 0.2309123        total: 25.3s    remaining: 230ms
991:    learn: 0.2308675        total: 25.4s    remaining: 205ms
992:    learn: 0.2308233        total: 25.4s    remaining: 179ms
993:    learn: 0.2307680        total: 25.4s    remaining: 153ms
994:    learn: 0.2307091        total: 25.4s    remaining: 128ms
995:    learn: 0.2306458        total: 25.5s    remaining: 102ms
996:    learn: 0.2306044        total: 25.5s    remaining: 76.7ms
997:    learn: 0.2305532        total: 25.5s    remaining: 51.2ms
998:    learn: 0.2304996        total: 25.6s    remaining: 25.6ms
999:    learn: 0.2304346        total: 25.6s    remaining: 0us
Accuracy:83.11 %
Standard Deviation:17.73 %
```

The above image shows the Hyperparameter and Cross Validation score of the mode

Saving the built Models

```
In [76]: joblib.dump(rscv, "cat2.pkl")
Out[76]: ['cat2.pkl']
```

Finally save the model using joblib library.

7.2.Feature-2:

User Interface

Index.html:

```
<!DOCTYPE html>
<html>
 <head>
 <meta charset="UTF-8">
 <title>Rainfall prediction Webpage</title>
 <link rel="stylesheet" href="{{ url_for('static',
filename='css/index.css') }}">
 <style>
 .head1{

 color: rgb(10, 73, 112);
 font-family: 'Century Gothic';
 font-size: 60px;
 width: 40%;
 margin-left: 30%;
 text-align: center;
 margin-top: 2vh;
 padding: 5px;
 }
```

```
 </style>
 </head>

<body>
 <div class="login">
 <div>

<div class="head1">
 RAINFALL PREDICTION
 </div>

 <form action="{{ url_for('predict')}}"method="post">
 <br>
 <br>
 <div class="body1">
 <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>Norfolksland</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>MountGinini</option>
 <option value=5>Ballarat</option>
 <option value=6>Bandigo</option>
 <option value=35>Sale</option>
 <option value=19>MelborneAirport</option>
 <option value=18>Melborne</option>
 <option value=20>Mildura</option>

 <option value=25>Nhill</option>
 <option value=33>Portland</option>
 <option value=44>Watsonia</option>
 </select>
 <label for="MinTemp">MinTemp:</label>
 <input type = "number" step="any" id="MinTemp"
name="MinTemp" value="MinTemp">
```

```
 <label for="MaxTemp">MaxTemp:</label>
 <input type = "number" step="any" id="MaxTemp"
name="MaxTemp" value="MaxTemp">
 <label for="Rainfall">Rainfall:</label>
 <input type = "number" step="any" id="Rainfall"
name="Rainfall" value="Rainfall">
 <label for="WindGustSpeed">WindGustSpeed:</label>
 <input type = "number" step="any" id="WindGustSpeed"
name="WindGustSpeed" value="WindGustSpeed">
<br>
<br>
 <label for="WindSpeed9am">WindSpeed9am:</label>
 <input type = "number" step="any" id="WindSpeed9am"
name="WindSpeed9am" val="WindSpeed9am">
 <label for="WindSpeed3pm">WindSpeed3pm:</label>
 <input type = "number" step="any" id="WindSpeed3pm"
name="WindSpeed3pm" val="WindSpeed3pm">
 <label for="Humidity9am">Humidity9am:</label>
 <input type = "number" step="any" id="Humidity9am"
name="Humidity9am" val="Humidity9am">
 <label for="Humidity3pm">Humidity3pm:</label>
 <input type = "number" step="any" id="Humidity3pm"
name="Humidity3pm" val="Humidity3pm">
<br>
```

```
<br>
 <label for="Pressure9am">Pressure9am:</label>
 <input type = "number" step="any" id="Pressure9am"
name="Pressure9am" val="Pressure9am">
 <label for="Pressure3pm">Pressure3pm:</label>
 <input type = "number" step="any" id="Pressure3pm"
name="Pressure3pm" val="Pressure3pm">
 <label for="Temp9am">Temp9am:</label>
 <input type = "number" step="any" id="Temp9am"
name="Temp9am" val="Temp9am">
 <label for="Temp3pm">Temp3pm:</label>
 <input type = "number" step="any" id="Temp3pm"
name="Temp3pm" val="Temp3pm">
<br>
<br>
<label for="RainToday">RainToday</label>
<select id="RainToday" name="RainToday">
 <option value="1">Yes</option>
 <option value="2">No</option>
</select>
<label for="WindGustDir">WindGustDir</label>
 <select id="WindGustDir" name="WindGustDir">
 <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=5>NA</option>
<option value=4>SSW</option>
</select>   
<label for="WindDir9am">WindDir9am</label>
<select id="WindDir9am" name="WindDir9am">
<option value=14>W</option>
<option value=15>NNW</option>
<option value=0>SE</option>
<option value=7>ENE</option>
<option value=13>SW</option>
<option value=10>SSE</option>
```

```
 <option value=2>S</option>
 <option value=1>NE</option>
 <option value=6>NA</option>
 <option value=11>SSW</option>
 <option value=12>N</option>
 <option value=9>WSW</option>
 <option value=3>ESE</option>
 <option value=8>E</option>
 <option value=5>NW</option>
 <option value=4>WNW</option>
 <option value=4>NNE</option>
 </select>   
<label for="WindDir3pm">WindDir3pm</label>
 <select id="WindDir3pm" name="WindDir3pm">

 <option value=14>WNW</option>
 <option value=15>WSW</option>
 <option value=0>E</option>
 <option value=7>NW</option>
 <option value=13>W</option>
 <option value=10>SSE</option>
 <option value=2>ESE</option>
 <option value=1>ENE</option>
 <option value=6>NNW</option>
```

```
 <option value=11>SSW</option>
 <option value=12>SW</option>
 <option value=9>SE</option>
 <option value=3>N</option>
 <option value=8>S</option>
 <option value=5>NNE</option>
 <option value=5>NA</option>
 <option value=4>NE</option>
 </select>   
 </div>
<br>
<br><br>
<center><button class="button-3"
role="button">PREDICT</button></center>
</form>
<br>
 {{output}}
 <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>
 <footer><p>IBM - Nalaiya Thiran</p></footer>
</html>
```

INDEX.CSS

```
body{
 background-image: url('background.jpg');
 background-size:cover;
 background-repeat: no-repeat;
 }
 footer {
 display: flex;
 justify-content: center;
 padding: 5px;
 margin-top: -40px;
 background-color:rgb(10, 73, 112);
```

```
 color: #fff;
 }
 .body1{
 background-color:rgb(255, 255, 255,0.8);
 background-origin: padding-box;
 color: #3d3d3d;
 font-size: 20px;
 width: 90%;
 height: 280px;
 margin-left:6%;
 margin-top: 6vh;
 text-align: center;
 padding: 5px;
 background-blend-mode: lighten;

 }
/* CSS */
.button-3 {
 appearance: none;
 background-color: rgb(10, 73, 112);
 border: 1px solid rgba(27, 31, 35, .15);
 border-radius: 6px;
 box-shadow: rgba(27, 31, 35, .1) 0 1px 0;
 box-sizing: border-box;
```

```
 color: #fff;
 cursor: pointer;
 display: inline-block;
 font-family: -apple-system,system-ui,"Segoe
UI",Helvetica,Arial,sans-serif,"Apple Color Emoji","Segoe UI
Emoji";
 font-size: 14px;
 font-weight: 600;
 line-height: 20px;
 padding: 6px 16px;
 position: relative;
 text-align: center;
 text-decoration: none;
 user-select: none;
 -webkit-user-select: none;

touch-action: manipulation;
 vertical-align: middle;
 white-space: nowrap;
}
.button-3:focus:not(:focus-visible):not(.focus-visible) {
 box-shadow: none;
 outline: none;
}
```

```
.button-3:hover {
 background-color: rgb(119, 214, 224);
}
.button-3:focus {
 box-shadow: rgba(46, 164, 79, .4) 0 0 0 3px;
 outline: none;
}
.button-3:disabled {
 background-color: #94d3a2;
 border-color: rgba(27, 31, 35, .1);
 color: rgba(255, 255, 255, .8);
 cursor: default;
}
.button-3:active {
 background-color: rgb(10, 73, 112);
 box-shadow: rgba(20, 70, 32, .2) 0 1px 0 inset;
}
```

HOME.HTML

```
<!DOCTYPE html>
<html>
 <head>

 <meta charset="utf-8">
 <title>Rainfall Prediction Webpage</title>
```

```
 <link rel="stylesheet" href="{{ url_for('static',
filename='css/home.css') }}">

<link rel="preconnect"
href="https://fonts.googleapis.com">
<link rel="preconnect"
href="https://fonts.gstatic.com" crossorigin>
<link
href="https://fonts.googleapis.com/css2?family=Raleway&family=Robot
o
:wght@100;500&display=swap" rel="stylesheet">
<style type="text/css"></style>
 </head>
<body>
<header>
 <div class="header1">

<img src="{{url_for('static', filename='css/logo.png')}}"
alt="logo"
/>
 </div>
 <div class="navbar">
<ul>
```

```
<div class="nav">
<a href="">HOME</a></div>
<div class="nav">
<a href="{{ url_for('predict') }}">PREDICTOR</a>
</div>
<!-- <div class="nav">
<a href="{{ url_for('about') }}">ABOUT</a></div> -->
<!-- <div class="nav">
<a href="{{ url_for('help') }}">HELP</a></div> -->
<!-- <div class="nav">
<a href="{{ url_for('contact') }}">CONTACT</a></div> -->
</ul>

</div>
</header>
 <div>
 <div class="head1">
 Forecast Rainfall
 </div>
 <div class="body1">
We serve as an early warning system to exactly determine the
rainfall for effective use of water resources, crop productivity,
and pre-planning of water structures.
</div>
```

```
</div>
 </body>
 <footer><p>IBM - Nalaiya Thiran</p></footer>
 </html>
```

HOME.CSS

```
body{
 background-image: url('drops.jpg');
 background-size:cover;
 }
 #title{
 margin-right: 11vw;
 width: 5vw;
 }

 .nav{
 margin:0 auto;
 float: left;
 list-style-type:none;
 width: 11vw ;
 height: 2vw;
 background-color: rgb(10, 73, 112,0.7);
 color: white;
 padding: 0.5%;
```

```
}
header{
display: block ;
width: 100%;
height: 7vh;

}
a{
text-decoration: none;
}
a:link{
color: inherit;
}
a:link:hover{
color: inherit;
}
a:visited{
color: inherit;
}
a:visited:hover{

color: inherit;
}
.nav:hover,.nav:active{
```

```
 background-color: #d3d3d3;

 color: black;
 }

.head1{

 color: rgb(10, 73, 112);
 font-family: 'Century Gothic';
 font-size: 60px;

 width: 40%;
 margin-left: 30%;
 text-align: center;
 margin-top: 2vh;
 padding: 5px;
 }
 .body1{
 background-color:rgb(255, 255, 255,0.8);
 background-origin: padding-box;
 color: #3d3d3d;
 font-size: x-large;
 width: 75%;
 margin-left:12.5%;
```

```
 margin-top: 10vh;
 text-align: center;
 padding: 35px;
 background-blend-mode: lighten;

 }
 .navbar{
 margin-left: 20%;
 }


 *{
 font-family: 'Raleway', sans-serif;
 /*font-family: 'Roboto', sans-serif;*/
 }


footer {
 display: flex;
 justify-content: center;
 padding: 5px;
 margin-top: 43vh;
 background-color:rgb(10, 73, 112);
 color: #fff;
```

```
}
 .header1 img {
 float: left;
 width: 100px;
 height: 100px;
 background: #555;
 margin-left: 5%;
 }
```

CHANCE.HTML

```
<!DOCTYPE html>
<html>
<head>
 <title>Rainfall Prediction</title>
<style>
 body {
 background: linear-gradient(45deg, rgb(169, 250, 247), rgb(216,
77, 255)) fixed;
}
div.gallery {
 border: 1px solid rgb(0, 0, 0);
}
```

```
div.gallery:hover {

border: 1px solid #777;
}
div.gallery img {
 width: 366px;
 height: 280px;
}
div.desc {
 padding: 15px;
 text-align: center;
}
* {
 box-sizing: border-box;
}
.responsive {
 margin-top: 15vh;
 padding: 0 6px;
 float: left;
 width: 24.99999%;
}
@media only screen and (max-width: 700px) {
 .responsive {
 width: 49.99999%;
```

```
 margin: 6px 0;
 }
}
@media only screen and (max-width: 500px) {
 .responsive {
 width: 100%;
 }
}
.clearfix:after {
 content: "";
 display: table;

clear: both;
}
.icon img {
 float: left;
 width: 100px;
 height: 100px;
 background: #555;
 margin-left: 5%;
 }
</style>
</head>
<body>
```

```
 <div class="icon">
 <img src="{{url_for('static',
filename='css/raincloud.png')}}" alt="logo" />
 </div>
<center><h1>Chances of rain today</h1>

<br>
<h2>You can follow any of the following process for effective
agriculture!</h2></center>
<div class="responsive">
 <div class="gallery">
 <a target="_blank"
href="https://affordabletreeservice.com.au/shared/content/uploads/La
nd-clearing-hero-1-1024x550.jpg">
 <img
src="https://affordabletreeservice.com.au/shared/content/uploads/Lan
d-clearing-hero-1-1024x550.jpg" alt="Cinque Terre" width="600"
height="400">
 </a>
 <div class="desc">Land Clearing</div>

</div>
```

```
</div>

<div class="responsive">

 <div class="gallery">

 <a target="_blank"

href="https://www.britannica.com/explore/savingearth/wpcontent/uplo
ads/sites/4/2019/04/0000156670-1024x681.jpg ">

 <img
src="https://www.britannica.com/explore/savingearth/wpcontent/uploa
ds/sites/4/2019/04/0000156670-1024x681.jpg"

alt="Forest" width="600" height="400">

 </a>

 <div class="desc">Ensure proper drainage</div>

 </div>

</div>

<div class="responsive">

<div class="gallery">

 <a target="_blank"
href="https://4.imimg.com/data4/VQ/KS/MY3597886/img_0112-
500x500.jpg">

 <img src="https://4.imimg.com/data4/VQ/KS/MY-3597886/img_0112-

500x500.jpg" alt="Northern Lights" width="600" height="400">

 </a>

 <div class="desc">Set up a rain cover</div>

 </div>

</div>

<div class="responsive">
```

```
 <div class="gallery">
 <a target="_blank"
href="https://soilsmatter.files.wordpress.com/2015/02/sensor-
basednitrogen-application-to-corn-maize-in-the-midwest-
photoraun.jpg?w=1200">
 <img
src="https://soilsmatter.files.wordpress.com/2015/02/sensor-
basednitrogen-application-to-corn-maize-in-the-midwest-
photoraun.jpg?w=1200" alt="Mountains" width="600" height="400">
 </a>
 <div class="desc">Pre-plant Fertilizer</div>


</div>
</div>
</body>
</html>
```

NOCHANCE.HTML

```
<!DOCTYPE html>
<html>
<head>
 <title>Rainfall Prediction</title>
<style>
 body {
 background: linear-gradient(45deg, rgb(250, 250, 169), rgb(255,
202, 77)) fixed;
```

```
}
div.gallery {
 border: 1px solid rgb(0, 0, 0);
}
div.gallery:hover {
 border: 1px solid #777;
}
div.gallery img {
 width: 366px;
 height: 280px;
}
div.desc {
padding: 15px;
 text-align: center;
}
* {
 box-sizing: border-box;
}
.responsive {
 margin-top: 15vh;
 padding: 0 6px;
 float: left;
 width: 24.99999%;
}
```

```
@media only screen and (max-width: 700px) {
 .responsive {
 width: 49.99999%;
 margin: 6px 0;
 }
}
@media only screen and (max-width: 500px) {
 .responsive {
 width: 100%;
 }
}
.clearfix:after {
 content: "";
 display: table;
 clear: both;
}
.icon img {
 float: left;

 width: 100px;
 height: 100px;
 margin-left: 5%;
 }
</style>
```

```
</head>
<body>
 <div class="icon">
 <img src="{{url_for('static', filename='css/sun.png')}}"
alt="logo" />
 </div>
<center><h1>No chances of rain today</h1>
<br>
<h2>You can follow any of the following process for effective
agriculture!</h2></center>
<div class="responsive">
 <div class="gallery">
 <a target="_blank"
href="https://westafricanagri.com/wpcontent/uploads/2020/06/Irrirga
ting.jpg">
 <img
src="https://westafricanagri.com/wpcontent/uploads/2020/06/Irrirgat
ing.jpg" alt="Cinque Terre"
width="600" height="400">
 </a>

<div class="desc">Calculate the water needed for Irrigation</div>
 </div>
</div>
<div class="responsive">
```

```
 <div class="gallery">


<a target="_blank"

href="https://media.istockphoto.com/id/1126541751/photo/handsplanting-the-seeds-into-
thedirt.jpg?s=612x612&w=0&k=20&c=aVUr7F_H4ZSJX89Nmiw59F8WvneKegYsBoOiDQw0SA= ">

 <img

src="https://media.istockphoto.com/id/1126541751/photo/handsplanting-the-seeds-into-
thedirt.jpg?s=612x612&w=0&k=20&c=aVUr7F_H4ZSJX89Nmiw59F8WvneKegYsBoOiDQw0SA=" alt="Forest" width="600" height="400">

 </a>

 <div class="desc">Sowing</div>

 </div>

</div>

<div class="responsive">

 <div class="gallery">

 <a target="_blank"

href="https://media.istockphoto.com/id/1151784210/photo/ripe-
ricefield-and-sky-background-
atsunset.jpg?s=612x612&w=0&k=20&c=DZz4wxIbPXnMhmoTsEV06uYKup9MEZTtRFe2

XkDb0mY=">

 <img

src="https://media.istockphoto.com/id/1151784210/photo/ripe-
ricefield-and-sky-background-
```

```
atsunset.jpg?s=612x612&w=0&k=20&c=DZz4wxIbPXnMhmoTsEV06uYKup9MEZTtR
Fe2

XkDb0mY=" alt="Northern Lights" width="600" height="400">

 </a>


<div class="desc">Sundrying / Threshing of crops</div>

 </div>

</div>

<div class="responsive">

 <div class="gallery">

 <a target="_blank"

href="https://static.vecteezy.com/system/resources/previews/010/508
/

297/large_2x/old-farmers-spray-fertilizer-or-chemical-pesticides-
inthe-rice-fields-chemical-fertilizers-free-photo.jpg">


<img

src="https://static.vecteezy.com/system/resources/previews/010/508/
2

97/large_2x/old-farmers-spray-fertilizer-or-chemical-pesticides-
inthe-rice-fields-chemical-fertilizers-free-photo.jpg"
alt="Mountains"

width="600" height="400">


</a>

 <div class="desc">Apply Agricultural chemicals</div>
```

```
 </div>
</div>
</body>
</html>
```

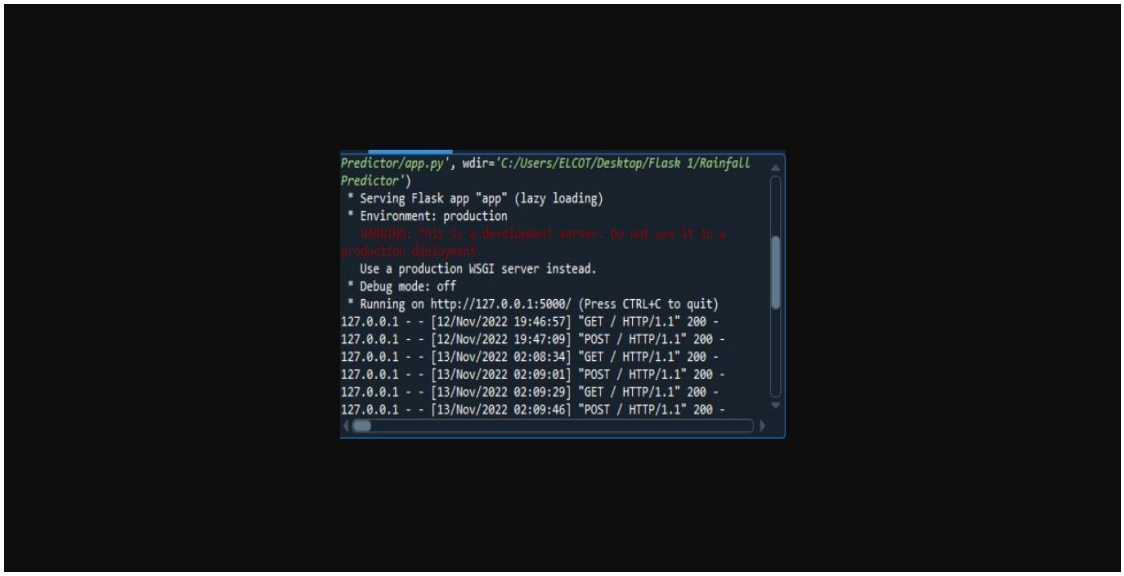
APP.PY

```
#from crypt import methods
from pyexpat import features, model
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,
redirect, url_for
# Declare a Flask app
app = Flask(__name__,template_folder='template')
model = pickle.load(open("rainfall.pkl",'rb'))
scale = pickle.load(open("scale.pkl",'rb'))
@app.route('/')
def home():
 return render_template("home.html")
```

```
@app.route('/chance/',methods=['GET', 'POST'])
def chance():
 return render_template("chance.html")
@app.route('/nochance/',methods=['GET', 'POST'])
def nochance():
 return render_template("noChance.html")
@app.route('/help/')
def help():
 return render_template("help.html")
@app.route('/contact/')
def contact():

return render_template("contact.html")
@app.route('/about/')
def about():
 return render_template("about.html")
@app.route('/predict',methods=["POST","GET"])
def predict():
 res = " "

# If a form is submitted
 if request.method == "POST":
 input_feature=[x for x in request.form.values() ]
```

```
features_values=[np.array(input_feature)]
 names =
[['Location','MinTemp','MaxTemp','Rainfall','WindGustSpeed',
 'WindSpeed9am','WindSpeed3pm','Humidity9am','Humadity3pm',

'Pressure9pm','Pressure3am','Temp9pm','Temp3pm','RainyTodaty',
 'WindGustDir','WindDir9pm','WindDir3pm']]
 data = pandas.DataFrame(features_values,columns=names)
 data = scale.fit_transform(data)
 data = pandas.DataFrame(data,columns=names)
 #Get prediction
 prediction = model.predict(data)
 else:
 prediction = ""
 if prediction == 1:
 return redirect(url_for('chance'))
 elif prediction == 0:
 return redirect(url_for('nochance'))
 return render_template("index.html", output = res)
#Running the app
if __name__== "___main___":
 app.run(debug = True,host='0.0.0.0',port=80)
```

RUN THE APP


Predictor/app.py', wdir='C:/Users/ELCOT/Desktop/Flask 1/Rainfall Predictor')
* Serving Flask app "app" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [12/Nov/2022 19:46:57] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [12/Nov/2022 19:47:09] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [13/Nov/2022 02:08:34] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [13/Nov/2022 02:09:01] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [13/Nov/2022 02:09:29] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [13/Nov/2022 02:09:46] "POST / HTTP/1.1" 200 -

8.TESTING

7.3.Test Cases

Test case ID	Feature Type	Component	Test Scenario	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Executed By
LoginPage_TC_001	UI	Home Page	Verify user is login by entering email,password,and confirming password.	1.Enter URL and click go 2.Enter the email id, password and confirm password. 3.click the login button.	https://rainfalldata.w3spaces.com	Login/ registering for the application	Working as expected	Pass	Mathusudhan
LoginPage_TC_002	UI	Home Page	Verify the can access the dashboard with the LinkedIn login.	1. Enter the URL and click enter 2.enter the valid mail id in the Email text box. 3.enter the valid password in the password text box. 4.click on tthe join now button in linked in.	https://rainfalldata.w3spaces.com/	Application should show below UI elements: a.email text box b.password text box c.join now button d.shows the dashboard page	Working as expected	pass	Vishnudev
LoginPage_TC_003	Functional	Home page	Verify user is able to log into application with Valid credentials and get the confirmation mail.	1.Enter URL and click go 2.Click on My Account dropdown button 3.Enter Valid username/email in Email text box 4.Enter valid password in password text box 5.Click on login and get mail.	Username: ibmmsec@gamil.com password: Testing123	Application should send the confirmation mail	Working as expected	Pass	Mohammedasath
Test case ID	**Feature Type**	**Component**	**Test Scenario**	**Steps To Execute**	**Test Data**	**Expected Result**	**Actual Result**	**Status**	**Executed By**
LoginPage_TC_004	Functional	Login page	Verify user is able to log into application with Valid credentials	1.Enter URL(https://shopenzer.com/) and click go 2.Click on My Account dropdown button 3.Enter Valid username/email in Email text box 4.Enter valid password in password text box 5.Click on login button	Username: ibmmsec@gmail.com password: Testing123	User should navigate to tne home page.	Working as expected	Pass	Mohamed Abhuthahir Khan
LoginPage_TC_005	Functional	Login page	Verify user is able to log into application with InValid credentials	1.Enter URL(https://shopenzer.com/) and click go 2.Click on My Account dropdown button 3.Enter Valid username/email in Email text box 4.Enter Invalid password in password text box 5.Click on login button	Username: chalam@gmail.com password: Testing123678686786876876	Application should show "Incorrect email or password " validation message.	Working as expected	pass	Mathusudhan
LoginPage_TC_006	Functional	Login page	Verify user is able to log into application with InValid credentials	1.Enter URL(https://shopenzer.com/) and click go 2.Click on My Account dropdown button 3.Enter InValid username/email in Email text box 4.Enter Invalid password in password text box 5.Click on login button	Username: ibmmseec@gamil.com password: Testing654	Application should show "Incorrect email or password " validation message.	Working as expected	pass	Vishnudev

7.4. User Acceptance Testing

8.2.1. Defect Analysis

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	4	2	3	20
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	11	2	4	20	37
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Totals	24	14	13	26	77

8.2.2. Testcase Analysis

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	7	0	0	7
Client Application	51	0	0	51
Security	2	0	0	2
Outsource Shipping	3	0	0	3
Exception Reporting	9	0	0	9
Final Report Output	4	0	0	4
Version Control	2	0	0	2

8. RESULTS

8.1. Performance Metrics

9.1.1. Machine Learning

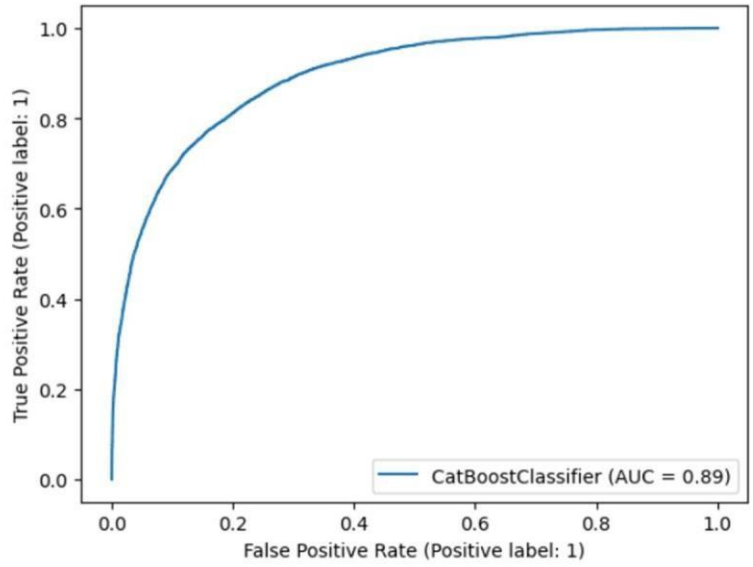
S.No.	Parameter	Values	Screenshot

1.	Metrics	**Classification Model:** Confusion Matrix - Accuracy Score- Classification Report -	`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))`
			`[[21510 1207]`
			` [ 2795 3580]]`
			`0.8624364086346762`
			`precision recall f1-score support`
			`0 0.89 0.95 0.91 22717`
			`1 0.75 0.56 0.64 6375`
			`accuracy 0.86 29092`
			`macro avg 0.82 0.75 0.78 29092`
			`weighted avg 0.85 0.86 0.85 29092`

2.	Tune the Model	Hyperparameter Tuning – Validation Method -	`{'learning_rate': 0.1, 'max_depth': 8}`
			`0.8892227301457538`
			`Accuracy:83.11 %`
			`Standard Deviation:17.73 %`

9.1.2. Artificial Intelligence

S.No.	Parameter	Values	Screenshot

1.	Model Summary	-	`metrics.plot_roc_curve(cat, X_test, y_test)`
			`metrics.roc_auc_score(y_test, y_pred, average=None)`
			`0.7542183058899486`
			 

2.	Accuracy	Training - Accuracy Validation Accuracy -	Epoch 40/150 2537/2537 [==========] - 11s 4ms/step - loss: 0.3941 - accuracy: 0.8425 - val_loss: 0.3656 - val_accuracy: 0.8495 Epoch 41/150 2537/2537 [==========] - 11s 4ms/step - loss: 0.3931 - accuracy: 0.8421 - val_loss: 0.3655 - val_accuracy: 0.8497 Epoch 42/150 2537/2537 [==========] - 11s 4ms/step - loss: 0.3930 - accuracy: 0.8423 - val_loss: 0.3656 - val_accuracy: 0.8494 Epoch 43/150 2537/2537 [==========] - 11s 4ms/step - loss: 0.3924 - accuracy: 0.8422 - val_loss: 0.3654 - val_accuracy: 0.8498 Epoch 44/150 2537/2537 [==========] - 11s 4ms/step - loss: 0.3921 - accuracy: 0.8418 - val_loss: 0.3654 - val_accuracy: 0.8496 Epoch 45/150 2537/2537 [==========] - 10s 4ms/step - loss: 0.3903 - accuracy: 0.8424 - val_loss: 0.3652 - val_accuracy: 0.8488 Epoch 46/150 2537/2537 [==========] - 11s 4ms/step - loss: 0.3914 - accuracy: 0.8429 - val_loss: 0.3652 - val_accuracy

9. ADVANTAGES AND DISADVANTAGES

9.1. Advantages

- Farmers can know when to plant or harvest their crops
- People can choose where and when to take their holidays to take advantages of good weather
- Surfers known when large waves are expected
- Regions can be e vacuated if hurricanes or floods are expected
- Aircraft and shipping rely heavily on accurate weather forecasting

9.2. Disadvantages

- Weather is extremely difficult to forecast correctly
- It is expensive to monitor so many variables from so many sources
- The computers needed to perform the millions of calculations necessary are expensive
- The weather forecasters get blamed if the weather is different from the forecast

10. CONCLUSION

The weather prediction has become one of the most essential entities now a days. To improve the risk management systems and to know the weather in coming days in an automatic and in scientific way, many models have been emerging to assist in weather Prediction. In this paper, we have seen building a Weather Prediction Web Application from scratch by making use of 6 different ML algorithms namely CatBoost Classifier, Random Forset Classifier, Logistic Regression, GaussianNB, KNN and XGB Classifier. In the result section, the results from the all the six models and its results such as Accuracy, Error rate, mean absolute error, Root mean squared error, Relative squared error, Root relative squared error and time taken to build the model are tabulated. The results show that the CatBoost Classifier and XGB Classifier has output the results of high accuracy than all the other classifiers that were used. When coming to the time taken to build the model, The CatBoost Classifier outperforms all the other classifiers in solving the Problem under scrutiny.

11. FUTURE SCOPE

In upcoming future updates, the WEATHER FORECASTING application will have additional features such as:

- Live Location tracking
- News on Live Disasters
- Weather Forecast for next one week
- Will deploy as android app
- Help in predicting which crop will be best suited according to weather conditions

12. APPENDIX

12.1. Source Code

13.1.1. Ipynb file Link: RAINFALL PREDICTION

13.1.2. UI Link: FILE

12.2. Links

13.2.1. GITHUB

13.2.2.DEMO LINK