88Project Report

INTRODUCTION

1.1 Overview

This project is used to analyze rainfall based on several fields of data which are collected by various methods, these data are well analyzed by the model created in python and the result derived from it. By utilizing the results generated one can improve their Agricultural Field.

1.2 Purpose

The main purpose of the project is to predict whether there will be heavy rainfall tomorrow so that the farmer may take precautionary measures to safe guard his crop fields.

1. LITERATURE SURVEY

2.1 Existing problem

Some of the existing solutions for solving this problem are:

1.How can I predict rainfall using machine learning techniques?

For rainfall/precipitation or for any kind of numerical weather prediction you can use any classification problem algorithm such as XGBoost , Logistic Regression or any other relevant technique based upon binary classification problems (Those resulting in a binary output).

2. What are the objectives of rainfall prediction?

Rainfall Prediction Model has a main objective in prediction of the amount of rain in a specific well or division in advance by using various regression technique and find out which one is best for rainfall prediction. This model also helps the farmer for agriculture to decide the crop, helping the watershed department for water storage and also helps to analyze the ground water level.

1.2 References

[1]Parmar, Aakash, Kinjal Mistree, and Mithila Sompura.

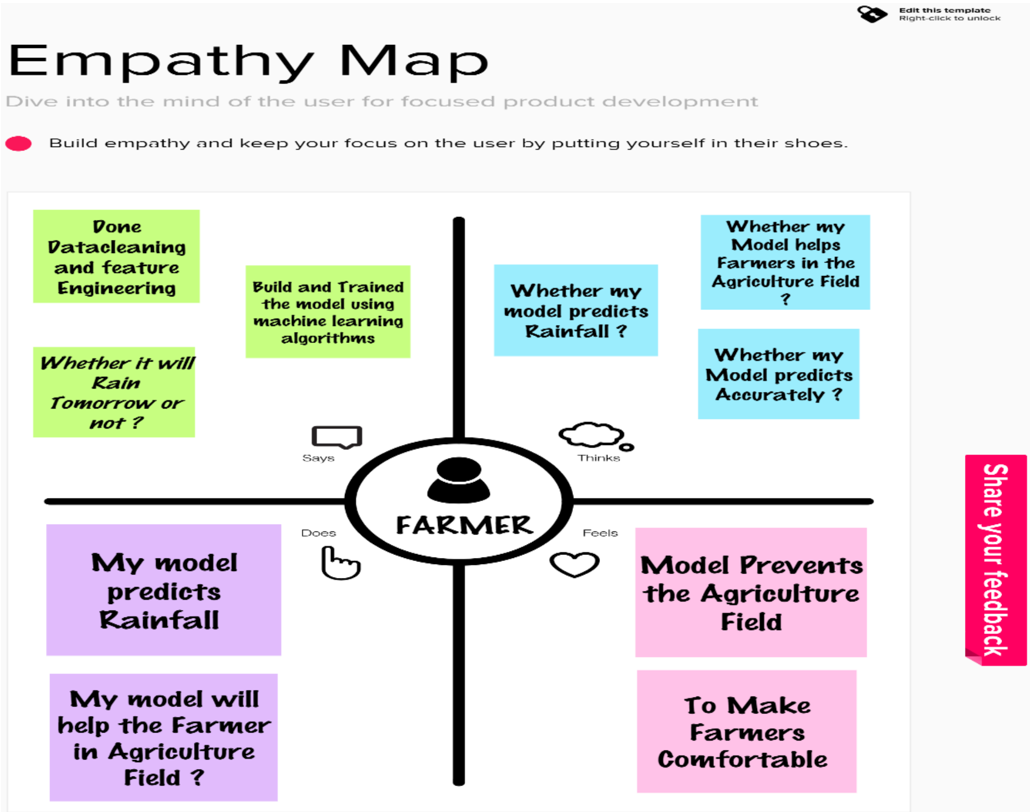
"Machine learning techniques for rainfall prediction: A review." 2017 International Conference on Innovations in information Embedded and Communication Systems. 2017.
[2] Dash, Yajnaseni, Saroj K. Mishra, and Bijaya K. Panigrahi. "Rainfall prediction for the Kerala state of India using artificial intelligence approaches." Computers & Electrical Engineering 70 .

2.3 Problem Statement Definition

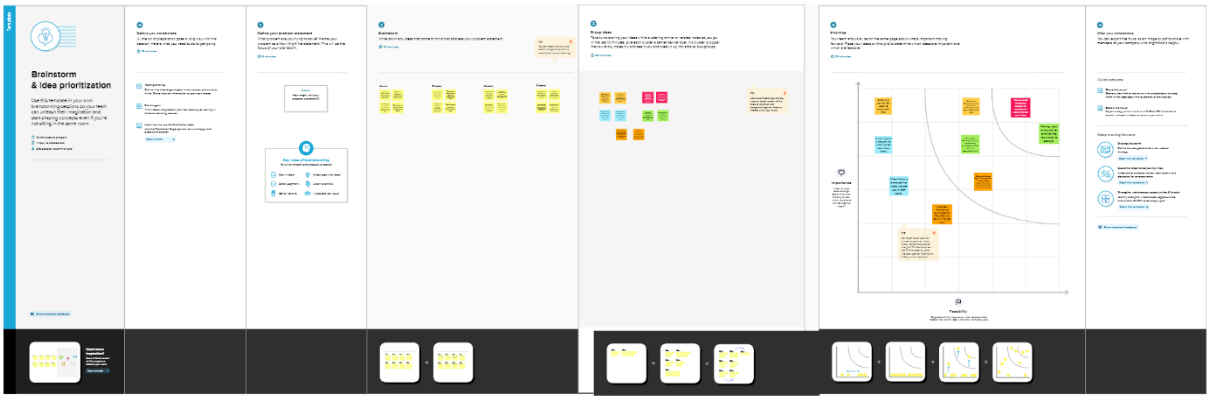
This paper introduces current supervised learning models which are based on machine learning algorithm for Rainfall prediction in India. Rainfall is always a major issue across the world as it affects all the major factor on which the human being is depended. In current, Unpredictable and accurate rainfall prediction is a challenging task. We apply rainfall data of India to different machine learning algorithms and compare the accuracy of classifiers such as XGBOOST , Logistic Regression.

2. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation & Brainstorming

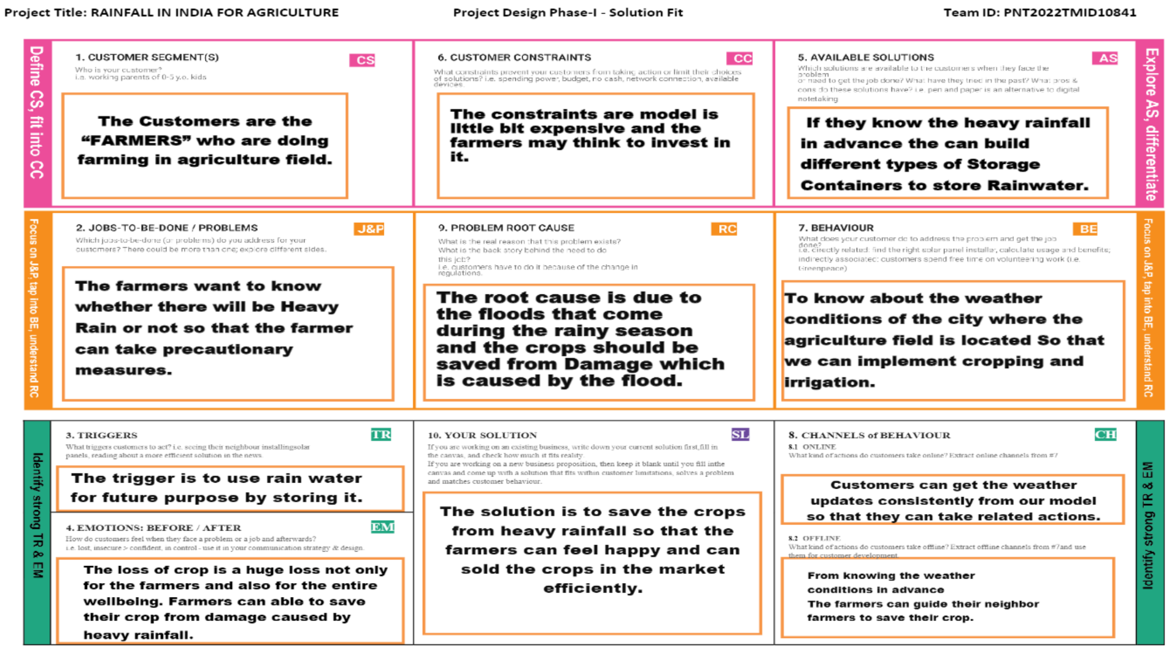


3.3 Proposed Solution

S. No.	Parameter	Description

1.	Problem Statement (Problem to be solved)	The Farmer wants to Know whether there is a Rainfall Tomorrow or not so that His Agricultural Field can be maintained properly. The Farmer wants to Harvest/Save the water so that it can be used for Future Purpose. The Farmer wants to Know There will be a Heavy Rainfall so that the Precautionary Measures can be tken.
2.	Idea / Solution description	1 .Pesticides should be used after the rain in order to avoid wastage. 2 .When It is Sunny the probability of Rain is low. 3. Drip Irrigation can be used in order to save water. 4 .Setting up the soil to equally distribute the rain water to all the plants in the field. 5. Creating a way for rain water to move from field. 6.We can avoid using motor pumps while raining. 7. We can build temporary storage for storing rain water and can use it later. 8. When storm is predicted don't yield the crop. 9.Making ways in the soil for overflowing rain water to storage. 10.Creating a Drainage in order to preserve overflowing of Rain water in Agriculture field.
3.	Novelty / Uniqueness	Applied appropriate machine learning algorithms to get the best results. 2. Forecasted rainfall with Time Series.
4.	Social Impact / Customer Satisfaction	1 . Farmers will be satisfied to save the rain water. 2 . It will be useful to avoid flood in Agriculture field in order to take precautionary measures.
5.	Business Model (Revenue Model)	Since we predicted the rainfall in advance so we may able to avoid the loss of cost for the Farmers. Using this idea, we can make a stable business and get a profitable revenue.
6.	Scalability of the Solution	Our project has better scalability since our model analysis all information provides better refined solution. With the help of this Prediction it will be easy for the farmers to cultivate in the agricultural field.

3.4 Problem Solution fit



3. REQUIREMENT ANALYSIS

4.1Functional requirement

Following are the functional requirements of the proposed solution.

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	Reset Password	Reset password through Gmail Reset password through Mobile number
FR-4	Feedback	The user can submit the feedback through a contact form in the website or through Gmail.

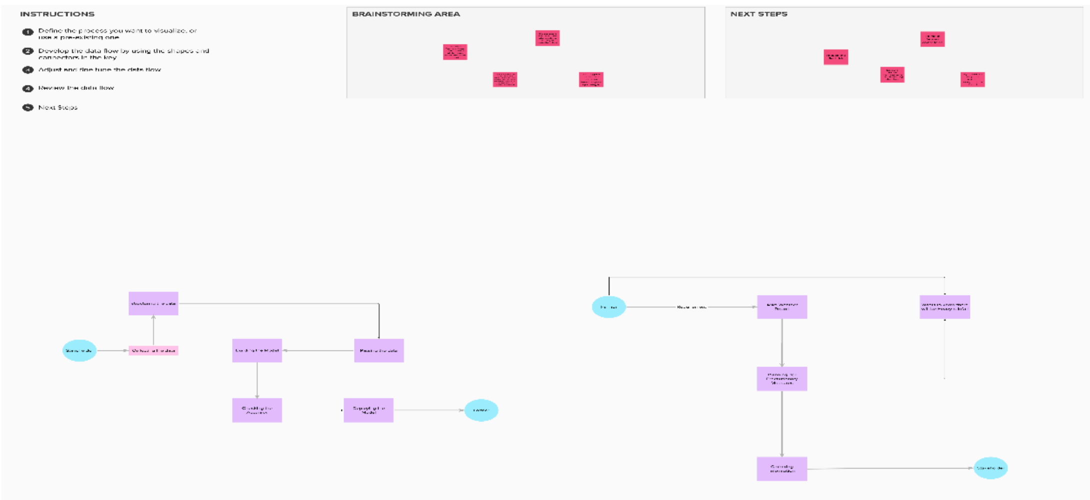
4.2 Non-Functional requirements

Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The Analyser allows the user to know whether there will be a heavy rainfall or not using the data provided.
NFR-2	**Security**	By knowing the rain in advance we can secure the humans and animals by turning off the electrical fences.
NFR-3	**Reliability**	The reliability rating is good due to best performance, less frequency of problem occurrence and cost for repairing is low.
NFR-4	**Performance**	The model is built in an optimised manner and the model will be more accurate on predictions.
NFR-5	**Availability**	Weather Data of various cities in India is collected from the Google weather Database.
NFR-6	**Scalability**	Since we use various machine learning algorithms for predicting the chance of rainfall, we use performance metrics like accuracy, AUC, Precision, Recall, etc , to measure the performance of our model.

4. PROJECT DESIGN

4.2 5.1 Data Flow Diagrams



5.2 Solution & Technical Architecture:

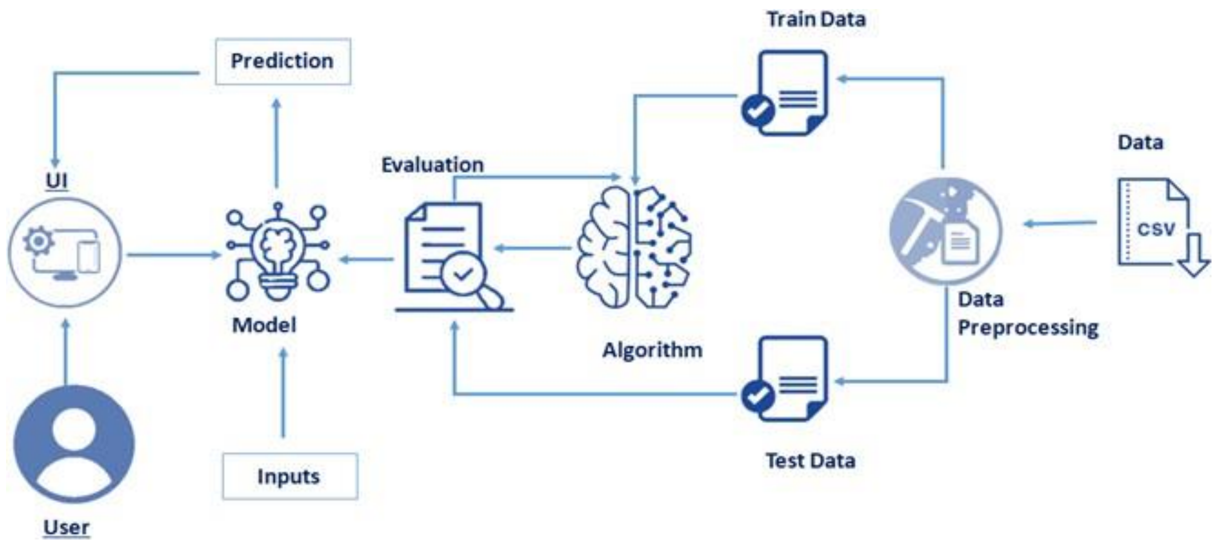


Table-5.2.1: Components & Technologies:

S. No	Component	Description	Technology
1.	User Interface	With the help of web UI, user has better experience And can access the website user-friendly.	HTML, CSS, JavaScript, React JS.
2.	Application Logic-1	Customer can login with username and password.	Java / Python
3.	Application Logic-2	Farmer can give their feedback about weatherconditions.	IBM Watson STT service
4.	Application Logic-3	Farmer can check whether there will be heavy rainfall tomorrow or not and can able to take Precautionary measures.	IBM Watson Assistant
5.	Database	Data Type, Configurations etc.	MySQL.
6.	Cloud Database	Database Service on Cloud	IBM DB2, IBM Cloudant etc.
7.	File Storage	File storage requirements	IBM Block Storage or Other Storage Service or Local Filesystem
8.	External API-1	Purpose of External API used in the application	Aadhar API
9.	External API-2	Purpose of External API used in the application	Aadhar API
10	Machine Learning Model	To create model for analysis	1. Logistic Regression 2. Decision Tree Classifier

			3. Random Forest Classifier 4.KNN 5. SVM xgboost
11	Infrastructure (Server / Cloud)	Application Deployment on Local System / CloudLocal Server Configuration: Cloud Server Configuration :	Local, Cloud Foundry, Kubernetes, etc.

Table-5.2.2: Application Characteristics:

S. No	Characteristics	Description	Technology
1.	Open-Source Frameworks	List the open-source frameworks used REACT JS EXPRESS JS NODE JS	Technology of Opensource frameworkJAVASCRIPT and PYTHON
2.	Security Implementations	List all the security / access controls implemented, use of firewalls etc.	e.g. SHA-256, Encryptions, IAM Controls, OWASP etc.
3.	Scalable Architecture	Justify the scalability of architecture (3 – tier, Micro-services) This improves scalability, because application servers can be deployed on many machines. The database does not make longer connections with every client – it only requires connections from a smaller number ofapplication servers	Presentation Layer – React JS (HTML, CSS ,JS) Application Layer – Flask (Python) Data Layer – IBM DB2
4.	Availability	Justify the availability of application (e.g. use of load balancers, distributed servers etc.)	Technology used
5.	Performance	Design consideration for the performance of the application (number of requests per sec, use of Cache, use of CDN's) etc.	Technology used

5.3 User Stories:

User Type	Functional Requirements	User Story Number	User Story/Task	Acceptance Criteria	Priority	Release
Farmer	Registration	USN -1	As a Farmer, I can register for the Model service by entering my email, password, and confirming my password	I can access my account / dashboard	High	Sprint-1

Farmer	Registration	USN - 2	As a Farmer, I will receive confirmation email once I have registered for the service	I can receive confirmation email & click confirm	High	Sprint-2
Farmer	Accuracy to check the performance and health of the car	USN -3	After checking , the model build by the Admin will be sent to the Farmer.	The rainfall data can be predicted predicted.	High	Sprint-3

5. PROJECT PLANNING & SCHEDULING

5.2 6.1 Sprint Planning

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Required Data	USN-1	Model Building	2	High	Aswin N S Divakar U
Sprint-2		USN-2	Application Building	1	High	Aswin N S Divakar U
Sprint-3		USN-3	Train the model on IBM	2	Low	Aswin N S Divakar U Krishna R
Sprint-4		USN-4	Integrate flask with scoring end-point	2	Medium	Aswin N S Kailash N

6.2 **Estimation:**

Pre-Requisites	M-01	The following software concepts and packages, including Machine learning, Python, KNN, Python Flask, IBM Cloudland DB, and Watson Studio, should have been familiar to us by the time we finished this project.	Yes
Data Collection	M-02	To create a project structure, create a Dataset.	Yes
Data Preprocessing	M-03	The dataset collection is separated into a various collection, first reading the dataset, handling the missing values, label encoding and one hot coding, splitting the dataset into dependent and independent variable, and into trainset and test set and normalizing and finally importing libraries.	Yes
Model Building	M-04	Build the model with the random forest regressor, predict the values and model the evaluation	Yes
Application Building	M-05	First, build an Index, HTML file, python code and python code-II, Run the app and finally output.	Yes
Train the model on IBM	M-06	Register on cloud IBM, train the model on IBM and integrate with the flask with scoring end point.	Yes

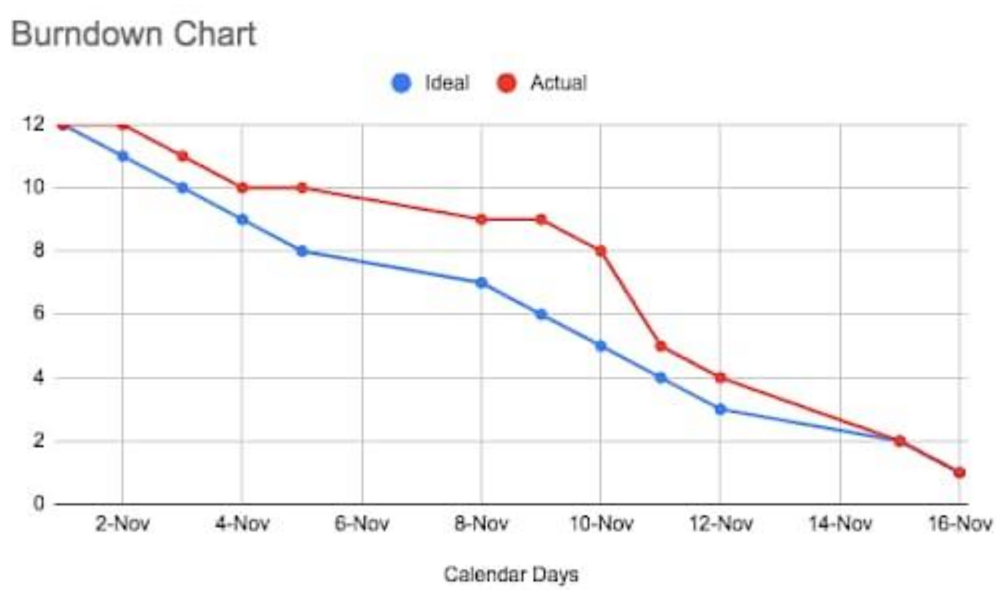
Ideation Phase	M-07	Prepare empathy map, take literature survey and Ideation.	Yes
Project Design Phase-I	M-08	Proposed Solution, Problem solution fit, Solution architecture.	Yes
Project Design Phase-II	M-09	Preparation of the technological stack architecture, functional requirements, data flow diagrams, and customerjourney mapping.	Yes
Project Planning Phase	M-10	Prepare Milestone & Activity List and Sprint Delivery Plan.	Yes
Project developme nt phase	M-11	Develop Sprint 1, Sprint 2, Sprint 3, Sprint 4.	Yes

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 ov 2022	20	19 Nov 2022

6.3 Reports from JIRA:

Burndown Chart: A burndown chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.



6. CODING & SOLUTIONING:

6.1 Features:

FR No.	Feature	Description
FR-1	Enter the input	Get input through the form
FR-2	User Essential	Predict the whether there will be rainfall tomorrow
FR-3	Data preprocessing	Sample dataset for training purpose
FR-4	User input Evaluation	Evaluating the given user values
FR-5	Prediction	Predict the whether there will be rainfall tomorrow

DFD:

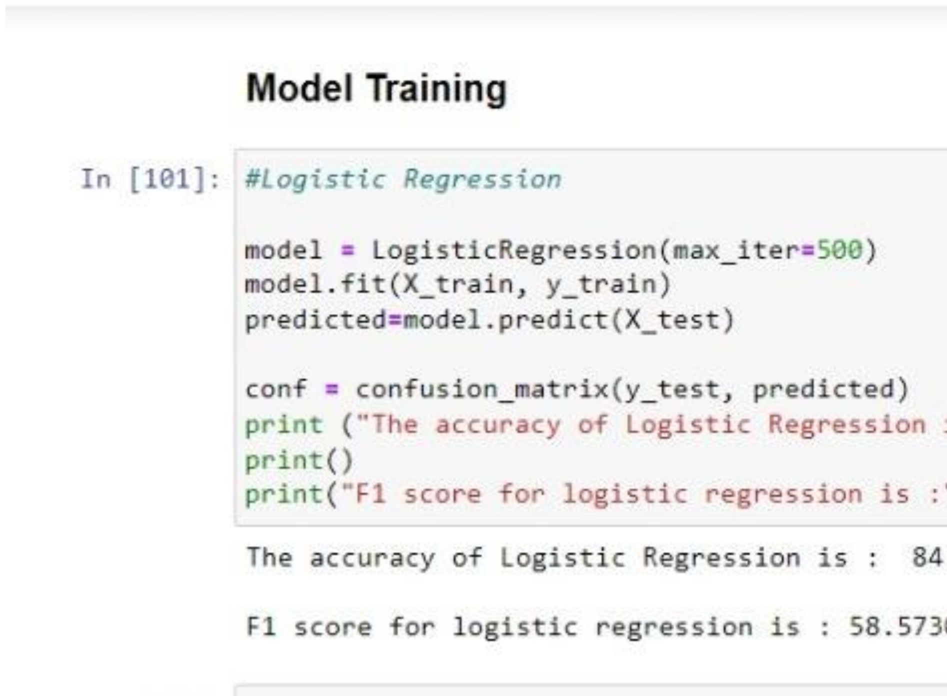
7. RESULTS:

7..1 PERFORMANCE METRICS:

S. NO	PARAMETER	VALUES	SCREENSHOT

1.	Metrics	XGBOOST model Accuracy-85.54	Xgboost model
			
			In [46]: #XGBoost
			X_train=X_train.values
			y_train = y_train.values
			y_test=y_test.values
			xgbc = XGBClassifier(objective='binary:logisti
			xgbc.fit(X_train,y_train)
			predicted = xgbc.predict(X_test)
			print ("The accuracy of XGBoost is : ", accura
			print("F1 score for XGBoost is :",f1_score(y_t
			
			The accuracy of XGBoost is : 85.5492919015536
			F1 score for XGBoost is : 62.34324614833393 %

2.	Accuracy	Logistic regression Accuracy-84.55 F1 Score - 58.57	Logistic Regression

			 Model Training In [101]: #Logistic Regression model = LogisticRegression(max_iter=500) model.fit(X_train, y_train) predicted=model.predict(X_test) conf = confusion_matrix(y_test, predicted) print ("The accuracy of Logistic Regression i print() print("F1 score for logistic regression is :" The accuracy of Logistic Regression is : 84. F1 score for logistic regression is : 58.573

8. PROS AND CONS:

8.1 PROS:

- XGB consists of a number of hyper-parameters that can be tuned — a primary advantage over gradient boosting machines.
- XGBoost has an in-built capability to handle missing values.
- It provides various intuitive features, such as parallelisation, distributed computing, cache optimisation, and more.

8.2 CONS:

- XGBoost does not perform so well on sparse and unstructured data.
- A common thing often forgotten is that Gradient Boosting is very sensitive to outliers since every classifier is forced to fix the errors in the predecessor learners.
- The overall method is hardly scalable.

9. CONCLUSION:

Rainfall forecasting has gained utmost research relevance in recent times due to its complexities and persistent applications such as flood forecasting and monitoring of pollutant concentration levels, among others. Existing models use complex statistical models that are often too costly, both computationally and budgetary, or are not applied to downstream applications. Therefore, approaches that use Machine Learning algorithms in conjunction with time-series data are being explored as an alternative to overcome these drawbacks. To this end, this study presents a comparative analysis using simplified rainfall estimation models based on conventional Machine Learning algorithms XGBoost Classifier , Logistic Regression were compared in the task of forecasting hourly rainfall volumes using time-series data. Climate data from 2000 to 2022 from five major cities in the United Kingdom were used. The evaluation metrics of Accuracy were used to evaluate the models' performance.

10.FUTURE WORKS:

Rainfall Prediction Model has a main objective in prediction of the amount of rain in a specific well or division in advance by using various regression technique and find out which one is best for rainfall prediction. Weather warnings are important forecasts because they are used to protect life and property. Forecasts based on temperature and precipitation are important to agriculture, and therefore to traders within commodity markets. Temperature forecasts are used by utility companies to estimate demand over coming days. So we Build this model using xgboost and logistic Algorithms , so that it will be useful in the future.

11.APPENDIX:

Source code:

This dataset contains about 10 years of daily weather observations from many locations across India.

The prediction is all about Tomorrow it going to rain or not based on all factors like temperature,humidity,pressure etc. Target_variable:Rain_tomorrow.

```
import os
import pandas as pd
pd.set_option('display.max_columns', None)
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

import seaborn as sns
sns.set_style("whitegrid")
from sklearn.preprocessing import RobustScaler, LabelEncoder
from sklearn.model_selection import train_test_split
import missingno as msno

import time
import pickle
from collections import Counter
from imblearn.over_sampling import SMOTE
from xgboost import XGBClassifier
from sklearn.linear_model import LogisticRegression
import warnings
warnings.filterwarnings('ignore')

df = pd.read_excel ("Dataset.xlsx")
```

```
# lets check the shape of dataset
df.shape
```

```
(145460, 23)
```

```
# Lets check the first five rows of dataset
df.head()
```

```
         Date Location  MinTemp  MaxTemp  Rainfall  Evaporation  Sunshine  \
0  2008-12-01    Delhi     13.4     22.9       0.6          NaN       NaN
1  2008-12-02    Delhi      7.4     25.1       0.0          NaN       NaN
2  2008-12-03    Delhi     12.9     25.7       0.0          NaN       NaN
3  2008-12-04    Delhi      9.2     28.0       0.0          NaN       NaN
4  2008-12-05    Delhi     17.5     32.3       1.0          NaN       NaN

  WindGustDir  WindGustSpeed WindDir9am WindDir3pm  WindSpeed9am  \
0           W           44.0          W        WNW          20.0
1         WNW           44.0        NNW        WSW           4.0
2         WSW           46.0          W        WSW          19.0
3          NE           24.0         SE          E          11.0
4           W           41.0        ENE         NW           7.0

   WindSpeed3pm  Humidity9am  Humidity3pm  Pressure9am  Pressure3pm  Cloud9am  \
0          24.0         71.0         22.0       1007.7       1007.1       8.0
1          22.0         44.0         25.0       1010.6       1007.8       NaN
2          26.0         38.0         30.0       1007.6       1008.7       NaN
3           9.0         45.0         16.0       1017.6       1012.8       NaN
4          20.0         82.0         33.0       1010.8       1006.0       7.0

   Cloud3pm  Temp9am  Temp3pm RainToday RainTomorrow
0       NaN     16.9     21.8        No           No
1       NaN     17.2     24.3        No           No
2       2.0     21.0     23.2        No           No
3       NaN     18.1     26.5        No           No
4       8.0     17.8     29.7        No           No
```

```
# Let's get an overview of features datatype
df.dtypes
```

```
Date              object
Location          object
MinTemp          float64
MaxTemp          float64
Rainfall         float64
Evaporation      float64
Sunshine         float64
WindGustDir       object
WindGustSpeed    float64
WindDir9am        object
WindDir3pm        object
WindSpeed9am     float64
WindSpeed3pm     float64
Humidity9am      float64
Humidity3pm      float64
Pressure9am      float64
Pressure3pm      float64
```

```
Cloud9am         float64
Cloud3pm         float64
Temp9am          float64
Temp3pm          float64
RainToday         object
RainTomorrow      object
dtype: object
```

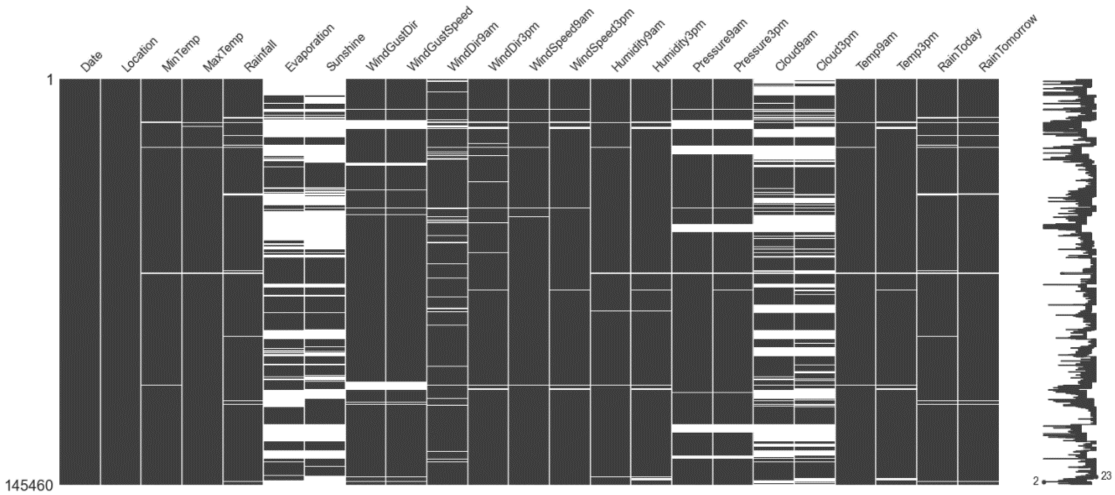
```
# Lets check the missing values if any
df.isnull().sum()
```

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

We can clearly see that there are lots of missing values in a dataset which we'll have to fill or drop to start the further analysis.

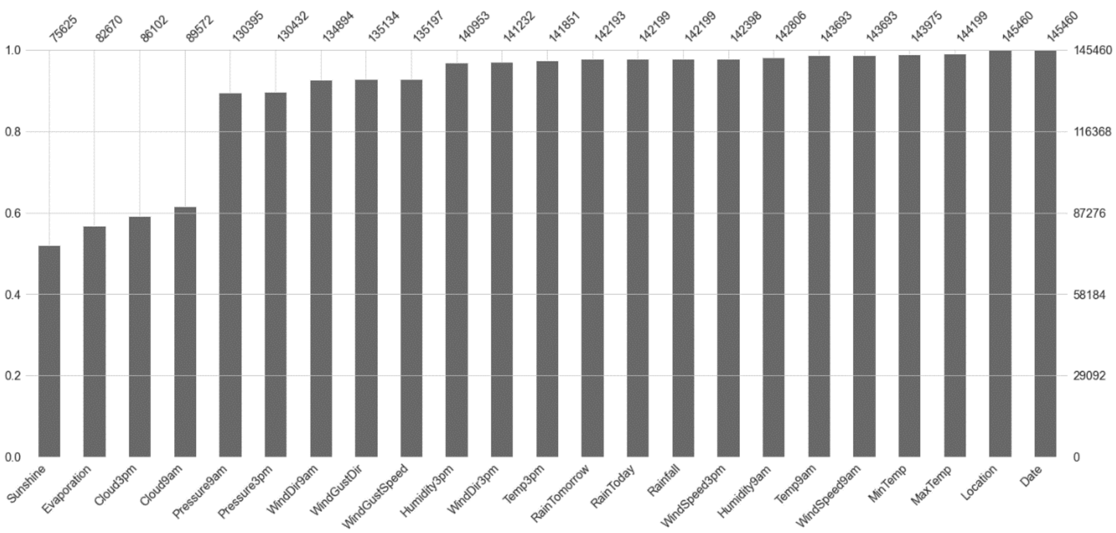
```
#Visualizing the missing values
msno.matrix(df)
```

```
<AxesSubplot:>
```




```
#Visualizing the missing values
msno.bar(df,sort='ascending')
```

```
<AxesSubplot:>
```

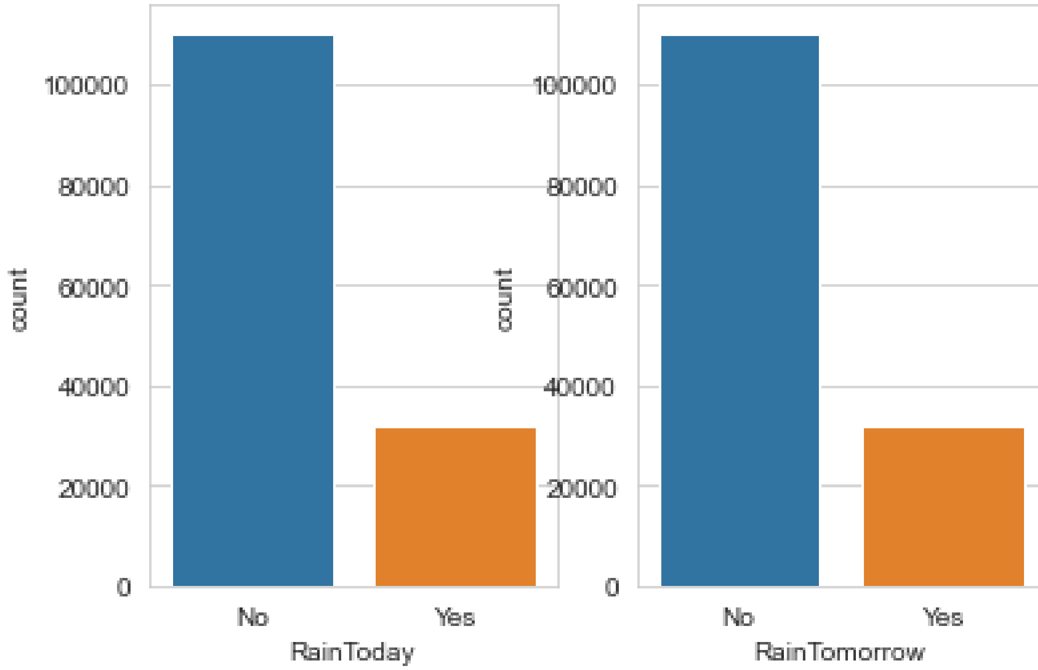


Analysis of Target Feature

```
#Data Visualization
#Count of rain today and tomorrow

fig, ax =plt.subplots(1,2)
print(df.RainToday.value_counts())
print(df.RainTomorrow.value_counts())

plt.figure(figsize=(20,20))
sns.countplot(data=df,x='RainToday',ax=ax[0])
sns.countplot(data=df,x='RainTomorrow',ax=ax[1])
```

```
No     110319
Yes     31880
Name: RainToday, dtype: int64
No     110316
Yes     31877
Name: RainTomorrow, dtype: int64
```

```
<AxesSubplot:xlabel='RainTomorrow', ylabel='count'>
```




```
<Figure size 1440x1440 with 0 Axes>
```

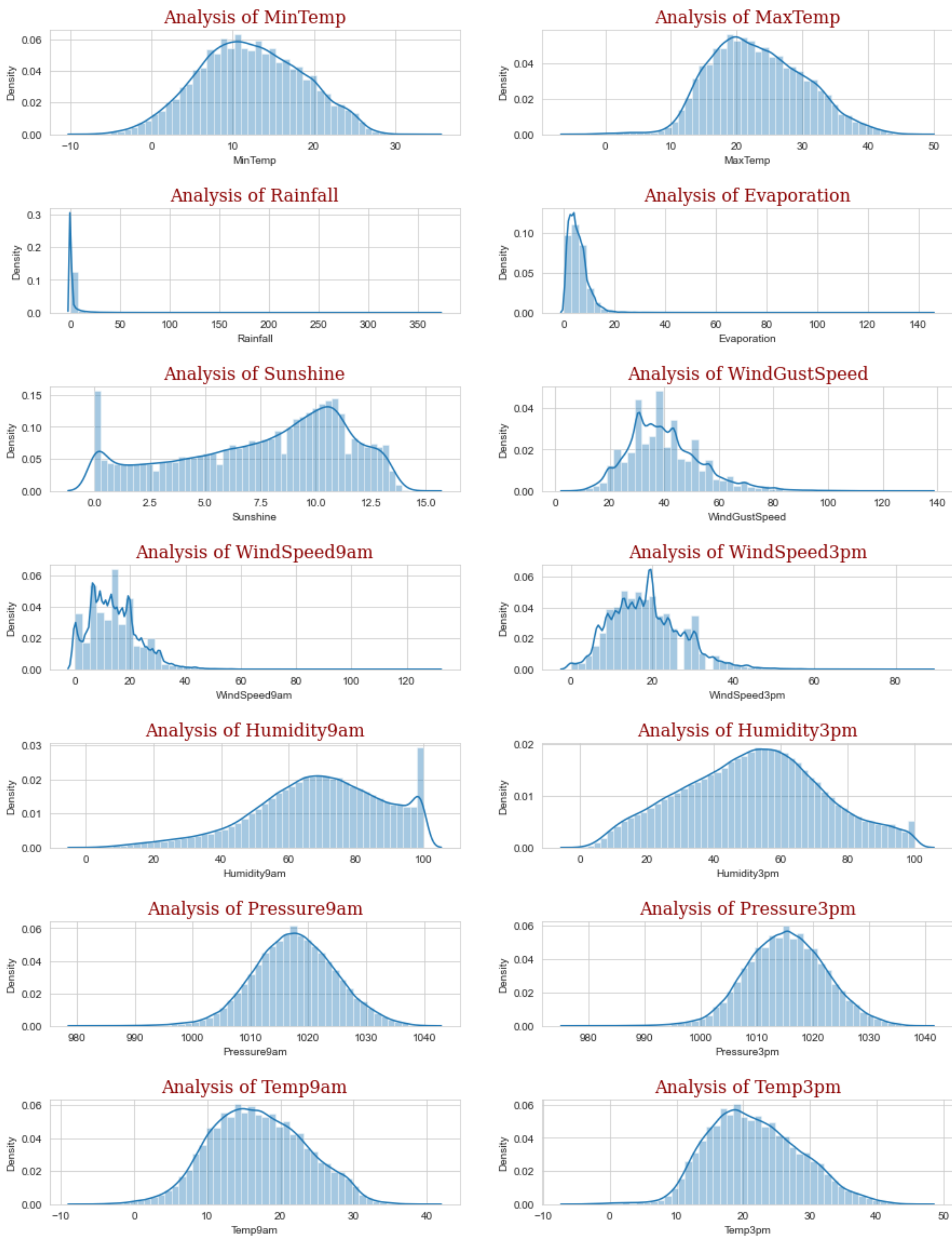
Analysis of Continuous Features

```
numerical_features = [feature for feature in df.columns if df[feature].dtypes != 'O']
discrete_features = [feature for feature in numerical_features if
len(df[feature].unique())<25]
continuous_features = [feature for feature in df.columns if rain[i].dtype == 'object']
categorical_features = [feature for feature in df.columns if feature not in
numerical_features]
binary_categorical_features = [feature for feature in categorical_features if
len(df[feature].unique()) <=3]
print("Numerical Features Count {}".format(len(numerical_features)))
print("Discrete features Count {}".format(len(discrete_features)))
print("Continuous features Count {}".format(len(continuous_features)))
print("Categorical features Count {}".format(len(categorical_features)))
print("Binary Categorical features Count {}".format(len(binary_categorical_features)))
```

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

```
def generate_distribution_plot(df, continuous_features):
    # create copy of dataframe
    data = df[continuous_features].copy()
    # Create subplots
    fig, axes = plt.subplots(nrows=len(data.columns)//2, ncols=2,figsize=(15,20))
    fig.subplots_adjust(hspace=0.7)

    # set fontdict
    font = {'family': 'serif',
```

```python
        'color':  'darkred',
        'weight': 'normal',
        'size': 16,
        }

    # Generate distplot
    for ax, feature in zip(axes.flatten(), data.columns):
        sns.distplot(data[feature],ax=ax)
        ax.set_title(f'Analysis of {feature}', fontdict=font)
    plt.show()

generate_distribution_plot(df, continuous_features)
```



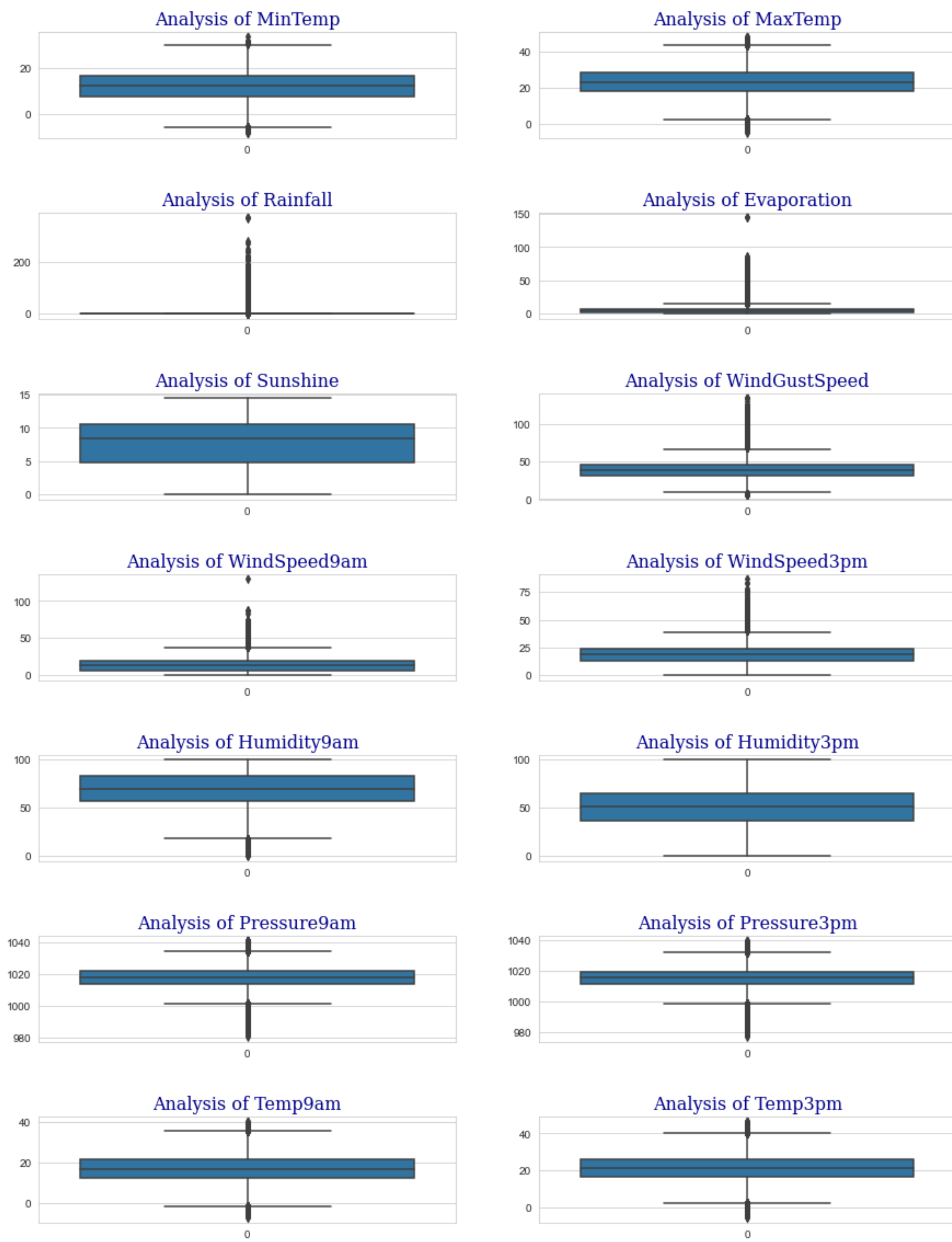
```
sample_imputation_features = [col for col in df.columns if (df.isnull().sum()[col] >
50000)]

# Random Sampling for high number of missing values features-
def randomsampleimputation(df, columns):
    data = df.copy()
    for column in columns:
        random_sample =
data[column].dropna().sample(data[column].isnull().sum(),random_state=2022)
        random_sample.index = data[data[column].isnull()].index
        data.loc[data[column].isnull(),column] = random_sample
    return data

df = randomsampleimputation(df,sample_imputation_features)

# list of numeric features with null values
missing_values_numeric_features  = [col for col in df.columns if (df.isnull().sum()[col]
> 0) & (df[col].dtypes != 'object')]

# Filling the Missing Values – Imputation
# Filling the missing data with the mean value for a numerical variable


# function for missing values substitution
def impute_means(df, missing_values_columns):
    data = df.copy()
    '''Filling missing values with mean'''
    for col in missing_values_columns:
        data[col] = data[col].fillna(data[col].mean())

    return data

# lets use this function to fill the missing values
df = impute_means(df,missing_values_numeric_features)

# checking the missing values again
df.isnull().sum()
```

```
Date                 0
Location             0
MinTemp              0
MaxTemp              0
Rainfall             0
Evaporation          0
Sunshine             0
WindGustDir      10326
WindGustSpeed        0
WindDir9am       10566
WindDir3pm        4228
WindSpeed9am         0
WindSpeed3pm         0
Humidity9am          0
Humidity3pm          0
Pressure9am          0
Pressure3pm          0
Cloud9am             0
```

```
Cloud3pm             0
Temp9am              0
Temp3pm              0
RainToday         3261
RainTomorrow      3267
dtype: int64
```

OnehotEncoding handles categorical features null values very cleverly so we will use get_dummies function from pandas to handle null values and convert the data into proper format to use machine learning model.

```
# sns.pairplot( data=df, vars=('MaxTemp','MinTemp','Pressure9am','Pressure3pm', 'Temp9am', 'Temp3pm', 'Evaporation'), hue='RainTomorrow' )
# plt.show()

def plot_boxplot(df, continuous_features):
    # create copy of dataframe
    data = df[continuous_features].copy()
    # Create subplots
    fig, axes = plt.subplots(nrows=len(data.columns)//2, ncols=2,figsize=(15,20))
    fig.subplots_adjust(hspace=0.7)

    # set fontdict
    font = {'family': 'serif',
        'color':  'darkblue',
        'weight': 'normal',
        'size': 16,
        }

    # Generate distplot
    for ax, feature in zip(axes.flatten(), data.columns):
        sns.boxplot(data[feature],ax=ax)
        ax.set_title(f'Analysis of {feature}', fontdict=font)
    plt.show()

plot_boxplot(df, continuous_features)
```




```
outliers_features = [feature for feature in continuous_features if feature not in
['Sunshine','Humidity3pm']]
print(outliers_features)
```

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

```
def remove_outliers(df,outliers_features):
    # create copy of dataframe

    data = df.copy()

    for feature in data[outliers_features].columns:
```

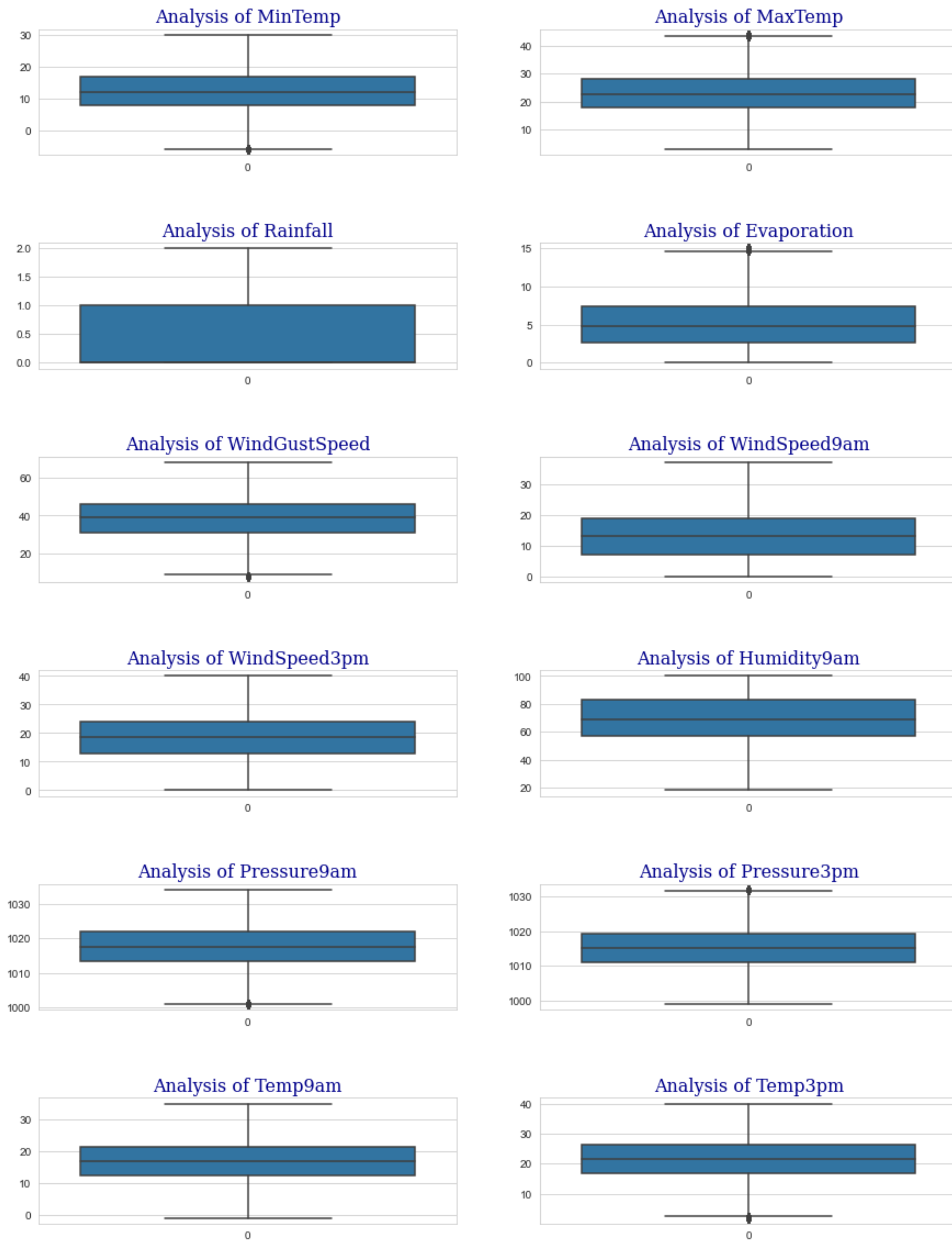
```
        Q3 = data[feature].quantile(0.75)
        Q1 = data[feature].quantile(0.25)
        IQR = Q3 - Q1
        lower_limit = round(Q1 - 1.5 * IQR)
        upper_limit = round(Q3 + 1.5 * IQR)
        data.loc[data[feature]>= upper_limit,feature] = upper_limit
        data.loc[data[feature]<=lower_limit,feature] = lower_limit
#         data = data[(data[feature] < upper_limit) & (data[feature] > lower_limit)]
    return data
```

```
df = remove_outliers(df,outliers_features)
```

```
df.shape
```

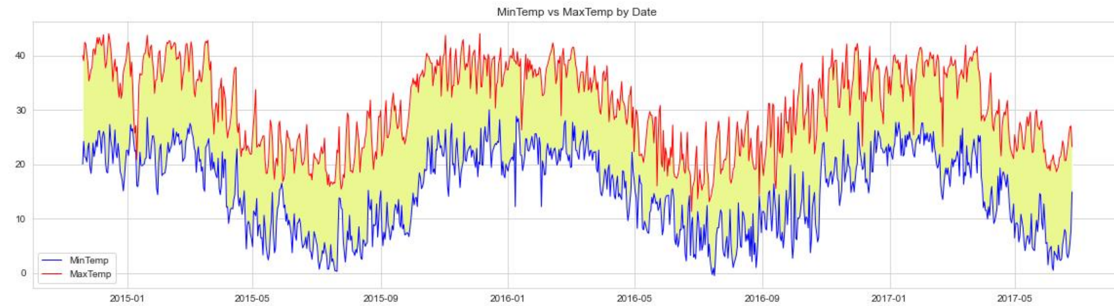
```
(145460, 23)
```

```
plot_boxplot(df, outliers_features)
```




```
# Raintoday
df['RainToday'] = df['RainToday'].fillna('No')
# Raintomorrow
df['RainTomorrow'] = df['RainTomorrow'].fillna('No')

df["Date"] = pd.to_datetime(df["Date"])
# using data from last 3 years.
df_last_3_years = df.iloc[-950:,:]
plt.figure(figsize=[20,5])
plt.plot(df_last_3_years['Date'],df_last_3_years['MinTemp'],color='blue',linewidth=1,
label= 'MinTemp')
plt.plot(df_last_3_years['Date'],df_last_3_years['MaxTemp'],color='red',linewidth=1,
label= 'MaxTemp')
```

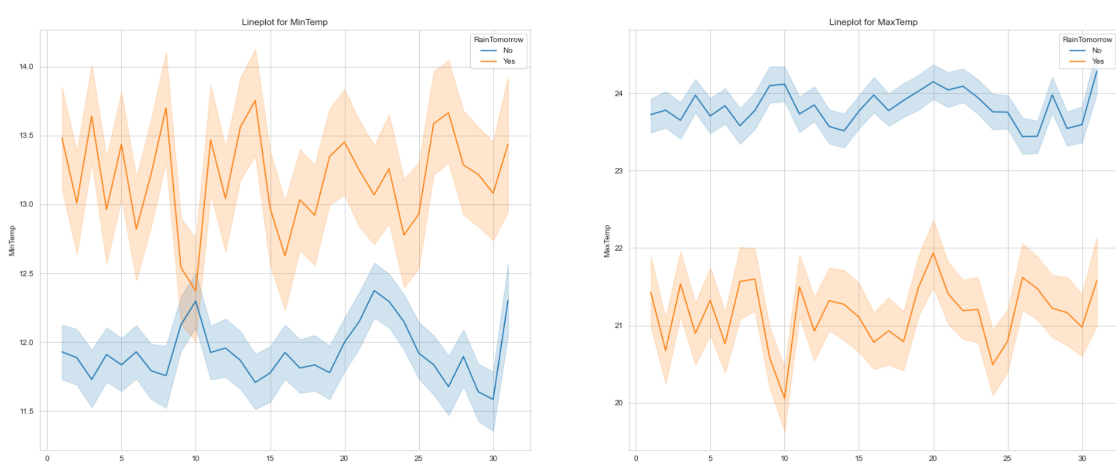
```
plt.fill_between(df_last_3_years['Date'],df_last_3_years['MinTemp'],df_last_3_years['MaxTemp'], facecolor = '#EBF78F')
plt.title('MinTemp vs MaxTemp by Date')
plt.legend(loc='lower left')
plt.show()
```



- Above plot shows that the MinTemp and MaxTemp relatively increases and decreases every year. As you can see that, December to February is summer; March to May is autumn; June to August is winter; and September to November is spring.
- The weather conditions are always opposite in the two hemispheres. As, the Australia is situated in the southern hemisphere. The seasons are bit different.

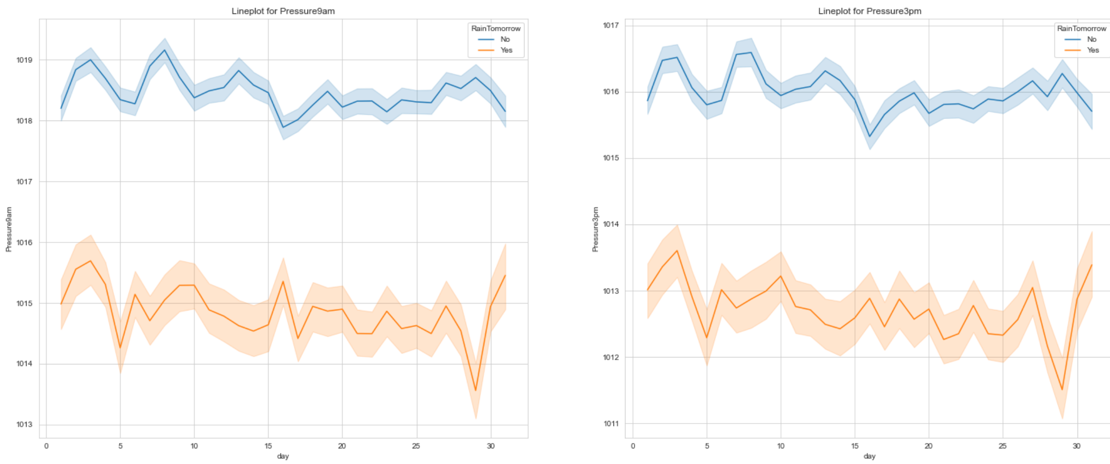
Handling DateTime Feature

```
df["year"] = df["Date"].dt.year
df["month"] = df["Date"].dt.month
df["day"] = df["Date"].dt.day

# We don't need date feature anymore for model building
df.drop('Date', axis=1, inplace=True)

fig, axes = plt.subplots(1, 2, figsize=(25, 10))

# Mintemp
sns.lineplot(ax=axes[0],x="day",y="MinTemp",hue="RainTomorrow",data=df)
axes[0].set_title('Lineplot for MinTemp')
# Maxtemp
sns.lineplot(ax=axes[1],x="day",y="MaxTemp",hue="RainTomorrow",data=df)
axes[1].set_title('Lineplot for MaxTemp')
plt.show()
```

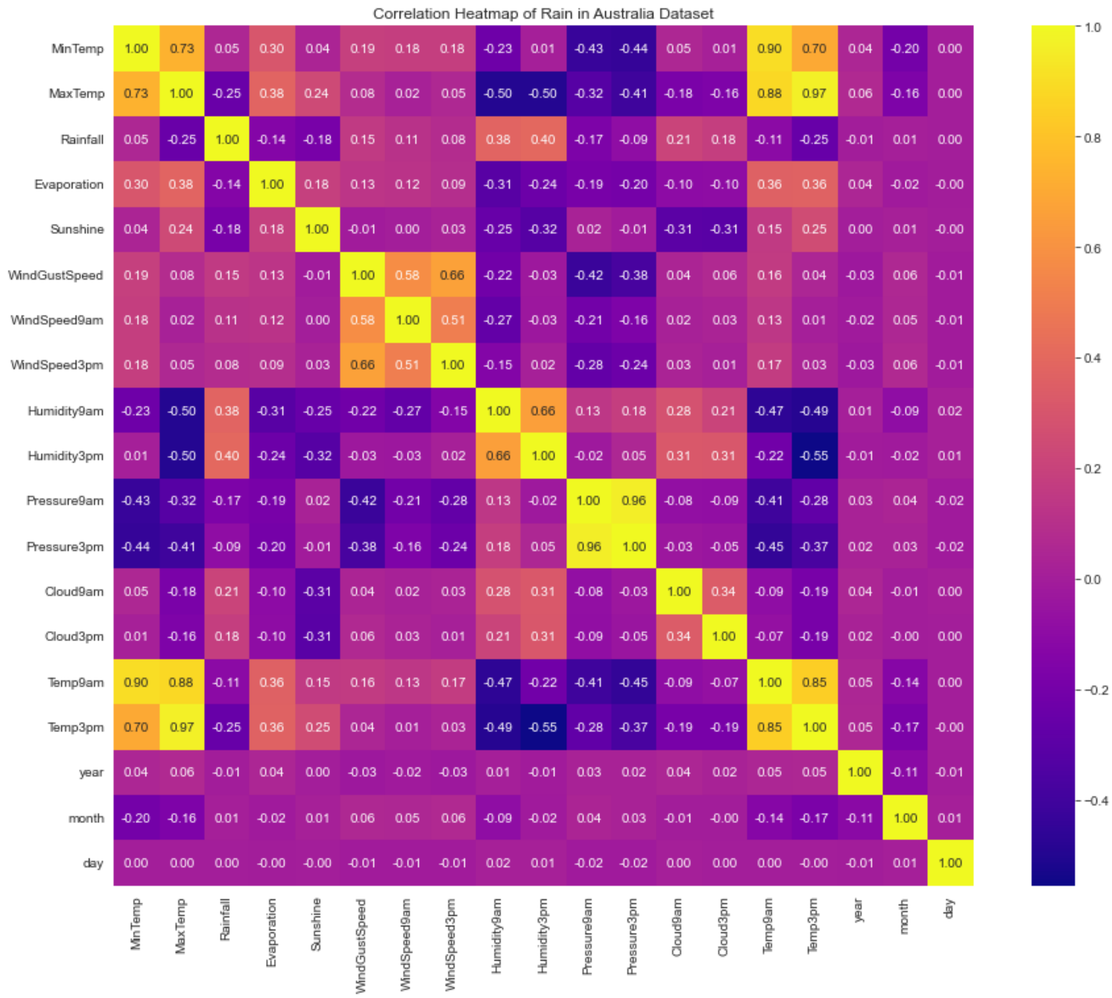


If temperature difference between min and max temperature is low then probality of rain occuring tomorrow is more.

```
fig, axes = plt.subplots(1, 2, figsize=(25, 10))

# Pressure9am
sns.lineplot(ax=axes[0],x="day",y="Pressure9am",hue="RainTomorrow",data=df)
axes[0].set_title('Lineplot for Pressure9am')
# Pressure3pm
sns.lineplot(ax=axes[1],x="day",y="Pressure3pm",hue="RainTomorrow",data=df)
axes[1].set_title('Lineplot for Pressure3pm')
plt.show()
```




```
# Lets check correlation again
corrmat = df.corr()

# heatmap
plt.figure(figsize=(16,12))
sns.heatmap(corrmat, square=True, annot=True, fmt='.2f', linecolor='white',
cmap='plasma')
plt.title('Correlation Heatmap of Rain in Australia Dataset')
plt.show()
```



Interpretation

From the above correlation heatmap, we can conclude that:-

- MinTemp and MaxTemp variables are highly positively correlated (correlation coefficient = 0.73).

- MinTemp and Temp3pm variables are also highly positively correlated (correlation coefficient = 0.70).

- MinTemp and Temp9am variables are strongly positively correlated (correlation coefficient = 0.90).

- MaxTemp and Temp9am variables are strongly positively correlated (correlation coefficient = 0.88).

- MaxTemp and Temp3pm variables are also strongly positively correlated (correlation coefficient = 0.97).

- WindGustSpeed and WindSpeed3pm variables are highly positively correlated (correlation coefficient = 0.66).

- Pressure9am and Pressure3pm variables are strongly positively correlated (correlation coefficient = 0.96).

- Temp9am and Temp3pm variables are strongly positively correlated (correlation coefficient = 0.85).

```
# features_to_be_dropped = ['Temp9am','Temp3pm','Pressure3pm']
# df.drop(features_to_be_dropped,inplace= True,axis=1)
```

LabelEncoding For Binary Features

```
# For binary features, we'll use labelencoding
le = LabelEncoder()

label_encoder_features = binary_categorical_features

for col in label_encoder_features:
    df[col] = le.fit_transform(df[col])

# let's check the head again
df.head()
```

```
  Location  MinTemp  MaxTemp  Rainfall  Evaporation  Sunshine WindGustDir  \
0    Delhi     13.4     22.9       0.6          3.2       9.9           W
1    Delhi      7.4     25.1       0.0          3.0      10.8         WNW
2    Delhi     12.9     25.7       0.0          8.0      10.1         WSW
3    Delhi      9.2     28.0       0.0         15.0       6.1          NE
4    Delhi     17.5     32.3       1.0          9.0       8.5           W

   WindGustSpeed WindDir9am WindDir3pm  WindSpeed9am  WindSpeed3pm  \
0           44.0          W        WNW          20.0          24.0
1           44.0        NNW        WSW           4.0          22.0
2           46.0          W        WSW          19.0          26.0
3           24.0         SE          E          11.0           9.0
4           41.0        ENE         NW           7.0          20.0

   Humidity9am  Humidity3pm  Pressure9am  Pressure3pm  Cloud9am  Cloud3pm  \
0         71.0         22.0       1007.7       1007.1       8.0       8.0
1         44.0         25.0       1010.6       1007.8       7.0       4.0
2         38.0         30.0       1007.6       1008.7       7.0       2.0
3         45.0         16.0       1017.6       1012.8       5.0       3.0
4         82.0         33.0       1010.8       1006.0       7.0       8.0

   Temp9am  Temp3pm  RainToday  RainTomorrow  year  month  day
0     16.9     21.8          0             0  2008     12    1
1     17.2     24.3          0             0  2008     12    2
2     21.0     23.2          0             0  2008     12    3
3     18.1     26.5          0             0  2008     12    4
4     17.8     29.7          0             0  2008     12    5
```

OneHotEncoding for Categorical Features

```
# creating list of categorical columns for one hot encoding
categorical_columns = [col for col in df.columns if df.dtypes[col] == 'object']
print('Categorical Features are : ',categorical_columns)
```

```
Categorical Features are :  ['Location', 'WindGustDir', 'WindDir9am', 'WindDir3pm']
```

```
# one hot encoding function for categorical features
def onehot_encoder(df, cols):
    data = df.copy()
```

```python
    for col in cols:
        dummies = pd.get_dummies(data[col],drop_first=True,prefix=col)
        # concatenating dummies and original dataframe
        data = pd.concat([data, dummies], axis=1)

        # dropping original columns for which encoding is applied.
        data.drop(col, axis=1,inplace=True)
    return data

# Apply onehotencoder on categorical features
df = onehot_encoder(df,categorical_columns)

# Dataframe shape after data preprocessing
df.shape
```

```
(145460, 114)
```

```python
# first five rows of dataframe
df.head()
```

```
   MinTemp  MaxTemp  Rainfall  Evaporation  Sunshine  WindGustSpeed  \
0     13.4     22.9       0.6          3.2       9.9           44.0
1      7.4     25.1       0.0          3.0      10.8           44.0
2     12.9     25.7       0.0          8.0      10.1           46.0
3      9.2     28.0       0.0         15.0       6.1           24.0
4     17.5     32.3       1.0          9.0       8.5           41.0

   WindSpeed9am  WindSpeed3pm  Humidity9am  Humidity3pm  Pressure9am  \
0          20.0          24.0         71.0         22.0       1007.7
1           4.0          22.0         44.0         25.0       1010.6
2          19.0          26.0         38.0         30.0       1007.6
3          11.0           9.0         45.0         16.0       1017.6
4           7.0          20.0         82.0         33.0       1010.8

   Pressure3pm  Cloud9am  Cloud3pm  Temp9am  Temp3pm  RainToday  RainTomorrow  \
0       1007.1       8.0       8.0     16.9     21.8          0             0
1       1007.8       7.0       4.0     17.2     24.3          0             0
2       1008.7       7.0       2.0     21.0     23.2          0             0
3       1012.8       5.0       3.0     18.1     26.5          0             0
4       1006.0       7.0       8.0     17.8     29.7          0             0

   year  month  day  Location_Albany  Location_Albury  Location_AliceSprings  \
0  2008     12    1                0                0                      0
1  2008     12    2                0                0                      0
2  2008     12    3                0                0                      0
3  2008     12    4                0                0                      0
4  2008     12    5                0                0                      0

   Location_BadgerysCreek  Location_Ballarat  Location_Bendigo  \
0                       0                  0                 0
1                       0                  0                 0
2                       0                  0                 0
3                       0                  0                 0
4                       0                  0                 0

   Location_Brisbane  Location_Cairns  Location_Canberra  Location_Cobar  \
```

```
0                  0                0                  0                0
1                  0                0                  0                0
2                  0                0                  0                0
3                  0                0                  0                0
4                  0                0                  0                0

   Location_CoffsHarbour  Location_Dartmoor  Location_Darwin  Location_Delhi  \
0                      0                  0                0               1
1                      0                  0                0               1
2                      0                  0                0               1
3                      0                  0                0               1
4                      0                  0                0               1

   Location_GoldCoast  Location_Hobart  Location_Katherine  \
0                   0                0                   0
1                   0                0                   0
2                   0                0                   0
3                   0                0                   0
4                   0                0                   0

   Location_Launceston  Location_Melbourne  Location_MelbourneAirport  \
0                    0                   0                          0
1                    0                   0                          0
2                    0                   0                          0
3                    0                   0                          0
4                    0                   0                          0

   Location_Moree  Location_MountGambier  Location_MountGinini  \
0               0                      0                     0
1               0                      0                     0
2               0                      0                     0
3               0                      0                     0
4               0                      0                     0

   Location_Newcastle  Location_Nhil  Location_NorahHead  \
0                   0              0                   0
1                   0              0                   0
2                   0              0                   0
3                   0              0                   0
4                   0              0                   0

   Location_NorfolkIsland  Location_Nuriootpa  Location_PearceRAAF  \
0                       0                   0                    0
1                       0                   0                    0
2                       0                   0                    0
3                       0                   0                    0
4                       0                   0                    0

   Location_Penrith  Location_Perth  Location_PerthAirport  Location_Portland  \
0                 0               0                      0                  0
1                 0               0                      0                  0
2                 0               0                      0                  0
3                 0               0                      0                  0
4                 0               0                      0                  0
```

```
   Location_Richmond  Location_Sale  Location_SalmonGums  Location_Sydney  \
0                  0              0                    0                0   
1                  0              0                    0                0   
2                  0              0                    0                0   
3                  0              0                    0                0   
4                  0              0                    0                0   

   Location_SydneyAirport  Location_Townsville  Location_Tuggeranong  \
0                       0                    0                     0   
1                       0                    0                     0   
2                       0                    0                     0   
3                       0                    0                     0   
4                       0                    0                     0   

   Location_Uluru  Location_WaggaWagga  Location_Walpole  Location_Watsonia  \
0               0                    0                 0                  0   
1               0                    0                 0                  0   
2               0                    0                 0                  0   
3               0                    0                 0                  0   
4               0                    0                 0                  0   

   Location_Williamtown  Location_Witchcliffe  Location_Wollongong  \
0                     0                     0                    0   
1                     0                     0                    0   
2                     0                     0                    0   
3                     0                     0                    0   
4                     0                     0                    0   

   Location_Woomera  WindGustDir_ENE  WindGustDir_ESE  WindGustDir_N  \
0                 0                0                0              0   
1                 0                0                0              0   
2                 0                0                0              0   
3                 0                0                0              0   
4                 0                0                0              0   

   WindGustDir_NE  WindGustDir_NNE  WindGustDir_NNW  WindGustDir_NW  \
0               0                0                0               0   
1               0                0                0               0   
2               0                0                0               0   
3               1                0                0               0   
4               0                0                0               0   

   WindGustDir_S  WindGustDir_SE  WindGustDir_SSE  WindGustDir_SSW  \
0              0               0                0                0   
1              0               0                0                0   
2              0               0                0                0   
3              0               0                0                0   
4              0               0                0                0   

   WindGustDir_SW  WindGustDir_W  WindGustDir_WNW  WindGustDir_WSW  \
0               0              1                0                0   
1               0              0                1                0   
2               0              0                0                1   
```

```
3               0              0                0                0
4               0              1                0                0

   WindDir9am_ENE  WindDir9am_ESE  WindDir9am_N  WindDir9am_NE  \
0               0               0             0              0
1               0               0             0              0
2               0               0             0              0
3               0               0             0              0
4               1               0             0              0

   WindDir9am_NNE  WindDir9am_NNW  WindDir9am_NW  WindDir9am_S  WindDir9am_SE  \
0               0               0              0             0              0
1               0               1              0             0              0
2               0               0              0             0              0
3               0               0              0             0              1
4               0               0              0             0              0

   WindDir9am_SSE  WindDir9am_SSW  WindDir9am_SW  WindDir9am_W  \
0               0               0              0             1
1               0               0              0             0
2               0               0              0             1
3               0               0              0             0
4               0               0              0             0

   WindDir9am_WNW  WindDir9am_WSW  WindDir3pm_ENE  WindDir3pm_ESE  \
0               0               0               0               0
1               0               0               0               0
2               0               0               0               0
3               0               0               0               0
4               0               0               0               0

   WindDir3pm_N  WindDir3pm_NE  WindDir3pm_NNE  WindDir3pm_NNW  WindDir3pm_NW  \
0             0              0               0               0              0
1             0              0               0               0              0
2             0              0               0               0              0
3             0              0               0               0              0
4             0              0               0               0              1

   WindDir3pm_S  WindDir3pm_SE  WindDir3pm_SSE  WindDir3pm_SSW  WindDir3pm_SW  \
0             0              0               0               0              0
1             0              0               0               0              0
2             0              0               0               0              0
3             0              0               0               0              0
4             0              0               0               0              0

   WindDir3pm_W  WindDir3pm_WNW  WindDir3pm_WSW
0             0               1               0
1             0               0               1
2             0               0               1
3             0               0               0
4             0               0               0
```

```
# Now,lets check missing values again
df.isnull().sum().sum()
```

```
0
```

```python
# splitting the data into X and y
X = df.drop('RainTomorrow', 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,
random_state=2022)

# Scaling Numerical Features - Imbalanced data

scaler = RobustScaler()

X_train = pd.DataFrame(scaler.fit_transform(X_train), columns=X_train.columns)
X_test = pd.DataFrame(scaler.transform(X_test), columns=X_test.columns)

# oversampling using SMOTE
sm = SMOTE(random_state=2022)
X_train_sm, y_train_sm = 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_sm)))
```

```
The number of classes before fit Counter({0: 90889, 1: 25479})
The number of classes after fit Counter({1: 90889, 0: 90889})
```

Model Training

```python
#Logistic Regression

model = LogisticRegression(max_iter=500)
model.fit(X_train, y_train)
predicted=model.predict(X_test)

conf = confusion_matrix(y_test, predicted)
print ("The accuracy of Logistic Regression is : ", accuracy_score(y_test,
predicted)*100, "%")
print()
print("F1 score for logistic regression is :",f1_score(y_test, predicted,)*100, "%")
```

```
The accuracy of Logistic Regression is :  84.55245428296439 %

F1 score for logistic regression is : 58.5730088495575 %
```

```python
#XGBoost
X_train=X_train.values
y_train = y_train.values
y_test=y_test.values
xgbc = XGBClassifier(objective='binary:logistic')
xgbc.fit(X_train,y_train)
predicted = xgbc.predict(X_test)
print ("The accuracy of XGBoost is : ", accuracy_score(y_test, predicted)*100, "%")
print("F1 score for XGBoost is :",f1_score(y_test, predicted,)*100, "%")
```

```
The accuracy of XGBoost is :  85.54929190155369 %
F1 score for XGBoost is : 62.34324614833393 %
```

```python
import pickle
```

```
pickle.dump(xgbc,open('model.pkl','wb'))
```

```
pickled_model=pickle.load(open('model.pkl','rb'))
pickled_model.predict(X_test)
```

```
array([0, 1, 1, ..., 0, 0, 1])
```

DEMO LINK

https://drive.google.com/drive/folders/17I57I0HCS0AL8GNgTmbeADgl6TAAZpNE