MAR ATHANASIUS COLLEGE OF ENGINEERING , KOTHAMANGALAM

Initial Project Report

Rainfall Prediction Using Machine Learning

Done by

DONA JINCE

Reg No: MAC23MCA-2024

Under the guidance of

Prof. Sonia Abraham

ABSTRACT

Topic : Rainfall Prediction Using Machine Learning

This project aims to develop a machine learning model to predict whether it will rain in the next 24 hours based on today's weather data. The project will utilize advanced machine learning algorithms to build and compare predictive models. Additionally, a user-friendly web interface will be created to allow users to input current weather parameters and receive immediate predictions regarding rainfall.

Accurate rainfall prediction is crucial for various sectors including agriculture, water resource management, and disaster preparedness. Traditional methods often fall short in capturing the intricate and nonlinear relationships between different meteorological factors. Machine learning models, however, can handle such complexities effectively, providing more accurate and reliable predictions.

Studies show that ANN, RF, and SVM are highly effective for rainfall prediction. Ghosh et al. highlighted Random Forest's accuracy and robustness. Hassan et al. found ANN to be the most accurate for nonlinear data. Sarasa-Cabezuelo's research confirmed the superior performance of neural networks for short-term forecasting. These findings justify using ANN, RF, and SVM in this project.

The project will implement and compare ANN, RF, and SVM algorithms to determine the best model for predicting rainfall within 24 hours. ANN will capture complex patterns, RF will enhance robustness with decision trees, and SVM will handle high-dimensional data. The models will be trained and validated using a comprehensive meteorological dataset and evaluated based on accuracy.

The dataset is from Kaggle's "weatheraus.csv" and includes features like temperature, humidity, wind speed, and historical rainfall records. Data preprocessing will involve cleaning, handling missing values, normalization, and feature selection to ensure quality input data.

Dataset Link : https://www.kaggle.com/datasets/trisha2094/weatheraus

Conclusion:

Three models were built using ANN, SVM and RF and they have the following training and testing accuracies:

1. **RF MODEL**
    - **Training accuracy:** 0.966508832118858
    - **Testing accuracy:** 0.8994261346882301
2. **ANN MODEL**
    - **Training accuracy:** 0.8817148203805041
    - **Testing accuracy:** 0.8789665887903464
3. **SVM MODEL**
    - **Training accuracy:** 0.8104624457739091
    - **Testing accuracy:** 0.8099439744142264

INTRODUCTION

Rainfall prediction is an essential aspect of meteorology with significant implications for agriculture, water resource management, and disaster preparedness. Accurate and timely forecasts of rainfall can help farmers plan their irrigation schedules, assist authorities in managing water resources efficiently, and provide early warnings for floods and other natural disasters. Traditional statistical methods for rainfall prediction often fall short in capturing the complex, nonlinear relationships among various meteorological variables. As a result, there is a growing interest in leveraging advanced machine learning techniques to improve the accuracy and reliability of rainfall predictions.

This project focuses on developing a machine learning model to predict whether it will rain in the next 24 hours based on current weather data. The primary objective is to build and compare predictive models using three sophisticated machine learning algorithms: Artificial Neural Networks (ANN), Support Vector Machines (SVM), and Random Forest (RF). These algorithms have been chosen due to their proven efficacy in handling complex data patterns and their high accuracy in previous studies.

The dataset for this project is sourced from Kaggle's "weatheraus.csv" dataset, which includes extensive historical meteorological data from various regions in Australia. The dataset comprises features such as temperature, humidity, wind speed, atmospheric pressure, and historical rainfall records. To ensure high-quality input data, preprocessing steps will involve cleaning the data, handling missing values, normalizing features, and selecting relevant features.

A significant component of this project is the development of a user-friendly web interface that allows users to input current weather parameters and receive immediate predictions on whether it will rain in the next 24 hours. This interface will make the model accessible to a broader audience, including farmers, policymakers, and the general public, enabling them to make informed decisions based on the predictions.

By integrating ANN, SVM, and RF, this project aims to enhance the accuracy and reliability of rainfall prediction models. The comprehensive evaluation and comparison of these algorithms will provide valuable insights into their performance, ultimately contributing to better decision-making in agriculture, water resource management, and disaster preparedness. The development of this advanced rainfall prediction model represents a significant step forward in leveraging machine learning for practical and impactful applications in meteorology.

LITERATURE REVIEW

Paper 1: Machine Learning-Based Rainfall Prediction: Unveiling Insights and Forecasting for Improved Preparedness

The paper "Machine Learning-Based Rainfall Prediction" by M. M. Hassan et al. explores the application of machine learning techniques to enhance the accuracy of rainfall predictions. The study uses a dataset comprising ten years of weather data from various Australian stations, including parameters such as temperature, humidity, wind speed, and historical rainfall records. Emphasizing the importance of data preprocessing, the authors include steps like handling missing values, outlier analysis, correlation analysis, and feature selection to improve the quality and reliability of the input data.

The methodology involves a comprehensive approach to model development and evaluation. Starting with data preprocessing, the authors apply multiple machine learning algorithms, comparing their performance to identify the most effective one for rainfall prediction. The key algorithms evaluated include **Naive Bayes, Decision Tree, Support Vector Machine (SVM), Random Forest, Logistic Regression**, and **Artificial Neural Network (ANN)**. Among these, the **ANN model stands out, achieving a maximum accuracy of 90% before feature selection and 91% after feature selection**. This improvement underscores the significance of feature selection in enhancing model performance. The ANN's superior accuracy is attributed to its ability to capture complex nonlinear relationships within the data.

The study by M. M. Hassan et al. significantly contributes to the field of rainfall prediction by systematically evaluating various machine learning algorithms and demonstrating the effectiveness of ANNs. The findings highlight the potential of machine learning techniques to improve weather forecasts, benefiting sectors reliant on precise rainfall predictions. While the study has some limitations, its comprehensive approach and practical proposals for future work provide a solid foundation for ongoing research and development in this area.

Title of the paper	M. M. Hassan *et al*., "Machine Learning-Based Rainfall Prediction: Unveiling Insights and Forecasting for Improved Preparedness," in *IEEE Access*, vol. 11, pp. 132196-132222, 2023, doi: 10.1109/ACCESS.2023.3333876.
Area of work	The goal is to improve the accuracy and reliability of rainfall predictions by applying various machine learning algorithms to historical weather data.
Dataset	Kaggle: weatheraus.csv :145461 instances, 22 features
Methodology/Strategy	Data Pre-processing => Dataset Summary => Visualization => Outlier Detection => Classification => Comparison =>
Algorithm	Logistic Regression, Decision Tree, Random Forest, Support Vector Machine (SVM), and Artificial Neural Networks (ANN).
Result/Accuracy	**ANN : 91%** LR:83% RF:81% SVM :84% LSTM :83% NB :81%
Advantages	Effective rainfall prediction model using a combination of machine learning algorithms and feature selection techniques
Future Proposal	The study proposes that the use of feature selection techniques can further improve the accuracy of machine learning models for rainfall prediction. It suggests integrating feature selection as a standard step in the modeling process to enhance performance.

Paper 2: Sarasa-Cabezuelo, A. Prediction of Rainfall in Australia Using Machine Learning *Information* 2022, *13*, 163. https://doi.org/10.3390/info13040163

The paper "Prediction of Rainfall in Australia Using Machine Learning" by Sarasa-Cabezuelo, A. (2022) investigates machine learning techniques to enhance rainfall forecasting accuracy. This review synthesizes the study's methodologies, algorithms, and findings, situating them within the broader context of contemporary rainfall prediction research.

The study employs meteorological data from various Australian cities, incorporating parameters such as temperature, humidity, wind speed, and historical rainfall records. Data preprocessing involves addressing missing values, normalization, and feature selection to ensure high-quality input data, essential for optimizing model performance.

Sarasa-Cabezuelo adopts a comparative approach, evaluating several machine learning algorithms such as **k-Nearest Neighbors (k-NN) ,Decision Trees ,Random Forest ,Neural Networks**. **Neural Networks achieved the highest accuracy** for short-term rainfall forecasts. Their ability to capture complex nonlinear relationships within the data led to more precise predictions compared to k-NN, Decision Trees, and Random Forest.

The study's main advantage is its comprehensive comparison of multiple machine learning models, offering valuable insights into their efficacy for rainfall prediction. Neural Networks' superior accuracy (**0.84)** underscores their potential for weather forecasting.

Sarasa-Cabezuelo suggests incorporating additional meteorological variables, such as atmospheric pressure and solar radiation, to enhance model accuracy. Developing a real-time prediction system leveraging cloud computing could address computational challenges. Exploring advanced techniques like deep learning models and hybrid approaches combining multiple algorithms may further improve accuracy and robustness.

Title of the paper	Sarasa-Cabezuelo, A. Prediction of Rainfall in Australia Using Machine Learning. *Information* **2022**, *13*, 163. https://doi.org/10.3390/info13040163
Area of work	Rainfall prediction using machine learning techniques
Dataset	Kaggle: weatheraus .csv :145461 instances, 22 features
Methodology/Strategy	The study applies various machine learning algorithms to historical weather data to predict rainfall. It focuses on comparing the effectiveness of different models.
Algorithm	**Neural Networks(0.84)**,k-Nearest Neighbors(0.83),Decision Tree(0.83), Random Forest(0.83)
Result/Accuracy	Neural Networks provided the best prediction accuracy among the tested models, demonstrating the potential of machine learning in improving rainfall predictions.
Advantages	High prediction accuracy,Ability to handle complex datasets
Limitations	Computationally intensive, Requires extensive training data
Future Proposal	The study suggests further exploration of additional meteorological variables to enhance prediction accuracy, and developing an integrated system for real-time rainfall prediction.

Paper 3: Ghosh, S., Gourisaria, M.K., Sahoo, B. *et al.* A pragmatic ensemble learning approach for rainfall prediction. *Discov Internet Things* 3, 13 (2023). https://doi.org/10.1007/s43926-023-00044-3

The paper "A Pragmatic Ensemble Learning Approach for Rainfall Prediction" by Soumili Ghosh et al. investigates the use of ensemble learning techniques to enhance the accuracy and robustness of rainfall predictions. The study utilizes meteorological data from various Australian cities encompassing parameters such as temperature, humidity, wind speed, and historical rainfall records. Preprocessing steps include handling missing values, normalizing the data, and selecting relevant features to ensure high-quality input data. This preprocessing is essential for improving the performance of the machine learning models.

The authors adopt an ensemble learning approach, combining multiple machine learning models to enhance predictive performance. Ensemble methods leverage the strengths of individual models, reducing overfitting and improving generalization. The key ensemble algorithms evaluated include: **Random Forest**, **Gradient Boosting**, **AdaBoost**.Ensemble methods generally outperform individual models in terms of accuracy and robustness. The study finds that ensemble techniques, particularly Gradient Boosting and Random Forest, provide more accurate and reliable rainfall predictions compared to single models but concluded that **ANN outperformed these methods in terms of prediction accuracy.**

In summary, the study shows that combining multiple machine learning models leads to more accurate and robust predictions compared to individual models. Despite the computational challenges, the ensemble approach offers a promising direction for future research and practical applications in weather forecasting.

Title of the paper	Ghosh, S., Gourisaria, M.K., Sahoo, B. *et al.* A pragmatic ensemble learning approach for rainfall prediction. *Discov Internet Things* **3**, 13 (2023). https://doi.org/10.1007/s43926-023-00044-3
Area of Work	This research focuses on using ensemble learning techniques for predicting rainfall.
Dataset	Kaggle: weatheraus .csv :145461 instances, 22 features
Methodology/Strategy	The study applies ensemble learning techniques to improve the accuracy of rainfall predictions. It compares the performance of Bagging and Boosting methods.
Algorithm	• Bagging (Bootstrap Aggregating) • Boosting (combining weak learners to form a strong learner)
Result/Accuracy	The Boosting technique significantly reduced bias by 6% and variance by 13.6%, and decreased the false negative rate by more than 20%.
Advantages	• Improved prediction accuracy • Enhanced robustness against data variability
Limitations	• Computational complexity • Requires large datasets for training
Future Proposal	The study proposes further exploration of ensemble methods to optimize model performance, and integrating these techniques into real-time weather forecasting systems.

LITERATURE SUMMARY

PAPER	TITLE	DATASET	ALGORITHM	ACCURACY
PAPER 1	M. M. Hassan *et al.*, "Machine Learning-Based Rainfall Prediction: Unveiling Insights and Forecasting for Improved Preparedness," in *IEEE Access*, vol. 11, pp. 132196-132222, 2023, doi: 10.1109/ACCESS.2023.3333876.	Kaggle: weatheraus .csv :145461 instances, 22 features	Artificial Neural Network Support Vector Machine Logistic Regression Lons Short Term Memory Random Forest Naïve Bayes	**91%** **84%** **83%** **83%** **81%** **81%**
PAPER 2	Sarasa-Cabezuelo, A. Prediction of Rainfall in Australia Using Machine Learning. *Information* **2022**, *13*, 163. https://doi.org/10.3390/info13040163	Kaggle: weatheraus .csv :145461 instances, 22 features	Neural Networks Random Forest K Nearest Neighbour Decision Trees	**84%** **83%** **83%** **83%**
PAPER 3	Ghosh, S., Gourisaria, M.K., Sahoo, B. *et al.* A pragmatic ensemble learning approach for rainfall prediction. *Discov Internet Things* **3**, 13 (2023). https://doi.org/10.1007/s43926-023-00044-3	Kaggle: weatheraus .csv :145461 instances, 22 features	Random Forest Decision Trees Logistic Regression K Nearest Neighbour	**87%** **82%** **79%** **78%**

PROJECT PROPOSAL

Based on the findings from the previous studies, ANN,SVM and RF achieved the highest accuracy in rainfall prediction.This project aims to develop machine learning models to predict whether it will rains in 24 hours based on today's weather data. The project will utilize Artificial Neural Networks (ANN), Support Vector Machines (SVM), and Random Forest (RF) algorithms to build predictive models. Additionally, a web interface will be created to allow users to input weather parameters and receive predictions.

Objectives:

- Develop a machine learning model to predict rainfall using ANN, SVM, and RF algorithms.
- Compare the performance of these models to determine the most accurate one.
- Create a user-friendly web interface for users to input weather data and obtain rainfall predictions.

Motivation for Choosing ANN,SVM,RF

1. Handling Complex Relationships: All three algorithms—Artificial Neural Networks (ANN), Support Vector Machines (SVM), and Random Forest (RF)—are capable of modeling complex relationships within the data.Rainfall prediction involves multiple meteorological factors with intricate interactions, requiring robust algorithms to capture these complexities effectively.

2. High Accuracy: Each of these algorithms has demonstrated high accuracy in various machine learning tasks, including rainfall prediction.Accurate rainfall prediction is crucial for applications in agriculture, disaster management, and water resource planning. High accuracy models ensure better decision-making and resource allocation.

3. Robustness to Noise and Variability: ANN, SVM, and RF are all known for their robustness in handling noisy and variable datasets.Meteorological data often contains noise and variability due to natural fluctuations and measurement errors. Robust algorithms are essential to provide reliable predictions despite these challenges.

Methodology

The project will follow a structured methodology encompassing data preprocessing, model development, training, evaluation, and deployment.

1. Data Preprocessing:

- **Handling Missing Values**: Imputation techniques will be applied to address missing data points.
- **Normalization:** Features will be normalized to ensure uniformity and improve model performance.
- **Feature Selection:** Relevant features will be selected based on correlation analysis to enhance prediction accuracy.

2. Model Development : Using ANN,SVM,RF

3. Model Training : The dataset will be split into training and testing sets (80% and 20% respectively).

4. Model Evaluation : Models will be evaluated using Accuracy

5. Model Deployment

- Web Interface Development: A user-friendly web interface will be created using Flask. This interface will allow users to input meteorological parameters and receive rainfall predictions.
- Backend Integration: The trained model will be integrated with the web application backend to process user inputs and generate predictions.

DATASET

The dataset used in this project is the "WeatherAUS" dataset from Kaggle, which provides comprehensive weather data (145461 instances and 22 features) from various locations in Australia spanning several years.. This dataset is pivotal in developing robust rainfall prediction models using Machine Learning Algorithms .The dataset includes various meteorological parameters recorded daily, providing a rich source of information for developing predictive models.

Features

1. **Date**: The day on which the measurement is carried out.
2. **Location**: The name of the weather station where the data was recorded.
3. **MinTemp**: Minimum temperature recorded in degrees Celsius.
4. **MaxTemp**: Maximum temperature recorded in degrees Celsius.
5. **Rainfall**: The amount of rain recorded during the day in millimeters.
6. **Evaporation**: The "Class A pan evaporation" (in millimeters) in 24 hours until 9 a.m.
7. **Sunshine**: The number of hours of radiant sun during the day.
8. **WindGustDir**: The direction of the strongest wind gust in the 24 hours to midnight.
9. **WindGustSpeed**: The speed (km/h) of the strongest wind gust in the 24 hours to midnight.
10. **WindDir9am**: Wind direction at 9 a.m.
11. **WindDir3pm**: Wind direction at 3 p.m.
12. **WindSpeed9am**: Average wind speed (km/h) in the 10 minutes before 9 a.m.
13. **WindSpeed3pm**: Average wind speed (km/h) in the 10 minutes before 3 p.m.
14. **Humidity9am**: Humidity (%) at 9 a.m.
15. **Humidity3pm**: Humidity (%) at 3 p.m.
16. **Pressure9am**: Atmospheric pressure (hPa) at the level of the station at 9 a.m.
17. **Pressure3pm**: Atmospheric pressure (hPa) at the level of the station at 3 p.m.
18. **Cloud9am**: Fraction of sky obscured by clouds at 9 a.m. (measured in "oktas").
19. **Cloud3pm**: Fraction of sky obscured by clouds at 3 p.m. (measured in "oktas").
20. **Temp9am**: Temperature (degrees Celsius) at 9 a.m.
21. **Temp3pm**: Temperature (degrees Celsius) at 3 p.m.
22. **RainToday**: Boolean indicator of whether rainfall was recorded today (1 if yes, 0 if no).

Class Label

- **RainTomorrow**: A boolean indicator of whether rainfall was recorded the following day (1 if yes, 0 if no).

The detailed review of the "WeatherAUS" dataset highlights the importance of each feature in developing an accurate rainfall prediction model. The ANN-based model, coupled with a user-friendly web interface, aims to provide reliable and accessible rainfall predictions, supporting various sectors dependent on weather forecasts. This project demonstrates the potential of machine learning in enhancing weather prediction accuracy and usability.

EXPLORATORY ANALYSIS

```
[4]  df.shape

     (145460, 23)
```

Figure 1

Dataset has 145460 instances and 23 columns.

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 145460 entries, 0 to 145459
Data columns (total 23 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   Date           145460 non-null  object
 1   Location       145460 non-null  object
 2   MinTemp        143975 non-null  float64
 3   MaxTemp        144199 non-null  float64
 4   Rainfall       142199 non-null  float64
 5   Evaporation    82670 non-null   float64
 6   Sunshine       75625 non-null   float64
 7   WindGustDir    135134 non-null  object
 8   WindGustSpeed  135197 non-null  float64
 9   WindDir9am     134894 non-null  object
 10  WindDir3pm     141232 non-null  object
 11  WindSpeed9am   143693 non-null  float64
 12  WindSpeed3pm   142398 non-null  float64
 13  Humidity9am    142806 non-null  float64
 14  Humidity3pm    140953 non-null  float64
 15  Pressure9am    130395 non-null  float64
 16  Pressure3pm    130432 non-null  float64
 17  Cloud9am       89572 non-null   float64
 18  Cloud3pm       86102 non-null   float64
 19  Temp9am        143693 non-null  float64
 20  Temp3pm        141851 non-null  float64
 21  RainToday      142199 non-null  object
 22  RainTomorrow   142193 non-null  object
dtypes: float64(16), object(7)
memory usage: 25.5+ MB
```

Figure 2

Dataset contains 16 numerical attributes and 7 non numerical attributes.

```
print(df['RainTomorrow'].value_counts())

RainTomorrow
No     110316
Yes     31877
Name: count, dtype: int64
```

Figure 3

Target variable is **RainTomorrow**. It has two values: **Yes**: it will rains in 24 hours; **No**: it will not rains in 24 hours

Number of rows with value Yes is 31877

Number of rows with value No is 110316

DATA PREPROCESSING

1. Identification of Missing Values

Count and percentage of missing values and the datatypes of each feature in the original dataset is given

	missing_values	percent_missing %	data type
Date	0	0.000000	object
Location	0	0.000000	object
MinTemp	1485	1.020899	float64
MaxTemp	1261	0.866905	float64
Rainfall	3261	2.241853	float64
Evaporation	62790	43.166506	float64
Sunshine	69835	48.009762	float64
WindGustDir	10326	7.098859	object
WindGustSpeed	10263	7.055548	float64
WindDir9am	10566	7.263853	object
WindDir3pm	4228	2.906641	object
WindSpeed9am	1767	1.214767	float64
WindSpeed3pm	3062	2.105046	float64
Humidity9am	2654	1.824557	float64
Humidity3pm	4507	3.098446	float64
Pressure9am	15065	10.356799	float64
Pressure3pm	15028	10.331363	float64
Cloud9am	55888	38.421559	float64
Cloud3pm	59358	40.807095	float64
Temp9am	1767	1.214767	float64
Temp3pm	3609	2.481094	float64
RainToday	3261	2.241853	object
RainTomorrow	3267	2.245978	object

Figure 4

2. Remove rows with target variable null

⇨ df.dropna(subset=['RainTomorrow'], inplace=True)

3. Missing Data Pattern in dataframe

X-label: names of the columns (features) in dataframe.

Y-label: numerical indices representing the rows (data instances) in dataframe.

Legend: Since cbar=False, there is no explicit legend. However, the color intensity within the heatmap represents the presence (darker shade) or absence (lighter shade) of missing values.

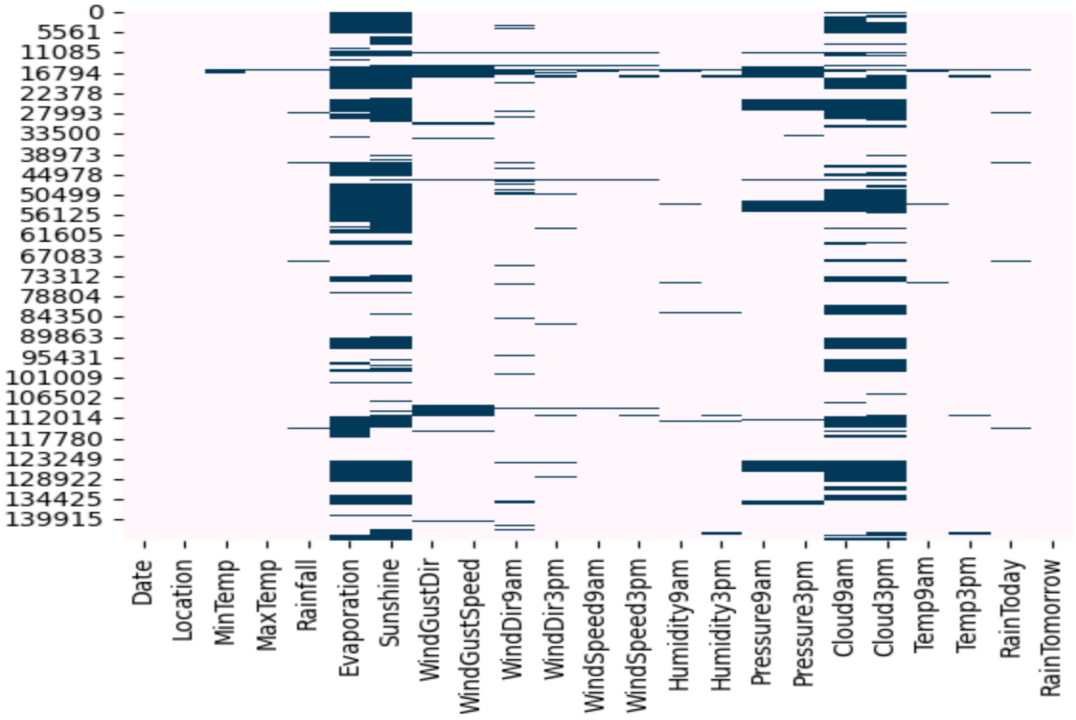


Figure 5

From Figure 5, “Evaporation”, “Sunshine”, “Cloud9am”, “Cloud3pm” are the features with a high missing percentage. So we will check the details of the missing data for these 4 features.

	Total	Percent
Sunshine	67816	0.476929
Evaporation	60843	0.427890
Cloud3pm	57094	0.401525
Cloud9am	53657	0.377353

Figure 6

Figure 6 shows the count and percentage of the 4 features in the dataframe,which has less than 50 percent missing data. So instead of rejecting them completely, we'll consider them in our model with proper imputation.

4. Imputation and Transformation

Impute the categorical columns with mode, and then by using the label encoder , converting them to numeric numbers. Once all the columns in the full data frame are converted to numeric columns, then impute the other missing values using the Multiple Imputation by Chained Equations (MICE) package.

```
# Impute categorical var with Mode
df['Date'] = df['Date'].fillna(df['Date'].mode()[0])
df['Location'] = df['Location'].fillna(df['Location'].mode()[0])
df['WindGustDir'] = df['WindGustDir'].fillna(df['WindGustDir'].mode()[0])
df['WindDir9am'] = df['WindDir9am'].fillna(df['WindDir9am'].mode()[0])
df['WindDir3pm'] = df['WindDir3pm'].fillna(df['WindDir3pm'].mode()[0])
```

Figure 7

Figure 7 shows the replacing of all null values in the non numerical features with the mode.

```
# Convert categorical features to continuous features with Label Encoding
lencoders = {}
for col in df.select_dtypes(include=['object']).columns:
    lencoders[col] = LabelEncoder()
    df[col] = lencoders[col].fit_transform(df[col])
```

Figure 8

Figure 8 shows label Encoding; categorical features are converted to continuous features.

```
warnings.filterwarnings("ignore")
# Multiple Imputation by Chained Equations
MiceImputed = df.copy(deep=True)
mice_imputer = IterativeImputer()
MiceImputed.iloc[:, :] = mice_imputer.fit_transform(df)
```

Figure 9

Figure 9 shows Multiple Imputation by Chained Equations .After these process there will be no missing values in the dataset.

5. Check whether dataset is balanced or not

Kind of Plot: Bar plot (kind='bar')
X-label: the x-label represents the two classes of 'RainTomorrow': 'No' (represented as 0) and 'Yes' (represented as 1).
Y-label: Ithe y-label represents the normalized count or proportion of each class.
Legend: No explicit legend is used, but the colors of the bars (skyblue for 'No' and navy for 'Yes') differentiate the classes.

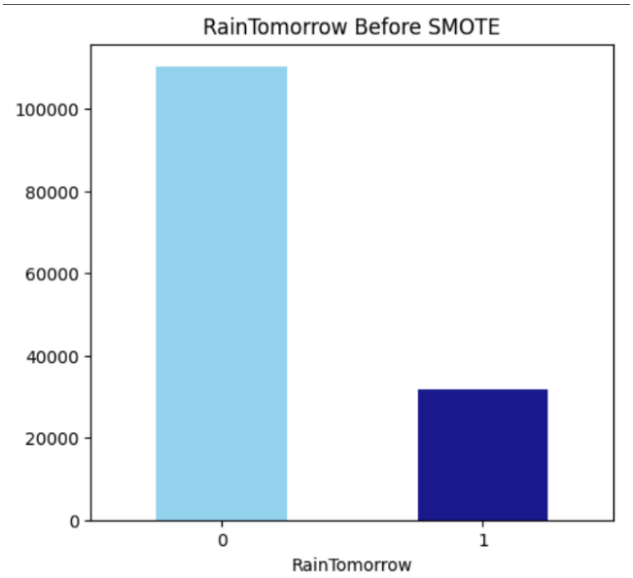


Figure 10

Dataset contains value No more than Yes for target variable RainTomorrow, so it is imbalanced and has to be balanced. Used smote for balancing the dataset.

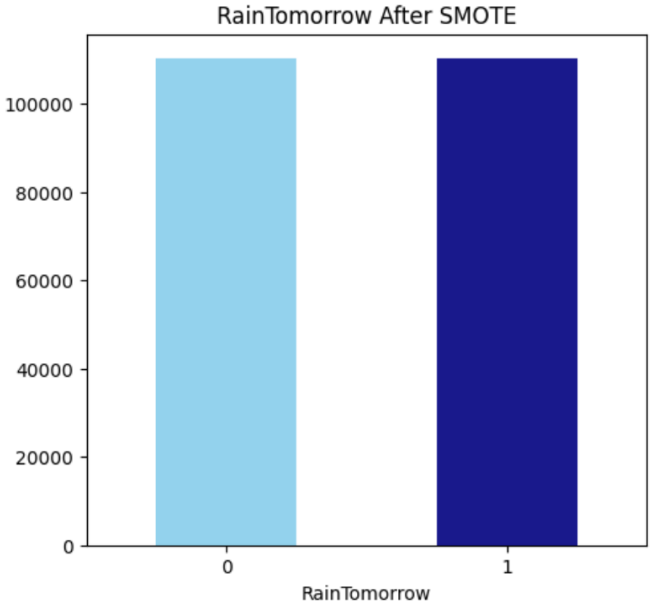


Figure 11

After smote , the number of instances became 220632.

DATASET VISUALIZATION

1. Detecting Outliers

Xlabel: Numerical Features (implicitly defined by the boxplot function based on DataFrame columns).
Ylabel: No explicit ylabel is set, but it represents the range of values for each feature.
Legend: No legend is present in this plot.
Type: boxplot

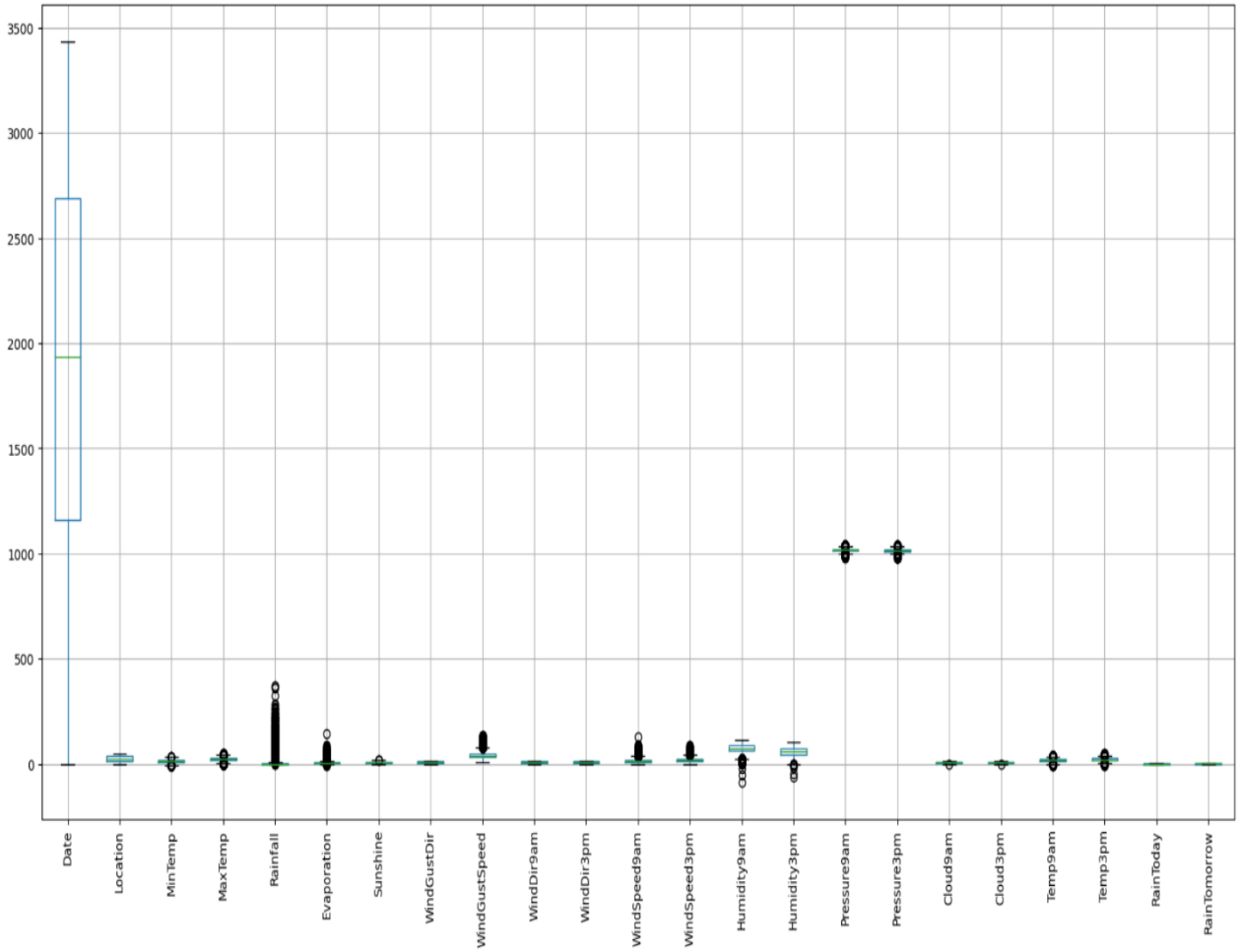


Figure 12

No severe outliers detected.

2. Correlation between the different variables

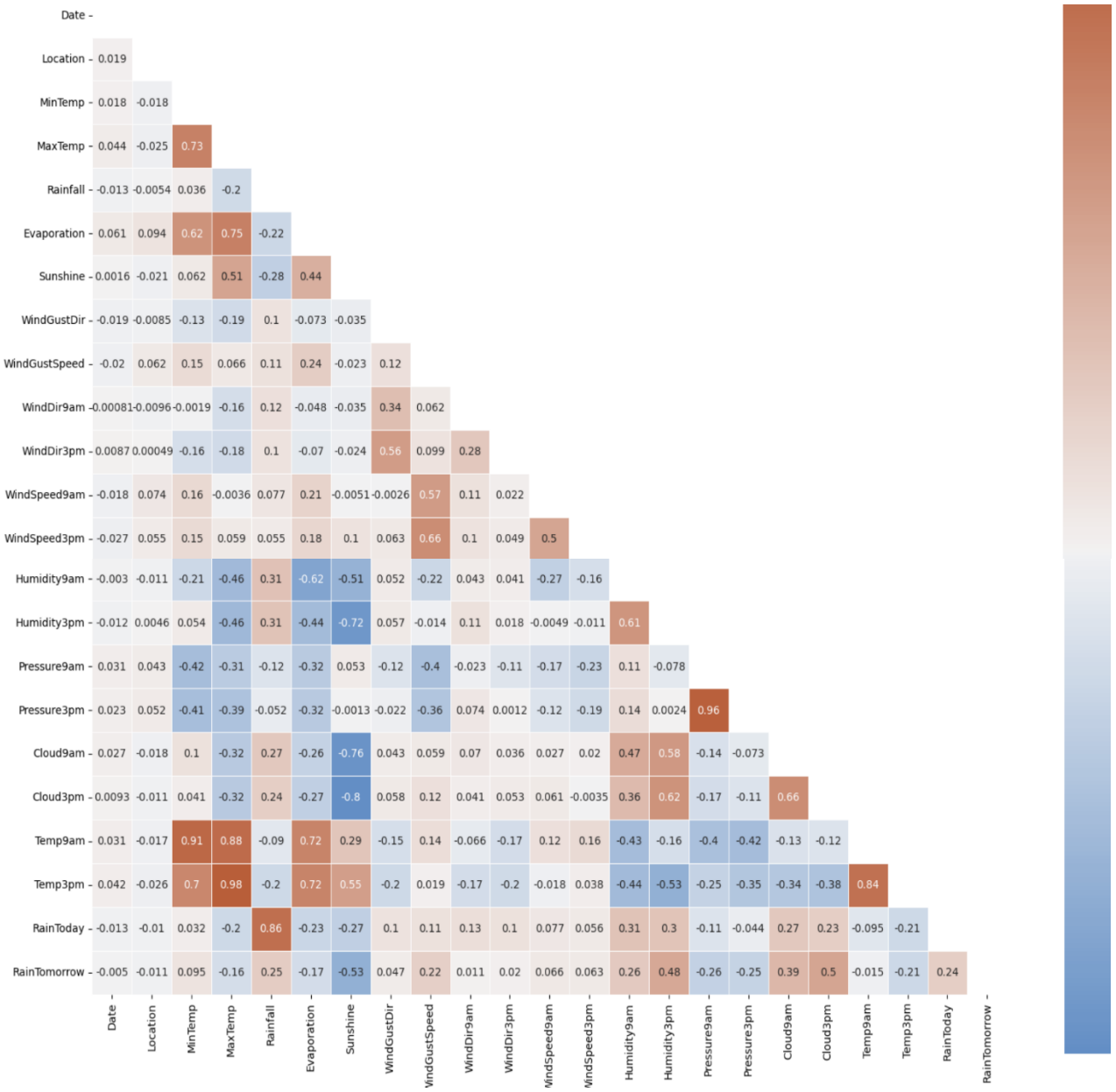


Figure 13

Type: Heatmap.
X-label: names of the columns (features) in the preprocessed DataFrame
Y-label: names of the columns (features) in the preprocessed DataFrame
Legend: While there's no explicit legend, the color intensity and annotations on the heatmap convey the information:
Color Intensity: Represents the strength of the correlation. Darker colors typically indicate stronger positive correlations, lighter colors indicate weaker correlations, and colors in the middle (around zero) indicate no or weak correlations
Annotations: The numbers within each cell of the heatmap represent the actual correlation coefficient between the corresponding pair of features.
This heatmap visualizes the pairwise correlations between all numerical features in the dataset, helping identify potential multicollinearity (high correlations between independent variables) and understand relationships between features.

In this case, the following feature pairs have a strong correlation with each other:

- MaxTemp and Temp3pm
- Pressure3pm and Pressure9am
- Temp9am and MinTemp
- Temp9am and MaxTemp
- RainToday and RainFall

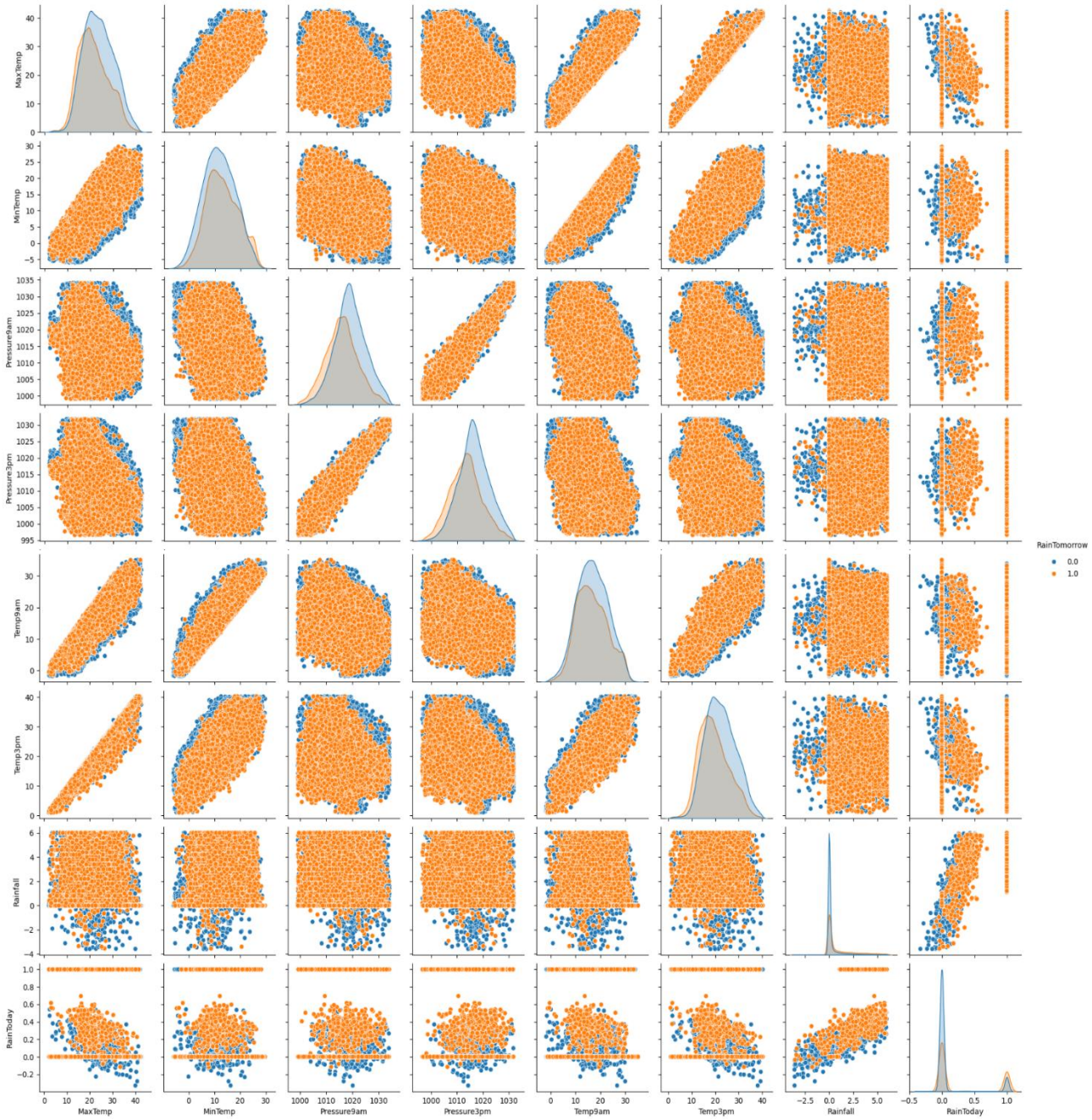


Figure 14

Type: Scatterplot
X-label: The range of values of the feature being plotted.
Y-label: The range of values of the feature being plotted.
Legend: if yellow it rains, else no rain

Each of the paired plots shows very clear distinct clusters of RainTomorrow's "yes" and "no" clusters. There is very minimal overlap between them.

Feature Selection for Rainfall Prediction

Filter method is used for feature selection to train the rainfall prediction models. Features are selected by filtering method (chi-square value). Before doing this, it is important to normalize our dataset. Using MinMaxScaler instead of StandardScaler in order to avoid negative values.

- **Standardizing data**

```
# Standardizing data
r_scaler = preprocessing.MinMaxScaler()
r_scaler.fit(modified_data_remaining)
modified_data = pd.DataFrame(r_scaler.transform(modified_data_remaining), index=modified_data_remaining.index,
                             columns=modified_data_remaining.columns)
modified_data.head()
```

Figure 15

- **Feature Importance using Filter Method (Chi-Square)**

```
# Ensure 'RainTomorrow' is excluded from the feature selection process
# Separate features (X) and target variable (y)
X = modified_data.drop('RainTomorrow', axis=1)  # Ensure 'RainTomorrow' is excluded
y = modified_data['RainTomorrow']
# Select the 10 best features from the feature set (excluding 'RainTomorrow')
selector = SelectKBest(chi2, k=10)
X_selected = selector.fit_transform(X, y)

# Get the selected feature names (without the target variable)
selected_columns = X.columns[selector.get_support()]  # Use X.columns here
print("Selected columns:", selected_columns)

# Create a new dataframe with the selected features
modified_data_selected = pd.DataFrame(X_selected, columns=selected_columns)
modified_data_selected['RainTomorrow'] = y.values
```

Figure 16

```
Selected columns: Index(['Rainfall', 'Sunshine', 'WindGustSpeed', 'Humidity3pm', 'Pressure9am',
       'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp3pm', 'RainToday'],
      dtype='object')
```

Figure 17

Figure 17 shows the 10 best features selected using Chi –Square.

WORKING OF ALGORITHMS

Algorithms used in ‘Rainfall Prediction Using Machine Learning’ are Artificial Neural Networks,Support Vector Machine and Random Forest .

1. Artificial Neural Networks (ANN)

Artificial Neural Networks (ANNs) are computational models inspired by the human brain's neural networks. They consist of interconnected nodes (neurons) arranged in layers: an input layer, one or more hidden layers, and an output layer. Each neuron processes input data through weighted connections and activation functions to produce output. ANNs are particularly powerful in capturing complex patterns in data, making them suitable for a wide range of tasks, including classification, regression, and more.

Algorithm:

1. **Initialization**:
    - Initialize the weights and biases for all layers of the network randomly.
2. **Forward Propagation**:
    - For each input sample, calculate the output of each neuron by computing the weighted sum of inputs and applying the activation function.
    - Pass the output from the input layer through the hidden layers and finally to the output layer.
3. **Loss Calculation**:
    - Compute the loss by comparing the predicted output with the actual target using a loss function (e.g., Mean Squared Error for regression or Cross-Entropy for classification).
4. **Backpropagation**:
    - Calculate the gradient of the loss function with respect to each weight and bias using the chain rule.
    - Update the weights and biases by moving them in the direction that reduces the loss, typically using an optimization algorithm like Gradient Descent.
5. **Iteration**:
    - Repeat the forward propagation, loss calculation, and backpropagation steps for multiple epochs until the model converges (i.e., the loss stops decreasing significantly).
6. **Prediction**:
    - After training, use the trained network to make predictions on new data.

Pseudocode:

```
Initialize weights and biases
For each epoch:
   For each input sample:
      Perform forward propagation through all layers
      Calculate the loss
      Perform backpropagation to update weights and biases
   End For
End For
Use the trained model to make predictions
```

2. Support Vector Machines (SVM)

Support Vector Machines (SVMs) are supervised learning models used for classification and regression tasks. SVMs work by finding the optimal hyperplane that best separates data points of different classes in a feature space. The data points that are closest to the hyperplane are called support vectors. SVMs are effective in high-dimensional spaces and are especially useful for binary classification.

Algorithm:

1. **Data Preparation**:
    - Represent the input data as feature vectors.
2. **Hyperplane Selection**:
    - Find the hyperplane that maximizes the margin between the two classes. The margin is the distance between the hyperplane and the nearest data point from either class.
3. **Optimization**:
    - Use optimization techniques (like Quadratic Programming) to minimize the classification error and maximize the margin.
4. **Support Vectors**:
    - Identify the support vectors, which are the data points closest to the hyperplane.
5. **Kernel Trick (Optional)**:
    - If the data is not linearly separable, apply a kernel function (e.g., RBF, Polynomial) to transform the data into a higher-dimensional space where a linear separation is possible.
6. **Prediction**:
    - Use the optimal hyperplane (or decision boundary) to classify new data points.

Pseudocode:

Transform the input data using a kernel function (if needed)
Find the optimal hyperplane that maximizes the margin
Identify support vectors
For each new input sample:
  Calculate the decision function based on the hyperplane
  Assign the sample to a class based on the sign of the decision function

3. Random Forest (RF)

Random Forest is an ensemble learning method that builds multiple decision trees and merges their outputs to make a more accurate and stable prediction. Each tree is built using a random subset of features and data samples, making the model less prone to overfitting and more robust. Random Forest is widely used for both classification and regression tasks.

Algorithm:

- **Bootstrap Sampling**:
    - Randomly select samples from the training data (with replacement) to create multiple datasets (bootstrap samples).
- **Tree Construction**:
    - For each bootstrap sample, construct a decision tree by recursively splitting the data based on the best feature that maximizes the separation of classes (or minimizes the error in regression).
    - During the construction, at each split, randomly select a subset of features to choose the best split from.
- **Aggregation**:
    - After all trees are built, aggregate their predictions:
        - For classification, use majority voting (i.e., the class that gets the most votes is the final prediction).
        - For regression, average the predictions from all trees.
- **Prediction**:
    - Use the aggregated output from all trees to make the final prediction for new data.

Pseudocode:

For each tree in the Random Forest:
  Bootstrap sample the data
  Randomly select a subset of features for each split
  Build the decision tree using the selected features and bootstrap sample
  Repeat until the maximum depth or minimum sample size is reached
End For

For each new input sample
  Pass the sample through each decision tree in the forest
  Record the output (class or regression value) from each tree
  Aggregate the outputs:
    - For classification: Use majority voting to determine the final class label
    - For regression: Compute the average of the outputs to get the final prediction
  Return the aggregated result as the final prediction

PACKAGES AND FUNCTIONS

Packages

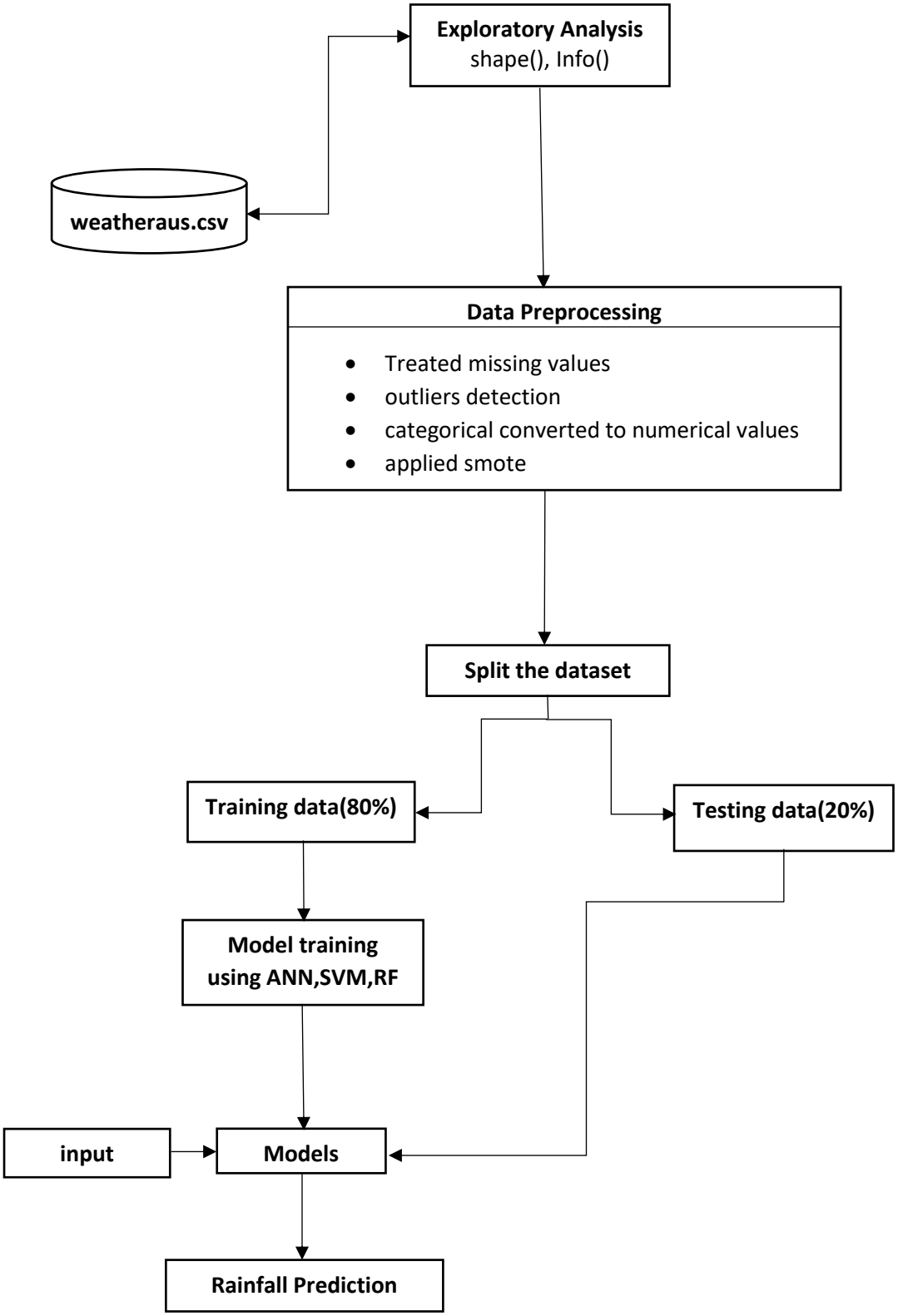
1. **pandas as pd**
2. **numpy as np**
3. **seaborn as sns**
4. **matplotlib.pyplot as plt**
5. **warnings**
6. **imblearn**
   - smote
7. **sklearn:**
   - preprocessing
   - feature_selection
     - SelectKBest
     - chi2
     - SelectFromModel
   - ensemble
     - RandomForestClassifier
   - experimental
     - enable_iterative_imputer
   - impute
     - IterativeImputer
   - preprocessing
     - LabelEncoder
   - model_selection
     - train_test_split
   - neural_network
     - MLPClassifier
   - svm
     - SVC
   - metrics
     - accuracy_score

Functions

- read_csv(): Reads data from a CSV file into a Pandas DataFrame.
- shape: Returns the dimensions (rows, columns) of a DataFrame.
- info(): Prints a summary of a DataFrame, including data types and non-null values.
- replace(): Replaces values in a DataFrame.
- value_counts(): Counts the occurrences of unique values in a Series.
- plot(): Creates various types of plots (bar plot in this case).
- title(): Sets the title of a plot.
- show(): Displays the plot.
- concat(): Concatenates DataFrames along an axis.
- isnull(): Detects missing values in a DataFrame.
- sum(): Calculates the sum of values.
- sort_values(): Sorts a DataFrame by values in one or more columns.
- head(): Returns the first few rows of a DataFrame.
- select_dtypes(): Selects columns based on their data types.
- fillna(): Fills missing values.
- mode(): Returns the most frequent value(s) in a Series.
- LabelEncoder(): Creates a LabelEncoder object for converting categorical labels to numerical values.
- fit_transform(): Fits a transformer (like LabelEncoder) to data and then transforms it.
- filterwarnings(): Controls the display of warning messages.

- copy(): Creates a deep copy of a DataFrame.
- IterativeImputer(): Creates an IterativeImputer object for multiple imputation.
- quantile(): Computes quantiles of a DataFrame.
- any(): Returns True if any element in a boolean array is True.
- corr(): Computes the pairwise correlation of columns in a DataFrame.
- triu(): Returns the upper triangle of an array.
- subplots(): Creates a figure and a set of subplots.
- heatmap(): Generates a heatmap plot.
- diverging_palette(): Creates a diverging color palette.
- pairplot(): Generates a pair plot to visualize relationships between variables.
- MinMaxScaler(): Creates a MinMaxScaler object for scaling numerical features.
- transform(): Transforms data using a fitted scaler or transformer.
- loc[]: Accesses a group of rows and columns by labels.
- SelectKBest(): Selects features based on the k highest scores.
- fit(): Fits a feature selector to data.
- transform(): Transforms data using a fitted feature selector.
- get_support(): Returns a boolean mask indicating selected features.
- drop(): Removes rows or columns from a DataFrame.
- SelectFromModel(): Selects features based on importance weights from a model.
- feature_importances_: Returns feature importance scores from a fitted model.
- train_test_split(): Splits data into training and testing sets.
- MLPClassifier(): Creates a Multi-Layer Perceptron (neural network) classifier.
- SVC(): Creates a Support Vector Classifier.
- predict(): Predicts target values using a trained model.
- accuracy_score(): Calculates the accuracy score of model predictions.

PROJECT PIPELINE



TIMELINE

- Submission of project synopsis with Journal Papers - 22.07.2024
- Project proposal approval - 26.07.2024
- Presenting project proposal before the Approval Committee - 30.07.2024
- Initial report submission - 12.08.2024
- Analysis and design report submission - 16.08.2024
- First Project Presentation - 23.08.2024
- ANN model Release - 29.08.2024
- SVM and RF model Release - 04.10.2024
- Interim Project Presentation - 08.10.2024
- Web Application Release - 22.10.2024
- Submission of the project report to the guide - 28.10.2024
- Final project presentation - 28.10.2024 & 29.10.2024
- Submission of project report after corrections - 01.11.2024

SYSTEM DESIGN

Model Building

Dataset is split in to two sets: 80% of datapoints for training and 20% for testing. train_test_split function is used to divide the data into training and testing sets.

```
# Split the  data into train and test sets
# Create a new dataframe with the selected features
X_new = modified_data_selected.drop(columns=['RainTomorrow'])
y_remaining =modified_data_selected['RainTomorrow']
X_train, X_test, y_train, y_test = train_test_split(X_new, y_remaining, test_size=0.2, random_state=42)
```

Figure 18

1.Training and testing of ANN model

```
# 1. Artificial Neural Network (ANN)
ann_model = MLPClassifier(random_state=42, verbose=1, hidden_layer_sizes=(100, 50), activation='relu',
                          solver='adam')
ann_model.fit(X_train, y_train)
ann_predictions = ann_model.predict(X_test)
ann_accuracy = accuracy_score(y_test, ann_predictions)
print("ANN Accuracy:", ann_accuracy)
```

Figure 19

Creates an ANN model with the following parameters:

- random_state=42: Ensures consistent results for reproducibility.
- verbose=1: Prints progress messages during training.
- hidden_layer_sizes=(100, 50): Defines a network with two hidden layers containing 100 and 50 neurons, respectively.
- activation='relu': Uses the ReLU activation function for neurons.
- solver='adam': Employs the Adam optimization algorithm for weight updates.

Trains the ANN model on the training data and this model is used to predict the target variable for the test data .The accuracy of the model is checked by comparing the predicted values (ann_predictions) to the actual values (y_test).

Test accuracy obtained for ANN model : 0.8789665887903464
Training accuracy obtained for ANN model: 0.8817148203805041

2 . Training and testing of SVM model

```
# 2. Support Vector Machine (SVM)
svm_model = SVC(random_state=42, C=1, kernel='linear')
svm_model.fit(X_train, y_train)
svm_predictions = svm_model.predict(X_test)
svm_accuracy = accuracy_score(y_test, svm_predictions)
print(f"SVM Accuracy: {svm_accuracy}")
```

Figure 20

Creates an SVM model with the following parameters:

- random_state=42: Ensures consistent results for reproducibility.
- C=1: Sets the regularization parameter (a higher value implies a smaller margin and potentially more overfitting).
- kernel='linear': Uses a linear kernel for the SVM.

Trains the SVM model on the training data and this model is used to predict the target variable for the test data .The accuracy of the model is checked by comparing the predicted values (svm_predictions) to the actual values (y_test).

Test accuracy obtained for SVM model : 0.8099439744142264
Training accuracy obtained for SVM model : 0.8104624457739091

3. Training and testing of RF model

```
# 3. Random Forest
params_rf = {'max_depth': 16, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100,
             'random_state': 12345}
model_rf = RandomForestClassifier(**params_rf)
model_rf.fit(X_train, y_train)
rf_predictions = model_rf.predict(X_test)
rf_accuracy = accuracy_score(y_test, rf_predictions)
print(f"Random Forest Accuracy: {rf_accuracy}")
```

Figure 21

Creates an RF model with the following hyperparameters:

- max_depth: 16
- min_samples_leaf: 1
- min_samples_split: 2
- n_estimators: 100
- random_state: 12345

Trains the RF model on the training data and this model is used to predict the target variable for the test data .The accuracy of the model is checked by comparing the predicted values (rf_predictions) to the actual values (y_test).

Test accuracy obtained for RF model : 0.8994261346882301
Training accuracy obtained for RF model : 0.966508832118858

4. Prediction using unseen data

```
# Apply the same standardization to balanced_df using the already fitted scaler
balanced_df_scaled = pd.DataFrame(r_scaler.transform(balanced_df),
                                  index=balanced_df.index,
                                  columns=balanced_df.columns)


# Print the row 0 values from balanced_df to check its content
print(balanced_df_scaled.iloc[0].values.reshape(1, -1))


#unseen data
X_unseen = balanced_df_scaled[selected_columns].iloc[[0]].values.reshape(1, -1)

# Labels for the target
class_labels = ['No', 'Yes']

# Predict unseen data using the ANN model
ann_prediction = ann_model.predict(X_unseen)
print(f"ANN Prediction: {class_labels[ann_prediction[0].astype(int)]}")

# Predict unseen data using the SVM model
svm_prediction = svm_model.predict(X_unseen)
print(f"SVM Prediction: {class_labels[svm_prediction[0].astype(int)]}")

# Predict unseen data using the RF model
rf_prediction = model_rf.predict(X_unseen)
print(f"Random Forest Prediction: {class_labels[rf_prediction[0].astype(int)]}")
```

Figure 22

X_unseen has the unseen data. All three models predict whether it rains in 24 hours or not.If ‘Yes’ it prints ‘Yes’ and if it’s ‘No’ it prints ‘No’. Figure 23 shows the predictions of three models.

```
--
ANN Prediction: Yes
SVM Prediction: Yes
Random Forest Prediction: Yes
```

Figure 23

GIT HISTORY

Git Repository RainfallPrediction contains all colab files, py files,dataset,images,documentation and three related research papers. It is maintained for systematic way of project presentation and mainly for future reference. The colab files are created separately for two sprint releases during the project.

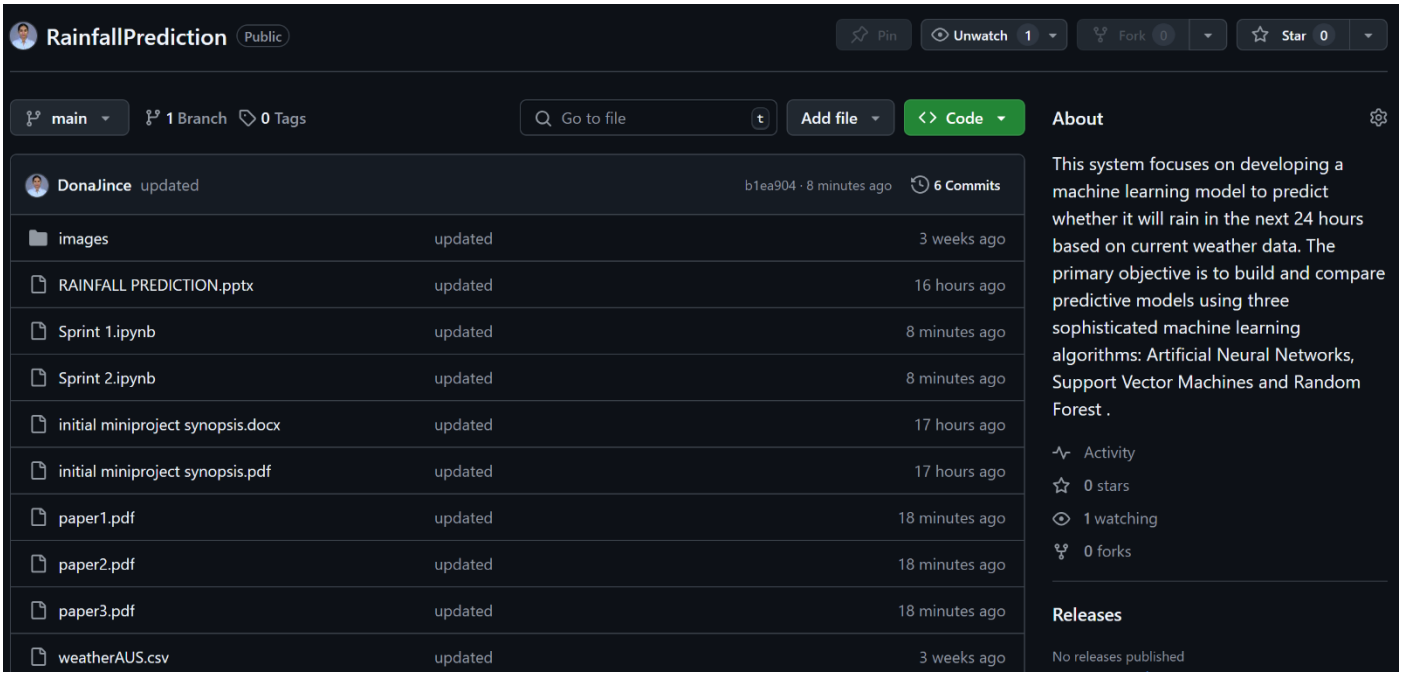


Figure 24

Figure 24 shows the git history of 2 sprints.