THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON, DC

Machine Learning I DATS 6202

(MS In Data Science)

Group Report

Group 3

Rainfall Prediction

Instructor: Amir Jafari

Authors:

Tanmay Vivek Kshirsagar

Sudhanshu Deshpande

Shreyas Sunku Padmanabha

Date: 05/01/2023

TABLE OF CONTENTS

1: Introduction

Rainfall prediction is a critical application of machine learning that has a wide range of practical applications, from agriculture to transportation to disaster management. In this project, our objective is to use the dataset to predict whether it will rain on the next day based on various weather observations made on the current day.

We will first investigate the dataset using exploratory data analysis (EDA). We will pre-process the data after examining it to manage missing values, encode category variables, and scale numerical characteristics. Then, using different algorithms such as logistic regression, MLP Classifier and random forest, we will train and evaluate the machine learning models. Finally, we will select the best performing model and fine-tune its hyperparameters using cross-validation. The result will be a machine learning model that can accurately predict whether it will rain on a given day based on the weather observations. This project has practical applications in weather forecasting and risk assessment and can help inform decision-making in various industries.

2: Dataset Description

The weather dataset contains daily weather observations from various weather stations across Australia, spanning from 2007 to 2017. The dataset includes 142,193 instances and 24 features, including temperature, humidity, rainfall, wind speed, and direction, among others. The target variable is *'RainTomorrow'*, which indicates whether it rained on the following day. The data is in a structured format, with mostly numerical and categorical features. The dataset has missing values, which will require data pre-processing before modelling. The weather dataset is a suitable candidate for binary classification tasks related to rain prediction and risk assessment.

The dataset includes observations of the climate in Australia of a period of 10 years and how the rainfall varies depending on other climate changes.

- Target Variable:
  - RainTomorrow : The amount of next day rain in mm. Used to create response variable RainTomorrow. A kind of measure of the "risk".
- Independant Variables:
  - Date : date of observation
  - Location : region within Australia
  - MinTemp : Minimum temperature in the day in degree Celsius
  - MaxTemp : Maximum temperature in the day in degree Celsius
  - Rainfall : The amount of rainfall recorded for the day in mm
  - Evaporation : The so-called Class A pan evaporation (mm) in the 24 hours to 9am
  - Sunshine : The number of hours of bright sunshine in the day.
  - WindGustDir : The direction of the strongest wind gust in the 24 hours to midnight
  - WindGustSpeed : The speed (km/h) of the strongest wind gust in the 24 hours to midnight

  - WinDir9am : Direction of the wind at 9am
  - WinDir3pm : Direction of the wind at 3pm
  - WindSpeed9am : Wind speed (km/hr) averaged over 10 minutes prior to 9am
  - WindSpeed3pm : Wind speed (km/hr) averaged over 10 minutes prior to 3pm
  - Humidity9am : Humidity (percent) at 9am
  - Humidity3pm : Humidity (percent) at 3pm
  - Pressure9am : Atmospheric pressure (hpa) reduced to mean sea level at 9am
  - Pressure3pm : Atmospheric pressure (hpa) reduced to mean sea level at 3pm
  - Cloud9am : Fraction of sky obscured by cloud at 9am. This is measured in "oktas", which are a unit of eights.
  - Cloud3pm : Fraction of sky obscured by cloud at 3pm. This is measured in "oktas", which are a unit of eights.
  - Temp9am : Temperature (degrees C) at 3pm
  - Temp3pm : Temperature (degrees C) at 9am
  - RainToday : Boolean: 1 if precipitation (mm) in the 24 hours to 9am exceeds 1mm, otherwise 0
- There are a total of 145460 observations.

3: Machine Learning Algorithms

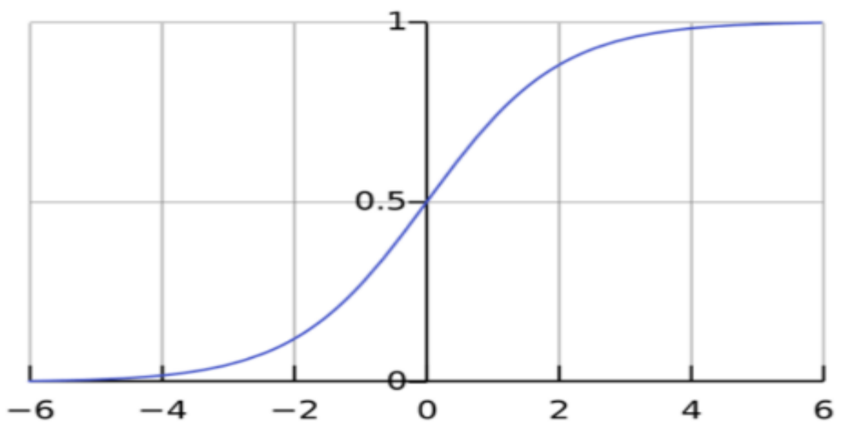
Machine learning algorithms use statistical models and algorithms to identify patterns in data and make predictions or decisions based on that data. As we have a binary classification problem, we have used the following methods.

3.1. Logistic regression

Logistic regression is a machine learning algorithm used for binary classification tasks. It models the relationship between input features and the output variable using a logistic function, which returns a value between 0 and 1. Once trained, the model can be used to predict the probability of the output variable being one of two values based on new input features. Logistic regression is simple and interpretable but has limitations in handling non-linear relationships.

The model parameters are learned by minimizing the cost function using an optimization algorithm like gradient descent. The logistic function has an S-shaped curve that approaches 1 as input values increase and approaches 0 as input values decrease.

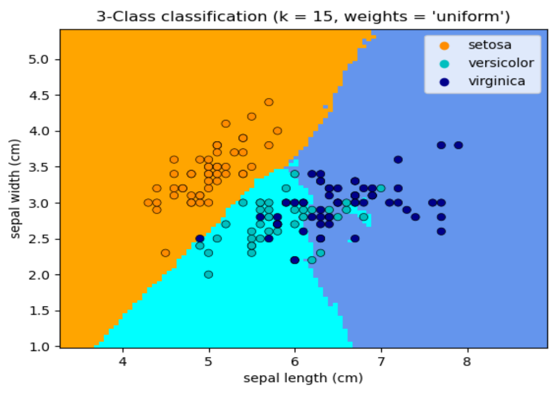
$$f(x) = \frac{1}{1 + e^{-x}}$$



3.2. KNN (K-Nearest Neighbors) Classifier

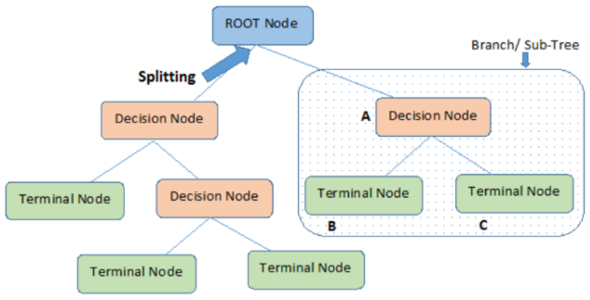
KNN Classifier is a non-parametric machine learning algorithm used for classification tasks. It is based on the idea that similar data points tend to belong to the same class. The algorithm determines the class of a new data point by finding the k nearest neighbors in the training data and assigning the most common class among them as the predicted class for the new data point. The algorithm is simple and easy to implement but can be computationally expensive for large datasets. It also requires careful selection of the hyperparameter k, which determines the number of neighbors to consider.

In summary, K Neighbors Classifier is a simple and effective non-parametric algorithm for classification tasks. It works by finding the k nearest neighbors in the training data and assigning the most common class among them as the predicted class for a new data point.



3.3. Decision Tree Classifier

Decision Tree Classifier is a machine learning algorithm used for classification tasks. It partitions the feature space recursively into subsets based on input features and assigns a class label to each leaf node of the tree. The algorithm determines the best split for each node by maximizing a measure of purity. The decision tree can be graphically represented as a tree structure with each internal node representing a split on an input feature and each leaf node representing a class label. It can handle both categorical and numerical data but is prone to overfitting.

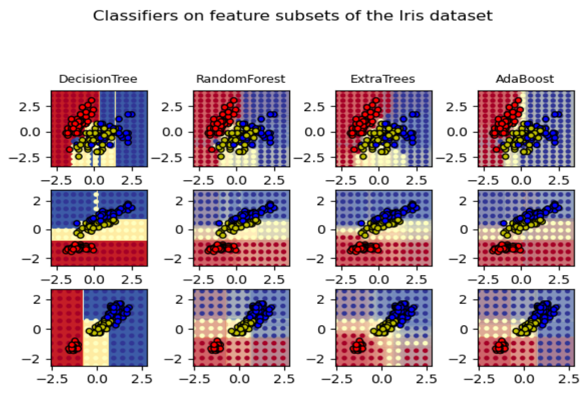


3.4. Random Forest Classifier

Random Forest Classifier is an ensemble machine learning algorithm used for classification tasks. It works by combining multiple decision trees, each trained on a different subset of the training data and a different subset of the input features. The algorithm determines the class of a new data point by aggregating the predictions of all the trees in the forest.

Random Forest Classifier is a powerful algorithm that can handle high-dimensional data and is less prone to overfitting than Decision Tree Classifier. However, it can be computationally expensive and difficult to interpret.

The random forest can be represented graphically as a collection of decision trees, with each tree trained on a different subset of the training data and input features.



3.5. Naïve Bayes

Naive Bayes is a popular classification algorithm used in machine learning. It is based on Bayes' theorem, which provides a way to calculate the probability of a hypothesis based on the probability of its evidence. In Naive Bayes, we assume that the predictors (also known as features or attributes) are conditionally independent given the class variable. This assumption simplifies the probability calculations and makes the algorithm computationally efficient.

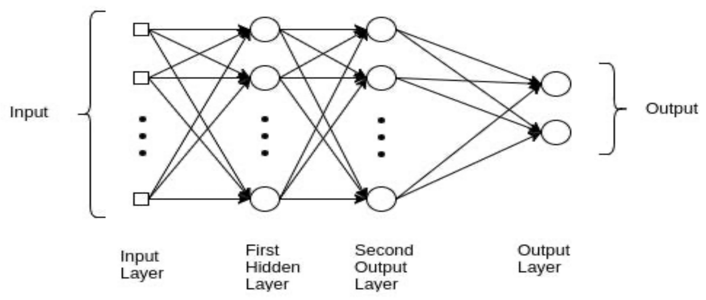
The algorithm works by first training a model on a labeled dataset, where each instance in the dataset is labeled with its corresponding class. During training, the algorithm estimates the probabilities of each feature given each class. Then, when presented with a new instance, the algorithm calculates the probability of each class given the values of its features. The class with the highest probability is then assigned as the predicted class for the instance.

Naive Bayes is particularly useful when working with high-dimensional datasets, where the number of features is large. It has also been shown to perform well even when the independence assumption is not strictly true, making it a robust and versatile algorithm.

Overall, Naive Bayes is a simple yet effective classification algorithm that is widely used in various applications such as text classification, spam filtering, and sentiment analysis.

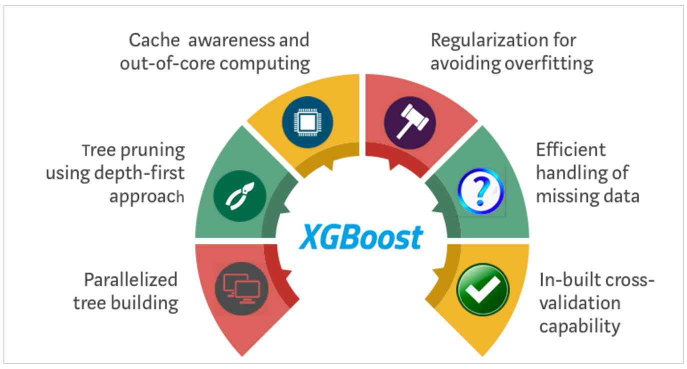
3.6. MLP Classifier

MLP Classifier is a type of neural network used for classification tasks that consists of multiple layers of nodes connected to each other. It was developed in the 1980s to handle non-linearly separable problems and can learn complex non-linear relationships between inputs and outputs by adjusting weights to minimize a loss function. MLP Classifier uses activation functions and softmax function to produce class probabilities. It is a powerful algorithm but can overfit if the model is too complex or if there is not enough training data. The basic architecture of MLP Classifier includes an input layer, one or more hidden layers, and an output layer with nodes corresponding to class labels.



3.7. XGBoost Classifier

XGBoost, short for eXtreme Gradient Boosting, is an advanced implementation of gradient boosting algorithm. It is designed to be efficient, flexible, and portable. The algorithm uses decision trees as base learners and builds an ensemble of these trees by iteratively adding new trees that focus on the errors made by the previous ones. XGBoost can handle both numerical and categorical data and can be used for both regression and classification tasks. It has become very popular in machine learning competitions due to its speed and accuracy and has been used in a variety of applications including fraud detection, click-through rate prediction, and image classification.


Cache awareness and out-of-core computing
Regularization for avoiding overfitting
Tree pruning using depth-first approach
Efficient handling of missing data
XGBoost
Parallelized tree building
In-built cross-validation capability

4. Experimental Setup

The weather dataset from Kaggle was used for this machine learning project. The dataset contains weather observations from various weather stations across Australia from 2007 to 2017. The objective of this project was to predict whether it would rain tomorrow in a given location based on the weather variables.

The dataset was first preprocessed to remove missing values and unnecessary columns. Categorical variables were encoded using one-hot encoding. The target variable, RainTomorrow, was binary encoded as 1 for yes and 0 for no. The preprocessed dataset was then split into training and testing sets using 80:20 ratio. The training set was used to train the machine learning models, while the testing set was used to evaluate their performance.

Three machine learning algorithms were used in this project: logistic regression, decision tree, random forest classifiers, K Neighbor Classifier, Naïve Bayes and Multilayer perceptron classifier. Each algorithm was trained on the training set, and we validated ROC- AUC score on testing set.

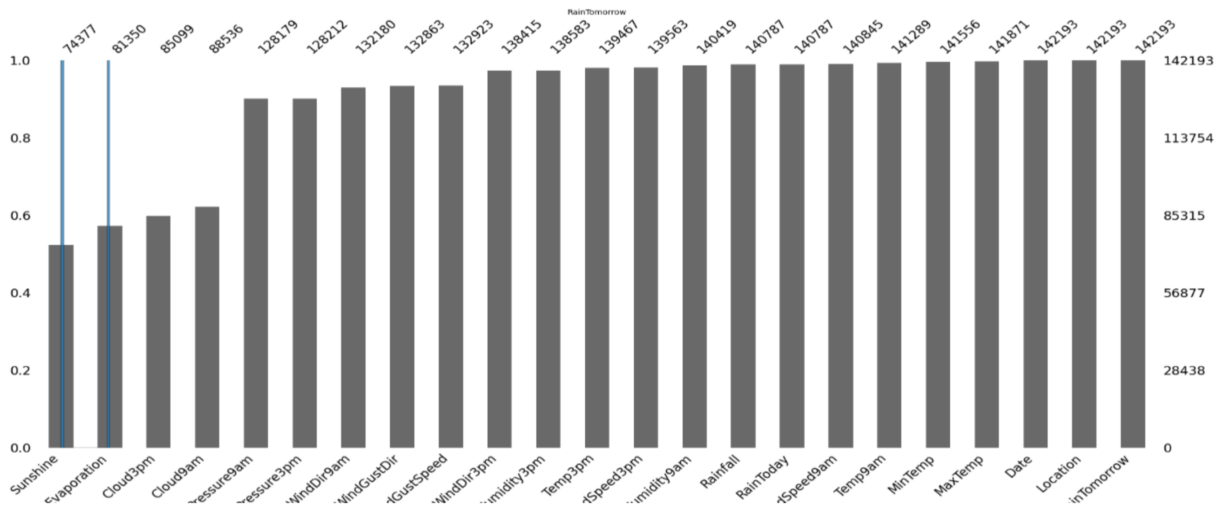
The performance of each algorithm was evaluated using accuracy, precision, recall, and F1-score metrics on the testing set. A confusion matrix was also created to visualize the performance of each algorithm. We also performed additional experimentation on dataset using Principle Component Analysis (PCA) selecting component until 95% variance also, we use k-fold cross validation to compare the accuracy with the models before doing PCA.

The code was separated into three python scripts: main.py, tools.py and models.py. The tools.py script contains functions imputer() which Imputes missing data, impute_scale() which standardizes the numerical columns, Impute_one_hot_encode() which applies one hot encoding to the categorical columns, yes_no_to_binary() that converts “Yes” and “No” response to 1 and 0, and plot_confusion_matrix() that plots confusion matrix for the given model. The models.py file contains a single function called models() that builds all the machine learning models for the given data and prints the results. All these methods are used in the main.py that contains core code.

All experiments were performed on a computer with an Intel Core i5 processor, 8 GB of RAM, and Python 3.11. The machine learning models were implemented using the scikit-learn library.

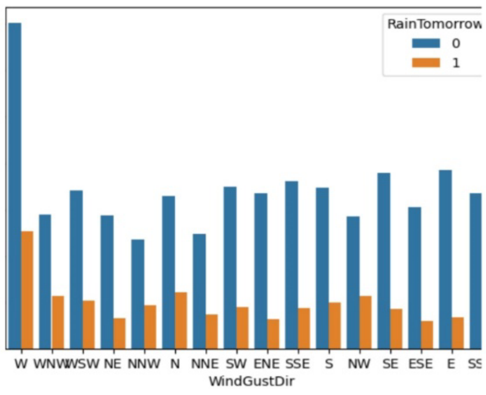
5. Results

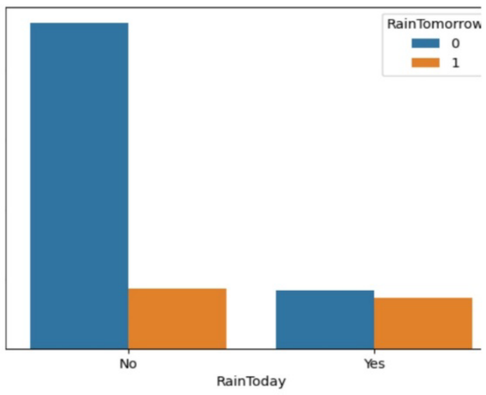
5.1. Missing Values

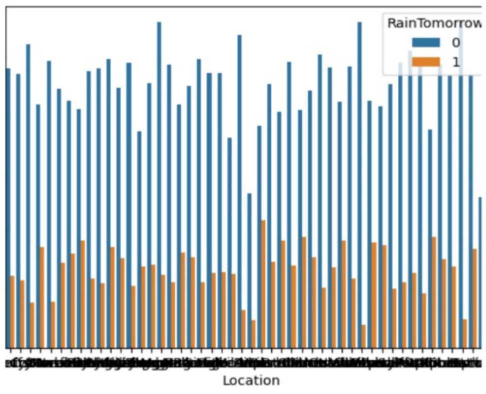


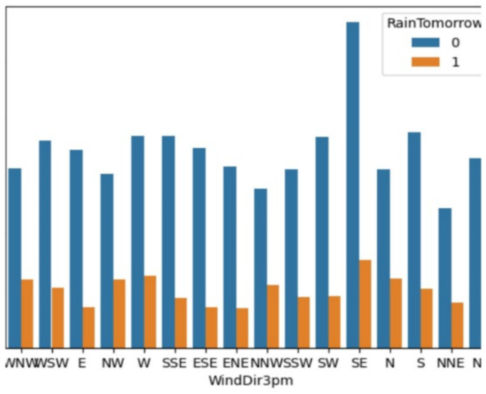
The above picture summarizes the null values in the dataset. We filled these missing values with mean for numerical data and mode for categorical data.

5.2.1. Categorical Variables Plot

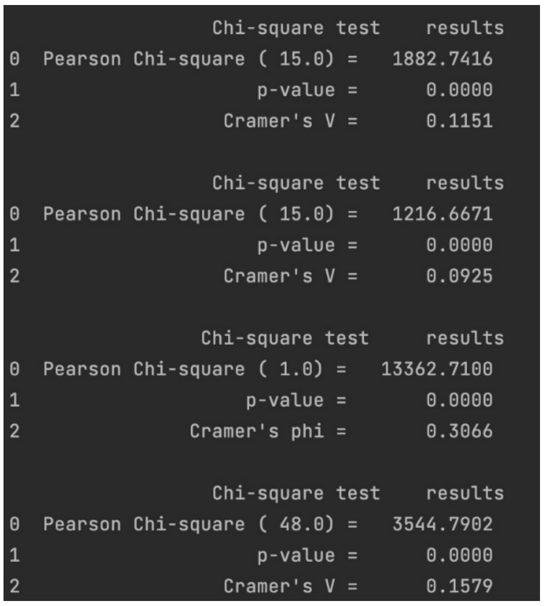








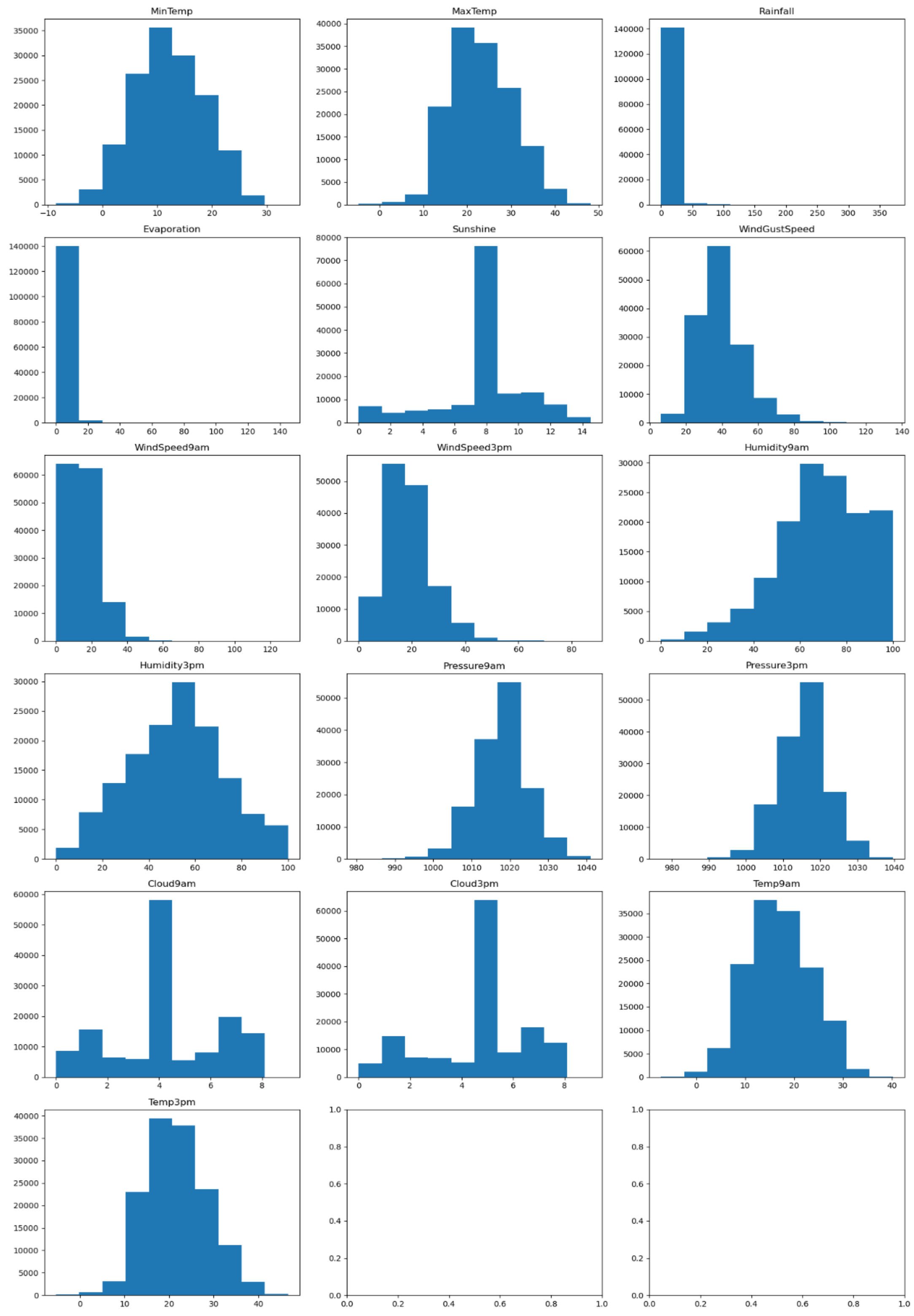
- Chi-square test to show the significance



```
                  Chi-square test    results
0  Pearson Chi-square ( 15.0) =   1882.7416
1                      p-value =      0.0000
2                   Cramer's V =      0.1151

                  Chi-square test    results
0  Pearson Chi-square ( 15.0) =   1216.6671
1                      p-value =      0.0000
2                   Cramer's V =      0.0925

                 Chi-square test     results
0  Pearson Chi-square ( 1.0) =   13362.7100
1                     p-value =       0.0000
2                 Cramer's phi =       0.3066

                  Chi-square test    results
0  Pearson Chi-square ( 48.0) =   3544.7902
1                      p-value =      0.0000
2                   Cramer's V =      0.1579
```

From the chi-square test, we can see that the p-value for all the categorical variables is less than 0.05, which shows the variables are significant.

5.2.2. Numerical Variables Plot

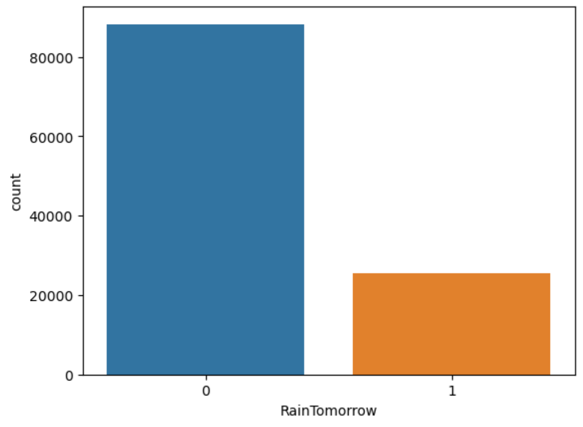


```
['MaxTemp', 'Rainfall', 'Sunshine', 'WindGustSpeed', 'Humidity9am', 'Humidity3pm', 'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp3pm', 'WindSpeed9am',
 'Evaporation', 'WindSpeed3pm', 'MinTemp', 'Temp9am']
```

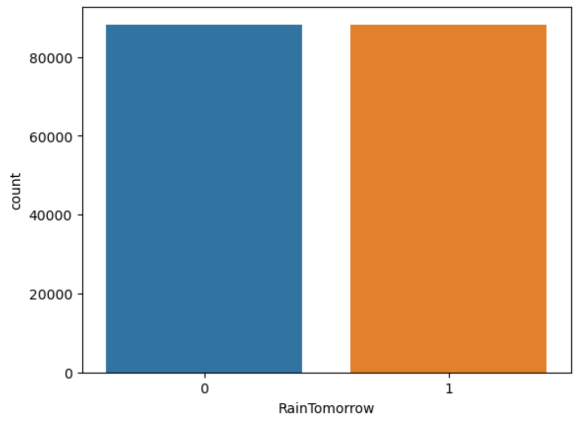
From the histogram that we have plotted for the numerical variables, we can see that the data is fairly distributed. Additionally, the SelectKBest method selected all the numerical variables as significant.

5.2.3. Balancing dependent Variable

- Before balancing the data



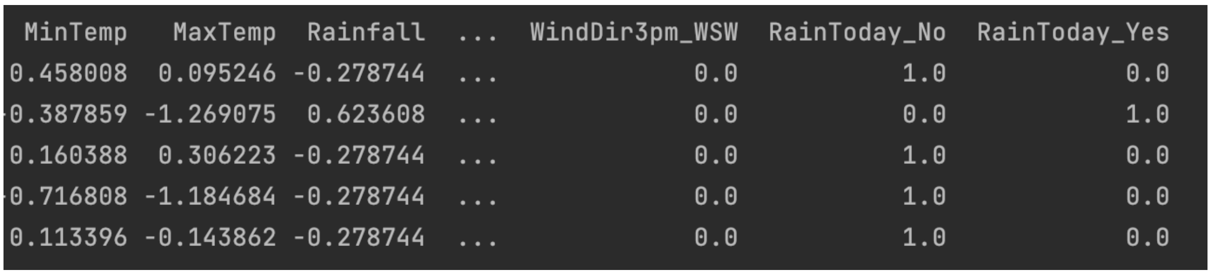
- After Balancing the data



As we can see from the plot, the raw data is imbalanced with approximately 70% for ‘0’ and 30% for ‘1’. So we performed the balancing of the data using technique called SMOTE which mainly perform oversampling and helps to handle the overfitting of the data.

5.2.4. Standardization & One-hot Encoding on independent Features

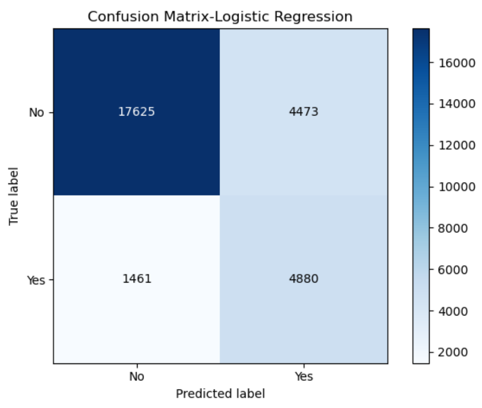
Standardize the numerical features by scaling them to have zero mean and unit variance using standard scaling. One hot encoding creates dummy variables which is a duplicate variable which represents one level of a categorical variable. Presence of a level is represented by 1 and absence is represented by 0. Here is a snapshot of one hot encoding on the categorical features and standard scaling on numerical features.



```
 MinTemp   MaxTemp  Rainfall  ...  WindDir3pm_WSW  RainToday_No  RainToday_Yes
0.458008  0.095246 -0.278744  ...             0.0           1.0            0.0
0.387859 -1.269075  0.623608  ...             0.0           0.0            1.0
0.160388  0.306223 -0.278744  ...             0.0           1.0            0.0
0.716808 -1.184684 -0.278744  ...             0.0           1.0            0.0
0.113396 -0.143862 -0.278744  ...             0.0           1.0            0.0
```

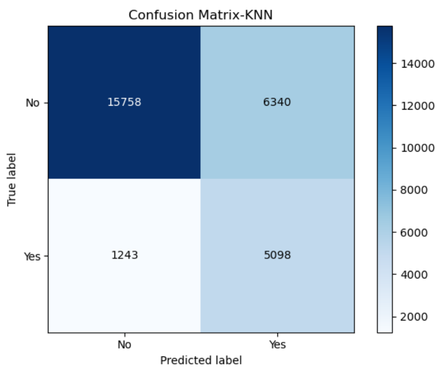
5.3. Classification Modelling

5.3.1. Logistic Regression



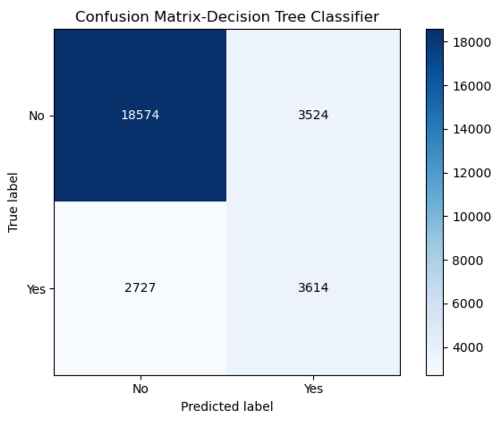
- The total number of instances in the test set is 26,439 (17625 + 4473 + 1461 + 4880).
- Out of these instances, 17,625 were actually negative and were correctly classified as negative (true negatives).
- 4,473 instances were actually positive and were incorrectly classified as negative (false negatives).
- 1,461 instances were actually negative and were incorrectly classified as positive (false positives).
- 4,880 instances were actually positive and were correctly classified as positive (true positives).

5.3.2. KNN



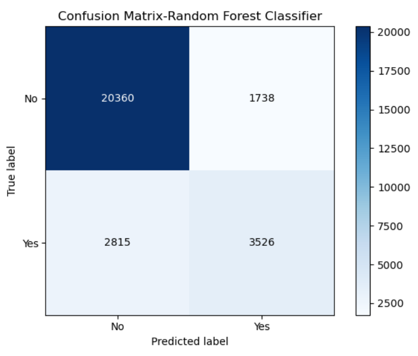
- The total number of instances in the test set is 26,439 (15758 + 6340 + 1243 + 5098).
- Out of these instances, 15,758 were actually negative and were correctly classified as negative (true negatives).
- 6,340 instances were actually positive and were incorrectly classified as negative (false negatives).
- 1,243 instances were actually negative and were incorrectly classified as positive (false positives).
- 5,098 instances were actually positive and were correctly classified as positive (true positives).

5.3.3. Decision Tree



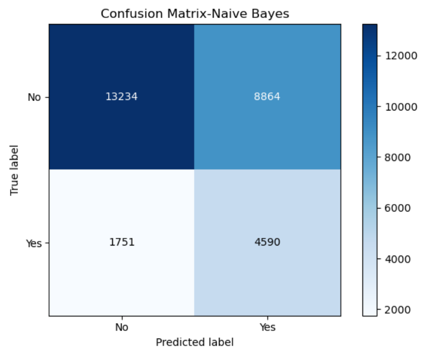
- The total number of instances in the test set is 26,439 (18538 + 3560 + 2784 + 3557).
- Out of these instances, 18,538 were actually negative and were correctly classified as negative (true negatives).
- 3,560 instances were actually positive and were incorrectly classified as negative (false negatives).
- 2,784 instances were actually negative and were incorrectly classified as positive (false positives).
- 3,557 instances were actually positive and were correctly classified as positive (true positives).

5.3.4. Random Forest



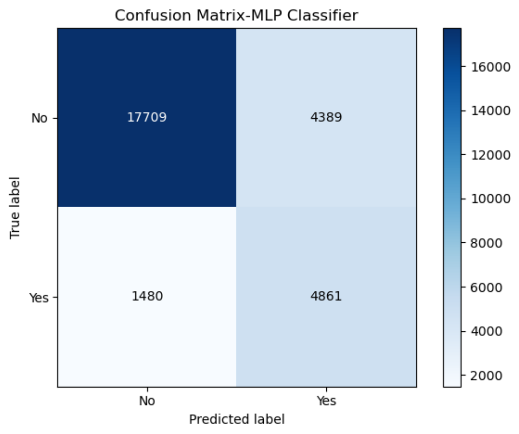
- The total number of instances in the test set is 26,439 (20360 + 1738 + 2815 + 3526).
- Out of these instances, 20,360 were actually negative and were correctly classified as negative (true negatives).
- 1,738 instances were actually positive and were incorrectly classified as negative (false negatives).
- 2,815 instances were actually negative and were incorrectly classified as positive (false positives).
- 3,526 instances were actually positive and were correctly classified as positive (true positives).

5.3.5. Naïve Bayes



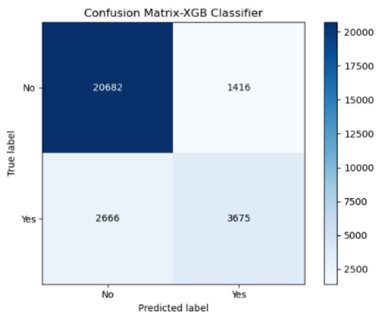
- The total number of instances in the test set is 26,439 (13234 + 8864 + 1751 + 4590).
- Out of these instances, 13,234 were actually negative and were correctly classified as negative (true negatives).
- 8,864 instances were actually positive and were incorrectly classified as negative (false negatives).
- 1,751 instances were actually negative and were incorrectly classified as positive (false positives).
- 4,590 instances were actually positive and were correctly classified as positive (true positives).

5.3.6. Multi Layer Perceptron



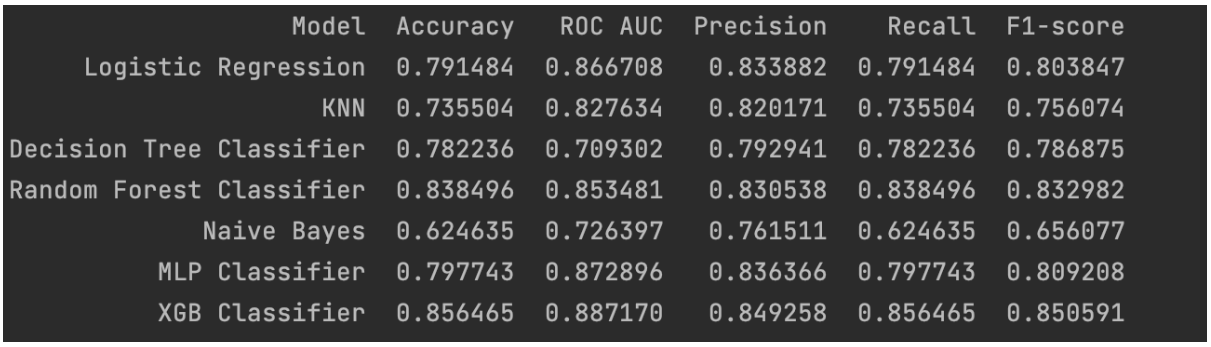
- The total number of instances in the test set is 26,439 (17709 + 4389 + 1480 + 4861).
- Out of these instances, 17,709 were actually negative and were correctly classified as negative (true negatives).
- 4,389 instances were actually positive and were incorrectly classified as negative (false negatives).
- 1,480 instances were actually negative and were incorrectly classified as positive (false positives).
- 4,861 instances were actually positive and were correctly classified as positive (true positives).

5.3.7. XG Boost



- The total number of instances in the test set is 26,439 (20,682 + 1,416 + 2,666 + 3,675).
- Out of these instances, 20,682 were actually negative and were correctly classified as negative (true negatives).
- 2,666 instances were actually positive and were incorrectly classified as negative (false negatives).
- 1,416 instances were actually negative and were incorrectly classified as positive (false positives).
- 3,675 instances were actually positive and were correctly classified as positive (true positives).

5.3.8. Comparison of Models



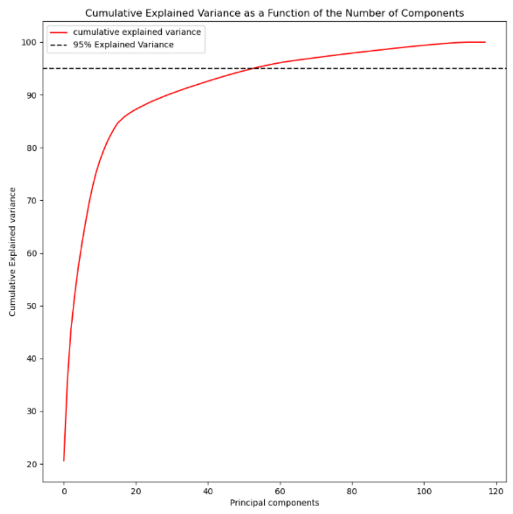
```
                   Model  Accuracy   ROC AUC  Precision    Recall  F1-score
     Logistic Regression  0.791484  0.866708   0.833882  0.791484  0.803847
                     KNN  0.735504  0.827634   0.820171  0.735504  0.756074
Decision Tree Classifier  0.782236  0.709302   0.792941  0.782236  0.786875
Random Forest Classifier  0.838496  0.853481   0.830538  0.838496  0.832982
             Naive Bayes  0.624635  0.726397   0.761511  0.624635  0.656077
          MLP Classifier  0.797743  0.872896   0.836366  0.797743  0.809208
          XGB Classifier  0.856465  0.887170   0.849258  0.856465  0.850591
```

we can see that the XGB Classifier achieved the highest accuracy (0.85) and ROC AUC (0.88) among all the models, indicating that it performed the best in terms of overall classification accuracy and ability to distinguish between positive and negative instances. It also achieved a good balance between precision and recall, as indicated by its high F1-score (0.85).

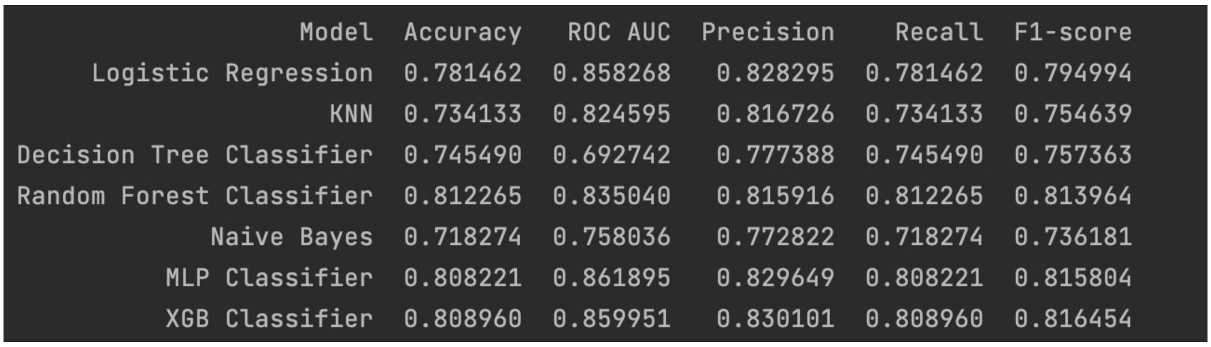
The Random Forest and MLP Classifier models also performed well, with accuracy and ROC AUC scores close to that of the Random Forest Classifier.

The Naive Bayes model had the lowest accuracy (0.63) and ROC AUC (0.73) among all the models, indicating that it performed the worst in terms of overall classification accuracy and ability to distinguish between positive and negative instances. However, it achieved the highest precision (0.76) among all the models, which suggests that it is better at identifying true positive instances than the other models, albeit at the cost of misclassifying a larger number of negative instances.

5.4. Principal Component Analysis



Based on the graph presented above, we can observe that using 50 components covers 95% of the explained variance. Therefore, we selected those 50 components and used them to build our model. The results of the model are presented below.


```
                   Model  Accuracy   ROC AUC  Precision    Recall  F1-score
     Logistic Regression  0.781462  0.858268   0.828295  0.781462  0.794994
                     KNN  0.734133  0.824595   0.816726  0.734133  0.754639
Decision Tree Classifier  0.745490  0.692742   0.777388  0.745490  0.757363
Random Forest Classifier  0.812265  0.835040   0.815916  0.812265  0.813964
             Naive Bayes  0.718274  0.758036   0.772822  0.718274  0.736181
          MLP Classifier  0.808221  0.861895   0.829649  0.808221  0.815804
          XGB Classifier  0.808960  0.859951   0.830101  0.808960  0.816454
```

By reducing the dimensionality of the dataset and selecting only 50 components, which is approximately half of the initial dataset's components, we were able to achieve comparatively similar metric scores.

5.4. K-Cross Validation

We utilized the K-Nearest Neighbors Classifier algorithm to build a classification model. The model was trained on the training data using 10-fold cross-validation, where k was set to 3. The accuracy of the model was evaluated using the cross_val_score() function in the scikit-learn library.

The results of the cross-validation indicate that the model had an average accuracy of 0.865 with a standard deviation of 0.007, which suggests that the model's performance is consistent across the different folds of the cross-validation process. These results demonstrate that the K-Nearest Neighbors Classifier algorithm is a suitable choice for this classification problem.

6. Summary and Conclusions

Using different machine learning algorithm models, we predicted whether on the next day it will rain or not. Out of the models that we built, we found out that XGB classifier gave out the best performance. We learned about different techniques to handle data pre-processing, such as data imputation, feature selection, sampling technique SMOTE. We also learned how different classification algorithms work and the importance of doing principal component analysis (PCA).

In the future, we would like to perform in depth data processing techniques and apply neural networks.

References

https://aws.amazon.com/what-is/logistic-regression/
https://www.analyticsvidhya.com/blog/2021/08/decision-tree-algorithm/
https://scikit-learn.org/stable/modules/neighbors.html/
https://medium.com/@AI_with_Kain/understanding-of-multilayer-perceptron-mlp-8f179c4a135f
https://machinelearningmastery.com/develop-first-xgboost-model-python-scikit-learn/
https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/
https://www.kaggle.com/datasets/jsphyg/weather-dataset-rattle-package
https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-imbalanced-classification/

Appendix

Computers used:

- Macbook Air M1
- HP Pavilion
- MSI Gaming

Software used:

- Pycharm Professional
- Jupyter Notebook
- GitHub
- Microsoft Word
- Microsoft Powerpoint