THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON, DC

Machine Learning I DATS 6202

(MS In Data Science)

Individual Report

Group 3

Rainfall Prediction

Instructor: Amir Jafari

Authors:

Tanmay Vivek Kshirsagar

Date: 05/01/2023

TABLE OF CONTENTS

1: Introduction

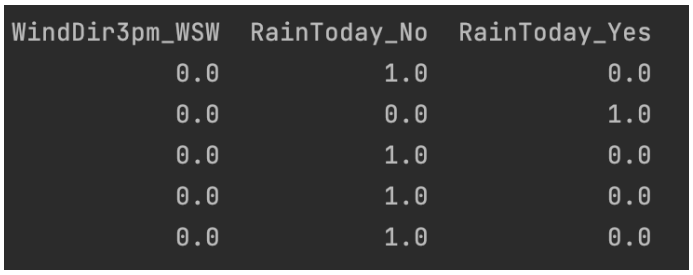
Rainfall prediction is a critical application of machine learning that has a wide range of practical applications, from agriculture to transportation to disaster management. In this project, our objective is to use the dataset to predict whether it will rain on the next day based on various weather observations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

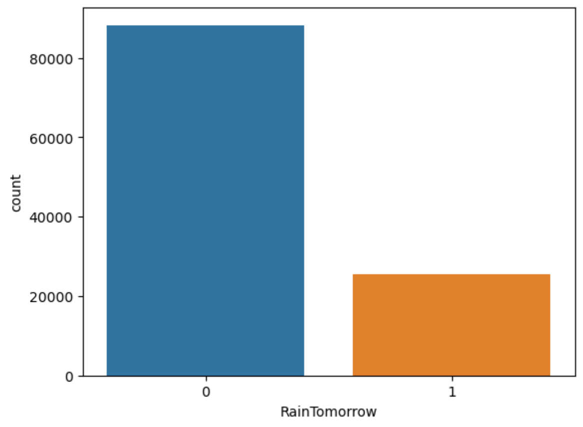
2.1. One Hot Encoding

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.

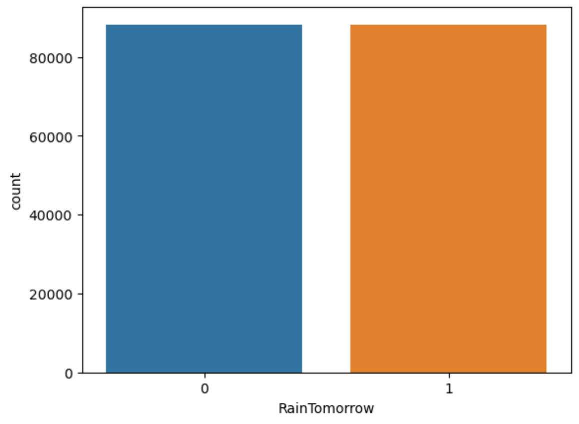


```
WindDir3pm_WSW  RainToday_No  RainToday_Yes
           0.0           1.0            0.0
           0.0           0.0            1.0
           0.0           1.0            0.0
           0.0           1.0            0.0
           0.0           1.0            0.0
```

2.2. SMOTE



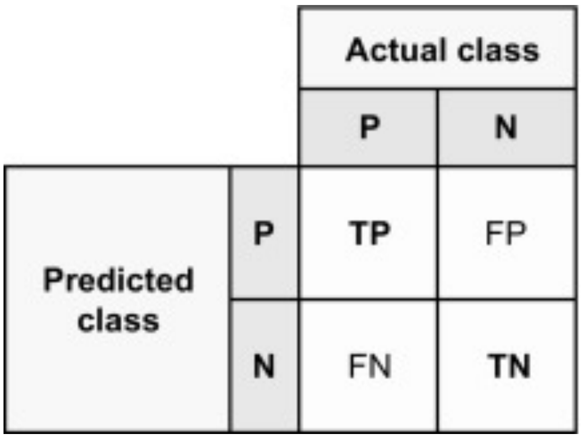
- After Balancing the data



As we can see from the plot, the raw data is imbalanced with 80% for ‘0’ and 20% 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.

2.3. Confusion Matrix

A confusion matrix represents the prediction summary in matrix form. It shows how many prediction are correct and incorrect per class. It helps in understanding the classes that are being confused by model as other class.



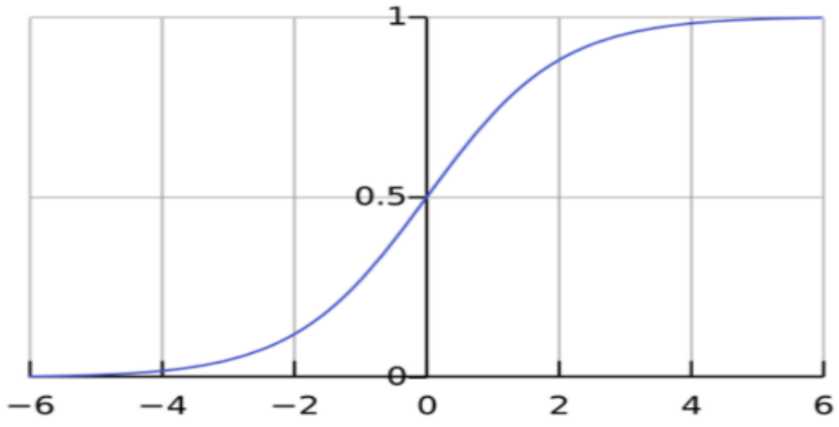
		Actual class	
		P	**N**
Predicted class	**P**	**TP**	FP
	N	FN	**TN**

It is often used to measure the performance of classification models, which aim to predict a categorical label for each input instance. The matrix displays the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) produced by the model on the test data.

2.4. Logistic regression

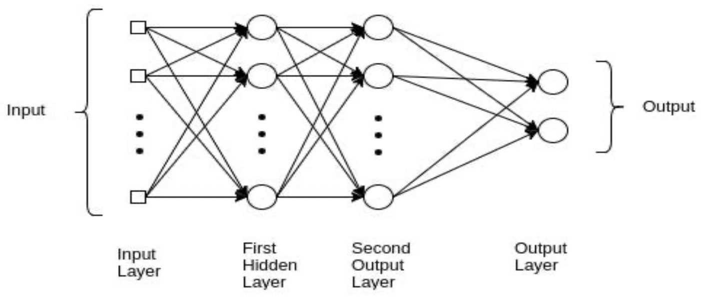
Logistic regression is a machine learning algorithm used for binary classification tasks. It models the relationship between input features and the output variable using a logistic function, which returns a value between 0 and 1. Once trained, the model can be used to predict the probability of the output 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.

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



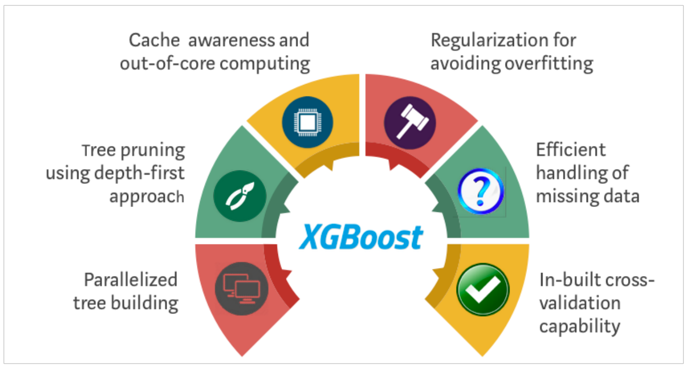
2.5. MLP Classifier

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.



2.6. 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

3: Description of Portion of my work

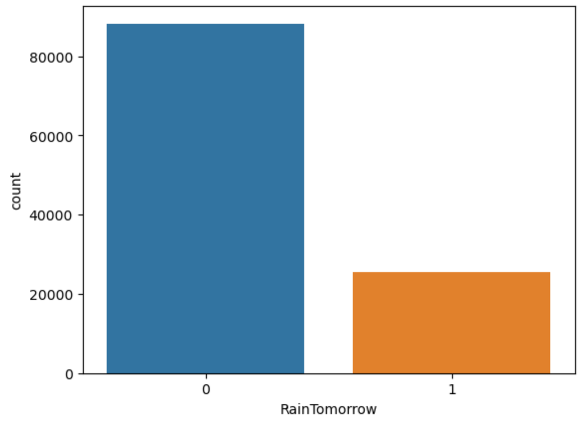
All of us in the team worked on the initial data understanding. I worked on developing the one hot encoding function, plotting confusion matrix function, developed machine learning models on XGBoost Classification, MLP Classifier & Logistic Regression, and modularized the code. I created the Github repository, folder structure and was responsible for solving any issues that my teammates faced.

The impute_one_hot_encode function handles missing values in categorical features and performs one-hot encoding on the same. The plot_confusion_matrix function plots the confusion matrix for a given method. The models function creates and trains various machine learning models and computes several performance metrics like accuracy, ROC-AUC, precision, recall, and F1-score for each model.

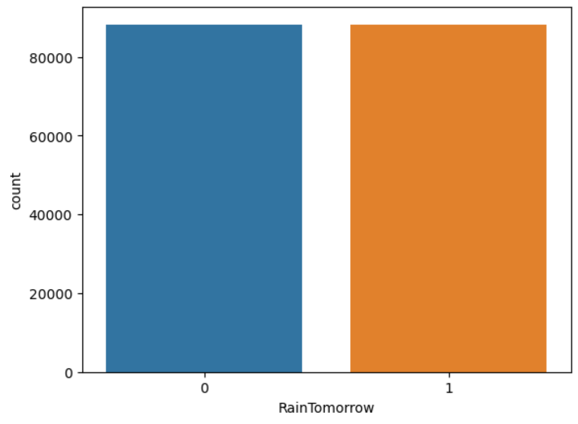
4. Results

4.1. Data Sampling

- Before balancing the data



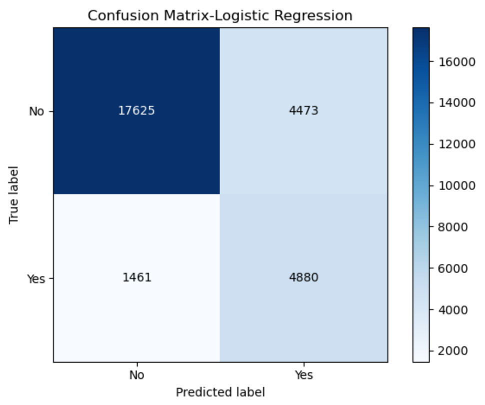
- After Balancing the data



As we can see from the plot, the raw data is imbalanced with 80% for ‘0’ and 20% 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.

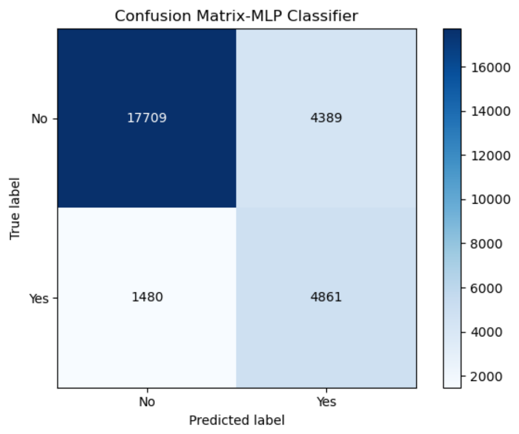
4.2. Classification Modelling

4.2.1. 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).

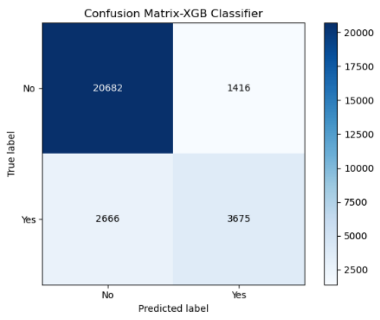
4.2.2. Multi Layer Perceptron



- The total number of instances in the test set is 26,439 (17709 + 4389 + 1480 + 4861).

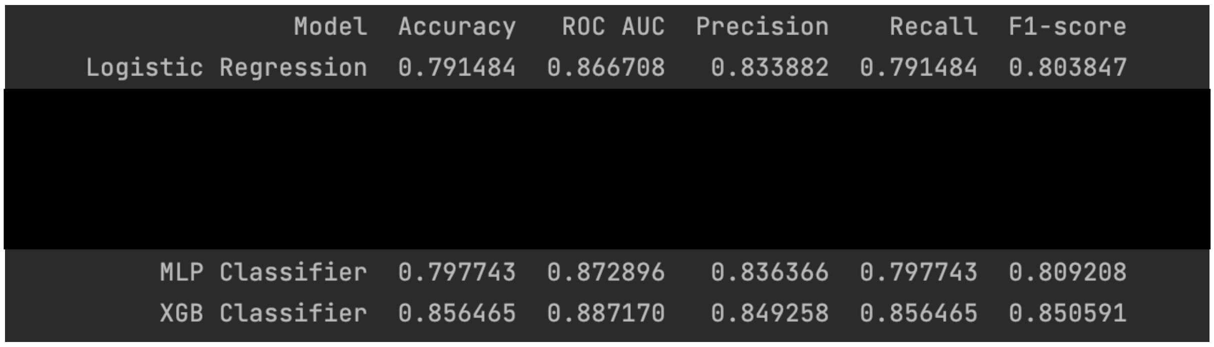
- Out of these instances, 17,709 were actually negative and were correctly classified as negative (true negatives).
- 4,389 instances were actually positive and were incorrectly classified as negative (false negatives).
- 1,480 instances were actually negative and were incorrectly classified as positive (false positives).
- 4,861 instances were actually positive and were correctly classified as positive (true positives).

4.2.3. 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).

4.2.4. Comparison of Models



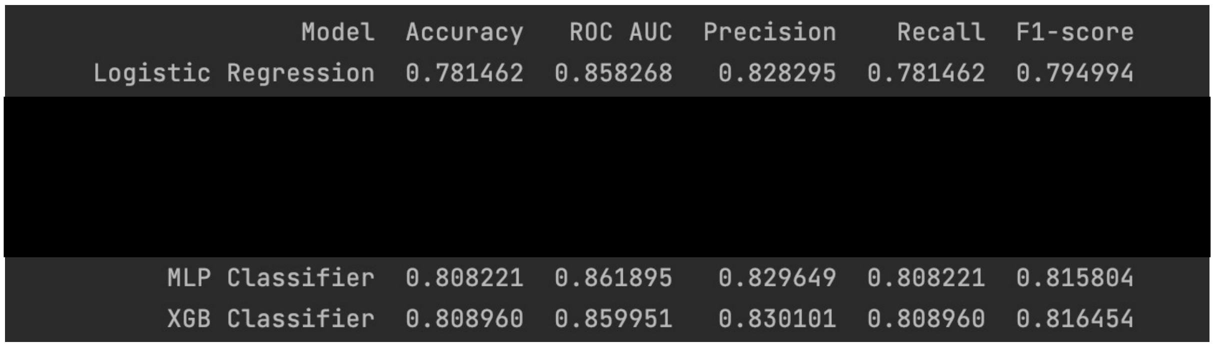
```
              Model  Accuracy   ROC AUC  Precision    Recall  F1-score
Logistic Regression  0.791484  0.866708   0.833882  0.791484  0.803847
[redacted]
     MLP Classifier  0.797743  0.872896   0.836366  0.797743  0.809208
     XGB Classifier  0.856465  0.887170   0.849258  0.856465  0.850591
```

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.

After performing PCA, that was done by my teammate, I got the following results:



```
              Model  Accuracy   ROC AUC  Precision    Recall  F1-score
Logistic Regression  0.781462  0.858268   0.828295  0.781462  0.794994
[redacted]
     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, the models that I built were able to achieve comparatively similar metric scores.

5. 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 I built, we found out that XGB classifier gave out the best performance. I learned about different techniques to handle data pre-processing, such as data imputation, feature selection from my colleagues. I learned how to perform sampling technique SMOTE, how to do one hot encoding, how to plot confusion matrix and how to build models using XGBoost Classifier, MLP Classifier and Logistic Regression. I also learned about to modularize the code which helps in greatly reducing the redundant code. In the future, I would like to perform in depth data processing techniques and apply neural networks as well.

6. Code Percentage

I have 233 lines in the mywork.py out of which, after removing blank lines, import package lines and some common code, my code occupies 136 lines of code. Out of this 136, I used 98 lines of code from the internet, edited 60 lines of code and added 36 lines of my own code. Thus, using the given formula, percentage will be around 29%.

References

https://aws.amazon.com/what-is/logistic-regression/
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/
https://towardsdatascience.com/one-hot-encoding-standardization-pca-data-preparation-steps-for-segmentation-in-python-24d07671cf0b