DATS 6202

Machine Learning I

Prof. Amir Jafari

Heart Disease Prediction

Individual Final Project

Kyuri Kim

Team 8

Table of Contents

1. Introduction

Heart disease is one of the leading causes of death globally, and according to the Centers for Disease Control and Prevention (CDC), it is also one of the leading causes of death in the United States. In the US, heart disease accounts for approximately one in every four deaths, and it affects people of most races, including African Americans, American Indians and Alaska Natives, and white people. The prevalence of heart disease is largely due to factors such as high blood pressure, high cholesterol, smoking, diabetes, obesity, physical inactivity, and excessive alcohol consumption. Detecting and preventing these factors is therefore crucial in healthcare.

The Behavioral Risk Factor Surveillance System (BRFSS) dataset, collected annually by the CDC, provides a wealth of information on the health status of US residents. The dataset includes responses to questions about various health indicators and risk factors, including those related to heart disease.

The purpose of this project is to explore the BRFSS dataset and to apply machine learning algorithms to predict the likelihood of heart disease based on selected variables. In particular, we will focus on the most relevant variables and clean the dataset to make it usable for the machine learning project.

Our team worked together to clean the dataset, ensuring its quality and reliability, followed by an Exploratory Data Analysis (EDA) individually to have a better understanding of the dataset. Once the EDA was completed, we collectively performed data preprocessing tasks, including balancing the dataset, encoding categorical variables, and removing outliers. Finally, we collaborated on building predictive models using various algorithms such as logistic regression, decision tree, random forest, and multilayer perceptron (MLP). This comprehensive and collaborative effort enabled us to gain valuable insights from the dataset and develop predictive models.

2. Description of my individual work

2.1. Exploratory Data Analysis

Exploratory Data Analysis (EDA) plays a pivotal role in machine learning projects as it enables a thorough understanding of the dataset, unveiling patterns, and extracting valuable insights about the underlying data structure. EDA involves visually and statistically analyzing the data, identifying relationships, distributions, and potential outliers. It serves as a foundation for subsequent data preprocessing and model building steps. For this project, we used histogram, pie chart for categorical features and boxplot and histogram for numerical features for the analysis of the dataset.

2.2. Pandas

Pandas is a robust data manipulation library in Python that provides powerful data structures and functions for efficient handling of structured data. Its centerpiece is the DataFrame object, which facilitates the storage and manipulation of tabular data. By implementing Pandas, we can

easily load, clean, transform, and analyze datasets. The library offers convenient methods for filtering, sorting, aggregating, and handling missing values, enabling seamless data preparation for analysis.

2.3. Matplot

Matplotlib is a widely adopted plotting library in Python, providing a comprehensive set of tools for creating static, animated, and interactive visualizations. It offers a low-level interface, granting researchers fine-grained control over plot elements, such as axes, labels, legends, and styles. Matplotlib supports an extensive range of plot types and customization options, making it suitable for generating publication-quality figures. Researchers can utilize Matplotlib to create a diverse array of visualizations to effectively communicate their findings.

2.4. Seaborn

Seaborn is a data visualization library built on top of Matplotlib, offering a higher-level interface for creating visually appealing and informative statistical graphics. It simplifies the creation of various plot types, including scatter plots, bar plots, box plots, and heatmaps. Seaborn also provides built-in functionality for visualizing relationships between variables, data distributions, and pattern discovery. Its user-friendly API enables researchers to generate insightful visualizations with ease.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

3. Describe the portion of my work

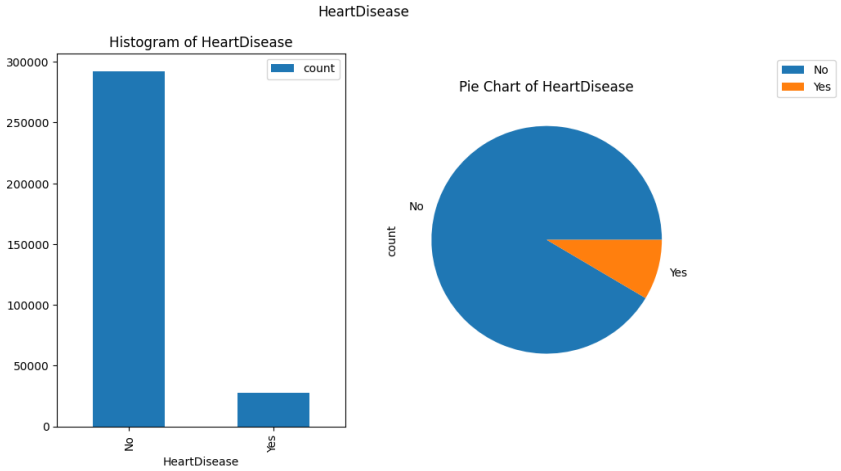
During the initial phase of our project, our team collaboratively focused on data understanding by collectively cleaning the dataset and conducting individual EDA. My specific responsibility within the EDA process was to analyze variables such as Smoke, AlcoholDrinking, DiffWalking, PhysicalActivity, SleepTime, PhysicalHealth, and MentalHealth. Once everyone completed their individual analysis, I consolidated all the code, ensuring consistent formatting and generating formatted plots that would be utilized for our presentation and report.

Additionally, I took charge of creating a cohesive team environment by creating and managing our GitHub repository, organizing relevant folders, and providing necessary documentation. Lastly, I assisted my teammates with any issues they encountered during their respective tasks.

4. Results

4.1. Exploratory Data Analysis

Based on the value counts of our target variable, Heart Disease, we can see that the dataset is highly imbalanced.



4.2. Logistic Regression

The result of logistic regression is shown below:

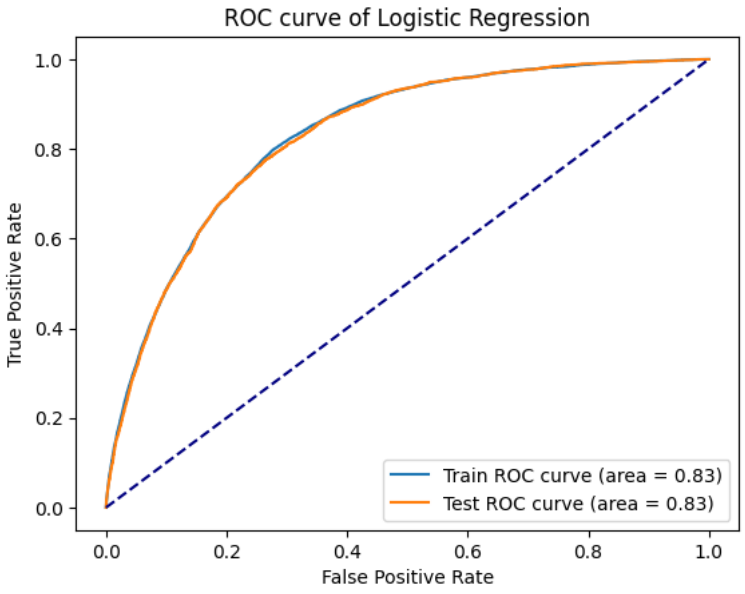
```
Train:
---------------------------------
                precision    recall  f1-score   support

           0       0.98      0.74      0.84    167914
           1       0.18      0.79      0.30     12462

    accuracy                           0.74    180376
   macro avg       0.58      0.76      0.57    180376
weighted avg       0.92      0.74      0.80    180376

---------------------------------
ROC AUC score: 0.8364346464569035
---------------------------------
Test:
              precision    recall  f1-score   support

           0       0.98      0.74      0.84     41926
           1       0.18      0.78      0.30      3169

    accuracy                           0.74     45095
   macro avg       0.58      0.76      0.57     45095
weighted avg       0.92      0.74      0.80     45095

---------------------------------
                  Predicted No Disease  Predicted Disease
Actual No Disease                30946              10980
Actual Disease                     703               2466
---------------------------------
```

```
ROC-AUC score: 0.8346955635533716
---------------------------------
Balanced accuracy: 0.7581367309217384
---------------------------------
```

The logistic regression model was trained and tested on an imbalanced dataset. The train dataset was balanced, while the test dataset was raw and imbalanced. The performance of the model was evaluated using various metrics, including precision, recall, F1-score, ROC AUC, and the precision-recall curve.

The classification report for the train dataset showed an overall accuracy of 0.73. The precision for predicting the positive class was low (0.18), while the recall was relatively high (0.78). This indicates that the model correctly identified most of the positive cases but also misclassified a large number of negative cases as positive. The F1-score for the positive class was 0.29, which is relatively low, indicating that the model has difficulty in correctly classifying the positive class.

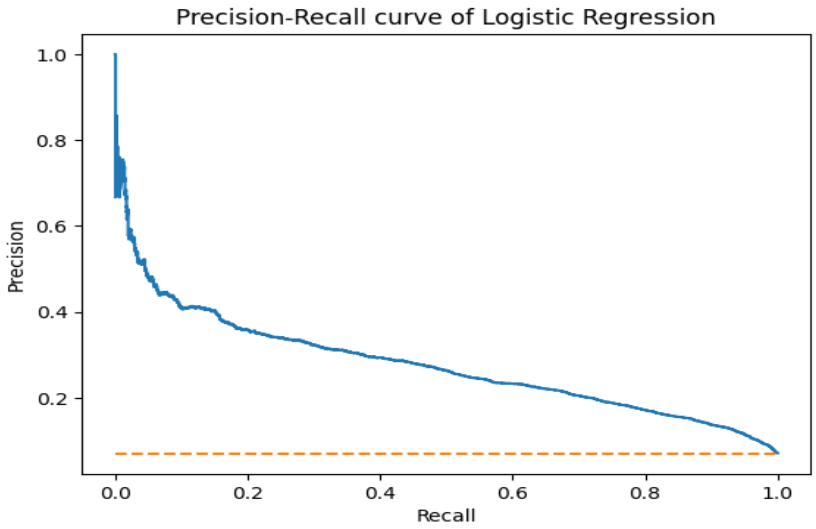
The ROC AUC score for the train dataset was 0.83, indicating good overall performance of the model. However, it is not the best indicator for imbalance data, we could see that the model has similar performance on both train and test dataset.



The precision-recall curve showed a trade-off between precision and recall, with the highest F1-score achieved at a threshold value of around 0.2. The graph of the precision-recall curve just above the no skill line which indicated the same result from the classification report.

On the other hand, the test dataset showed similar performance to the train dataset. The classification report for the test dataset showed an overall accuracy of 0.74, with a precision of 0.18 and recall of 0.78 for the positive class. The F1-score for the positive class was 0.29, similar to that of the train dataset.

The confusion matrix for the test dataset showed that the model correctly classified most of the negative cases but misclassified a large number of positive cases as negative. The ROC AUC score for the test dataset was 0.834, slightly lower than that of the train dataset. The precision-recall curve for the test dataset showed a similar trade-off between precision and recall as the train dataset. It can be seen that the model is just barely above the no skill line for most of the thresholds.



The balanced accuracy for the test dataset was 0.752, which is similar to that of the train dataset. This indicates that the model's performance on the imbalanced test dataset was also slightly better at correctly classifying the negative cases than the positive cases.

In summary, the logistic regression model showed similar performance on both the train and test datasets, with relatively low precision and F1-score for the positive class. The ROC AUC score and precision-recall curve indicated good overall performance of the model. However, the misclassification of positive cases suggests that the model may not be suitable for practical use in identifying the positive cases in an imbalanced dataset.

4.3. Decision Tree

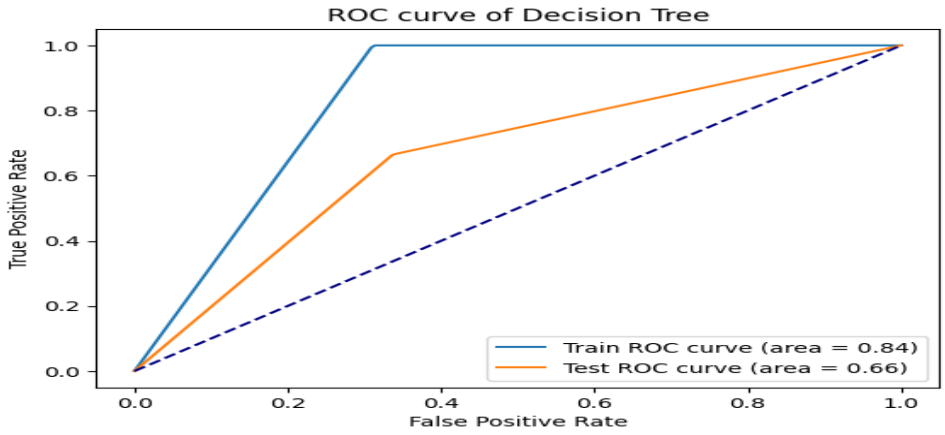
The Results from Decision Tree is shown below:

```
Train:
--------------------------------
              precision    recall  f1-score   support

           0       1.00      0.70      0.82    167914
           1       0.20      0.99      0.33     12462

    accuracy                           0.72    180376
   macro avg       0.60      0.84      0.57    180376
weighted avg       0.94      0.72      0.79    180376
```

```
---------------------------------
ROC AUC score: 0.8468571002771255
---------------------------------

Test:
              precision    recall  f1-score   support

           0       0.96      0.67      0.79     41926
           1       0.13      0.65      0.22      3169

    accuracy                           0.67     45095
   macro avg       0.55      0.66      0.51     45095
weighted avg       0.90      0.67      0.75     45095

---------------------------------
                   Predicted No Disease  Predicted Disease
Actual No Disease                 28274              13652
Actual Disease                     1104               2065
---------------------------------
ROC-AUC score: 0.6634858744569821
---------------------------------
Balanced accuracy: 0.6630018927546795
---------------------------------
```

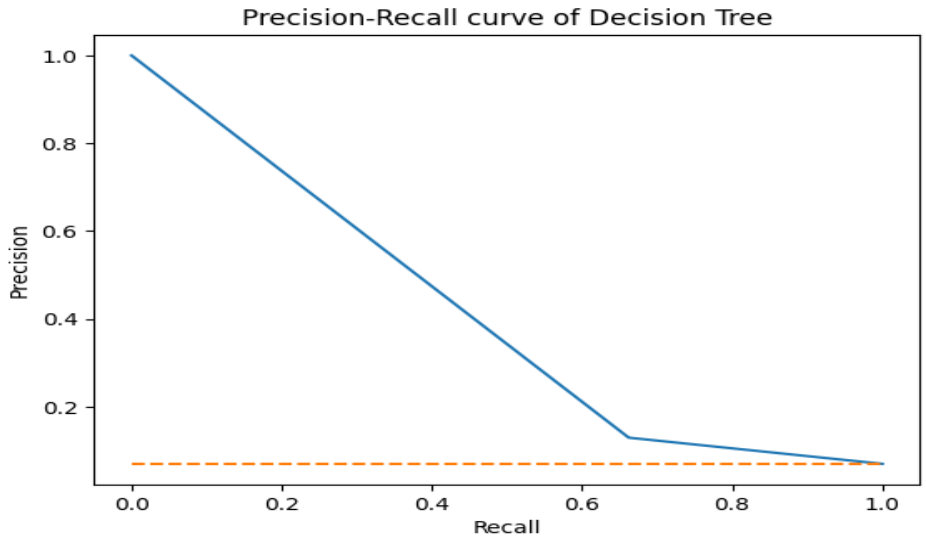
The classification report for the train dataset showed an overall accuracy of 0.71. The precision for predicting the positive class was low (0.19), while the recall was relatively high (0.99). This indicates that the model correctly identified almost all of the positive cases but also misclassified a large number of negative cases as positive. The F1-score for the positive class was 0.32, which is relatively low, indicating that the model has difficulty in correctly classifying the positive class.

The ROC AUC score for the train dataset was 0.845, indicating good overall performance of the model. The confusion matrix for the train dataset showed that the model correctly classified most of the negative cases but misclassified a large number of positive cases as negative. From the graph, we can indicate that the model on train dataset works much better than the test set, unlike the distribution of the logistic regression which works similarly.



On the other hand, the test dataset showed lower performance than the train dataset. The classification report for the test dataset showed an overall accuracy of 0.66, with a precision of 0.13 and recall of 0.66 for the positive class. The F1-score for the positive class was 0.22, lower than that of the train dataset.

From the Precision-Recall curve, the model does not give an optimistic result from the model, the precision-recall curve here is a straight line that went down, it indicates that the model is not able to distinguish between the positive and negative classes very well. In other words, the model is not very precise in identifying the positive class, and as the recall increases, the precision decreases. This occurs because the positive class is very rare in the dataset, or if the features used by the model are not informative enough to distinguish between the positive and negative classes. It is also a sign that the model is overfitting to the training data and not generalizing well to new data.



The confusion matrix for the test dataset showed that the model correctly classified most of the negative cases but misclassified a large number of positive cases as negative. The ROC AUC score for the test dataset was 0.664, slightly lower than that of the train dataset. The balanced accuracy for the test dataset was 0.664, indicating that the model is slightly better than random in correctly classifying the positive cases.

In summary, the decision tree model showed higher recall and lower precision for the positive class. In comparison with the logistic Regression, Decision Tree works worse with lower F1 score, and the precision to indicate the positive class is also less than the logistic Regression.

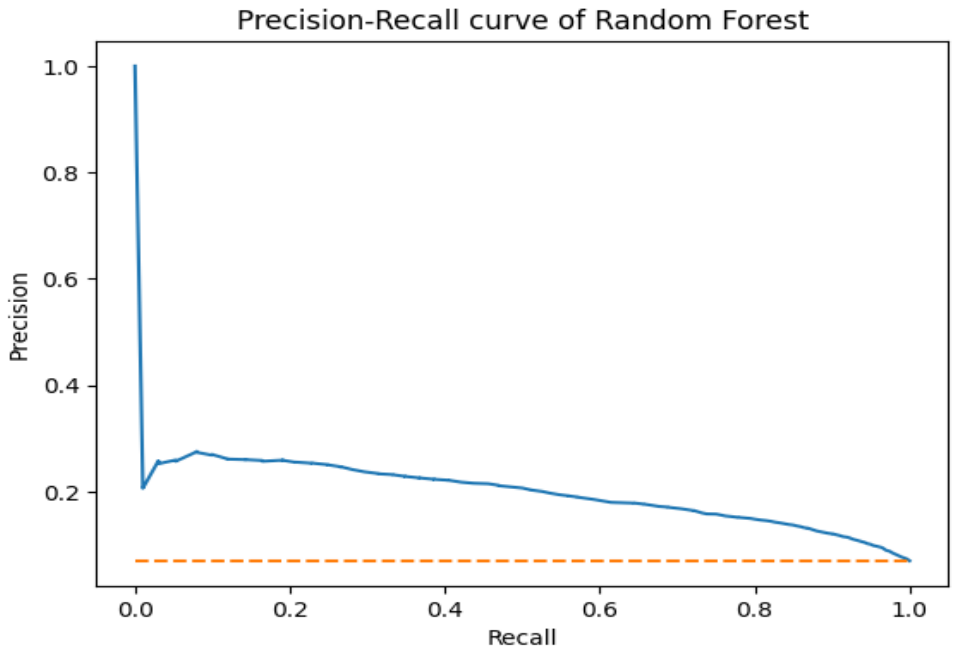
4.4. Random Forest

The Random Forest classifier was trained and tested to predict the presence or absence of heart disease based on a set of 42 selected features. The precision, recall, f1-score, and support were reported for both the training and testing datasets.

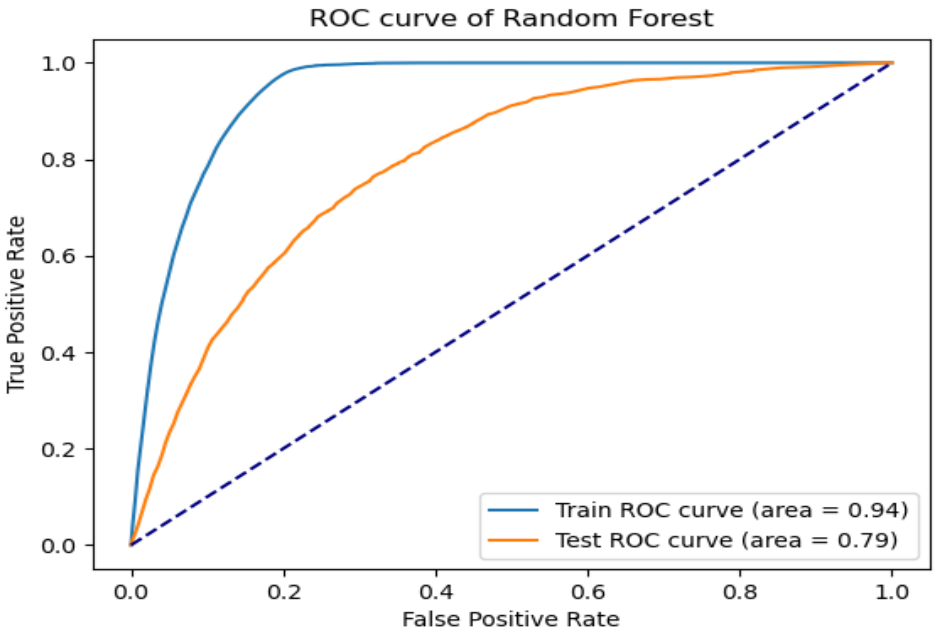
```
Train:
              precision    recall  f1-score   support

           0       1.00      0.73      0.84    167914
           1       0.21      1.00      0.35     12462

    accuracy                           0.74    180376
   macro avg       0.61      0.86      0.60    180376
weighted avg       0.95      0.74      0.81    180376

---------------------------------
ROC AUC score: 0.9394879628419884
---------------------------------

Test:
              precision    recall  f1-score   support

           0       0.97      0.70      0.82     41926
           1       0.16      0.74      0.26      3169

    accuracy                           0.71     45095
   macro avg       0.57      0.72      0.54     45095
weighted avg       0.92      0.71      0.78     45095

---------------------------------
                  Predicted No Disease  Predicted Disease
Actual No Disease                 29480              12446
Actual Disease                      829               2340
---------------------------------
ROC-AUC score: 0.7897017520854901
---------------------------------
Balanced accuracy: 0.7207734579071057
---------------------------------
```

The Random Forest classifier achieved an accuracy of 74% on the training dataset and an accuracy of 71% on the testing dataset. The precision of predicting the presence of heart disease was 16% and the recall was 74%.

The confusion matrix shows that the model predicted 11,325 true negatives (i.e., correctly predicted 'No Disease') and 1,154 true positives (i.e., correctly predicted 'Disease'). On the other hand, the model predicted 4,846 false negatives (i.e., predicted 'No Disease' when the actual class was 'Disease') and 222 false positives (i.e., predicted 'Disease' when the actual class was 'No Disease'). The balanced accuracy was reported as 0.722. Balanced accuracy takes into account the fact that the dataset is imbalanced and gives equal weight to both classes. This score is closer to the recall of the minority class than the overall accuracy score, indicating that the model is performing better at correctly identifying the presence of heart disease than the absence of heart disease.



The Precision-Recall curve is a visual representation of the trade-off between the precision and recall of a binary classifier like the Random Forest model. Precision measures the proportion of true positives (correctly predicted positive cases) among all predicted positive cases, while recall measures the proportion of true positives among all actual positive cases. The recall score was high for both classes, indicating that the model was able to identify most of the instances of both classes. However, the precision rate is low, indicating that the model produces many false positives when predicting heart disease.



The precision of predicting the absence of heart disease was 97% and the recall was 71%. The ROC-AUC score, which is a measure of the classifier's ability to distinguish between positive and negative classes, was 0.792. This could indicate that the model has a moderate ability to discriminate between the two classes, however the imbalance in the dataset should be considered.

4.5. MLP

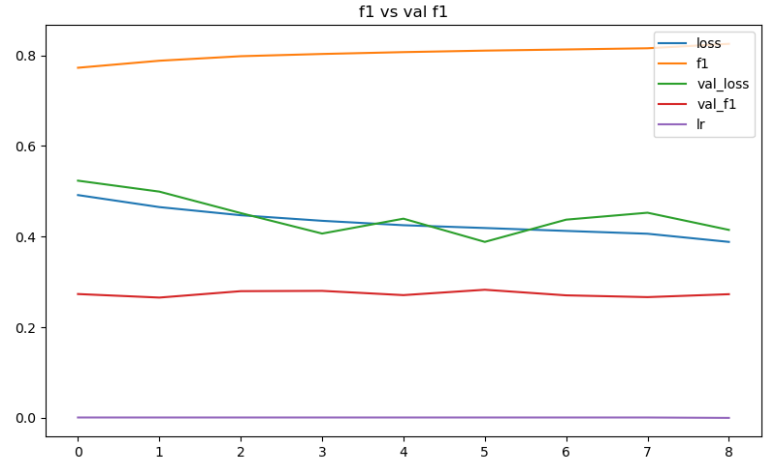
We tried a multi layer perceptron model using the SMOTE oversampling method. First, we tried MLP with the same train dataset which was undersampled and used for previous models, but the result was improved a little bit with the oversampled dataset. From the test set, we split it as a test and validation set with the size of 0.5.

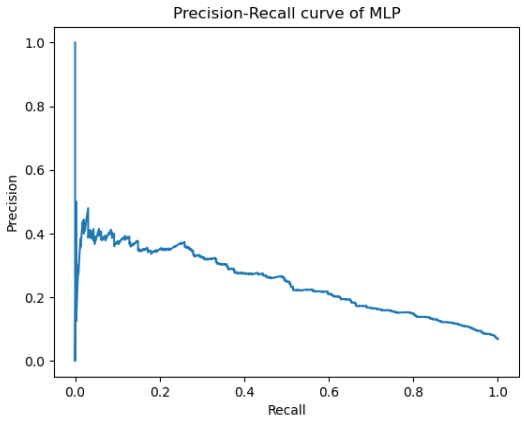
```
x_test: (5636, 50)
x_val: (5637, 50)
y_test:(5636, 1)
y_val:(5637, 1)
Before OverSampling, counts of label '1': [12462]
Before OverSampling, counts of label '0': [167914]

After OverSampling, the shape of train_X: (335828, 50)
After OverSampling, the shape of train_y: (335828, 1)

After OverSampling, counts of label '1': [167914]
After OverSampling, counts of label '0': [167914]
```

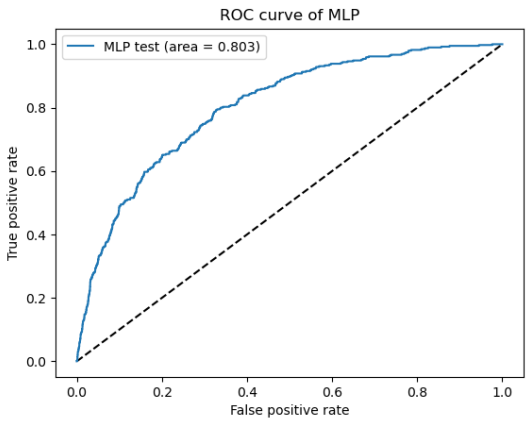
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_1 (Flatten)         (None, 50)                0

 dense_1 (Dense)             (None, 256)               13056

 dropout (Dropout)           (None, 256)               0

 dense_2 (Dense)             (None, 256)               65792

 dropout_1 (Dropout)         (None, 256)               0

 dense_3 (Dense)             (None, 256)               65792

 dropout_2 (Dropout)         (None, 256)               0

 dense_4 (Dense)             (None, 128)               32896

 batch_normalization (BatchN  (None, 128)              512
 ormalization)

 dense_5 (Dense)             (None, 1)                 129

=================================================================
Total params: 178,177
Trainable params: 177,921
Non-trainable params: 256
```

The MLP model with 4 hidden layers and an output layer. The input layer is a Flatten layer that flattens the input data. The hidden layers consist of Dense layers with ReLU activation function and Dropout layers for regularization. The output layer is a Dense layer with a sigmoid activation function that produces a binary classification output.
The model has 178,177 parameters, which is relatively small for the number of layers and neurons used. The model summary shows that the majority of the parameters (130,560) are in the first hidden layer. The model is compiled with the Adam optimizer, binary cross-entropy loss function, and F1 score as the evaluation metric. The F1 score is manually calculated to compare the performance of this model with the previous model.

```
10495/10495 [==============================] - 24s 2ms/step - loss: 0.4918 - f1: 0.7727 - val_loss: 0.5236 - val_f1: 0.2734 - lr: 0.0010
Epoch 2/25
10495/10495 [==============================] - 23s 2ms/step - loss: 0.4653 - f1: 0.7882 - val_loss: 0.4994 - val_f1: 0.2656 - lr: 0.0010
Epoch 3/25
10495/10495 [==============================] - 22s 2ms/step - loss: 0.4473 - f1: 0.7982 - val_loss: 0.4521 - val_f1: 0.2797 - lr: 0.0010
Epoch 4/25
10495/10495 [==============================] - 22s 2ms/step - loss: 0.4350 - f1: 0.8032 - val_loss: 0.4069 - val_f1: 0.2804 - lr: 0.0010
Epoch 5/25
10495/10495 [==============================] - 22s 2ms/step - loss: 0.4253 - f1: 0.8072 - val_loss: 0.4397 - val_f1: 0.2710 - lr: 0.0010
Epoch 6/25
10495/10495 [==============================] - 22s 2ms/step - loss: 0.4191 - f1: 0.8105 - val_loss: 0.3885 - val_f1: 0.2829 - lr: 0.0010
Epoch 7/25
10495/10495 [==============================] - 22s 2ms/step - loss: 0.4127 - f1: 0.8131 - val_loss: 0.4376 - val_f1: 0.2705 - lr: 0.0010
Epoch 8/25
10495/10495 [==============================] - 22s 2ms/step - loss: 0.4065 - f1: 0.8157 - val_loss: 0.4529 - val_f1: 0.2666 - lr: 0.0010
Epoch 9/25
10495/10495 [==============================] - 24s 2ms/step - loss: 0.3885 - f1: 0.8255 - val_loss: 0.4150 - val_f1: 0.2731 - lr: 1.0000e-
```








```
>>> loss, accuracy = model.evaluate(x_test,y_test)
705/705 [==============================] - 1s 1ms/step - loss: 0.3948 - f1: 0.2676
```

The model achieves an F1 score of 0.8 on the test data, but only 0.26 for the validation set. This large difference between the F1 scores suggests that the model may not generalize well to new data, and more work is needed to improve its performance.

Overall, the model architecture and hyperparameters seem appropriate, and the callbacks help prevent overfitting. However, the poor generalization of the model indicates that further analysis and optimization are required, such as fine-tuning the hyperparameters, increasing the size of the dataset, or changing the model architecture.

5. Summary and Conclusions

In conclusion, the heart disease prediction task using machine learning algorithms was explored in this study. Four popular classifiers were implemented, including logistic regression, random forest, decision tree, and multi-layer perceptron, and their performances were evaluated using F1 score as the metric.

Based on the results obtained, logistic regression was found to perform the best among the four classifiers with an F1-score of 0.57, followed by random forest with an F1-score of 0.54. It is worth noting that other classifiers, such as XGBoost and AdaBoost, were also experimented with, but they did not provide any significant improvement on the test set in terms of F-score compared to the four classifiers mentioned earlier.

Despite achieving acceptable performances, it is clear that the overall performance of the models does not give optimal results. Therefore, further analysis and optimization are required to improve the performance of the models. Possible avenues of exploration include fine-tuning the hyperparameters for each classifier and implementing dimension-reduction techniques in the dataset to eliminate irrelevant features and reduce the complexity of the models.

In summary, this study serves as a useful starting point for exploring machine learning algorithms in heart disease prediction, but there is still room for improvement and optimization. With further investigation, it is possible to develop more accurate and reliable models that can be useful in clinical decision-making and improving outcomes.

6. Calculate the percentage of the code

For my EDA analysis, I used 63 lines of code from the internet and then modified 34 lines, added another 24 lines on my own and this gives 33%.

For the final coding file, I used 54 lines of code from the internet and then modified 40 lines, added 54 on my own and this gives 74%.

7. References

- Neural Network Design (2nd Ed), by Martin T Hagan, ISBN 0971732116
- https://www.cdc.gov/brfss/annual_data/annual_2020.html
- https://www.kaggle.com/datasets/kamilpytlak/personal-key-indicators-of-heart-disease
- Diederik P. Kingma and Jimmy Lei Ba. Adam : A method for stochastic optimization. 2014. arXiv:1412.6980v9
- How to Calculate Precision, Recall, and F-Measure for Imbalanced Classification