DATS 6202

Machine Learning I

Prof. Amir Jafari

Heart Disease Prediction

TEAM 8

Guoshan Yu, Jiwoo Suh, Kyuri Kim

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.

With recent advancements in computational developments, machine learning methods can now be applied to detect patterns from large datasets that can predict a patient's condition, including the likelihood of heart disease. 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 report 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. The report will also investigate which variables have a significant effect on the likelihood of heart disease.

In the following sections, we will provide more detailed information on the dataset, including the variables that were selected for analysis. We will also describe the machine learning methods that were applied and discuss the results obtained from these methods. Finally, we will present our conclusions and recommendations for further research in this area.

2. Description of Dataset

2.1. Data Information

The dataset was originally obtained from the Centers for Disease Control and Prevention (CDC) and is a part of the Behavioral Risk Factor Surveillance System (BRFSS), which conducts annual telephone surveys to gather data on the health status of U.S. residents. The dataset contains data from 2020 and consists of 401,958 rows and 279 columns, with most columns representing questions asked to respondents about their health status. The dataset was reduced to approximately 20 variables relevant to heart disease, which is treated as a binary variable. "Yes" represents respondents who reported having coronary heart disease (CHD) or myocardial infarction (MI), while "No" represents respondents who reported not having heart disease.

2.2. Variables

- HeartDisease: This variable indicates whether the respondent has ever reported having coronary heart disease (CHD) or myocardial infarction (MI). CHD is a condition where plaque builds up inside the coronary arteries, which supply blood to the heart. MI, also known as a heart attack, occurs when blood flow to a part of the heart is blocked for a long enough time that part of the heart muscle is damaged or dies.

- BMI: Body Mass Index (BMI) is a measure of body fat based on height and weight. It is calculated by dividing a person's weight in kilograms by their height in meters squared. A BMI of less than 18.5 is considered underweight, a BMI between 18.5 and 24.9 is considered normal weight, a BMI between 25 and 29.9 is considered overweight, and a BMI of 30 or higher is considered obese.

- Smoking: This variable indicates whether the respondent has smoked at least 100 cigarettes in their entire life. This is a commonly used criterion to define smoking status in epidemiological studies.

- AlcoholDrinking: This variable indicates whether the respondent is a heavy drinker. Heavy drinking is defined as consuming more than 14 drinks per week for adult men and more than 7 drinks per week for adult women.

- Stroke: This variable indicates whether the respondent has ever been told by a doctor or other health professional that they have had a stroke. A stroke occurs when blood flow to the brain is blocked or reduced, which can cause brain damage and other complications.

- PhysicalHealth: This variable asks the respondent to rate their physical health over the past 30 days. Respondents are asked to indicate the number of days during the past 30 days when their physical health was not good.

- MentalHealth: This variable asks the respondent to rate their mental health over the past 30 days. Respondents are asked to indicate the number of days during the past 30 days when their mental health was not good.

- DiffWalking: This variable indicates whether the respondent has serious difficulty walking or climbing stairs. This is often used as a measure of mobility impairment.

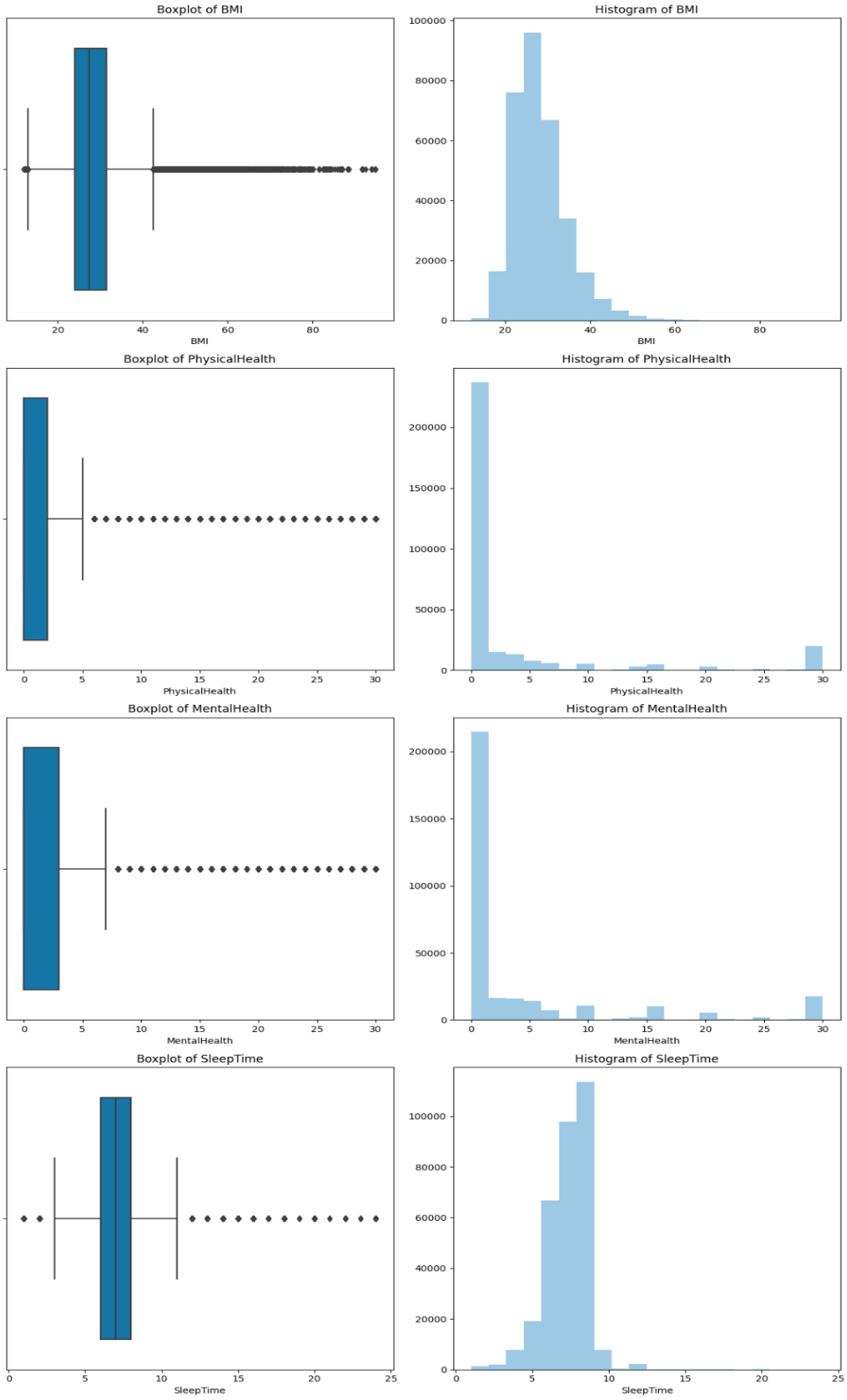
- Sex: This variable asks the respondent to indicate whether they are male or female.

- AgeCategory: This variable categorizes respondents into fourteen age groups, based on their age in years. The age groups are as follows: 18-24, 25-29, 30-34, 35-39, 40-44, 45-49, 50-54, 55-59, 60-64, 65-69, 70-74, 75-79, 80-84, and 85 or older.

- Race: This variable asks the respondent to indicate their race/ethnicity.
- Diabetic: This variable indicates whether individuals were ever diagnosed with diabetes.
- Physical Activity: This variable indicates whether adults engaged in physical activity or exercise, aside from their regular job, within the past 30 days.
- General Health: Participants were asked to rate their overall health, and their responses were categorized into different levels (e.g., "Good," "Fair").
- Sleep Time: Participants reported the average number of hours of sleep they obtained within a 24-hour period.
- Asthma: This variable indicates whether individuals were ever diagnosed with asthma.
- Kidney Disease: This variable indicates whether individuals were ever diagnosed with kidney disease, excluding kidney stones, bladder infections, or incontinence.
- Skin Cancer: This variable indicates whether individuals were ever diagnosed with skin cancer.

2.3. EDA Analysis

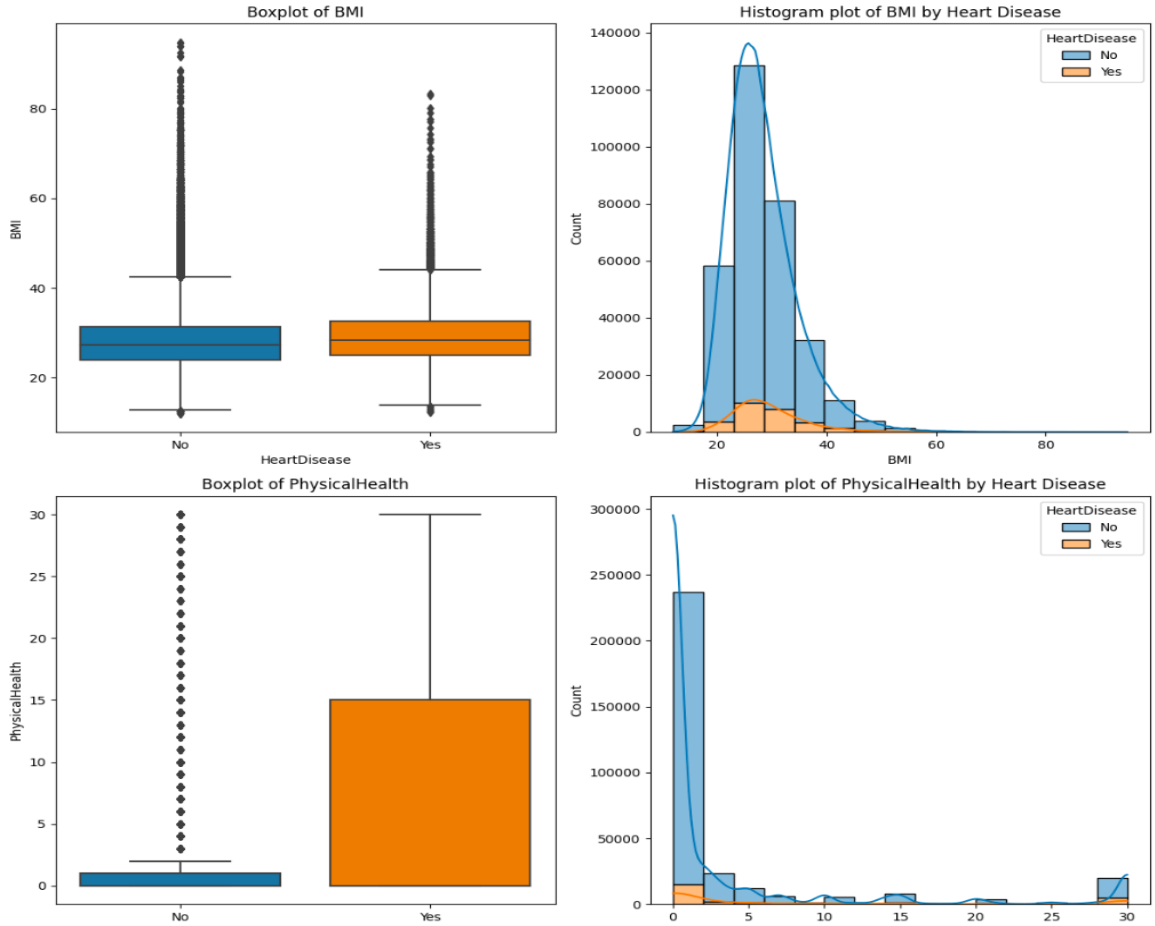
2.3.1.Univariate Analysis - Numerical features

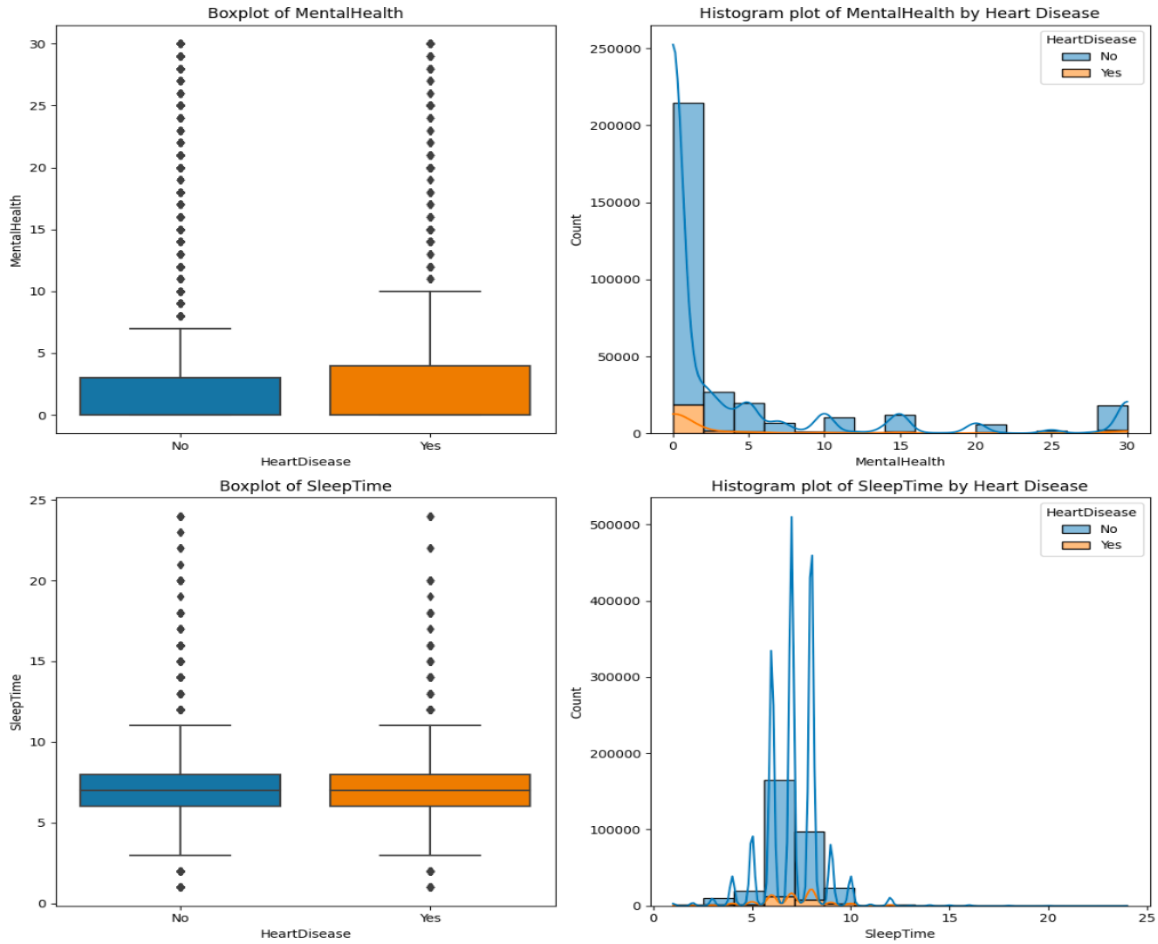


Exploratory Data Analysis (EDA) is an important first step in any data analysis project. In this case, we will be examining the four numerical variables in our dataset: BMI, PhysicalHealth, MentalHealth, and SleepTime. Looking at the boxplots, we can see that Physical Health and Mental Health have outliers, as indicated by the dots beyond the whiskers. From the histograms, we can see that all four variables are not normally distributed. BMI is skewed to the right, indicating that mean is greater than median. The rest of the variables are close to bimodal. In summary, the numerical variables in the dataset appear to be not normally distributed, with the right skewness present, or are bimodal distributions. Also, they have outliers, which may need to be examined further.

2.3.2. Multivariate Analysis - Numerical features

In this analysis, we compare the four numerical variables with respect to the target variable, HeatDisease. Histograms struggle to highlight differences between the two groups due to dataset imbalance. Regardless of whether an individual has heart disease or not, both groups exhibit similar patterns and average values for each variable. The box plots effectively illustrate this similarity.

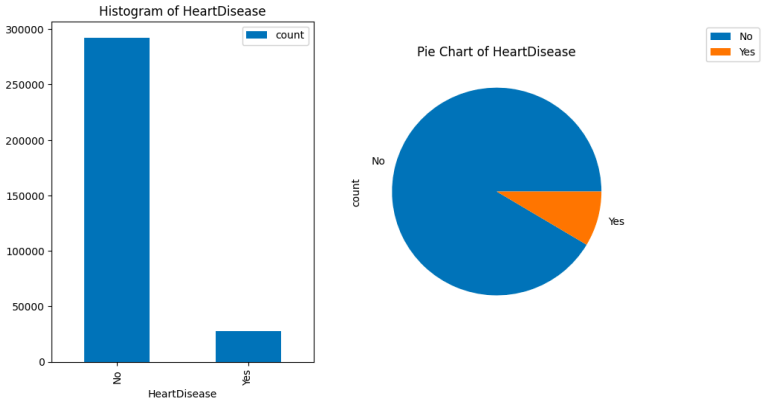




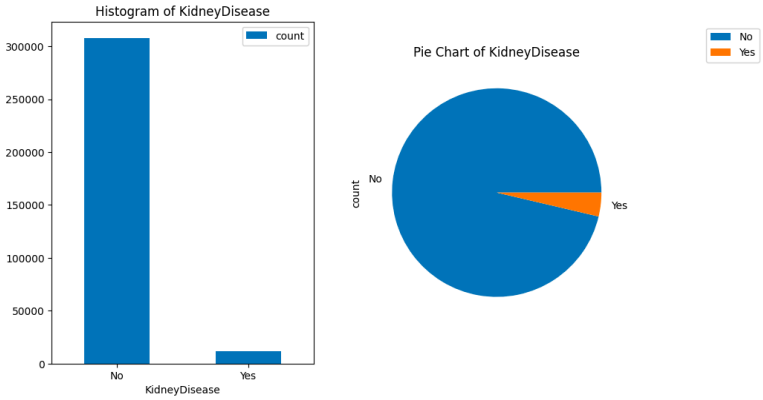
2.3.3. Univariate Analysis - Categorical features

Based on the value counts of the categorical variables, we can observe that the majority of the participants do not have heart disease, which makes the dataset imbalanced. For stroke, difficulty walking, diabetes, asthma, kidney disease, or skin cancer, the majority of respondents do not have them. In terms of alcohol drinking, the majority of participants do not drink alcohol. For smoking, more than 40% of respondents smoke. The data shows that there are more female participants than male participants, and the age distribution is relatively evenly spread among the different age categories. Finally, the majority of participants report engaging in physical activity, have good to very good general health, and are of White race, indicating that the dataset is imbalanced.

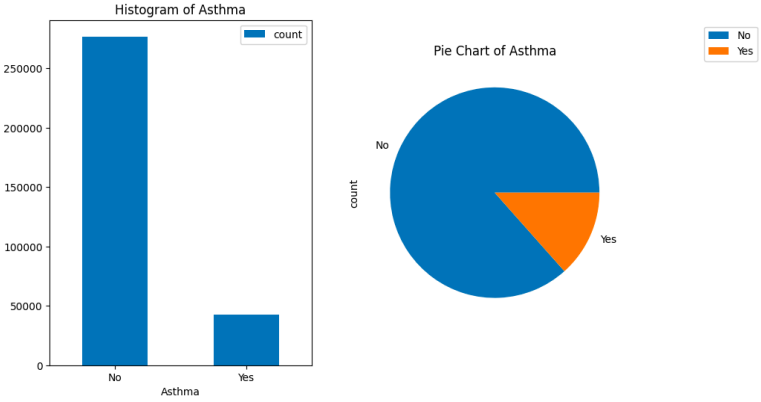
HeartDisease



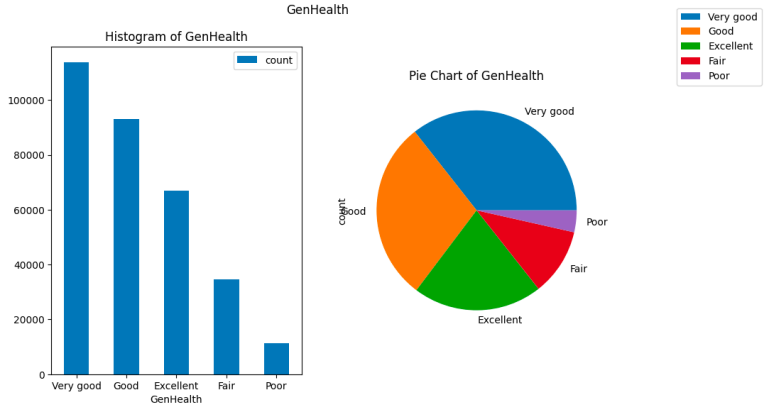
KidneyDisease



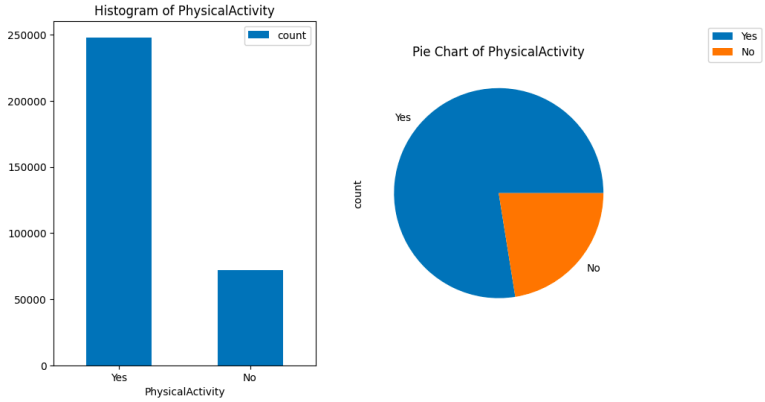
Asthma



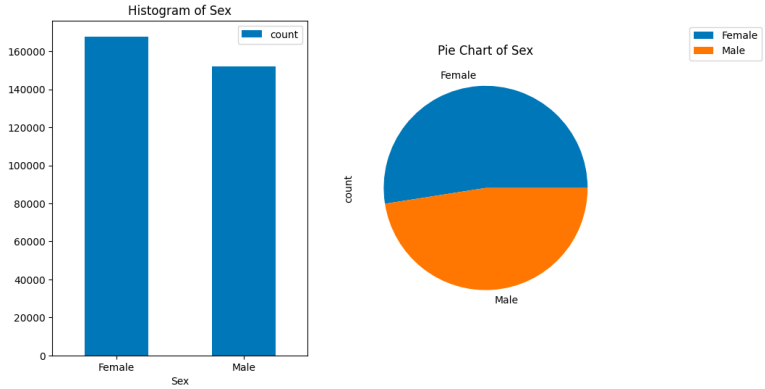
GenHealth



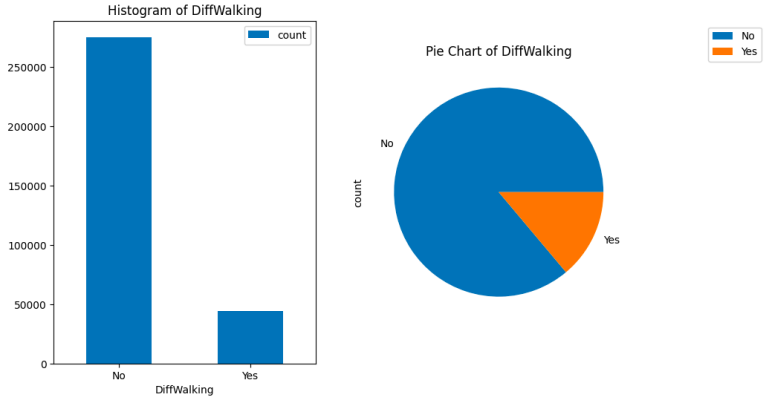
PhysicalActivity



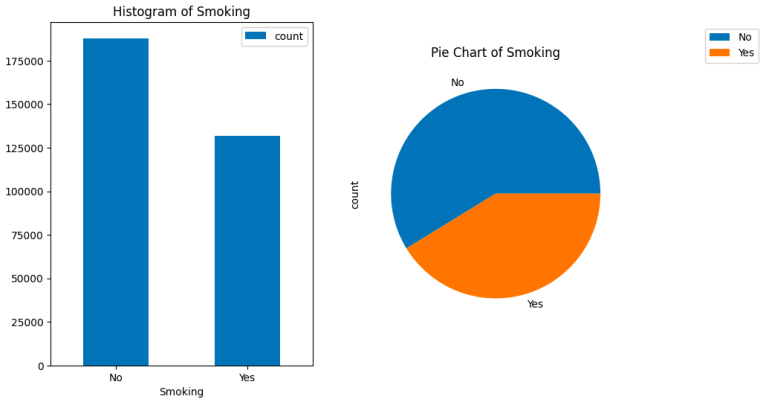
Sex



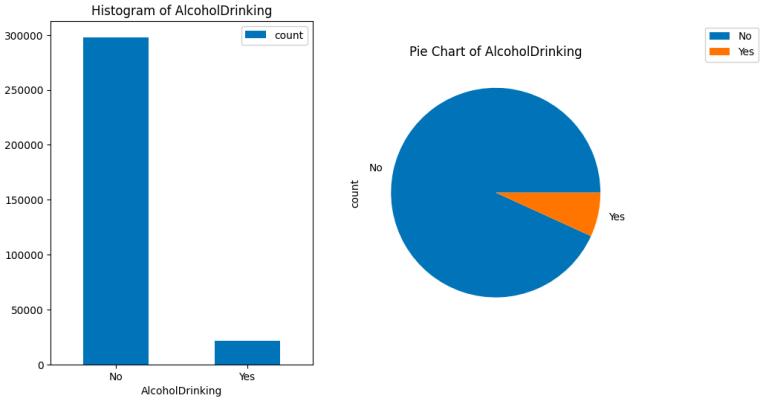
DiffWalking



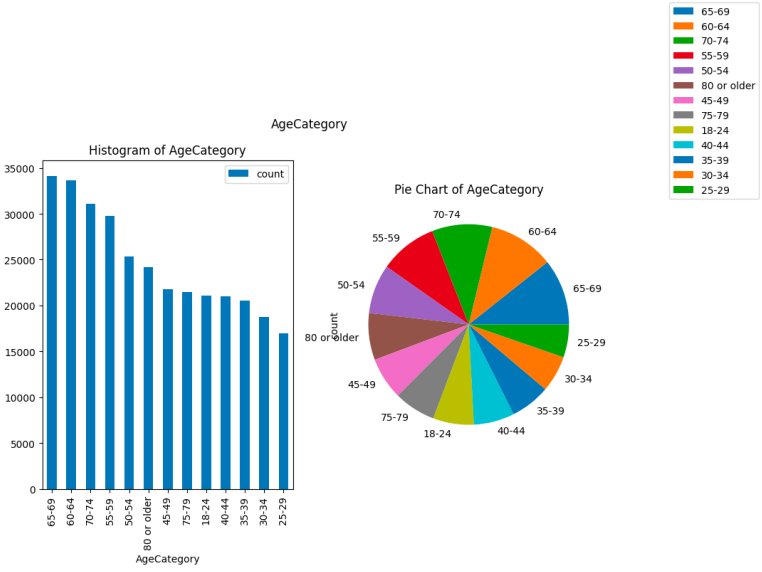
Smoking

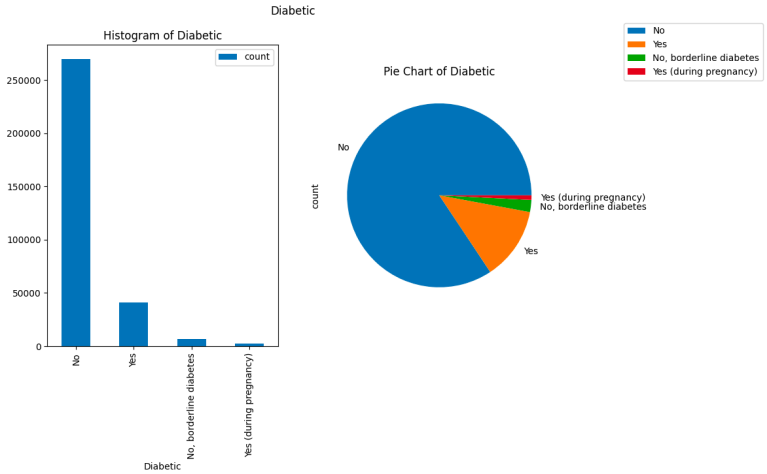


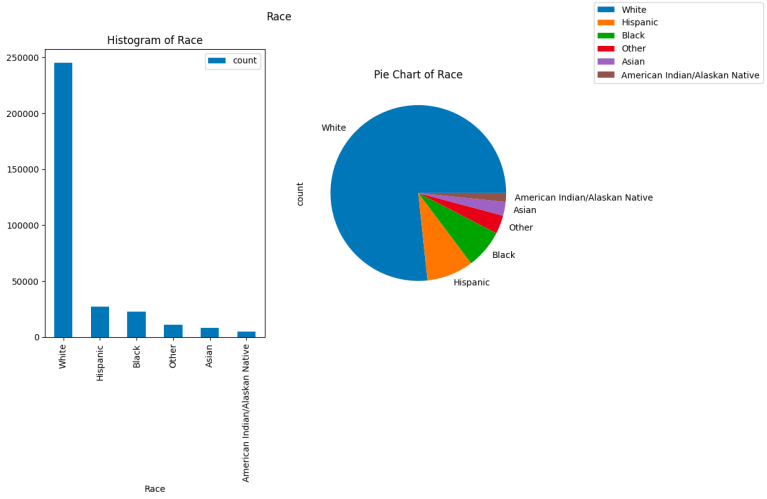
AlcoholDrinking



AgeCategory

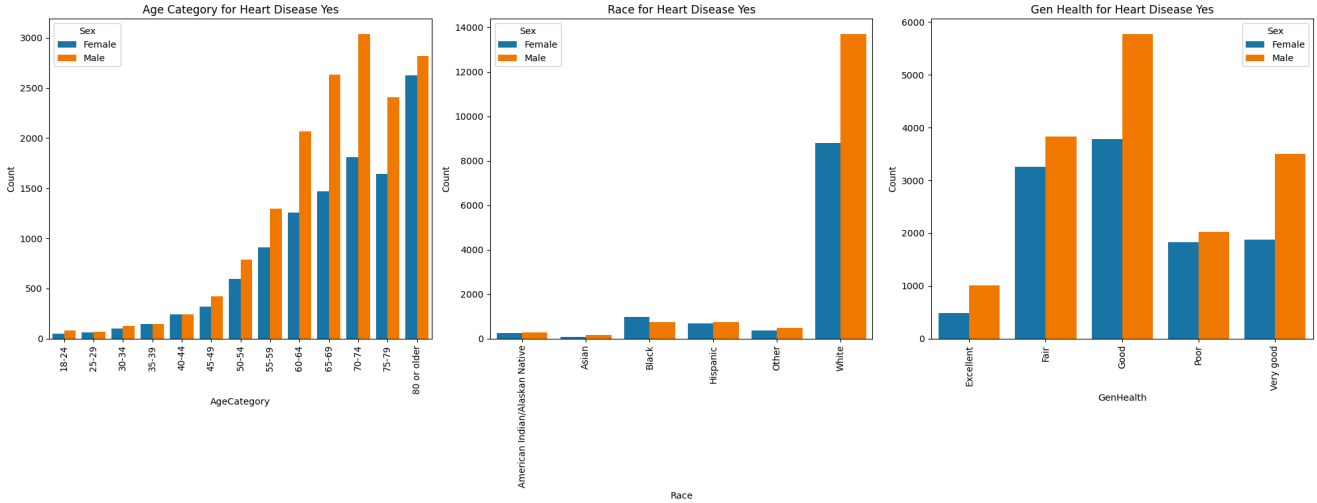



Diabetic
Histogram of Diabetic
count
250000
200000
150000
100000
50000
0
No
Yes
No, borderline diabetes
Yes (during pregnancy)
Diabetic
Pie Chart of Diabetic
No
Yes
No, borderline diabetes
Yes (during pregnancy)
count

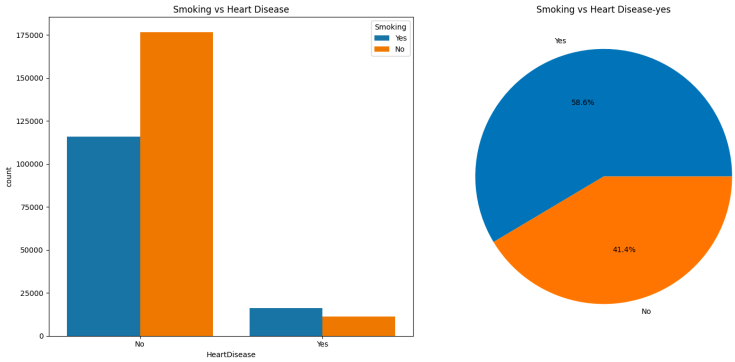

Race
Histogram of Race
count
250000
200000
150000
100000
50000
0
White
Hispanic
Black
Other
Asian
American Indian/Alaskan Native
Race
Pie Chart of Race
White
Hispanic
Black
Other
Asian
American Indian/Alaskan Native
count

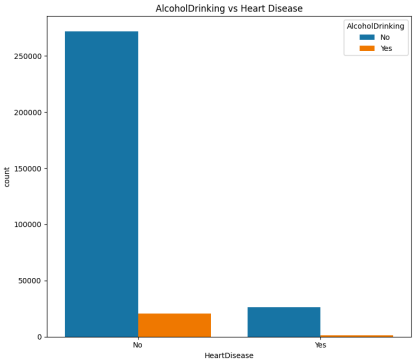
2.3.4. Multivariate Analysis - Categorical feature

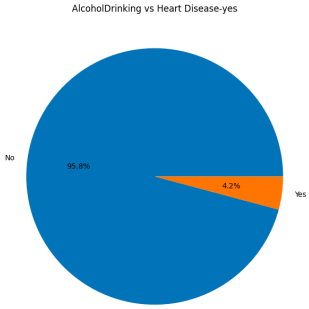
The analysis of the three histograms reveals patterns regarding Age, Race, GeneralHealth and their correlation with sex & heart disease. In terms of age and sex, the data illustrates that among males with heart disease, the age group 70-74 exhibits the highest proportion, followed by the age group 80 or older. Conversely, among females with heart disease, the age group 80 or older has the highest representation, followed by the 70-74 age group. Regarding race, the data indicates that males have a higher likelihood of developing heart disease across all racial groups, except for Black individuals. Finally, when examining general health, the histogram demonstrates that individuals with heart disease predominantly report having "Good" health conditions, followed by those who perceive their health as "Fair."

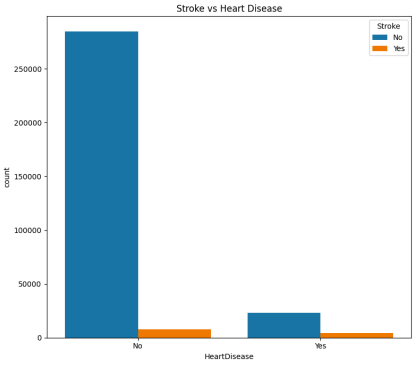


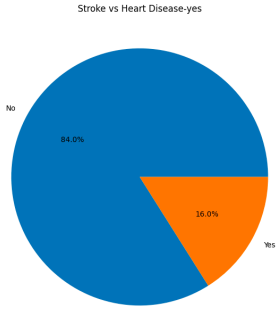
While data imbalance makes it challenging to discern patterns, the plots indicate that individuals who smoke and engage in physical activity are more likely to have heart disease. Conversely, alcohol drinking, stroke, difficulty walking, diabetes, asthma, kidney disease, and skin cancer do not seem to significantly impact the risk of heart disease. Moreover, the data reveals that males have a higher propensity for heart disease, with the age group 80 or older and White individuals exhibiting a higher portion of heart disease. Lastly, respondents who reported "Good" health conditions also have a higher likelihood of developing heart disease.

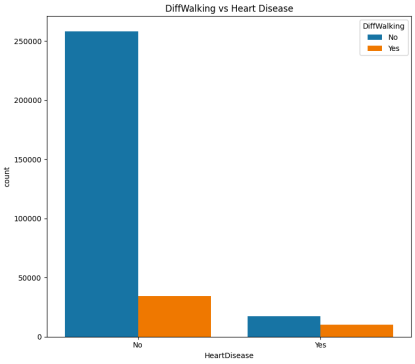


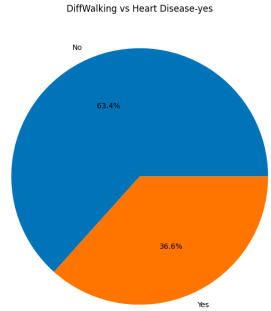

AlcoholDrinking vs Heart Disease
AlcoholDrinking
No
Yes
count
HeartDisease

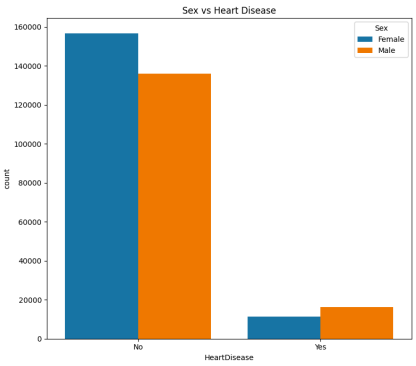

AlcoholDrinking vs Heart Disease-yes
No
95.8%
4.2%
Yes

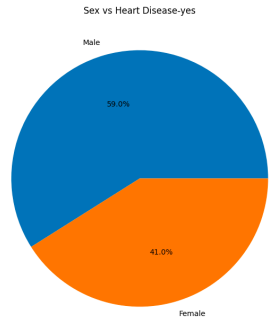

Stroke vs Heart Disease
Stroke
No
Yes
count
HeartDisease

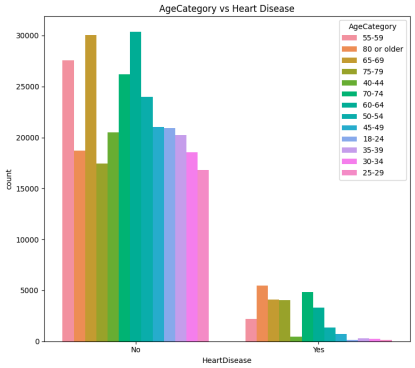

Stroke vs Heart Disease-yes
No
84.0%
16.0%
Yes

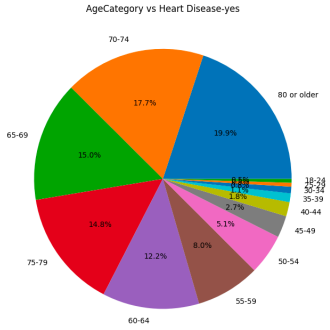

DiffWalking vs Heart Disease
DiffWalking
No
Yes
count
HeartDisease

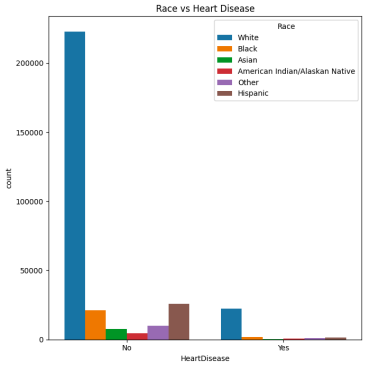

DiffWalking vs Heart Disease-yes
No
63.4%
36.6%
Yes

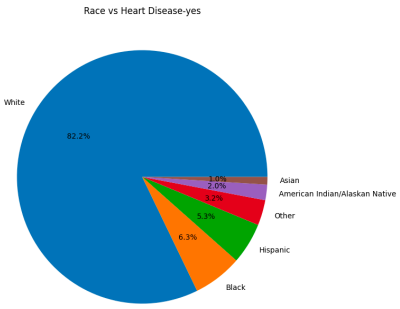

Sex vs Heart Disease
Sex
Female
Male
count
HeartDisease

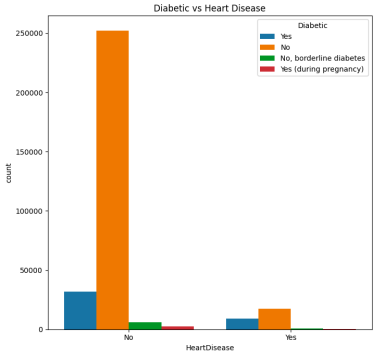

Sex vs Heart Disease-yes
Male
59.0%
41.0%
Female

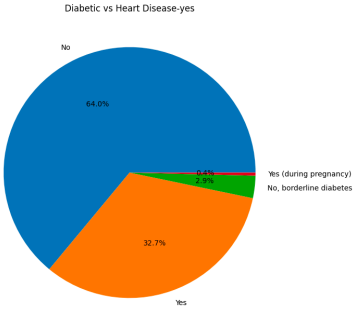

AgeCategory vs Heart Disease
AgeCategory
55-59
80 or older
65-69
75-79
40-44
70-74
60-64
50-54
45-49
18-24
35-39
30-34
25-29
count
HeartDisease
No
Yes

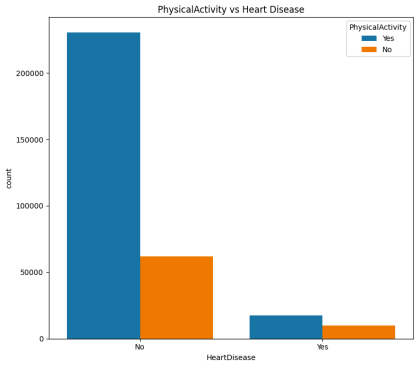

AgeCategory vs Heart Disease-yes
70-74
80 or older
65-69
17.7%
19.9%
15.0%
14.8%
12.2%
8.0%
5.1%
2.7%
1.8%
75-79
60-64
55-59
50-54
45-49
40-44
35-39
18-24

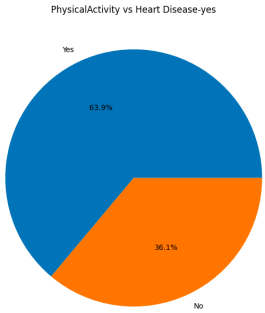

Race vs Heart Disease
Race
White
Black
Asian
American Indian/Alaskan Native
Other
Hispanic
count
HeartDisease
No
Yes

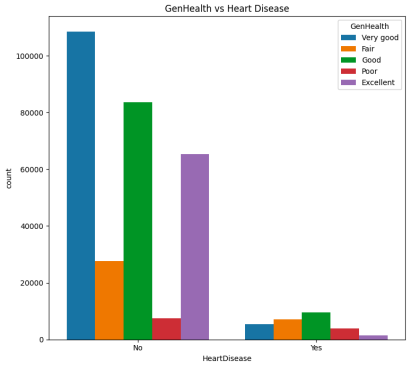

Race vs Heart Disease-yes
White
82.2%
Asian
American Indian/Alaskan Native
Other
Hispanic
Black
1.0%
2.0%
3.2%
5.3%
6.3%

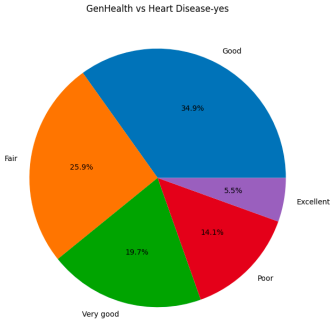

Diabetic vs Heart Disease
Diabetic
Yes
No
No, borderline diabetes
Yes (during pregnancy)
count
HeartDisease
No
Yes

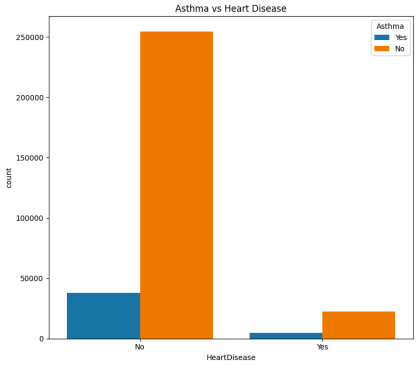

Diabetic vs Heart Disease-yes
No
64.0%
Yes (during pregnancy)
No, borderline diabetes
0.4%
2.9%
32.7%
Yes

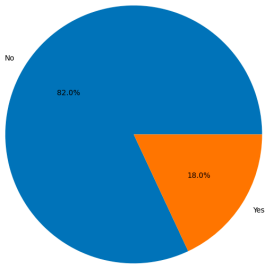

PhysicalActivity vs Heart Disease
PhysicalActivity
Yes
No
count
HeartDisease
No
Yes

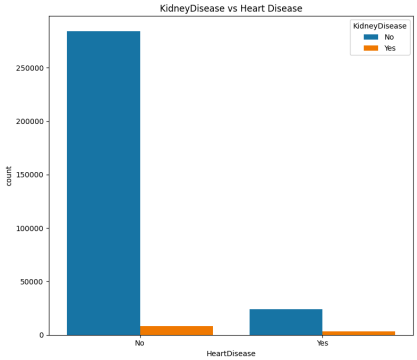

PhysicalActivity vs Heart Disease-yes
Yes
63.9%
36.1%
No

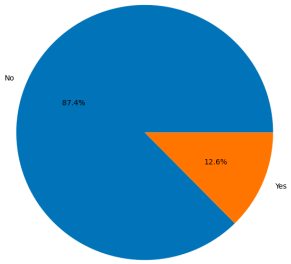

GenHealth vs Heart Disease
GenHealth
Very good
Fair
Good
Poor
Excellent
count
HeartDisease
No
Yes

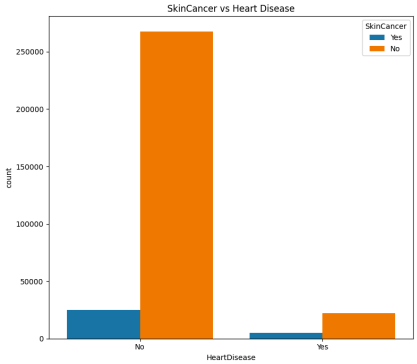

GenHealth vs Heart Disease-yes
Good
34.9%
Fair
25.9%
Excellent
5.5%
14.1%
19.7%
Poor
Very good

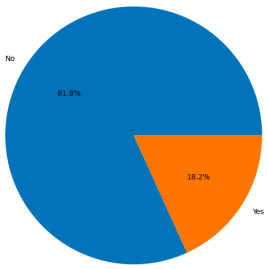

Asthma vs Heart Disease
Asthma
Yes
No
count
HeartDisease
No
Yes


Asthma vs Heart Disease-yes
No
82.0%
18.0%
Yes


KidneyDisease vs Heart Disease
KidneyDisease
No
Yes
count
HeartDisease
No
Yes


KidneyDisease vs Heart Disease-yes
No
87.4%
12.6%
Yes


SkinCancer vs Heart Disease
SkinCancer
Yes
No
count
HeartDisease
No
Yes


SkinCancer vs Heart Disease-yes
No
81.8%
18.2%
Yes

3. Description of the Machine Learning Algorithms

3.1. Logistic Regression

Logistic Regression is a powerful and widely used classification algorithm that can be used for both binary and multi-class classification problems. It is a parametric algorithm that works by finding the best fit of a sigmoid function to the training data. The sigmoid function transforms a linear equation into a probability between 0 and 1, which can be interpreted as the likelihood of a particular class given a set of input features.

The logistic regression model is trained using the maximum likelihood estimation method, which involves finding the parameter values that maximize the likelihood of observing the training data. The parameter values are updated iteratively until convergence is achieved. The most commonly used optimization algorithm for logistic regression is the gradient descent algorithm.

The decision boundary for logistic regression is a linear hyperplane that separates the different classes in the feature space. The logistic regression algorithm is sensitive to the scaling of the input features, and it is important to normalize or standardize the input features before training the model.

Logistic regression is a simple yet powerful algorithm that can achieve high accuracy on a wide range of classification problems. It is widely used in various fields such as finance, healthcare, and social sciences. However, it may not perform well in cases where the decision boundary is non-linear or where there are complex interactions between the input features.

3.2. Decision Tree

Decision Tree is a type of non-parametric supervised learning algorithm used for both classification and regression tasks. The algorithm works by recursively partitioning the data into subsets, with the goal of maximizing the information gain at each partition. The algorithm begins by selecting the feature that provides the maximum information gain as the root node, and then splits the data based on the values of that feature. This process is repeated for each subsequent node, with the goal of creating a tree that accurately predicts the target variable.

The decision tree algorithm uses the Gini Index or Entropy to determine the quality of the split. Gini Index is a measure of impurity used to evaluate the splits.

The decision tree algorithm can suffer from overfitting, where the tree becomes too complex and fits the noise in the data. To prevent overfitting, we can prune the tree by removing nodes that do not improve the overall accuracy of the tree.

3.3. Random Forest

Random Forest is a popular ensemble learning algorithm that combines multiple decision trees to improve the overall performance and reduce overfitting. The algorithm works by creating a set of decision trees, where each tree is trained on a random subset of the training data and a random subset of the features.

During training, the algorithm creates decision trees using a technique called bootstrapped sampling, which involves randomly selecting samples with replacement from the training data. This technique helps to create a diverse set of decision trees that are less likely to overfit the training data.

To make a prediction, the algorithm uses the ensemble of decision trees to calculate the average of the predictions made by each individual tree. This approach helps to reduce the variance and improve the overall accuracy of the model.

The equation for the random forest algorithm can be summarized as:

- For each tree in the forest:
  - Randomly select a subset of the training data (with replacement).
  - Randomly select a subset of the features.
  - Train a decision tree on the selected data and features.
- To make a prediction:
  - Pass the input data through each of the decision trees in the forest.
  - Calculate the average of the predictions made by each tree.
  - Return the final prediction.

Random Forest is a powerful algorithm that can be used for both classification and regression tasks. It has several hyperparameters that can be tuned to optimize its performance, such as the number of trees in the forest, the maximum depth of each tree, and the number of features to consider when splitting a node.

3.4. Multi Layer Perceptron

The Multi-Layer Perceptron (MLP) is a type of artificial neural network that consists of multiple layers of interconnected nodes or neurons. Each neuron in a layer receives input from the neurons in the previous layer and processes the input using an activation function to produce an output.

The MLP can be used for both classification and regression tasks. For classification tasks, the output layer of the MLP consists of multiple neurons, each corresponding to a different class label. The output of the MLP is a probability distribution over the class labels, and the class with the highest probability is predicted as the output.

The training of the MLP is performed using a variant of stochastic gradient descent called backpropagation. The goal of the training process is to adjust the weights of the connections between the neurons in order to minimize the difference between the predicted output and the actual output.

The equations for forward propagation and backpropagation in the MLP are as follows:

Forward Propagation:

$$\mathbf{a}^0 = \mathbf{p},$$

$$\mathbf{a}^{m+1} = \mathbf{f}^{m+1}(\mathbf{W}^{m+1}\mathbf{a}^m + \mathbf{b}^{m+1}) \text{ for } m = 0, 1, \ldots, M-1,$$

$$\mathbf{a} = \mathbf{a}^M.$$

Backpropagation:

$$\mathbf{s}^M = -2\dot{\mathbf{F}}^M(\mathbf{n}^M)(\mathbf{t} - \mathbf{a}),$$

$$\mathbf{s}^m = \dot{\mathbf{F}}^m(\mathbf{n}^m)(\mathbf{W}^{m+1})^T\mathbf{s}^{m+1}, \text{ for } m = M-1, \ldots, 2, 1,$$

where

$$\dot{\mathbf{F}}^m(\mathbf{n}^m) = \begin{bmatrix} \dot{f}^m(n_1^m) & 0 & \ldots & 0 \\ 0 & \dot{f}^m(n_2^m) & \ldots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & & \dot{f}^m(n_{S^m}^m) \end{bmatrix},$$

$$\dot{f}^m(n_j^m) = \frac{\partial f^m(n_j^m)}{\partial n_j^m}.$$

The MLP is a powerful machine learning algorithm that can learn complex non-linear relationships between inputs and outputs. However, it is also prone to overfitting and requires careful tuning of the hyperparameters to achieve optimal performance.

4. Experimental setup

From the Experimental Setup, we are following the steps as follows:

4.1. Data Preprocessing

4.1.1.Remove the outliers

Identifying and handling outliers is an essential step in data preprocessing that involves assessing the quality and accuracy of the dataset. In the data preprocessing step, we followed a standard approach to identify and remove the outliers using the Interquartile Range (IQR) method.

First, we calculated the first and third quartiles (Q1 and Q3) for each numerical feature in the dataset. The interquartile range (IQR) is then computed as the difference between Q3 and Q1. This method ensures that we capture the variability of the central 50% of the data, avoiding the skewness that can result from using the mean and standard deviation as measures of variability.

To identify outliers in each feature, I used the upper and lower bounds, calculated as Q3 + 1.5 * IQR and Q1 - 1.5 * IQR, respectively. Any data point that falls outside these bounds is considered an outlier and must be removed or transformed.

To handle the outliers, I opted to remove them from the dataset by creating a function that iterates over each numerical feature and removes the rows containing outliers using boolean indexing. I stored the new dataset with outliers removed in a new variable named `df_clean`, to ensure that the original dataset is not altered.

The graph below is the outliers information from the Dataset:

```
feature: BMI
Q1: 24.03
Q3: 31.42
IQR: 7.390000000000001
number of outliners in upper bound:10351
number of outliners in lower bound:45
---------------------------------
feature: PhysicalHealth
Q1: 0.0
Q3: 2.0
IQR: 2.0
number of outliners in upper bound:47146
number of outliners in lower bound:0
---------------------------------
feature: MentalHealth
Q1: 0.0
Q3: 3.0
IQR: 3.0
number of outliners in upper bound:51576
number of outliners in lower bound:0
---------------------------------
feature: SleepTime
Q1: 6.0
Q3: 8.0
IQR: 2.0
number of outliners in upper bound:3204
number of outliners in lower bound:1339
---------------------------------
```

4.1.2. Encoding the Data

Following the removal of outliers from the numerical features in the dataset, the categorical variables were encoded using one-hot encoding to convert them into a format that can be used in machine learning algorithms.

To encode the target variable, `y_clean` was replaced with a new variable `y_encoded`, which was generated using the `replace()` method. Specifically, the `HeartDisease` column in `y_clean` was replaced with numerical values, such that 'Yes' was replaced with 1 and 'No' with 0. This binary transformation was necessary to ensure that the target variable was in a numerical format suitable for machine learning.

Next, the `get_dummies()` function was applied to the categorical variables in `x_clean` to perform one-hot encoding. This transformation created new columns for each unique category within the categorical variables, with a value of 1 indicating the presence of that category and 0 indicating its absence.

The use of one-hot encoding ensured that the categorical variables were encoded in a way that could be easily used in machine learning models. After the encoding, the size of the features increased to 50.

4.1.3. Standardized the Data

After encoding the categorical variables using one-hot encoding, the continuous features in the dataset were normalized using the `MinMaxScaler` from the `sklearn.preprocessing` module.

The `MinMaxScaler` scales the values of each feature to be within the range of 0 to 1, where 0 represents the minimum value in the feature and 1 represents the maximum value. This scaling is necessary to ensure that all features are on a similar scale, preventing the influence of one feature from dominating the others.

The `fit_transform()` method was used to both fit the scalar to the dataset and transform the continuous features in `x_encoded`. By normalizing the continuous features, it ensured that they were on a comparable scale, enabling the machine learning algorithm to more effectively learn from the data.

4.1.4. Split the Data

After encoding and scaling the dataset, the data was split into training and testing sets using the `train_test_split()` function from the `sklearn.model_selection` module.

The `train_test_split()` function takes in the scaled input features `x_scaled` and the encoded target variable `y_encoded`, and splits them into two sets of data: one for training the machine learning model, and another for testing its performance.

The `test_size` parameter was set to 0.2, indicating that 20% of the data should be used for testing, while the remaining 80% should be used for training. The `random_state` parameter was set to 42, ensuring that the same random split is used each time the code is run, allowing for reproducibility of the results.

The resulting four arrays were assigned to the variables `x_train`, `x_test`, `y_train`, and `y_test`, respectively. The `x_train` and `y_train` arrays were used to train the machine learning model, while the `x_test` and `y_test` arrays were used to evaluate its performance.

4.2. Implement Pipeline

The code starts with defining a dictionary of three popular classifiers: Decision Tree, Random Forest, and Logistic Regression. The purpose of this dictionary is to iterate over each of these classifiers and evaluate their performance on the dataset.

4.2.1.Balance the Data - Undersampling

For each classifier, an undersampling technique called *RandomUnderSampling* is applied to balance the dataset, which means reducing the number of samples from the majority class. The aim of this technique is to address the class imbalance problem in the dataset, where the number of samples in one class is much higher than the other.

```
Before Undersampling, counts of label '1': [12462]
Before Undersampling, counts of label '0': [167914]

After Undersampling, the shape of train_X: (24924, 50)
After Undersampling, the shape of train_y: (24924, 1)

After Undersampling, counts of label '1': [12462]
After Undersampling, counts of label '0': [12462]
```

4.2.2.Feature Selection - RFECV

After applying the undersampling technique, a feature selection technique called Recursive Feature Elimination with Cross-Validation (RFECV) is used to select the most important features for the classifier. RFECV is a wrapper method that works by recursively removing features, fitting the model, and evaluating performance until the optimal number of features is reached. The number of features that result in the best performance is determined by cross-validation.

The `StratifiedKFold` function is a cross-validation method that is used for our dataset to shuffle and split the data into `k` folds while preserving the ratio of each class in the dataset. It is especially useful when working with imbalanced datasets as it ensures that each fold has a representative sample of the minority class. In this particular case, the `n_splits` parameter is set to `10`, which means that the dataset will be split into ten equal parts, and each part will be used once as a validation set while the other nine parts are used as the training set. This process will be repeated nine times, with each part being used once as the validation set, ensuring that every sample is used for validation exactly once.

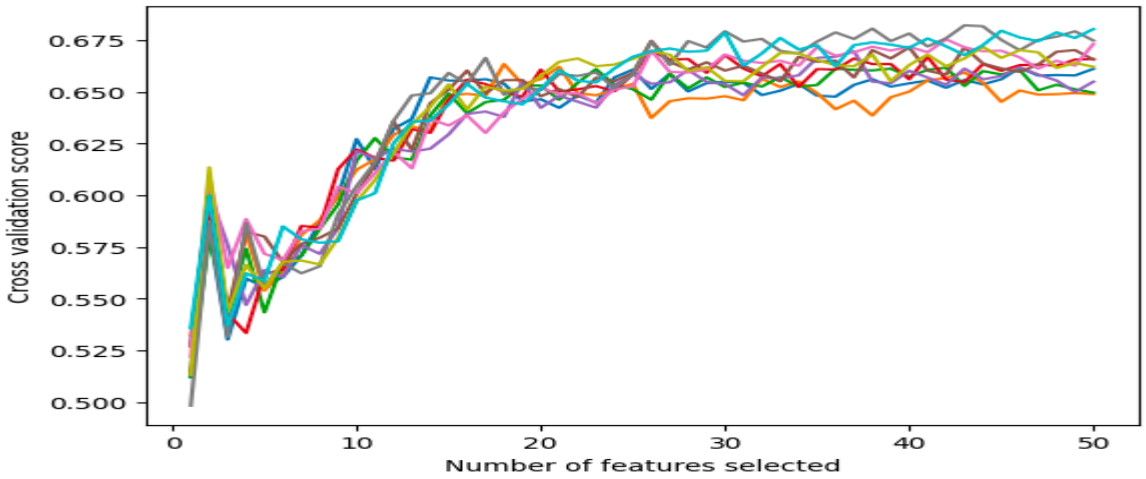
The outcomes of the RFECV are shown below:

Decision Tree:
The feature selection step of the pipeline selected 44 features from the dataset, which were then used to train a decision tree classifier. The selected features include various demographic and health-related variables such as BMI, physical and mental health, smoking and alcohol consumption, and incidence of stroke, among others. The decision tree was evaluated using a stratified 10-fold cross-validation with F1 score as the metric. The cross-validation score varied between 0.65 and 0.68 for different numbers of selected features, with the highest score achieved using all 44 features. This indicates that the selected features are relevant for predicting the target variable, and that the decision tree classifier performs moderately well on this dataset.

```
Number of features selected: 44

Selected features: ['BMI', 'PhysicalHealth', 'MentalHealth', 'SleepTime',
'Smoking_No', 'Smoking_Yes', 'AlcoholDrinking_No', 'AlcoholDrinking_Yes',
'Stroke_No', 'Stroke_Yes', 'DiffWalking_No', 'DiffWalking_Yes', 'Sex_Female',
'Sex_Male', 'AgeCategory_35-39', 'AgeCategory_45-49', 'AgeCategory_50-54',
'AgeCategory_55-59', 'AgeCategory_60-64', 'AgeCategory_65-69', 'AgeCategory_70-
74', 'AgeCategory_75-79', 'AgeCategory_80 or older', 'Race_American
```

```
Indian/Alaskan Native', 'Race_Asian', 'Race_Black', 'Race_Hispanic',
'Race_Other'Race_White', 'Diabetic_No', 'Diabetic_No, borderline diabetes',
'Diabetic_Yes', 'PhysicalActivity_No', 'PhysicalActivity_Yes',
'GenHealth_Excellent', 'GenHealth_Fair', 'GenHealth_Good', 'GenHealth_Very
good', 'Asthma_No', 'Asthma_Yes', 'KidneyDisease_No', 'KidneyDisease_Yes',
'SkinCancer_No', 'SkinCancer_Yes']

Cross-validation score:
[0.6536 0.65447154 0.66004141 0.66343434 0.65600656 0.66370251 0.66961771
0.68171021 0.67177419 0.67252396]
```

From the cross-validation graph, we could see how cross validation performed in each fold. It reached a peak when the number of features achieved was around 3 and experienced a drop. After that, the 'f1' score gradually climbed up as more features were selected.



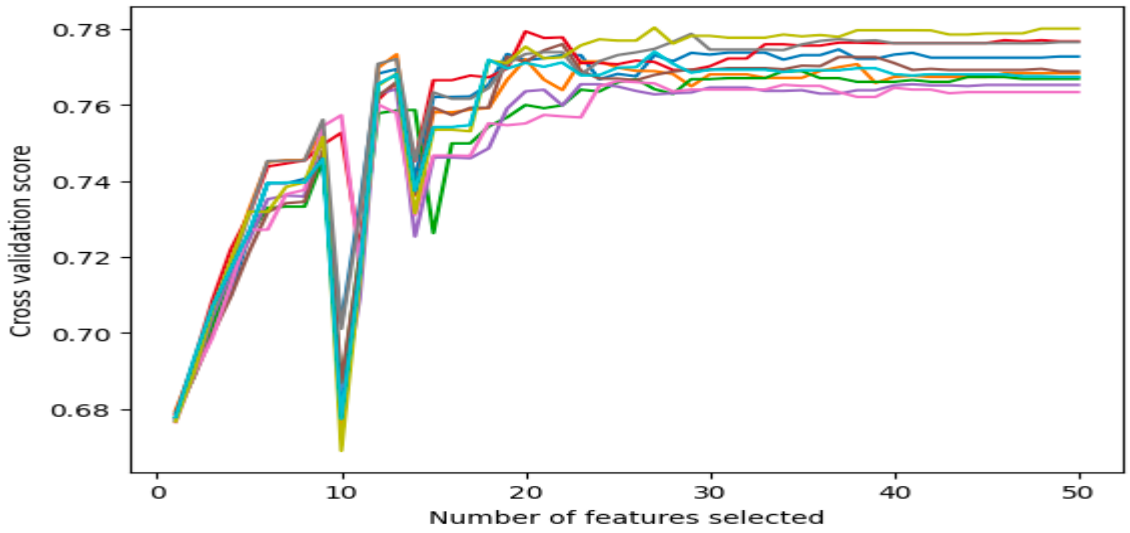
Logistic Regression:
The logistic regression model with 37 features selected by RFECV achieved a cross-validation score of around 0.77, which is a significant improvement compared to the Decision Tree.

Overall, the logistic regression model with the selected features by RFECV is a comparable better model with a higher f1 score.

```
Number of features selected: 37

Selected features: ['BMI', 'PhysicalHealth', 'MentalHealth', 'Smoking_No',
'Smoking_Yes', 'AlcoholDrinking_No', 'Stroke_No', 'Stroke_Yes',
'DiffWalking_No', 'DiffWalking_Yes', 'Sex_Female', 'Sex_Male', 'AgeCategory_18-
24', 'AgeCategory_25-29', 'AgeCategory_30-34', 'AgeCategory_35-39',
'AgeCategory_40-44', 'AgeCategory_45-49', 'AgeCategory_55-59', 'AgeCategory_60-
64', 'AgeCategory_65-69', 'AgeCategory_70-74', 'AgeCategory_75-79',
'AgeCategory_80 or older', 'Race_American Indian/Alaskan Native', 'Race_Asian',
'Race_Black', 'Diabetic_Yes', 'GenHealth_Excellent', 'GenHealth_Fair',
'GenHealth_Poor', 'GenHealth_Very good', 'Asthma_No', 'Asthma_Yes',
'KidneyDisease_No', 'KidneyDisease_Yes', 'SkinCancer_Yes']
```

```
Cross-validation score:
[0.77464789 0.76996424 0.76706984 0.77642436 0.76299213 0.77262181 0.7633946
0.77734375 0.77795152 0.76917027]
```

Given the cross validation graph of Logistic Regression, the ‘f1’ score experienced a dramatic drop in the cross-validation score at some subset when the features achieved 10, this can occur due to the particular combination of samples resulting in a less representative training set for the model, leading to a drop in performance. However, as more features are added, the overall performance of the model improves again, leading to the upward trend in the graph.



Random Forest:
The number of features selected by the random forest algorithm was 43, which is slightly more than the logistic regression model. The selected features included various demographic, lifestyle, and health-related factors such as BMI, smoking status, physical activity, and chronic conditions like asthma and diabetes.

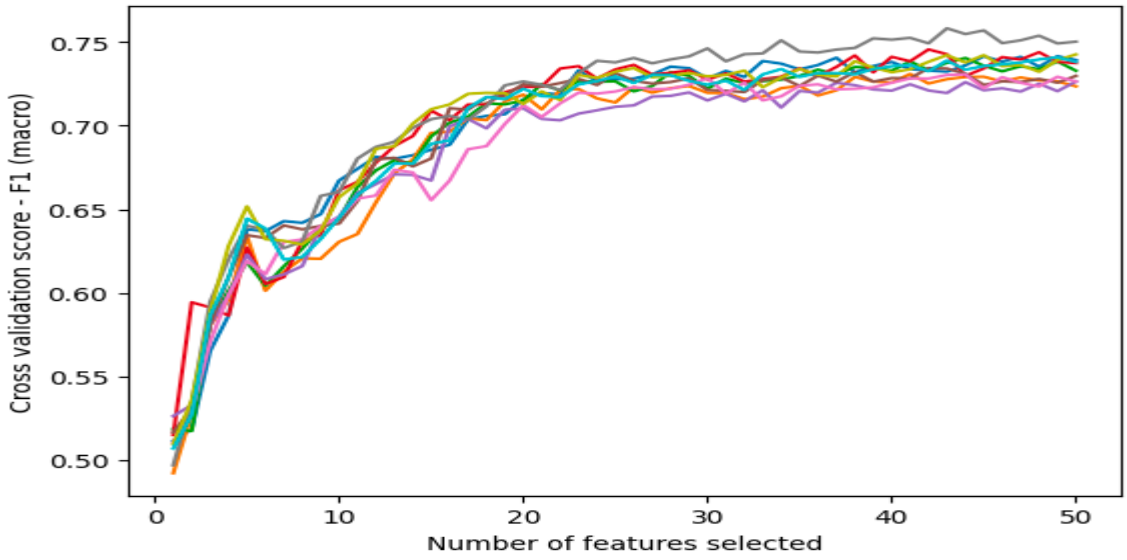
The cross-validation scores obtained from the random forest model ranged between 0.719 and 0.758, which are lower than the logistic regression model. However, the scores are still above 0.7, indicating that the model is performing moderately well in predicting the presence of heart disease. It is also worth noting that the scores are relatively consistent across the 10 folds of cross-validation, which suggests that the model is not overfitting the data.

Overall, the random forest model provides an alternative approach to predicting heart disease risk using a larger number of features than the logistic regression model. While the performance of the model is slightly lower, it is still useful in identifying the most important factors that contribute to heart disease risk.

```
Number of features selected: 43
```

```
Selected features: ['BMI', 'PhysicalHealth', 'MentalHealth', 'SleepTime',
'Smoking_No', 'Smoking_Yes', 'AlcoholDrinking_Yes', 'Stroke_No', 'Stroke_Yes',
'DiffWalking_No', 'DiffWalking_Yes', 'Sex_Female', 'Sex_Male', 'AgeCategory_18-
24', 'AgeCategory_25-29', 'AgeCategory_30-34', 'AgeCategory_35-39',
'AgeCategory_40-44', 'AgeCategory_45-49', 'AgeCategory_50-54', 'AgeCategory_55-
59', 'AgeCategory_60-64', 'AgeCategory_65-69', 'AgeCategory_70-74',
'AgeCategory_75-79', 'AgeCategory_80 or older', 'Race_Black', 'Race_Hispanic',
'Race_White', 'Diabetic_No', 'Diabetic_Yes', 'PhysicalActivity_No',
'PhysicalActivity_Yes', 'GenHealth_Excellent', 'GenHealth_Fair',
'GenHealth_Good', 'GenHealth_Very good', 'Asthma_No', 'Asthma_Yes',
'KidneyDisease_No', 'KidneyDisease_Yes', 'SkinCancer_No', 'SkinCancer_Yes']

Cross-validation score:
[0.73234942 0.72785062 0.73768522 0.74276655 0.71956437 0.73420739 0.73035926
0.75826772 0.74224344 0.73894325]
```

Upon the cross validation given below, random forest did not experience a dramatic drop and the performance is better when more features are selected.



4.3. MLP

For Multi Layer Perceptron, we used a different pipeline since the grid search is not working for it. Initially, for feature selection for MLP, we tried chi-squared test to see the statistical significance for each feature. With the 40 percent of features to keep from chi-squared test, the selected features were Smoking', 'Stroke', 'DiffWalking', 'Sex', 'AgeCategory', 'Diabetic', 'PhysicalActivity', 'GenHealth', 'KidneyDisease', 'SkinCancer', 'BMI'.

```
from sklearn.feature_selection import SelectPercentile, chi2

FeatureSelection = SelectPercentile(score_func=chi2, percentile=40)
X_new = FeatureSelection.fit_transform(x_train, y_train)
print('X_new shape is', X_new.shape)
print('selected features are:', FeatureSelection.get_support())
print('number of selected features:', FeatureSelection.get_support().sum())
```

```
print('features names are', FeatureSelection.get_feature_names_out())

selectedf40 = ['Smoking', 'Stroke', 'DiffWalking', 'Sex', 'AgeCategory',
'Diabetic',
      'PhysicalActivity', 'GenHealth', 'KidneyDisease',
      'SkinCancer', 'BMI']
x_clean2 = x_clean[selectedf40]
x_encoded2 = pd.get_dummies(x_clean2)
```

However, it turned out that the selected feature with this step could not improve the result of the MLP model. Therefore, we decided to keep all features.

The MLP model we trained is built with four 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 is compiled with the Adam optimizer, binary cross-entropy loss function, and F1 score as the evaluation metric. Adam optimizer is a stochastic gradient descent optimizer that is commonly used in deep learning. It is an adaptive learning rate optimization algorithm that combines the advantages of two other popular optimization algorithms: AdaGrad and RMSProp.

Adam optimizer calculates adaptive learning rates for each parameter based on estimates of the first and second moments of the gradients. The first moment is the average of the gradients, and the second moment is the average of the squared gradients. The algorithm uses these moments to update the parameters, which leads to faster convergence and better generalization.

Binary cross-entropy loss function is a popular loss function used for binary classification problems. It is used to measure the difference between the predicted probabilities and the actual labels. The formula for binary cross-entropy loss is:

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

Where N is the number of samples, y is the actual label of the i-th sample(either 0 or 1) and p(y) is the predicted probability of the point being 1 for all N points.

The data is oversampled by the Synthetic Minority Over-sampling Technique (SMOTE) method to balance the dataset as we wanted to see the result from the train dataset with oversampling. Before oversampling, we split our test dataset into a test set and validation set with the test size of 0.5. After data splitting, the size of test set and validation set is below here:

```
x_test: (5636, 50)
x_val: (5637, 50)
y_test:(5636, 1)
y_val:(5637, 1)
```

The reason we used the SMOTE method is to see the difference between oversampling and undersampling. Since there is no significant difference in result from the f1-score, which is our evaluation, we decided to see how MLP is doing with oversampled data.

```
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]
```

Therefore after oversampling, we have 167,914 counts for heart disease ‘Yes’ from 12,462. The model is then trained for 25 epochs with early stopping and learning rate reduction on plateau callbacks. The EarlyStopping callback monitors the validation loss and stops the training process when the loss does not improve for a certain number of epochs, while the ReduceLROnPlateau callback reduces the learning rate when the validation loss does not improve for a certain number of epochs. These callbacks help prevent overfitting and improve generalization.

4.4. Evaluation

To evaluate the performance of the machine learning models, several metrics have been used in the project. The metrics used in the code include cross-validation score, classification report, ROC AUC score, confusion matrix, precision-recall curve, and ROC curve.

In an imbalanced dataset, there is a significant difference in the number of samples between the majority class and the minority class. This means that a model can achieve high accuracy by simply predicting the majority class for all samples, but it may perform poorly on the minority class, which is the class of interest. Therefore, accuracy is not a suitable metric for evaluating model performance in an imbalanced dataset.

On the other hand, precision, recall, and F1-score are metrics that take into account the number of true positives, false positives, and false negatives, and are more appropriate for evaluating model performance in an imbalanced dataset. Precision measures the proportion of true positive predictions among all positive predictions, while recall measures the proportion of true positive predictions among all actual positive samples. F1-score is a harmonic mean of precision and recall, and gives equal weight to both metrics. In an imbalanced dataset, these metrics are particularly important because they help to evaluate how well the model is performing on the minority class.

$$\mathrm{F1}_{macro} = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}} = \frac{\mathrm{TP}}{\mathrm{TP} + \frac{1}{2}(\mathrm{FP} + \mathrm{FN})}$$

For F1-score to evaluate MLP models, we decided to compute recall and precision manually since the built in metrics are removed from Keras because of the probability for misleading from a batch-wise computation. However, since the test dataset is a single batch, we will compute F1-score manually to compare the performance for MLP with previous models.

```
def f1(y_true, y_pred):
    def recall(y_true, y_pred):
        """Recall metric.

        Only computes a batch-wise average of recall.

        Computes the recall, a metric for multi-label classification of
        how many relevant items are selected.
        """
        true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
        possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
        recall = true_positives / (possible_positives + K.epsilon())
        return recall

    def precision(y_true, y_pred):
        """Precision metric.

        Only computes a batch-wise average of precision.

        Computes the precision, a metric for multi-label classification of
        how many selected items are relevant.
        """
        true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
        predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
        precision = true_positives / (predicted_positives + K.epsilon())
        return precision
    precision = precision(y_true, y_pred)
    recall = recall(y_true, y_pred)
    return 2*((precision*recall)/(precision+recall+K.epsilon()))
```

The ROC AUC score measures the ability of the model to distinguish between the two classes by calculating the area under the receiver operating characteristic (ROC) curve. The ROC curve plots the true positive rate (sensitivity) against the false positive rate (1-specificity) for different threshold values. The AUC score ranges from 0.5 to 1, with 1 indicating perfect classification performance.In an imbalanced dataset, the ROC AUC and ROC curve are useful metrics to evaluate model performance because they are insensitive to class imbalance.

However, the ROC AUC can be misleading in highly imbalanced datasets because it is a global metric that does not take into account the imbalance of the classes. In such datasets, the positive class is often rare, and a classifier that predicts only negative samples can still achieve a high ROC AUC. This is because the ROC curve plots the True Positive Rate (TPR) against the False Positive Rate (FPR), which can be very low in the case of imbalanced datasets. As a result, a classifier that predicts all samples as negative can still achieve a high TPR, as there are only a few positive samples to predict. In this case, we use it to compare how it varied between the performance on the training dataset and test dataset.

Resulting from the misleading evaluation from the ROC AUC performance, we imply the information from the precision-recall curve because it focuses on the positive class and is insensitive to changes in the negative class distribution. The curve plots the relationship between the precision and recall for various threshold values of the classifier output, which

allows us to evaluate how well the model is performing at different trade-offs between precision and recall. Precision is the proportion of true positives among the predicted positives, while recall is the proportion of true positives among the actual positives. Precision measures the classifier's ability to correctly identify positive samples, while recall measures the classifier's ability to identify all positive samples. In an imbalanced dataset, high precision may be more important than high recall, as misclassifying positive samples can have more significant consequences than misclassifying negative samples.

The balanced accuracy is the average of sensitivity and specificity, and it ranges from 0 to 1, with 1 indicating the best performance. By taking the average of sensitivity and specificity, it provides a more accurate measure of overall performance by considering both the positive and negative classes. It ensures that the model is performing well on both classes, and not just on the majority class. Therefore, we also choose the balanced accuracy metric as a good choice for evaluating the performance of models on imbalanced datasets.

5. Results

5.1. Logistic Regression

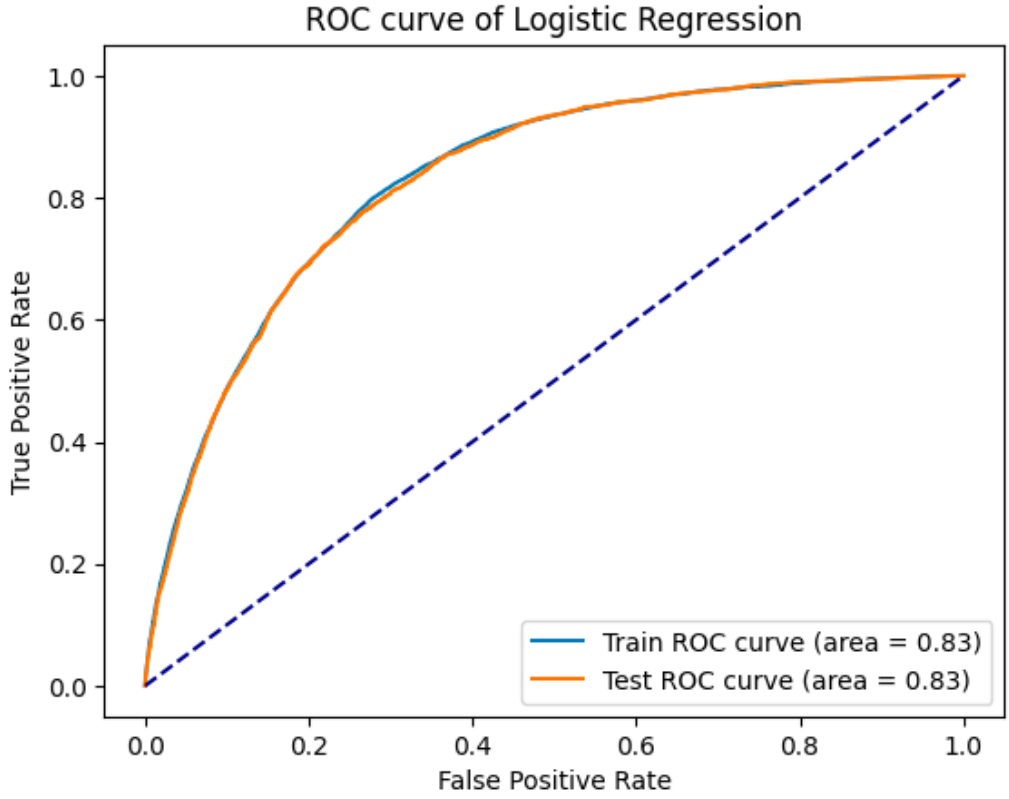
The result of logistic regression is shown below:

```
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.

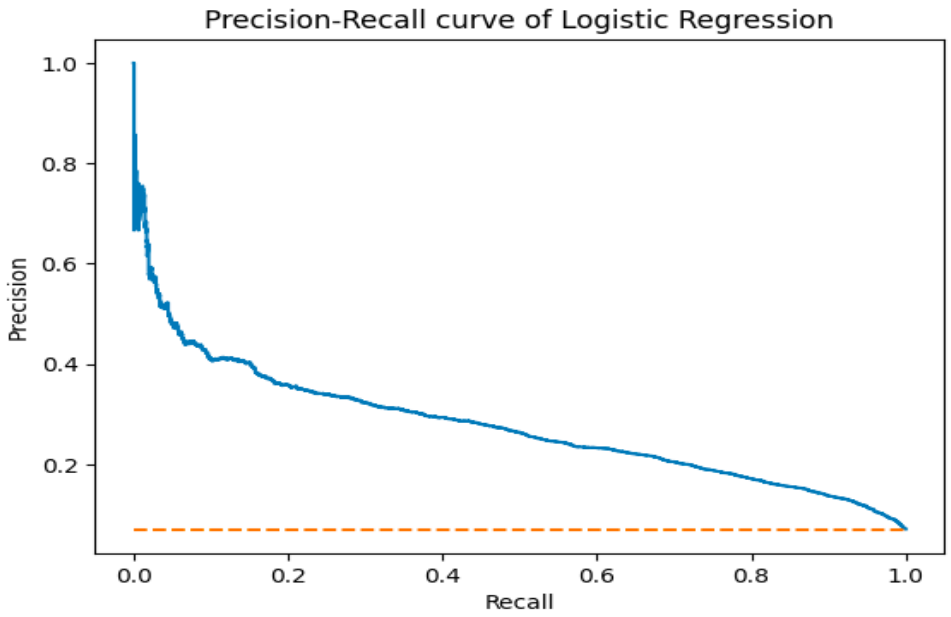
The ROC AUC score for the train dataset was 0.83, indicating good overall performance of the model. However, it is not the best indicator for imbalance data, we could see that the model has similar performance on both train and test dataset.



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.

5.2. Decision Tree

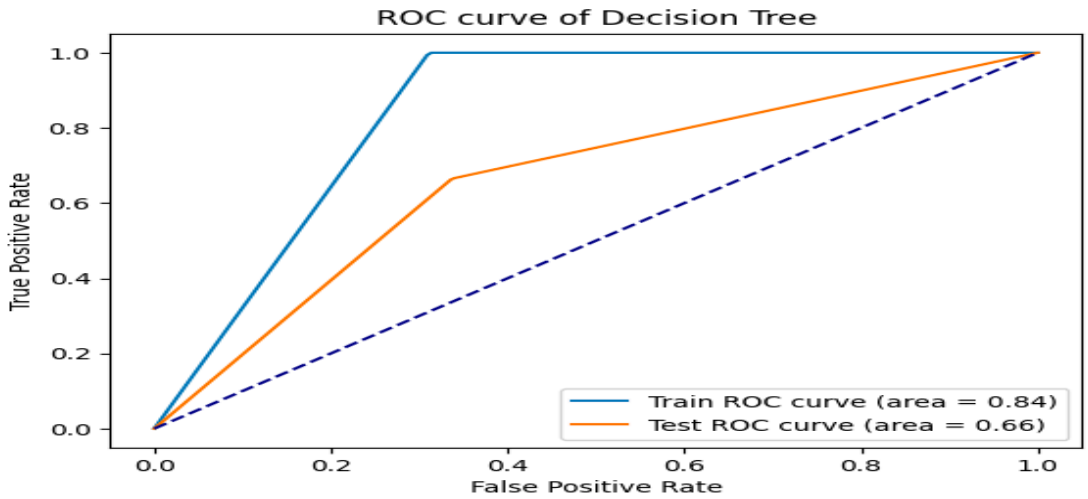
The Results from Decision Tree is shown below:

```
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.

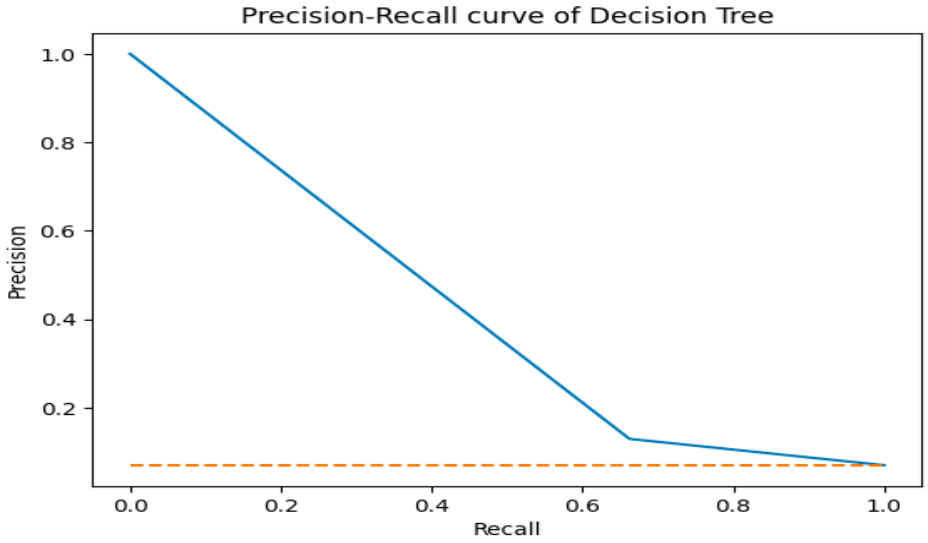
The ROC AUC score for the train dataset was 0.845, indicating good overall performance of the model. The confusion matrix for the train dataset showed that the model correctly classified most of the negative cases but misclassified a large number of positive cases as negative. From the graph, we can 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.

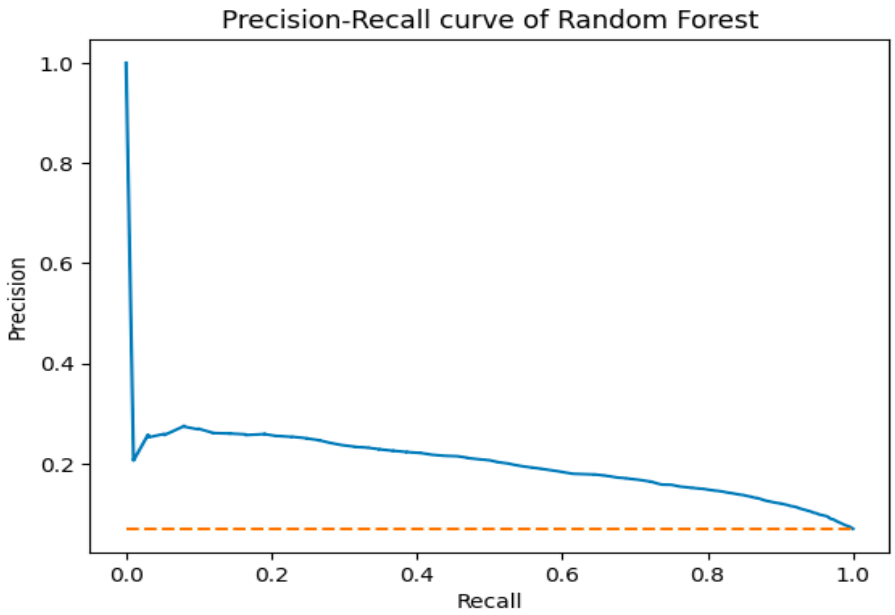
5.3. Random Forest

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.

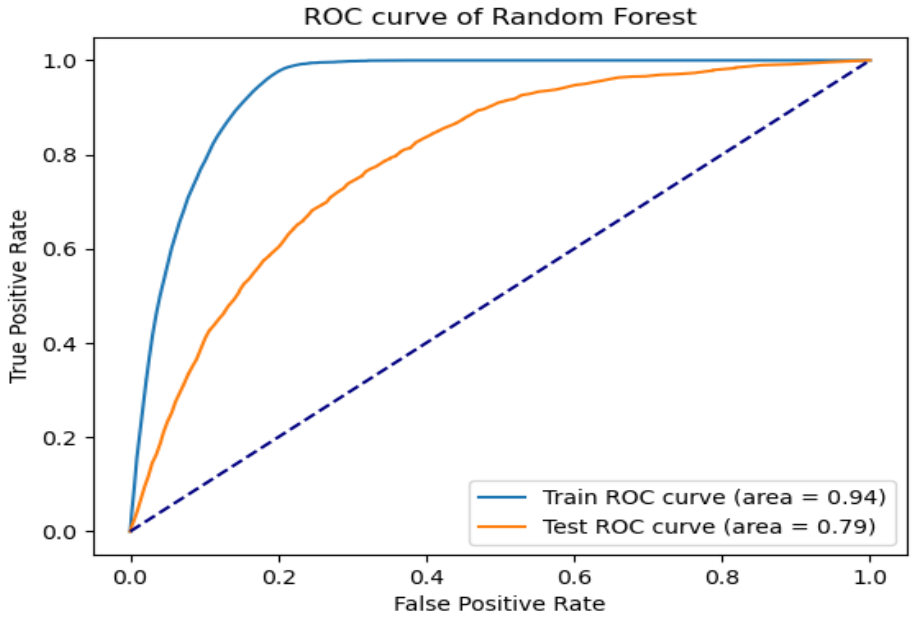
```
Train:
              precision    recall  f1-score   support

           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 29,480 true negatives (i.e., correctly predicted 'No Disease') and 2,340 true positives (i.e., correctly predicted 'Disease'). On the other hand, the model predicted 829 false negatives (i.e., predicted 'No Disease' when the actual class was 'Disease') and 12,446 false positives (i.e., predicted 'Disease' when the actual class was 'No Disease'). The balanced accuracy was reported as 0.72. 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 74%. 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.

5.4. MLP

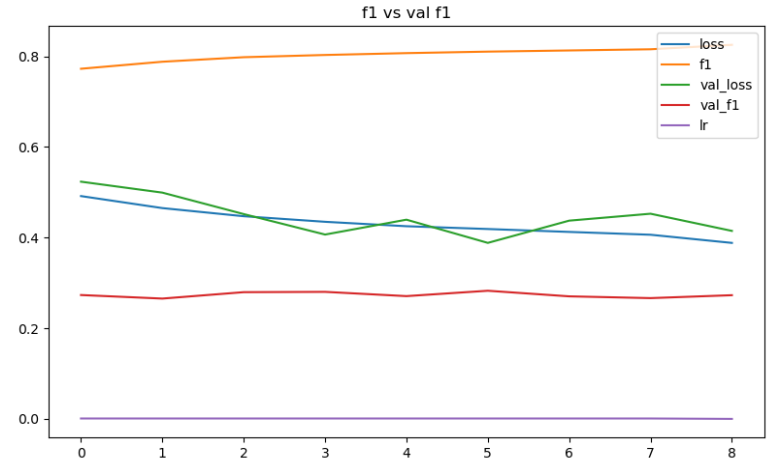
We tried a multi layer perceptron model using the SMOTE oversampling method. First, we tried MLP with the same train dataset which was undersampled and used for previous models, but the result was improved a little bit with the oversampled dataset. From the test set, we split it as a test and validation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

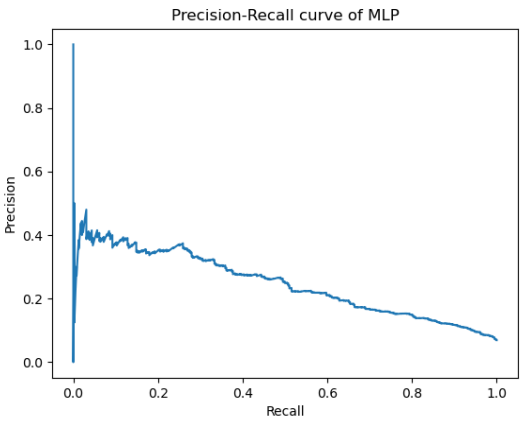
```
x_test: (5636, 50)
x_val: (5637, 50)
y_test:(5636, 1)
y_val:(5637, 1)
Before OverSampling, counts of label '1': [12462]
Before OverSampling, counts of label '0': [167914]

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]
```

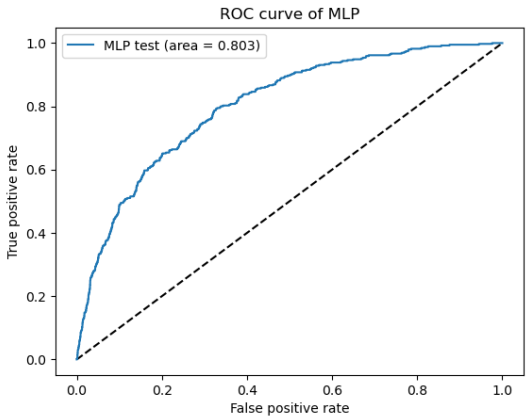
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_1 (Flatten)         (None, 50)                0

 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 since the test set was a single batch.

```
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.

6. 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.

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: https://machinelearningmastery.com/precision-recall-and-f-measure-for-imbalanced-classification/

8. Appendix

- All the libraries used in the project:

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

```
import seaborn as sns
from sklearn.model_selection import train_test_split
from dython.nominal import associations
from dython.nominal import identify_nominal_columns
from imblearn.pipeline import Pipeline
from sklearn.model_selection import StratifiedKFold
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler
from sklearn.feature_selection import RFECV
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
import xgboost as xgb
from sklearn.ensemble import AdaBoostClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report, confusion_matrix,
roc_auc_score,roc_curve,f1_score,precision_recall_curve
from imblearn.over_sampling import SMOTE
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from keras import backend as K
```