Heart Failure Prediction Using Machine Learning

Preprint · February 2025

DOI: 10.13140/RG.2.2.31411.13606

1 author:

Dineth Jayakody
University of Ruhuna
6 PUBLICATIONS **0** CITATIONS

SEE PROFILE

DEPARTMENT OF ELECTRICAL AND INFORMATION ENGINEERING

FACULTY OF ENGINEERING

UNIVERSITY OF RUHUNA

A PROJECT REPORT SUBMITTED IN PARTIAL FULFILMENT OF THE DEGREE OF THE BACHELOR OF THE SCIENCE OF ENGINEERING HONOURS

19th APRIL 2024

Heart Failure Prediction Using Machine Learning

Contents

List of Figures

List of Tables

Acronyms

ASY	-	Asymptomatic
ATA	-	Atypical Angina
BS	-	Blood Sugar
CVDs	-	Cardiovascular Diseases
ECG	-	Electrocardiography
HD	-	Heart Diseases
HDL	-	High-Density Lipoprotein
HR	-	Heart Rate
LDL	-	Low-Density Lipoprotein
LVH	-	Left Ventricular Hypertrophy
ML	-	Machine Learning
NAP	-	Non-Anginal Pain
ROC	-	Receiver Operating Characteristic
AUC	-	Area Under the Curve
SMOTE	-	Synthetic Minority Over-sampling Technique
SVM	-	Support Vector Machine
SVC	-	Support Vector Classification
TA	-	Typical Angina
TG	-	Triglycerides

Chapter 1

Introduction

Heart diseases have been among the major causes of death all over the world. It's really important to find heart failure in early stages. If we can identify it in early stages, we can take action to prevent it from causing serious problems or even death. So, spotting it early and acting quickly can save lives. Startups are using data and machine learning to predict who might be at risk of heart failure. And the way machine learning and data analytics have emerged, the domain of healthcare especially in predictive diagnostics has not lagged behind. With large amounts of data coming in from the medical sector, ranging from clinical records up to data gathered from real-time monitoring, new approaches to disease prediction and prevention are possibly emerging from it.

This is especially applicable to the case of heart failure: a complex clinical syndrome that results from the heart not being able to provide adequate output to maintain blood flow to meet the body's requirements [2]. The problem of predicting heart failure is multifactorial, due to the influence of many predisposing factors, which can include but are not limited to genetics, lifestyle choice, environmental factor, and existing conditions like diabetes or hypertension [3]. A problem of such nature is solved by machine learning models through their capability to scan through big sets of data to spot patterns or correlations that cannot be noticed by human analysts at first look. Early detection and prediction of heart failure can significantly affect patient outcomes.

Subsequently, closer follow-up will be done with high-risk patients in which there will be all the measures that can prevent the risk factor, with the possibility of receiving lifestyle recommendations and intervention treatments [4]. In that respect, it would be not only a potential lifesaver but also relieve the overall pressure on healthcare systems through prevention and being able to handle them more effectively in the earlier stage of the disease.

Generally, most machine learning-based heart failure prediction procedures include data collection, data pre-processing, feature selection, training of the model, and validation. Algorithms, not the same but relatively simple supervised learning techniques like regression, classification, and complicated ones like neural networks, deep learning, etc., are used primarily based on data features and the task of prediction at hand. This project report analyzes the heart failure prediction dataset [5], then describes the different methods taken to clean up the data set and then builds models using support vector classifier and logistic regression to fine tune them to predict whether a person suffers from a heart disease or not.

1.1 Project Aim and Description

The project aims to predict the presence or absence of heart disease using a dataset comprising 12 attributes, including age, gender, and various cardiovascular indicators. The features of the data set are named as below and their description are written with them.

- Age: age of the patient [years]
- Sex: sex of the patient [M: Male, F: Female]
- ChestPainType: chest pain type [TA: Typical Angina, ATA: Atypical Angina, NAP: Non-Anginal Pain, ASY: Asymptomatic]
- RestingBP: resting blood pressure [mm Hg]
- Cholesterol: serum cholesterol [mm/dl]
- FastingBS: fasting blood sugar [1: if FastingBS>120 mg/dl , 0: otherwise]
- RestingECG: resting electrocardiogram results [Normal: Normal, ST: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of>0.05 mV), LVH: showing probable or definite left ventricular hypertrophy by Estes' criteria]
- MaxHR: maximum heart rate achieved [Numeric value between 60 and 202]
- ExerciseAngina: exercise-induced angina [Y: Yes, N: No]
- Oldpeak: oldpeak which is equal to ST [Numeric value measured in depression]
- ST Slope: the slope of the peak exercise ST segment [Up: upsloping, Flat: flat, Down: downsloping]
- HeartDisease: output class [1: heart disease, 0: Normal]

The target variable indicates whether an individual has heart disease, with 1 denoting its presence and 0 indicating absence. Leveraging the dataset, two powerful machine learning algorithms, Logistic Regression and Support Vector Machine (SVM), will be employed for classification. Logistic Regression estimates probabilities for binary outcomes, while SVM identifies the optimal hyperplane to separate classes in the feature space. The project involves preprocessing the data, training the models, and evaluating their performance using metrics like accuracy, precision, recall, and F1-score. Ultimately, the goal is to analyze the data effectively and make accurate predictions regarding individuals' heart disease status.

1.2 Project Objectives

This section outlines the project objectives, setting the foundation for utilizing advanced machine learning techniques to predict heart disease. The objectives are as follows:

1. Predict the presence or absence of heart disease based on a dataset with 12 attributes (with the target variable).
2. Utilize the selected machine learning algorithms, Logistic Regression and Support Vector Machine (SVM), for classification and obtain a resonable accuracy for predictions (training and testing accuracies more than 80%).
3. Preprocess the dataset, handling missing values and scaling features as necessary.
4. Train the models using the preprocessed data.
5. Evaluate the performance of the models using metrics such as accuracy, precision, recall, and F1-score.

1.3 Project Scope

The project scope can be briefly listed as below.

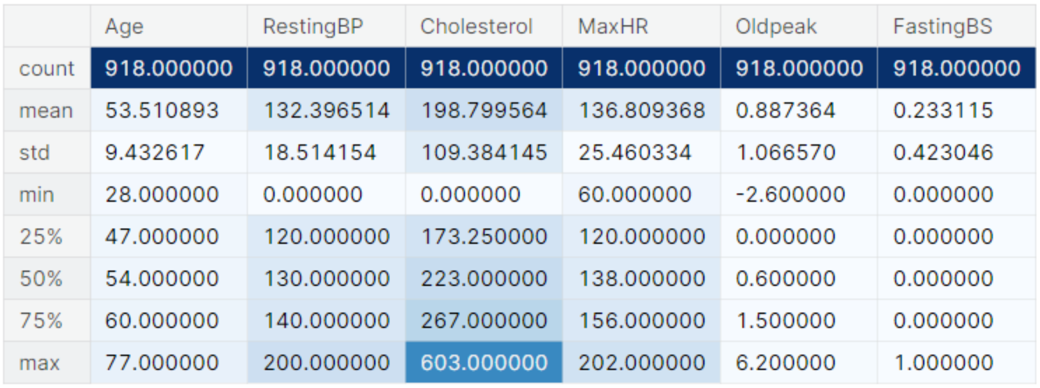
1. Analyze a single dataset comprising 12 attributes to predict heart disease.(no multiple data sets are used)
2. Only two machine learning algorithms, named Logistic Regression and Support Vector Machine (SVM) are used, for classification.
3. No deep learning methods such as neural networks are being used for the classification. The focus is on traditional machine learning algorithms.

Chapter 2

Methodology

2.1 Exploratory Data Analysis

The machine learning project relied on the heart failure prediction dataset [5]. The data set consists of 918 records, each containing 12 distinct features. Among them seven are numerical, including Age, RestingBP, Cholesterol, FastingBS, MaxHR, Oldpeak, and HeartDisease, while the remaining five are categorical: Sex, Chest-PainType, RestingECG, ExerciseAngina, and ST Slope. The statistical overview of numerical data provides insightsinto the distribution of data and provides measures such as the mean value and deviation. This understanding highlights the importance of feature scaling in data preprocessing. Figure 2.1 represents the distribution of numerical data.

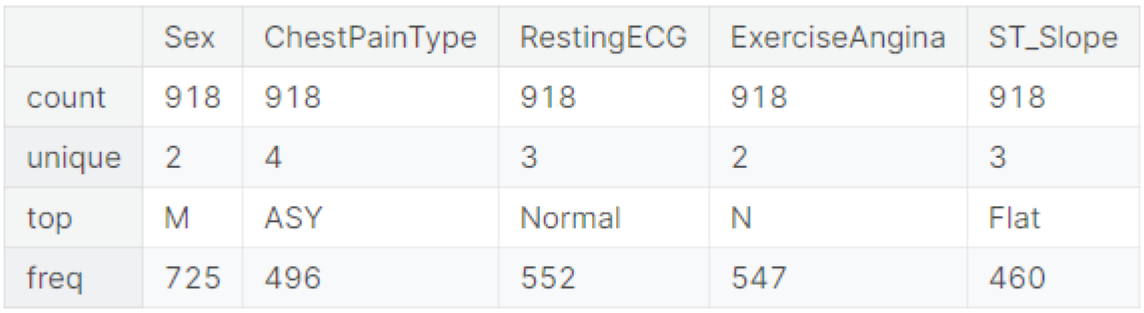


	Age	RestingBP	Cholesterol	MaxHR	Oldpeak	FastingBS
count	918.000000	918.000000	918.000000	918.000000	918.000000	918.000000
mean	53.510893	132.396514	198.799564	136.809368	0.887364	0.233115
std	9.432617	18.514154	109.384145	25.460334	1.066570	0.423046
min	28.000000	0.000000	0.000000	60.000000	-2.600000	0.000000
25%	47.000000	120.000000	173.250000	120.000000	0.000000	0.000000
50%	54.000000	130.000000	223.000000	138.000000	0.600000	0.000000
75%	60.000000	140.000000	267.000000	156.000000	1.500000	0.000000
max	77.000000	200.000000	603.000000	202.000000	6.200000	1.000000

Figure 2.1: Overview of numerical data distribution

By analysing the categorical features, it indicates the males are predominant in the dataset. Also asymptomatic pain is the most common chest pain type, while Normal resting electrocardiographic results prevail. Exercise-induced angina appears to be infrequent, with N being the dominant category. Flat is the most prevalent ST segment slope. Furthermore the overview of categorical data distribution can be seen in the Figure 2.2.

Data imbalance checking is a crucial criteria since models trained on imbalanced data may have difficulty to accurately predict the minority class. Imbalanced datasets, can lead to biased models favoring the majority class. Following Figure 2.3 shows that the difference between the percentages indicating the presence and absence of heart



	Sex	ChestPainType	RestingECG	ExerciseAngina	ST_Slope
count	918	918	918	918	918
unique	2	4	3	2	3
top	M	ASY	Normal	N	Flat
freq	725	496	552	547	460

Figure 2.2: Overview of categorical Data

disease is relatively small, with 55.3% indicating "yes" and 44.7% indicating "no", the dataset exhibits a minor imbalance. Therefore it was decided to move forward in the beginning.

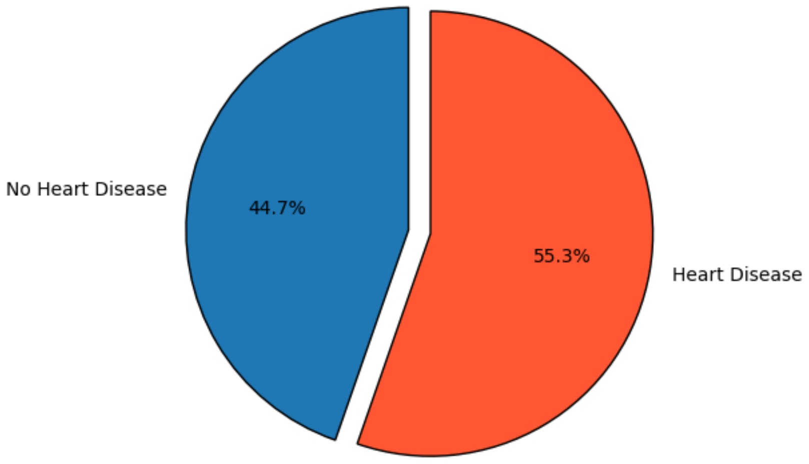


Figure 2.3: Pie chart representing data imbalance.

Duplicate data points and null values in the data set may negatively impact the training process of machine learning model. Using .duplicated() function we can detect duplicated data points. And also containing missing values degrade the performance of model and misinterpreted certain algorithms. isnull() function takes data as array-like object and indicate whether values are missing or not. In this dataset it does not contain repeated data or null values in either the training or test set. The checks were conducted independently to avoid data snooping and maintain generality of findings.

By checking the mean value of numerical features for cases of heart diseases and non-heart diseases is important in data analysis because it provides the central tendency of the data set. Also this provides baseline for the further statistical analysis. By observing the result in the Figure 2.4, it can identify potential risk factors for heart disease. .

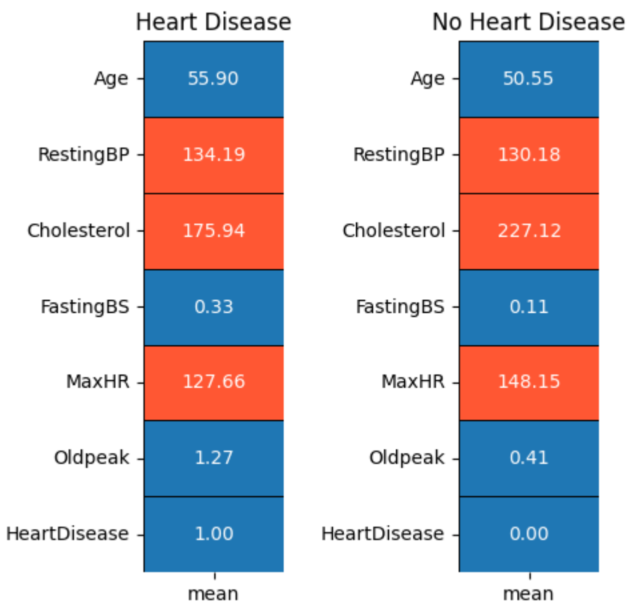


Figure 2.4: Mean values of all the features for cases of heart diseases and non-heart diseases

2.2 Data Visualization according to Features

In this section the features of the data set are analyzed separately to study the patterns and its' effects on whether the person is more likely to be a heart patient or not.

2.2.1 Visualization of patients with Heart Disease and No Heart Disease

Figure 2.5, shows the distribution of heart patients in the dataset. According to figure the number of heart patients is more than the number of healthy people.This shows that within the dataset there are 381 instances of individuals diagnosed with heart disease while only 307 individuals are classified as healthy. This gives an overall idea about the count of the heart patients.

2.2.2 Visualization of Patients with Respect to Gender

Visualization of heart disease with respect to gender shows that majority of male individuals are heart patients while majority of female individuals are healthy within the selected dataset. By observing the results in Figure 2.6, it can be determined that men have a major risk of suffering heart diseases than women .

2.2.3 Distribution of Data Related to Chest Pain Type

Distribution of Chest Pain Type is summarized in the below Figure 2.7. Most of the patients are suffering from Asymptomatic chest pain while Typical Angina is the least.

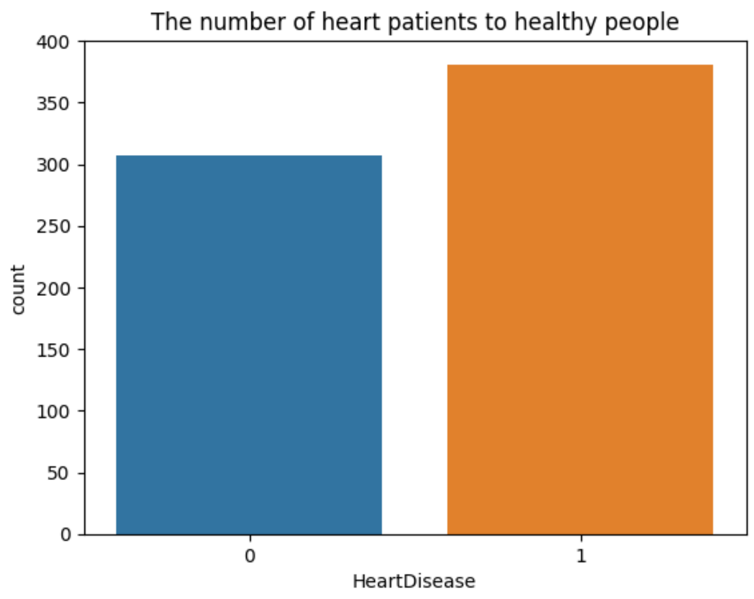


Figure 2.5: Visualization of patients with heart disease and no heart disease

2.2.4 Distribution of Data Related to Resting Blood Pressure for Heart Disease

High blood pressure leads to higher possibility of occurring heart diseases [6]. Figure 2.8 shows the histogram of the distribution of resting blood pressure. A slight increase in the heart disease cases when the resting blood pressure is increased can be observed. Also it can be observed that minimum value of blood pressure is 92 and the maximum blood pressure value is 200, so the blood pressure readings are reasonable.

2.2.5 Distribution of Data Related to Cholesterol Count

Elevated cholesterol levels are associated with increased risk of cardiovascular diseases (CVDs) such as stroke, heart attack, and heart failure. However, recent insights suggest that not all serum cholesterol is harmful. Serum cholesterol comprises three main types: HDL (high-density lipoprotein) or 'good' cholesterol, LDL (low-density lipoprotein) or 'bad' cholesterol, and triglycerides (TG) [7]. Among these, higher levels of HDL are preferable to higher levels of LDL and TG. The calculation for serum cholesterol (SC) involves summing the levels of HDL, LDL, and TG:

$$\mathrm{SC[mm/dl]} = \mathrm{HDL[mm/dl]} + \mathrm{LDL[mm/dl]} + \mathrm{TG[mm/dl]}$$

While the exact concentrations of these components may not be known in our dataset, a serum cholesterol level greater than 200 mm/dl is generally considered a risk factor for adverse health outcomes, including Heart Diseases. To illustrate this threshold, we can visualize it on a histogram as shown as in Figure 2.9 to show that the majority of Heart Disease cases in our dataset have serum cholesterol levels exceeding 200 mm/dl. This visualization can provide valuable insights into the distribution of cholesterol levels and their association with Heart Disease risk in our dataset. It can be observed that there are zero values and extremely high values (atypical cases) of serum cholesterol. This is incorrect, and we must treat them as anomalies. In a

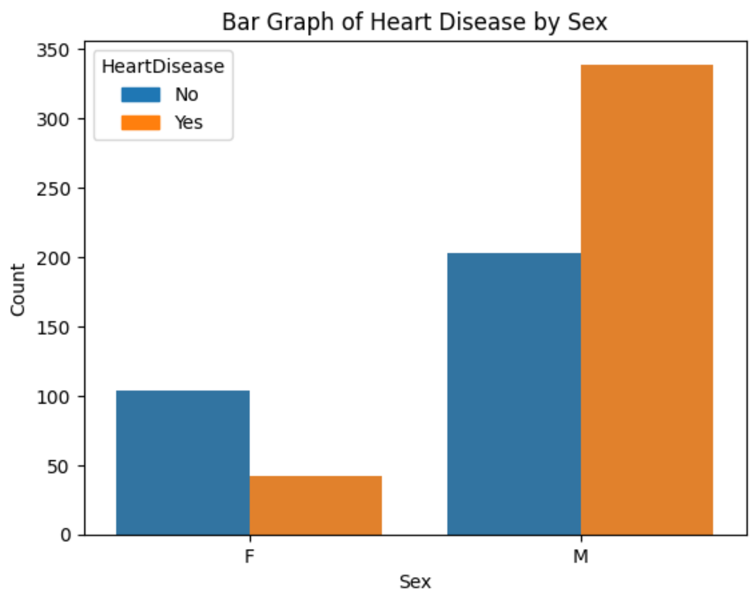


Figure 2.6: Visualization of patients with respect to gender

biological context, it's highly unlikely for a person to have exactly zero cholesterol in their body. Cholesterol is a vital molecule used in various functions such as cell membrane structure, hormone synthesis, and bile acid production. It's produced by the liver and can also be obtained from dietary sources [8]. Hence having zero cholesterol may mostly be due to incorrect readings.

2.2.6 Visualization of Data Related to Fasting Blood Sugar

High blood sugar can lead to damage of blood vessels and nerves that control the heart. Therefore individuals with high count of fasting blood sugar are at a higher risk of becoming a victim of Heart Diseases [9]. It can be observed in Figure 2.10, that a majority of patients with elevated blood sugar levels suffer from Heart Diseases.

2.2.7 Visualization of Data Related to Resting ECG

In cases of Heart Diseases, most patients have normal resting electrocardiograms (ECGs). However, when abnormalities like ST is present, the likelihood of having Heart Diseases doubles compared to those without these abnormalities. This can also be seen in the cases of LVH as well. This suggests that specific ECG findings can indicate a higher risk of Heart Diseases in patients. Figure 2.11 illustrates the distribution of each Resting ECG category corresponding to individuals at risk of developing heart conditions.

2.2.8 Visualization of the Effect of Maximum Heart Rate for Heart Disease

In cases where the measurement conditions are unspecified, it's challenging to determine the validity of the values obtained. In real life, the relationship between maximum heart rate and heart diseases can vary depending on individual health

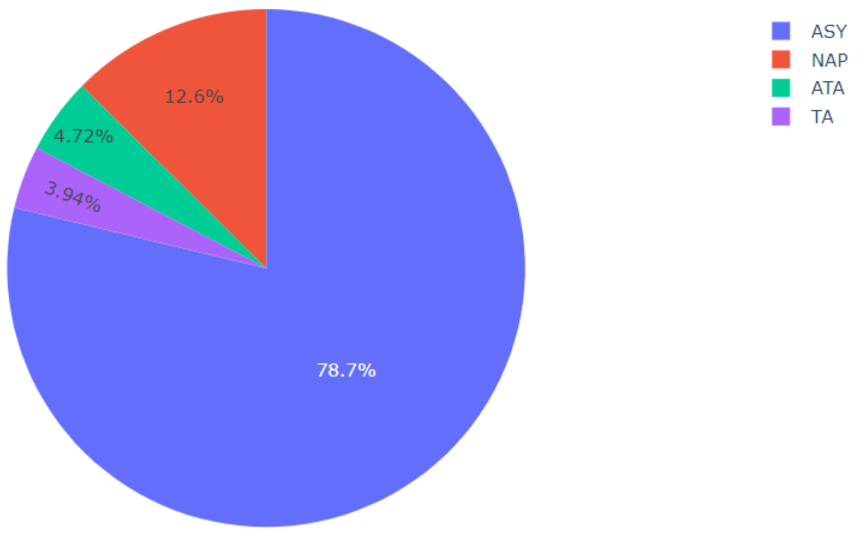


Figure 2.7: Distribution of chest pain type

conditions, lifestyle factors, and other variables [10]. While there may be observations suggesting that individuals with heart diseases tend to have lower maximum heart rates, it's essential to consider various factors that can influence heart rate and cardiovascular health.

Additionally, maximum heart rate alone may not be a definitive indicator of heart disease, as it's influenced by factors such as age, fitness level, medications, and underlying health conditions. Therefore, a comprehensive assessment by healthcare professionals, including medical history, diagnostic tests, and risk factors, is necessary for accurate evaluation and diagnosis of heart diseases[11]. When observing the histogram (Figure 2.12) it can be realized that there is a majority of heart patients are having high heart rate.

2.2.9 Visualization of the Effect of Exercise Angina for Heart Disease

Exercise-induced angina, also known as exertional angina, is chest pain or discomfort triggered by physical activity due to reduced blood flow to the heart muscle. It typically indicates underlying coronary artery disease and resolves with rest or medication. Figure 2.13 illustrates the relationship between Exercise-Induced Angina and heart disease status. It indicates a higher prevalence of individuals without exercise-induced angina in both heart disease-negative and heart disease-positive groups. However, there's a notable association between the presence of exercise-induced angina and heart disease, with a higher frequency observed in the heart disease-positive group. This suggests exercise-induced angina may serve as a clinical indicator of underlying heart issues, warranting further investigation.

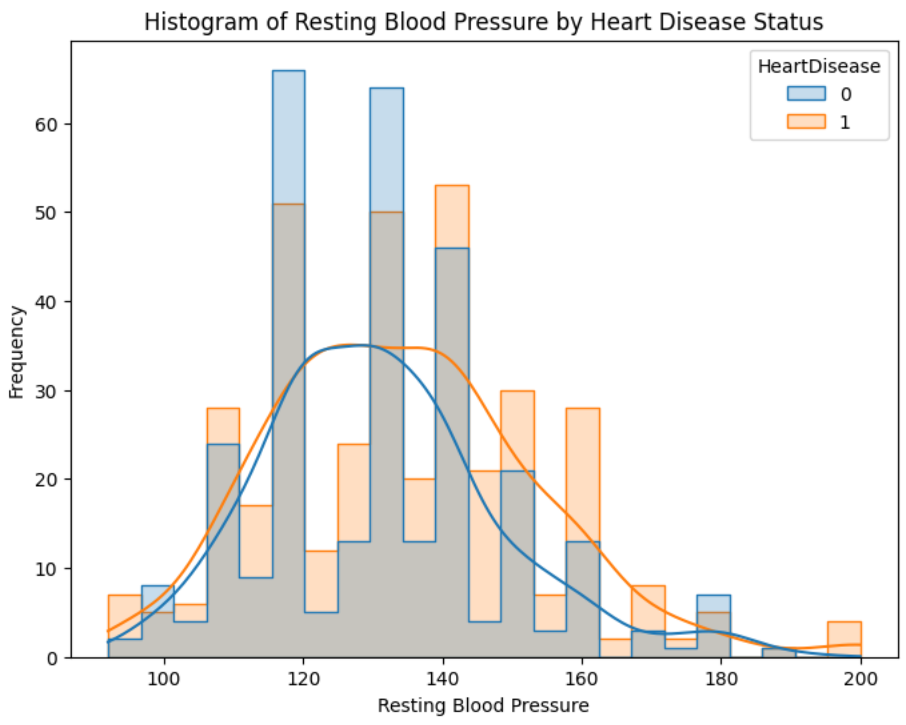


Figure 2.8: Effect of resting blood pressure for heart disease

2.2.10 Visualization of the Effect of Old Peak for Heart Disease

Old Peak also know as ST segment depression is a factor related to several heart diseases, but in some cases, this depression can be normal. It can be observed in Figure 2.14 that patients with heart diseases exhibit high values of depression; however, some patients with heart diseases also present zero values of depression.

2.2.11 Visualization of the Effect of ST Slope for Heart Disease

An up-sloping ST segment is typically considered as normal, while a flat or horizontally positioned ST segment, as well as a down sloping ST segment, may be seen as anomalies[12, 13]. These anomalies are commonly observed in the majority of cases involving patients with heart disease, as reflected in Figure 2.15. In summary an up sloping ST segment is generally normal, while flat or horizontal, and down sloping ST segments are considered anomalies. These anomalies are frequently seen in most cases of patients with heart disease, as evidenced by the histogram data.

2.3 Treating Anomalies

In the Exploratory Data Analysis all the features were analyzed and looked on to the possibilities and to the factors that affect on probable heart diseases.

Almost all features had reasonable data expect for the cholesterol data where there were several readings of 0 mm/dL which is not possible. Also there were readings above 400 mm/dL which is extremely rare and almost impossible to have in majority

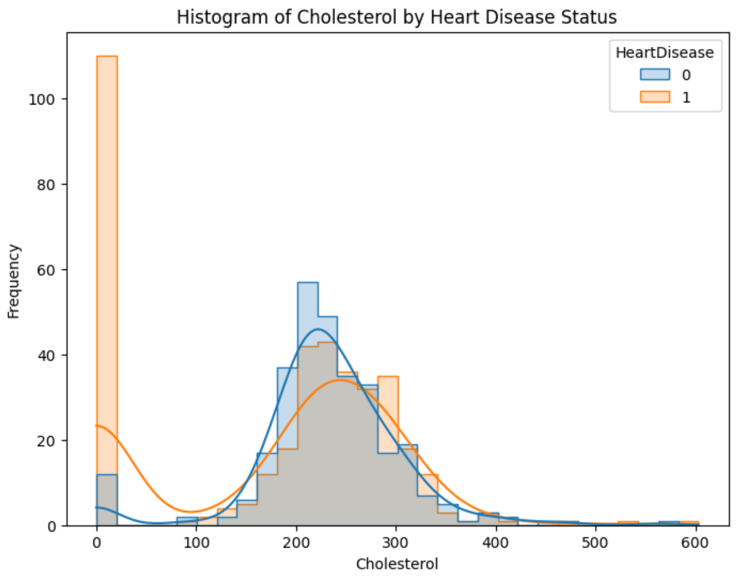


Figure 2.9: Visualization of effect of cholesterol for heart disease

cases. High cholesterol levels are considered a significant risk factor for heart disease and stroke. The Cleveland Clinic [14] states that total cholesterol levels of 200 mm/dL or higher are generally considered high, with levels between 200-239 mm/dL categorized as borderline high and 240 mm/dL and above categorized as high. Very high levels of cholesterol, such as those exceeding 300 mm/dL, are less common and may indicate severe hypercholesterolemia, which could be due to genetic conditions or other health issues. So we need to treat these data points accurately before starting to build the model.

.

As in Figure 2.16, it can be observed that there exist values of 0 mm/dL and values exceeding 400 mm/dL. But we cannot simply trim those values since, as observed 17% of the training data is of cholesterol values of 0 mm/dL. Given that the zero values represent a substantial portion of the dataset (17.73%), simply removing them might lead to significant information loss. So we need to look on to another reasonable technique.We can either do,

1. Capping
2. Imputation
3. Discretization

From the provided options, choosing capping (or Winsorization) as a technique for treating outliers, specifically for this case seems reasonable . Capping involves setting upper and lower threshold values and assigning those values to all data points above or below the thresholds, respectively. This method is suitable for small datasets where outliers cannot be removed without significant loss of information.

Capping ensures that extreme values are brought within a specified range, helping to mitigate their impact on the analysis without eliminating them entirely. This approach is particularly useful when outliers are due to measurement errors or data entry errors, as it allows to maintain the integrity of the dataset while addressing

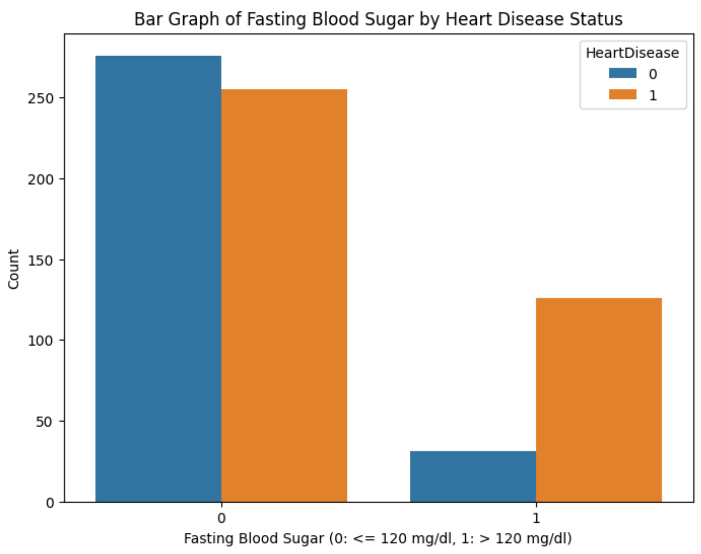


Figure 2.10: Bar graph of fasting blood sugar count by heart disease status

extreme values. Therefore, considering dataset's characteristics and the potential impact on model accuracy, capping would be a suitable technique for handling outliers. An upper threshold of 400 mm/dL and a lower threshold of 0 mm/dL was set and capping was done as in Figure 2.17.

.

The box plot after capping was obtained as in Figure 2.18. Here it can be observed that the extreme values are being treated properly and the data has been arranged in a accepted manner.

.

2.4 Performing Feature Encoding

The categorical features here, which are Sex, Chest Pain Type, Resting ECG, Exercise Angina, and ST Slope are not ordnial. So it is not possible to use label encoding here. Hence it was opted to go with binary encoding and one-hot encoding. Here instead of these we can use pd.getdummies() which automatically applies one-hot encoding to all categorical columns in the dataset, making the encoding process more efficient and concise. The data set after being encoded can be observed as in Figure 2.19.

.

Using getdummies() for encoding has several advatanges as mentioned below.

- Simplicity: Easy-to-use function without the need for additional libraries or objects.
- Automation: Automatically identifies and encodes categorical columns.
- Flexibility: Allows customization by specifying columns to encode.
- Drop First Category: Option to drop the first category to avoid multi-collinearity.
- Integration: Seamlessly integrates with other pandas functions and workflows.
- Performance: Optimized for efficiency, capable of handling large datasets.

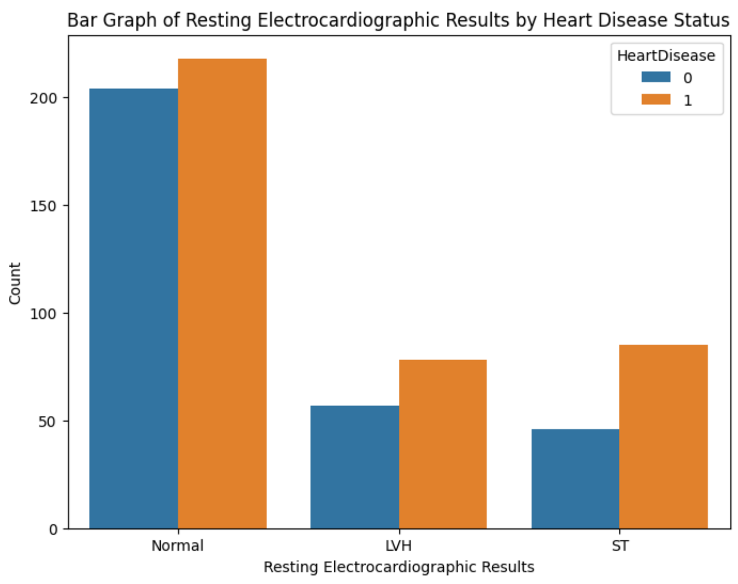


Figure 2.11: Visualization of effect of resting ECG for heart disease

2.5 Feature Selection

After encoding was done, it was focused on feature selection and feature scaling. Instead of giving priority for feature scaling it was opted to look for the options for feature selection due to several reasons as mentioned below.

- Preservation of Information: Standardizing the feature does standardize the feature scale but does not change the feature's relative importance. Feature selection before scaling enforces that the important features carry their importance into the derived model and evidently rank high for the most accurate modeling process.
- Efficiency: Feature selection, therefore, reduces the data dimensionality from the feature point of view by eliminating redundant or useless features. Feature selection before scaling would save worthless computation with regards to the discarded features at the end and hence might be more efficient.
- Interpretability: Feature selection will provide those which are most important for the prediction and, in that way, enhance model interpretability. All these steps bring a reasonable assurance that the selected features are really a representative subset of the original data, hence easy for their interpretation.

In general, to make sure that the selected features capture the dataset information and maintain its importance in the model-building process, feature selection should follow the sequence of feature scaling. Generally, if feature scaling is applied before the feature selection process, the selected features will fail to capture the right information from the dataset.

The obtained correlation matrix can be seen in Figure 2.20. Here it can be observed that some features less relevancy with the target variable. But the correlation matrix will only give information on whether there is a linear relationship between variables. It won't capture non-linear relationships or dependencies between features and the target variable.

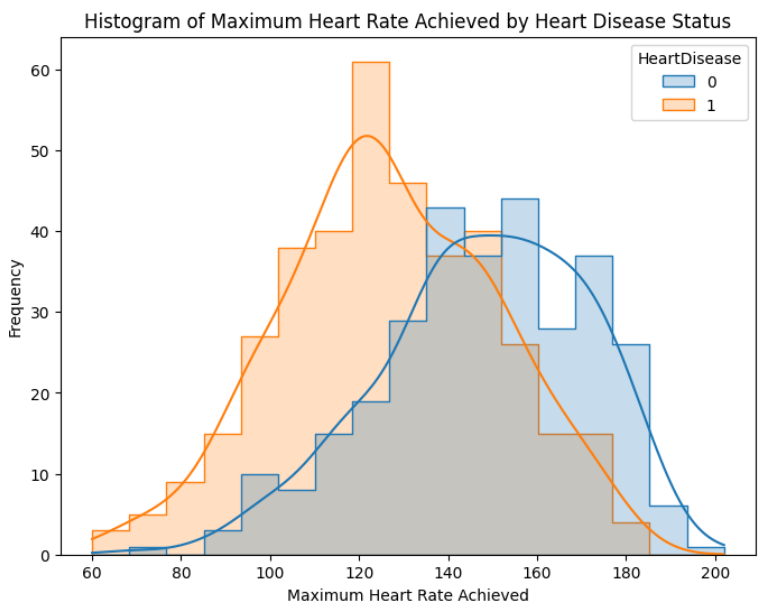


Figure 2.12: Visualization of effect of maximum heart rate for heart disease

.

Therefore, in order to interpret, even with due caution, data from medical research without an explicit background provided, the interpretation of the Chi-square tests concerning categorical variables and the interpretation of correlation matrices regarding continuous variables are most likely best avoided. Almost impossible to establish in the absence of the knowledge of a medical context. The interpretation can be further complicated by potential confounding factors, complex interactions, and, above all, concerns with respect to the quality of data. Quality issues of the data may also pose as a compromise in the obtained outcome results. When this is the case, domain experts should be taken for clinical relevance such that the right, accurate, and sensible conclusion could be ascertained while using these statistical methods in medical analysis. Hence here it was opted to move forward with the analysis without dropping down any feature columns.

2.6 Feature Scaling

As we can observe in our data set different features got different ranges of values. Therefore, feature scaling is important when dealing with features that have different ranges of values. Without scaling, features with larger magnitudes may dominate in the model that we create, leading to biased results. Different techniques can be used for feature scaling such as,

- Min-Max Scaling
- Normalization
- Standardization

For modeling with Logistic Regression and SVM (Support Vector Machine), standardization (Z-score normalization) is generally the most suitable feature scaling

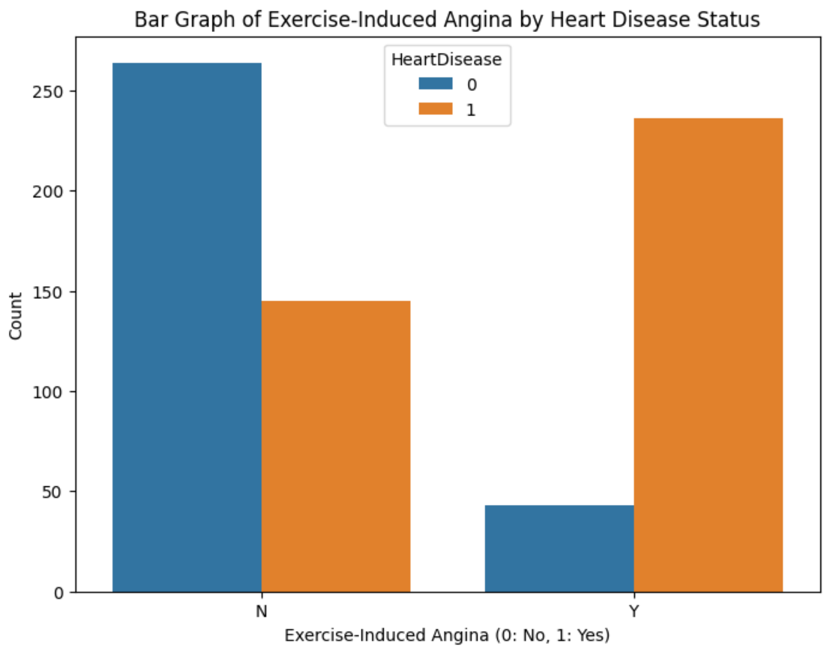


Figure 2.13: The bar graph of exercise angina by heart disease

method. Standardization was pefromed as in Figure 2.21. Both of these algorithms can benefit from standardization due to the mentioned reasons below.

- For Logistic Regression: Although it doesn't require feature scaling to converge, standardization can make the optimization process faster and more stable. It's particularly beneficial because logistic regression coefficients can be interpreted as the importance of each feature, and standardizing ensures these features are on the same scale.
- SVM: This algorithm is particularly sensitive to the scale of the input features because it relies on calculating the distance between data points. If one feature has a much larger scale than another, it will dominate the distance metric, potentially leading to inaccurate models. Standardization ensures that each feature contributes equally to the distance calculation.

.

The data set after scaling can be viewed as in Figure 2.22. Here it can be observed how features like Age Resting BP, Cholesterol, Fasting BS, Max HR, Old peak has been standardized. Also it is observed that features like age and Max HR which are clearly positive values are also being given negative values at some occasions. is perfectly acceptable and expected behavior. Standardization (Z-score normalization) transforms the data to have a mean of 0 and a standard deviation of 1. This means that values below the mean will be transformed into negative values, and values above the mean will be transformed into positive values. It does not imply anything negative about the age itself; it's merely a mathematical adjustment to help certain algorithms process the data more effectively.

The concept of negative values does not imply "negative age" or anything unrealistic in the context of the data. It's a relative measure indicating how much and in what direction the values deviate from the mean of the dataset. For machine learning

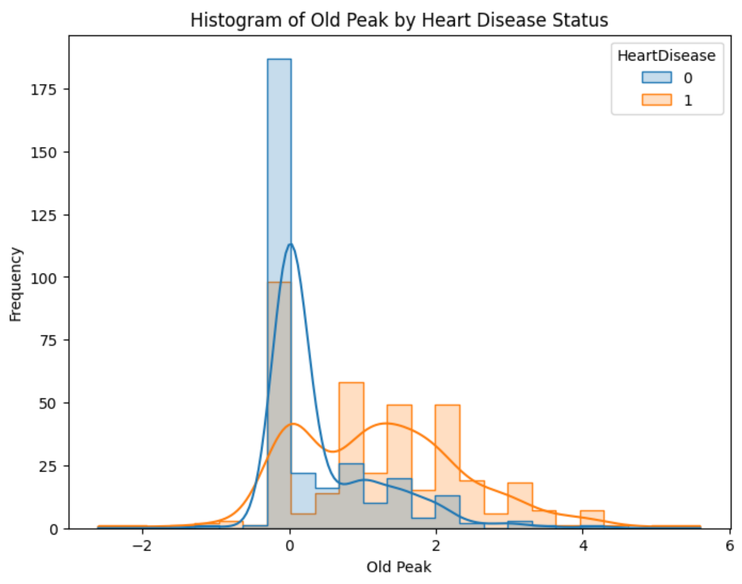


Figure 2.14: Histogram of old peak by heart disease status

algorithms, especially those like Logistic Regression and SVM, the absolute values of features are not as important as their distribution and how they relate to each other in the feature space. Standardization helps to put all features on the same scale, making the model training process more efficient and often leading to better performance.

.

2.7 Developing the Model

The dataset is quite balanced; however, since the purpose of this dataset is going to use a classification algorithm, this situation could introduce a model-bias case favoring the majority class. Class imbalance may impact the performance of the model, especially logistic regression and Support Vector Machine (SVM) algorithms. In most cases, the minority class (in this case, the instances labeled as 'HeartDisease' = 1) would be underrepresented.

In the case of logistic regression, before fitting the logistic regression model, Synthetic Minority Over-sampling Technique (SMOTE) can be applied to oversample the minority class (instances labeled as 'HeartDisease' = 1). SMOTE generates synthetic samples for the minority class by interpolating between existing minority class samples. The over-sampled dataset can then be used to train the logistic regression model, ensuring that both classes are represented more equally.

Similarly, before training the SVM model, SMOTE can be applied to balance the dataset. SMOTE helps create a more balanced dataset by generating synthetic samples for the minority class. Balancing the dataset with SMOTE can lead to a more robust decision boundary for the SVM classifier, improving its performance on imbalanced data. Figure 2.23 shows how SMOTE is being performed on the data set to balance the number of positive cases and the negative case to 381.

.

After the data pre-processing stage, the analysis moved on to building models using logistic regression and support vector machine (SVM) algorithms. These tech-

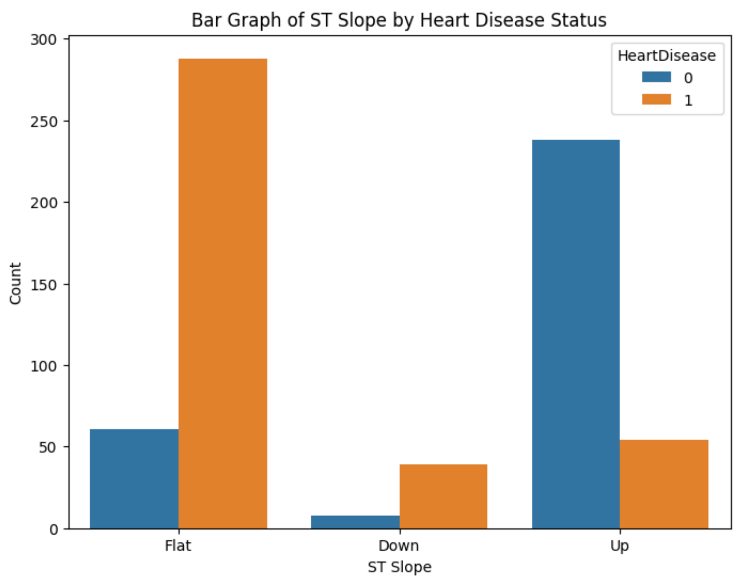


Figure 2.15: Bar graph of ST slope by heart disease status

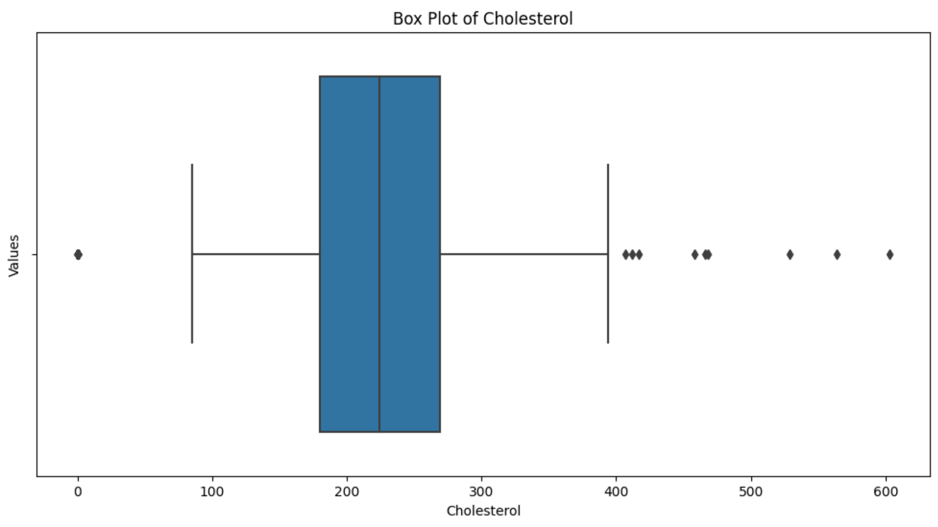
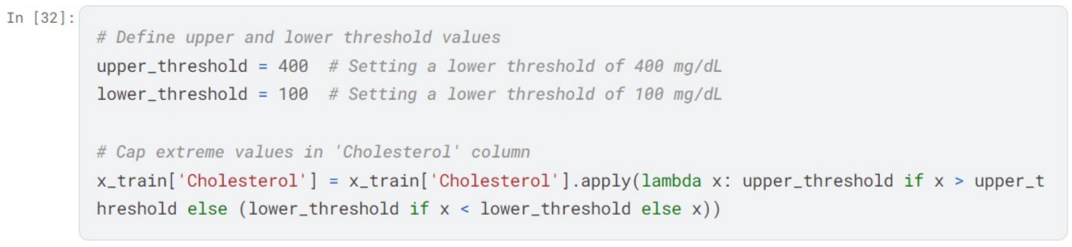


Figure 2.16: Box plot for cholesterol value distribution

niques are well-suited for classification tasks and have their own unique advantages depending on the data and application requirements.Logistic Regression is a basic statistical method that predicts probabilities using a logistic function, which is beneficial for binary classification tasks. It is easy to interpret, simple to use, and quick to train, making it a good starting point for modeling. Logistic regression evaluates odds and probabilities to make predictions. SVM is a powerful and flexible machine learning model. It works by finding the best hyper-plane to separate classes in the feature space. SVMs are great for high-dimensional spaces and can handle linear and nonlinear boundaries using kernel functions. The kernel trick lets SVMs work in a transformed feature space without needing to calculate data coordinates, saving on computation. This makes SVMs useful for complex classification problems with unclear decision boundaries initially.

Both models were fine-tuned and validated to ensure they were performing at their best. Logistic regression's performance was assessed by analyzing coefficients



```
In [32]:
# Define upper and lower threshold values
upper_threshold = 400  # Setting a lower threshold of 400 mg/dL
lower_threshold = 100  # Setting a lower threshold of 100 mg/dL

# Cap extreme values in 'Cholesterol' column
x_train['Cholesterol'] = x_train['Cholesterol'].apply(lambda x: upper_threshold if x > upper_t
hreshold else (lower_threshold if x < lower_threshold else x))
```

Figure 2.17: Code for performing capping

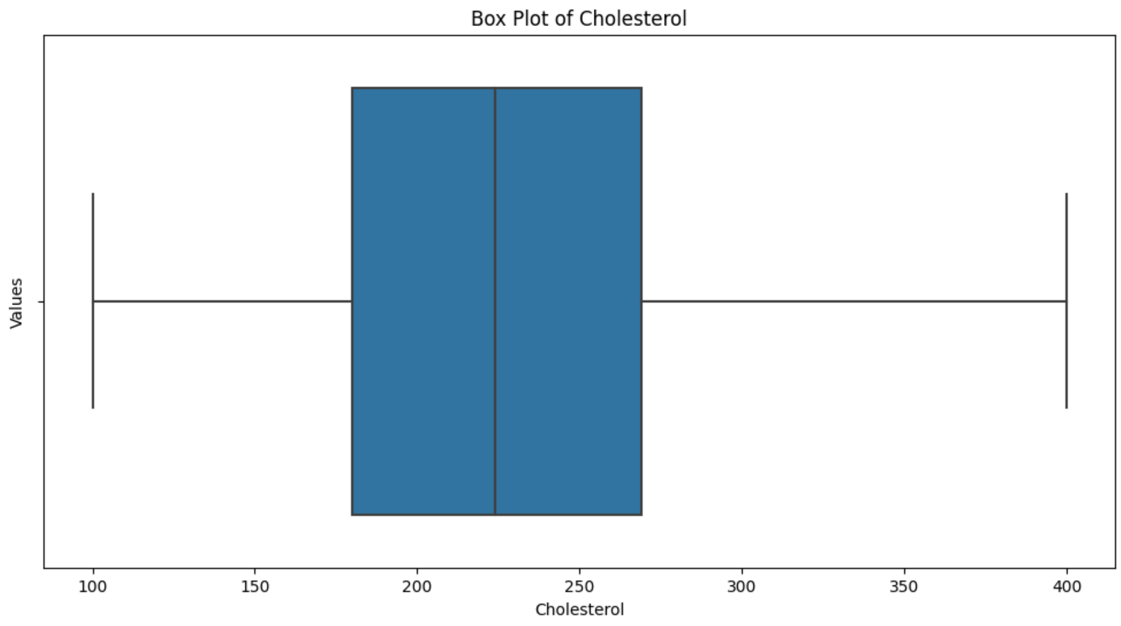
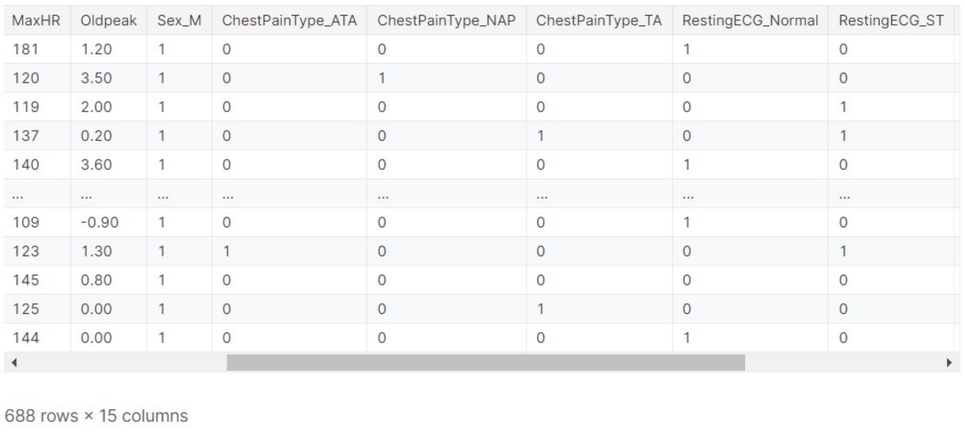


Figure 2.18: Box Plot for cholesterol value distribution after capping

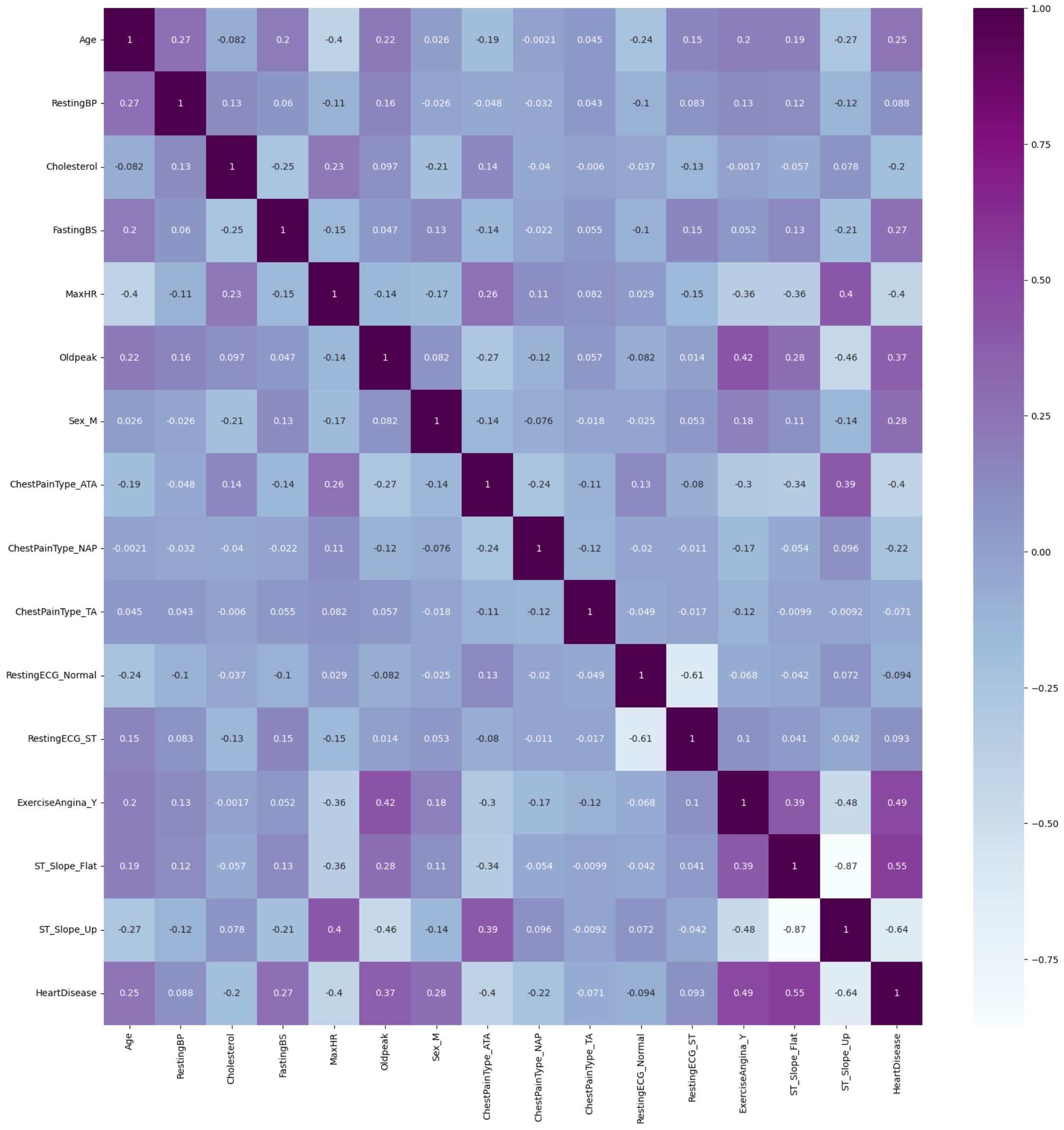
and using metrics such as the confusion matrix, Receiver Operating Characteristic (ROC) curves, and AUC scores, which offer detailed information about how accurately the model can classify data and the balance between sensitivity and specificity. SVM's effectiveness was measured by testing different kernel types (linear, polynomial, radial basis function, etc.) and adjusting parameters like C (regularization parameter) and gamma (kernel coefficient) to find the right balance between minimizing training error and testing error, thereby avoiding overfitting.



MaxHR	Oldpeak	Sex_M	ChestPainType_ATA	ChestPainType_NAP	ChestPainType_TA	RestingECG_Normal	RestingECG_ST
181	1.20	1	0	0	0	1	0
120	3.50	1	0	1	0	0	0
119	2.00	1	0	0	0	0	1
137	0.20	1	0	0	1	0	1
140	3.60	1	0	0	0	1	0
...	...	...	...	...	...	...	...
109	-0.90	1	0	0	0	1	0
123	1.30	1	1	0	0	0	1
145	0.80	1	0	0	0	0	0
125	0.00	1	0	0	1	0	0
144	0.00	1	0	0	0	1	0

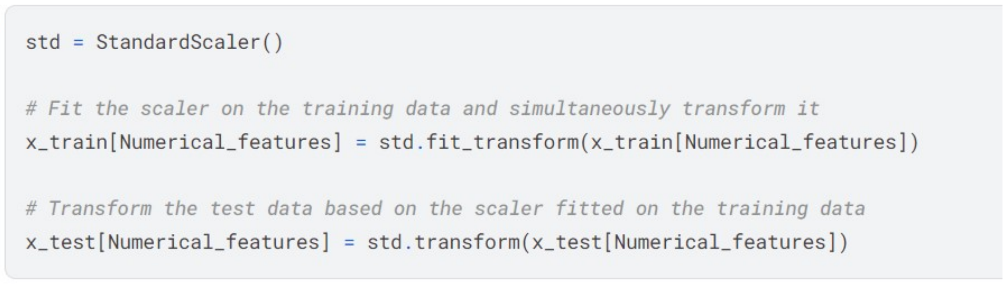
688 rows × 15 columns

Figure 2.19: Part of the data table after being encoded



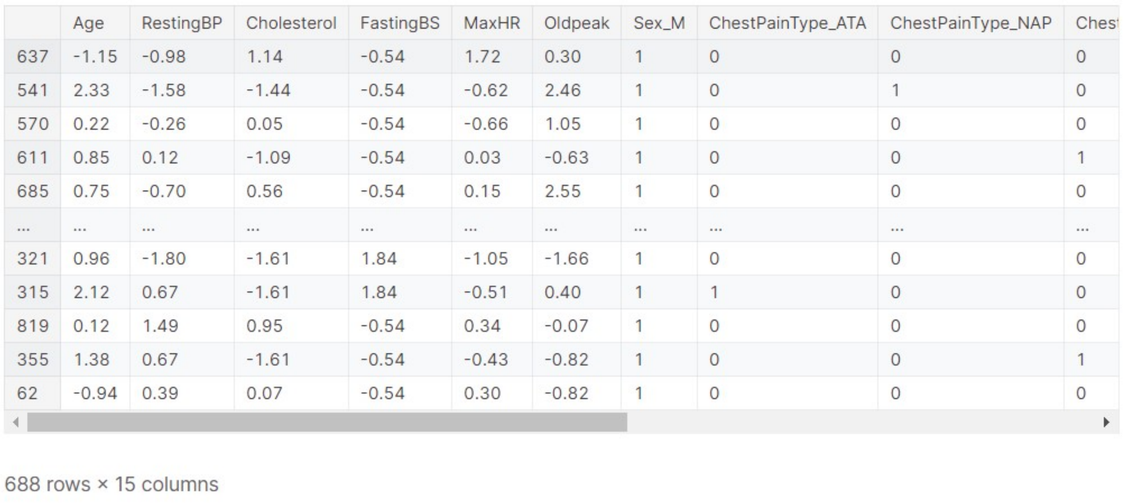
	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	Sex_M	ChestPainType_ATA	ChestPainType_NAP	ChestPainType_TA	RestingECG_Normal	RestingECG_ST	ExerciseAngina_Y	ST_Slope_Flat	ST_Slope_Up	HeartDisease
Age	1	0.27	-0.082	0.2	-0.4	0.22	0.026	-0.19	-0.0021	0.045	-0.24	0.15	0.2	0.19	-0.27	0.25
RestingBP	0.27	1	0.13	0.06	-0.11	0.16	-0.026	-0.048	-0.032	0.043	-0.1	0.083	0.13	0.12	-0.12	0.088
Cholesterol	-0.082	0.13	1	-0.25	0.23	0.097	-0.21	0.14	-0.04	-0.006	-0.037	-0.13	-0.0017	-0.057	0.078	-0.2
FastingBS	0.2	0.06	-0.25	1	-0.15	0.047	0.13	-0.14	-0.022	0.055	-0.1	0.15	0.052	0.13	-0.21	0.27
MaxHR	-0.4	-0.11	0.23	-0.15	1	-0.14	-0.17	0.26	0.11	0.082	0.029	-0.15	-0.36	-0.36	0.4	-0.4
Oldpeak	0.22	0.16	0.097	0.047	-0.14	1	0.082	-0.27	-0.12	0.057	-0.082	0.014	0.42	0.28	-0.46	0.37
Sex_M	0.026	-0.026	-0.21	0.13	-0.17	0.082	1	-0.14	-0.076	-0.018	-0.025	0.053	0.18	0.11	-0.14	0.28
ChestPainType_ATA	-0.19	-0.048	0.14	-0.14	0.26	-0.27	-0.14	1	-0.24	-0.11	0.13	-0.08	-0.3	-0.34	0.39	-0.4
ChestPainType_NAP	-0.0021	-0.032	-0.04	-0.022	0.11	-0.12	-0.076	-0.24	1	-0.12	-0.02	-0.011	-0.17	-0.054	0.096	-0.22
ChestPainType_TA	0.045	0.043	-0.006	0.055	0.082	0.057	-0.018	-0.11	-0.12	1	-0.049	-0.017	-0.12	-0.0099	-0.0092	-0.071
RestingECG_Normal	-0.24	-0.1	-0.037	-0.1	0.029	-0.082	-0.025	0.13	-0.02	-0.049	1	-0.61	-0.068	-0.042	0.072	-0.094
RestingECG_ST	0.15	0.083	-0.13	0.15	-0.15	0.014	0.053	-0.08	-0.011	-0.017	-0.61	1	0.1	0.041	-0.042	0.093
ExerciseAngina_Y	0.2	0.13	-0.0017	0.052	-0.36	0.42	0.18	-0.3	-0.17	-0.12	-0.068	0.1	1	0.39	-0.48	0.49
ST_Slope_Flat	0.19	0.12	-0.057	0.13	-0.36	0.28	0.11	-0.34	-0.054	-0.0099	-0.042	0.041	0.39	1	-0.87	0.55
ST_Slope_Up	-0.27	-0.12	0.078	-0.21	0.4	-0.46	-0.14	0.39	0.096	-0.0092	0.072	-0.042	-0.48	-0.87	1	-0.64
HeartDisease	0.25	0.088	-0.2	0.27	-0.4	0.37	0.28	-0.4	-0.22	-0.071	-0.094	0.093	0.49	0.55	-0.64	1

Figure 2.20: Feature correlation matrix



```
std = StandardScaler()

# Fit the scaler on the training data and simultaneously transform it
x_train[Numerical_features] = std.fit_transform(x_train[Numerical_features])

# Transform the test data based on the scaler fitted on the training data
x_test[Numerical_features] = std.transform(x_test[Numerical_features])
```

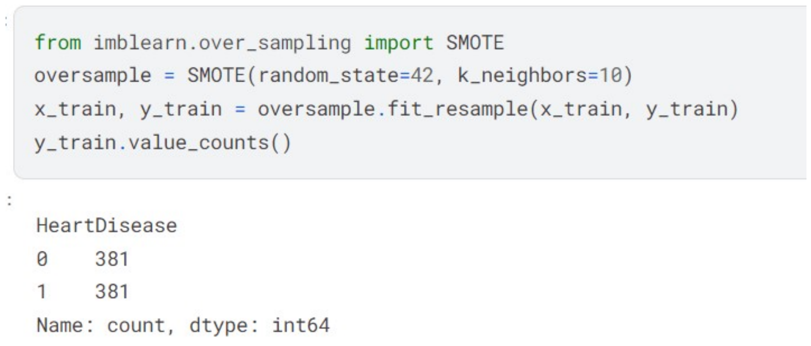
Figure 2.21: Feature correlation matrix



	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	Sex_M	ChestPainType_ATA	ChestPainType_NAP	Ches
637	-1.15	-0.98	1.14	-0.54	1.72	0.30	1	0	0	0
541	2.33	-1.58	-1.44	-0.54	-0.62	2.46	1	0	1	0
570	0.22	-0.26	0.05	-0.54	-0.66	1.05	1	0	0	0
611	0.85	0.12	-1.09	-0.54	0.03	-0.63	1	0	0	1
685	0.75	-0.70	0.56	-0.54	0.15	2.55	1	0	0	0
...	...	...	...	...	...	...	...	...	...	...
321	0.96	-1.80	-1.61	1.84	-1.05	-1.66	1	0	0	0
315	2.12	0.67	-1.61	1.84	-0.51	0.40	1	1	0	0
819	0.12	1.49	0.95	-0.54	0.34	-0.07	1	0	0	0
355	1.38	0.67	-1.61	-0.54	-0.43	-0.82	1	0	0	1
62	-0.94	0.39	0.07	-0.54	0.30	-0.82	1	0	0	0

688 rows × 15 columns

Figure 2.22: Data table feature scaling



```
from imblearn.over_sampling import SMOTE
oversample = SMOTE(random_state=42, k_neighbors=10)
x_train, y_train = oversample.fit_resample(x_train, y_train)
y_train.value_counts()
```

```
HeartDisease
0    381
1    381
Name: count, dtype: int64
```

Figure 2.23: Code for performing SMOTE

Chapter 3

Results

3.1 For Basic models

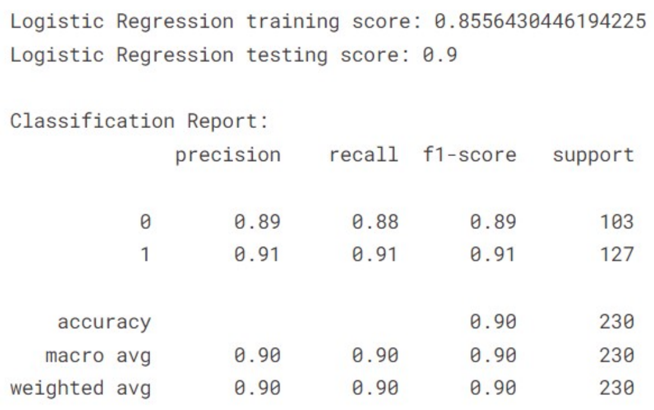
For both the algorithms the model fitting is performed on the training data and performance scores are calculated for both the training and testing datasets, which provide a quick insight into the model's accuracy and potential over-fitting. Predictions and probability estimates are then generated for the test data. A detailed classification report provides precision, recall, F1-score, and support for each class, giving a comprehensive overview of model performance across classes. The confusion matrix is visualized using a heat map, which helps in understanding the true versus predicted classifications visually. Finally, an ROC curve is plotted and the area under the curve (AUC) is calculated, offering a graphical representation of the model's ability to discriminate between the classes at various threshold settings. This curve and its associated AUC score are critical for evaluating the effectiveness of the model in classification tasks, particularly in scenarios where there is an imbalance in class distribution.

3.1.1 For Logistic Regression

The results for the logistic regression is analyzed in this section. From Figure 3.1 we can observe that the performance metrics from the Logistic Regression model demonstrate a robust classification capability on both training and testing datasets. The model achieves a training score of approximately 0.856 and a testing score of 0.9. These scores indicate a high level of accuracy, and the close values between training and testing suggest that the model is well-generalized to new data, showing no significant signs of overfitting.

The detailed classification report further highlights the model's effectiveness. It presents precision, recall, and F1-score for two classes, labeled as 0 and 1. For class 0, the model achieves a precision of 0.89, which indicates that 89% of the instances predicted as class 0 are indeed class 0. The recall for class 0 is slightly lower at 0.88, meaning the model correctly identifies 88% of all actual class 0 instances. The F1-score, which is a harmonic mean of precision and recall, stands at 0.89, underscoring a balanced classification performance for this class.

Similarly, for class 1, the precision and recall are both at 0.91, with an F1-score also at 0.91, indicating a slightly higher performance for class 1 than for class 0. These values suggest that the model is slightly more effective at identifying and predicting class 1 instances compared to class 0. The overall accuracy of the model across both classes is 0.90, which is excellent for many applications. The macro and weighted averages of precision, recall, and F1-scores are all at 0.90, confirming the model's consistent performance across different evaluation metrics.



```
Logistic Regression training score: 0.8556430446194225
Logistic Regression testing score: 0.9

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.88      0.89       103
           1       0.91      0.91      0.91       127

    accuracy                           0.90       230
   macro avg       0.90      0.90      0.90       230
weighted avg       0.90      0.90      0.90       230
```

Figure 3.1: Performance metrics for logistic regression

As observed in Figure 3.2 from the confusion matrix, it can be observed that the model correctly predicted 91 instances of Class 0 and 116 instances of Class 1, indicating high true positive rates. There are small counts of misclassifications (12 as false positives and 11 as false negatives), suggesting that the model has a good balance between sensitivity and precision.

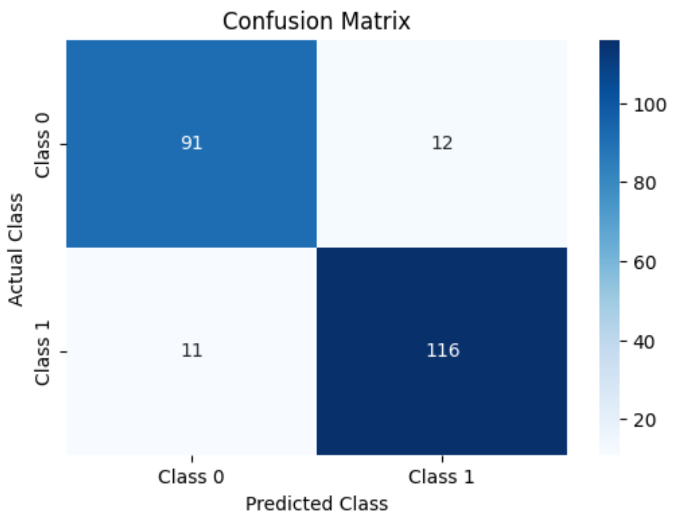


Figure 3.2: Confusion matrix for logistic regression

As seen in Figure 3.3 the ROC curve with its AUC score of 0.94 shows that the model is quite accurate in distinguishing between the classes. The curve stays far above the diagonal line of no-discrimination, which means the model does an excellent job of classifying between the two classes across all thresholds. Since the AUC is close to the perfect score of 1, it tells us that the model is very likely to correctly identify true positives and true negatives. Overall, these visualizations suggest that the logistic regression classifier performs robustly on this dataset.

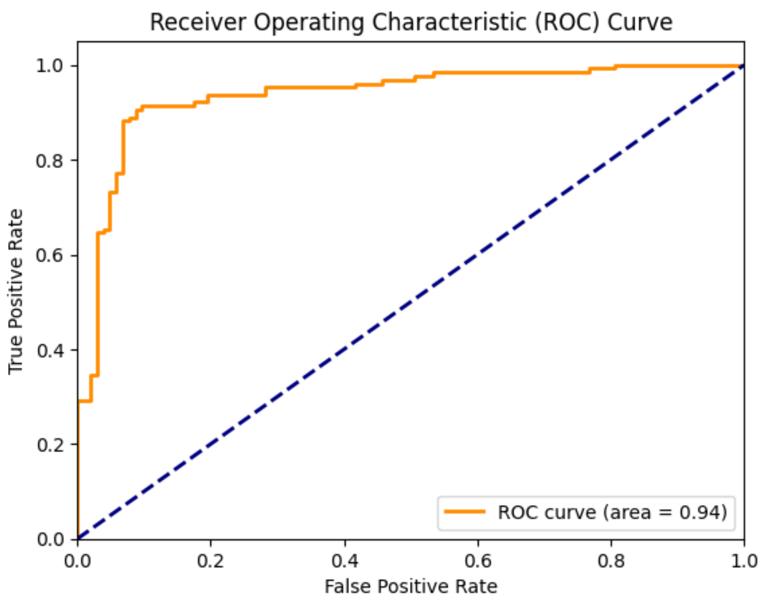


Figure 3.3: ROC curve for logistic regression

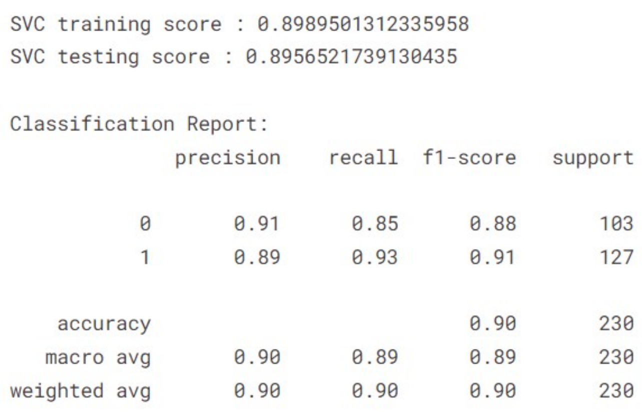
3.1.2 For Support Vector Classification

As seen in Figue 3.4, Support Vector Classification (SVC) has demonstrated a high level of proficiency in classifying the test data, with training and testing scores of approximately 0.899 and 0.896, respectively. These scores indicate that the model has learned well from the training data and has a strong ability to generalize its predictions to unseen data, with very little difference between its performance on training and testing sets—suggesting that the model is neither overfitting nor underfitting.

The classification report provides a breakdown of the model's precision, recall, and F1-score for each class. For Class 0, the precision is 0.91, meaning that 91% of the instances predicted to be in this class are correct. However, the recall of 0.85 for Class 0 indicates that the model captures 85% of all actual instances of this class, missing 15%. For Class 1, the model shows a precision of 0.89 and a recall of 0.93, indicating that while it's slightly less precise in its predictions for Class 1, it's better at capturing most of the instances of this class. The F1-scores, which balance precision and recall, are 0.88 for Class 0 and 0.91 for Class 1, confirming a strong performance, particularly for Class 1. The overall accuracy stands at 0.90, confirming consistency across various aspects of the model's performance. These metrics shows the SVC's solid classification capability, making it a reliable option for our classification model.

As in Figure 3.5 the confusion matrix for the Support Vector Classifier (SVC) model indicates a total of 88 true positives and 118 true negatives, meaning the model correctly identified these many instances of Class 0 and Class 1 respectively. There were, however, 15 false positives (Class 0 instances incorrectly labeled as Class 1) and 9 false negatives (Class 1 instances incorrectly labeled as Class 0). The number of mis-classifications is relatively small, indicating a solid overall performance.

Furthermore as in Figure 3.6, the ROC curve complements the confusion matrix by showing a strong AUC score of 0.95. This is close to the ideal score of 1 and suggests that the model is highly capable of differentiating between the two classes.



```
SVC training score : 0.8989501312335958
SVC testing score : 0.8956521739130435

Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.85      0.88       103
           1       0.89      0.93      0.91       127

    accuracy                           0.90       230
   macro avg       0.90      0.89      0.89       230
weighted avg       0.90      0.90      0.90       230
```

Figure 3.4: Performance metrics for SVM

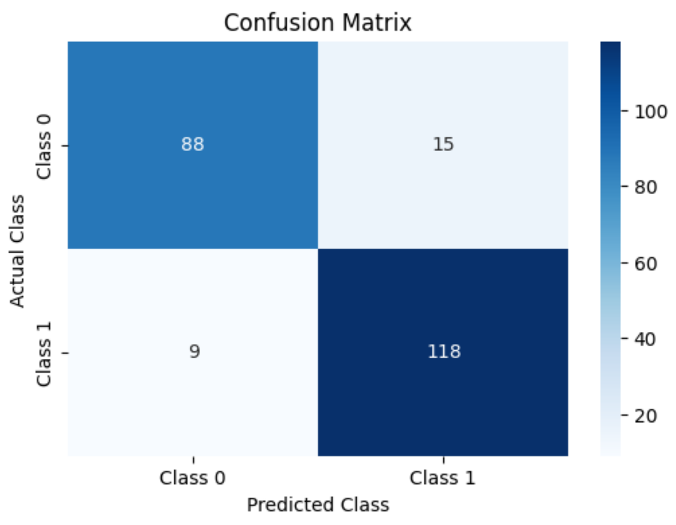


Figure 3.5: Confusion matrix for SVM

3.2 Hyper Parameter Tuning

Hyperparameter tuning is the process of finding the optimal set of parameters for a machine learning algorithm that are not directly learned from the data. These parameters, known as hyperparameters, govern the overall behavior of a machine learning model and can significantly affect its performance. Unlike model parameters that are learned during training, hyperparameters are set prior to the learning process and remain constant during training. Under this section, hyperparameter tuning was performed for logistic regression and SVM in order to explore the possibility of obtaining the optimal hyperparameter variables to maximize the accuracies.

3.2.1 For Logistic Regression

In the initial stage the default hyperparameters for logistic regression was explored as in Figure 3.7. Then hyperparameters like 'C' (regularization strength), 'penalty' (type of regularization) and 'solver' which significantly impact model performance [15] were chosen and they were altered as in Figure 3.8 to find the best performing parameters using GridsearchCV.

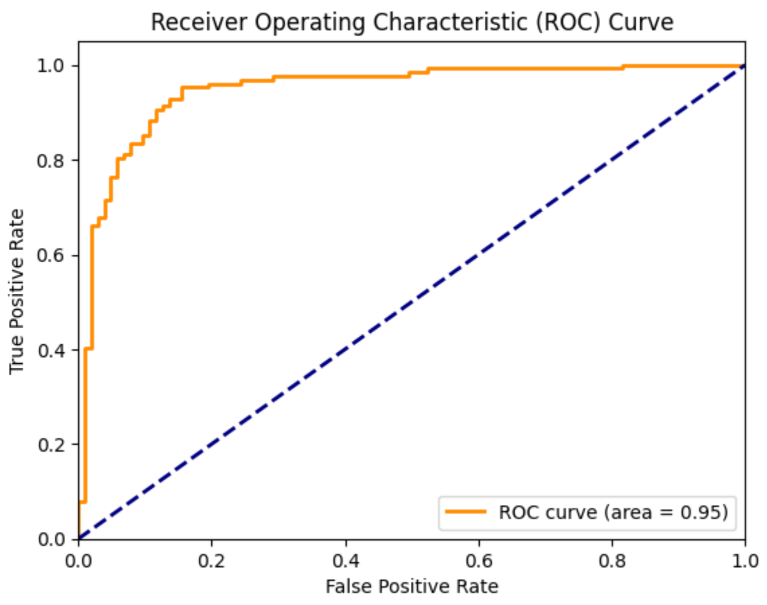


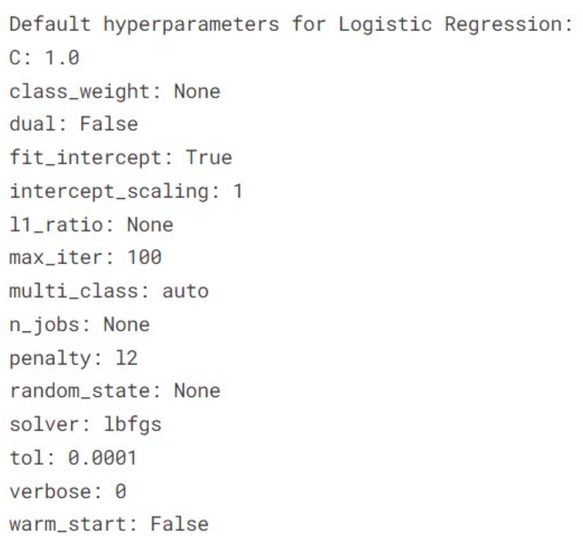
Figure 3.6: ROC curve for SVM

GridSearchCV is a hyperparameter tuning method that streamlines model validation and optimization without the need for manually segmenting a separate validation set from the training data. By implementing k-fold cross-validation internally, GridSearchCV systematically splits the training data into 'k' subsets. For each hyperparameter combination from the specified grid, it trains the model on 'k-1' subsets and validates performance on the remaining subset. This can be visualized as in Figure 3.9. This process repeats such that each subset is used for validation once, ensuring comprehensive evaluation across the entire training dataset. Therefore, the exhaustive nature of GridSearchCV not only simplifies the model selection process but also maximizes the use of available data for training and validation, providing a robust assessment of model performance.

The convenience of GridSearchCV is particularly evident in its automated approach to model validation. Since it internally handles the division of data into training and validation sets, it eliminates the additional step of creating a validation set manually. This aspect of GridSearchCV ensures that the model is exposed to various training and validation scenarios, reflecting a more generalized performance.

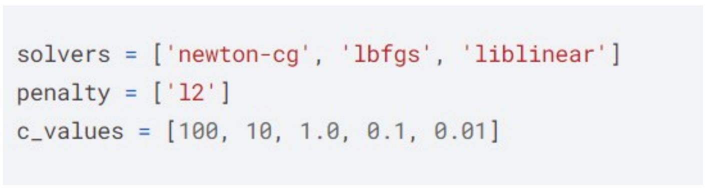
Once GridSearchCV has assessed all possible combinations across the cross-validation folds, it selects the combination that yields the best validation performance as in Figure 3.10. This 'best' model is usually determined by a scoring function like accuracy, precision, recall, or a combination thereof. Next the model is retrained using the best parameters as in Figure 3.11 and evaluated on the unseen testing data set.

After hyperparameter tuning was performed the performance metrics were obtained as in Figure 3.12. As it can be observed there were no change of metric after hyperparameter tuning was done. When the performance metrics remain unchanged after hyperparameter tuning, as indicated in Figure 3.12, it often suggests that the original model was already operating at or near its optimal capacity given the current features and dataset. This could mean that the default hyperparameters used by the model were suitable for the data and the problem at hand. In such cases, the stability of performance metrics post-tuning signifies that the model is robust, and



```
Default hyperparameters for Logistic Regression:
C: 1.0
class_weight: None
dual: False
fit_intercept: True
intercept_scaling: 1
l1_ratio: None
max_iter: 100
multi_class: auto
n_jobs: None
penalty: l2
random_state: None
solver: lbfgs
tol: 0.0001
verbose: 0
warm_start: False
```

Figure 3.7: Default hyperparameters for logistic regression



```
solvers = ['newton-cg', 'lbfgs', 'liblinear']
penalty = ['l2']
c_values = [100, 10, 1.0, 0.1, 0.01]
```

Figure 3.8: Hyperparameter variations for logistic regression

further hyperparameter adjustments may yield little to no improvement.

In this scenario, since hyperparameter tuning didn't improve the model, it makes sense to stick with the current parameters, particularly if the time and computational resources required for tuning don't seem worth the minimal benefits. However, using the"best" parameters identified during the tuning process could still be valuable. Even though they didn't enhance performance now, they might offer useful insights or serve as a good foundation for future adjustments when new data comes in or as the project's requirements change. Adopting these tuned parameters can also help the model handle small data variations better, ensuring stability over time.

3.2.2 For Support Vector Classification

The same procedure as for logistic regression was followed for the support vector classification as well. For the SVC model, hyperparameters such as the kernel type, the penalty parameter 'C', and the gamma value were carefully selected and varied as in Figure 3.13 within a predefined grid. GridSearchCV was employed to search through this space, utilizing k-fold cross-validation to robustly assess the impact of each parameter combination on the model's accuracy and overall effectiveness. After 200 fits the ideal hyperparaemters were obtained as in Figure 3.14.

Next as in logistic regression the SVC model was trained using the 'best' parameters and the accuracies were obtained for the unseen test data set.

The performance metrics obtained with the best parameters after hyperparameter tuning indicate that the support vector classification (SVC) model is performing robustly. The training score is 0.899 and the testing score is 0.896, which are commend-

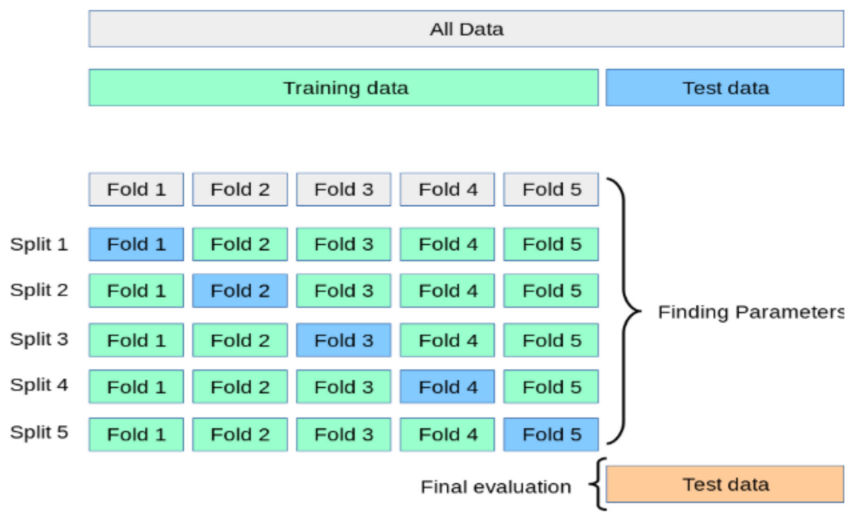


Figure 3.9: k-fold cross-validation during GridsearchCV [1]

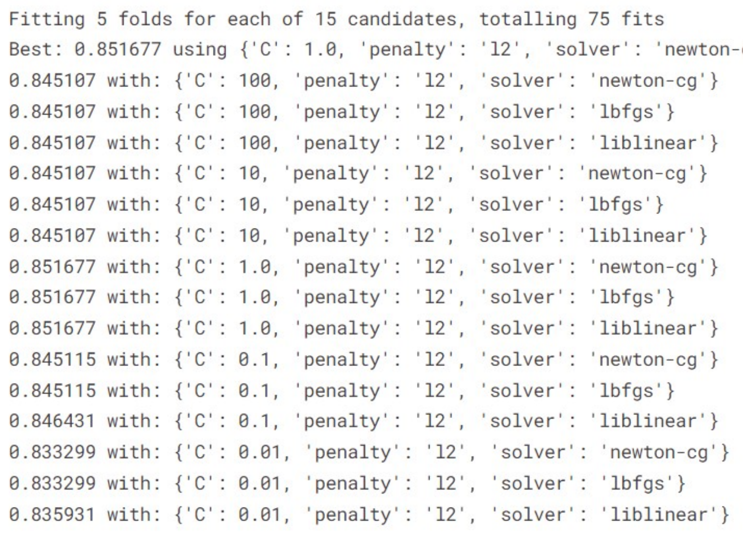
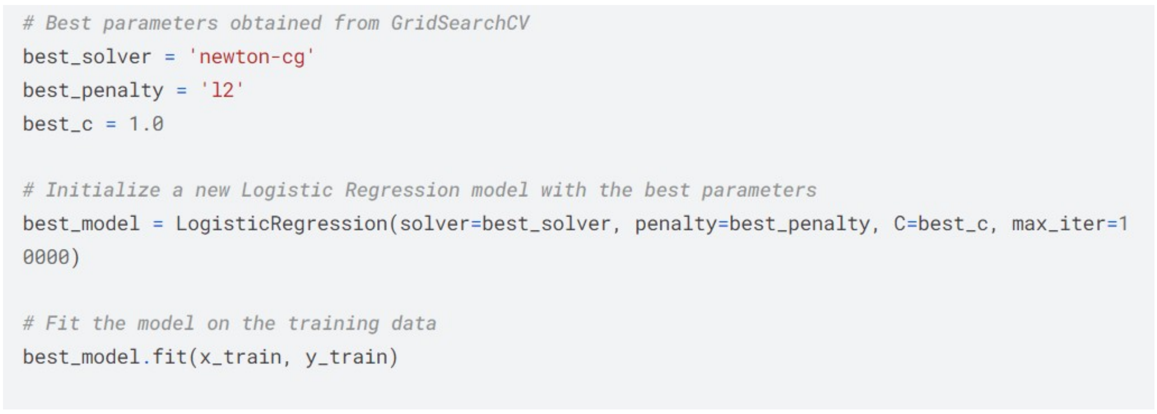

```
Fitting 5 folds for each of 15 candidates, totalling 75 fits
Best: 0.851677 using {'C': 1.0, 'penalty': 'l2', 'solver': 'newton-cg'}
0.845107 with: {'C': 100, 'penalty': 'l2', 'solver': 'newton-cg'}
0.845107 with: {'C': 100, 'penalty': 'l2', 'solver': 'lbfgs'}
0.845107 with: {'C': 100, 'penalty': 'l2', 'solver': 'liblinear'}
0.845107 with: {'C': 10, 'penalty': 'l2', 'solver': 'newton-cg'}
0.845107 with: {'C': 10, 'penalty': 'l2', 'solver': 'lbfgs'}
0.845107 with: {'C': 10, 'penalty': 'l2', 'solver': 'liblinear'}
0.851677 with: {'C': 1.0, 'penalty': 'l2', 'solver': 'newton-cg'}
0.851677 with: {'C': 1.0, 'penalty': 'l2', 'solver': 'lbfgs'}
0.851677 with: {'C': 1.0, 'penalty': 'l2', 'solver': 'liblinear'}
0.845115 with: {'C': 0.1, 'penalty': 'l2', 'solver': 'newton-cg'}
0.845115 with: {'C': 0.1, 'penalty': 'l2', 'solver': 'lbfgs'}
0.846431 with: {'C': 0.1, 'penalty': 'l2', 'solver': 'liblinear'}
0.833299 with: {'C': 0.01, 'penalty': 'l2', 'solver': 'newton-cg'}
0.833299 with: {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
0.835931 with: {'C': 0.01, 'penalty': 'l2', 'solver': 'liblinear'}
```

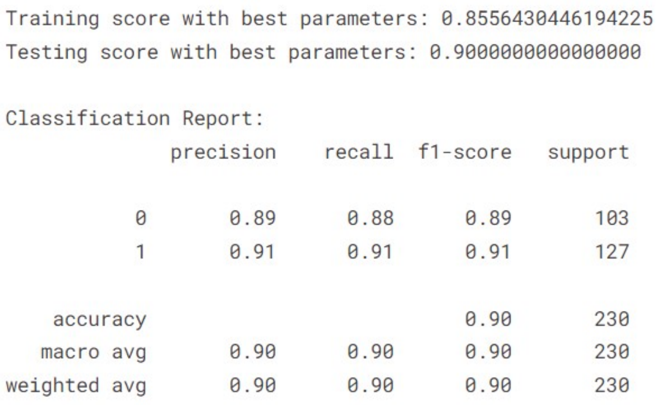
Figure 3.10: Obtaining the ideal hyperparameters for logistic regression

ably high and suggest that the model is well-calibrated and generalizes effectively to unseen data. The classification report provides further insight into the model's performance across different classes. For Class 0, the model achieves a precision of 0.91 and a recall of 0.85, resulting in an F1-score of 0.88. For Class 1, the precision is 0.89 with a recall of 0.93, leading to an F1-score of 0.91. These scores result in an overall accuracy of 0.90, with the macro and weighted averages for precision, recall, and F1-score also standing around 0.90. These metrics reflect a well-balanced performance across both classes, indicating that the model effectively handles different types of classification errors. But as previously in this case also, results were found to be the same as those obtained before the hyperparameter tuning process. This suggests that the original model configuration was already optimal, and the default settings provided by the SVC were well-suited for this particular dataset. In practical terms, this means that while the hyperparameter tuning process did not yield improvements in this instance, it did confirm the robustness and efficacy of the initial model setup.



```python
# Best parameters obtained from GridSearchCV
best_solver = 'newton-cg'
best_penalty = 'l2'
best_c = 1.0

# Initialize a new Logistic Regression model with the best parameters
best_model = LogisticRegression(solver=best_solver, penalty=best_penalty, C=best_c, max_iter=1
0000)

# Fit the model on the training data
best_model.fit(x_train, y_train)
```

Figure 3.11: Retraining the model using the best hyperparameters for logistic regression



```
Training score with best parameters: 0.8556430446194225
Testing score with best parameters: 0.9000000000000000

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.88      0.89       103
           1       0.91      0.91      0.91       127

    accuracy                           0.90       230
   macro avg       0.90      0.90      0.90       230
weighted avg       0.90      0.90      0.90       230
```

Figure 3.12: Obtained Performance metrics after hyperparameter tuning for logistic regression

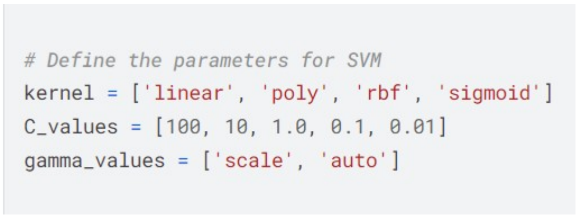
3.3 Developing Ensemble Models

Since any significance improvement was not seen in the logistic regression and SVM models after hyperparameter tuning it was decided to look on to the possibility of achieving any possible improved accuracy by combining the two models, which is said to be developing a ensemble model.

Ensemble models are a powerful technique in machine learning that combines the predictions of multiple individual models to produce a more accurate and robust prediction. By using the collective wisdom of diverse models, ensemble methods often outperform any single model alone. In the project three main ensemble techniques were used to work eith the two models, logistic regression and SVM. They are soft voting, hard voting and model stacking.

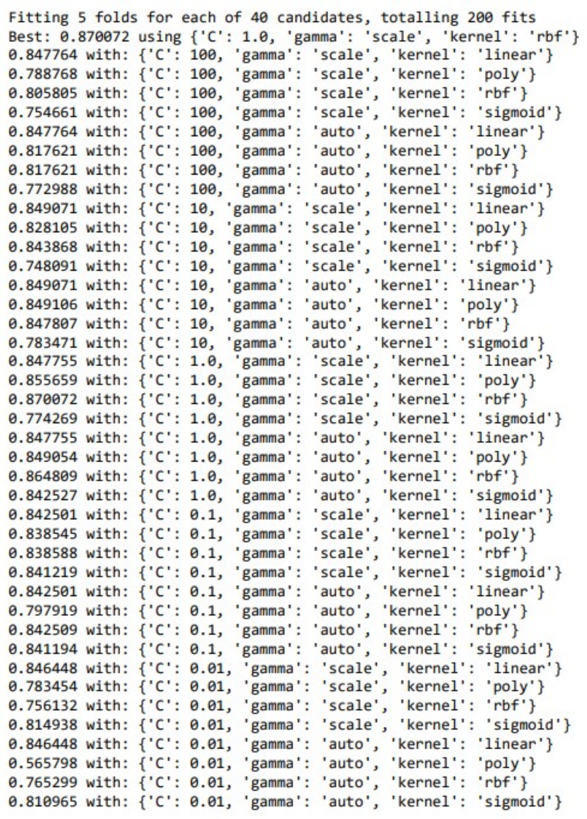
3.3.1 Ensemble Model using Soft Voting

Soft voting, considers not only the predictions of each individual model but also the confidence or probability associated with those predictions. The final prediction is a weighted average of the predicted probabilities across all models, taking into account the level of confidence in each prediction.



```python
# Define the parameters for SVM
kernel = ['linear', 'poly', 'rbf', 'sigmoid']
C_values = [100, 10, 1.0, 0.1, 0.01]
gamma_values = ['scale', 'auto']
```

Figure 3.13: Hyperparameter variations for SVC



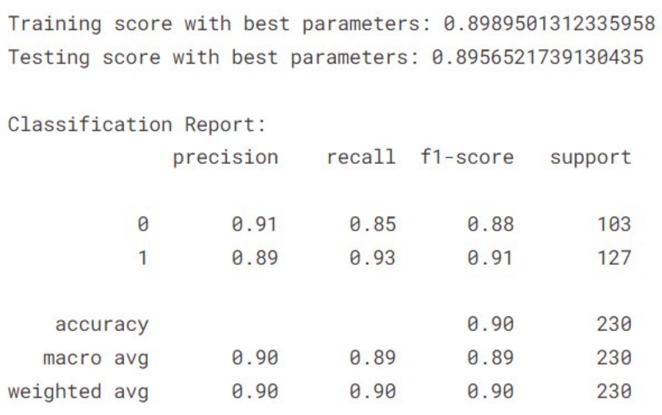
```
Fitting 5 folds for each of 40 candidates, totalling 200 fits
Best: 0.870072 using {'C': 1.0, 'gamma': 'scale', 'kernel': 'rbf'}
0.847764 with: {'C': 100, 'gamma': 'scale', 'kernel': 'linear'}
0.788768 with: {'C': 100, 'gamma': 'scale', 'kernel': 'poly'}
0.805805 with: {'C': 100, 'gamma': 'scale', 'kernel': 'rbf'}
0.754661 with: {'C': 100, 'gamma': 'scale', 'kernel': 'sigmoid'}
0.847764 with: {'C': 100, 'gamma': 'auto', 'kernel': 'linear'}
0.817621 with: {'C': 100, 'gamma': 'auto', 'kernel': 'poly'}
0.817621 with: {'C': 100, 'gamma': 'auto', 'kernel': 'rbf'}
0.772988 with: {'C': 100, 'gamma': 'auto', 'kernel': 'sigmoid'}
0.849071 with: {'C': 10, 'gamma': 'scale', 'kernel': 'linear'}
0.828105 with: {'C': 10, 'gamma': 'scale', 'kernel': 'poly'}
0.843868 with: {'C': 10, 'gamma': 'scale', 'kernel': 'rbf'}
0.748091 with: {'C': 10, 'gamma': 'scale', 'kernel': 'sigmoid'}
0.849071 with: {'C': 10, 'gamma': 'auto', 'kernel': 'linear'}
0.849106 with: {'C': 10, 'gamma': 'auto', 'kernel': 'poly'}
0.847807 with: {'C': 10, 'gamma': 'auto', 'kernel': 'rbf'}
0.783471 with: {'C': 10, 'gamma': 'auto', 'kernel': 'sigmoid'}
0.847755 with: {'C': 1.0, 'gamma': 'scale', 'kernel': 'linear'}
0.855659 with: {'C': 1.0, 'gamma': 'scale', 'kernel': 'poly'}
0.870072 with: {'C': 1.0, 'gamma': 'scale', 'kernel': 'rbf'}
0.774269 with: {'C': 1.0, 'gamma': 'scale', 'kernel': 'sigmoid'}
0.847755 with: {'C': 1.0, 'gamma': 'auto', 'kernel': 'linear'}
0.849054 with: {'C': 1.0, 'gamma': 'auto', 'kernel': 'poly'}
0.864809 with: {'C': 1.0, 'gamma': 'auto', 'kernel': 'rbf'}
0.842527 with: {'C': 1.0, 'gamma': 'auto', 'kernel': 'sigmoid'}
0.842501 with: {'C': 0.1, 'gamma': 'scale', 'kernel': 'linear'}
0.838545 with: {'C': 0.1, 'gamma': 'scale', 'kernel': 'poly'}
0.838588 with: {'C': 0.1, 'gamma': 'scale', 'kernel': 'rbf'}
0.841219 with: {'C': 0.1, 'gamma': 'scale', 'kernel': 'sigmoid'}
0.842501 with: {'C': 0.1, 'gamma': 'auto', 'kernel': 'linear'}
0.797919 with: {'C': 0.1, 'gamma': 'auto', 'kernel': 'poly'}
0.842509 with: {'C': 0.1, 'gamma': 'auto', 'kernel': 'rbf'}
0.841194 with: {'C': 0.1, 'gamma': 'auto', 'kernel': 'sigmoid'}
0.846448 with: {'C': 0.01, 'gamma': 'scale', 'kernel': 'linear'}
0.783454 with: {'C': 0.01, 'gamma': 'scale', 'kernel': 'poly'}
0.756132 with: {'C': 0.01, 'gamma': 'scale', 'kernel': 'rbf'}
0.814938 with: {'C': 0.01, 'gamma': 'scale', 'kernel': 'sigmoid'}
0.846448 with: {'C': 0.01, 'gamma': 'auto', 'kernel': 'linear'}
0.565798 with: {'C': 0.01, 'gamma': 'auto', 'kernel': 'poly'}
0.765299 with: {'C': 0.01, 'gamma': 'auto', 'kernel': 'rbf'}
0.810965 with: {'C': 0.01, 'gamma': 'auto', 'kernel': 'sigmoid'}
```

Figure 3.14: Hyperparameter variations for the SVC model

Here the models were set to the soft voting classifier as in Figure 3.16. Next the accuracy score were obtained as in previous cases as in Figure 3.17. As observed the accuracies were again in the same range as of the pure logistic regression and SVM models. Next hyper-parameter tuning was done and yet the accuracies were obtained as in Figure 3.18. Still the accuracies were as the same range as in previous cases.

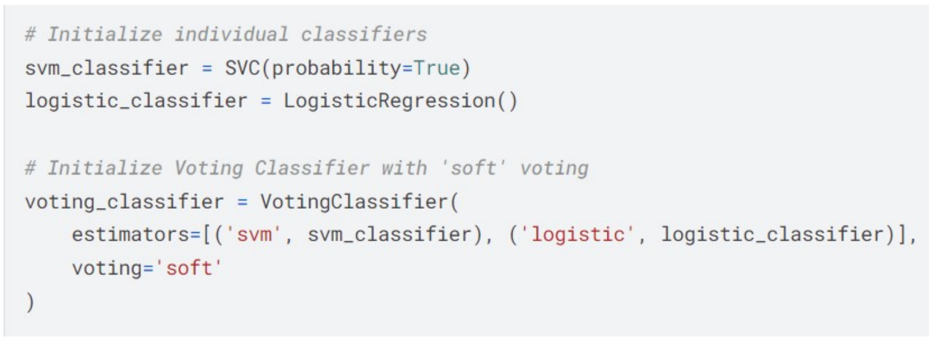
3.3.2 Ensemble Model using Hard Voting

In hard voting, each individual model in the ensemble independently analyzes the input data and makes its own prediction. Once each model has made its prediction, the final decision is determined by a simple majority vote among all the models. This method is particularly effective when the base models have diverse strengths and weaknesses, allowing them to complement each other's performance and make more robust predictions. By aggregating the predictions of multiple models, hard voting can mitigate the impact of individual model biases and uncertainties, resulting in a more reliable and accurate ensemble prediction.



```
Training score with best parameters: 0.8989501312335958
Testing score with best parameters: 0.8956521739130435

Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.85      0.88       103
           1       0.89      0.93      0.91       127

    accuracy                           0.90       230
   macro avg       0.90      0.89      0.89       230
weighted avg       0.90      0.90      0.90       230
```

Figure 3.15: Obtained Performance metrics after hyperparameter tuning for the SVC model



```python
# Initialize individual classifiers
svm_classifier = SVC(probability=True)
logistic_classifier = LogisticRegression()

# Initialize Voting Classifier with 'soft' voting
voting_classifier = VotingClassifier(
    estimators=[('svm', svm_classifier), ('logistic', logistic_classifier)],
    voting='soft'
)
```

Figure 3.16: soft voting classifier

Here as in the previous case the models were set to the hard voting classifier as in Figure 3.19. Next the accuracy score were obtained as in previous cases as in Figure 3.20. As observed the accuracies were again in the same range as of the pure logistic regression and SVM models. Next hyper-parameter tuning was done and yet the accuracies were obtained as in Figure 3.21. Still the accuracies were as the same range as in previous cases.

3.3.3 Ensemble Model using Model Stacking

Model stacking is a technique in machine learning where predictions from multiple base models are combined using a meta-model to make the final prediction. Base models are trained on subsets of the data, and their predictions serve as features for the meta-model. The meta-model learns how to effectively combine these predictions, capturing more complex relationships and often leading to improved performance compared to individual models.

As in Figure 3.22 model stacking is implemented using scikit-learn's Stacking Classifier. It initializes individual classifiers (SVC and LogisticRegression), then constructs a stacking ensemble with these base models. The ensemble is trained on the training data and evaluated for performance on both training and testing sets. This approach enables the combination of predictions from multiple models to potentially enhance predictive accuracy, though a final estimator for the meta-model is not spec-

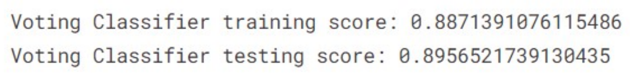

```
Voting Classifier training score: 0.8871391076115486
Voting Classifier testing score: 0.8956521739130435
```

Figure 3.17: Accuracies of soft voting classifier

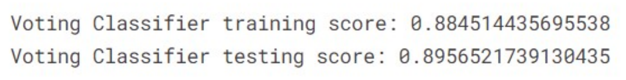

```
Voting Classifier training score: 0.884514435695538
Voting Classifier testing score: 0.8956521739130435
```

Figure 3.18: Accuracies of soft voting classifier after hyper-parameter tuning

ified in this instance. Accuracies were obtained as in Figure 3.23. As observed the accuracies were again in the same range as of the pure logistic regression and SVM models. Next hyper-parameter tuning was done and yet the accuracies were obtained as in Figure 3.24. Still the accuracies were as the same range as in previous cases.

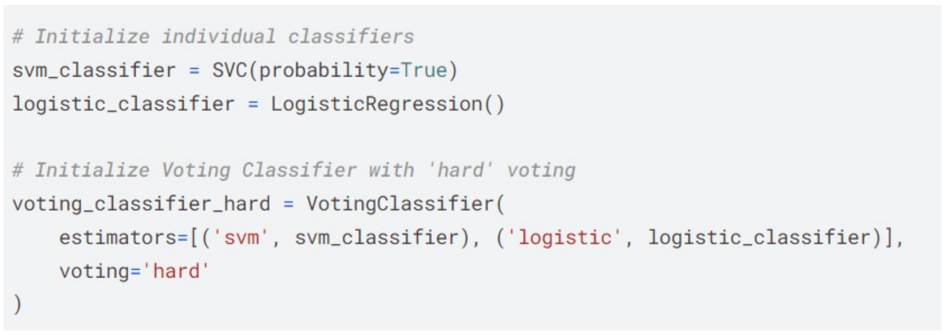
3.4 Summary of Results

The summary of the best test accuracies for all the models developed using logistic regression and SVM can be studied as in Table 3.1.

Table 3.1: Summary of the Best Test Accuracies of the Models

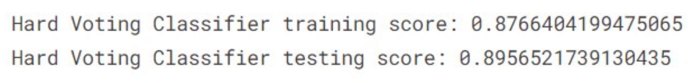
Model Name	Best Test Accuracy
Logistic Regression	0.900
SVM	0.895
Ensemble model with soft voting	0.895
Ensemble model with hard voting	0.895
Ensemble model with model stacking	0.895
Logistic Regression with hyper-parameter tuning	0.900
SVM with hyper-parameter tuning	0.895
Ensemble model with soft voting and hyper-parameter tuning	0.895
Ensemble model with hard voting and hyper-parameter tuning	0.895
Ensemble model with model stacking and hyper-parameter tuning	0.891

As seen in Table 3.1, the vanilla logistic regression and SVM models demonstrated strong performance, even without the implementation of advanced techniques such as hyper-parameter tuning or ensemble methods. This resilience suggests that these models were inherently well-optimized for the task at hand. Despite the absence of additional optimizations, both models achieved impressive accuracy rates of nearly 90% and AU-ROC scores of 0.95. These results underscore the effectiveness of these fundamental machine learning approaches and highlight their suitability for the given classification task.



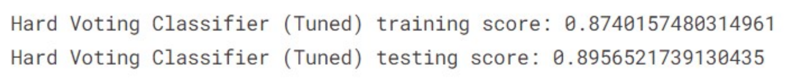
```
# Initialize individual classifiers
svm_classifier = SVC(probability=True)
logistic_classifier = LogisticRegression()

# Initialize Voting Classifier with 'hard' voting
voting_classifier_hard = VotingClassifier(
    estimators=[('svm', svm_classifier), ('logistic', logistic_classifier)],
    voting='hard'
)
```

Figure 3.19: Hard voting classifier



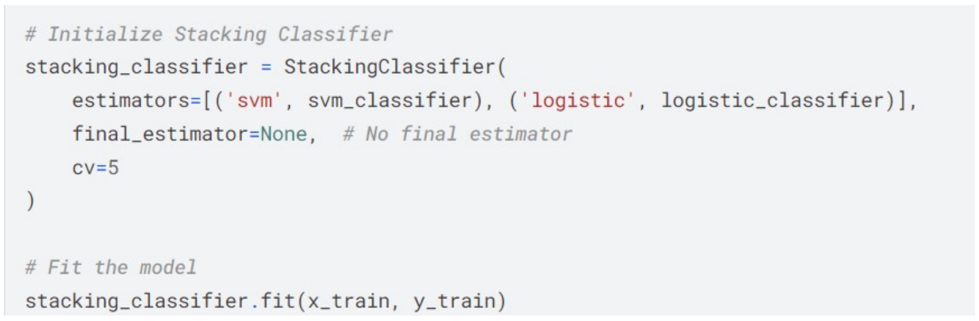
```
Hard Voting Classifier training score: 0.8766404199475065
Hard Voting Classifier testing score: 0.8956521739130435
```

Figure 3.20: Accuracies of hard voting classifier



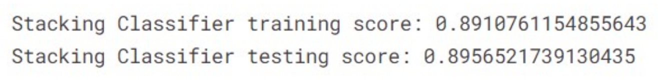
```
Hard Voting Classifier (Tuned) training score: 0.8740157480314961
Hard Voting Classifier (Tuned) testing score: 0.8956521739130435
```

Figure 3.21: Accuracies of hard voting classifier after hyper-parameter tuning



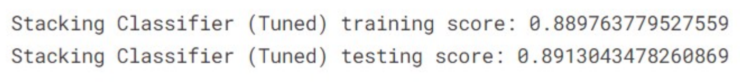
```
# Initialize Stacking Classifier
stacking_classifier = StackingClassifier(
    estimators=[('svm', svm_classifier), ('logistic', logistic_classifier)],
    final_estimator=None,  # No final estimator
    cv=5
)

# Fit the model
stacking_classifier.fit(x_train, y_train)
```

Figure 3.22: Model stacking classifier



```
Stacking Classifier training score: 0.8910761154855643
Stacking Classifier testing score: 0.8956521739130435
```

Figure 3.23: Accuracies of Model stacking classifier



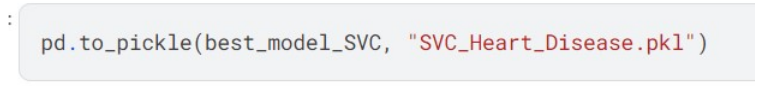
```
Stacking Classifier (Tuned) training score: 0.889763779527559
Stacking Classifier (Tuned) testing score: 0.8913043478260869
```

Figure 3.24: Accuracies of hard Model stacking after hyper-parameter tuning

Chapter 4

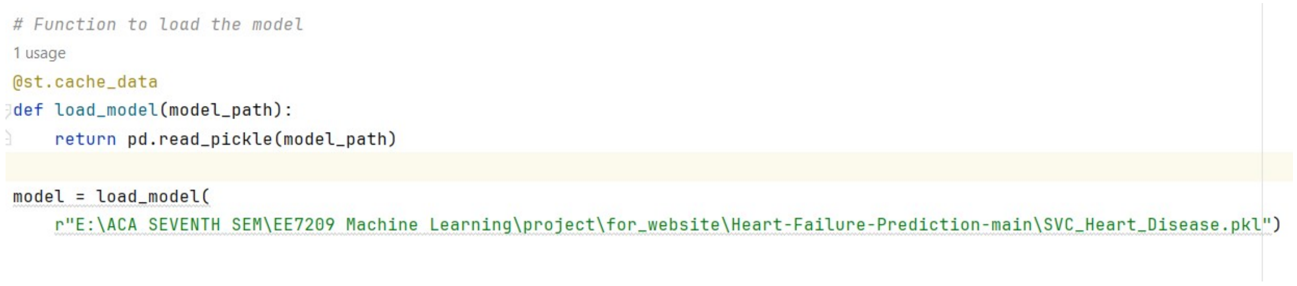
Developing a Website Using the Model Built

Both logistic regression and SVM models showed almost equal accuracies for training and testing scores. The SVM model after hyper-parameter optimization was selected as the best model since it performed equally well for both the train and the test set data. As in Figure 4.1 the build model was downloaded as a pickle file and used to built the website to predict heart failure as in Figure 4.2.



```
pd.to_pickle(best_model_SVC, "SVC_Heart_Disease.pkl")
```

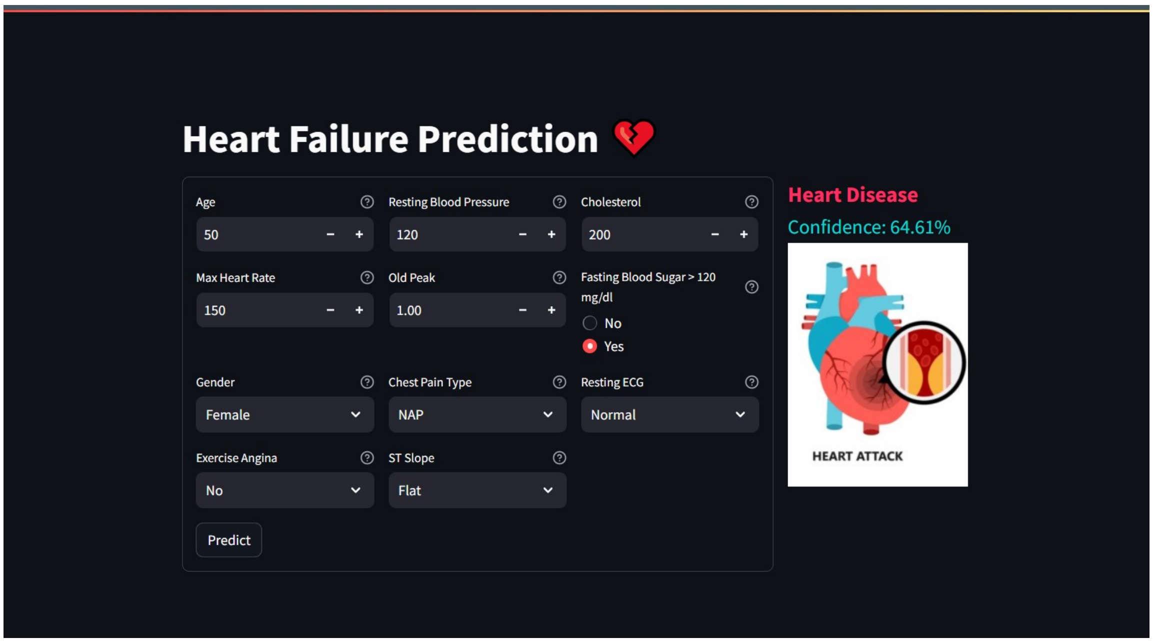
Figure 4.1: Downloading the best SVC model as pickle file



```
# Function to load the model
1 usage
@st.cache_data
def load_model(model_path):
    return pd.read_pickle(model_path)

model = load_model(
    r"E:\ACA SEVENTH SEM\EE7209 Machine Learning\project\for_website\Heart-Failure-Prediction-main\SVC_Heart_Disease.pkl")
```

Figure 4.2: Importing the model to do the predictions

The website was developed using Streamlit, a popular Python library for building interactive web applications for machine learning and data science projects. Streamlit simplifies the process of creating web applications by allowing developers to focus on the core functionality of their machine learning models or data analysis tools without focusing much about web development intricacies. With Streamlit, it can be easily created widgets and interactive components to input data, visualize results, and make predictions, all within a Python script. The snaps of the website developed can be seen in Figure 4.3 and Figure 4.4.



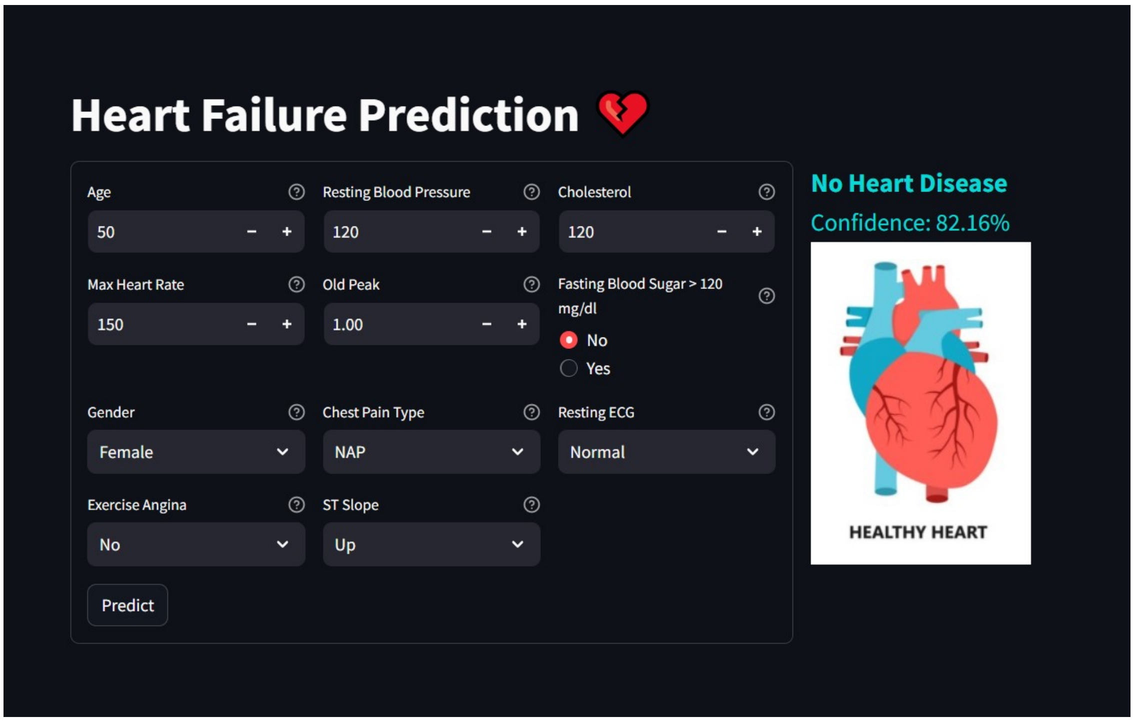
Heart Failure Prediction 💔

Age: 50
Resting Blood Pressure: 120
Cholesterol: 200
Max Heart Rate: 150
Old Peak: 1.00
Fasting Blood Sugar > 120 mg/dl: No [✓ Yes]
Gender: Female
Chest Pain Type: NAP
Resting ECG: Normal
Exercise Angina: No
ST Slope: Flat

Predict

Heart Disease
Confidence: 64.61%

Figure 4.3: Snap shot of the website for a captured heart disease occasion



Heart Failure Prediction 💔

Age: 50
Resting Blood Pressure: 120
Cholesterol: 120
Max Heart Rate: 150
Old Peak: 1.00
Fasting Blood Sugar > 120 mg/dl: [✓ No] Yes
Gender: Female
Chest Pain Type: NAP
Resting ECG: Normal
Exercise Angina: No
ST Slope: Up

Predict

No Heart Disease
Confidence: 82.16%

Figure 4.4: Snap shot of the website for a captured heart non-disease occasion

Chapter 5

Impact of the Project on Real-World Applications

The successful deployment of our machine learning models, particularly the SVM model as in Figure 4.3 via a Streamlit web interface, marks a significant advancement in making predictive healthcare tools accessible and functional in real-world settings. The models' commendable accuracy, achieved without complex tuning or ensemble methods, highlights their suitability for this critical task. This approach not only gives access to predictive analytics in healthcare but also paves the way for wider adoption and trust in machine learning solutions across the sector.

- Enhancing Healthcare Delivery : By integrating such predictive models with a user-friendly web interface, we can provide instant, actionable insights straight into the hands of the healthcare professional. This would greatly speed up the decision-making processes, reducing the administrative burden and therefore directing itself more towards the care of the person. Early prediction of heart failure would result in effective strategies in intervening, thus reducing the rate of patient readmission into the hospital and possibly improving patient outcomes.

- Scalability and Future Prospects: The framework established through this project provides a model that can scale to other predictive needs within the healthcare industry. Future work will have to make these decisions a little more informed and fine-tune complex learning algorithms to finally enhance accuracy. Furthermore, it has the capability of incorporating electronic health record (EHR) systems so that it smooths the workflow while improving the care of patients within the health service.

- Educational and Collaborative Opportunities: It also serves to prove a concept that basic machine learning techniques can be used quite effectively in such environments and should be the basis for future research and development in the area. Overall, the project emphasizes not just the practical benefits of machine learning applied to healthcare but establishes a benchmark for further innovations in the industry that may possibly change the delivery and management landscape of healthcare by means of technology solutions.

References

[1] scikit-learn developers, "scikit-learn: Machine learning in python." https://scikit-learn.org/stable/index.html, Accessed: 2024.

[2] National Center for Biotechnology Information, *Bookshelf.* National Center for Biotechnology Information (US), 2000–. Accessed: April 17, 2024.

[3] N. A. Chatterjee, S. Gupta, V. Gupta, A. Juneja, and T. M. Patel, "Prevalence and predictors of anxiety and depression among family physicians: an indian study," *Journal of Family Medicine and Primary Care*, vol. 4, no. 2, pp. 299–303, 2015.

[4] S. Dumont, N. Jacobs, and P. Fournier, "The impact of free obstetric care in low-income countries: a case study in a private congolese hospital," *BMC Health Services Research*, vol. 13, no. 1, p. 497, 2013.

[5] F. Soriano, "Heart failure prediction dataset," 2024. Last accessed April 10, 2024.

[6] M. C. Staff, "High blood pressure dangers: Hypertension's effects on your body," 2023. Last accessed 3 April 2024.

[7] J. Huizen, "What is serum cholesterol?," 2021. Last accessed 10 April 2024.

[8] H. H. Publishing, "How it's made: Cholesterol production in your body." https://www.health.harvard.edu/heart-health/how-its-made-cholesterol-production-in-your-body. Accessed: April 17, 2024.

[9] C. for Disease Control and prevention, "How diabetes affects your heart," 2022. Last accessed 10 April 2024.

[10] M. McMillen, "Your heart rate: What does it say?," 2024. Last accessed 10 April 2024.

[11] Physiopedia, "Heart rate," 2022. Last accessed 10 April 2024.

[12] E. Burns and R. Buttner, "The st segment," 2022. Last accessed 10 April 2024.

[13] D. L. G. Elias B. Hanna, "St-segment depression and t-wave inversion: Classification, differential diagnosis, and caveats," 2011. Last accessed 10 April 2024.

[14] Cleveland Clinic, "Cholesterol numbers: What do they mean." https://my.clevelandclinic.org/health/articles/11920-cholesterol-numbers-what-do-they-mean. Accessed on April 11, 2024.

[15] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning.* Springer, 2009.