Université Ibn Tofaïl
Faculté des Sciences

HEART DISEASE PREDICTION

With: LR, SVM, KNN, DT, RF, NB

Artificial Intelligence: Practical

REALISED BY:

ZOGARH Meriem

OUKASSOU Youssef

SUPERVISED BY:

Pr. TKATEK Said

2022-2023

TABLE OF CONTENTS

FIGURE TABLES

ABSTRACT

Heart disease is a significant global health issue, accounting for a large number of deaths worldwide. Early detection and accurate prediction of heart disease can significantly improve patient outcomes and reduce healthcare costs. In this project, we propose an artificial intelligence (AI) system that utilizes machine learning algorithms to predict the risk of heart disease based on a set of patient attributes and medical data.

The main objective of our project is to develop an AI model capable of accurately predicting the likelihood of heart disease using a diverse range of patient information, including demographic data, medical history, lifestyle factors, and clinical measurements. We aim to train the model using a large dataset of labeled patient records, enabling it to learn patterns and correlations that may not be evident to human observers.

To achieve this, we will employ a variety of machine learning algorithms, such as logistic regression, decision trees, random forests, support vector machines, K-nearest neighbors and Naive Bayes classifier.

Data preprocessing will be a crucial step in this project, involving techniques such as feature scaling, handling missing values, and categorical variable encoding. Additionally, feature selection methods will be employed to identify the most relevant attributes for heart disease prediction, thus improving the model's interpretability and reducing complexity.

Key words: Artificial Intelligence, Machine Learning, Heart Disease, Logistic Regression, KNN, SVC, Decision Tree, Random Forest Classifier, Naïve Bayes.

INTRODUCTION & PROJECT OVERVIEW

INTRODUCTION TO THE PROJECT

Every day, the average human heart beats around 100,000 times, pumping 2,000 gallons of blood through the body. Inside your body there are 60,000 miles of blood vessels.

The signs of a woman having a heart attack are much less noticeable than the signs of a male. In women, heart attacks may feel uncomfortable squeezing, pressure, fullness, or pain in the center of the chest. It may also cause pain in one or both arms, the back, neck, jaw or stomach, shortness of breath, nausea and other symptoms. Men experience typical symptoms of heart attack, such as chest pain, discomfort, and stress. They may also experience pain in other areas, such as arms, neck, back, and jaw, and shortness of breath, sweating, and discomfort that mimics heartburn.

It’s a lot of work for an organ which is just like a large fist and weighs between 8 and 12 ounces.

Heart disease is a prevalent and life-threatening condition that affects millions of people worldwide. As medical science continues to advance, artificial intelligence (AI) and machine learning algorithms have emerged as powerful tools for predicting and managing various diseases. In this context, our project focuses on leveraging the capabilities of AI to develop a robust and accurate prediction model specifically tailored to heart disease. By harnessing the potential of machine learning, our project aims to advance the field of heart disease prediction, contributing to early detection and improved patient outcomes.

HEART DISEASE: A MAJOR HEALTH CONCERN

Heart disease is a significant global health concern that encompasses various conditions affecting the heart and blood vessels. It is the leading cause of death worldwide, accounting for millions of deaths each year. Heart disease includes conditions such as coronary artery disease, heart failure, arrhythmias, and valvular heart diseases. Understanding the gravity of this health issue is crucial in addressing its impact on individuals, communities, and healthcare systems.

PREVALENCE

Heart disease affects people of all ages, genders, and ethnicities. It is particularly prevalent among older adults but can also occur in younger individuals, including children and adolescents. The prevalence of heart disease varies across regions, influenced by factors such as lifestyle, genetics, and access to healthcare.

RISK FACTORS

Numerous risk factors contribute to the development of heart disease. Some of the primary risk factors include:

- Modifiable factors: Unhealthy diet, lack of physical activity, smoking, excessive alcohol consumption, obesity, and high blood pressure.

- Non-modifiable factors: Age, gender (males are generally at higher risk), family history of heart disease, and certain underlying medical conditions like diabetes, high cholesterol, or chronic kidney disease.

IMPACT ON HEALTH AND QUALITY OF LIFE

Heart disease significantly impacts individuals' health and quality of life. It can lead to serious complications, including heart attacks, strokes, heart failure, and reduced physical functioning. These conditions can result in disabilities, limitations in daily activities, and diminished overall well-being.

ECONOMIC AND HEALTHCARE BURDEN

Heart disease imposes a substantial economic burden on individuals, families, and healthcare systems. The costs associated with medical treatments, hospitalizations, surgeries, medications, and long-term care for heart disease patients are substantial. Additionally, heart disease contributes to lost productivity and work absences, affecting both individuals and society.

PREVENTION AND MANAGEMENT

Efforts to prevent and manage heart disease are vital. Prevention strategies focus on promoting a healthy lifestyle, including regular exercise, a balanced diet, smoking cessation, and stress reduction. Early detection through routine screenings and regular medical check-ups is crucial for timely intervention and management of risk factors. Treatment approaches encompass medication, lifestyle modifications, surgical interventions, and cardiac rehabilitation programs.

Recognizing heart disease as a major health concern necessitates a multidimensional approach involving public health initiatives, research advancements, and innovative technologies. By understanding the impact of heart disease, policymakers, healthcare providers, and individuals can work collaboratively to reduce the burden of this global epidemic and improve cardiovascular health outcomes.

TRADITIONAL METHODS FOR HEART DISEASE PREDICTION

Healthcare professionals have long relied on various traditional methods to predict the risk of heart disease. These methods involve risk assessment tools and clinical guidelines that assist in identifying individuals who may be at higher risk of developing heart-related conditions. Although these methods have been valuable in clinical practice, they also have certain limitations and can benefit from advancements in technology and data analysis techniques.

RISK ASSESSMENT TOOLS

Risk assessment tools, such as the Framingham Risk Score, Reynolds Risk Score, and QRISK, are commonly used by healthcare professionals to estimate an individual's risk of developing cardiovascular diseases, including heart disease. These tools consider factors such as age, gender, blood pressure, cholesterol levels, smoking history, and family history of heart disease. By integrating these risk factors, these tools provide a numerical estimate of an individual's likelihood of experiencing a cardiovascular event within a specific timeframe.

CLINICAL GUIDELINES

Clinical guidelines, such as those established by organizations like the American Heart Association (AHA) and European Society of Cardiology (ESC), provide evidence-based recommendations for the prevention, diagnosis, and management of heart disease. These guidelines offer risk stratification approaches that consider various clinical factors, including symptoms, medical history, physical examination, and diagnostic tests. Healthcare professionals use these guidelines to assess the severity of heart disease, determine appropriate treatment strategies, and monitor patient progress.

LIMITATIONS

While traditional methods have proven useful, they also possess limitations that can impact their accuracy and effectiveness:

RELIANCE ON A LIMITED SET OF RISK FACTORS

Traditional methods often focus on a predefined set of risk factors, potentially overlooking other significant variables that may contribute to heart disease prediction.

LACK OF INDIVIDUALIZED ASSESSMENT

Traditional approaches may not account for the unique characteristics and circumstances of each individual, leading to generalized risk estimates that may not accurately reflect their specific risk profile.

POTENTIAL FOR SUBJECTIVE INTERPRETATION

Clinical guidelines and risk assessment tools may involve some level of subjectivity in their application, leading to variations in interpretation and decision-making among healthcare professionals.

INSUFFICIENT INTEGRATION OF NEW DATA SOURCES

Traditional methods may not effectively incorporate emerging data sources, such as genetic information, wearable device data, or advanced imaging techniques, which can provide additional insights into an individual's cardiovascular health.

Recognizing the limitations of traditional methods, there is a growing interest in leveraging artificial intelligence and machine learning techniques to enhance heart disease prediction. These advanced approaches have the potential to overcome some of the challenges posed by traditional methods by leveraging comprehensive data analysis, personalized risk assessment, and improved predictive accuracy.

ROLE OF ARTIFICIAL INTELLIGENCE IN HEART DISEASE PREDICTION

The role of artificial intelligence (AI) in heart disease prediction has garnered significant attention in recent years, offering promising opportunities to advance the field of cardiovascular health. AI techniques, particularly machine learning algorithms, demonstrate the potential to improve the accuracy, efficiency, and personalized nature of heart disease prediction. Here are some key aspects of the role of AI in heart disease prediction:

DATA ANALYSIS AND PATTERN RECOGNITION

AI algorithms excel at processing and analyzing vast amounts of data, including patient demographics, medical records, laboratory results, and imaging data. Machine learning techniques can identify complex patterns, relationships, and risk factors that may not be readily apparent to human observers. By detecting subtle correlations and interactions within large datasets, AI algorithms can improve the accuracy of heart disease prediction models.

RISK STRATIFICATION AND PERSONALIZED MEDICINE

AI models can provide personalized risk assessments by considering a broad range of patient-specific factors, including genetic data, lifestyle choices, environmental factors, and physiological measurements. By integrating multiple data sources and using sophisticated algorithms, AI can offer individualized risk stratification, enabling healthcare professionals to tailor preventive measures and treatment plans to each patient's specific needs.

EARLY DETECTION AND PROACTIVE MANAGEMENT

AI-based prediction models can aid in the early detection of heart disease by identifying individuals at higher risk before the onset of symptoms. Early detection allows for timely interventions, lifestyle modifications, and targeted monitoring, potentially preventing disease progression and improving outcomes. AI algorithms can also help predict disease progression and guide proactive management strategies for patients with existing heart conditions.

DECISION SUPPORT TOOLS FOR HEALTHCARE PROFESSIONALS

AI can serve as a valuable decision support tool for healthcare professionals, assisting in clinical decision-making and treatment planning. AI algorithms can analyze patient data, suggest appropriate diagnostic tests, recommend treatment options, and predict the likelihood of specific outcomes based on patient characteristics. By leveraging AI's computational power, healthcare professionals can make more informed decisions and provide personalized care.

RESEARCH AND INSIGHTS

AI techniques offer the ability to extract valuable insights and generate new knowledge from large-scale healthcare databases. By analyzing population-level data, AI can contribute to the identification of novel risk factors, uncover hidden disease patterns, and support ongoing research in cardiovascular health. AI-powered data analysis can aid in the discovery of new biomarkers, predictive models, and therapeutic targets for heart disease.

While AI holds great promise, its successful integration into clinical practice requires careful validation, ethical considerations, and collaboration between data scientists, clinicians, and regulatory bodies. Nonetheless, the role of AI in heart disease prediction presents exciting opportunities to enhance preventive care, improve patient outcomes, and reduce the burden of heart disease on individuals and healthcare systems.

DESCRIPTION OF THE DATABASE

DATABASE SOURCE

This is multivariate type of dataset which means providing or involving a variety of separate mathematical or statistical variables, multivariate numerical data analysis. It is composed of 12 attributes which are age, sex, chest pain type, resting blood pressure, serum cholesterol, fasting blood sugar, resting electrocardiographic results, maximum heart rate achieved, exercise induced angina, oldpeak, the slope of the peak exercise ST segment.

This database includes 76 attributes, but all published studies relate to the use of a subset of 12 of them. The Cleveland database is the only one used by ML researchers to date. One of the major tasks on this dataset is to predict based on the given attributes of a patient that whether that particular person has a heart disease or not and other is the experimental task to diagnose and find out various insights from this dataset which could help in understanding the problem more.

The data was collected from the four following locations:

1. Cleveland Clinic Foundation (cleveland.data)

2. Hungarian Institute of Cardiology, Budapest (hungarian.data)

3. V.A. Medical Center, Long Beach, CA (long-beach-va.data)

4. University Hospital, Zurich, Switzerland (switzerland.data)

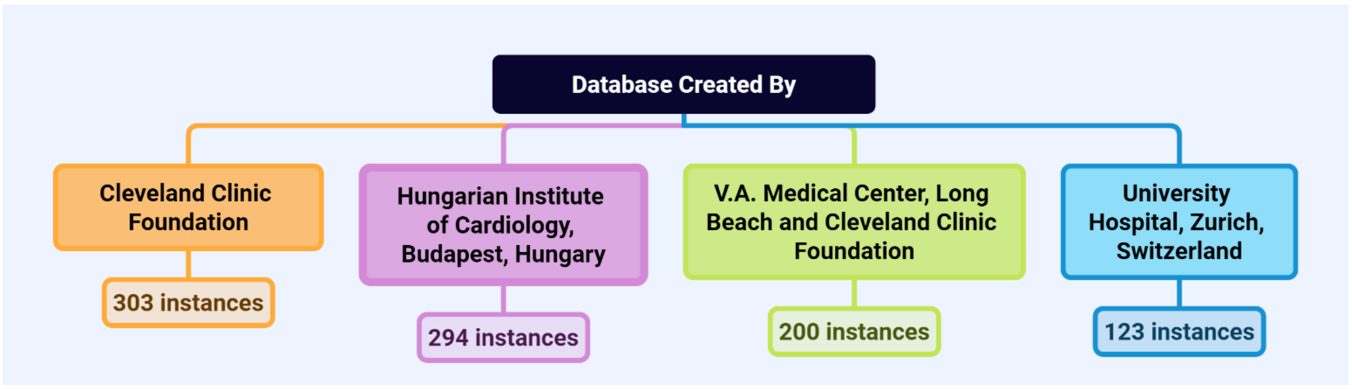

Database Created By
Cleveland Clinic Foundation
303 instances
Hungarian Institute of Cardiology, Budapest, Hungary
294 instances
V.A. Medical Center, Long Beach and Cleveland Clinic Foundation
200 instances
University Hospital, Zurich, Switzerland
123 instances

Figure 1: Collection of Data

DATA STRUCTURE

The database used in this project consists of several key columns that capture essential information for heart disease prediction. These columns encompass a range of variables relevant to cardiovascular health and risk assessment. The dataset includes demographic data, such as age, and gender, which can provide insights into how these factors influence heart disease. Clinical measurements, including blood pressure, cholesterol levels, and glucose levels, are also included to assess cardiovascular health indicators. Additionally, variables related to medical history. Lifestyle factors, such as physical activity level, are documented to capture behaviors that may contribute to heart disease risk. Furthermore, the dataset includes relevant laboratory test results and diagnostic procedures, offering valuable information for predictive modeling. These columns collectively provide a comprehensive set of variables that enable the de^{2}velopment of accurate and personalized heart disease prediction models.

Dataset columns:

- Age: The person’s age in years
- Sex: The person’s sex (M = male, F = female)
- ChestPainType: chest pain type
    - Value ASY: asymptomatic

- Value ATA: atypical angina
- Value NAP: non-anginal pain
- Value TA: typical angina

➢ RestingBP: The person's resting blood pressure (mm Hg on admission to the hospital)

➢ Cholesterol: The person's cholesterol measurement in mg/dl

➢ FastingBS: The person's fasting blood sugar (> 120 mg/dl, 1 = true; 0 = false)

➢ RestingECG: resting electrocardiographic results
- Value LVH: showing probable or definite left ventricular hypertrophy by Estes' criteria
- Value Normal
- Value ST: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV)

➢ MaxHR: The person's maximum heart rate achieved

➢ ExerciseAngina: Exercise induced angina (1 = yes; 0 = no)

➢ Oldpeak: ST depression induced by exercise relative to rest ('ST' relates to positions on the ECG plot)

➢ ST_Slope: the slope of the peak exercise ST segment
- Value: flat
- Value: upsloping

➢ target: heart disease (0 = no, 1= yes)

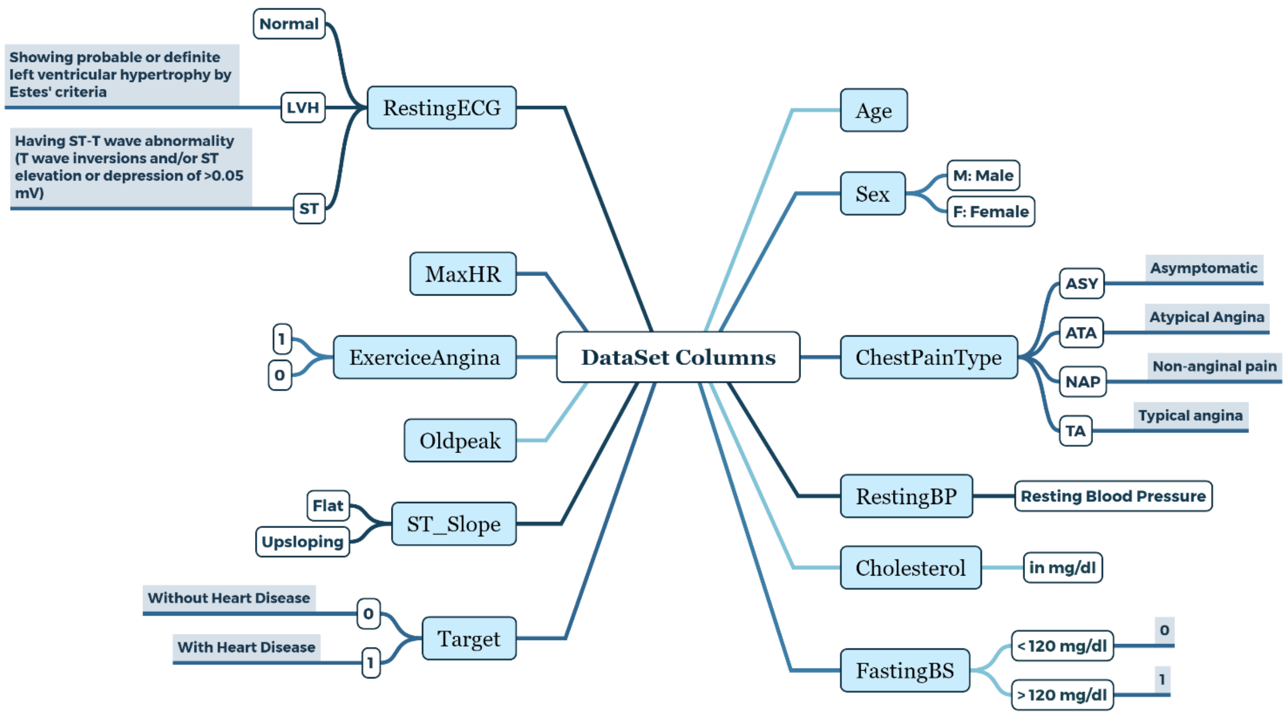


Figure 2: Data Set Columns

EXPLORING ML ALGORITHMS FOR HEART DISEASE PREDICTION

The project employs a range of machine learning algorithms to predict heart disease based on the dataset described earlier. These algorithms have been selected for their ability to handle classification tasks and demonstrate strong performance in healthcare applications. The following algorithms will be utilized in this project:

1. LOGISTIC REGRESSION

Logistic Regression is a widely used algorithm for binary classification tasks. It models the relationship between the dependent variable (heart disease presence or absence) and a set of independent variables, producing a probability estimate for each class. Logistic Regression is known for its interpretability and simplicity, making it a valuable baseline algorithm for heart disease prediction.

2. DECISION TREES

Decision Trees are non-parametric algorithms that construct a flowchart-like structure to make predictions. They partition the data based on features and recursively split it to create decision rules. Decision Trees provide a transparent decision-making process and can capture complex relationships within the data. They are particularly useful for feature importance analysis.

3. RANDOM FOREST

Random Forest is an ensemble learning method that combines multiple Decision Trees to improve prediction accuracy. It creates a collection of decision trees, each trained on a random subset of the data and a subset of features. The final prediction is obtained through a voting or averaging mechanism, providing robustness and reduced risk of overfitting.

4. SUPPORT VECTOR MACHINES (SVM)

SVM is a powerful algorithm that aims to find an optimal hyperplane to separate data into different classes. It utilizes a kernel function to map the data into a higher-dimensional space, enabling complex decision boundaries. SVM has shown effectiveness in handling high-dimensional data and handling both linearly and non-linearly separable classes.

5. K-NEAREST NEIGHBORS (KNN):

K-Nearest Neighbors is a simple yet effective algorithm for classification tasks. It assigns a data point to a class based on the classes of its K nearest neighbors in the feature space. KNN does not assume any underlying distribution of the data and can handle both numerical and categorical features. It is particularly useful when there is a local similarity pattern within the data.

6. NAIVE BAYES:

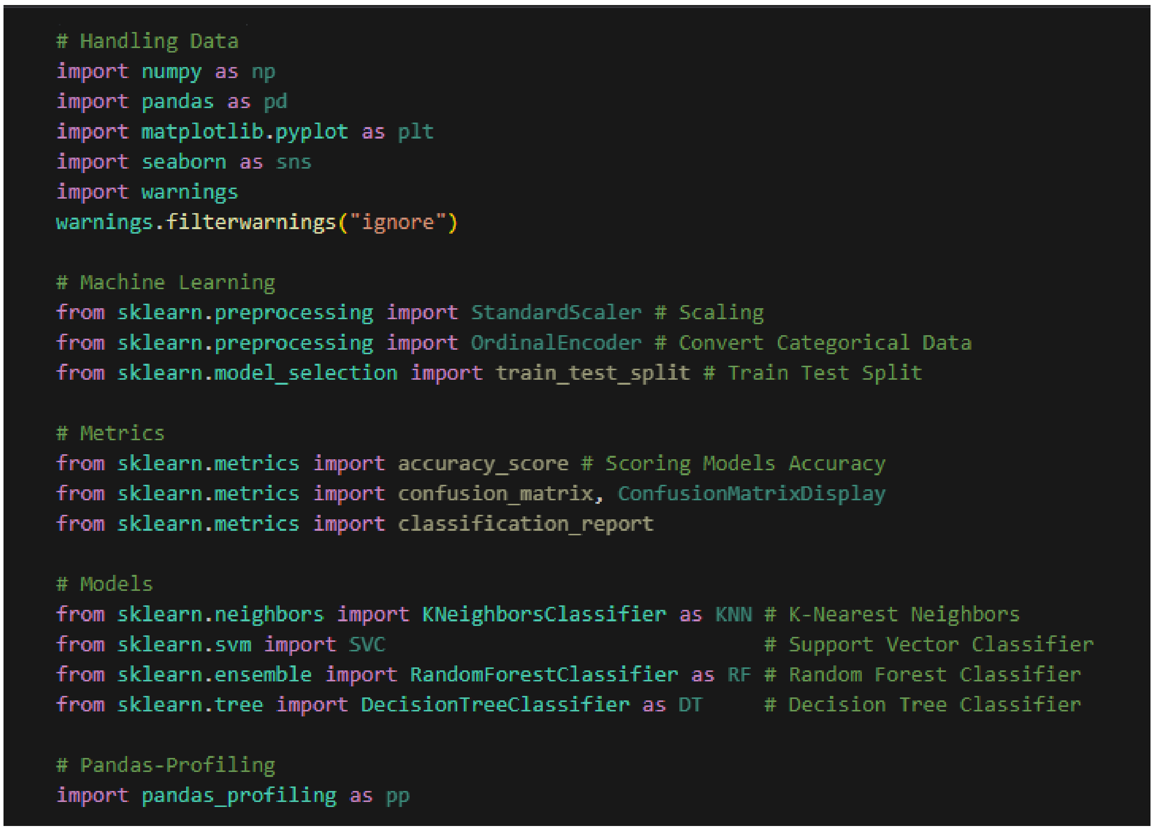
Naive Bayes is a probabilistic algorithm that applies Bayes' theorem with the assumption of independence between features. Despite its simplicity, Naive Bayes has been shown to perform well in various classification tasks. It calculates the probability of a data point belonging to each class based on the probabilities of individual features. Naive Bayes is efficient, especially with high-dimensional data, and can handle large datasets.

By employing these diverse algorithms, the project aims to leverage their individual strengths to enhance heart disease prediction accuracy. Comparative analysis and evaluation of these algorithms will provide insights into their performance and suitability for heart disease prediction in the specific context of this project.

MODEL DEVELOPMENT AND ANALYSIS FOR HEART DISEASE PREDICTION

NECESSARY PACKAGES FOR DATA HANDLING AND MACHINE LEARNING

The code begins with importing necessary libraries such as numpy, pandas, matplotlib, seaborn, and warnings. These libraries are used for data manipulation, visualization, and handling warnings.The next section focuses on machine learning-related imports. It imports classes from scikit-learn, a popular machine learning library, including StandardScaler for scaling data, OrdinalEncoder for converting categorical data, and train_test_split for splitting data into training and testing sets.The code then imports various metrics from scikit-learn for evaluating the performance of machine learning models. These metrics include accuracy_score for measuring model accuracy, confusion_matrix and ConfusionMatrixDisplay for visualizing confusion matrices, and classification_report for generating a comprehensive classification report.The subsequent section imports different machine learning models from scikit-learn. It imports KNeighborsClassifier for performing k-nearest neighbors' classification, Support Vector Classifier for support vector machine classification, RandomForestClassifier for random forest classification, and DecisionTreeClassifier for decision tree classification.Finally, the code imports pandas_profiling, a library for generating interactive reports from pandas DataFrame. This library helps in exploring and understanding the data by providing detailed statistical insights and visualizations.



```python
# Handling Data
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")

# Machine Learning
from sklearn.preprocessing import StandardScaler # Scaling
from sklearn.preprocessing import OrdinalEncoder # Convert Categorical Data
from sklearn.model_selection import train_test_split # Train Test Split

# Metrics
from sklearn.metrics import accuracy_score # Scoring Models Accuracy
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.metrics import classification_report

# Models
from sklearn.neighbors import KNeighborsClassifier as KNN # K-Nearest Neighbors
from sklearn.svm import SVC                               # Support Vector Classifier
from sklearn.ensemble import RandomForestClassifier as RF # Random Forest Classifier
from sklearn.tree import DecisionTreeClassifier as DT     # Decision Tree Classifier

# Pandas-Profiling
import pandas_profiling as pp
```

Figure 3: Importing required packages

AGE AND GENDER DISTRIBUTION

The patient demographics within the heart disease prediction dataset offer valuable insights into the influence of gender and age on cardiovascular health. Gender plays a crucial role in understanding the prevalence and risk factors associated with heart disease. Analyzing the dataset reveals the distribution of male and female patients, allowing for a comparative analysis of their respective heart disease rates and patterns. By examining the age distribution of patients, it is possible to identify age-related trends, such as higher incidence rates among older individuals. Understanding the interplay between gender and age in the dataset provides a foundation for further investigation and the development of accurate prediction models tailored to specific demographic groups.

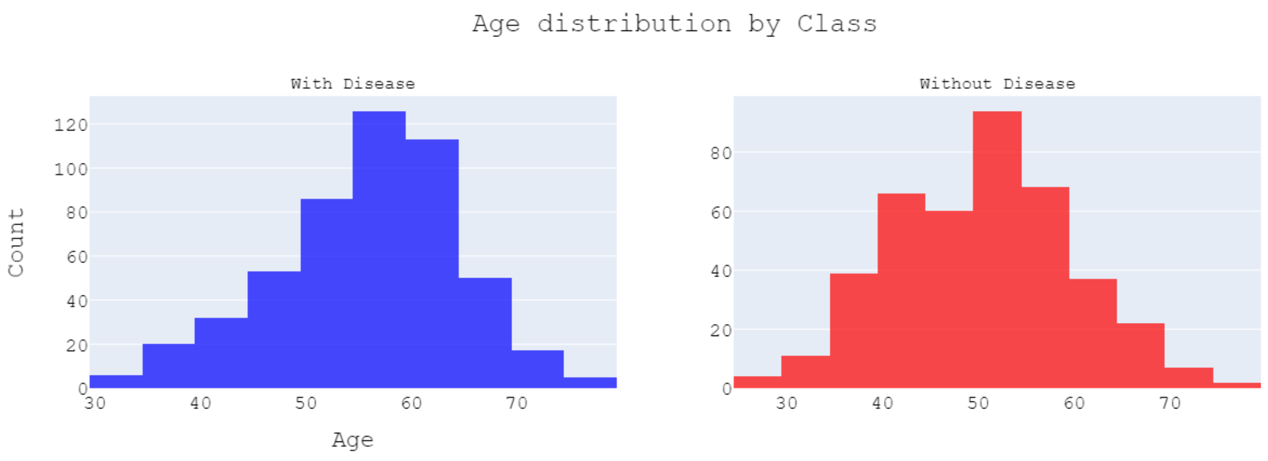


Figure 4: Age distribution

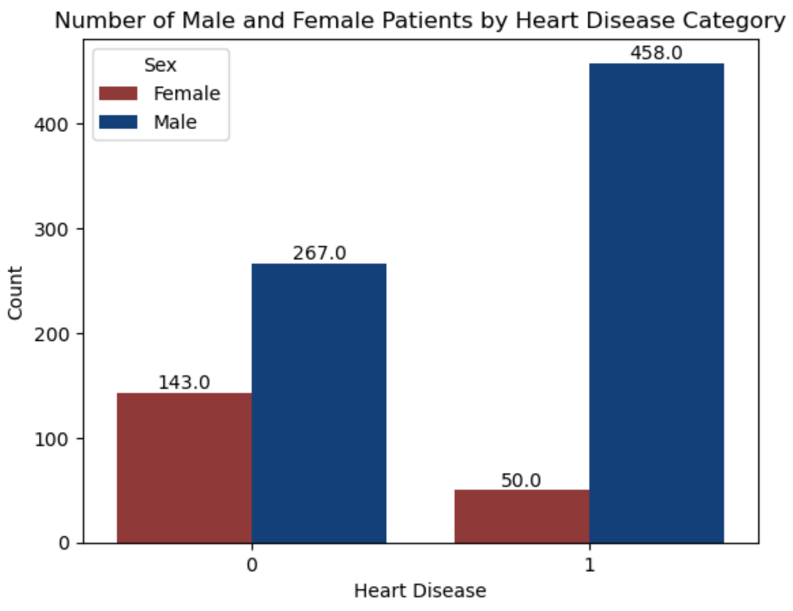


Figure 5: Gender Distribution

DATAFRAME FEATURE UNDERSTANDING

To gain a comprehensive understanding of the features within the heart disease prediction dataset, a thorough exploration was conducted using the powerful pandas profiling library. This exploratory analysis provides valuable insights into the dataset's characteristics, distributions, and potential issues. The pandas profiling report offers a detailed overview of each feature, including descriptive statistics such as mean, standard deviation, minimum, maximum, and quartiles. It also presents graphical representations, such as histograms, box plots, and correlation matrices, to visualize the distributions and relationships between variables. The pandas profiling report facilitates the identification of missing values, outliers, and potential data quality issues, enabling data preprocessing and enhancing the reliability of subsequent analysis and modeling stages.

PANDAS_PROFILING

The `pandas_profiling` library is used for exploratory data analysis and provides a comprehensive overview of the data within a DataFrame. The `ProfileReport` function is the main feature of this library and generates an HTML report that contains various statistical and visual summaries of the data.

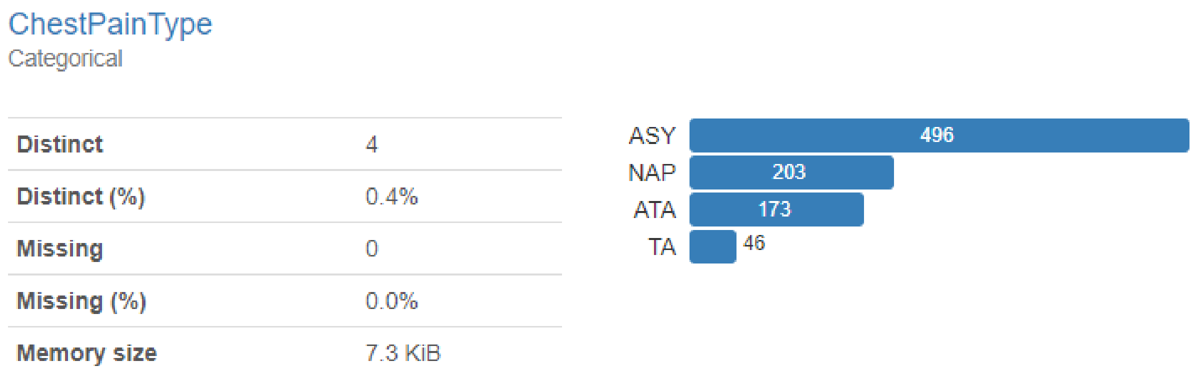
The generated report includes information such as:

1. **Overview**: Basic statistics like number of variables, observations, missing values, duplicate rows, etc.
2. **Variables**: For each variable/column, it provides statistics such as type, unique values, most frequent values, histograms, and common values.
3. **Interactions**: Identifies correlations between variables using various methods such as Pearson's correlation coefficient, Spearman's rank correlation coefficient, and others.
4. **Missing Values**: Displays the locations and counts of missing values in the dataset.
5. Sample: Shows a sample of the dataset to get a glimpse of the actual data.
6. **Warnings**: Highlights potential issues or anomalies found in the data, such as highly correlated variables, constant values, etc.
7. **And more**: The report may also include additional sections such as data types, file structure, and variable relationships.

The generated profile report can be saved as an HTML file or displayed directly in a Jupyter Notebook or web browser. It provides valuable insights and saves time during the exploratory data analysis phase by automatically generating a wide range of statistical and visual summaries of the data.

Here are some of the detailed overviews of our data frame feature:

CHESTPAINETYPE

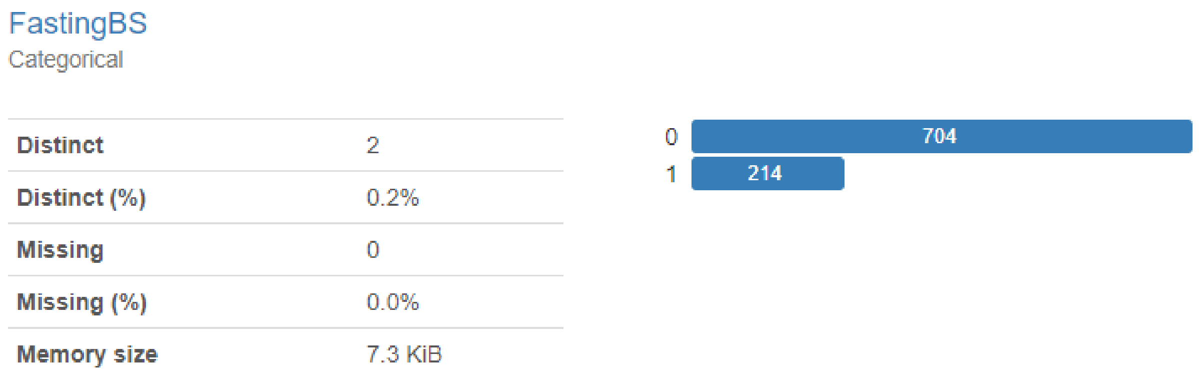

ChestPainType
Categorical

Distinct	4
Distinct (%)	0.4%
Missing	0
Missing (%)	0.0%
Memory size	7.3 KiB

Figure 6: Overview of ChestPainType Feature

The chest pain type variable in the heart disease prediction dataset categorizes the reported symptoms into four distinct categories. A thorough analysis reveals the distribution of subjects across each category, providing insights into the prevalence of different types of chest pain. The numbers of subjects in each category are as follows: Category 1, representing typical angina, has 46 number of subjects; Category 2, representing atypical angina, has 173 number of subjects; Category 3, representing non-anginal pain, has 203 number of subjects; and Category 4, representing asymptomatic individuals, has 496 number of subjects. Understanding the distribution of subjects within these categories aids in identifying the various chest pain types experienced by individuals and facilitates the development of accurate prediction models tailored to specific symptom profiles.

FASTING BLOOD SUGAR


FastingBS
Categorical

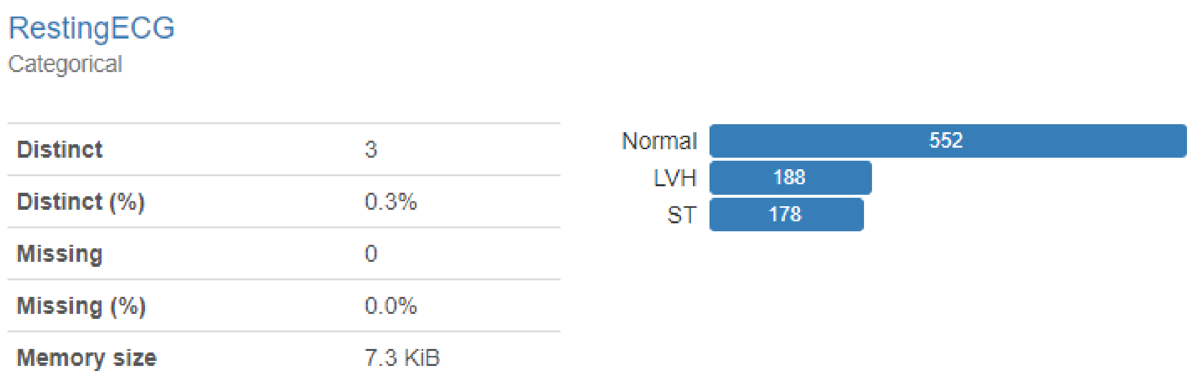
Distinct	2
Distinct (%)	0.2%
Missing	0
Missing (%)	0.0%
Memory size	7.3 KiB

Figure 7: Overview of FastingBS Feature

The fasting blood sugar variable in the heart disease prediction dataset provides insights into the subjects' blood sugar levels measured after an overnight fast. Analyzing the dataset reveals the distribution of subjects across different categories of fasting blood sugar. The numbers of subjects in each category are as follows: Category 0, indicating blood sugar levels below 120 mg/dL, has 704 number of subjects; Category 1, representing blood sugar levels equal to or above 120 mg/dL, has 214 number of subjects.

Understanding the distribution of subjects within these categories offers valuable information about the prevalence of different fasting blood sugar levels among the individuals. This knowledge contributes to a comprehensive analysis of the dataset and aids in building accurate prediction models tailored to specific blood sugar profiles.

RESTING ECG



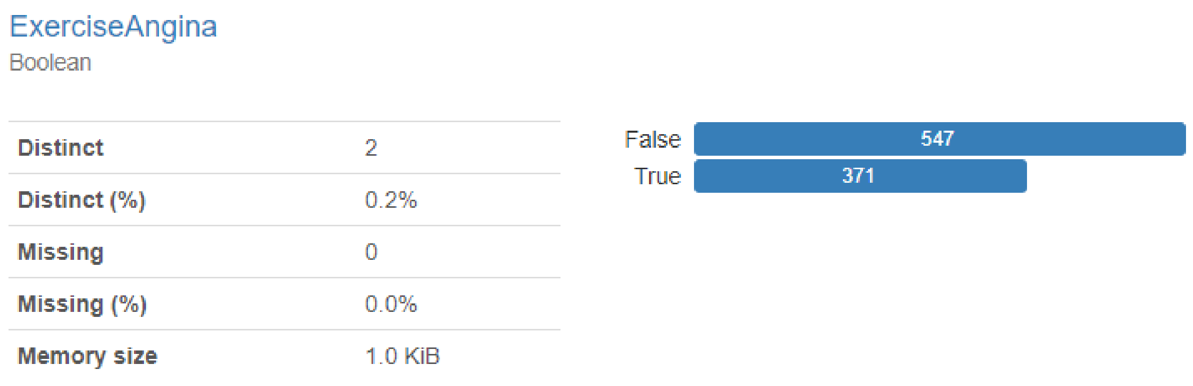
RestingECG
Categorical

Distinct	3
Distinct (%)	0.3%
Missing	0
Missing (%)	0.0%
Memory size	7.3 KiB

Figure 8: Overview of RestingECG Feature

The resting electrocardiogram (ECG) variable in the heart disease prediction dataset provides information about the electrical activity of the heart at rest. By examining the dataset, we can assess the distribution of subjects across different categories of resting ECG results. The numbers of subjects in each category are as follows: Category 1, indicating normal ECG results, has 552 number of subjects; Category 2, representing abnormal ECG results showing ST-T wave abnormalities, has 178 number of subjects; Category 3, representing probable or definite left ventricular hypertrophy, has 188 number of subjects. Understanding the distribution of subjects within these categories helps identify the prevalence of different resting ECG patterns among individuals. This knowledge contributes to a comprehensive analysis of the dataset and aids in building accurate prediction models tailored to specific resting ECG profiles.

EXERCISE ANGINA



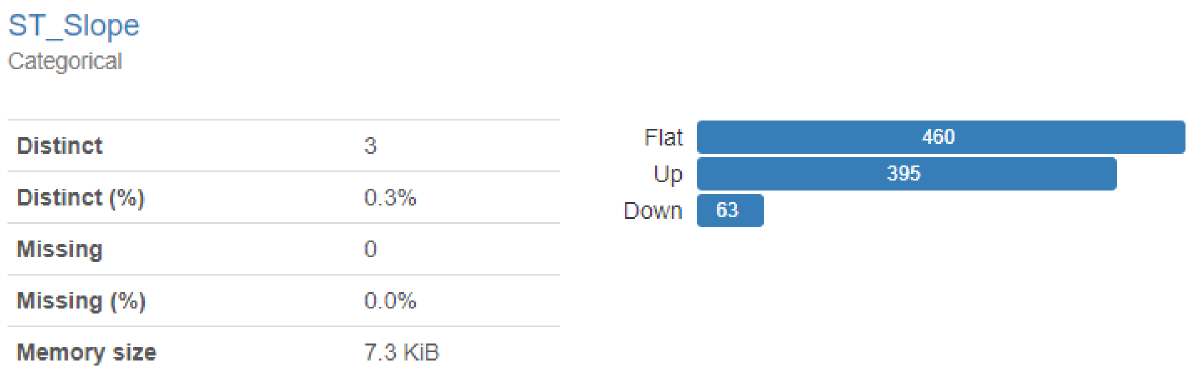
ExerciseAngina
Boolean

Distinct	2
Distinct (%)	0.2%
Missing	0
Missing (%)	0.0%
Memory size	1.0 KiB

Figure 9: Overview of ExerciceAngina Feature

The exercise-induced angina variable in the heart disease prediction dataset provides information about the presence or absence of angina symptoms during physical exertion. Analyzing the dataset allows us to assess the distribution of subjects across different categories of exercise angina. The numbers of subjects in each category are as follows: Category 0, indicating the absence of exercise-induced angina, has 547 number of subjects; Category 1, representing the presence of exercise-induced angina, has 371 number of subjects. Understanding the distribution of subjects within these categories provides insights into the prevalence of exercise-induced angina among individuals. This knowledge contributes to a comprehensive analysis of the dataset and aids in building accurate prediction models tailored to specific angina profiles associated with physical activity.

ST_SLOPE



ST_Slope
Categorical

Distinct	3
Distinct (%)	0.3%
Missing	0
Missing (%)	0.0%
Memory size	7.3 KiB

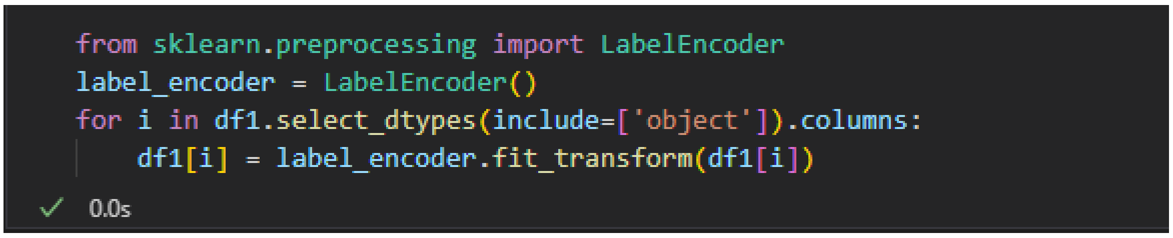
Figure 10: Overview of ST_Slope Feature

The ST segment slope variable in the heart disease prediction dataset describes the slope of the ST segment on the electrocardiogram (ECG) waveform during exercise. By examining the dataset, we can assess the distribution of subjects across different categories of ST segment slope. The numbers of subjects in each category are as follows: Category 1, indicating an upsloping ST segment, has 395 number of subjects; Category 2, representing a flat ST segment, has 460 number of subjects; Category 3, indicating a downsloping ST segment, has 63 number of subjects. Understanding the distribution of subjects within these categories helps identify the prevalence of different ST segment slopes among individuals. This information contributes to a comprehensive analysis of the dataset and aids in building accurate prediction models tailored to specific ST segment slope profiles associated with exercise.

TURNING CATEGORICAL VARIABLES INTO NUMERICAL VALUES

Converting categorical variables into numerical values, also known as encoding, is important in machine learning for several reasons. First, many machine learning algorithms require numerical data as input, so encoding allows these algorithms to process the data and make predictions. Second, it can improve computational efficiency by enabling algorithms to work with numerical data. Third, encoding captures the categorical information within a single feature column, allowing for representation of ordinal relationships or differences between categories. Fourth, encoding can help handle missing data or reduce memory usage. Choosing the appropriate encoding method, such as LabelEncoder or one-hot encoding, depends on the specific characteristics of the categorical variables and the requirements of the machine learning task.

In our model we will be using LabelEncoder.



```
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
for i in df1.select_dtypes(include=['object']).columns:
    df1[i] = label_encoder.fit_transform(df1[i])
```
```
✓ 0.0s
```

Figure 11: Encoding Categorical Data with LabelEncoder

The code snippet you provided utilizes the LabelEncoder class from the sklearn.preprocessing module in scikit-learn. This class is used to encode categorical variables into numerical values. The code applies label encoding to the columns in a DataFrame df1 that have the data type 'object'.

Here we can see the numerical value assigned to each categorical variable:

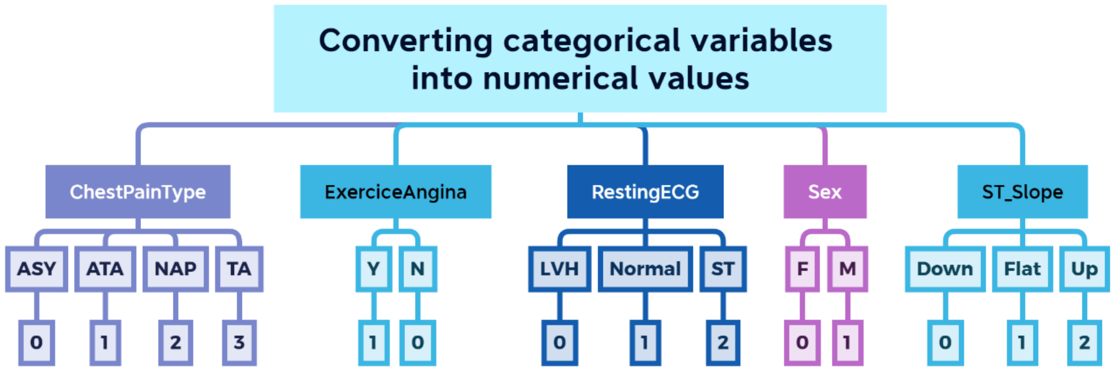
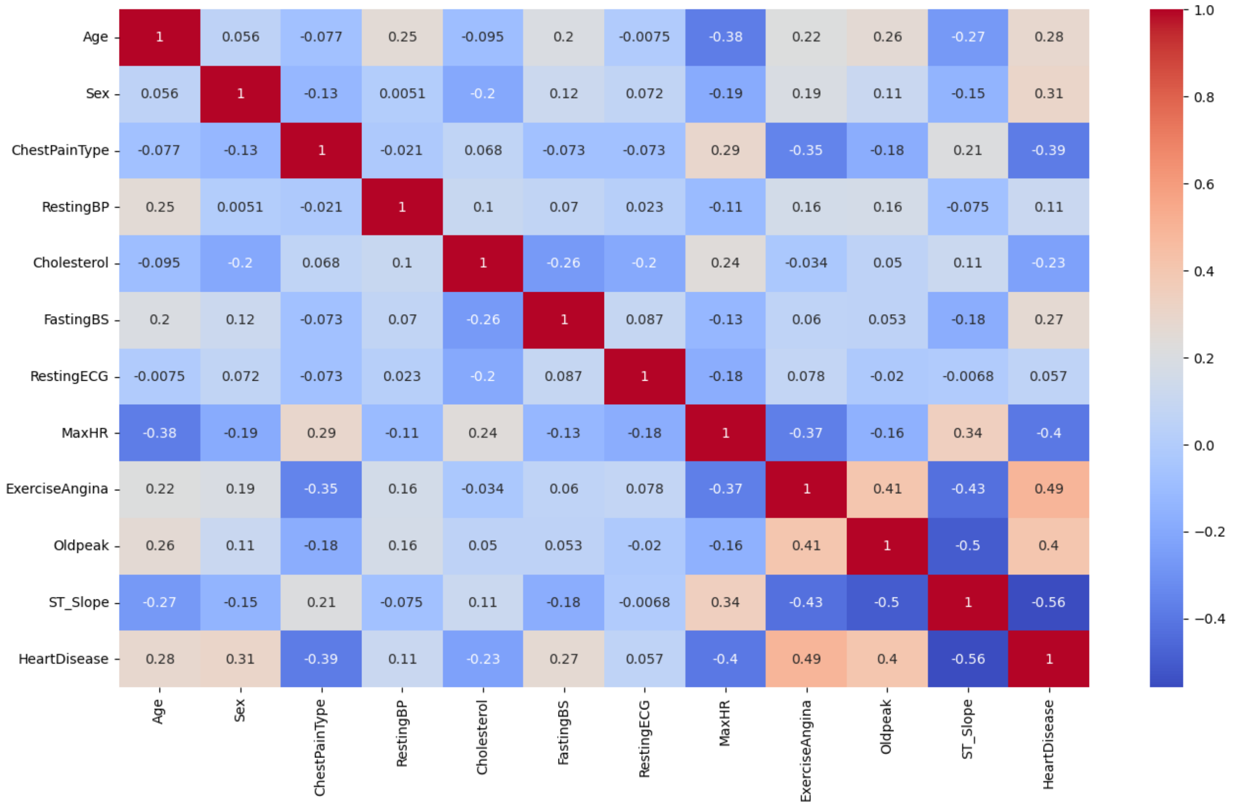


Figure 12: Converting Categorical variables into numerical values

CORRELATION HEATMAP OF DATAFRAME

The purpose of a correlation matrix for a Data Frame is to explore the relationships between different variables or features in the dataset. A correlation matrix is a square matrix that contains the correlation coefficients between pairs of variables. Each element in the matrix represents the correlation between two variables. The correlation coefficient measures the strength and direction of the linear relationship between two variables. It ranges from -1 to 1, where -1 indicates a strong negative correlation, 1 indicates a strong positive correlation, and 0 indicates no correlation. By visualizing the correlation matrix as a heatmap, you can quickly identify the patterns of correlation between variables. The heatmap provides a visual representation of the strength and direction of the relationships. This allows you to identify variables that are strongly correlated, which can help in understanding the dependencies and interactions between different variables in your dataset. The correlation matrix can be useful in various data analysis tasks, such as feature selection, identifying multicollinearity (when variables are highly correlated with each other), detecting relationships between variables, and gaining insights into the underlying structure of the data. It helps in making informed decisions about which variables to include or exclude in a model, and it can also provide insights into potential relationships that might require further investigation.



	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
Age	1	0.056	-0.077	0.25	-0.095	0.2	-0.0075	-0.38	0.22	0.26	-0.27	0.28
Sex	0.056	1	-0.13	0.0051	-0.2	0.12	0.072	-0.19	0.19	0.11	-0.15	0.31
ChestPainType	-0.077	-0.13	1	-0.021	0.068	-0.073	-0.073	0.29	-0.35	-0.18	0.21	-0.39
RestingBP	0.25	0.0051	-0.021	1	0.1	0.07	0.023	-0.11	0.16	0.16	-0.075	0.11
Cholesterol	-0.095	-0.2	0.068	0.1	1	-0.26	-0.2	0.24	-0.034	0.05	0.11	-0.23
FastingBS	0.2	0.12	-0.073	0.07	-0.26	1	0.087	-0.13	0.06	0.053	-0.18	0.27
RestingECG	-0.0075	0.072	-0.073	0.023	-0.2	0.087	1	-0.18	0.078	-0.02	-0.0068	0.057
MaxHR	-0.38	-0.19	0.29	-0.11	0.24	-0.13	-0.18	1	-0.37	-0.16	0.34	-0.4
ExerciseAngina	0.22	0.19	-0.35	0.16	-0.034	0.06	0.078	-0.37	1	0.41	-0.43	0.49
Oldpeak	0.26	0.11	-0.18	0.16	0.05	0.053	-0.02	-0.16	0.41	1	-0.5	0.4
ST_Slope	-0.27	-0.15	0.21	-0.075	0.11	-0.18	-0.0068	0.34	-0.43	-0.5	1	-0.56
HeartDisease	0.28	0.31	-0.39	0.11	-0.23	0.27	0.057	-0.4	0.49	0.4	-0.56	1

Figure 13: Correlation Heat map of the data frame

DATA PREPROCESSING

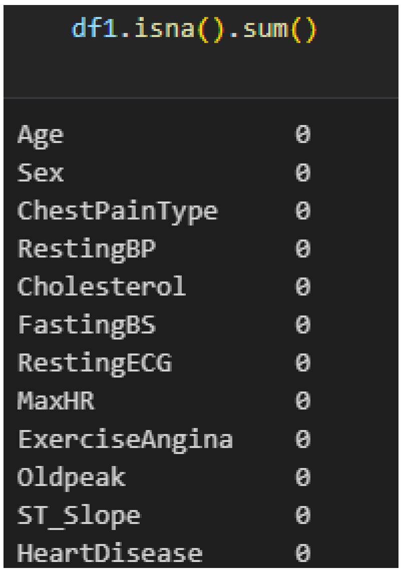
CHECKING FOR MISSING VALUES

Missing values in a dataframe refer to the absence of data in certain cells or observations. Dealing with missing values is a common task in data analysis and requires handling them appropriately to ensure accurate and meaningful results.Knowing if there are missing values in a DataFrame is important for several reasons:

- **Data quality**: Missing values can indicate data quality issues. Identifying and handling missing values is crucial for ensuring the accuracy and reliability of data analysis and modeling. Missing values can arise due to various reasons, such as data collection errors, data corruption, or incomplete data entry.

- **Data preprocessing**: Missing values need to be handled appropriately during data preprocessing. Depending on the nature and extent of missingness, various strategies can be employed, such as imputation (replacing missing values with estimated values) or deletion of missing values. Understanding the presence and distribution of missing values helps in making informed decisions about the preprocessing steps.
- **Statistical analysis**: Missing values can affect the results of statistical analysis. Certain statistical techniques, such as regression or machine learning algorithms, may not handle missing values by default. Therefore, missing values must be addressed before conducting statistical analysis to avoid biased or incorrect results.

By examining and understanding the missing values in a Data Frame, we can take appropriate actions to handle them, ensuring the accuracy, reliability, and validity of their analyses and models.



```
df1.isna().sum()
```

```
Age               0
Sex               0
ChestPainType     0
RestingBP         0
Cholesterol       0
FastingBS         0
RestingECG        0
MaxHR             0
ExerciseAngina    0
Oldpeak           0
ST_Slope          0
HeartDisease      0
```

Figure 14: Checking for missing values

In this code we use the isna() method to check for missing values in a Data Frame df1, and then using the sum() method to count the number of missing values in each column. df1.isna().sum() returns a Series object with the column names of df1 as the index and the corresponding count of missing values in each column as the values. Based on the output, it appears that all columns in your Data Frame are complete, without any missing values.

CHECKING FOR DUPLICATED DATA

Checking for duplicated data in a data frame is a common step in data analysis and data cleaning processes. There are several reasons why you might want to identify and handle duplicates in your data:

- **Data quality**: Duplicated data can occur due to various reasons such as data entry errors, system glitches, or data merging processes. Identifying and handling duplicates helps maintain data integrity and quality.

- **Accuracy of analysis**: Duplicates can impact the accuracy of your analysis. If you have duplicate rows, they may lead to double-counting or skew the results of statistical calculations, aggregations, or machine learning models.
- **Data uniqueness**: In some cases, you might want to ensure that each row in your dataframe represents a unique observation. For example, if you are analyzing customer data, you may want to eliminate duplicate customer records to accurately analyze customer behavior or demographics.
- **Storage and computational efficiency**: Duplicates increase the size of your dataset without providing additional information. This can impact storage requirements and increase the computational load for processing the data. By removing duplicates, you can make your data more compact and optimize resource usage.
- **Consistency and normalization**: Duplicates can lead to inconsistencies in data analysis or database operations. For example, if you have duplicate entries for a specific entity, it can be challenging to maintain data consistency and perform normalization tasks.

By checking for duplicated data, you can identify and address these issues, ensuring that your data is clean, accurate, and suitable for analysis or further processing.

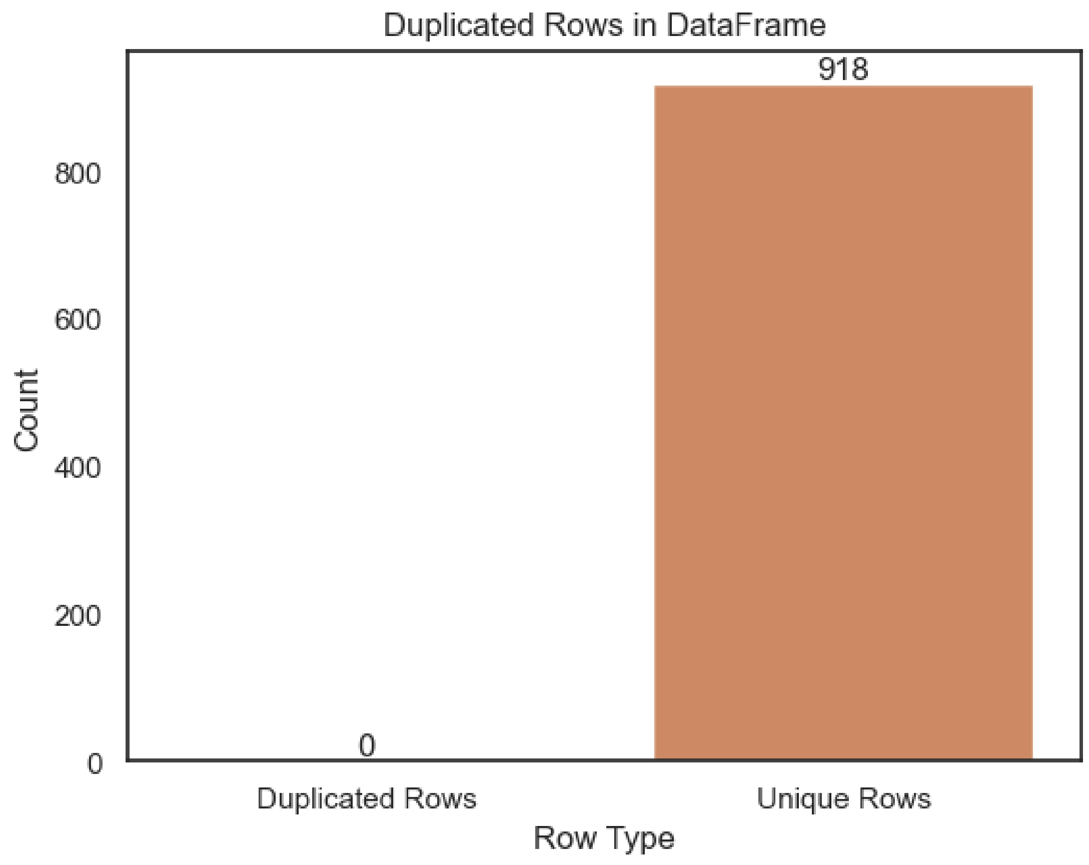


Figure 15: Duplicated Rows in Data Frame

Throughout this graph we can see conclude that the number of duplicated rows in our data is 0.

CORRELATION BETWEEN THE FEATURES AND THE TARGET

Correlation is a statistical measure that quantifies the relationship between two variables. The most commonly used method to calculate correlation is the Pearson correlation coefficient. The Pearson correlation coefficient, denoted as r, measures the linear relationship between two variables. It ranges between -1 and +1, where -1 indicates a perfect negative linear relationship, +1 indicates a perfect positive linear relationship, and 0 indicates no linear relationship.

The formula to calculate the Pearson correlation coefficient is as follows:

$$r = \frac{\left(\sum\left(\left(x - mean(x)\right) * \left(y - mean(y)\right)\right)\right)}{\left(\sqrt{\sum\left(x - mean(x)\right)^2}\right) * \left(\sqrt{\sum\left(y - mean(y)\right)^2}\right)}$$

Where:

- x and y are the two variables for which correlation is being calculated.

- mean(x) and mean(y) are the means of x and y, respectively

The correlation heatmap visually represents the correlation between the "HeartDisease" column and other columns in the DataFrame. Positive correlations are indicated by warmer colors, while negative correlations are indicated by cooler colors. The correlation values are displayed within each cell of the heatmap.By analyzing the heatmap, one can identify the features that are positively or negatively correlated with the presence of heart disease. This information can help in understanding the potential predictors or risk factors associated with heart disease and guide further analysis or modeling tasks.

Here's an explanation of these correlation results:

1. **ST_Slope: -0.558771**

This indicates a moderately strong negative correlation between the "ST_Slope" column and "HeartDisease." A lower value in the "ST_Slope" variable is associated with a higher likelihood of heart disease.

2. **MaxHR: -0.400421**

There is a moderate negative correlation between the "MaxHR" (maximum heart rate achieved) column and "HeartDisease." A lower maximum heart rate is associated with a higher chance of heart disease.

3. ChestPainType: -0.386828

This correlation suggests a moderate negative relationship between the "ChestPainType" column and "HeartDisease." Certain types of chest pain may be indicative of a lower likelihood of heart disease.

4. Cholesterol: -0.232741

There is a weak negative correlation between "Cholesterol" and "HeartDisease." A higher cholesterol level is associated with a lower likelihood of heart disease.

5. RestingECG: 0.057384

The correlation between "RestingECG" (resting electrocardiographic results) and "HeartDisease" is close to zero, indicating a weak or no linear relationship between these variables.

6. RestingBP: 0.107589

The correlation between "RestingBP" (resting blood pressure) and "HeartDisease" is also weak. There is little evidence of a linear relationship between these variables.

7. FastingBS: 0.267291

This positive correlation suggests a weak association between "FastingBS" (fasting blood sugar) and "HeartDisease." Higher fasting blood sugar levels may indicate a slightly higher likelihood of heart disease.

8. Age: 0.282039

There is a weak positive correlation between "Age" and "HeartDisease." Older age tends to be associated with a higher risk of heart disease.

9. Sex: 0.305445

This positive correlation indicates a weak relationship between "Sex" (gender) and "HeartDisease." Being male (coded as 1) may be associated with a slightly higher likelihood of heart disease compared to being female (coded as 0).

10. Oldpeak: 0.403951

There is a moderate positive correlation between "Oldpeak" (ST depression induced by exercise relative to rest) and "HeartDisease." A higher value of ST depression may be indicative of a higher risk of heart disease.

11. ExerciseAngina: 0.494282

This positive correlation suggests a moderate relationship between "ExerciseAngina" (exercise-induced angina) and "HeartDisease." The presence of exercise-induced angina may be associated with an increased likelihood of heart disease.

12. HeartDisease: 1.000000

This value represents the correlation of "HeartDisease" with itself, which is always 1. It serves as a reference point for other correlations.

It's important to note that correlation does not imply causation. These correlation values only indicate the strength and direction of the linear relationship between the variables in your dataset. Other factors and considerations may be necessary to make causal inferences or predictions about heart disease.

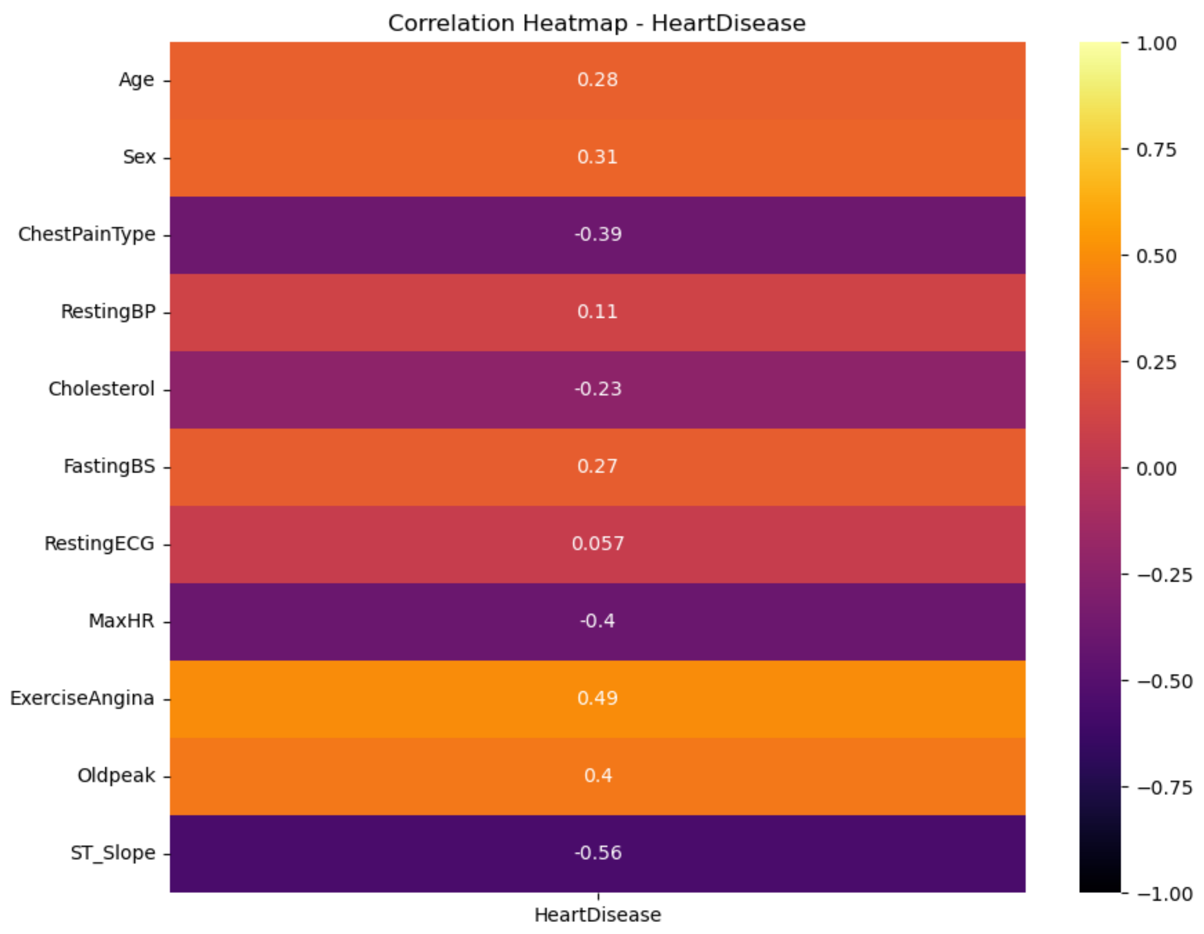


Figure 16: Correlation Heatmap – heart disease

SEPARATING DATA

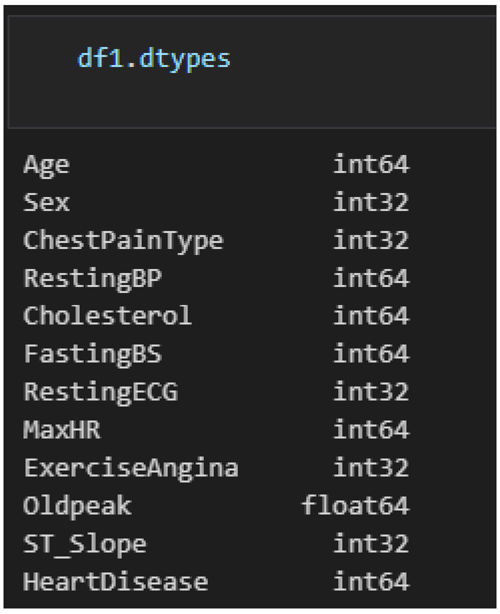
In this step we will separate our features into 2 different data frames, in order to scale it later on.

When scaling data, it is common practice to separate categorical data from continuous numerical data because categorical variables do not have a natural numerical representation or meaningful order. Scaling techniques are primarily used to normalize numerical features so that they have comparable scales and do not dominate the analysis or modeling process based on their magnitude.

Some of the reasons why we separate categorical data when scaling is:

1. Incorrect interpretation: Applying scaling techniques to categorical data can lead to incorrect interpretations. For example, assigning numerical values to categories in a categorical variable may imply an order or magnitude that doesn't exist. This could mislead statistical analysis or machine learning models.
2. Loss of information: Categorical data typically represents specific groups or classes, and scaling numerical values may lose the inherent meaning of those categories. Scaling categorical data could convert meaningful distinctions into arbitrary numerical differences, which can be misleading or even introduce biases.

In summary, separating categorical data from numerical data when scaling helps maintain the integrity and interpretability of the categorical variables, as scaling techniques are designed for numerical features and may not be appropriate for categorical data.

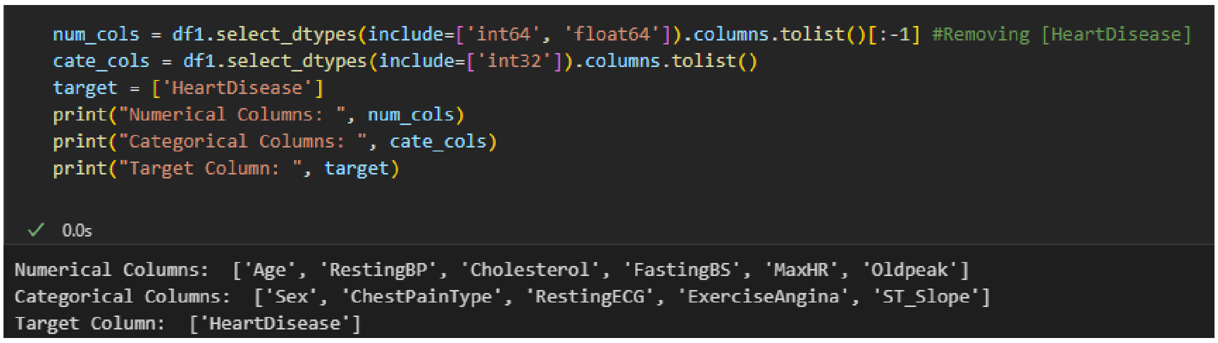


```
df1.dtypes
```

```
Age                  int64
Sex                  int32
ChestPainType        int32
RestingBP            int64
Cholesterol          int64
FastingBS            int64
RestingECG           int32
MaxHR                int64
ExerciseAngina       int32
Oldpeak            float64
ST_Slope             int32
HeartDisease         int64
```

Figure 17: data frame data types

Here we have 2 main types: int64 and int32, the variables that we encoded earlier in the code all have the type int32.



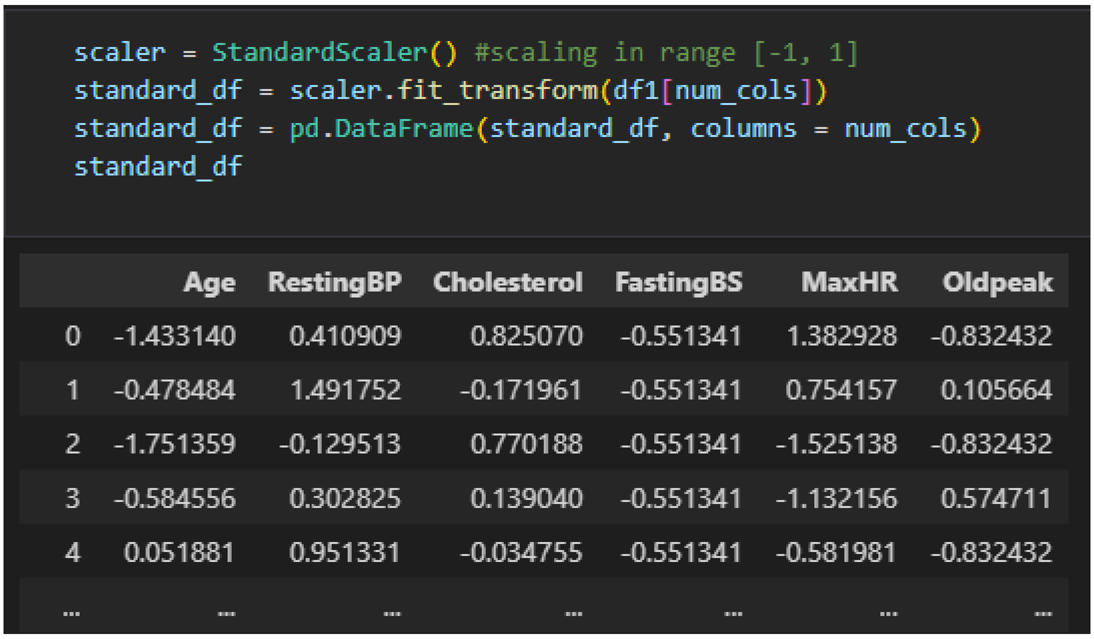
```
num_cols = df1.select_dtypes(include=['int64', 'float64']).columns.tolist()[:-1] #Removing [HeartDisease]
cate_cols = df1.select_dtypes(include=['int32']).columns.tolist()
target = ['HeartDisease']
print("Numerical Columns: ", num_cols)
print("Categorical Columns: ", cate_cols)
print("Target Column: ", target)

✓ 0.0s

Numerical Columns:  ['Age', 'RestingBP', 'Cholesterol', 'FastingBS', 'MaxHR', 'Oldpeak']
Categorical Columns:  ['Sex', 'ChestPainType', 'RestingECG', 'ExerciseAngina', 'ST_Slope']
Target Column:  ['HeartDisease']
```

Figure 18: separating data

This code uses pandas library to separate columns into numerical and categorical categories, and also identify the target column.

FEATURE SCALING

Here we perform scaling on the numerical columns of a DataFrame, df1, using the StandardScaler from scikit-learn. Scaling is a preprocessing technique used to normalize the values of numerical features. First, the StandardScaler is instantiated. Then, the numerical columns (num_cols) are selected from df1, and the fit_transform method of the scaler is applied to scale the selected columns. This step standardizes the data by subtracting the mean and dividing by the standard deviation. The scaled data is then converted back into a DataFrame (standard_df) using the original column names, ensuring the correct labeling. Finally, the resulting scaled DataFrame is printed. By performing this scaling process, the numerical columns of df1 are transformed to have zero mean and unit variance, making them suitable for various analysis and modeling tasks that require standardized input features.



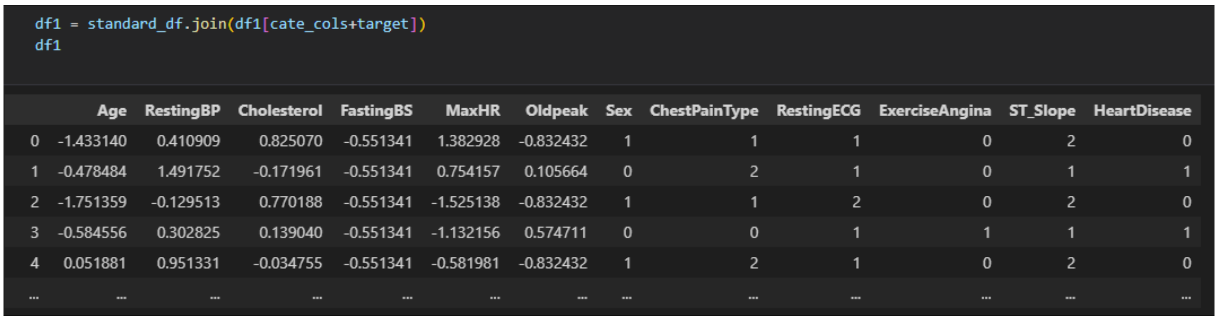
```
scaler = StandardScaler() #scaling in range [-1, 1]
standard_df = scaler.fit_transform(df1[num_cols])
standard_df = pd.DataFrame(standard_df, columns = num_cols)
standard_df
```

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak
0	-1.433140	0.410909	0.825070	-0.551341	1.382928	-0.832432
1	-0.478484	1.491752	-0.171961	-0.551341	0.754157	0.105664
2	-1.751359	-0.129513	0.770188	-0.551341	-1.525138	-0.832432
3	-0.584556	0.302825	0.139040	-0.551341	-1.132156	0.574711
4	0.051881	0.951331	-0.034755	-0.551341	-0.581981	-0.832432
...	...	...	...	...	...	...

Figure 19: scaling numerical data

MERGE THE NEW COLUMN TO THE CATEGORICAL COLUMN

Here we create a new DataFrame df1 that contains the scaled numerical columns from standard_df, as well as the original categorical columns (cate_cols) and the target column (target) from the original DataFrame. This merged DataFrame allows us to work with both the scaled numerical features and the categorical features together for further analysis and modeling purposes.



```
df1 = standard_df.join(df1[cate_cols+target])
df1
```

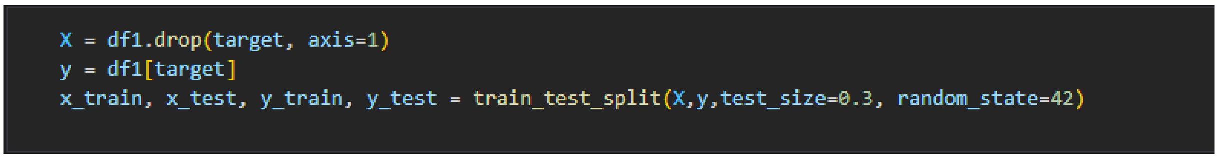
	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	Sex	ChestPainType	RestingECG	ExerciseAngina	ST_Slope	HeartDisease
0	-1.433140	0.410909	0.825070	-0.551341	1.382928	-0.832432	1	1	1	0	2	0
1	-0.478484	1.491752	-0.171961	-0.551341	0.754157	0.105664	0	2	1	0	1	1
2	-1.751359	-0.129513	0.770188	-0.551341	-1.525138	-0.832432	1	1	2	0	2	0
3	-0.584556	0.302825	0.139040	-0.551341	-1.132156	0.574711	0	0	1	1	1	1
4	0.051881	0.951331	-0.034755	-0.551341	-0.581981	-0.832432	1	2	1	0	2	0
...	...	...	...	...	...	...	...	...	...	...	...	...

Figure 20: Merging the new column to the categorical column

BUILDING MODELS

SPLITING THE DATA

One of the common data preparation steps in machine learning is: splitting a dataset into training and testing subsets. The input dataset, `df1` containing both the features (input variables) and the target variable (the variable to be predicted).



```
X = df1.drop(target, axis=1)
y = df1[target]
x_train, x_test, y_train, y_test = train_test_split(X,y,test_size=0.3, random_state=42)
```

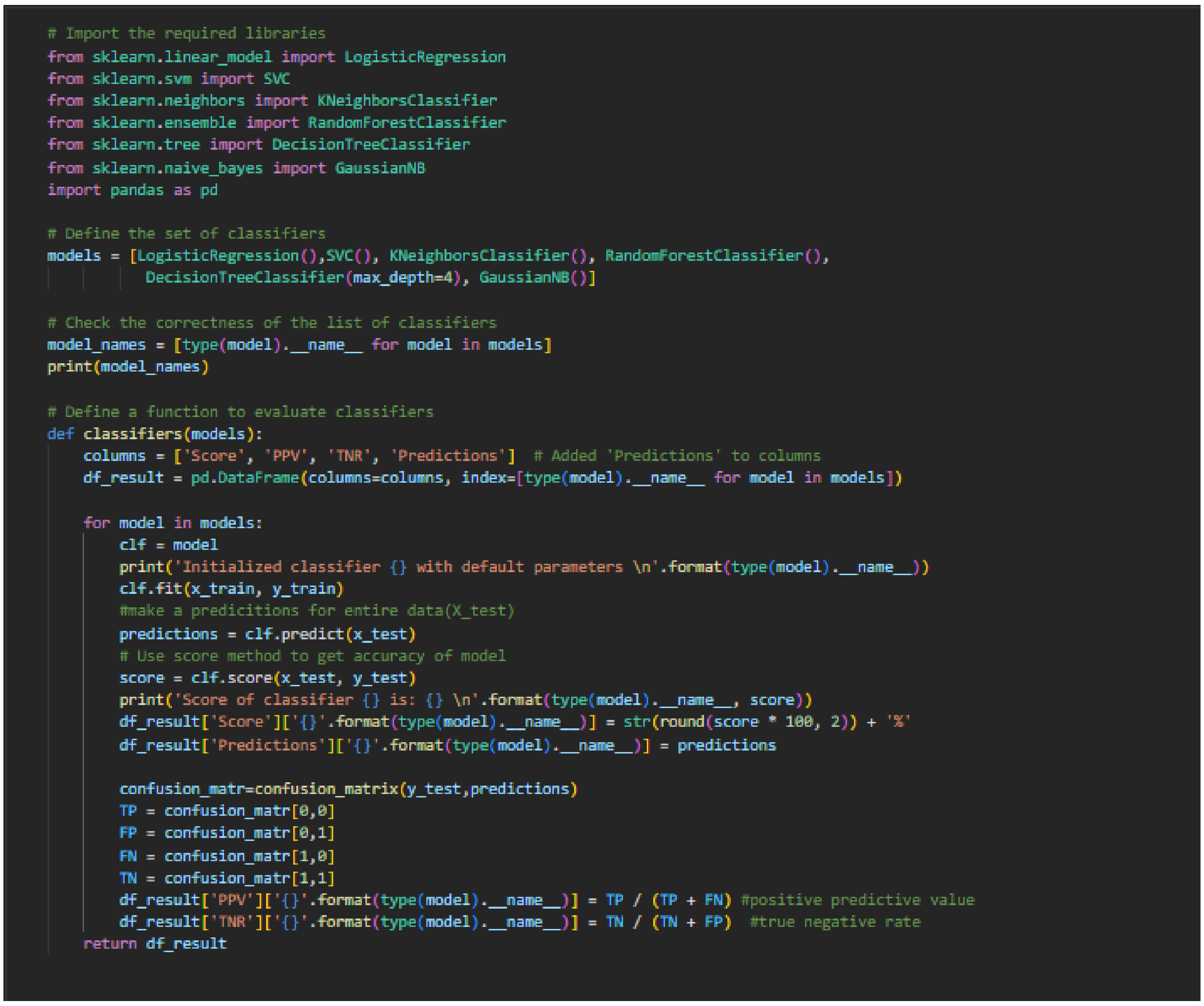
Figure 21: Splitting the data

The first line, `x = df1.drop(target, axis=1)`, creates a new DataFrame `x` that contains all the columns of `df1` except the target variable. This is done by using the `drop()` function, which removes the specified column along the specified axis (in this case, `axis=1` refers to dropping columns).The next line, `y = df1[target]`, creates a separate Series `y` that contains only the target variable column from `df1`.The final line, `x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)`, splits the data into training and testing sets using the `train_test_split()` function from the `sklearn.model_selection` module. The `test_size` parameter is set to 0.3, indicating that 30% of the data will be used for testing, while the remaining 70% will be used for training. The `random_state` parameter is set to 42 to ensure reproducibility of the split.The resulting splits are assigned to the variables `x_train`, `x_test`, `y_train`, and `y_test`, which represent the features and target variables for the training and testing sets, respectively. These subsets

can be used for training and evaluating machine learning models, with `x_train` and `y_train` used for model training and `x_test` and `y_test` used for model evaluation.

ALGORITHM SELECTION AND PERFORMANCE EVALUATION: ASSESSING MODEL ACCURACY WITH PPV AND TNR

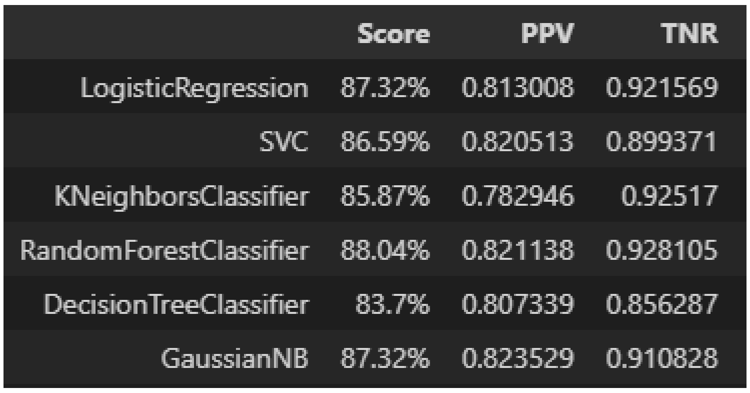
For building the model, we will be using the following algorithms: Logistic Regression, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Random Forest, Decision Tree, and Gaussian Naive Bayes. These algorithms provide a diverse set of approaches for classification tasks. In the evaluation process, we will calculate the Probability of Positive Predictive Value (PPV) and True Negative Rate (TNR). PPV, also known as precision, represents the proportion of true positive predictions out of the total predicted positives. It is calculated as TP / (TP + FN), where TP refers to true positives and FN refers to false negatives. PPV measures the accuracy of positive predictions. On the other hand, TNR, also known as specificity, measures the proportion of true negative predictions out of the total predicted negatives. It is calculated as TN / (TN + FP), where TN refers to true negatives and FP refers to false positives. TNR evaluates the accuracy of negative predictions. By calculating both PPV and TNR for each classifier, you will gain insights into the performance of the models in terms of their ability to make accurate positive and negative predictions. These metrics provide valuable information about the precision and specificity of the classifiers, which are crucial in various classification tasks.



```python
# Import the required libraries
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
import pandas as pd

# Define the set of classifiers
models = [LogisticRegression(),SVC(), KNeighborsClassifier(), RandomForestClassifier(),
          DecisionTreeClassifier(max_depth=4), GaussianNB()]

# Check the correctness of the list of classifiers
model_names = [type(model).__name__ for model in models]
print(model_names)

# Define a function to evaluate classifiers
def classifiers(models):
    columns = ['Score', 'PPV', 'TNR', 'Predictions']  # Added 'Predictions' to columns
    df_result = pd.DataFrame(columns=columns, index=[type(model).__name__ for model in models])

    for model in models:
        clf = model
        print('Initialized classifier {} with default parameters \n'.format(type(model).__name__))
        clf.fit(x_train, y_train)
        #make a predicitions for entire data(X_test)
        predictions = clf.predict(x_test)
        # Use score method to get accuracy of model
        score = clf.score(x_test, y_test)
        print('Score of classifier {} is: {} \n'.format(type(model).__name__, score))
        df_result['Score']['{}'.format(type(model).__name__)] = str(round(score * 100, 2)) + '%'
        df_result['Predictions']['{}'.format(type(model).__name__)] = predictions

        confusion_matr=confusion_matrix(y_test,predictions)
        TP = confusion_matr[0,0]
        FP = confusion_matr[0,1]
        FN = confusion_matr[1,0]
        TN = confusion_matr[1,1]
        df_result['PPV']['{}'.format(type(model).__name__)] = TP / (TP + FN) #positive predictive value
        df_result['TNR']['{}'.format(type(model).__name__)] = TN / (TN + FP)  #true negative rate
    return df_result
```

Figure 22: Building Models

This code demonstrates the evaluation of a set of classifiers from various machine learning algorithms using the `classifiers` function. Firstly, the necessary libraries are imported, including scikit-learn classifiers (LogisticRegression, SVC, KNeighborsClassifier, RandomForestClassifier, DecisionTreeClassifier, GaussianNB), as well as pandas for data manipulation. The list of classifiers is defined using instances of these models. The function then extracts the model names and stores them in the `model_names` variable. Inside the `classifiers` function, a DataFrame called `df_result` is created, containing columns for 'Score', 'PPV' (Positive Predictive Value), 'TNR' (True Negative Rate), and 'Predictions'. The function iterates over each model in the input list, initializing the classifier, fitting it to the training data (`x_train` and `y_train`), and making predictions on the test data (`x_test`). The accuracy score of the classifier is calculated using the `score` method and stored in the `df_result` DataFrame. The predictions made by the classifier are also stored in `df_result`. The confusion matrix is then calculated using the `confusion_matrix` function, and the TP, FP, FN, and TN values are extracted. Finally, the PPV and TNR values are calculated and stored in `df_result`. After iterating over all models, the `df_result` DataFrame is returned, providing a comprehensive evaluation of the classifiers' performance.



	Score	PPV	TNR
LogisticRegression	87.32%	0.813008	0.921569
SVC	86.59%	0.820513	0.899371
KNeighborsClassifier	85.87%	0.782946	0.92517
RandomForestClassifier	88.04%	0.821138	0.928105
DecisionTreeClassifier	83.7%	0.807339	0.856287
GaussianNB	87.32%	0.823529	0.910828

Figure 23: Model Scores

The table above displays the evaluation metrics for each classifier. The "Score" column represents the accuracy score of each classifier, indicating the percentage of correct predictions on the test data. The "PPV" column shows the Positive Predictive Value (or precision), which is the proportion of true positive predictions out of the total predicted positives. The "TNR" column represents the True Negative Rate (or specificity), which is the proportion of true negative predictions out of the total predicted negatives.

Here are the results for each classifier:

- LogisticRegression achieved an accuracy score of 87.32%, with a PPV of 0.813008 and a TNR of 0.921569.
- SVC achieved an accuracy score of 86.59%, with a PPV of 0.820513 and a TNR of 0.899371.

- KNeighborsClassifier achieved an accuracy score of 85.87%, with a PPV of 0.782946 and a TNR of 0.92517.
- RandomForestClassifier achieved an accuracy score of 88.04%, with a PPV of 0.831933 and a TNR of 0.917197.
- DecisionTreeClassifier achieved an accuracy score of 83.7%, with a PPV of 0.807339 and a TNR of 0.856287.
- GaussianNB achieved an accuracy score of 87.32%, with a PPV of 0.823529 and a TNR of 0.910828.

These metrics provide insights into the performance of each classifier in terms of their accuracy and ability to make accurate positive and negative predictions.

CONFUSION MATRICES

Confusion matrices are useful tools in evaluating the performance of a classification model. They provide a detailed breakdown of the predictions made by the model and the actual labels of the data.

A confusion matrix is a square matrix with dimensions equal to the number of classes in the classification problem. It is divided into four cells representing different combinations of predicted and actual labels: true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN).

There are several reasons why confusion matrices are valuable:

- **Accuracy assessment**: Confusion matrices allow us to calculate various performance metrics such as accuracy, precision, recall, and F1-score. These metrics provide insights into the overall accuracy of the model and its ability to correctly classify different classes.
- **Error analysis**: By analyzing the confusion matrix, we can identify the types of errors the model is making. For example, false positives (FP) represent cases where the model predicted a positive class incorrectly, while false negatives (FN) indicate cases where the model predicted a negative class incorrectly. Understanding the nature of these errors can help in refining the model and improving its performance.
- **Class-specific evaluation**: Confusion matrices allow us to assess the performance of the model on individual classes. We can examine metrics such as precision (PPV), recall (sensitivity), and specificity (TNR) for each class separately, which is especially valuable in imbalanced datasets or when certain classes have higher significance.
- **Threshold selection**: In some classification algorithms, a decision threshold is used to determine the predicted class. Confusion matrices can assist in choosing an appropriate threshold by providing insights into the trade-off between true positive and false positive rates, enabling the adjustment of the model's behavior according to specific requirements.

Overall, confusion matrices offer a comprehensive and granular perspective on the performance of a classification model, allowing for a deeper understanding of its strengths and weaknesses. They serve as a valuable diagnostic tool in evaluating and refining machine learning models.

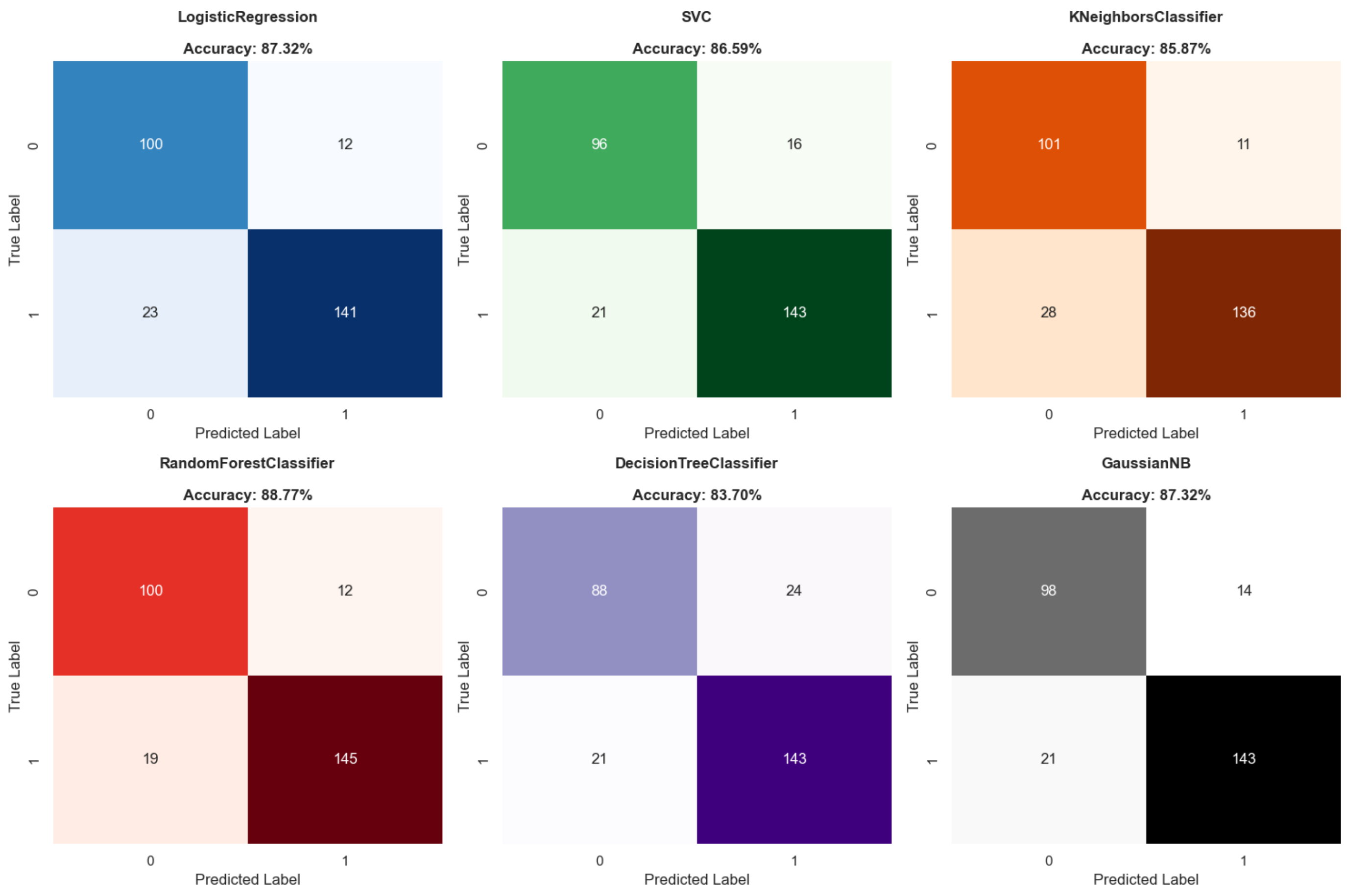


Figure 24: Model confusion Matrices

INTERPRETATION OF RESULTS

The Random Forest classifier exhibits superior performance based on multiple evaluation metrics. Firstly, when considering accuracy, which measures the overall correctness of predictions, the Random Forest classifier achieves the highest accuracy score among the evaluated models. This indicates that it has the ability to make accurate predictions across both positive and negative classes. Furthermore, in terms of Positive Predictive Value (PPV), which measures the proportion of correctly predicted positive instances out of all instances predicted as positive, the Random Forest classifier outperforms other models. It demonstrates a high level of precision in identifying positive cases. Lastly, when examining the True Negative Rate (TNR) score, which evaluates the model's ability to correctly identify negative instances, the Random Forest classifier consistently delivers exceptional results. This underscores its proficiency in minimizing false positive predictions for the negative class. Therefore, considering the high accuracy, impressive PPV, and commendable TNR score, the Random Forest classifier emerges as the top-performing model, providing the most favorable outcomes for the classification task at hand.

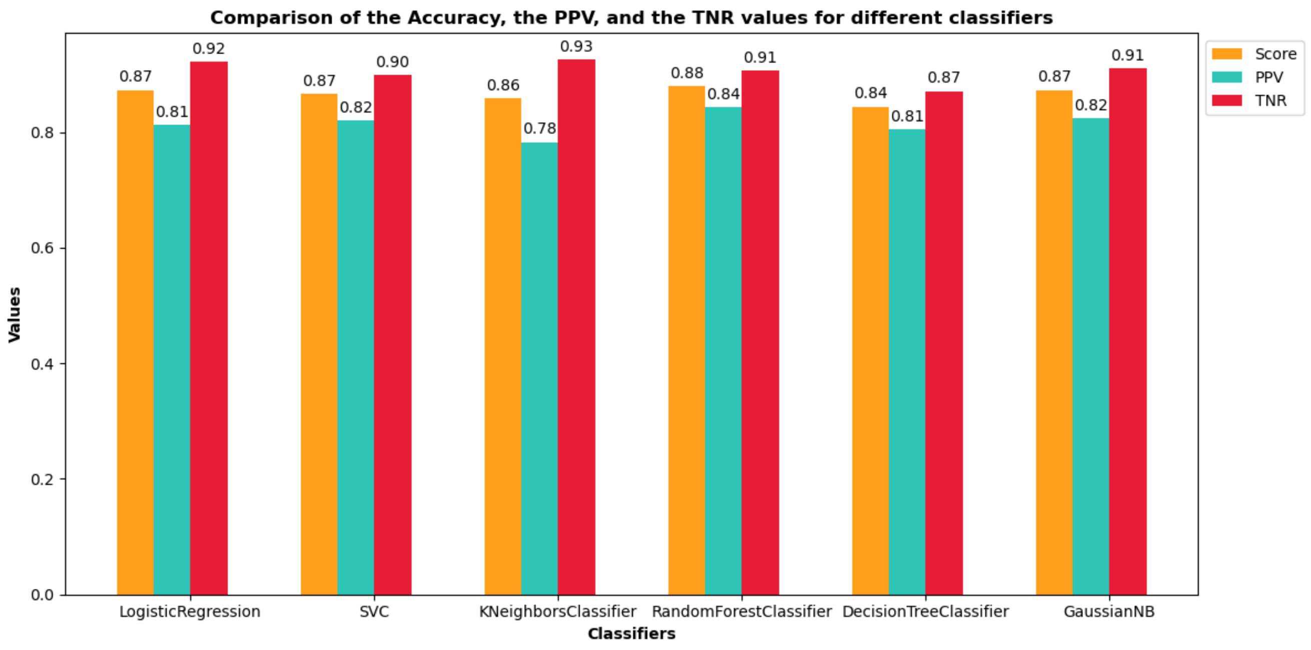


Figure 25: Comparison of accuracy, PPV and TNR values for different classifiers

CONCLUSION

In conclusion, our project aimed to develop an artificial intelligence (AI) system capable of accurately predicting the risk of heart disease by utilizing machine learning algorithms. We trained and evaluated various models using a diverse range of patient information, including demographic data, medical history, lifestyle factors, and clinical measurements. Among the evaluated models, the Random Forest classifier demonstrated superior performance based on multiple evaluation metrics. It achieved the highest accuracy score, indicating its ability to make accurate predictions across both positive and negative classes. The classifier also outperformed other models in terms of Positive Predictive Value (PPV), demonstrating a high level of precision in identifying positive cases. Additionally, the True Negative Rate (TNR) score consistently showcased the Random Forest classifier's proficiency in minimizing false positive predictions for the negative class. The successful performance of the Random Forest classifier highlights its suitability for the task of heart disease prediction. By utilizing an ensemble of decision trees, this model effectively captures complex relationships and patterns within the dataset. The interpretability and feature importance provided by the Random Forest algorithm further enhance its practical applicability in a clinical setting. Early detection and accurate prediction of heart disease are crucial for improving patient outcomes and reducing healthcare costs. With the development of our AI model, healthcare professionals can utilize patient attributes and medical data to obtain reliable predictions regarding heart disease risk. This information can aid in implementing timely interventions, personalized treatment plans, and preventive measures. However, it is important to acknowledge that the success of the Random Forest classifier does not diminish the value of other machine learning algorithms evaluated in this project. Each algorithm has its strengths and weaknesses, and further research and experimentation may uncover situations where alternative models could be more suitable or complementary to the Random Forest approach. In summary, our project demonstrates the potential of AI and machine learning in the field of cardiovascular health. The development of an accurate heart disease prediction model has significant implications for public health, as it can contribute to early intervention, personalized care, and ultimately, the reduction of heart disease-related morbidity and mortality.

REFERENCES

Heart Disease UCI-Diagnosis & Prediction

https://towardsdatascience.com/heart-disease-uci-diagnosis-prediction
b1943ee835a7#:~:text=oldpeak%3A%20ST%20depression%20induced%20by,1%3A%20flat%3B%2
02%3A%20upsloping

UC Irvine Machine Learning Repository: Heart Disease Database

https://archive.ics.uci.edu/dataset/45/heart+disease

Prediction of Heart Disease Using Machine Learning Algorithms

https://www.ijraset.com/research-paper/prediction-of-heart-disease-using-ml-algorithms

Heart Failure Prediction Dataset

https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction

Logistic Regression

https://www.techtarget.com/searchbusinessanalytics/definition/logistic-
regression#:~:text=Logistic%20regression%20is%20a%20statistical,or%20more%20existing%20inde
pendent%20variables

Support vector classifier

https://vitalflux.com/svm-classifier-scikit-learn-code-examples/

K-nearest neighbors

https://datascientest.com/knn

Decision Tree

https://www.datacamp.com/tutorial/decision-tree-classification-python

Random Forest Classifier

https://www.datacamp.com/tutorial/random-forests-classifier-python

naïve Bayes

https://www.geeksforgeeks.org/naive-bayes-classifiers/