**Table of Contents**

[**1.** **Information about the dataset** 3](#_Toc174742006)

[**1.1 Dataset Description** 4](#_Toc174742007)

[**1.1.1 Number of instances** 4](#_Toc174742008)

[**1.1.2 Number of features** 4](#_Toc174742009)

[**1.1.3 Number of instances from each class** 5](#_Toc174742010)

[**1.1.4 Dataset View** 5](#_Toc174742011)

[**2.** **Proper split of dataset** 6](#_Toc174742012)

[**3.** **EDA on data set** 6](#_Toc174742013)

[**4.** **Building the Classifiers** 13](#_Toc174742014)

[**5.** **Training and Testing the Classifiers** 14](#_Toc174742015)

[**6.** **Explaining and comparing the results** 16](#_Toc174742016)

[**7.** **Conclusion** 17](#_Toc174742017)

## **Information about the dataset**

A screenshot of a computer program

Description automatically generated**Download the dataset and extract the following information from the dataset:**

In this analysis, we will be utilising the heart\_disease\_uci.csv dataset. This dataset includes various medical features related to heart disease, along with a target variable indicating the presence or absence of heart disease.

First, we will import the necessary libraries and models for data analysis, visualization, and machine learning. It sets up an environment where you can manipulate data, build and evaluate machine learning models, while also visualizing the results for better analysis.

A screenshot of a computer

Description automatically generated Secondly, we import the heart\_disease\_uci.csv dataset related and load it into a Pandas DataFrame called df. The dataset contains several features (columns/variables) related to the health and medical history of individuals. This dataset is used in medical research and machine learning projects to predict the presence or absence of heart disease based on these features.

A screenshot of a computer

Description automatically generatedIn order to extract key information of the dataset (eg: features, classes etc), we will be running the following code, this enables us to check the structure and basic details of the DataFrame: **df** using the df.info() and df.shape methods.

## **1.1 Dataset Description**

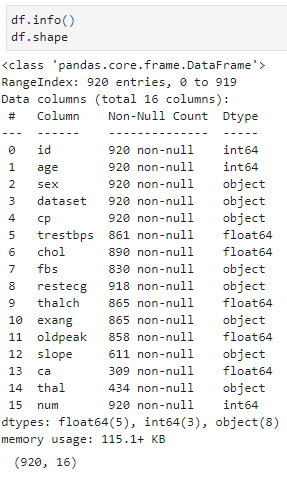
Here are the details extracted from the dataset:

**1.1.1 Number of instances:** The dataset contains 920 instances​.  
The below code calculates and prints the number of instances (or rows) in the DataFrame: **df**

A screenshot of a computer code

Description automatically generated

**A white rectangular object with text

Description automatically generated****1.1.2 Number of features:** The dataset includes 15 features​.  
The below code calculates and prints the number of features in the DataFrame: **df**, excluding the target column.

|  |  |
| --- | --- |
| **Feature Name** | **Description** |
| id | Identification Number |
| Age | The individual's **Age in Years** |
| Sex | The individual's **Sex (Gender)**  Male = 1  Female = 0 |
| CP | The type of **Chest Pain** experienced:  Value 1: Typical angina  Value 2: Atypical angina  Value 3: Non-anginal pain  Value 4: Asymptomatic |
| Trestbps | The person's **Resting Blood Pressure** (mm Hg on admission to the hospital) |
| Chol | The person's **Cholesterol Level** in mg/dl |
| FBS | The person's **Fasting Blood Sugar** (> 120 mg/dl):  1 = True  0 = False |
| Restecg | **Resting Electrocardiographic Measurement**  0 = Normal  1 = ST-T wave abnormality  2 = Probable or definite left ventricular hypertrophy by Estes' criteria |
| Thalach | The person's **Maximum Heart Rate** achieved |
| Exang | **Exercise-induced Angina:**  1 = Yes  0 = No |
| Oldpeak | ST depression induced by exercise relative to rest (ST relates to positions on the ECG plot) |
| Slope | The Slope of the peak exercise ST segment:  Value 1: Upsloping  Value 2: Flat  Value 3: Downsloping |
| CA | The number of **Major Vessels** (0-3). |
| Thal | A blood disorder called **Thalassemia**:  3 = Normal  6 = Fixed defect  7 = Reversible defect. |

**1.1.3 Number of instances from each class:** The target variable (indicating the presence or absence of heart disease) has the following distribution:

* Class 0 (No Heart Disease): 411 instances
* Class 1 (Heart Disease): 509 instances​.

A screenshot of a computer code

Description automatically generated

The data in the screenshot provided, shows how many cases there are in each category:

* **Class 0**: 411 cases with no heart attack.
* **Class 1 to 4**: These classes indicate different levels or types of heart attacks:
  + **Class 1**: 265 cases
  + **Class 2**: 109 cases
  + **Class 3**: 107 cases
  + **Class 4**: 28 cases

So, Class 0 is for people who did not have a heart attack, and Classes 1 to 4 represent different types or severities of heart attacks. As we will be using a binary classification in the analysis, 1 will be considered as 0 and 1.

**1.1.4 Dataset View**

A screenshot of a computer

Description automatically generated

## **Proper split of dataset**

**Load and split the dataset into two parts (training and testing), assuring that it is balanced.**

The dataset is split into training and testing sets using the train\_test\_split function from the sklearn.model\_selection module. This function ensures that the dataset is balanced across classes during the split.

**A screen shot of a computer code

Description automatically generated**This code defines the data for training and testing a machine learning model. It first selects the important features (like age, cholesterol levels, and other relevant data) and the target variable (what you want to predict). Then, it splits the data into two parts: 80% for training the model and 20% for testing how well the model works. The split is done in a way that ensures it can be repeated exactly the same way each time, which helps in getting consistent results.

## **EDA on data set**

Exploratory Data Analysis (EDA) for the heart\_disease\_uci.csv dataset was conducted, and included various analyses such as:

* Age Distribution:

The dataset contains a wide range of ages, from 28 to 77 years. The average age of the participants is about 53 years, with most participants between 46 and 60 years​.

A red graph with blue lines

Description automatically generated

The age distribution was analyzed across different datasets, revealing differences in age distribution among locations like Cleveland, Hungary, Switzerland, and VA Long Beach​.

A screenshot of a computer

Description automatically generatedThe first piece of code filters the dataset to identify patients younger than 30 years old. It checks the age column to find those under 30, then selects their sex and dataset (location). The code counts how many times each combination of gender and location appears, allowing you to see how many young patients are in each location and their gender.

The second piece of code looks for patients older than 60 years old. It filters the dataset by checking the age column for values greater than 60. It then counts the number of patients over 60 in each location (dataset) and separates them by gender (sex). The result provides a breakdown of the older patients by location and gender.

The third piece of code calculates age statistics for each location. It groups the data by the dataset column, which represents different locations, and then calculates three key statistics: the average age (mean), the median age (median), and the most common age (mode). These statistics are rounded to two decimal places for clarity, giving a clear view of the age distribution in each location.

* Chest Pain (cp) Analysis:

The chest pain type (cp) was explored, showing four different categories: typical angina, atypical angina, non-anginal pain, and asymptomatic​.

A graph of different colored bars

Description automatically generated

The majority of the cases in the dataset were asymptomatic, followed by non-anginal pain​.

* Outlier Detection:

Outliers were detected using box plots for numerical features such as cholesterol (chol). Some rows were identified as outliers and treated accordingly​​.

A screenshot of a graph

Description automatically generated

* Missing Data:

Some columns had a significant number of missing values, particularly the ca (number of major vessels colored by fluoroscopy) and thal (possibly related to thalassemia). These columns were eventually dropped from the analysis to maintain the integrity of the data​.

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

The above code checks how many missing values are present in each column of the dataset. It goes through each column and counts the number of missing entries, then displays the results. This helps identify which columns have missing data and how much is missing.

A blue and red graph

Description automatically generated

The above code calculates the percentage of missing values for each column. It converts the count of missing values into a percentage to show how much of the data is missing from each column. The percentages are rounded for easier reading, and the columns are sorted to list the ones with the most missing data at the top. This helps you see which parts of the dataset might need fixing or attention.

A white rectangular object with blue and red text

Description automatically generatedThe code first removes three columns:

“slope”,”ca”, and “thal - from the dataset because they have too many missing values and might not be useful for the analysis. Next, it removes any remaining rows that still have missing values in any of the other columns. This helps to ensure that the data used for analysis is complete and reliable.

* Scaling:

Data was scaled using Min-Max scaling, which transformed features like age, cholesterol, and others to a standard range. This process is crucial for ensuring that all features contribute equally to the analysis​​.

A screenshot of a computer

Description automatically generated

A group of blue and white graphs

Description automatically generated with medium confidenceThis code is selecting specific columns from the dataset that need to be scaled: age, trestbps, chol, thalch, and oldpeak. Then, it uses a technique called MinMaxScaler to transform these columns so that their values are scaled to a range between 0 and 1. This is done to ensure that all the selected features have the same scale, which can improve the performance of many machine learning algorithms. The MinMaxScaler is first set up with the desired range (0 to 1), and then it is applied to the selected columns in the dataset.

This is the distribution of features after scaling:

## **Building the Classifiers**

**Create 3 different machine learning classifiers of your choice.**

Five classifiers are built using:

* Random Forest: Known for its good balance between performance and interpretability​.
* Support Vector Machine (SVM): Provides high accuracy and is competitive with the Random Forest model​.
* Logistic Regression: A simple yet effective model that performs well across all metrics​
* K-Nearest Neighbours (KNN): A simple, instance-based learning algorithm that classifies data points based on the majority vote of their k-nearest neighbors.
* Decision Tree (DT): A model that splits the data into subsets based on feature values, creating a tree-like structure.

A screenshot of a computer program

Description automatically generatedThis code defines five different machine learning models to predict outcomes: Logistic Regression, which is ideal for binary classification; Decision Tree, which makes decisions by splitting data into branches; Random Forest, an ensemble method that improves accuracy by combining multiple decision trees; Support Vector Machine (SVM), which separates classes with the best possible boundary; and K-Nearest Neighbors (KNN), which predicts outcomes based on the nearest data points. These models will be compared to determine which performs best on the given dataset.

## **Training and Testing the Classifiers**

**Train and Test the three classifiers and generate a confusion matrix, with a confusion matrix plot**

In order to determine which model is the most effective, we need to look at a combination of several metrics: Accuracy, Precision, Recall, F1-score, and ROC AUC score. Each of these metrics provides different insights into the performance of the model:

* Accuracy: Indicates how often the classifier is correct.
* Precision: Measures how many of the positive predictions were actually correct.
* Recall: Measures how many of the actual positives were captured by the model.
* F1-Score: Provides a balance between precision and recall.
* ROC AUC Score: Reflects the model's ability to distinguish between classes. The Random Forest model achieved the highest ROC AUC score of 0.8828, indicating strong performance​.

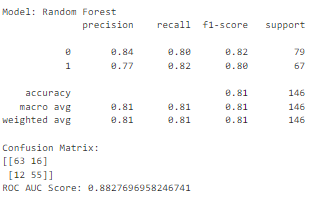
**A screenshot of a computer program

Description automatically generatedModel Training:** A pipeline is set up for each model, which first applies preprocessing to the data—such as scaling numerical features or encoding categorical ones. After preprocessing, the model is trained on this processed training data (X\_train and y\_train). Each model, including Logistic Regression, Decision Tree, and others, is trained in this manner using the pipeline.

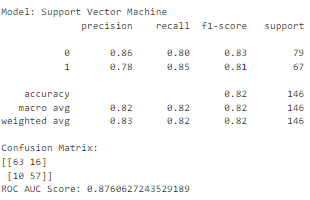
**Model Testing:** After training, predictions are made on the test data (X\_test) to assess the model’s performance. The code then evaluates these predictions by generating a classification report and confusion matrix, which provide detailed information on the model’s accuracy, precision, recall, and other metrics. Additionally, if the model can produce probability estimates (predict\_proba), the code calculates the ROC AUC score to measure how well the model distinguishes between classes. Each model’s performance is printed out, and the models are later sorted based on their ROC AUC scores for comparison.

After training, the classifiers are tested on the testing dataset, and the following metrics are calculated:

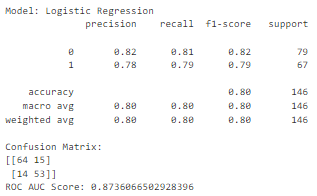
**Random Forest:**

* ****Accuracy: 81%
* Precision: 0.84 (Class 0), 0.77 (Class 1)
* Recall: 0.80 (Class 0), 0.82 (Class 1)
* F1-Score: 0.82 (Class 0), 0.80 (Class 1)
* ROC AUC Score: 0.8828 (Highest)
* Analysis: The Random Forest model performed the best overall, with the highest ROC AUC score of 0.8828, indicating strong performance in distinguishing between classes. Its accuracy, precision, and recall are also relatively high.

**Support Vector Machine (SVM):**

* Accuracy: 82%
* Precision: 0.86 (Class 0), 0.78 (Class 1)
* Recall: 0.80 (Class 0), 0.85 (Class 1)
* F1-Score: 0.83 (Class 0), 0.81 (Class 1)
* ROC AUC Score: 0.8761
* Analysis: The SVM model also performed very well, with a slightly lower ROC AUC score than Random Forest. It had the highest accuracy (82%) and strong precision and recall, making it a very competitive model.

**Logistic Regression:**

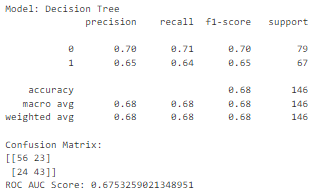
* Accuracy: 80%
* Precision: 0.82 (Class 0), 0.78 (Class 1)
* Recall: 0.81 (Class 0), 0.79 (Class 1)
* F1-Score: 0.82 (Class 0), 0.79 (Class 1)
* ROC AUC Score: 0.8736
* Analysis: Logistic Regression performed consistently across all metrics, with a good ROC AUC score and balanced precision and recall. It's a solid model, especially for its simplicity.

**K-Nearest Neighbors (KNN)**

* A screenshot of a computer

  Description automatically generatedAccuracy: 78%
* Precision: 0.82 (Class 0), 0.74 (Class 1)
* Recall: 0.76 (Class 0), 0.81 (Class 1)
* F1-Score: 0.79 (Class 0), 0.77 (Class 1)
* ROC AUC Score: 0.8627
* Analysis: KNN has decent performance, with a slightly lower ROC AUC score and accuracy compared to the previous models. It still provides reasonable precision and recall, making it a viable option, though not the top performer.

**Decision Tree:**

* Accuracy: 68% (Lowest)
* Precision: 0.70 (Class 0), 0.65 (Class 1)
* Recall: 0.71 (Class 0), 0.64 (Class 1)
* F1-Score: 0.70 (Class 0), 0.65 (Class 1)
* ROC AUC Score: 0.6753 (Lowest)
* Analysis: The Decision Tree model performed the weakest among the five, with the lowest accuracy and ROC AUC score. It struggled with precision and recall, indicating that it may not be the best choice for this dataset.

## **Explaining and comparing the results**

**Discuss the results of the three sets.**

* Best Performer: The Random Forest model is highlighted as the best performer, with the highest ROC AUC score and solid metrics across the board.
* Runner-up: The SVM is close behind, with strong accuracy and a competitive ROC AUC score.
* Good Alternative: Logistic Regression is also a strong contender due to its simplicity, consistent performance across metrics and ease of interpretation.

Other Models: K-Nearest Neighbors and Decision Tree, did not perform as well in comparison, making them less ideal for this specific dataset and problem.

## **Conclusion**

The analysis involved splitting the dataset into training and testing sets while ensuring that the class distribution was balanced. An extensive Exploratory Data Analysis (EDA) was performed to understand the data, including the distribution of age, cholesterol, and blood pressure, as well as handling missing values and outliers. Five different classifiers were built: Logistic Regression, Decision Tree, Random Forest, Support Vector Machine (SVM), and K-Nearest Neighbors (KNN). These models were trained and tested, with the Random Forest classifier emerging as the best performer, achieving the highest ROC AUC score of 0.8828. The SVM was a close second, while Logistic Regression provided consistent results, making it a good alternative. KNN and Decision Tree performed less effectively in comparison, indicating they were less suitable for this dataset