

COMP 4433

Data Mining and Data Warehousing

Project (individual work)

December 2024

Kaja Loennecken, 24010845x

Link to GitHub: <https://github.com/kajaloennecken/COMP4433-Project/blob/main/Final.ipynb>

The jupyter notebook can also be found as a pdf-file (Final.pdf) and an ipynb-file(final.ipynb)

## Introduction

In this report, I will present a data mining solution based on the Heart Attack Analysis and Prediction (HAAP) dataset, summarizing the work undertaken throughout the project, particularly focusing on the formulation, analysis, and evaluation phases. The objective of this project is to explore and implement data mining solutions and gain useful experience by applying this to a practical scenario.

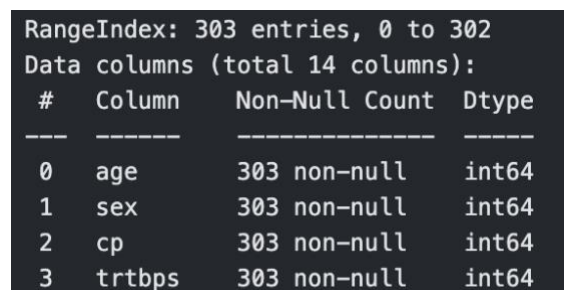
The dataset includes key medical features such as age, cholesterol levels, blood pressure, exercise-induced angina, and other relevant attributes, which can be leveraged to build predictive models for heart disease diagnosis. Through this project, I aim to design, implement, and evaluate a solution that predicts heart disease and also uncovers hidden patterns and relationships within the data. The analysis begins with a thorough understanding of the dataset, followed by preprocessing and preparing the data for further analysis.. The report will detail the steps involved in exploratory data analysis, visualization, and the exploration of clustering and association rule mining to gain deeper insights into the data. Additionally, both unsupervised and supervised learning methods, including clustering and classification models, will be applied. The performance of these models will be thoroughly evaluated and compared to ensure the best possible outcome. Throughout the report, I will document my methods, insights, and key decisions made during the project, providing a comprehensive overview of the entire data mining process.

## Formulation, design and implementation

### 1.Dataset preprocessing/cleaning

Before doing any analysis, I wanted to gain a brief understanding of the dataset and its content that would be useful for further analysis. In this part the dataset was also cleaned and preprocessed to ensure that the data was suitable for further analysis. The data cleaning process included handling missing values, correcting erroneous data entries, removing duplicates, and detecting outliers.

I was provided with a dataset, HEART\_DATA, where I will conduct our analysis and apply data mining techniques. When I ran HEART\_DATA.info(), I was able to understand the data types and check for missing values.

A screenshot of a terminal window showing the output of the pandas .info() method for a dataset named HEART\_DATA. The output indicates 303 entries from index 0 to 302 and 14 columns in total. A table lists the first four columns: #, Column, Non-Null Count, and Dtype. The first four columns are age, sex, cp, and trtbps, all with a non-null count of 303 and dtype of int64.

RangeIndex: 303 entries, 0 to 302			
Data columns (total 14 columns):			
#	Column	Non-Null Count	Dtype
0	age	303 non-null	int64
1	sex	303 non-null	int64
2	cp	303 non-null	int64
3	trtbps	303 non-null	int64

*Figure 1*

As shown in Fig. 1, the dataset does not contain any null values, but since all of the data types were numerical, I could not be sure if the value '0' represented a missing value. Therefore, in the first step of

cleaning the data, I further investigated the dataset to ensure the values matched the data description provided.

## 1.1 Missing values

The first step in the data cleaning process was to inspect the dataset for any missing values. After checking the dataset using the `isnull().sum()` method, we confirmed that there were no missing values in most of the columns. However, two columns ('caa' and 'thall') contained missing values after correcting some erroneous entries.

For the 'caa' column, we observed that it had a value of 4, which is not valid (the valid values are between 0 and 3). These entries were replaced with NaN (missing values) to signify invalid data. Similarly, the 'thall' column had a value of 0, which is also invalid (valid values are between 1 and 3). Therefore, rows with `thall = 0` were replaced with NaN. Given the small proportion of missing data (only 5 missing values in caa and 2 missing values in thall), it was decided to use imputation techniques to handle these missing values. Specifically, I opted to fill the missing values with the mode (the most frequent value) of each column, as it was a simple and effective strategy for columns with categorical data.

## 1.2 Outlier detection

Next in the dataset preprocessing I wanted to look for any significant outliers that could affect the accuracy of the models. Outlier detection was performed using visualizations such as boxplots, scatter plots, and cumulative distribution functions (CDFs) for the numerical features (age, chol, trtbps, thalachh, oldpeak). There did not seem like there were any significant outliers in the data, and the data is quite clean. So I will proceed with the data as is without removing any outliers.

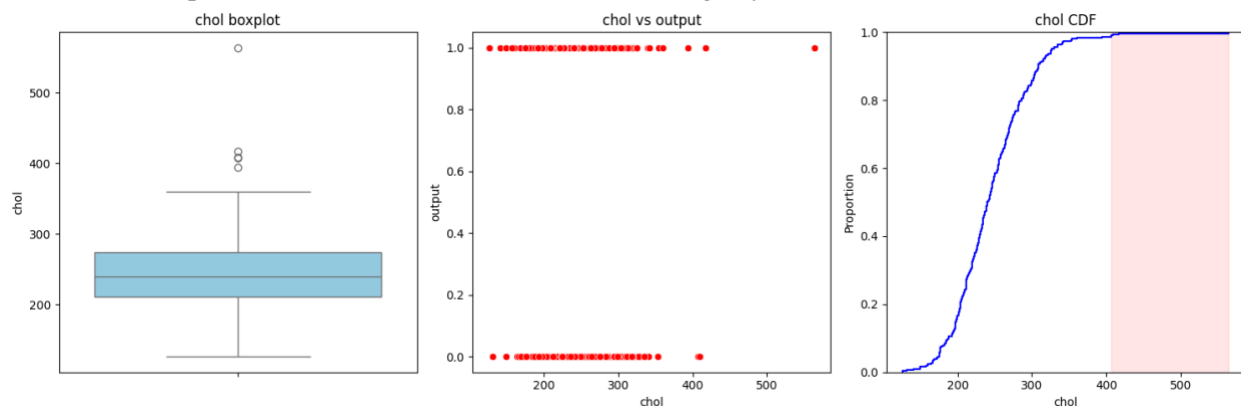


Figure 2: Outlier detection plots

## 1.3 Data visualization and analysis

To understand the dataset in detail, I decided to visualize different distributions of the dataset. I started by examining the output distribution to get a brief understanding of how the data is distributed between the two target classes (heart disease presence or absence). As shown in the pie chart below, the dataset has a balanced distribution of instances with and without heart disease

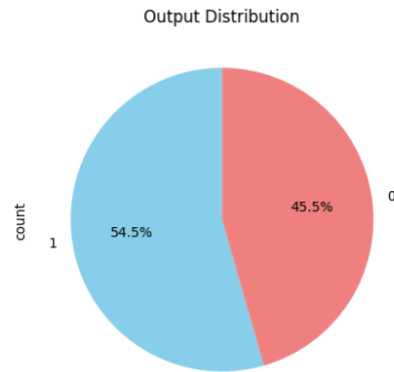


Figure 3: Distribution of output

Secondly, I explored the distribution of categorical features. For example, the distribution of the `sex` column indicates that the dataset is predominantly male, with a much smaller proportion of females. Similarly, the `cp` (chest pain type) column shows a varied distribution, with the most common chest pain type being 'typical angina', followed by atypical angina, non-anginal pain, and asymptomatic. Please refer to the notebook for additional categorical feature distributions.

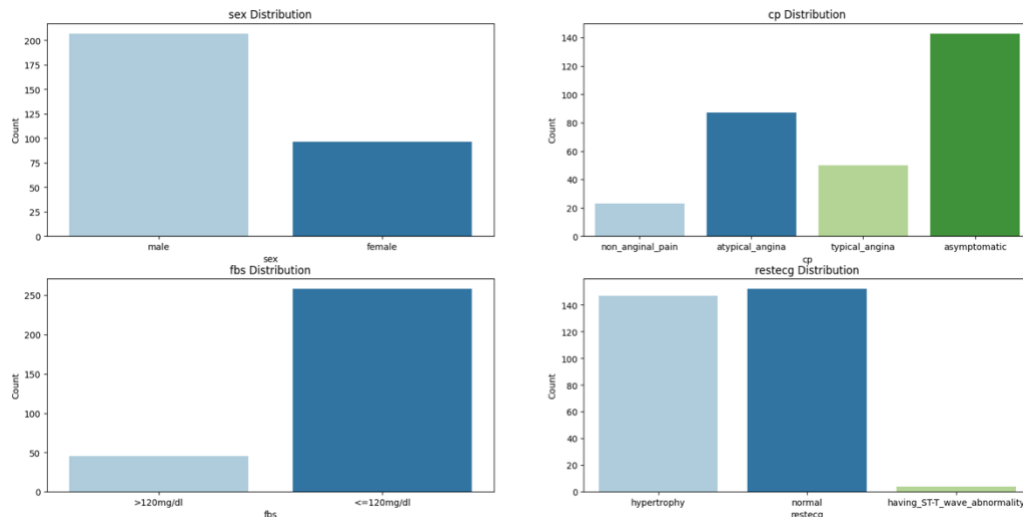


Figure 4: Distribution of Categorical columns

Lastly, I examined the distribution of continuous numerical feature columns. The distribution of variables like `chol` (serum cholesterol) and `thalachh` (maximum heart rate achieved) shows considerable variation,

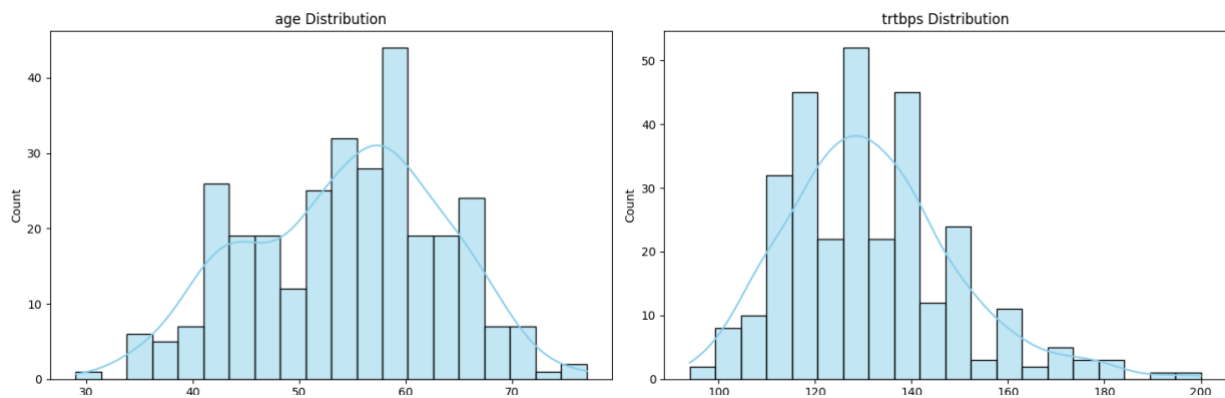


Figure 5: Numerical feature distribution

which may be useful for detecting patterns or trends that influence heart disease prediction. These visualizations help identify any skewness, outliers, or potential issues with the data, which can inform further preprocessing steps.

## 1.4 Association Rule Mining

By using a minimum support of 40% and a confidence of 80%, I aimed to uncover strong association rules between the categorical features in the dataset. The strong association rules were defined as those having a lift greater than 1, indicating that the features occur together more frequently than would be expected by chance. The rules were then visualized in a network graph, as shown below.



*Figure 6: Graph of association rules*

## 2 Feature engineering

Feature engineering was an essential step trying to improve model performance. I added several new features to the dataset, starting with Quantile-based Discretization. This method divides each numerical feature into three equal-sized bins, resulting in three additional categorical columns (after encoding) for each numerical feature that indicate which bin the values belong to. Other interesting feature I added were interaction features created by multiplying the age feature with other variables like thalachh (max heart rate), chol (cholesterol and old peak (ST depression)). These interaction terms might reveal more complex relationships between age and other health metrics that could provide a better estimate for the predictions.

Given the moderate size of the dataset, I decided not to use LASSO regression for feature selection but instead applied Principal Component Analysis (PCA) to reduce multicollinearity. PCA allows for

dimensionality reduction while retaining as much of the data's variance as possible. I compared the model performance with and without PCA to assess its impact on predictions.

## 2.1 Principal component analysis

Using PCA, I conducted a grid search to determine the optimal number of components. The grid search revealed that the best number of components was 5. This PCA transformation was integrated into the model pipeline, and its performance was compared to the model without PCA to evaluate its effectiveness in improving the model's predictive power.

## 3 Formulation

The goal of this project is to develop a data mining solution for predicting heart disease using the Heart Attack Analysis and Prediction (HAAP) dataset. In this formulation, I will outline the steps taken to understand the problem, design the solution, and select appropriate techniques for both classification and clustering. Furthermore, I explored the possibility of enhancing the performance of classification models by incorporating the results from clustering.

### 3.1 Problem definition

The objective of this project is to predict whether an individual is at risk for a heart disease based on a set of medical attributes. This is a binary classification problem where the target variable is:

- 0: No heart disease
- 1: Heart disease present

The challenge is to accurately classify patients into one of these two categories using the available medical features.

### 3.2 Dataset overview

The HAAP dataset consists of 303 entries with 14 features, including both numerical and categorical variables. Examples are:

- **Age:** The patient's age in years.
- **Sex:** The patient's gender (1 = male; 0 = female).
- **CP (Chest Pain Type):** The type of chest pain.

Trtbps (Resting Blood Pressure), Chol (Serum Cholesterol), Thalachh (Max Heart Rate), and other health-related measurements.

**Target Column (Output):** Whether the patient has heart disease (1) or not (0).

The dataset was thoroughly cleaned and preprocessed (as detailed in the previous section), including handling missing values, correcting erroneous entries and scaling features.

### 3.3 Selection of Data Mining Techniques

To solve the heart data disease prediction problem, I decided to use a combination of supervised and unsupervised data mining techniques. These techniques were selected to address both the prediction of heart disease and the exploration of potential hidden patterns in the data. I will now present the specific techniques that I decided to explore and use for this dataset.

### 3.4 Unsupervised learning – clustering

As part of the unsupervised learning approach, I applied clustering techniques to identify potential patterns in the data that may not be immediately apparent through supervised classification alone. To explore these patterns, I used two clustering algorithms: KMeans and DBSCAN.

For KMeans clustering, I applied PCA to reduce the dimensionality of the dataset and then determined the optimal number of clusters through silhouette analysis. The clustering results were visualized in a 2D scatter plot, with distinct colors representing different clusters and the centroids marked for reference. The figure below illustrates the KMeans clustering results. It's important to note that KMeans was initially performed on 2-dimensional data for better visualization. Later, clustering was also applied to the data without PCA and with higher-dimensional PCA components to assess the impact of dimensionality on the clustering results.

In addition to KMeans, I explored DBSCAN for density-based clustering. DBSCAN is particularly useful for identifying outliers and smaller subgroups within the data, though it is not typically used as a preprocessing step for classification. I visualized the DBSCAN results as well, where distinct clusters and noise points were marked differently, as shown in the figure below.

The analysis revealed that KMeans identified two clusters with a relatively good silhouette score, suggesting that this method provides a reasonable partitioning of the data. However, the clusters were not sharply defined, and there was some overlap. This indicated that the data may lack clear, natural clusters, and a more complex clustering algorithm, such as DBSCAN, might be better suited for identifying smaller, denser subgroups.

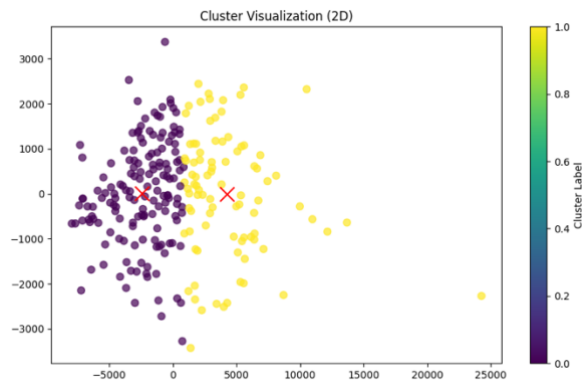


Figure 7: KMeans clusters

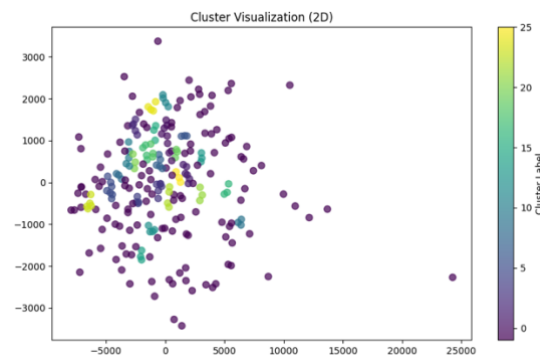


Figure 8: DBSCAN clusters

Additionally, I investigated the distribution of the target variable within the identified clusters.



Figure 9: Distribution of target variable within identified clusters

### 3.5 Supervised learning – classification

In this section, a predictive classification model was developed and implemented to predict the likelihood of heart disease in patients based on several medical attributes. The main goal was to explore different classification algorithms, compare their performances, and analyze how preprocessing techniques such as PCA influence the models' performance

#### 3.5.1 Model selection

To create a robust and high-performing classification model, I combined the preprocessing steps (discussed earlier) with a diverse set of classification algorithms. The aim was to determine which combination of models and data preprocessing would lead to the best performance.

The two types of data used for classification were:

- Raw data: The original dataset without dimensionality reduction applied.
- PCA-applied data: Data where Principal Component Analysis (PCA) was applied to reduce dimensionality.

The classification models explored included both basic and advanced algorithms. A basic model, such as Logistic Regression, was used as a baseline to compare against more advanced models, like Random Forest, XGBoost, and Support Vector Classifiers (SVC). Additionally, models such as Gradient Boosting and CatBoost were explored to evaluate their performance in the context of this dataset.

To ensure the best performance for each model, hyperparameter tuning was performed using two techniques, GridSearchCV and BayesSearchCV. Both methods were used with 5-fold cross-validation to validate the models and prevent overfitting. The goal was to find the optimal combination of hyperparameters that would result in the highest accuracy, precision, recall, and F1 score.

#### 3.5.2 Results:

After tuning the models and performing cross-validation, we obtained an overview of the best-performing models with their respective metrics. The results were summarized in a performance table (shown below), which compares the models' performance on both raw data and PCA-applied data. As seen in the table below, the best classification model according to the test accuracy is SCV with the data that is PCA applied with an accuracy of 90%. If we evaluate based on test precision, we can see that GaussianNB, XGBClassifier, RandomForestClassifier, KNeighborsClassifier and LogisticRegression all with PCA applied gives a precision of 93%.

	Classifier	Data	Selected parameters	CV score	Train_accuracy	Test_accuracy	Train_precision	Test_precision
3	SVC	PCA Best	{'classifier__C': 0.01, 'classifier__degree': ...}	-0.826276	0.822314	0.901639	0.808219	0.906250
23	GaussianNB	PCA Best	{'classifier__var_smoothing': 1e-09}	-0.838776	0.830579	0.885246	0.833333	0.931034
11	XGBClassifier	PCA Best	{'classifier__colsample_bytree': 1.0, 'classif...	-0.809779	0.871901	0.885246	0.859155	0.931034
5	RandomForestClassifier	PCA Best	{'classifier__max_depth': 10, 'classifier__max...	-0.805697	1.000000	0.885246	1.000000	0.931034
19	KNeighborsClassifier	PCA Best	{'classifier__n_neighbors': 7, 'classifier__p'...	-0.793793	0.826446	0.885246	0.818182	0.931034
1	LogisticRegression	PCA Best	{'classifier__C': 0.01, 'classifier__penalty':...	-0.838690	0.830579	0.885246	0.843284	0.931034
22	GaussianNB	Without PCA	{'classifier__var_smoothing': 1e-09}	-0.801531	0.830579	0.868852	0.828571	0.900000
21	MLPClassifier	PCA Best	{'classifier__activation': 'tanh', 'classifier...	-0.834609	0.818182	0.868852	0.820144	0.900000
17	AdaBoostClassifier	PCA Best	{'classifier__learning_rate': 0.01, 'classifie...	-0.809609	0.822314	0.868852	0.804054	0.900000

Figure 10: Performance dataframe



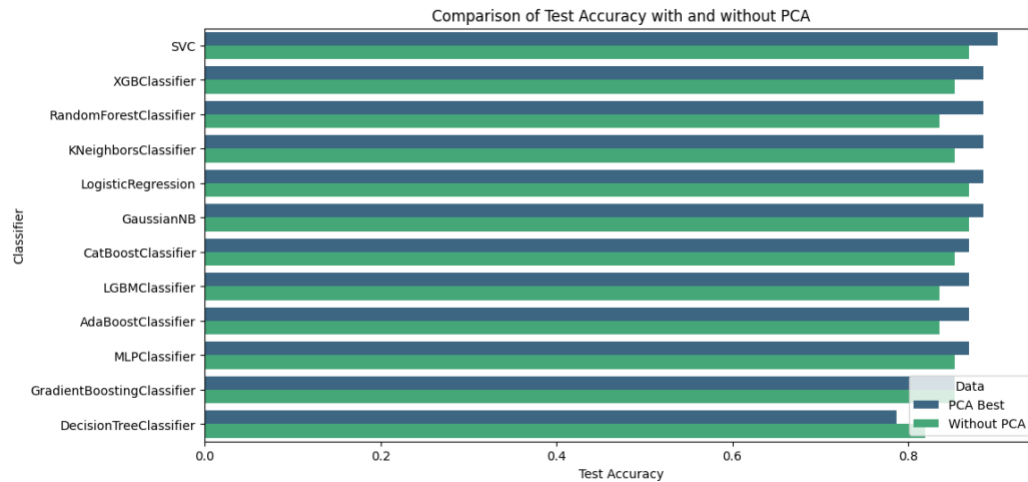


Figure 11: Plot of test accuracy for all classification models

We can also analyze the performance of our best models by looking at their confusion matrices. Below you can see an example of the confusion matrix for the SVC model with PCA.

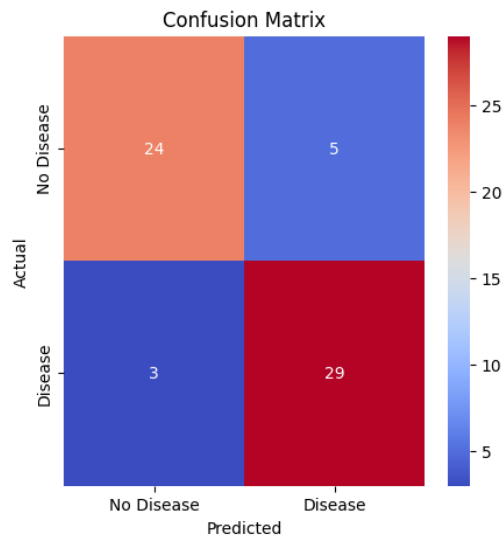


Figure 12: Confusion matrix for SVC with PCA

As we can see, the validation is done on a relatively small number of patients, as the dataset only contains 303 entries. This limited dataset size could have a significant impact on the classification results. With such a small dataset, it is important to consider the possibility of overfitting or the model being overly sensitive to the data splits. Furthermore, the presence of false positives or false negatives could have greater consequences depending on the application of the model, particularly in a medical setting.

## 2.6 Integrating clustering with classification

One of the key innovations in this project was the hypothesis that clustering results could enhance classification performance. The idea was to use the clusters identified by unsupervised learning

algorithms as additional features for the classification models. By doing this, I hoped to improve the model's ability to make predictions by leveraging the inherent groupings within the data. For this purpose, I chose to use KMeans as the clustering method, as this was the method that I thought was best suited for clustering as a preprocessing step for classification. As described above, it was also the method that obtained the best silhouette score. DBSCAN, on the other hand, is mostly useful for identifying small subgroups and detecting outliers (so-called "noise"). While DBSCAN can be useful in certain contexts, I decided to focus on KMeans for the task of clustering as a preprocessing step. After performing clustering with KMeans, the cluster labels were added as an additional feature to the dataset. These labels represented the group that each patient belonged to, providing the classification model with extra information about the structure of the data.

Comparing the performance of classification models with and without the clustering-based features, allowed me to evaluate whether the clusters provided meaningful insights that could help the models improve their predictions. By incorporating the clustering labels, we tested whether they helped the classifier understand data structure better.

From the table below, we can see that the test accuracy of the best model was slightly improved from 90% (using the raw features) to 91.8% (using the clustering-based features). The MLPClassifier with PCA obtained the best accuracy score with clustering, showing a small improvement when cluster labels were incorporated.

	Classifier	Data	Selected parameters	CV score	Train_accuracy	Test_accuracy	Train_precision	Test_precision
21	MLPClassifier	PCA Best	{'classifier__activation': 'tanh', 'classifier...	-0.826446	0.826446	0.918033	0.827338	0.935484
3	SVC	PCA Best	{'classifier__C': 0.01, 'classifier__degree': ...	-0.818112	0.809917	0.901639	0.800000	0.933333
23	GaussianNB	PCA Best	{'classifier__var_smoothing': 1e-09}	-0.805782	0.814050	0.885246	0.818841	0.931034
1	LogisticRegression	PCA Best	{'classifier__C': 0.1, 'classifier__penalty': ...	-0.834609	0.826446	0.885246	0.837037	0.931034
11	XGBClassifier	PCA Best	{'classifier__colsample_bytree': 0.80552577405...	-0.809949	0.917355	0.868852	0.900709	0.900000
4	RandomForestClassifier	Without PCA	{'classifier__max_depth': 5, 'classifier__max...	-0.834524	0.942149	0.868852	0.921986	0.852941
9	GradientBoostingClassifier	PCA Best	{'classifier__learning_rate': 0.1, 'classifier...	-0.793537	1.000000	0.868852	1.000000	0.900000
20	MLPClassifier	Without PCA	{'classifier__activation': 'relu', 'classifier...	-0.826190	1.000000	0.852459	1.000000	0.848485
19	KNeighborsClassifier	PCA Best	{'classifier__n_neighbors': 7, 'classifier__p'...	-0.805952	0.818182	0.852459	0.824818	0.896552
15	CatBoostClassifier	PCA Best	{'classifier__depth': 3, 'classifier__iteratio...	-0.814201	0.979339	0.852459	0.984848	0.896552
14	CatBoostClassifier	Without PCA	{'classifier__depth': 3, 'classifier__iteratio...	-0.846939	0.863636	0.852459	0.847222	0.848485

Figure 13: Performance dataframe when including clustering label

## Conclusion

KMeans clustering helped uncover patterns in the data, and integrating cluster labels into the classification models resulted in a slight accuracy improvement, from 90% to 91.8%. The MLPClassifier with PCA and clustering-based features performed best, demonstrating that combining clustering with classification can enhance model performance. While the results are promising, the small dataset suggests that further validation with larger, more diverse data could improve the outcomes. Overall, this project highlighted the effectiveness of combining different data mining techniques for predicting heart disease and improving diagnostic accuracy.