Introduction

Literature Review

Cardiovascular diseases (CVD) are a leading cause of death worldwide, making early diagnosis increasingly important in preventing severe health complications. CVD are complex and heterogeneous diseases, caused by environmental, genetic and behavioral factors [1]. This quality of the disease and nature of the clinical data makes CVD an ideal area for implementing machine learning to improve treatment and patient care [1]. Machine learning algorithms have been used to effectively predict the likelihood of heart disease. Modeling approaches using SVM and Neural Networks have demonstrated high accuracy in predicting heart disease across clinical data sets [2,3]. Logistic regression has been shown to be an informative model for CVD in the Cleveland Heart Disease dataset, due to its interpretability [4]. A modeling approach using Random Forest incorporated additional features, including obesity and smoking habits, and achieved an improved accuracy of 90% [5]. Overall, these studies demonstrate the importance of feature selection and inclusion of comprehensive health indicators in predictive modeling.

Dataset Description

The UCI Heart Disease dataset includes 920 individual patient records and 14 key attributes that are commonly used in predictive modeling to assess cardiovascular risk.

Dataset Link

UCI Heart Disease Dataset

Problem Definition

Problem

Insights regarding the prediction and the diagnosis of heart disease can be gained efficiently using Machine Learning models. Conventional diagnostic methods for CVD, such as stress tests and echocardiograms, are often time-consuming and expensive, presenting limitations.

Motivation

Accurate machine learning models for heart disease prediction can assist healthcare providers in making timely decisions and improving patient outcomes. These models could potentially be used to develop automated systems that predict a patient's likelihood of heart disease based on routine health data, enabling preventive care across different regions, including resource constrained areas.

Methods

Preprocessing for K-Means

Feature Selection

The first pre-processing step we took was analyzing the dataset for any attributes we deemed unnecessary for our ML model. We decided to drop two columns, “id”, which identified the unique id for

each patient, and “origin”, which cataloged the place of study. We dropped these two columns as neither has much indication of predicting heart disease.

```
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   id        920 non-null    int64
 1   age       920 non-null    int64
 2   sex       920 non-null    object
 3   dataset   920 non-null    object
 4   cp        920 non-null    object
 5   trestbps  861 non-null    float64
 6   chol      890 non-null    float64
 7   fbs       830 non-null    object
 8   restecg   918 non-null    object
 9   thalch    865 non-null    float64
 10  exang     865 non-null    object
 11  oldpeak   858 non-null    float64
 12  slope     611 non-null    object
 13  ca        309 non-null    float64
 14  thal      434 non-null    object
 15  num       920 non-null    int64
```

Figure 1. Original Dataset Columns

Imputation

Next, we took a look at the amount of NaN values in our dataset. We decided to drop three columns that exceeded our <50% NaN value threshold: “Ca”, “thal”, and “slope”. With the remaining features, we decided to impute the NaN values. We used two different imputers based on what kind of data we were working with: KNN Imputer for numerical values and Simple Imputer for categorical values. KNN imputer worked by replacing the NaN values by considering values of the nearest neighbors in the feature space. This takes into account the underlying structure and relationship within the data by leveraging similar data points. This is more contextually relevant than mean/median substitution. Simple Imputer addressed missing values in categorical variables with strategies like “most_frequent” or “constant”. This ensures that it treats categories as distinct entities without implying any numerical relationship between them. Doing these steps before label encoding ensured that the imputation process respected the categorical nature of the data and avoided numerical associations between columns.

```
trestbps: 6.42% NaN values
chol: 3.26% NaN values
fbs: 9.79% NaN values
restecg: 0.22% NaN values
thalch: 5.98% NaN values
exang: 5.98% NaN values
oldpeak: 6.75% NaN values
slope: 33.62% NaN values
ca: 66.38% NaN values
thal: 52.77% NaN values
```

Figure 2. Percentage of NaN Values in Dataset

Categorical Features Encoding

After that, we decided to encode the categorical columns in our dataset. We analyzed two different approaches for doing so: label encoding and one hot encoding. Label encoding will transform the categories into integers. PCA will then interpret this as ordinal and interval data. This creates artificial relationships between categories (which we don't want), leading to misinterpretation and misleading PCA components. One hot encoding creates a new column for each category with values of 0 and 1 indicating the absence or presence of the category. Since we wanted to use PCA as a later pre-processing step, we decided that label encoding is better as it allows PCA to process categorical data without misinterpreting the variance structure.

Data Scaling

Our next pre-processing step was to standardize our data values. We used StandardScaler to rescale our features to have a mean of 0 and a standard deviation of 1. This gives all features fair comparison, especially in K-means where everything is distance-based calculation.

Principal Components Analysis

Lastly, we implemented PCA on our dataset. By using PCA, we were able to reduce dimensionality of our dataset based on variance. We selected only the top X principal components based on their variance, and neglected ones with low variance on the rationale that they only generate noise and don't affect clustering as much. We used a threshold of 95%, meaning we selected the first 9 principal components, which was just enough to retain the desired amount of variance.

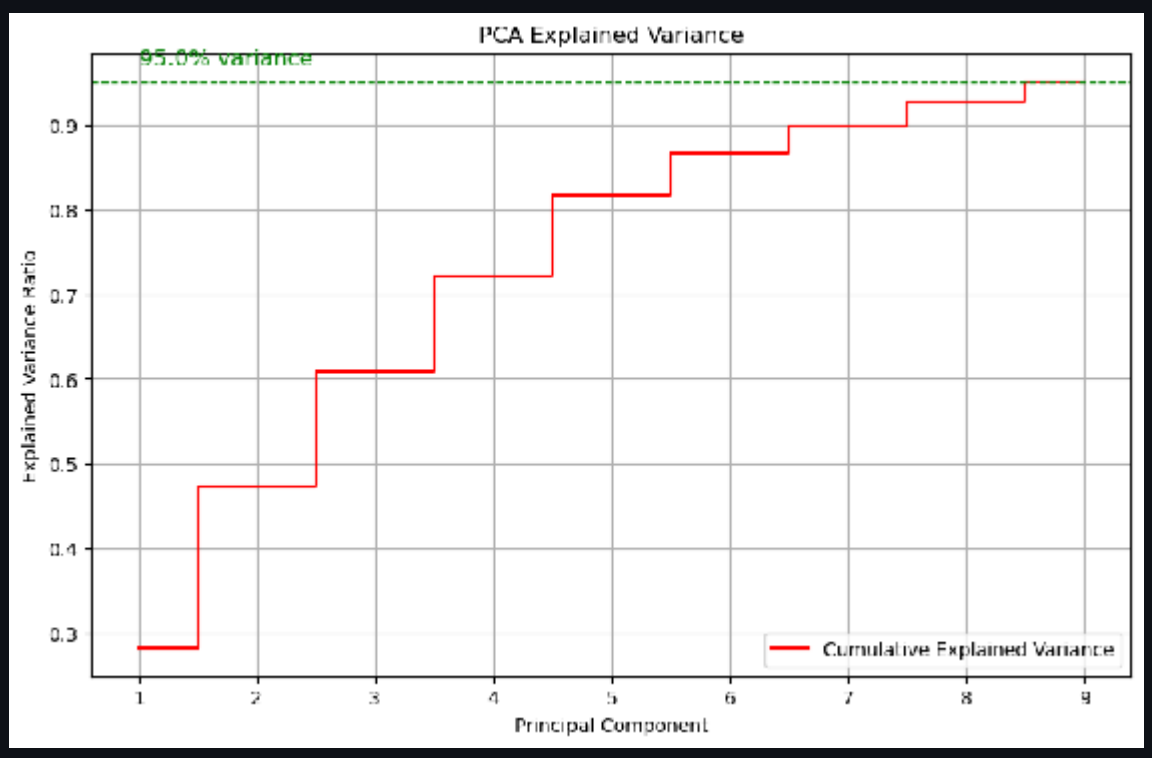


Figure 3. PCA Explained Variance

Preprocessing for Random Forest and Logistic Regression

The preprocessing strategies for regression and Random Forest models differ significantly in how they handle data, reflecting the unique requirements of each algorithm. These differences emerge at every stage of preprocessing, from data cleaning to feature scaling and dimensionality reduction, and play a crucial role in optimizing model performance.

Feature Selection

In both cases, the preprocessing begins with dropping redundant columns, such as "id" and "dataset," which do not contribute to the predictive power of the models. This step ensures the focus remains on meaningful features. While this step is the same for both preprocessing pipelines, subsequent stages diverge to cater to the specific needs of Logistic Regression and Random Forest models.

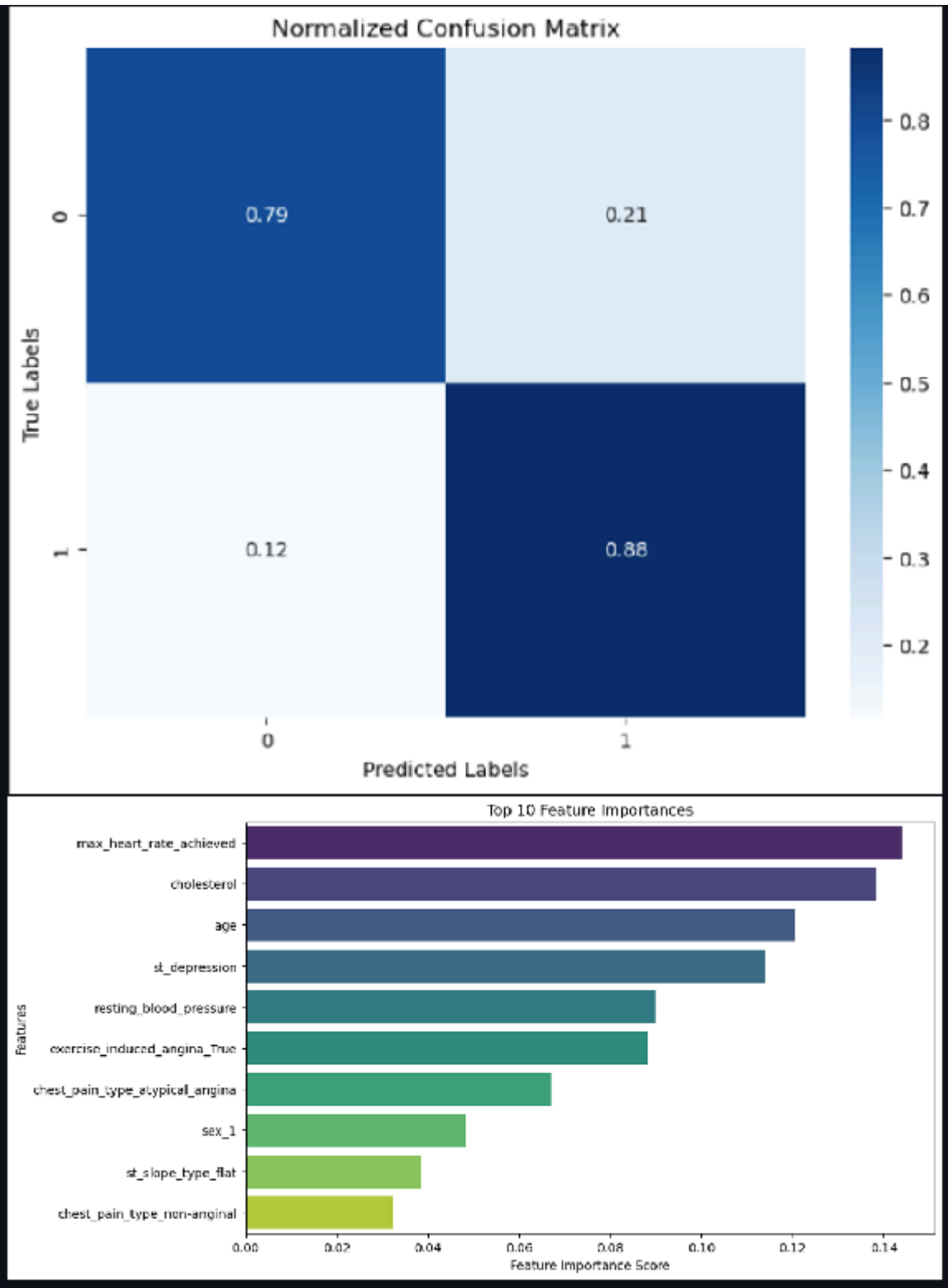


Figure 4. RF Outcomes with Feature Selection

Handling Missing Values

The next stage involves handling missing values. For Linear Regression, missing data is imputed using Iterative Imputer with a Random Forest estimator. The Iterative Imputer is robust in high-dimensional datasets because it does not rely on distance metrics. Instead, it models missing values iteratively using predictive models. This method captures complex feature relationships, ensuring imputed values are consistent with the dataset's structure, which is critical for regression models sensitive to data completeness. The Random Forest pipeline also uses Iterative Imputer with the same configuration, though the impact is less pronounced since Random Forest models are inherently robust to missing values. The KNN Imputer, however, struggles in high-dimensional datasets due to the "curse of dimensionality," where distance metrics lose meaning as the number of features increases. For comparison, below are the confusion matrices from using Random Forest on KNN and Iterative imputer respectively.

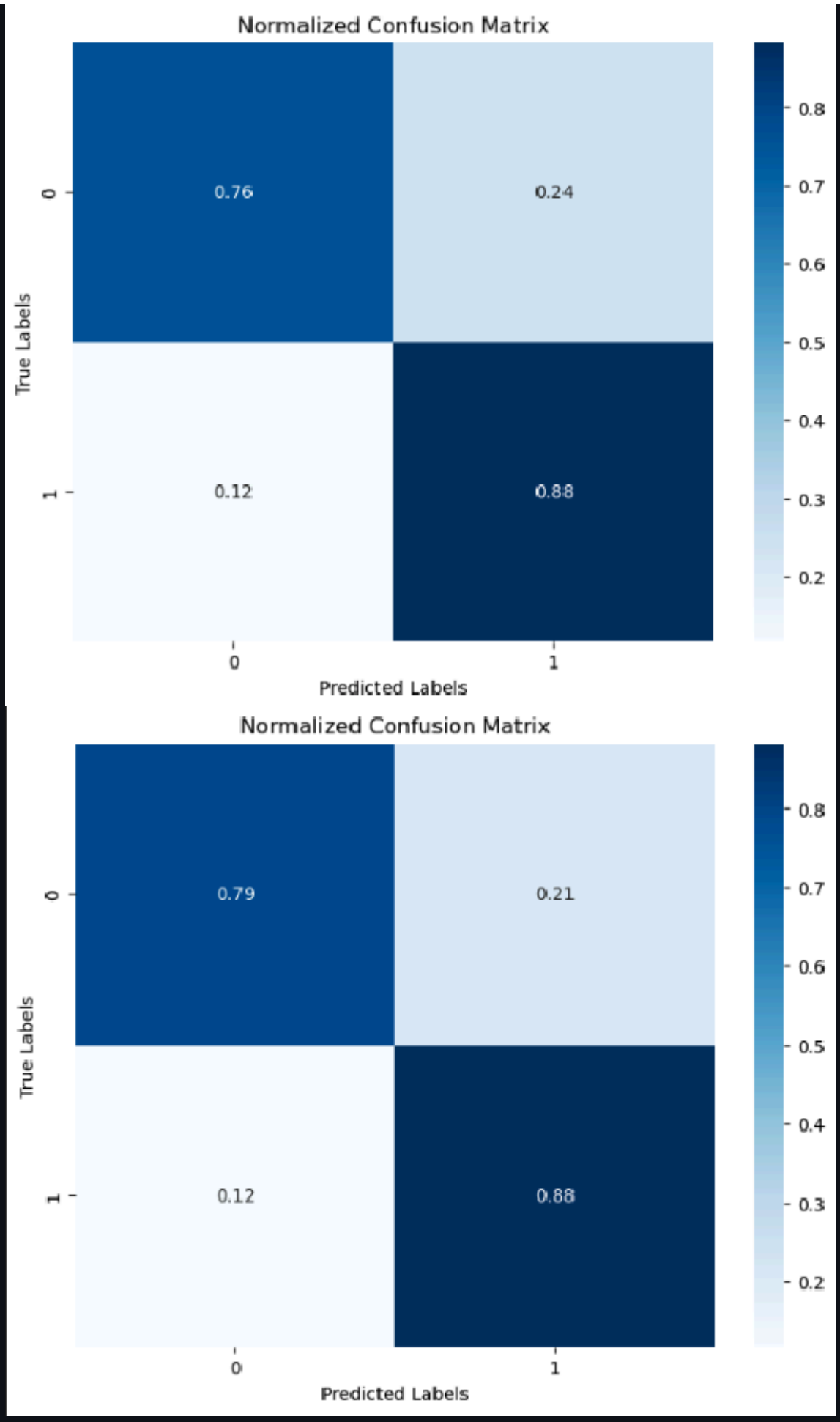


Figure 5. RF Outcomes with KNN Vs. with Iterative Imputer

Encoding

When encoding categorical variables, the Logistic Regression preprocessor uses One-Hot Encoding, converting categorical variables into binary columns, with the "drop_first" option enabled to prevent the dummy variable trap. This approach ensures categorical variables are represented without introducing ordinal relationships, which could mislead the regression model. The Random Forest pipeline also employs One-Hot Encoding with "drop_first," though this step is less critical for Random Forest models, as they can handle categorical data more flexibly in many implementations. Shown below are the confusion matrices and feature importance from using Random Forest on label encoding and one-hot encoding respectively.

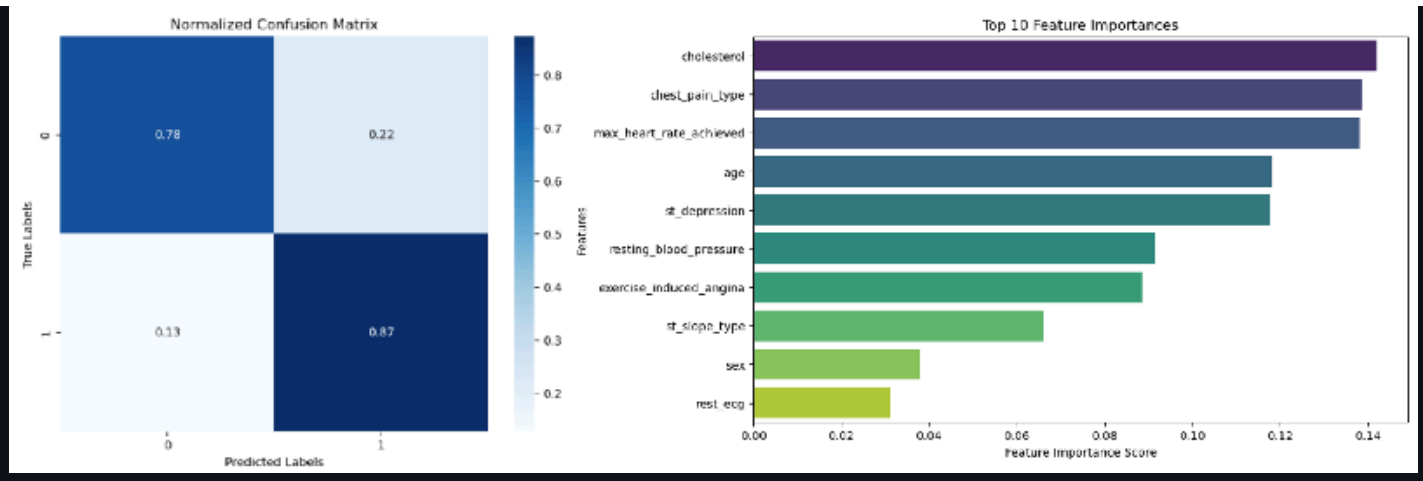


Figure 6. RF Outcomes with Label Encoding

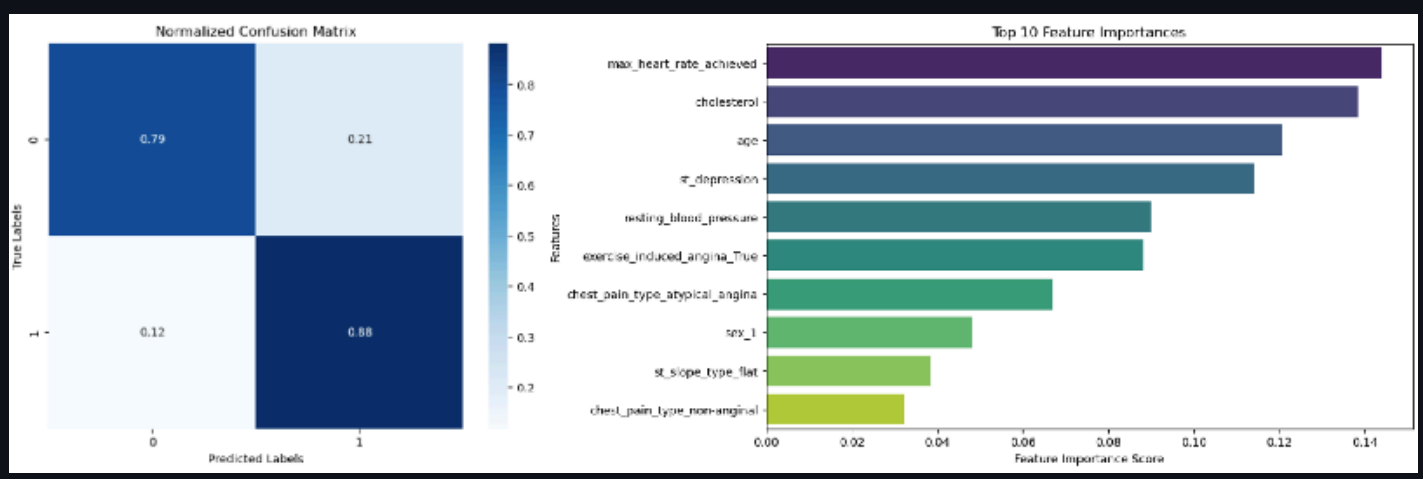


Figure 7. RF Outcomes with One-Hot Encoding

Scaling

Feature scaling represents a significant difference between the two pipelines. The Logistic regression preprocessor applies standard scaling, transforming numerical features to have a mean of zero and a standard deviation of one. This normalization is crucial for regression models, which rely on feature magnitudes to calculate coefficients. In contrast, Random Forest models are scale-invariant, allowing the Random Forest preprocessor to skip this step entirely, saving computational resources without sacrificing accuracy.

Dimensionality Reduction

Dimensionality reduction through Principal Component Analysis (PCA) is another area where the two preprocessors diverge. The Logistic Regression pipeline enables PCA, retaining 95% of the dataset's variance to reduce dimensionality and prevent overfitting. By focusing on the most significant components, the regression preprocessor simplifies the feature space, reducing computational complexity and improving model performance. On the other hand, the Random Forest preprocessor disables PCA entirely. Random Forest models are well-suited to handling high-dimensional data and do not benefit from the linear combinations of features that PCA provides.

Outlier Handling

Outlier detection and handling also differ in their emphasis. For Logistic Regression, outliers are detected using the Z-score method and removed if they exceed a threshold of three standard deviations. This step is critical for regression models, where extreme values can disproportionately influence predictions. While the Random Forest preprocessor employs the same detection and removal strategy, its importance is reduced due to the robustness of Random Forest models against outliers.

Data Labeling

The original dataset had target labels 0, 1, 2, 3, and 4 representing different severities of heart disease (0 being no heart disease and 4 being the most severe). However, due to insufficient data samples for each category, we transformed the target into a binary classification in which 0 indicates “no heart disease” and 1 indicates “has heart disease”. This simplifies the task, addresses class imbalance, and aligns with the goal of identifying the presence of heart disease.

Algorithms

Unsupervised Learning

K-Means Clustering

We chose to use K-means to model our dataset since it is an unsupervised clustering algorithm that draws insights to datasets by identifying clusters. No data label is required, potentially increasing the data available by reducing data pre-processing effort. K-means is also efficient when handling large datasets. The K-means approach involves grouping the data into clusters based on distance metrics, the default being euclidean distance. Alternative distance metrics can also be used for K-means, including squared Euclidean distance, Manhattan distance and Chebyshev. For this project, we compared 3-5 distance metrics and ended up using the Euclidean distance. A summary of performances across different distance metrics for K-means generated the following results. Based on the results when comparing the distance metrics, we will try different distance metrics in our model development and compare accuracy and performance. Both Chebyshev and Chi-square distance metric showed a comparable silhouette score to euclidean distance.

We then used the elbow method to find the ideal number of clusters to use in the K-means clustering of our dataset. The elbow method selects the point on the curve where the slope changes from greater than 1 to less than 1, or vice versa. was used to identify the ideal number of K clusters for K-means. We plotted the Within-Cluster Sum of Square (WCSS) against the number of clusters to output the following graph. Based on our elbow method graph, the ideal number of clusters was 5 for K-means.

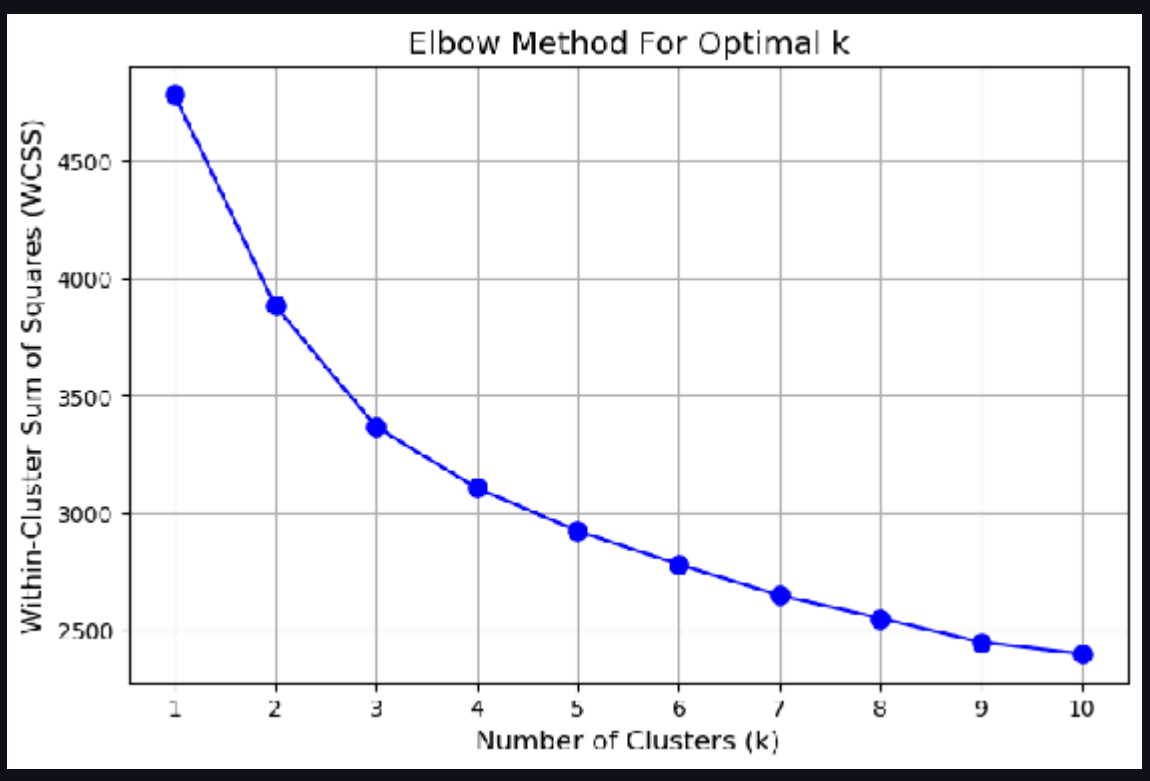


Figure 8. Elbow Method

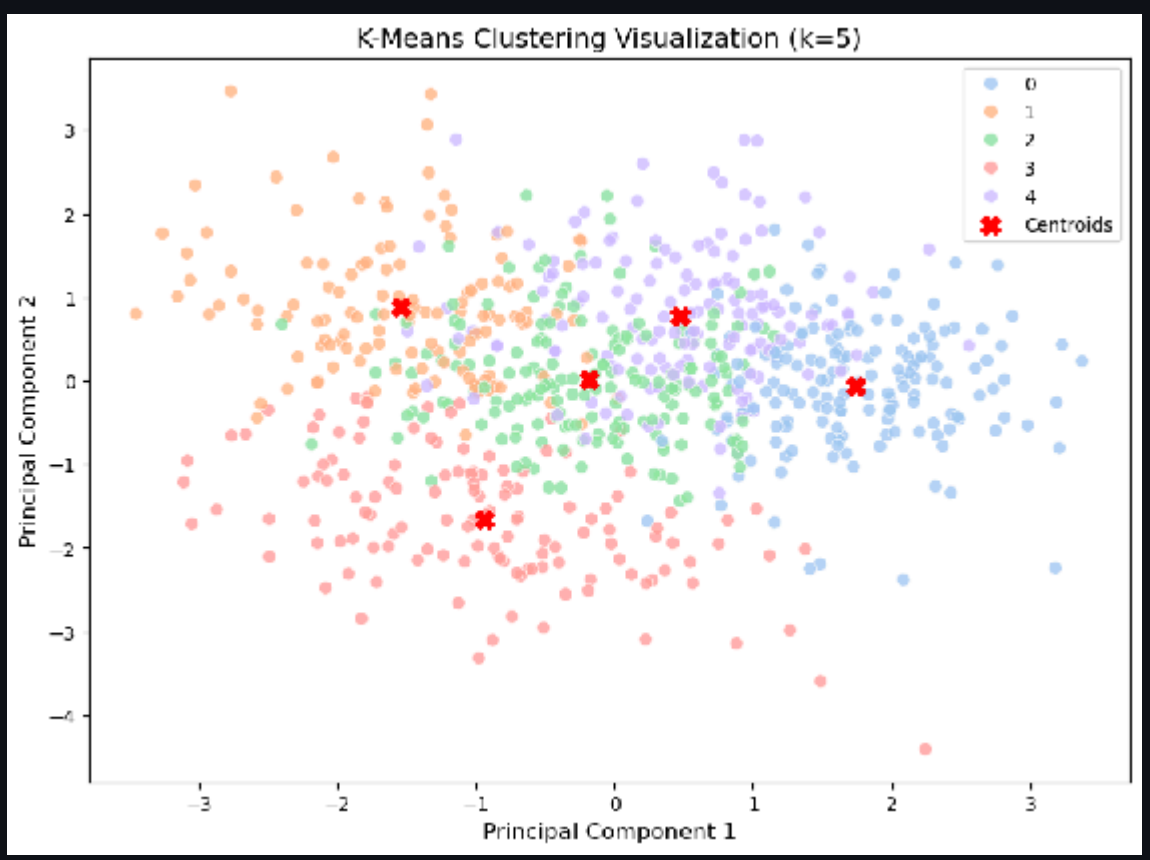


Figure 9. K-Means Clustering Visualization

Supervised Learning

Logistic Regression

Logistic regression is a widely used statistical method for binary classification, especially in domains like medical diagnostics. Unlike linear regression, which predicts continuous values, logistic regression estimates the probability of an observation belonging to a specific class. For instance, in heart disease prediction, it predicts the likelihood of a patient having or not having heart disease. This makes it ideal for analyzing complex datasets with binary outcomes, leveraging its capacity to transform linear relationships into probabilistic interpretations using the sigmoid function.
In this analysis, we compared two primary preprocessing approaches: unscaled and scaled methodologies. The unscaled preprocessing approach, which retains features in their original units and ranges, unexpectedly demonstrated superior performance in the heart disease prediction model. Feature scaling techniques typically aim to normalize features to a standard range, potentially reducing the impact of variables with larger magnitudes. Methods like standardization, min-max scaling, and robust scaling are commonly employed to transform data. However, in this specific study, preserving the intrinsic characteristics of the original data proved more effective, challenging conventional data preprocessing norms.

Random Forest

Random Forest is a robust ensemble machine learning algorithm used for classification tasks. It builds multiple decision trees during training and outputs the class with the majority vote for classification. We choose this method for our heart disease prediction task as it is particularly effective in handling high-dimensional data and preventing overfitting by averaging results from multiple models.
To optimize the performance of our Random Forest model, we used a hyperparameter tuning process through GridSearchCV with 5-fold cross-validation. We explored a comprehensive range of hyperparameter combinations, including the number of estimators, maximum tree depth, minimum samples for splitting and leaf nodes, feature selection strategies, and bootstrap sampling. The hyperparameter space explored is summarized in the following grid:

```
param_grid = {
'n_estimators': [50, 100, 200,250, 350],
'max_depth': [None, 10, 20, 30],
'min_samples_split': [2, 5, 10],
'min_samples_leaf': [1, 2, 4],
'max_features': ['sqrt', 'log2', None],
'bootstrap': [True, False]
}
```

The dataset was split into training and validation sets for each fold, and the model was trained and evaluated on these splits. GridSearchCV iterates through all possible combinations of hyperparameters. The best-performing hyperparameters were identified based on the mean accuracy score across all folds. However, the result provided does not yield the best performing hyperparameters. Thus we applied additional manual grid search based on the result of GridSearchCV. The following optimal hyperparameters were identified:

- Best Parameters: {'bootstrap': True, 'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 350}
- Best Model Accuracy: 84.78%

After determining the best hyperparameters, we trained the model and evaluated the model's performance on the test set using metrics such as precision, recall, F1-score, and ROC-AUC.

Results and Discussion

Supervised Learning

Logistic Regression

Accuracy Metrics

Class	Precision	Recall	F1-Score	Support
0	0.84	0.79	0.82	82
1	0.84	0.88	0.86	102

Metric	Value
Accuracy	0.8424
Precision	0.8425
Recall	0.8424
F1-Score	0.8418
ROC-AUC	0.9142

Table 1. Logistic Regression Model Performance

The model's performance was rigorously evaluated using multiple statistical metrics. Precision measures the model's accuracy in predicting positive instances. For both classes, precision was 0.84, indicating a balanced ability to avoid false positives. Recall measures sensitivity, or the model’s ability to correctly identify actual positives. Recall was slightly higher for the heart disease class (0.88) compared to the no heart disease class (0.79), suggesting the model was better at identifying patients with heart disease. F1 Score, the harmonic mean of precision and recall, provides a balanced measure of performance. The scores of 0.82 and 0.86 for the two classes confirm robust performance. The overall Accuracy was 0.84, indicating the model correctly classified 84% of all instances. This is a promising result, especially in medical diagnostics where high accuracy is critical.
To compare variations of the logistic regression model, a model containing scaled and preprocessed features was compared to a model built on the current dataset, which is unscaled and preprocessed. To select the top significant features were selected based on compute the chi square statistic between each feature of X and Y and selecting the top K features based on a more significant p-value. For testing purposes, K was set to 5, and the top 5 significant features were selected for the model. In addition, a model with scaled data and including all features was tested. However, these variations performed worse than the standard logistic regression model, with the Logistic Regression Top features model returning an AUC of 0.87 and the scaled data logistic regression model returning an AUC of 0.88. The similar performance of the top feature model compared to alternative models indicates that it is likely that a subset of features capture a majority of the information in the model. In addition, these results are similar to Random Forest, where the unscaled data features resulted in a higher accuracy than the scaled features.

ROC Curve and Confusion Matrix Evaluation

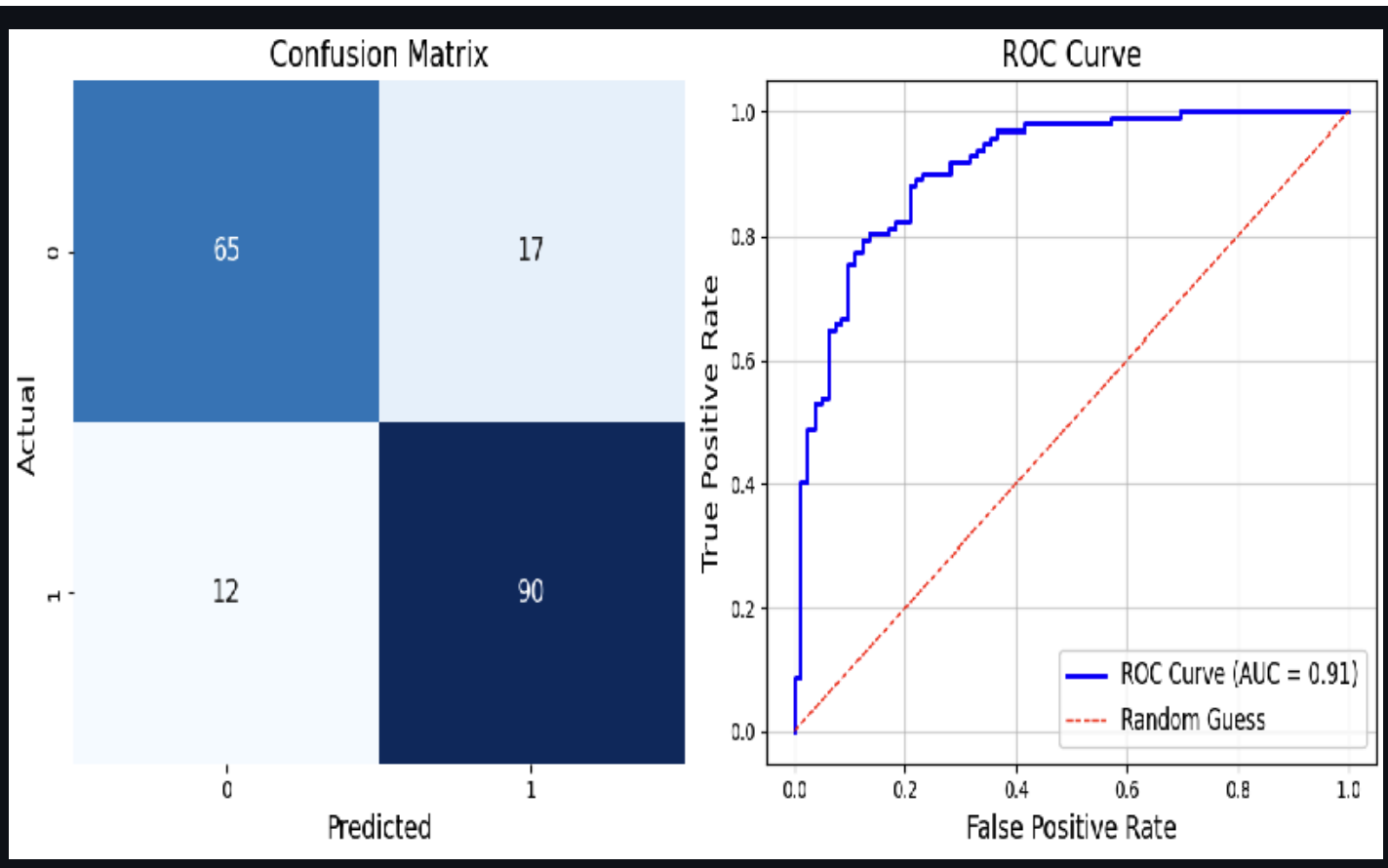


Figure 10. Logistic Regression Confusion Matrix and ROC Curve

The Receiver Operating Characteristic (ROC) curve, a critical tool in evaluating predictive models, provided additional insights into the model's discriminative power by plotting the True Positive Rate against the False Positive Rate across various classification thresholds. The area under our ROC curve (AUC) was calculated as 0.9106, which signifies excellent discriminative ability. An AUC close to 1.0 indicates that the model reliably differentiates between positive (heart disease) and negative (no heart disease) cases.

The ROC curve also reveals how sensitivity (ability to detect true positives) and specificity (ability to avoid false positives) change as the decision threshold is adjusted. For this heart disease prediction model, the curve's proximity to the top-left corner suggests an effective trade-off, minimizing both false positives and false negatives. This balance is essential in a medical context where misclassifications have significant consequences. In this case, with False Negatives (FN), missing a heart disease diagnosis could delay treatment, posing severe health risks, and False Positives (FP) could lead to misclassifying healthy patients, leading to unnecessary tests and anxiety.

The confusion matrix provided additional insights: True Positives (TP): 90 instances of heart disease correctly predicted. True Negatives (TN): 65 instances of no heart disease correctly predicted. False Positives (FP): 17 cases incorrectly classified as heart disease. False Negatives (FN): 12 cases incorrectly classified as no heart disease. These values indicate that the model has a strong balance between sensitivity and specificity, minimizing both false negatives and false positives.

The strong ROC curve performance and confusion matrix scores reflect the quality and separability of the features in the dataset. The dataset likely includes well-chosen variables that strongly correlate with the target variable (heart disease). There is also minimal overlap between the distributions of positive and negative classes, allowing the model to distinguish them effectively. The ROC curve performance suggests that the dataset’s features provide significant discriminatory power.

Several key observations emerged from the analysis. The unexpected superiority of the unscaled preprocessing approach suggests that original feature distributions contain meaningful information that might be lost through traditional scaling techniques. The balanced performance across both classes indicates a robust model with minimal class imbalance impact, pointing to effective feature selection and training methodologies.

Random Forest

Training Outcomes

Class	Precision	Recall	F1-Score	Support
0	0.85	0.80	0.82	82
1	0.85	0.88	0.87	102

Table 2. Random Forest Model Performance by Class

Metric	Value
Accuracy	0.8478
Precision	0.8478
Recall	0.8478
F1-Score	0.8474
ROC-AUC	0.9122

Table 3. Random Forest Overall Model Performance

Figure 11. Random Forest ROC Curve

Our Random Forest model achieved an accuracy of 84.78%, demonstrating an overall satisfactory performance. The model's precision, recall, and F1-score were consistent across both prediction classes, which shows that it is generally robust in handling the class imbalance. An ROC-AUC score (measuring the model's ability to distinguish between the two classes) of 90.75% indicates that the Random Forest model has good discriminatory power.

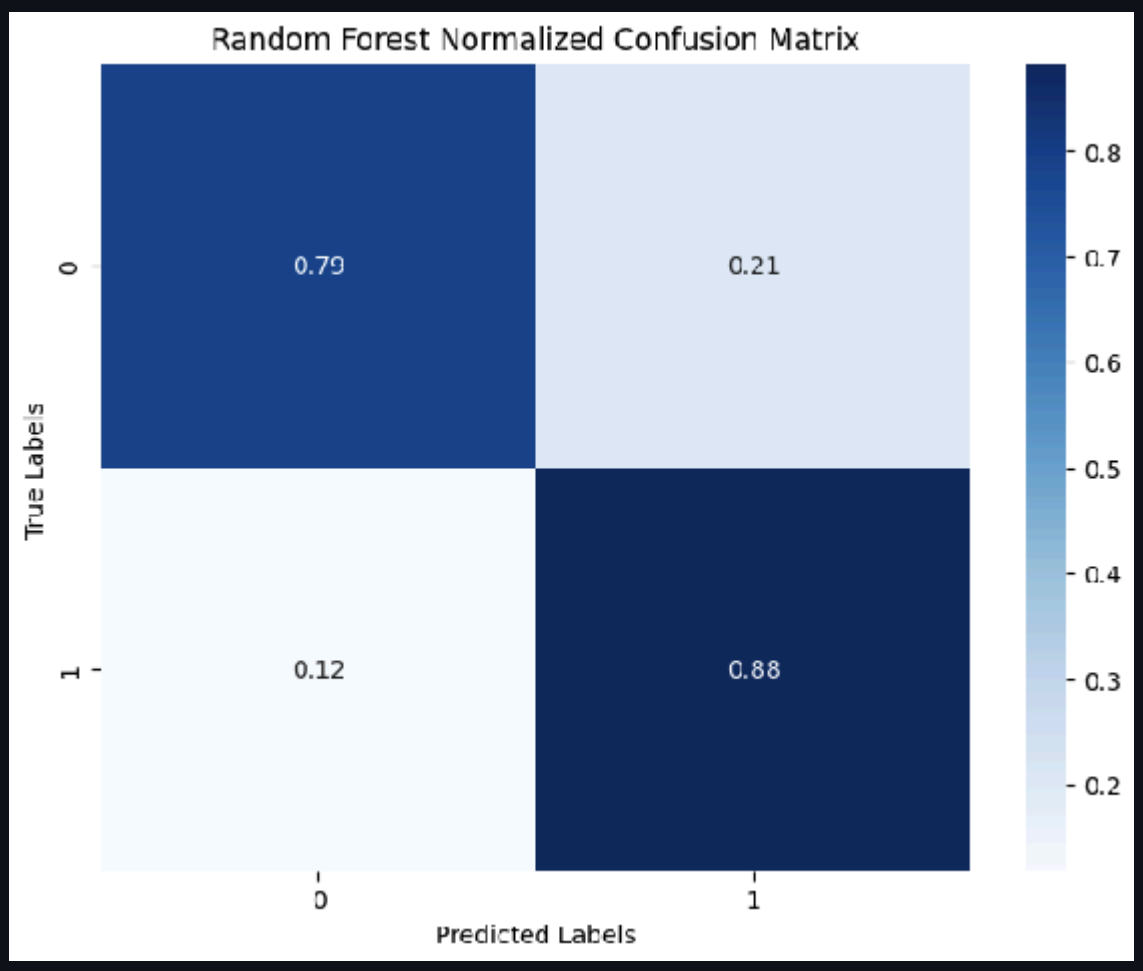


Figure 12. Random Forest Normalized Confusion Matrix

According to the normalized confusion matrix, For Class 0 (No Heart Disease), the model correctly classified 80% of instances as not having heart disease. However, it misclassified 20% of these cases as having heart disease. This false-positive rate, while moderate, suggests that the model occasionally raises false alarms, which could lead to unnecessary diagnostic tests for healthy individuals. For Class 1 (Heart Disease), the model shows stronger performance by correctly identifying 88% of the cases. The remaining 12% of actual heart disease cases were misclassified as healthy, representing the false-negative rate. The higher accuracy for Class 1 suggests that the model might be slightly biased toward predicting heart disease. This is possibly due to the dataset used for training having more instances of Class 1 (407) compared to Class 0 (329), the model could learn to favor predicting heart disease to achieve a higher overall accuracy.

Feature Importance

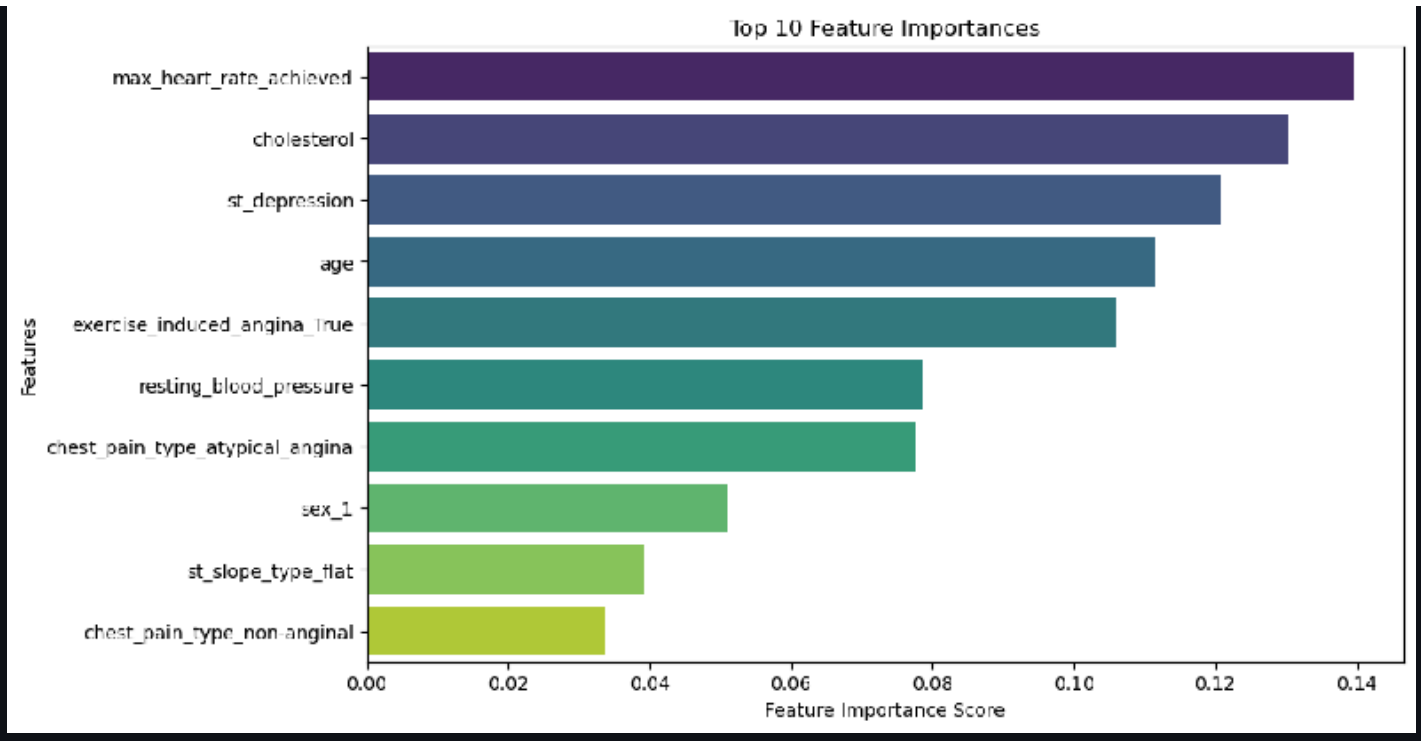


Figure 13. Random Forest Top 10 Feature Importances

In Random Forest, feature importance quantifies how much each feature reduces the impurity in the dataset during the decision-making process within the trees. A feature is considered important if it is frequently chosen for splits in the trees and significantly improves the accuracy of predictions. We plotted out the Top 10 important features from our trained model.

Among the features, max_heart_rate_achieved is the most significant predictor of heart disease. This makes sense as a person's maximum heart rate is often indicative of cardiovascular fitness and potential problems. The next most important factor is cholesterol, which is closely related to heart disease risk, as high cholesterol levels are known to contribute to problems like blocked arteries. The third most important feature, st_depression, reflects changes in the ST segment of an ECG during exercise, which is a strong indicator of heart stress and potential issues like reduced blood flow to the heart. This makes it relevant for detecting heart disease. Age also ranks as a key feature, which makes sense since the risk of heart disease increases with age.

This analysis shows how our Random Forest model uses a mix of physical, demographic, and clinical factors to predict heart disease. The importance of these features matches medical understanding, which adds to the model's credibility and makes it easier to interpret.

K-Means Clustering

Training for K-means Model

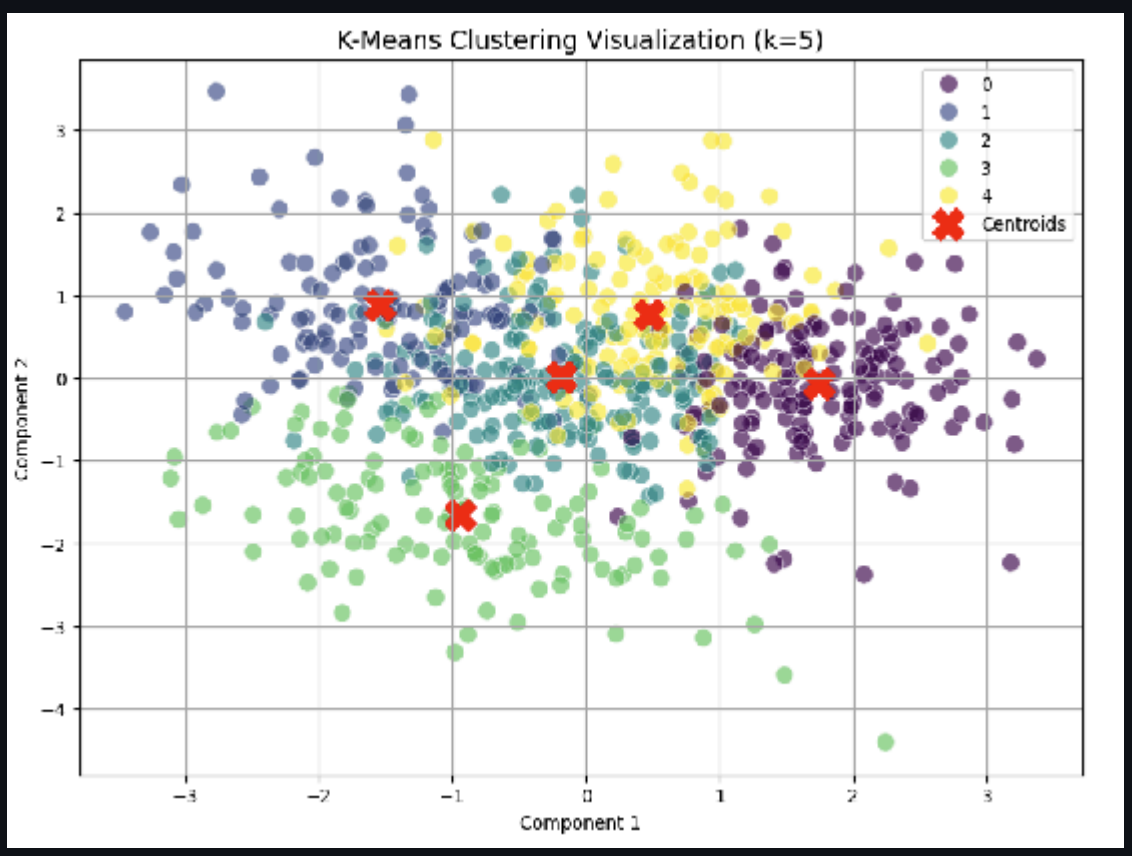


Figure 14. K-Means Clustering on Training Data

Silhouette Score	**0.1417**
Davies-Bouldin Index	1.8010
Calinski-Harabasz Index:	121.1728

Table 4. K-means Clustering Metrics on Training Data

The performance of the model on the training data showed a comparable performance to the evaluation on the test dataset.

Testing K-means Model on test data

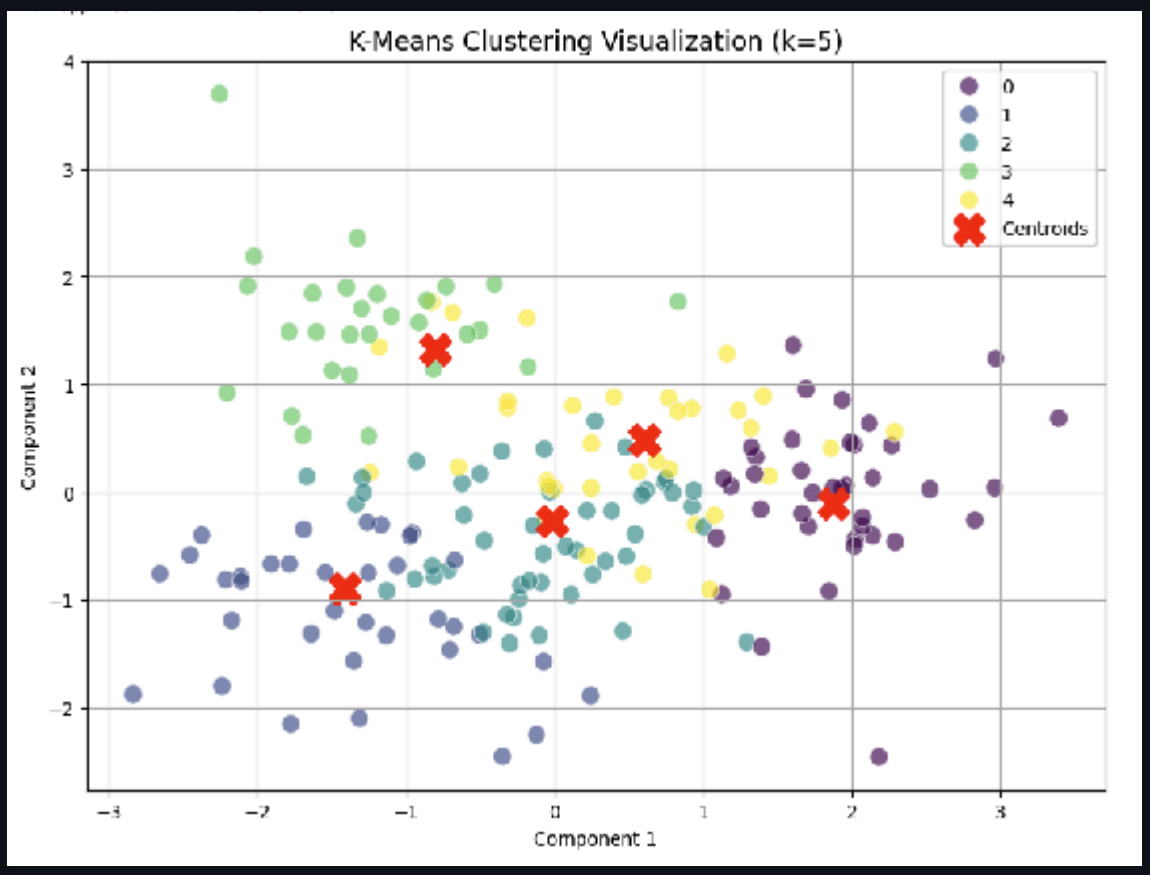


Figure 15. K-Means Clustering on Test Data

K-means was tested on 20% of the data which the model was not trained on. The performance on the test data was comparable to performance on the training data, and showed a slight improvement with respect to the silhouette score.

Silhouette Score	**0.1526**
Davies-Bouldin Index	1.8098
Calinski-Harabasz Index:	32.3768

Table 5. K-means Clustering Metrics on Test Data

Analysis of K-Means

We chose K-means clustering for this dataset because it can help uncover underlying patterns in cardiovascular health by segmenting the data into groups based on similarity. This unsupervised learning approach is useful for revealing trends among patients without relying on predefined labels, which is beneficial in health data where complex, non-linear relationships often exist. By applying K-means, we can identify clusters of patients with similar risk factors and health metrics, such as age, cholesterol levels, and blood pressure. This information can be useful for identifying patient profiles that may share a similar risk of heart disease. Furthermore, K-means clustering is computationally efficient, making it practical for datasets like this one, allowing for quick insights and the potential to iteratively adjust cluster numbers for more accurate representations of patient segments.
Our graph shows clusters of patients with similar risk factors and the associated category (0-4) of heart disease they fall under, with “0” indicating no heart disease and “4” indicating extreme heart disease. The graph shows overlaps of data points across clusters, indicating that the clustering model struggles to isolate distinct groups. This could be due to the fact that there aren’t necessarily a couple of distinct factors that lead to each person getting heart disease, but rather a combination of many different factors that varies from person to person, leading to clustering based on hard limits difficult to create.

Silhouette Coefficient

Scoring cluster separation can be based on the silhouette score. For the silhouette score, the variable *a* represents the mean distance between the sample and all other points in the same class. While the variable *b* represents the mean distance between the sample and all other points in the nearest cluster. To

optimize the performance of K-means, b should be maximized while a should be minimized, and the denominator will normalize the silhouette score.
s=b-amax(a,b)

We used internal evaluation metrics to assess performance and limitations of our clustering model. In K-means, a high Silhouette Score (closer to 1) would indicate well-separated clusters, whereas a low score indicates significant overlap. For our model, the Silhouette Score calculated was 0.1417, which suggests very weak clustering. This score implies that many data points lie near the boundary between clusters and that model is unable to form clearly defined groups. Additional evaluation through the Davies-Bouldin Index (1.8010) and the Calinski-Harabasz Index (121.1728) indicates that the clustering is not as well-distinguished as we would like. This could be due to the underlying structure of our data distributions. Even with uncorrelated principal components, the data distributions may still overlap in the reduced feature space. Medical data often has overlapping patterns because health attributes can be similar across different patient groups, leading to clusters that are not distinct enough.

Confusion Matrix

To further assess the effectiveness of our clustering model, we mapped clusters to the ground truth labels using the Hungarian algorithm, which optimally assigns each cluster to the closest true label. Our dataset has true labels “0” to ”4”, with “0” indicating no heart disease and “4” being extreme heart disease. The resulting confusion matrix indicates significant misalignment.

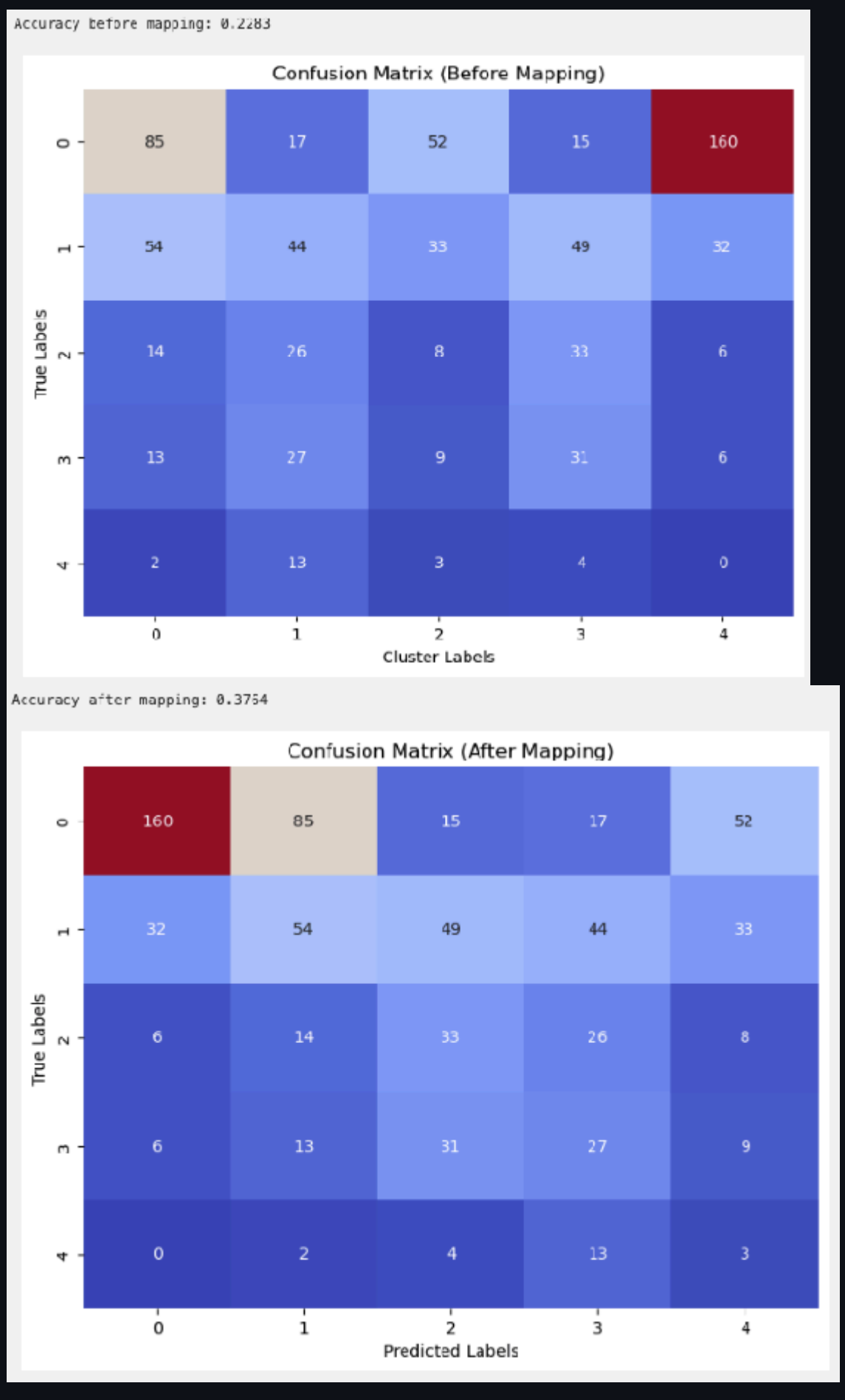


Figure 16. K-Means Confusion Matrix Before vs. After Mapping Clusters

The confusion matrices above indicate that the model performance has low accuracy. It is observed that significant misclassification happens across labels. Specifically, for label “0”, we have 85 instances misclassified as “1” and 52 instances misclassified as “4”. The prediction of “1”s were also spread across all 5 labels. Furthermore, only three instances of “4”s were correctly classified. Although it is expected that unsupervised models like k-means do not perform well on classification tasks, we have a few reasons as to why this happened:

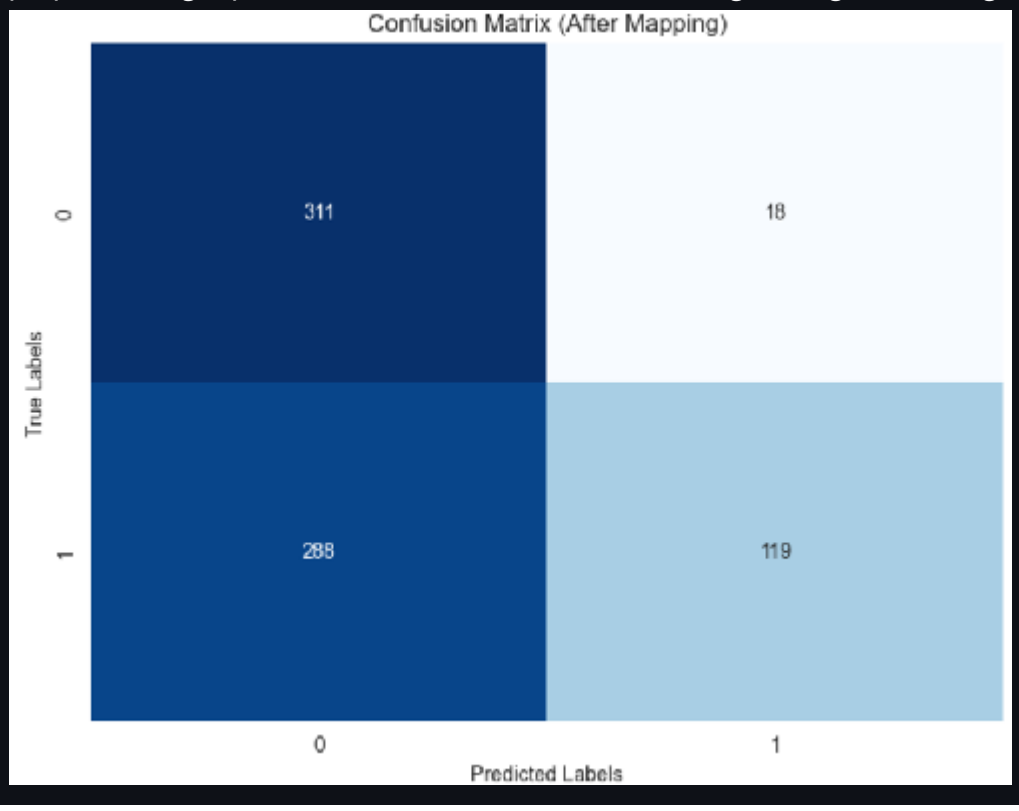
1. K-means calculates the euclidean distance between data points and each feature contributes to the distance metrics separately. This means that when dealing with medical data where features might be informative when considered together, K-means will fail to assign them to the correct cluster, or will find incorrect clusters completely.
2. From our K-means graph shown above, there exists significant overlap between clusters. This will confuse the K-means model and lead to poor performance. This overlap caused some labels like “1”s to have their prediction spread across multiple labels with roughly the same counts.
3. The dataset has too many features with not enough data samples. There are 9 features and around 700 data samples, which is insufficient for K-means to work effectively. Furthermore, the dataset does not have enough data samples for each true label. Labels like “4” have only 22 samples in the data set. For other labels such as “0”, we have an overwhelming amount of 329 samples with this label in our dataset, which is nearly 44% of the dataset. This results in significant low recall for “0”s at 0.486 and low precision for “4”s at 0.027. This class imbalance has caused over-representation for labels such as “4”, where the model misclassified significant amounts of “0” and “1” samples to be “4”s.

Implications

The implications behind these results suggest that it could be difficult to predict heart disease based on this dataset with Kmeans. The poor confusion matrix scores indicate that either the features we have in our dataset do not accurately predict heart disease, or K-means fails to capture the vital information to make correct predictions. This indicates that heart disease may not be caused by a specific combination of 3 or 4 attributes, but rather a vastly different cocktail of indicators that varies from person to person. In addition, this lack of predictive power raises the possibility that the features available in the dataset, such as age, blood pressure, cholesterol levels, and other lifestyle factors, may not fully capture the complexity of heart disease etiology. This could imply that heart disease is influenced more by factors not included in the dataset, like genetic predispositions, stress levels, inflammation markers, or detailed lifestyle data that are harder to quantify, such as dietary habits or psychosocial factors. The poor performance of the model may also indicate a need for a more comprehensive dataset in order for K-means to access more training data to be able to better cluster the data points. The findings emphasize that predicting heart disease likely requires a more expansive multi-dimensional approach to capture the full spectrum of risk factors and individual health profiles.

Kmeans Updates

Since we changed how we preprocessed the data and applied {0,1} labels for the supervised model after we had initially analyzed the Kmeans performance, an update of Kmeans performance using the updated preprocessing is provided here where the left is training and right is testing:



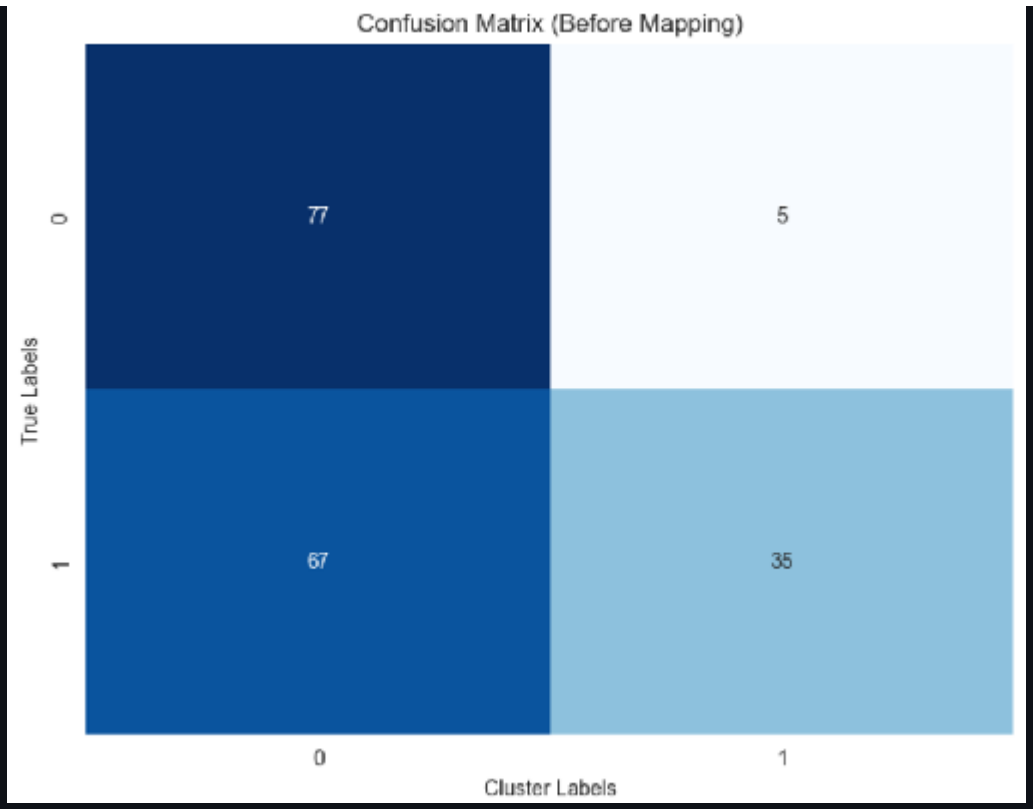


Figure 17. Updated K-Means Confusion Matrix Training vs. Testing

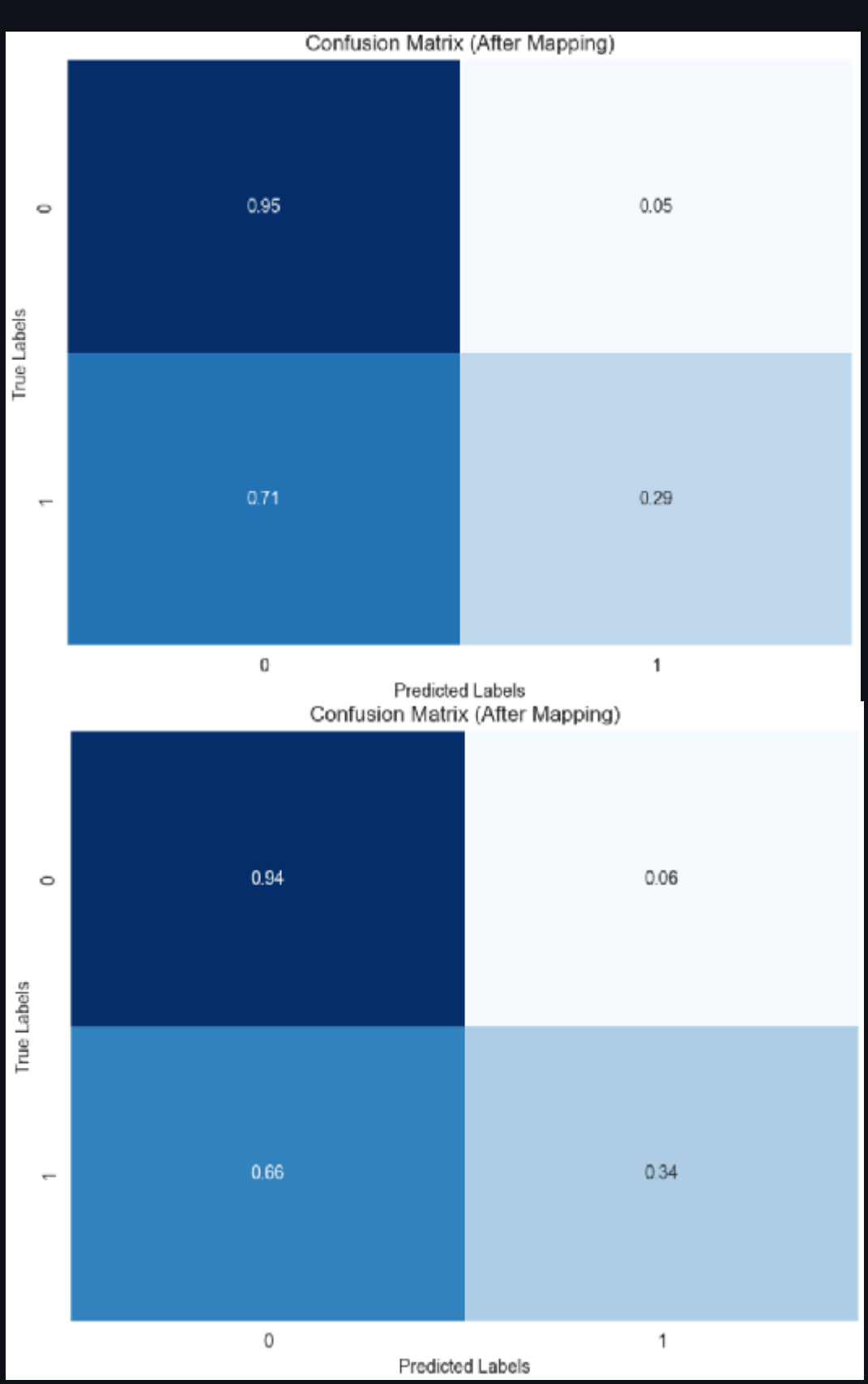


Figure 18. Updated K-Means Normalized Confusion Matrix Training vs. Testing

Overall, after updating to our new preprocessing and switching from severity labeling to positive labeling, the accuracy of Kmeans performance increased to 60.87%. We can observe from the graph that although we have high True negative but low True positive. This suggests that the model has poor separation in feature space and fails to generalize well for positive cases. This is expected for our dataset in which the “positive” cases are a combination of different severity of heart disease and they tend to differ at several features.

Model Comparison

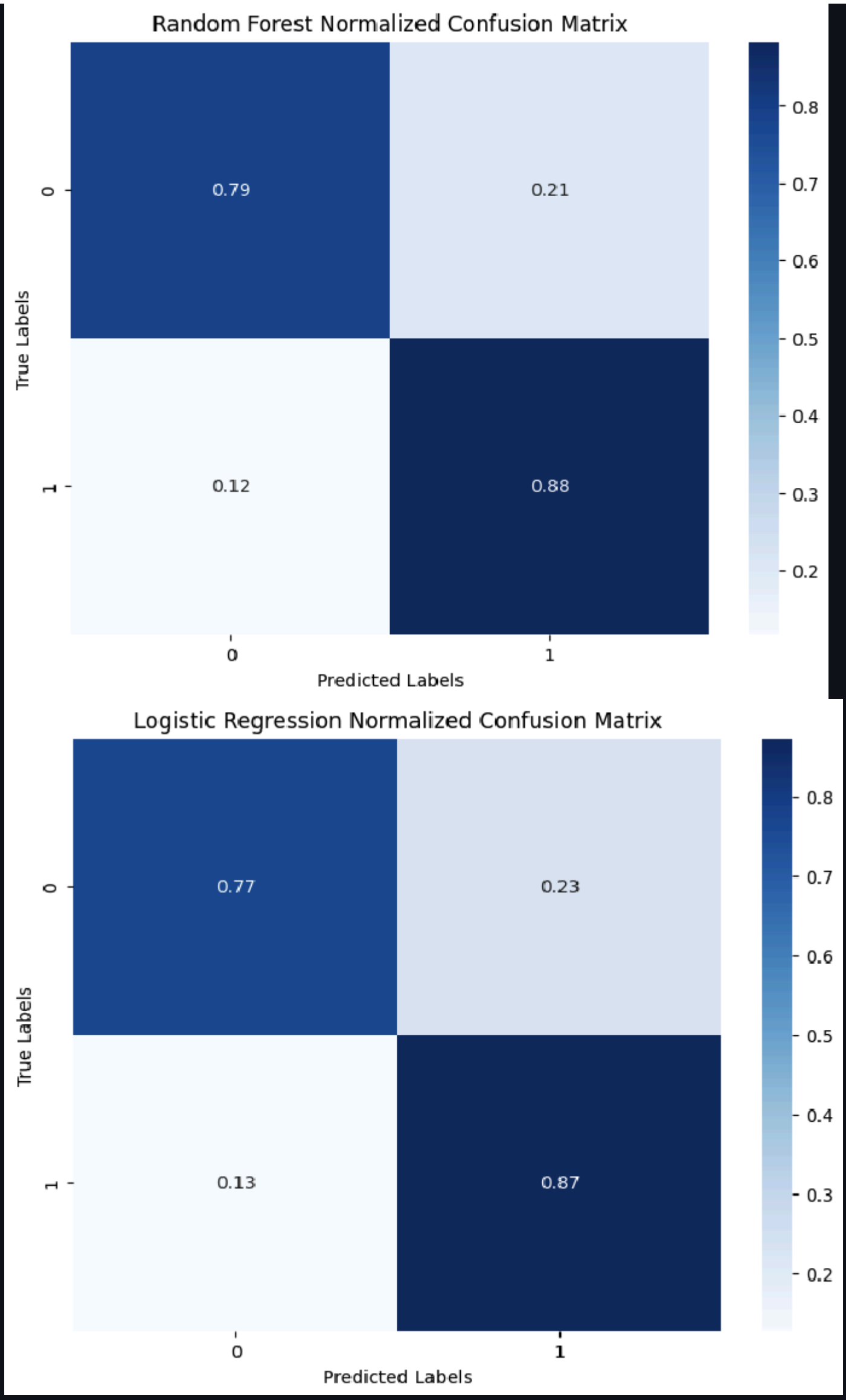
Accuracy and Confusion Matrix

The overall accuracy was comparable between the Logistic Regression and Random Forest models, as these two predicted labels from our datatset. As shown in Figure 18, the ROC curve scores were very similar, with Logistic Regression just having a marginally better score (+0.002). The confusion matrices in Figure 18 also show both models having nearly the same performance, with Random Forest performing just slightly better (+0.1-0.2) this time. The accuracy results showed comparable performances for both models.

Overall, the recall, precision, accuracy and F1 score show a very minor reduction for Logistic Regression when compared to the performance of Random Forest. This can be a result of the tuning process applied to the Random Forest, leading to slight improvement on accuracy scores, including F1. The ROC AUC showed a very minor improvement for Logistic regression when compared to the Random Forest model. There is a relatively lower level of noise and a reduced amount of missing values across the UCI Heart disease dataset compared to larger datasets, making it ideal for logistic regression. In addition, logistic regression performs well on medical datasets when a highly interpretable model is needed for clinical data. Due to these factors and the smaller size and number of features in the dataset, logistic regression slightly outperforms Random Forest in terms of ROC. However, the F1 metric along with accuracy is lower for Logistic Regression. A high ROC value for Logistic Regression indicates good overall performance of the model. However, at specific thresholds, for example with regard to F1, the Regression model is not performing as well, potentially due to class imbalances or skewed feature distributions. To adjust for this, logistic regression models containing scaled data features, as well as only the top five contributing features were selected. However, these two model variations showed a lower AUC value than a Logistic Regression model containing all features without scaling, as demonstrated in Figure 18.

Model	Recall	Precision	Accuracy	F1 Score	ROC AUC
Random Forest	0.8478	0.8478	0.8478	0.8474	0.9122
Logistic Regression	0.8424	0.8425	0.8424	0.8418	0.9142

Table 6. Random Forest vs. Logistic Regression Model Performance


Random Forest Normalized Confusion Matrix
0.79
0.21
0.12
0.88
True Labels
0
1
0
1
Predicted Labels
0.8
0.7
0.6
0.5
0.4
0.3
0.2
Logistic Regression Normalized Confusion Matrix
0.77
0.23
0.13
0.87
True Labels
0
1
0
1
Predicted Labels
0.8
0.7
0.6
0.5
0.4
0.3
0.2

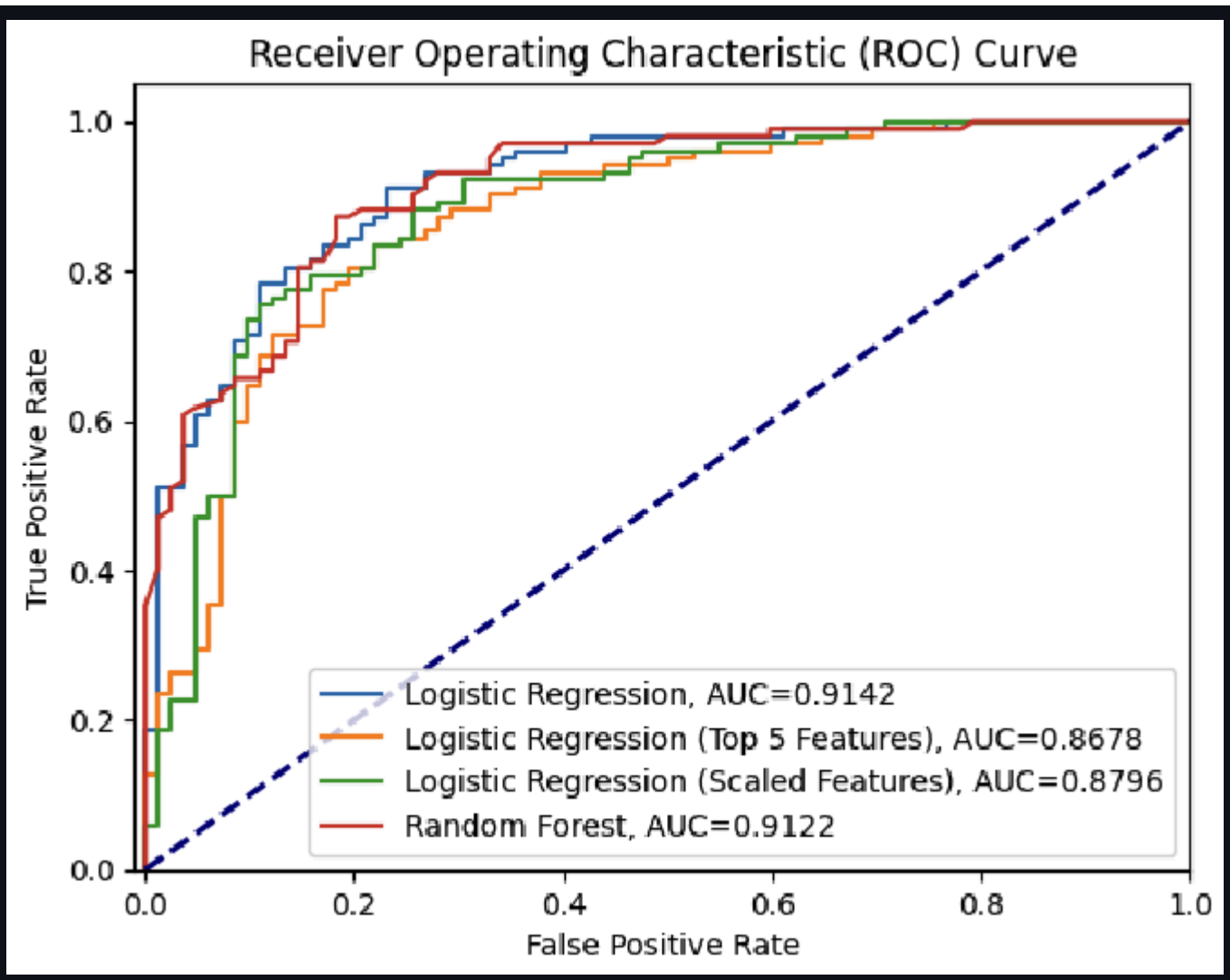


Figure 19. Random Forest vs. Logistic Regression Normalized Confusion Matrix and ROC Curves.

In summary, the variations of the Logistic regression model, including a model built with the Top 5 significant features and a model built with scaled preprocessed data showed reduced overall performance compared to the logistic regression model built with all features and unscaled. Both the random forest model with unscaled preprocessed data and the logistic regression model with unscaled preprocessed data demonstrated the highest overall AUC score and performance for heart disease prediction.

Speed of Execution

The speed of execution showed that the Logistic Regression model was faster than Random Forest and K-means on non-scaled and preprocessed data. This result was expected, as Logistic Regression which is based on a linear model and has a simpler model architecture than Random Forest, which trains the model across numerous decision trees. K-means can also be computationally intensive for highly dimensional data. Overall, logistic regression takes the lowest amount of computational time for training on the same dataset.

Model Architecture	User Time	Sys Time	Total Time	Wall Time
Random Forest (200 Trees)	167 ms	49.8 ms	217 ms	154 ms
Logistic Regression	75.1 ms	3.54 ms	78.7 ms	82.8 ms
Kmeans	328ms	203ms	531ms	88.7ms

Table 7. Random Forest vs. Logistic Regression vs Kmeans Speed of Execution

Next Steps

The models developed in this study hold significant potential for future clinical applications, particularly in the early diagnosis and prevention of heart disease. Hospital data systems have made patient data more readily available for machine learning models. The model weights developed here and the optimized model workflow customized for clinical data predictions can be applied to patient data related to heart disease in the future. Based on our analysis, several areas for improvement can be explored.

Dataset Enhancement

To enhance the models' performance, it would be helpful to improve the quality and diversity of the dataset. The dataset used in this study includes 920 patient records with 14 attributes. However, the

limited scope of these features, such as the exclusion of genetic markers, stress levels, and detailed lifestyle data, may not fully capture the complexity of heart disease. Expanding the dataset to include more variables and samples, as well as addressing the class imbalance noted in the current data (e.g., the overwhelming representation of Class 0), could significantly improve the models' performance.

Model Improvements

As shown in the Random Forest and Logistic Regression sections, hyperparameter tuning has already demonstrated the ability to enhance model performance. Building on this, advanced techniques such as Bayesian optimization or Ensemble methods like stacking could further improve accuracy and robustness. The limitations of the K-Means clustering model, as discussed earlier, shows the need for alternative approaches. The low silhouette score and the overlapping clusters indicate that the Euclidean distance metric used may not be the most suitable for this dataset. Exploring metrics such as Mahalanobis distance or cosine similarity, which account for feature interdependencies, could give more meaningful clustering. Additionally, exploring other unsupervised learning approaches, such as DBSCAN or hierarchical clustering could possibly uncover more meaningful patterns in the data and enhance model capability.

Real World Application

Applying these models in real-world settings is an essential next step. Developing a prototype application to test the models on live patient data would validate their utility and provide feedback for further refinement.
By focusing on these strategies, the methods and findings in this report can be extended to create more effective machine learning applications for cardiovascular health.

Conclusion

This study explored the potential of machine learning models for the prediction and diagnosis of heart disease using the UCI Heart Disease dataset. Through rigorous preprocessing, feature engineering, and algorithm selection, we analyzed and compared the performance of unsupervised and supervised learning approaches, including K-Means clustering, Logistic Regression, and Random Forest.
Our findings show that supervised models outperform unsupervised clustering for this dataset. Logistic Regression and Random Forest demonstrated strong predictive performance, achieving accuracies of 84% and 84.78% respectively. Both models showed good discriminative ability. From Random Forest, features such as maximum heart rate, cholesterol levels, and ST depression were identified as significant predictors of heart disease. In contrast, K-Means clustering initially struggled with poor performance in the multiclass setting, but its performance improved when we adjusted the preprocessing and simplified the task to binary classification, achieving an accuracy of 60.87%.
Apart from the encouraging results for the supervised models, our study also analyzed key limitations, including the dataset's narrow feature scope and class imbalance. Broader datasets that include additional variables are needed to enhance model reliability. Future work could focus on integrating more advanced preprocessing techniques, expanding the feature set, and using hybrid modeling approaches to further improve robustness.
In summary, our project demonstrates the promise of machine learning in improving heart disease prediction, and emphasizes the importance of comprehensive datasets and tailored modeling strategies for healthcare.

References

1. Krittanawong, Chayakrit, et al. "Artificial intelligence in precision cardiovascular medicine." Journal of the American College of Cardiology 69.21 (2017): 2657-2664.
2. Al'Aref, Subhi J., et al. "Clinical applications of machine learning in cardiovascular disease and its relevance to cardiac imaging." European heart journal 40.24 (2019): 1975-1986.
3. J. Abdollahi and B. Nouri-Moghaddam, "A hybrid method for heart disease diagnosis utilizing feature selection based ensemble classifier model generation," *Iran Journal of Computer Science*, vol. 5, pp. 229-246, Sep. 2022, doi: 10.1007/s42044-022-00104-x.
4. R. Detrano, A. Janosi, W. Steinbrunn, M. Pfisterer, J. J. Schmid, S. Sandhu, and K. H. Guppy, "International application of a new probability algorithm for the diagnosis of coronary artery disease," *American Journal of Cardiology*, vol. 64, no. 5, pp. 304-310, 1989.
5. C. S. Dangare and S. S. Apte, "Improved study of heart disease prediction system using data mining classification techniques," *International Journal of Computer Applications*, vol. 47, no. 10, pp. 44-48, 2012.

Contribution Table

Name	Contributions
Josh	- Logistic Regression Code - Logistic Regression Preprocessing - Logistic Regression & Comparison Report Section
YuChen	- Random Forest Hypertuning - Visualization - Kmeans Refurbish - Comparison & Random Forest Report Section
Annie	- Random Forest Code - Visualization - Random Forest, Next Steps & Conclusion Report Section
Theerat	- Random Forest Code - Random Forest Preprocessing - Preprocessing Report Section
Faiza	- Logistic Regression Code - Logistic Regression Preprocessing - Logistic Regression Report Section - Presentation Slides and Video Recording

Gantt Chart

GANTT Chart Link