Heart Disease Prediction

1st Phillip Pondo
Department of Computer Science
Stevens Institute of Technology
Rutherford NJ, USA
ppondo1@stevens.edu

2nd Jake Shinohara
Department of Computer Science
Stevens Institute of Technology
Hoboken NJ, USA
jshinoha@stevens.edu

3rd Klayton DalPra
Department of Computer Science
Stevens Institute of Technology
Rumson NJ, USA
kdalpra@stevens.edu

***Abstract*—In this project, we aim to predict the presence of heart disease using machine learning algorithms based on patient health data. The original dataset includes both categorical and numerical variables such as age, sex, chest pain type, cholesterol level, blood pressure, and exercise-related metrics. After cleaning the data and encoding categorical features, we applied logistic regression to classify patients as either having or not having heart disease. The model achieved an accuracy of approximately 84 percent on the test set, demonstrating strong performance, especially in identifying non-heart-disease cases. We normalized continuous variables and balanced class weights to address class imbalance and improve convergence. Major contributions of this work includes determining which attributes positively impact the model's performances. We did this by using a mutual information test to determine which features were not helpful for training the models. We also used K-Nearest Neighbors imputation to help with data pre processing. We trained two logistic regression models with different techniques to account for a binary classification and one with a multiclass classification. We created Principal Component Analysis (PCA) Projections to visualize and analyze the performance of models. These graphics give insights on certain models weaknesses. A Random Forest Model was also implemented using the dataset which produced a model with an Area Under the Curve (AUC) score of approximately 89 percent and an accuracy of approximately 84 percent. Lastly, we implemented a Support Vector Machine model which resulted in an accuracy of 85 percent and the lowest false negatives. We used GridSearchCV hyperparameter tuning to achieve these results.**

I. Introduction

Problem Statement: Heart disease is one of the leading causes of death, and early detection can improve the patient outcomes. Medical datasets often contain detailed clinical data collected from patients, including symptoms, test results, and demographics. While many models exist to detect whether a patient has heart disease, few focus on identifying which features should be included in the decision making. In this project, we aim not only to classify patients as having heart disease or not, but also to explore how data can be used and omitted to enhance the accuracy of the diagnostic capability of heart disease from machine learning models.

The data is originated from University of California, Irvine's Heart Disease dataset that was donated on 06/30/1998. UCI indicates that the dataset is created from four databases: Cleveland, Hungary, Switzerland, and the VA Long Beach and consists of approximately 1000 sample patients. Fourteen independent features of interest and the dependent feature of heart disease and it's severity are provided. Our team changed the multiclass severity representation and converted it to a binary class representation to determine if heart disease is present or not, ignoring severity. Two of the independent features were omitted by our team: restecg and fbs, after evaluating their predictive power of heart disease to not be significant enough to influence our models.

The team utilized three Machine Learning Models to help gain insight on the classification of heart disease. The first model used was logistic regression that returned an approximate 84 percent accuracy, 84 percent precision, an F1 score of 0.84, and recall score of 0.84 as well. The second model used was a Random Forest that returned an approximate 89 percent AUC score and 84 percent accuracy. When implementing the random forest, mean decrease in impurity (MDI) was used to consider the importance of each feature for the model and then comparing the amount of added important features vs AUC and accuracy helped determine the best amount of features that should be used in the model. The third model used was a Support Vector Machine (SVM). These three models were chosen because they are known to handle binary classification well.

II. Related Work

The UCI Heart Disease dataset has been extensively analyzed using various machine learning algorithms to predict the presence of heart disease. A review of existing solutions reveals several commonly employed models. These include Logistic Regression, Support Vector Machines (SVM), K-Nearest Neighbors (KNN), Random Forest (RF), Artificial Neural Networks (ANN), and Naive-Bayes Classification (NB), each with distinct advantages and limitations.

Work related to predicting heart disease has demonstrated that LR can achieve commendable accuracy, making it a reliable baseline for heart disease prediction [1]. However, its linear nature may limit performance when dealing with complex, non-linear relationships inherent in medical data.

SVMs, particularly when combined with feature selection techniques like the Jellyfish optimization algorithm, can achieve high accuracy in heart disease prediction. Nonetheless, SVMs can be computationally intensive, and their performance is sensitive to parameter tuning and kernel selection.

KNN has also shown effectiveness in heart disease classification tasks, but its performance can be degraded with high-dimensional data, and it may be computationally expensive due to the need to compute distances.

RF has also been reported to achieve high accuracy in heart disease prediction tasks. The main drawback is its complexity,

which can lead to longer training times and reduced interpretability compared to simpler models.

ANNs are capable of modeling complex, nonlinear relationships and have been applied to heart disease prediction. Although they can capture intricate patterns in data, they require large datasets for effective training and are often considered "black boxes" because of their lack of interpretability.

NB classifiers have also been used in heart disease prediction, but the strong independence assumption is often unrealistic for medical data, potentially limiting the model's accuracy.

Many of the existing solutions use all available data, which can lead to inaccuracies or improper biases. When reviewing existing solutions and related work, it became apparent that feature selection will be critical to improving the accuracy of predictions in this dataset. One study in particular where this was evident was a study title titled "Prediction of Heart Disease Based on Machine Learning Using Jellyfish Optimization Algorithm", which focuses on enhancing heart disease prediction through machine learning techniques. Using the dataset, the researchers aimed to develop an ML model with superior predictive performance. To mitigate overfitting associated with high-dimensional data, they employed the Jellyfish Optimization Algorithm for feature selection, known for its rapid convergence and flexibility in identifying optimal features. Among the four ML models tested (ANN, DT, AdaBoost, and SVM), the Support Vector Machine classifier, trained on the data set refined by the Jellyfish algorithm, achieved the highest performance metrics: Sensitivity of 0.9856, Specificity of 0.9837, Accuracy of 0.9847, and Area Under Curve of 0.9448. These findings suggest that optimizing the selection of characteristics can significantly improve the precision of heart disease predictions.

When determining our approach, we looked closely at how narrowing down and selecting the most relevant features could improve our predictions. We identified helpful and unhelpful attributes regarding proper prediction, adding a new dimension that helps to predict correctly despite the fact that patients have conflicting attributes.

III. Our Solution

Our first approach consists of logistic regression for binary classification (heart disease vs. no heart disease) as a baseline. StandardScaler is used to normalize continuous variables, and categorical features are numerically encoded. Class balancing techniques are applied to ensure the model doesn't bias toward the majority class. Another approach used is a Random Forest model that works well with binary classification. To ensure a model that performs efficiently with good predictions for the classification of our target variable of heart disease prevalence, we can determine Gini importance (mean decrease in impurity) of features and compare several Random Forest models with varying number of features. In future iterations, we plan to enhance our model by omitting features that negatively impact the accuracy. This approach may help the model prioritize learning patterns, which is highly beneficial in real-world clinical decision support.

A. Description of Dataset

We use the UCI Heart Disease dataset, which includes multiple patient records across different study origins. Each entry consists of features such as age, sex, chest pain type, blood pressure, cholesterol, fasting blood sugar, ECG results, maximum heart rate, exercise-induced angina, ST depression, the slope of the ST segment, number of vessels colored by fluoroscopy, and thalassemia test results. The original label (num) represents the stage of heart disease from 0 (no disease) to 4 (most severe). The dataset contains both categorical and numerical attributes, requiring pre-processing such as missing value imputation, encoding, and normalization. The dataset was pre-processed by converting categorical attributes to numerical values.

- **sex:** Male = 0, Female = 1
- **cp (chest pain type):** typical angina = 0, atypical angina = 1, non-anginal = 2, asymptomatic = 3
- **fbs (fasting blood sugar):** False = 0, True = 1
- **restecg (resting ECG):** normal = 0, st-t abnormality = 1, lv hypertrophy = 2
- **exang (exercise-induced angina):** False = 0, True = 1
- **slope (slope of ST segment):** downsloping = 0, flat = 1, upsloping = 2
- **thal:** normal = 0, fixed defect = 1, reversible defect = 2

For missing fields, we utilized Mode Imputation and used the most common value in each column for categorical attributes. For values that were originally numerical, we use Mean Imputation and filled in the missing fields with the average value. These numerical values include:

- **age**: the patient's age in years
- **trestbps**: resting blood pressure (in mm Hg) at admission
- **chol**: serum cholesterol level (in mg/dl)
- **thalch**: maximum heart rate achieved during exercise
- **oldpeak**: ST depression induced by exercise relative to rest (an ECG measure related to ischemia)

When evaluating our categorical features, the Scikit-learn library's mutual info classifier function was utilized to determine each feature's predictive power for our target value of heart disease detection. A thresh hold for a MI score of being less that 0.01 was used to determine if a feature was to be omitted. The only two features below 0.01 were fbs at 0.008 and restecg at 0.005. Therefore, we omitted these features.

For continuous numerical features, correlation coefficients were found for each independent feature's correlation to the target variable. A threshold of a coefficient's absolute value being less than 0.1 was used to determine if a feature should be omitted. All features had a coefficient greater than or equal to 0.1. Next, a correlation matrix was created to determine the independent features correlation to each other. A threshold of

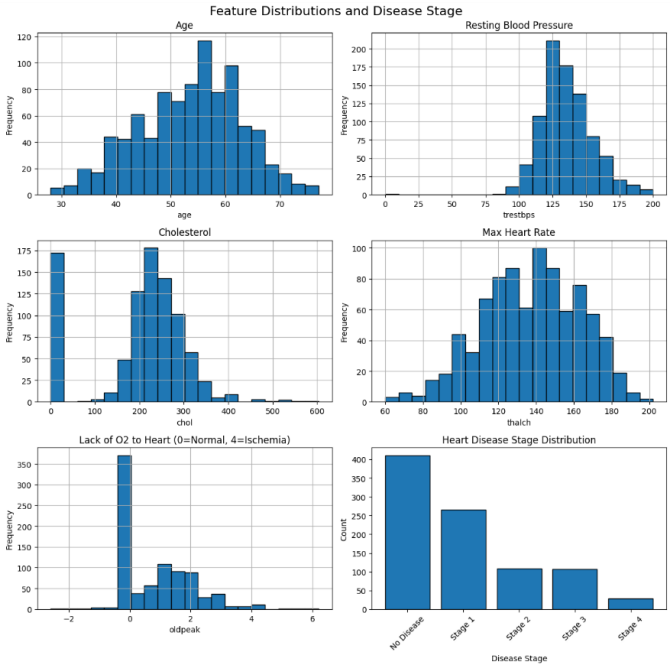

Fig. 1. Dataset Feature Distribution

a coefficient's absolute value being greater than 0.7 was used. If a feature met this threshold, it would be omitted to help prevent overfitting in the models. All features were below the threshold, so none were omitted.

After preprocessing and encoding all features, we applied normalization to the continuous numerical attributes using `StandardScaler` from Scikit-learn. This step standardizes features by removing the mean and scaling to unit variance (mean = 0, standard deviation = 1). It is especially important for gradient-based models like logistic regression, as it helps ensure faster convergence and avoids dominance by attributes with larger numerical ranges.

The following continuous features were scaled: age, trestbps, chol, thalch, oldpeak. Standardization was performed after splitting the dataset into training and testing sets. The scaler was fit only on the training data and then applied to both training and test sets to avoid data leakage.

B. Machine Learning Algorithms

We chose logistic regression as our baseline classifier due to its interpretability and speed. It is a supervised learning model and suitable for binary classification and can also be adapted for multiclass situations. Since our goal is to predict heart disease, and our dataset provides us with patients heart disease stages, we can implement both versions of logistic regression.

A Random Forest model was used as well. This model was chosen because it is well known for handling binary classifiers well. For the amount of n estimators for the Random Forrest, it was set to a limit of 100 since the dataset consisted of approximatley 1000 samples in order to reduce overfitting of the model. Mean decrease in impurity (MDI) was calculated to consider the importance of each feature for the model. Multiple Random Forest models were then trained with an incremental amount of the ranked features,adding the most important feature first and so on, and the performance of each model vs the amount of features used was evaluated.

Finally, a Support Vector Machine (SVM) model was also implemented to predict the presence of heart disease using the clinical dataset. The dataset was preprocessed as mentioned above, evaluated using various kernels, and further analyzed using an ROC curve, classification scores, and hyperparameter tuning via grid search. The Radial Basis Function (RBF) kernel with a balanced class weight was used in the initial implementation. After assessing its performance metrics, it was compared to performance of SVMs with Linear and Polynomial degree 3 kernels. Finally, a GridSearch was used in order to try to identify which hyperparameters of the baseline implementation (RBF kernel) could be tuned to improve results.

C. Implementation Details

1) Logistic Regression: We split the dataset into 80 percent training and 20 percent testing sets with stratification to preserve class distribution. Feature normalization was applied only to continuous variables to ensure that gradient-based learning converges efficiently. We also used StandardScaler to scale the numerical values so that the columns have a mean of 0 and standard deviation of 1. This helps with making the data more uniform and assists with convergence of the models. We used Scikit-learn's logistic regression with up to 5000 iterations to ensure convergence. To improve performance on imbalanced data, we applied the class_weight='balanced' option, which adjusts the learning based on class frequency. We also used different solver methods based on the goal of the model. We used liblinear for binary classification and lbfgs for multiclass.

A crucial debugging step included removing irrelevant, or low impact columns. Columns that scored very low on our Mutual Information test were dropped and improved scores by a percentage or two. Removing non predictive columns like the 'id' field drastically impacted the performance of the model. Accidentally including this column during model training created artificial patterns.

Another aspect I tested was the effect of regularization. In logistic regression, the C parameter controls the strength of regularization, which helps prevent overfitting by penalizing large coefficients. Lower C values apply stronger regularization, while higher C values reduce the penalty. I tested C values of [0.01, 0.1, 1, 10, 100] and plotted the resulting accuracy for each. For the binary model, the differences were negligible and did not improve any metrics. However, for the multiclass model, a C value of 1 produced slightly better metrics as compared to a value of 0.01.

The final optimization technique I implemented for this model was using the K-Nearest Neighbors algorithm for data pre processing. Instead of filling in the missing values in the data set with the most frequent option from each column, I used the KNN algorithm to find similar patients based on other features [2]. I then used the data from the similar patients to replace the missing entry. This provides the model with

more realistic information, as the average value might not be appropriate for each patient.

Model performance was evaluated using accuracy, precision, recall, F1-score, and confusion matrix. We also used bar plots to visualize precision, recall, and F1-score per class. These evaluations helped identify areas where the model was under performing, particularly in attempting to detect the specific stage of heart disease. A PCA projection helped show that the multiclass logistic regression model was only able to predict three out of the five stages. This was due to the data not having any clear separation between classes 2 and 3.

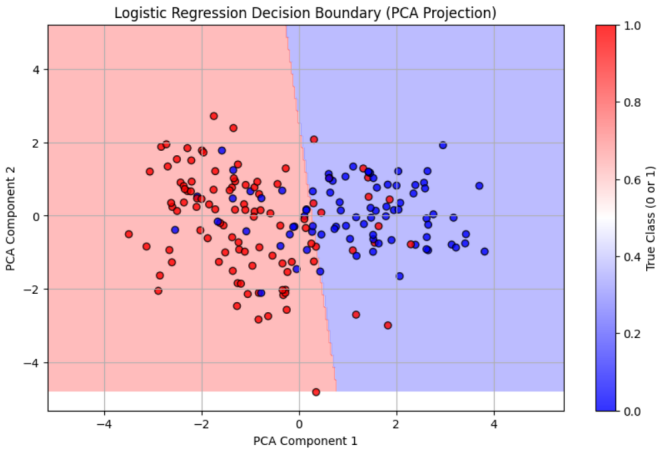

Fig. 2. PCA Projection Binary Logistic Regression

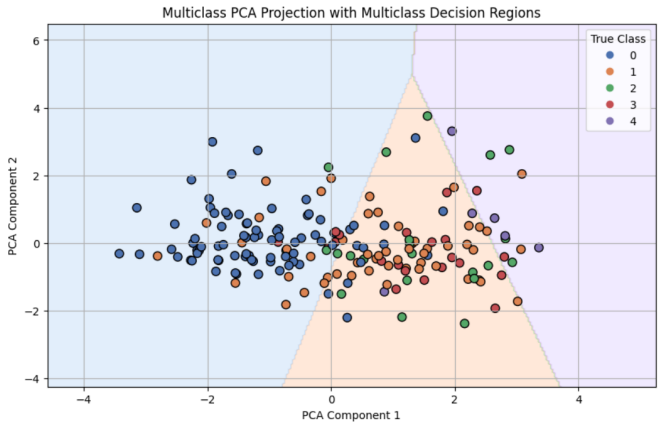

Fig. 3. PCA Projection Multiclass Logistic Regression

2) Random Forest: For the Random Forest Model, the features ranked based on importance were incrementally placed into separate models to evaluate the performance of each model vs the amount of features. For example, the first tested model contained only one feature, the feature with the highest importance ranking, then the second model used the highest and second highest ranked feature and so on until 13 Random Forest models were evaluated. Performance for the models were based on Area Under the Curve (AUC) calculations and Accuracy for each respective model. AUC and accuracy vs Number of Features to help visualize what amount of features were best for the model. Both AUC and accuracy start to plateau at seven features which is a good indication for the best number of features that should be used as selecting too many additional features for a minimal percent increase in AUC and accuracy can lead to overfitting in the model. The Random Forest model utilizing the top seven highest ranked features returned an AUC score of approximately 89 percent and an accuracy of approximately 84 percent.

A confusion matrix for the selected Random Forest model was created to further the understanding of the model and the incorrect predictions it was making. Out of 184 predictions, 30 predictions were incorrect with 12 predictions being a false positive (predicted the test sample had heart disease when they actually did not) and 18 predictions being a false negative (predicted the test sample did not have heart disease when they actually did).

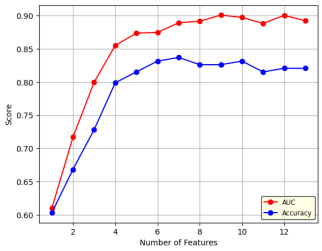

Fig. 4. AUC and Accuracy vs Number of Features for Random Forest

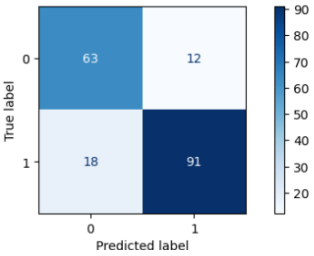

Fig. 5. Confusion Matrix for Random Forest Model with 7 features

3) Support Vector Machine: As in the other two approaches, the first step was cleaning and preparing the data from the dataset. Missing values for numeric columns are handled using the median, which we chose to offset the presence of potential outliers. For categorical columns such as cp, slope, thal, sex, and exang, missing values are imputed using the mode. This distinction between numeric and categorical treatment emerged through iterative testing, as we found improperly imputed categorical variables could break downstream encoding steps. The target column num was converted into a binary classification problem, where any presence of heart disease is labeled as 1, and absence as 0. Several categorical variables were then mapped into numeric representations using domain-appropriate dictionaries. These mappings are crucial for compatibility with SVMs, which require purely numeric input. The dataset, fbs, and restecg columns were dropped due to exploratory testing that revealed minimal or redundant contribution to prediction.

Following preprocessing, the data is split into training and testing sets with stratification on the target variable to preserve class balance. Feature scaling using StandardScaler is applied as Support Vector Machines (SVMs) are sensitive to feature magnitudes. We determined that the scaling should be applied

after the split. This was to ensure that no information from the test set leaks into the training process, which would artificially inflate performance.

A Principal Component Analysis (PCA) was used as a visualization tool to project the high-dimensional training data into two dimensions, and to provide feedback on how distinguishable the classes are. The two principal components capture a visible, though not perfectly clean, separation between the two classes (0 = no heart disease, 1 = heart disease), indicating that our feature engineering preserved enough structure in the data to make the classes at least partially distinguishable. While this doesn't directly impact model performance, it was valuable in affirming the decision to try a non-linear kernel, as linear separability appears limited.

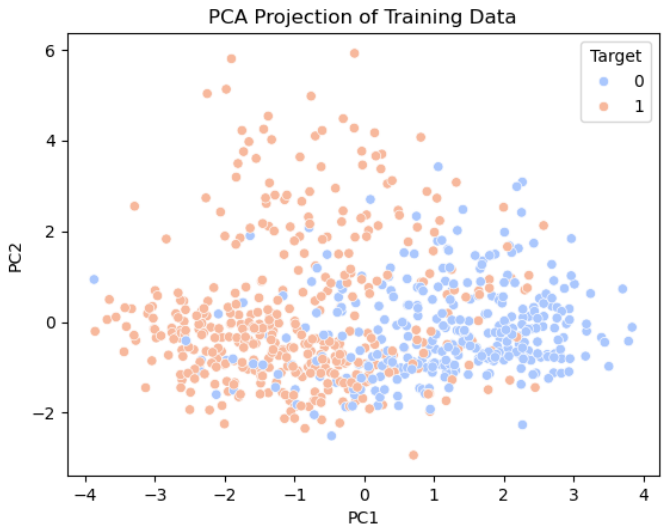

Fig. 6. PCA Projection of Training Data

A baseline SVM model was trained using the radial basis function (RBF) kernel with 'balanced' as the class weight parameter to account for class imbalance and to ensure the RBF does not just favor the majority class. We determined the RBF kernel was a good starting point based on a few key characteristics of the dataset. As noted in the PCA, the class distributions are not cleanly separated by a straight line, and the RBF can form flexible, curved boundaries around class clusters. Additionally, with a modest number of features like we have, the RBF is not likely to overfit the way it sometimes can in high-dimensional spaces. The inclusion of probability=True enables probabilistic outputs which we used for ROC curve generation. This model performs reasonably well and is visualized via confusion matrix and ROC curve. A key insight here is that while accuracy is informative, the ROC AUC provides a more holistic view of model discrimination.

SVM models using both linear and polynomial kernels were used to serve as a benchmark. We anticipated the linear kernel to underperform due to the non-linear nature of the dataset, but it appears that its classification scores were marginally better than that of the RBF kernel despite its lower boundary flexibility. The polynomial kernel also had similar classification scores using the default degree of 3, and would need to undergo parameter tuning to avoid overfitting on a relatively small dataset such as ours. Testing different kernel types (linear and polynomial) confirmed that the RBF kernel is best suited based on theoretical expectations, given the non-linear nature of the dataset, and despite how similar the accuracy and classification reports are.

The most significant improvement came from conducting hyperparameter tuning via GridSearchCV. The preprocessing and model were wrapped into a Pipeline, which ensured that scaling is performed within each cross-validation fold to prevent data leakage. The grid search initially used accuracy as the scoring metric, but we later decided that ROC AUC is much more appropriate than plain accuracy in a class-imbalanced setting. The parameter grid is focused on tuning C, gamma, and confirming the kernel as RBF. The narrowed values (C: [0.1, 0.5, 0.8] and gamma: ['scale', 0.1, 0.5, 0.8]) were informed by earlier tests that showed poor generalization from overly aggressive values like C=100 or gamma=1. The final tuned model shows a modest but consistent improvement in both accuracy and ROC AUC, confirming that the added tuning complexity was worth it.

IV. Comparison

This section includes the following: 1) comparing the performance of different machine learning algorithms that you used, and 2) comparing the performance of your algorithms with existing solutions if any. Please provide insights to reason about why this algorithm is better/worse than another one.

A. Logistic Regression vs. Support Vector Machine

The Logistic Regression Binary model performed slightly worse overall compared to the SVM model. A crucial metric to consider is the recall for class 1. SVM had a better recall with 91%, which means the model was better at correctly identifying patients with heart disease. This higher recall is ideal in real scenarios. The logistic regression model had 15 false negatives, while the SVM only had 9. Using the SVM would be a better choice for predicting heart disease as it would reduce the number of patients who think they are healthy when they are not. While the Logistic regression binary model is a good baseline because it is fast to train and easy to tune, the SVM is better for the real world because it handles non-linear boundaries.

Comparing to a top voted project by Fahad Rehman using the same dataset as ours [3], the logistic regression model that we optimized performed significantly better. Rehmans logistic regression model was determined to have an accuracy of 50%. The poor performance of their model seems to result from the training of a multiclass model without any multiclass specific techniques. Since we adjusted the dataset to account for a binary classification, we achieved significantly better results. For a fair comparison, I will compare Rehmans work to our multiclass model as well. With only a few optimizations, our multiclass model had an improved accuracy at 57%. This top rated project shows the importance of understanding the data that is being worked with. Our visualizations and optimizations show how we understood which techniques to apply to extract the most performance from these models.

B. Random Forest vs. Logistic Regression

When considering the performance between the Random Forest model and the Logistic Regression model, they preformed quite similarly with slight differences with predictions. The accuracy of the Random Forest model and Logistic Regression model were both approximately 84 percent. When considering the types of predictions made by each model, out of 184 predictions the Random Forest model made 30 incorrect predictions, 18 of which were false negatives and 12 that were false positive, while out of 184 predictions the Logistic Regression model made 29 incorrect predictions, 15 of which were false negatives and 14 that were false positive. Out of 102 subjects that were positive for heart disease in the testing data for the Logistic Regression model, it classified 87 of the subjects correctly. Out of the remaining 82 subjects that did not have heart disease, it correctly classified 68 of them. When compared to the Random Forest model, out of 109 subjects that were positive for heart disease in the testing data, 91 were correctly classified to have heart disease. Out of the remaining 75 subjects that did not have heart disease, the Random Forest classified 63 were correctly classified to not have heart disease.

After looking closer at the predictions made by the Random Forest model and the predictions made by the Logistic Regression model, we see that both their F1 scores for classifications of the presence of heart disease are approximately 86 percent. The main difference was found in the F1 score for classification of no heart disease. The Random Forest model had an F1 score of approximately 81 percent while the Logistic Regression model had an F1 score of approximately 82 percent. Although the difference is not large, this still makes the Logistic Regression model more ideal for classifying the presence of heart disease.

C. Support Vector Machine vs. Random Forest

When comparing the performance of the Support Vector Machine (SVM) and Random Forest (RF) models, the evaluation metrics we observed showed that both models perform well, but with distinct advantages. The SVM achieved slightly higher overall accuracy at 85%, compared to the Random Forest's 83%. Additionally, SVM outperformed RF in terms of recall for the positive class (91% vs. 81%), F1 score for the positive class (87% vs. 84%), and ROC AUC (92% vs. 89%). These metrics suggest that the SVM is particularly effective at identifying true positive cases (patients who actually have heart disease), which is critical in medical diagnosis, where false negatives can have serious consequences. SVM's higher recall and F1 score also highlight its strength in catching more positive heart disease cases, even at the risk of slightly more false positives. Its higher ROC AUC demonstrates its ability to distinguish between the two classes effectively across various classification thresholds. The ROC curve for the tuned SVM model visually confirms a high true positive rate at different false positive rates.

The Random Forest model provides competitive performance with notable strengths of its own. Though slightly behind in overall accuracy and recall, it achieved a higher precision score (88% vs. SVM's 84%), indicating that when it predicts a positive case, it is more likely to be correct. This suggests that the RF model is more conservative in its predictions, yielding fewer false alarms but potentially missing more actual cases. Its performance remains strong despite using a reduced feature set of seven variables, as shown in the AUC vs. Number of Features plot, which illustrates that predictive performance peaks early and remains stable with fewer features, enhancing interpretability and robustness.

In summary, both models are effective in distinguishing between patients with and without heart disease. However, the SVM slightly edges out the RF in terms of recall and overall balance, making it the more suitable option when maximizing disease detection is the priority. On the other hand, Random Forest offers advantages in precision and feature importance insight, making it a valuable alternative when interpretability or resilience to noise is more critical.

V. Future Directions

With an additional 6 months, we would improve our models initially by gathering more data. We would combine datasets and study the correlations and patterns that emerge with the increase of information. For missing data, we could also test other Machine Learning Model, like a Random Forest, to fill in the missing values with values that fit the subject better based on all their other values. We could essentially train a model with the missing values as the prediction for the model and get a more accurate representation of the missing data compared to using Mode Imputation or Mean Imputation. We also could train all our models to be more sensitive for predictions and handle multi-class predictions based on the severity of the diagnosed heart disease to help identify what subjects would need medical treatment the most. We would like to experiment with neural networks to explore how deep learning is able to handle the problem of predicting heart disease.

VI. Conclusion

Using multiple machine learning algorithms, we trained and optimized models with the goal of predicting heart disease in patients.

Our random forest model had an accuracy of 84% with 18 false negatives. Although it ranks in last place, its performance was impressive considering the limited dataset it was trained with.

The logistic regression models were a strong baseline especially when paired with a K-nearest neighbor algorithm during pre processing. The binary model had an accuracy of 84% and 15 false negatives. The multiclass model had a poor accuracy of 57% due to its difficulty with differentiation between stage 1, 2 and 3 heart disease. It was able to differentiate between no heart disease, stage 1 and very severe cases. Despite the poor accuracy, the model had only 11 false negatives.

Our support vector machine model performed the best with an accuracy of 85% and only 9 false negatives. This model was more difficult to tune, but the effort resulted in a more reliable prediction, and the most suitable algorithm for this problem. To further improve this model, we would use a KNN or random forest algorithm during pre processing to fill empty values with more realistic numbers rather than column averages.

The problem of predicting heart disease was well addressed and our models gave a realistic prediction.

The metrics of our models show promising results, demonstrating that machine learning can effectively support early heart disease detection and assist healthcare professionals in making informed decisions.

REFERENCES

[1] A. A. Ahmad, "Prediction of heart disease based on machine learning using jellyfish optimization algorithm," *MDPI*, 2023. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10378171/

[2] "Knnimputer — scikit-learn documentation," https://scikit-learn.org/stable/modules/generated/sklearn.impute.KNNImputer.html, accessed: 2025-04-15.

[3] F. Rehman, "Heart disease prediction using 9 models," https://www.kaggle.com/code/fahadrehman07/heart-disease-prediction-using-9-models/notebook, 2024, accessed: 2025-04-15.