Heart Disease Classification Using Neural Network

Ayushi Pitchika

1. Introduction

Heart diseases continue to pose significant challenges in global healthcare, especially in resource-constrained regions. Conventional detection methods, reliant on expert assessment of medical records, physical exams, and symptoms, are susceptible to human error and diagnostic delays.

Since accurate heart disease prediction is crucial for effective clinical management, to address the limitations of conventional approaches, researchers are actively building predictive models using artificial neural networks and machine learning techniques. These models can analyze vast medical data, offering comparable accuracy to traditional methods while expediting diagnostic processes. By leveraging extensive medical datasets, machine learning algorithms can automate intricate data analysis while continually refining their predictive accuracy. They show promise in navigating the complexities of heart disease diagnosis and improving patient outcomes.

This study investigates the development of such a deep neural network model, employing a classification approach within a machine learning framework to identify heart disease. The classification goal of the study is to predict whether the patient presents heart disease or not, leveraging PyTorch capabilities to set up a Deep Neural Network.

High precision is desirable as false positive predictions would be costly or undesirable, such as unnecessary medical interventions or treatments. High recall is also important as false negatives (missed diagnosis) are costly or unacceptable, and high recall ensures that the model identifies as many cases of heart disease as possible.

Given the importance of both precision and recall in the context of predicting heart disease, the f1 score, the harmonic mean of precision and recall, will be investigated as it provides a single value that combines both measures.

2. Data Exploration

To start off data exploration, basic information about the dataset was determined. As seen in Figure 2-1, the dataset was imported as a pandas dataframe. It consists of 14 columns, including the target variable, and 303 data points. The target variable has a binary classification with 1=yes and 0=no. All the features provided are numeric type and there are no missing values. There is a mix of categorical and numerical features in the dataset. Since the categorical features have already been encoded into numeric format, they do not require label encoding during the pre-processing step. This gives us a good idea of the

properties of the dataset and helps map out subsequent data pre-processing steps.

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
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   trestbps  303 non-null    int64  
 4   chol      303 non-null    int64  
 5   fbs       303 non-null    int64  
 6   restecg   303 non-null    int64  
 7   thalach   303 non-null    int64  
 8   exang     303 non-null    int64  
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64  
 11  ca        303 non-null    int64  
 12  thal      303 non-null    int64  
 13  target    303 non-null    int64  
dtypes: float64(1), int64(13)
```

Figure 2-1: Information about the dataset.

The description of the data along with the mean, min, max and the quantiles can be seen in Figure 2-2.

```
data.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000

Figure 2-2: Description of dataset.

A boxplot of all the numerical/continuous features was plotted in Figure 2-3 to provide further insight. Box plots are used for understanding the central tendency, spread, and distributional characteristics of continuous variables, as well as identifying potential outliers and asymmetries in the data.

For example, we notice that 'oldpeak' has a skewed distribution. There are marked differences in the spread of the positive and negative target values for age, where people between 45-60 years are more likely to present positively. Also, the positive group has a higher mean ‘thalach’ value than the negative.

Additionally, 'trestbps', 'chol' and 'oldpeak' have a few outliers. Because of the size of the dataset, it would be recommended to not drop the outlier datapoints as it would be challenging to train a neural network if the training set is too small.

If the decision is made to handle the outliers by data transformation or winsorization, testing the model's performance with and without outlier treatment as part of the sensitively analysis to ensure that the results are not overly influenced by outlier handling

techniques.

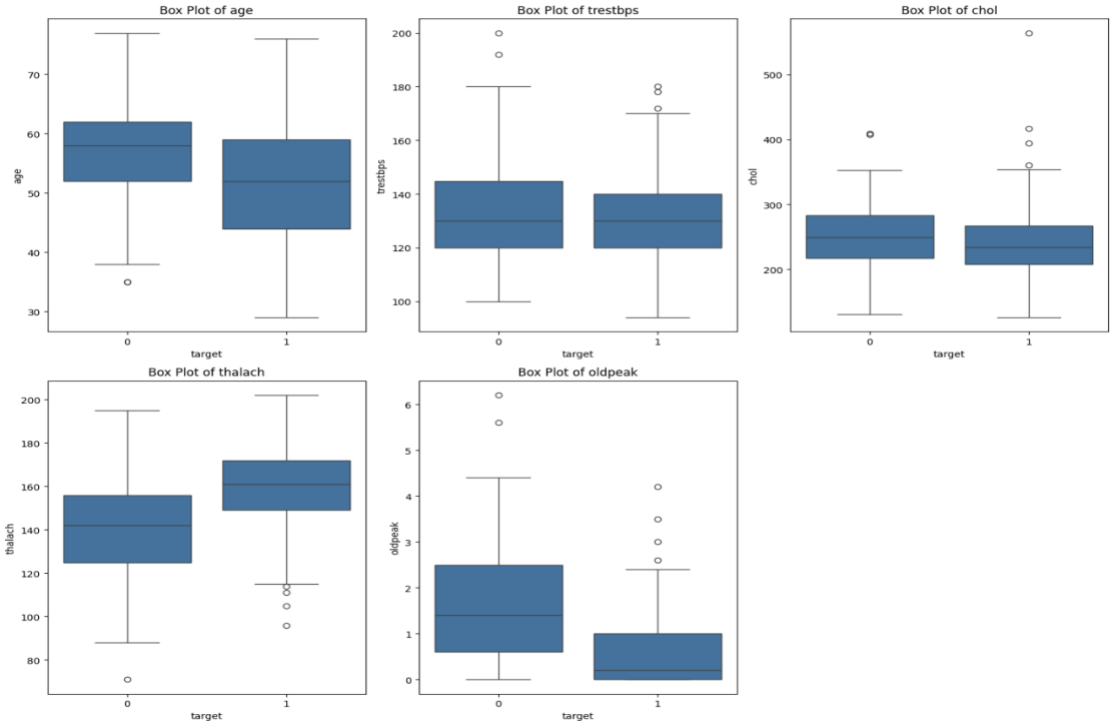

Figure 2-3: Feature visualization of numerical features.

Bar plots are plotted for categorical features in Figure 2-4 to investigate the distribution of the various categories. The dataset provided is well balanced as seen from the bar plot for the target variable. If a dataset is imbalanced, oversampling, undersampling or synthetic data generation (bootstrapping) can be performed to balance the classes and prevent the model from being biased towards the majority class. Therefore, this step can be skipped during pre-processing for this study.

There are double the number of males than females in the dataset. So it is very imbalanced for this feature and therefore not a good way to make observations on the basis of gender/sex. Similar distribution is seen in the 'exang' feature which indicates whether there was exercise induced angina. This can also lead to model bias if we wanted to make observations based on this feature.

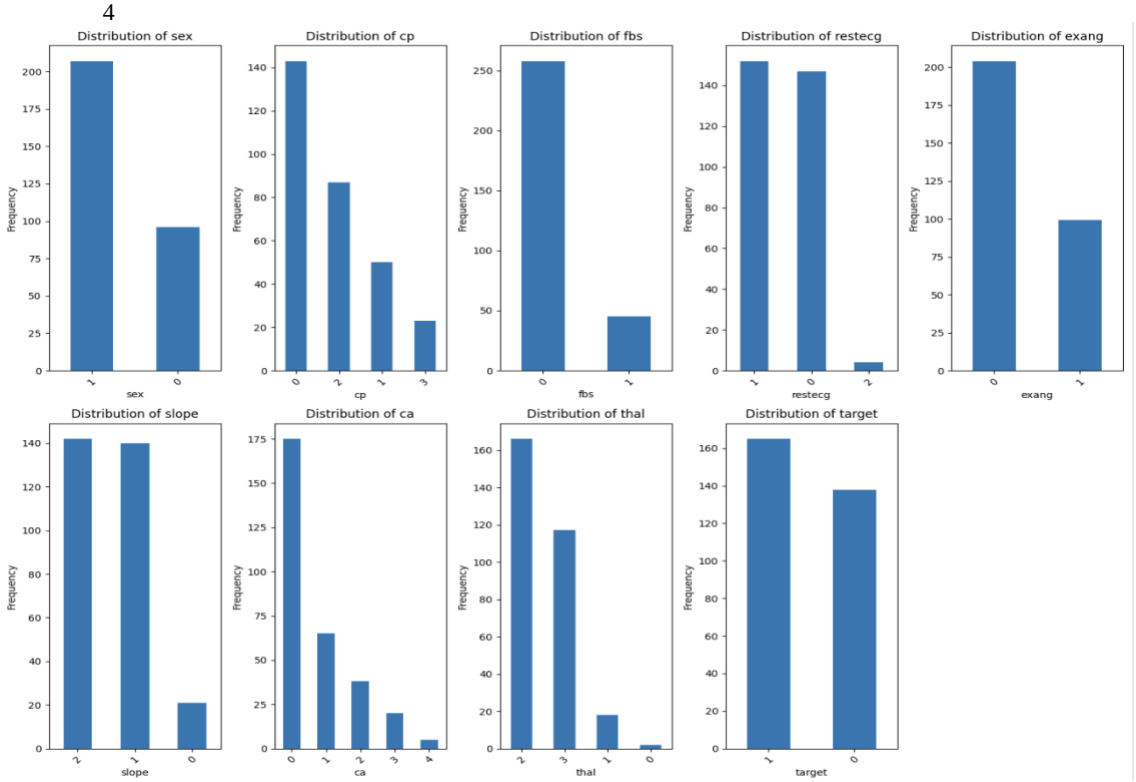

Figure 2-4: Feature visualization of categorical features.

To determine feature importance or which features had higher information gain, two methods were used. Since there are several outliers in the numerical features, we will use random forest to quickly determine feature importance as it uses bootstrapping and multiple decision trees to make predictions and is therefore robust to outliers. Given the low dimensionality of the dataset, we will not be dropping features based on their importance and will only consider it for exploring the data. So, a simple random forest model was created to leverage its feature importance attribute built-in into the model available in the scikit-learn library and the results were as seen in Figure 2-5.

	Feature	Importance
2	cp	0.135346
7	thalach	0.129879
11	ca	0.108091
12	thal	0.105813
9	oldpeak	0.102469
0	age	0.091921
4	chol	0.076270
3	trestbps	0.075079
8	exang	0.058675
10	slope	0.055157
1	sex	0.030980
6	restecg	0.021797
5	fbs	0.008522

Figure 2-5: Random Forest feature importance.

The plotting of a correlation matrix is another good way to explore correlations between features and the target variable and to compute correlations between numerical features

to identify potentially redundant features.

From the matrix in Figure 2-6, we can see that 'cp' and 'thalach' have a high positive correlation with the target and 'exang' and 'oldpeak' have high negative correlations. Also 'fbs' seems to be minimally correlated to the target. These may provide medical professionals insights what causes patients to present heart disease and how it can be prevented. Also, since no two features have a correlation of -/+ 1, it can be deduced that none of the features are redundant.

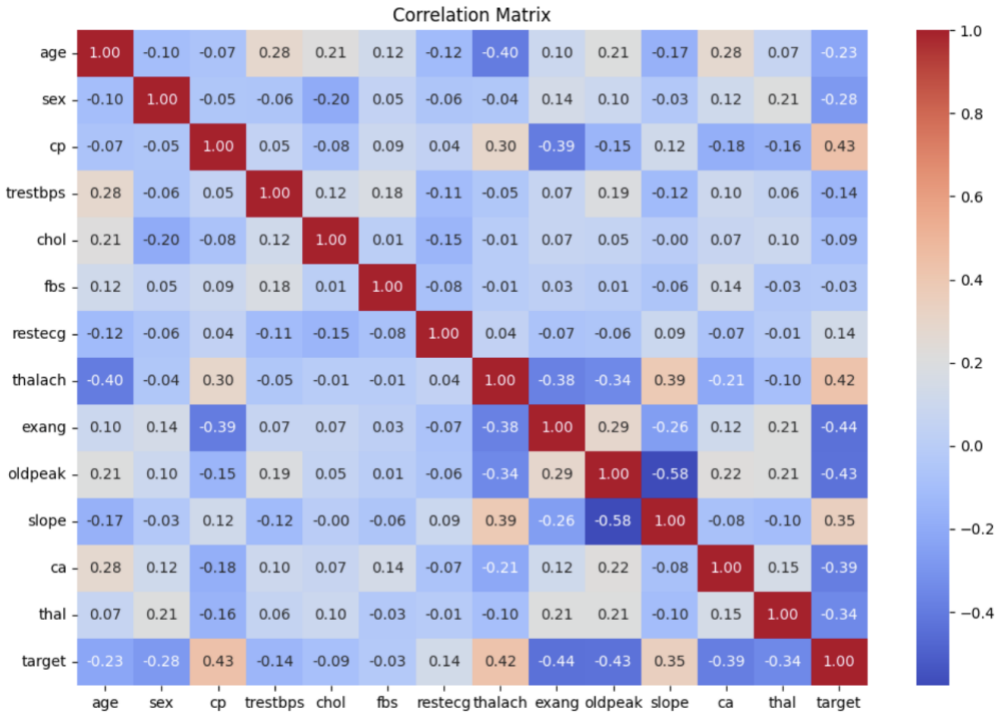

Figure 2-6: Correlation matrix for feature importance.

3. Data Preprocessing and Cleaning

The dataset did not require a lot of preprocessing steps as it was well processed as delivered. There was no need to handle missing values and the categorical variables were already encoded into numerical format.

So, we moved on to feature scaling since neural networks are sensitive to the scale of input features, and it's therefore essential to scale numerical features to a similar range. Based on the distribution of data and sensitivity to outliers, Z-score normalization (standardization) method was used which scales features to have a mean of 0 and a standard deviation of 1, to have a normal distribution in the features being scaled.

Depending on the size and complexity of the dataset, feature selection or dimensionality reduction techniques like PCA are often performed during pre-processing to reduce the number of features while retaining most of the information, which can help improve the efficiency and performance of neural network models. Since we have a relatively low-dimensional dataset, this step was skipped.

Then the dataset was split into training and testing sets with a 70-30 split as specified,

where 70% of the data is used for training and 30% for testing. The split was also stratified to ensure that it maintained the distribution of the target classes to avoid data leakage.

Target variables are often normalized to ensure it falls within a similar range as the input features. However, since the target variable is binary (0 or 1), normalization was not required in this case.

As the dataset can be considered balanced, no techniques had to be applied to avoid bias over majority class.

4. Neural Network Architecture

The neural network architecture for HeartDiseaseModel() is a feedforward neural network with configurable parameters for size of the input, the number of hidden layers, the size of each hidden layer, and the activation function used. The breakdown of the architecture is as follows:

- Input Layer: The input layer has a size determined by the input_size parameter, which corresponds to the number of input features in the dataset.

- Hidden Layers: A neural network can have multiple hidden layers, each with the same number of nodes determined by the hidden_size parameter. The number of hidden layers is specified by the num_layers parameter. Each hidden layer is followed by an activation function specified by the activation parameter.

- Activation Function: The activation function is applied after each hidden layer to introduce non-linearity into the network and allow it to learn complex patterns in the data.

- Output Layer: The output layer consists of a single node, which is used for binary classification (predicting the presence or absence of heart disease). It uses a sigmoid activation function to output probabilities between 0 and 1, representing the likelihood of the positive class.

- Model Construction: The layers of the neural network are defined in the __init__ method using nn.Linear for the linear layers and the specified activation function. The layers are then organized into a sequential model using nn.Sequential.

- Forward Pass: The forward method defines how input data flows through the network. It passes the input x through the sequential model and applies a sigmoid activation function to the output to produce the final predicted probabilities.

This architecture allows for flexibility in designing neural networks with varying numbers of layers, sizes of layers, and activation functions, making it suitable for experimentation and optimization for the task of heart disease prediction.

5. Evaluation and Parameter Sensitivity Analysis

A few performance metrics were leveraged to evaluate model performance. The *metrics* module available in the scikit-learn library enabled the evaluation process.

- Accuracy is a measure of the overall correctness of the model's predictions, representing the ratio of correctly predicted instances to the total number of instances.

- Precision measures the proportion of true positive predictions out of all positive predictions made by the model. It indicates the model's ability to avoid false positives. Precision is important in applications where false positives are costly or undesirable, such as medical diagnosis or fraud detection.

- Recall measures the proportion of true positive predictions out of all actual positive instances in the dataset. It indicates the model's ability to capture all relevant instances of a particular class. Recall is important in scenarios where false negatives are costly or unacceptable, such as disease detection or anomaly detection.

- The F1 score, which is the harmonic mean of precision and recall, provides a single value that combines both measures. Given the importance of both precision and recall in the context of predicting heart disease, a metric that balances these two aspects is often used.

- AUC (Area Under the ROC Curve) score measures the area under the Receiver Operating Characteristic (ROC) curve, which plots the true positive rate (recall) against the false positive rate for different threshold values. AUC score provides an aggregate measure of the model's performance across all possible classification thresholds and therefore evaluates the model's ability to discriminate between classes across different threshold values.

The hyperparameters explored during hyperparameter tuning process are as follows:

Hyperparameter	**Value 1**	**Value 2**	**Value 3**
Number of nodes in each layer	32	64	128
Number of layers	1	2	3
Activation Function	Rectified Linear Unit (ReLU)	Sigmoid	Leaky Rectified Linear Unit (LeakyReLU)

The results of the hyperparameter tuning for model performance optimization are included in Figure 5-1.

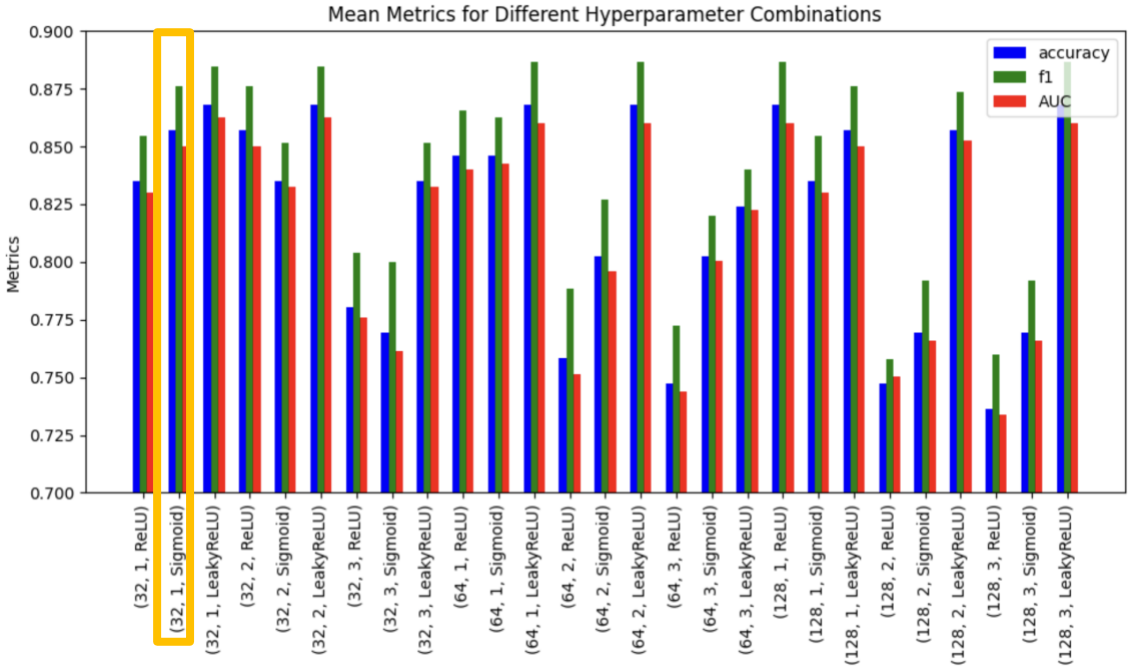

Figure 5-1: Mean Metrics for Different Hyperparameter Combinations

Based on highest accuracy, f1, recall and high AUC scores, the best neural network model is determined to have the following hyperparameters:

- 32 nodes per layer
- 1 layer
- Sigmoid activation function

The performance metrics for this model are as follows:

- Accuracy: 86.81%
- f1: 88.46%
- AUC score: 86.24%

Parameter sensitivity analysis is performed, and the results of the sensitivity analysis are interpreted to draw conclusions about the impact of each parameter on the model's performance. This information can be used to make informed decisions about model selection, hyperparameter tuning, or further experimentation. Accuracy will be used as the metric for the sensitivity evaluation. We will use heatmaps to provide a visual representation of the sensitivity of each parameter to changes in the evaluation metric i.e. accuracy. For easier interpretation, num_epochs and hidden_size are plotted together as a single plot, followed by num_layers and activation function, as seen in Figure 5-2.

- Change in number of nodes: Based on the heatmap, it is observed that the lower the number of nodes, the higher the accuracy. The accuracy changes by 1-3% as the number of nodes changes from 32 to 128.

- Change in number of epochs: Based on the heatmap, it is observed that the lower the number of nodes, the higher the accuracy. The accuracy changes by 2-4% as the number of epochs changes from 50 to 200.

- Change in number of layers: Based on the heatmap, it is observed that the lower the number of hidden layers, the higher the accuracy. The accuracy changes by 3-7% as the number of layers changes from 1 to 3.

- Activation function: Based on the heatmap, it is observed that the sigmoid activation function has the highest accuracy, followed by the LeakyReLU and then ReLU.

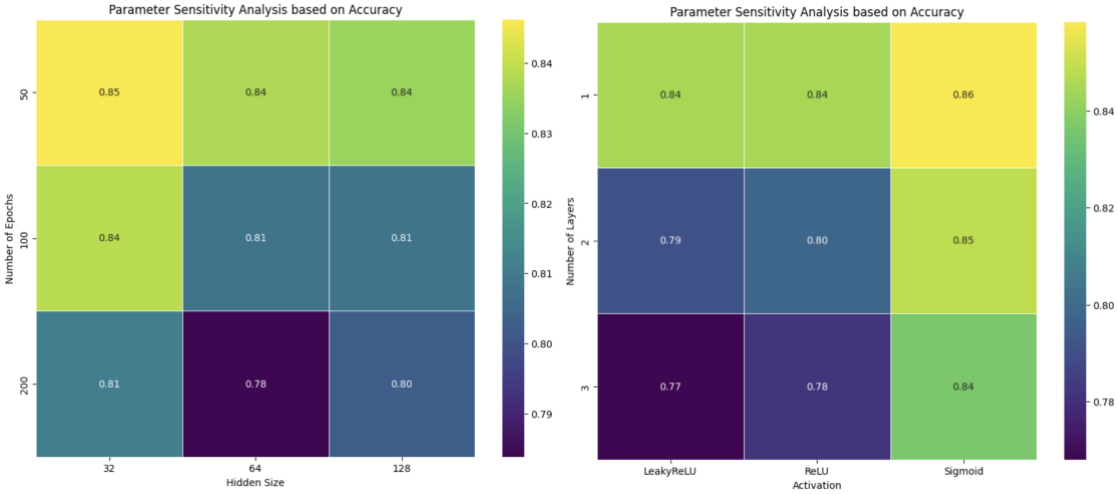

Figure 5-2: Heatmaps for parameter sensitivity analysis

6. Discussion and Conclusion

In this study, a feedforward neural network with hyperparameters for size of the input, the number of hidden layers, the size of each hidden layer, and the activation function used was created called *HeartDiseaseModel()*. This model was trained on a sample the dataset *heart.csv* and then tested using preciously split datapoints. Metrics such as accuracy, f1 and AUC score were used to evaluate the model performance and recommend the best model for the classification goal. Parameter sensitivity analysis was also conducted using the hyperparameters and number of epochs to determine how changes in these parameters affect the model performance metric, accuracy. Relevant tables and visualizations were created to report the findings to stakeholders in a clear and concise manner.

The model's performance was evaluated using metrics like accuracy, F1 score, and AUC. High values in these metrics indicates the model's effectiveness in distinguishing between healthy and diseased hearts. It would be crucial to assess how well the model generalizes to unseen data. Techniques like k-fold cross-validation would help ensure the model doesn't simply memorize the training data and can perform well on new patients. Also more closely monitoring the training process would allow us to identify potential issues like overfitting or underfitting. Techniques like learning rate scheduling and dropout regularization can also help optimize training.

It can be concluded that due to the smaller size of the dataset, a simpler architecture is better able to predict the classification goal as the data may not require the complexity introduced by larger architectures and may instead be capturing noise rather than true patterns. Smaller architectures are also less prone to overfitting than larger architectures.

Also it might be worthwhile to look into combining predictions from multiple trained models (ensemble methods) as they can sometimes outperform individual models.
A good final step to conclude would also be to validate the findings using additional datasets or cross-validation techniques to ensure the robustness of the conclusions.