Journal of Research High School

Vol. 2025 (3) 197 – 206
ISSN 2688-3651 [online]

A Comparative Analysis of Machine Learning Models on a Cardiovascular Disease Dataset

Vedaansh Vijaywargiya[1] *

[1]Oberoi International School JVLR Mumbai, India
*Corresponding Author: vedaansh.vijaywargiya@gmail.com

Advisor: Gurvansh Singh, gurvansh02@gmail.com

Received March 29, 2024; Revised October 29, 2024; Accepted January 2, 2025

Abstract

Cardiovascular diseases are a leading cause of death globally, making early detection imperative. This study aimed to compare various machine learning algorithms for predicting cardiovascular diseases based on parameters such as glucose levels, alcohol consumption, physical activity, and more. The primary objective was to identify the most effective algorithms for clinical prediction tasks by evaluating and optimizing their performance across key metrics. While previous research had utilized the same dataset from the University of California, Irvine, this study sought to expand on those efforts by incorporating datasets from additional sources. This broader approach aimed to provide a more comprehensive understanding of how to effectively apply machine learning algorithms in the diagnosis of cardiovascular diseases. The study employed methods such as bar plots for an initial analysis of the dataset and then experimented with every possible value for a parameter of the model within a specified range to assign the most appropriate value, maximizing the precision score of the model. The Decision Tree model achieved the best precision score among all the models, with a precision score of 78.8%, while the Multi-Layer Perceptron Classifier had the highest area under the receiver operating characteristic curve score of 0.784.

Keywords: Cardiovascular disease, Machine learning, Neural networks, Multi-layer perceptrons, Precision score, ROC - Receiver operating characteristic curve

1. Introduction

The World Health Organization estimated that about 17.9 million people died from Cardiovascular Diseases (CVDs) in 2019. According to the WHO, CVDs were the primary cause of death worldwide, causing 32% of all global deaths in 2019. CVDs are a group of maladies of the heart and blood vessels. Their primary risk factors include tobacco usage, excessive consumption of alcohol, an imbalanced diet, and a sedentary lifestyle. These risk factors may manifest in the form of increased blood pressure, increased glucose levels, and obesity. In 2019, 38% of all 17 million premature deaths (under the age of 70) caused by non-communicable ailments were caused by CVDs (World Health Organization: WHO, 2021).

More than 75% of deaths caused by CVD occurred in low- and middle-income countries. This was due to the scarcity of programs and technology that could timely predict and diagnose CVDs in these countries. CVDs were usually diagnosed extremely late in patients, thus a disproportionate amount of people died in their youth, during the most productive time of their life. However, with the development of advanced computational technology and an increase in the availability of medicinal data, machine learning flourished as a key tool in the prognosis of medical conditions (Kulkarni et al., 2022; Quer et al., 2021). Machine learning algorithms showcased their capability when it came to predicting CVDs and risk assessment, thus enabling medical experts to make more informed decisions. ML algorithms were also much more proficient than typical stochastic models at capturing intricate interactions and non-

Vol. 2025 (3) 197 – 206
ISSN 2688-3651 [online]

linear relations between features and outcomes.

This comparative study's primary goal was to optimize and evaluate the performance of various machine learning algorithms—such as K-Nearest Neighbors, Decision Trees, Random Forests, Sequential Neural Networks, and the Multi-Layer Perceptron Classifier—in predicting the presence of cardiovascular disease (CVD).

To determine which algorithms performed most effectively, the algorithms were evaluated based on metrics such as precision and AUC-ROC scores. Additionally, parameter optimization using iteration was implemented to enhance predictive performance.

The importance of this study lay in providing insights for selecting the most suitable algorithm for medical tasks, such as predicting cardiovascular diseases. These insights were intended to enhance the accuracy of prognosis and risk assessment for patients. The results of this study aim to assist experts in developing better healthcare strategies using technology, ultimately contributing to a reduction in CVD-related mortality.

2. Methodology

2.1 Models and Neural Networks

KNN

The K-nearest neighbors algorithm, also called the KNN algorithm, is a supervised machine learning algorithm. It works by finding distances between the input data whose output we are supposed to predict and then selects the "K" nearest data examples in a set of training data. To predict a value for classification, it chooses the label (output value) with the largest frequency and for regression problems, it averages out the labels. The baseline KNN model was first trained with default parameters of *[n_neighbors = 5, weights = 'uniform' , algorithm = 'auto' , leaf_size = '30', p = 2, metric = 'minkowski', metric_params = None, n_jobs = None]*. To find the optimum parameter values, the model was trained iteratively with a range of different *n_neighbors* values and a graph of *precision* against the *n_neighbors* value was plotted. Next, the *algorithm* parameter was experimented with and the value with the highest precision and recall was the *algorithm = 'kd_tree'* value. Finally, the optimum value of the parameter *leaf_size* was acquired using the same parameter optimisation method.

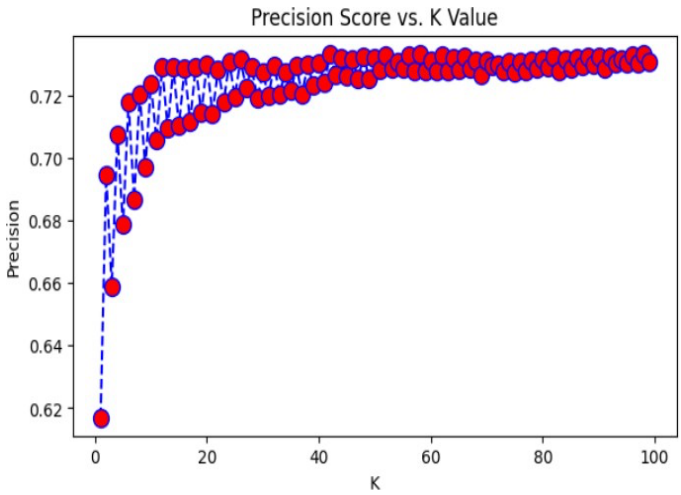

Figure 1. Graph of precision score against the n_neighbors value.

Decision Tree

The decision tree algorithm is a supervised machine learning algorithm. It works by starting out with the entire dataset as a root node. The root node then gets divided into two or more homogenous sets called sub-nodes by a process called splitting. This sub-node, if it splits into further sub-nodes, is called a decision node. Each of these decision nodes act as test cases for a particular attribute and the sub-nodes descending from that decision node are possible answers to that test case. The final node, and the one that classifies the data is called the leaf or terminal node of the tree and it does not split further.

The Decision Tree model was first trained as a baseline model and its performance was recorded. The parameters of this baseline model were set to default initial parameters. Then the model was run through the same optimization method as the KNN model, varying *max depth* values and finding the most optimum one. Next, the *criterion* parameter was set to *'entropy'*. The best value of the *random state* variable was also obtained by parameter optimization. Finally, the *splitter* parameter was set to *'random'*.

Vol. 2025 (3) 197 – 206
ISSN 2688-3651 [online]

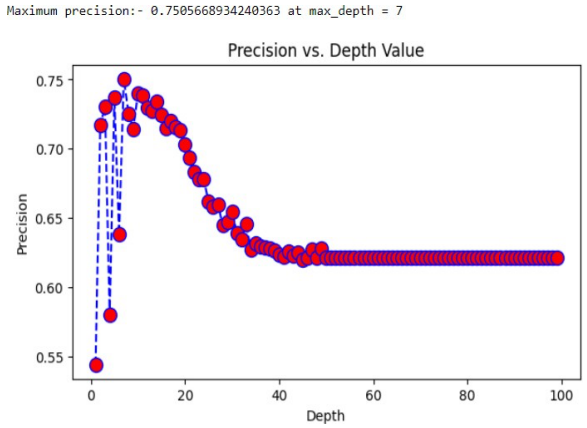

Figure 2. Graph of precision score against the max_depth value

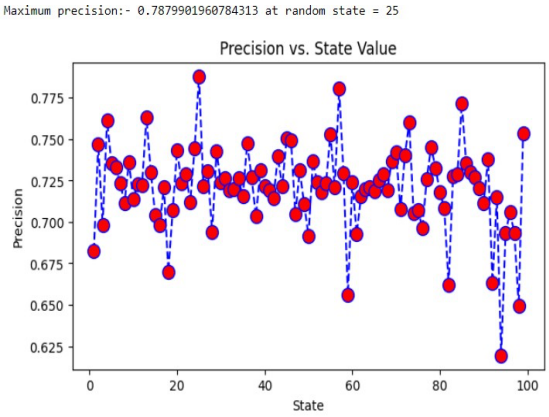

Figure 3. Graph of precision score against the random_state value (at max_depth = 7)

Random Forest

The random forest algorithm is a supervised machine learning algorithm. It works by selecting random samples from a given dataset, creating a decision tree for each training data, and then votes by averaging the outcome of each decision tree. The most voted prediction/outcome becomes the final prediction of the Random Forest algorithm.

The baseline Random Forest model was trained with initial default parameters. The model was then run through a loop to find the best possible values for the parameters *random_state* and *n_estimators* by storing the *precision_score* value for each *random_state* and *n_estimators* value in an array, plotting a graph of the *precision_score* against each value and thus maximizing the *precision_score.* I also tried changing the *criterion* parameter but got worse results and hence left it to default.

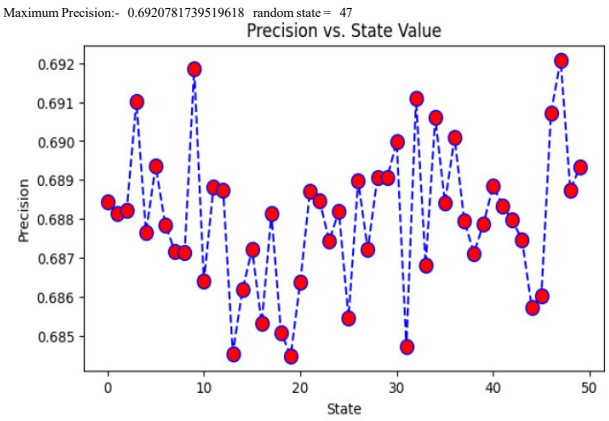

Figure 4. Graph of precision score against the random_state value

Sequential Neural Network

This model allows us to create a sequential neural network with multiple hidden neural layers. The model used was the Keras Sequential model. The baseline model trained first had a single hidden dense layer with *1 neuron* and *activation = "sigmoid".* The final model has *6 hidden dense layers* with the number of neurons decreasing from *22, 11, 6, 3, 2, and 1* from the first to the last hidden layer. The activation function used in each layer was the *'sigmoid'* function. The choice of hyperparameters, such as the number of layers and neurons, was guided by the objective of balancing computational efficiency with predictive power. The decision to use 6 layers for the Sequential Neural Network, with neurons decreasing across layers, reflects a common practice to reduce overfitting while maintaining complexity for learning non-linear patterns. The choice of batch size (512) was based on hardware constraints, balancing computational efficiency and convergence stability. The 'adam' optimizer was selected for its adaptive learning rate, which stabilizes training over multiple epochs. The sigmoid activation function maps outputs to probabilities between 0 and 1, suitable for binary classification. Thus, the sigmoid function is more suitable for binary classification tasks as compared to other popular activation functions like ReLU. The model was compiled with the parameters [*loss='mean_squared_error', optimizer='adam', metrics=['accuracy','Precision','AUC']].* Then the model was trained on the data with the number of epochs set to 36. Finally, the *batch_size* was set to *512* and the model was trained.

Vol. 2025 (3) 197 – 206
ISSN 2688-3651 [online]

```
model = Sequential()
model.add(Dense(22,activation = "sigmoid"))
model.add(Dense(11,activation = "sigmoid"))
model.add(Dense(6,activation = "sigmoid"))
model.add(Dense(3,activation = "sigmoid"))
model.add(Dense(2,activation = "sigmoid"))
model.add(Dense(1,activation = "sigmoid"))
```

```
model.compile(loss='mean_squared_error', optimizer='adam', metrics=['accuracy','Precision','AUC'])
model.fit(x,y,epochs = 36, batch_size = 512)
```

Figure 5. Creation and Training of the Sequential Model

Multi-Layer Perceptron Classifier

The MLPC model allows us to build a fully connected artificial neural network which trains iteratively since it calculates the partial derivative of the cost function at each step with respect to each of the features of the model. The MLPC model was trained with *random_state* set to 0 and *hidden_layer_sizes* set to 5

2.2 Dataset Pre-Processing and Analysis

The dataset is a cleaned version of the dataset provided by Svetlana Ulianova on Kaggle. The original dataset contains 70000 records and 11 features, but after pre-processing, only 68783 records were retained. The features of the dataset include age, gender, height, weight, systolic blood pressure, diastolic blood pressure, cholesterol levels, glucose levels, smoking status, alcohol consumption, and physical activity, with cardiovascular disease as the target variable. To enhance data quality and reliability, multiple pre-processing steps were taken. Firstly, outliers were removed by identifying implausible values for height, weight, and blood pressure. For instance, blood pressure values that were abnormally low or high beyond known physiological limits were removed to avoid distorting model outputs. Body Mass Index (BMI) was calculated using the standard formula (BMI = weight (kg) / height (m)²) to eliminate records with impossible values, such as BMI below 10 or above 60, which likely indicated data entry errors. A value of 1 represented women and a value of 2 represented men. The patient age was converted to years from its initial value in days to appropriately scale the dataset. Categorical features, such as cholesterol and glucose levels, were encoded with integer mappings—1 representing normal, 2 above normal, and 3 well above normal—while binary features (smoking, alcohol consumption, and physical activity) were encoded as 0 or 1. For example, a value of 0 for smoking indicates that the patient was not an active smoker whereas a value of 1 indicates that the patient was an active smoker.

First, let us see the relationship between the age of the patient and whether they had cardiovascular disease or not.

As observed in figure 6, ages 55+ had a greater 1 to 0 ratio, which, largely, kept increasing and hence were more prone to cardiovascular disease. This could imply that older patients have a higher chance of contracting cardiovascular disease.

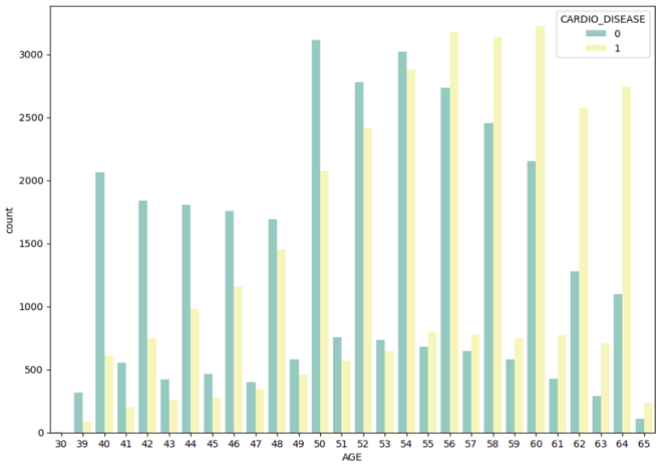

Figure 6. Count of Cardiovascular Disease vs Age

Figure 7a showcases the variation of the number of people with cardiovascular disease against their cholesterol levels. It can be seen that the proportion of people suffering from CVD is greater as their cholesterol level increases which could show a direct relationship between the cholesterol level and the probability of contracting CVDs. Figure 7b showcases the variation of the number of people with cardiovascular disease against their glucose levels. It can be seen that the proportion of people suffering from CVD, although greater, stays nearly the same as the glucose level increases. Figure 7c showcases the

Vol. 2025 (3) 197 – 206
ISSN 2688-3651 [online]

variation of the number of people with cardiovascular disease against whether they consume alcohol or not, with 0 indicating no consumption and 1 indicating consumption. The proportion of people who consume alcohol to those who don't consume alcohol is extremely small. This could be a potential drawback in the prediction of the ML algorithm.

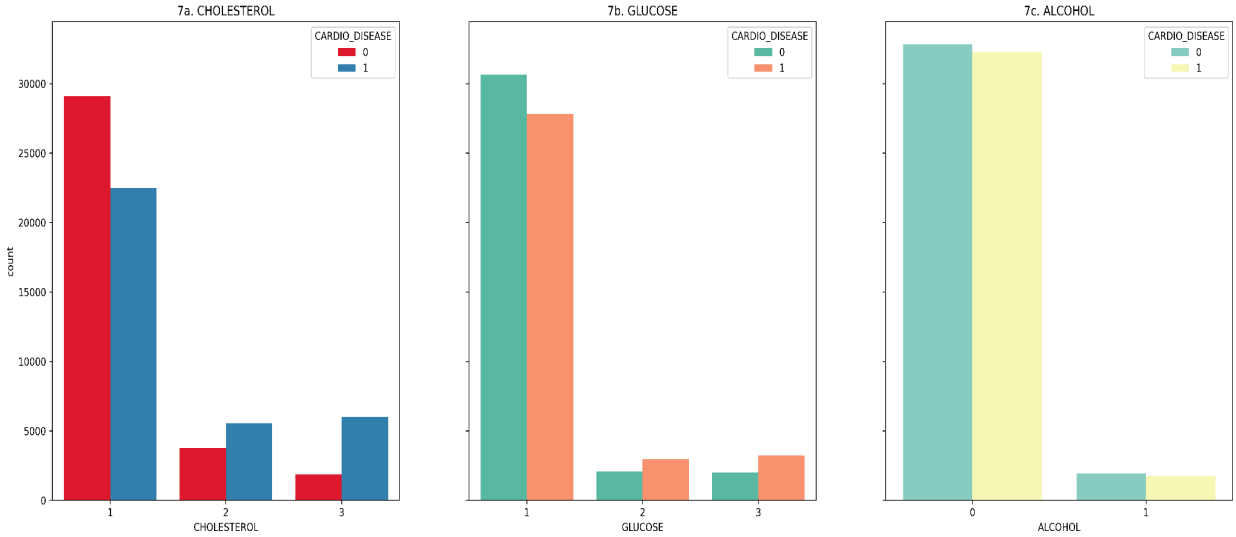

Figure 7. Count of Cardiovascular Disease (y-axis) vs Cholesterol (7a, x-axis), Glucose (7b, x-axis), and Alcohol Consumption (7c, x-axis)

3. Results and Discussion

3.1 Baseline Models

Figure 8 showcases how the true positive rate of each model varied with their respective false positive rates at different points in the prediction process. The blue line in Figure 8 showcases the baseline KNN model. It had a precision score of 67.73% and an area under the ROC curve of 0.73. The orange line in Figure 8 showcases the baseline decision tree model, which had a precision score of 62.41% and an area under the ROC curve of 0.63.The green line in Figure 8 showcases the baseline MLPC model, which had a precision score of 75.15% and an area under the ROC curve of 0.78. The red line in Figure 8 showcases the baseline Random Forest model, which had a precision score of 68.90% and an area under the ROC curve of 0.70.

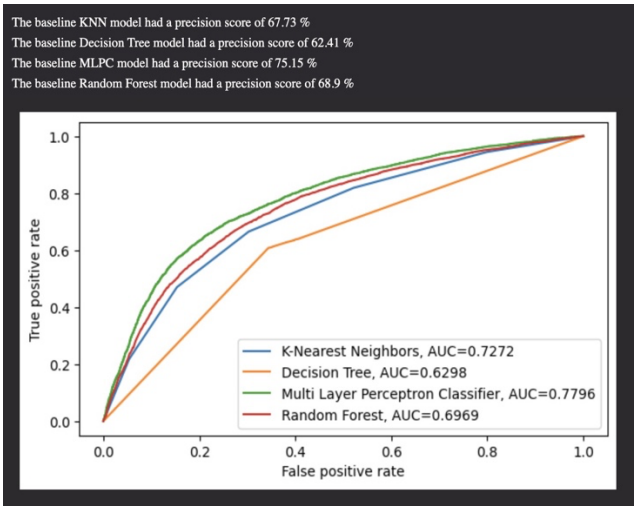

Figure 8. Performances of the baseline K-Nearest Neighbors, Decision Tree, Multi-Layer Perceptron Classifier, and Random Forest models.

Figure 9. showcases the training phase of the baseline sequential neural network model. As the epoch count increases, the value of the loss function, which is the distance between the actual value and the value predicted by the model, decreases from 0.505 to 0.274. The final precision score of the model is 70.3% and the roc-auc score is 0.73.

Vol. 2025 (3) 197 – 206
ISSN 2688-3651 [online]

```
Epoch 1/12
135/135 [==============================] - 1s 1ms/step - loss: 0.5050 - accuracy: 0.4950 - precision: 0.4949 - auc: 0.5001
Epoch 2/12
135/135 [==============================] - 0s 1ms/step - loss: 0.3950 - accuracy: 0.5818 - precision: 0.5631 - auc: 0.5964
Epoch 3/12
135/135 [==============================] - 0s 1ms/step - loss: 0.3351 - accuracy: 0.6317 - precision: 0.6569 - auc: 0.6597
Epoch 4/12
135/135 [==============================] - 0s 1ms/step - loss: 0.3256 - accuracy: 0.6403 - precision: 0.6643 - auc: 0.6709
Epoch 5/12
135/135 [==============================] - 0s 1ms/step - loss: 0.3169 - accuracy: 0.6490 - precision: 0.6744 - auc: 0.6809
Epoch 6/12
135/135 [==============================] - 0s 1ms/step - loss: 0.3098 - accuracy: 0.6547 - precision: 0.6787 - auc: 0.6899
Epoch 7/12
135/135 [==============================] - 0s 1ms/step - loss: 0.3030 - accuracy: 0.6610 - precision: 0.6838 - auc: 0.6982
Epoch 8/12
135/135 [==============================] - 0s 1ms/step - loss: 0.2965 - accuracy: 0.6666 - precision: 0.6885 - auc: 0.7067
Epoch 9/12
135/135 [==============================] - 0s 1ms/step - loss: 0.2908 - accuracy: 0.6705 - precision: 0.6912 - auc: 0.7138
Epoch 10/12
135/135 [==============================] - 0s 1ms/step - loss: 0.2843 - accuracy: 0.6781 - precision: 0.6983 - auc: 0.7209
Epoch 11/12
135/135 [==============================] - 0s 1ms/step - loss: 0.2790 - accuracy: 0.6833 - precision: 0.7034 - auc: 0.7270
Epoch 12/12
135/135 [==============================] - 0s 1ms/step - loss: 0.2744 - accuracy: 0.6869 - precision: 0.7030 - auc: 0.7328
```

Figure 9. Baseline Sequential Model

Table 1 showcases the precision scores and the ROC-AUC scores of the baseline model for each algorithm and analyzes the cause of the specific precision score or ROC-AUC score.

Baseline Models				
Algorithm	**Precision Score**	**ROC - AUC**	**Analysis of Performance**	**Possible Improvements**
K-Nearest Neighbors	67.73%	0.727	Seems to struggle with high dimensional data as the "curse of dimensionality" reduces neighbor search accuracy. (Kouiroukidis, Evangelidis, 2011)	Reducing the value of the n_neighbors parameter. Reducing the parameter could offset issues caused by high dimensionality as the model would be able to take an average of surrounding values and not overfit to local values.
Decision Tree	62.41%	0.630	The model had a lower precision score due to overfitting caused due to the max-depth being larger than optimal, even though it was the default.	Reducing max-depth could prevent overfitting as at high depths, the model catches too much noise from the leaf nodes. It overfits and leads to worse generalization.
Random Forest	68.90%	0.697	The random forest model might have performed better due to taking an average of all of its decision trees, which helped it reduce error.	Since the random forest algorithm takes an average of its decision trees, correlated trees do not contribute as much to improving performance. Thus, we could try reducing the n_estimators.
Sequential Neural Network	70.30%	0.733	The neural network might have performed better due to its ability to learn more effectively from its previous mistakes, and thus improve its performance in deeper epochs.	Sequential neural networks are particularly sensitive to changes in hyperparameters. To generalize more effectively, we should try increasing the number of hidden layers and batch size.
Multi-Layer Perceptron Classifier	75.15%	0.780	The MLPC already had a strong baseline due to its ability of optimizing itself iteratively, and had the best performance out of all other models.	Improving the architecture of the network by making it deeper with more hidden layers could enable it to capture more unseen and complex relationships between the features of the dataset and the target variable.

3.2 Final Models

The final parameters for the KNN algorithm were *[n_neighbors = 98, weights = 'uniform', algorithm = 'kd_tree', leaf_size = '30', p = 2, metric = 'minkowski', metric_params = None, n_jobs = None]*. The final KNN model, as seen by the blue line in Figure 10, had a precision score of 73.28% and an area under the ROC curve of 0.78.

The final parameters for the Decision Tree algorithm were *[ccp_alpha = 0.0, class_weight = None, criterion = entropy, max_depth = 7, max_features = None, max_leaf_nodes = None, min_impurity_decrease = 0.0,*

Vol. 2025 (3) 197 – 206
ISSN 2688-3651 [online]

min_samples_leaf = 1, min_samples_split = 2, min_weight_fraction_leaf = 0.0, random_state = 25, splitter = random] The final Decision Tree model, as seen by the orange line in Figure 10, had a precision score of 78.80%, the best so far and an area under the ROC curve of 0.71.

The final parameters for the Random Forest algorithm were *[bootstrap': True, 'ccp_alpha': 0.0, 'class_weight': None, 'criterion': 'gini', 'max_depth': None, 'max_features': 'sqrt', 'max_leaf_nodes': None, 'max_samples':None, 'min_impurity_ decrease': 0.0, 'min_samples_ leaf': 1, 'min_ samples_split': 2, 'min_weight_ fraction_leaf': 0.0, 'n_estimators': 50, 'n_jobs': None, 'oob_score': False, random_state: 47, 'verbose': 0, 'warm_start': False]*. The final Random Forest model, as seen by the red line in Figure 10, had a precision score of 69.21% and the ROC AUC of 0.70.

The final Multi-Layer Perceptron Classifier model, as seen by the green line in Figure 10, had a precision score of 78.25% and an area under the ROC curve of 0.784.

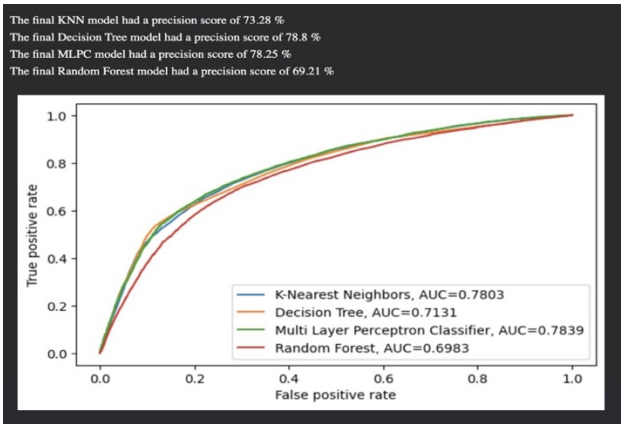

Figure 10. Performances of the final K-Nearest Neighbors, Decision Tree, Multi-Layer Perceptron Classifier, and Random Forest models.

```
Epoch 1/36
135/135 [==============================] - 2s 2ms/step - loss: 0.1892 - accuracy: 0.7197 - precision: 0.7415 - auc: 0.7820
Epoch 2/36
135/135 [==============================] - 0s 2ms/step - loss: 0.1889 - accuracy: 0.7195 - precision: 0.7381 - auc: 0.7825
Epoch 3/36
135/135 [==============================] - 0s 2ms/step - loss: 0.1887 - accuracy: 0.7209 - precision: 0.7413 - auc: 0.7830
Epoch 4/36
135/135 [==============================] - 0s 2ms/step - loss: 0.1889 - accuracy: 0.7201 - precision: 0.7408 - auc: 0.7826
Epoch 5/36
135/135 [==============================] - 0s 2ms/step - loss: 0.1888 - accuracy: 0.7199 - precision: 0.7394 - auc: 0.7828
Epoch 6/36
135/135 [==============================] - 0s 2ms/step - loss: 0.1893 - accuracy: 0.7182 - precision: 0.7375 - auc: 0.7817
Epoch 7/36
135/135 [==============================] - 0s 2ms/step - loss: 0.1892 - accuracy: 0.7192 - precision: 0.7405 - auc: 0.7818
Epoch 8/36
135/135 [==============================] - 0s 2ms/step - loss: 0.1891 - accuracy: 0.7193 - precision: 0.7388 - auc: 0.7823
Epoch 9/36
135/135 [==============================] - 0s 2ms/step - loss: 0.1889 - accuracy: 0.7196 - precision: 0.7412 - auc: 0.7827
Epoch 10/36
135/135 [==============================] - 0s 2ms/step - loss: 0.1888 - accuracy: 0.7210 - precision: 0.7418 - auc: 0.7829
```

Figure 11a. Final Sequential Model (First 10 Epochs)

```
Epoch 26/36
135/135 [==============================] - 0s 2ms/step - loss: 0.1886 - accuracy: 0.7197 - precision: 0.7399 - auc: 0.7832
Epoch 27/36
135/135 [==============================] - 0s 2ms/step - loss: 0.1887 - accuracy: 0.7207 - precision: 0.7409 - auc: 0.7832
Epoch 28/36
135/135 [==============================] - 0s 2ms/step - loss: 0.1886 - accuracy: 0.7207 - precision: 0.7407 - auc: 0.7831
Epoch 29/36
135/135 [==============================] - 0s 2ms/step - loss: 0.1885 - accuracy: 0.7205 - precision: 0.7388 - auc: 0.7837
Epoch 30/36
135/135 [==============================] - 0s 2ms/step - loss: 0.1884 - accuracy: 0.7207 - precision: 0.7424 - auc: 0.7837
Epoch 31/36
135/135 [==============================] - 0s 2ms/step - loss: 0.1884 - accuracy: 0.7201 - precision: 0.7393 - auc: 0.7837
Epoch 32/36
135/135 [==============================] - 0s 2ms/step - loss: 0.1885 - accuracy: 0.7211 - precision: 0.7421 - auc: 0.7835
Epoch 33/36
135/135 [==============================] - 0s 2ms/step - loss: 0.1884 - accuracy: 0.7211 - precision: 0.7419 - auc: 0.7837
Epoch 34/36
135/135 [==============================] - 0s 2ms/step - loss: 0.1886 - accuracy: 0.7200 - precision: 0.7396 - auc: 0.7832
Epoch 35/36
135/135 [==============================] - 0s 2ms/step - loss: 0.1884 - accuracy: 0.7208 - precision: 0.7407 - auc: 0.7838
Epoch 36/36
135/135 [==============================] - 0s 2ms/step - loss: 0.1886 - accuracy: 0.7209 - precision: 0.7423 - auc: 0.7832
```

Figure 11b. Final Sequential Model (Last 10 Epochs)

Vol. 2025 (3) 197 – 206
ISSN 2688-3651 [online]

Figures 11a and 11b show the first 10 and the last 10 epochs of the final sequential neural network model. The model's precision score is 74.2% and the roc-auc is 0.78

Table 2 displays the precision score and ROC-AUC scores of the final models for each algorithm after optimal parameters have been applied, and analyzes the changed precision and ROC-AUC scores.

Final Models			
Algorithm	**Precision Score**	**ROC - AUC**	**Analysis**
K-Nearest Neighbors	73.3%	0.780	The increase in the number of neighbors from 5 to 98 may have contributed to the algorithm's ability to reduce errors through the averaging of its "K" nearest examples. This highlights the importance of selecting an appropriate number of neighbors to balance bias and variance, ultimately improving classification accuracy.
Decision Tree	78.8%	0.713	As discussed before, limiting the maximum depth to 7 helped prevent overfitting during the training process, resulting in better generalization on unseen data. However, the ROC-AUC score indicates room for improvement in distinguishing between classes.
Random Forest	69.2%	0.698	Interestingly, the random forest algorithm did not exhibit significant improvements over its baseline model. Perhaps the hyperparameter values may need further tuning, or the feature set could be further optimized to enhance predictive power. The relatively low AUC score also indicates that the model struggles to effectively discriminate between the positive and negative classes.
Sequential Neural Network	74.2%	0.783	The increase in the number of neurons contributed to improved performance, as the model was able to learn from previous epochs. However, further investigation into the model architecture might yield even better results.
Multi-Layer Perceptron Classifier	78.3%	0.784	The precision score increased from 75.15% to 78.3% along with an increase in the AUC score from 0.780 to 0.784.

3.3 Final Discussion

All algorithms except random forest had substantial improvements from their baseline models. The Decision Tree classifier had the highest precision score of 78.8% followed by the Multi-Layer-Perceptron classifier with a precision score of 78.3%. The MLPC had the highest ROC AUC at 0.784 followed by the Sequential Neural Network model with a ROC AUC of 0.783.

When comparing the performance of these models to existing literature, the K-Nearest Neighbors (KNN) model achieved a precision of 73.3% and an AUC-ROC of 0.780, which slightly exceeded the precision of 72% reported by other researchers (Shirvaikar et al., 2021). This indicated that the KNN model was effective at correctly identifying positive cases. In contrast, the Random Forest (RF) model attained a precision of 69.2% and an AUC-ROC of 0.698, which was slightly lower than the precision of 73% observed in the literature. The lower AUC-ROC for the RF model suggested that it was less effective at distinguishing between classes compared to its KNN counterpart and the benchmarks established in previous studies. Overall, these results demonstrated that the KNN model performed competitively against established algorithms, while the RF model may have required further optimization to enhance its predictive capability.

4. Conclusion

Despite facing many challenges and limitations, machine learning algorithms continue to persevere and show results. Machine learning models, particularly those used for cardiovascular disease (CVD) prediction, hold promise for both research and clinical applications. They can assist in early disease detection, personalized treatment planning, and the automation of routine diagnostic processes. These capabilities align with the growing need for scalable healthcare solutions in resource-limited settings. Although the Decision Tree model had the best results with a precision score of 78.8%, it is important for future studies in the application of machine learning algorithms in healthcare to also experiment with a larger range of random state values, which is a limitation of this study. It is also

Vol. 2025 (3) 197 – 206
ISSN 2688-3651 [online]

important to note that the Multi-Layer- Perceptron Classifier also had major improvements from its baseline model and had a precision score of 78.3%. Thus, selecting a suitable algorithm for a particular task is extremely important. Furthermore, a comparison of machine learning algorithms with pre-existing CVD prediction algorithms is necessary to incorporate and validate machine learning algorithms in the medical field. The deployment of these models in real-world healthcare settings also raises important ethical considerations. Privacy concerns and biased datasets are two particularly important considerations that if left ignored, may propagate health disparities that disproportionately affect underrepresented populations.

Acknowledgments

I would like to thank my mentors and parents for supporting me in writing this paper.

References

Bradley, A. P. (1997). The use of the area under the ROC curve in the evaluation of machine learningalgorithms. Pattern Recognition, 30(7), 1145–1159. https://doi.org/10.1016/s0031-3203(96)00142-2

Gupta, B., et al. (2017). Analysis of various decision tree algorithms for classification in data mining. International Journal of Computer Applications, 163(8), 15–19. https://doi.org/10.5120/ijca2017913660

Kouiroukidis, N., & Evangelidis, G. (2011). The effects of dimensionality curse in high dimensional kNN search. 2011 15th Panhellenic Conference on Informatics, Kastoria, Greece, 41–45. https://doi.org/10.1109/PCI.2011.45

Kulkarni, P., Mahadevappa, M., & Chilakamarri, S. (2022). The emergence of artificial intelligence in cardiology: Current and future applications. Current Cardiology Reviews, 18(3). https://doi.org/10.2174/1573403x17666211119102220

Mathur, P., et al. (2020). Artificial intelligence, machine learning, and cardiovascular disease. Clinical Medicine Insights: Cardiology, 14, 117954682092740. https://doi.org/10.1177/1179546820927404

Quer, G., et al. (2021). Machine learning and the future of cardiovascular care. Journal of the American College of Cardiology, 77(3), 300–313. https://doi.org/10.1016/j.jacc.2020.11.030

Ray, S. (2019). A quick review of machine learning algorithms. 2019 International Conference on Machine Learning, Big Data, Cloud, and Parallel Computing (Com-IT-Con), India, 14–16 February 2019. https://doi.org/10.1109/comitcon.2019.8862451

Seaborn: Statistical data visualization. (Waskom, M.) (2021). Journal of Open Source Software, 6(60), 3021. https://doi.org/10.21105/joss.03021

Scikit-learn. (n.d.). DecisionTreeClassifier. In Scikit-learn documentation. https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

Scikit-learn. (n.d.). KNeighborsClassifier. In Scikit-learn documentation. https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

Scikit-learn. (n.d.). MLPClassifier. In Scikit-learn documentation. https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

Scikit-learn. (n.d.). RandomForestClassifier. In Scikit-learn documentation. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.htmll

Shirvaikar, A., Mandlik, A., & Ram, S. P. (2021). Prediction of cardiovascular disease by applying a combination of principal component analysis with machine learning techniques. International Research Journal of Engineering and Technology, 8(9), 2395–0056. https://www.irjet.net/archives/V8/i9/IRJET-V8I9310.pdf

Vol. 2025 (3) 197 – 206
ISSN 2688-3651 [online]

TensorFlow. (n.d.). The Sequential model. In TensorFlow documentation. https://www.tensorflow.org/guide/keras/sequential_model

Ulianova, S. (2019, January 20). Cardiovascular disease dataset [Dataset]. Kaggle. https://www.kaggle.com/datasets/sulianova/cardiovascular-disease-dataset

Varghese, D. (2021, December 6). Comparative study on classic machine learning algorithms. Towards Data Science. https://towardsdatascience.com/comparative-study-on-classic-machine-learning-algorithms-24f9ff6ab222

World Health Organization: WHO. (2021, June 11). Cardiovascular diseases (CVDs). https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)