Project Machine Learning

Sarah Girgis, Jalen Cyrus, Sayuri Chan, Thierno Sylla, Jayden Kashani Maleki

March 2025

Abstract

This paper compares different machine learning models, specifically a logistic regression model (LR), Linear Support Vector Machine (L-SVM), Decision Tree (DT), and a Feedforward Neural Network (MLP). The models predict the risk of heart disease based on a synthetic data set of 70,000 instances. The goal is to determine whether the neural network outperforms the three other traditional machine learning models. All models achieved high accuracy due to our data set containing trivial data. The neural network performed best overall with the lowest false negative (47) count, achieving an accuracy of 99.25%. The results suggest that, while the neural network outperforms the other models by a small margin in this context, the simplicity of the data makes it harder to draw strong conclusions. To get a better idea of how these models would work outside of this project, future work should focus on evaluating these models on more complex real-world data.

Contents

List of Figures

1 Introduction

Heart diseases remains a very common health issue affecting people worldwide; diseases many people suffer from at some point in their lives. Due to its direct impact and fluctuating nature, it may result in a fatal outcome. The unpredictable nature of this disease and its crucial time element make it very dangerous and necessary to act on quickly. Taking this into consideration, early cases of heart disease can be acted on in time. According to multiple researches, the underlying cause of heart disease lies mainly in a person's lifestyle. Activities such as smoking, or eating unhealthy foods increase the likelihood of someone experiencing heart disease [Ghodeshwar et al., 2023]. To prevent heart disease, it is crucial to know in advance whether someone is likely to experience heart disease. To do so, we will apply machine learning. Machine learning, or ML, is defined as "the field of study that gives computers the ability to learn without being explicitly programmed" [Mahesh, 2020]. This report conducts a comparative analysis of different ML models, specifically a (feedforward) neural network, support vector machine, decision tree, and linear regression model, to see which model applies best to the problem of heart disease risk. In this paper, we hypothesise that the feedforward neural network will outperform the other ML models.

2 Data Collection and Preprocessing

The data set contains 70,000 instances. It is synthetic data, which is not necessarily a problem in itself. However, this implies that the results may be biased based on how the data was retrieved. The target values are either a low (0) or high (1) risk of heart disease. In the image below (Figure 1), it can be seen that the data is perfectly distributed. This means that there is no class imbalance, which ensures there is no need for any concerns regarding the class distribution.

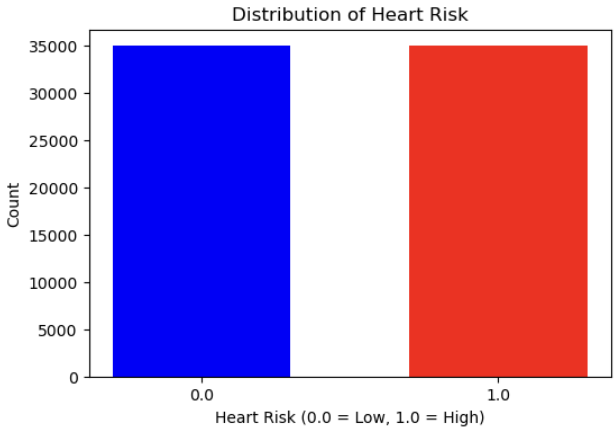

Figure 1: Heart Risk Distribution

The data contains a total of 18 features. Most features are binary, indicating whether a patient experiences specific symptoms (e.g. chest pain, fatigue, dizziness), or has certain risk factors (e.g. high cholesterol, diabetes). Age is the only continuous variable in the data set. Collectively, these features provide a broad view of each patient's clinical profile, which are used to assign the risk label. Similar to the target values, the features are evenly distributed, except for gender and age (Figure 2 and 3 respectively). Middle-aged people and one of the genders are the majority classes (the gender by a slight margin). However, the difference in distribution is not significant enough to cause any concerns or need for further adjustments. It could be the case that there are more men than women (relatively) in the data set who are labeled as having a high (or low) risk of heart disease, or vice versa. However, such concerns are not within the scope of this research. It should be noted that the description of the data set does not mention whether 1 (or 0) is male or female. We won't make any assumptions on this.

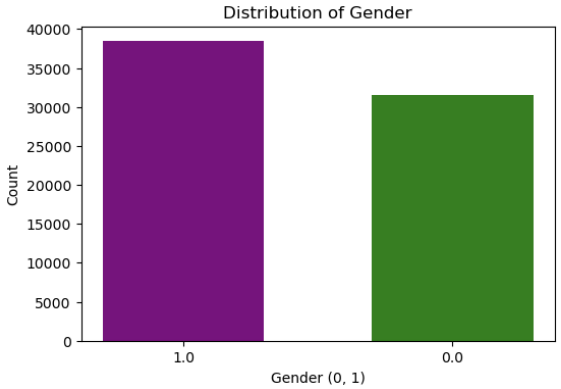

Figure 2: Gender Distribution

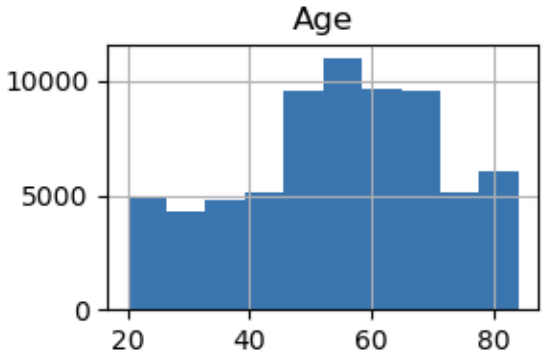

Figure 3: Age Distribution

After training a simple logistic regression model with the data, and analysing the results, we found that the data is trivial. This implies that our data is 'too perfect', which causes classifiers to have nearly 100% accuracy. Since this is not a reliable metric to determine which model will perform best, we will observe the confusion matrix. Specifically focusing on the false negatives. Since false negative classifications could have the most impact, it is important to ensure these are the least common in the confusion matrix of our classifiers. This implies a high cost imbalance. The misclassification of a positive class has a higher cost than the misclassification of a negative class. - add source for heart risk

The data set was split into training and test sets. The training data consists of 80% of the initial data set. The test set contains 20% of the initial data set.

3 Methodology

Since the data is trivial, the classifiers will perform very well. Therefore, we will not only compare their differences in classifying false positives, but also thoroughly analyze the mathematical aspects of these models to see why and how they perform the way they do.

3.1 Logistic Regression

Logistic (log) regression is a machine learning algorithm that completes binary classification tasks. The use of log regression is in line with our goal, since we have two possible outcomes (high risk/low risk). Log regression works well with a linearly separable data set like ours and is relatively simple to implement. The model predicts a binary outcome, 0 for a low risk of heart disease and 1 for a high risk of heart disease. Log regression uses the sigmoid function to transform predictions into probability. The sigmoid function is a formula that turns any number into a value between 0 and 1, kind of like squishing predictions into probabilities.

For our model, we set the decision boundary at 0.5 since our data is balanced. This implies that, given some new instance as input, if the output of the sigmoid function is above our decision boundary (0.5), that specific instance is classified as high risk (1), and low risk (0) otherwise.

3.1.1 Mathematical Process 1: Log Regression

Log regression is based on a linear combination of input features, in our case chest pain, shortness of breath, fatigue, etc. It is followed by the application of the sigmoid function to produce a probability score. Mathematically, the model computes:

$$z = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$

$$\hat{y} = \sigma(z) = \frac{1}{1 + e^{-z}}$$

Here, z is the linear combination of inputs and weights, $\sigma(z)$ is the sigmoid function, and $\hat{y}$ is the output probability of the sample being class 1 (high risk). The logistic model doesn't have layers like a neural network, but applies a single transformation to map the output to a probability. The model starts with a forward pass, computing the linear combination of input features and passing the result through the sigmoid function. This gives us a probability between 0 and 1 for each prediction. To evaluate how well the model performs, the loss is calculated. The log regression model uses the log loss as its loss function. The use of log loss ensures the cost is higher when the model makes confident errors. The weights are then updated using gradient descent [Zou et al., 2019].

3.2 Linear Support Vector Machine

Support Vector Machines (SVM) maximise the margin between different classes by finding an optimal hyperplane. Since the data only contains two classes this hyperplane will be a line. SVMs also support high-dimensional data, which should be noted because the number of features the data set includes is much higher than the number of classes. Lastly, it is less sensitive to outliers by only focusing on its support vectors. The implementation requires principal component analysis (PCA) to visualise the data in a 2D space, as done in Figure 4 in the Appendix section.

3.2.1 Mathematical Process 2: SVM

PCA was used to reduce the number of features to two [HH, 1932].

$$\widetilde{X} = \frac{X - \mu}{\sigma}$$

Data standardisation

$$C = \frac{1}{m}\widetilde{X}^T\widetilde{X}$$

Relationship between data

$$C\,x_i = \lambda_i\,x_i$$

Variance of each component

$$W = [\,x_1,\ x_2\,]$$

Vectors with the most variance

$$\hat{X} = \widetilde{X}\,W$$

The new reduced data set

SVM mathematical definition [Cortes and Vapnik, 1995].

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$

Line definition

$$y_i\big(\mathbf{w}^T\mathbf{x}_i + b\big) \geq 1$$

The decision function

$$\frac{1}{2}\,\|\mathbf{w}\|^2$$

Maximising the margin

$$C\sum \xi_i$$

Penalty term (loss function)

PCA standardises the data, computes variance, and reduces dimensionality by transforming correlated features into uncorrelated principal components. The reduced feature set is then fed to an SVM, which

maximises the margin between classes using the decision function, resulting in a hyperplane (a line in this case).

3.2.2 Hyperparameters and evaluation

The regularisation parameter (C) was set to 1 throughout the project. The model was trained for 100, 1000, and 5000 iterations. During testing it was found that the model performed well on 100 iterations (with 98.09% accuracy) but not on 1000 iterations (with 44.35% accuracy). To ensure that the model generalised the data well,the number of iterations was increased to 5000, which ultimately achieved consistent results. Since it is a linear model, it did not require additional kernel trick-related hyperparameters.

3.3 Decision Tree

A decision tree works by recursively splitting the data into subsets based on feature values. It splits the data in a way that maximises the separation of the target variable (heart risk). We tested our classifier with Gini impurity and information gain.

3.3.1 Mathematical process 3: Decision Tree

The mathematical process involves choosing the best feature and threshold at each node to split the data. This can be done using Gini impurity:

$$Gini(t) = 1 - \sum p_i^2$$

Where p_i is the probability of a sample belonging to class i. Gini impurity shows how likely it is that a randomly chosen sample would be incorrectly classified if we guessed its label based on the class distribution in that node. A lower Gini value means a purer node.

Another approach is using information gain. This measures the reduction in entropy after splitting the data set on a feature. Entropy measures the impurity or uncertainty in a data set. Using entropy, the information gain (IG) from a potential split is calculated as the difference between the parent node's entropy and the weighted sum of the child nodes' entropies. A higher information gain means a better feature split, as it results in purer child nodes [Raileanu and Stoffel, 2004].

$$Entropy(t) = -\sum p_i log_2(p_i) IG = Entropy(parent) - \sum (N_k/N) Entropy(k)$$

Here, N_k is the number of samples in a child node k, and N is the total number of samples in the parent node.

3.3.2 Hyperparameters

For the hyperparamaters of the decision tree, a variety of values was tested. The maximum depth of the tree was adjusted to try and limit the growth of the decision tree. Despite multiple trials, the model's performance remained largely unchanged, indicating that the tree's depth did not significantly affect the results. We did not apply other constraints like minimum samples in a leaf, or the minimum number of samples required to split an internal node to allow the tree to grow freely, but with the potential to overfit. Moreover, the model evaluation was performed using a simple train-test split rather than k-fold cross-validation.

As previous research suggests, hyperparameter tuning tends to be most beneficial for data sets with many classes or nonlinear decision boundaries, whereas default values often suffice for simpler classification tasks. Given the relatively clear decision boundaries in our dataset, extensive tuning does not significantly improve performance [Mantovani et al., 2018].

3.4 Neural Network

A feedforward neural network is conceptually not as complex as the name might indicate. You could say that a neural network is just a 'fancy squiggle fitting machine', just like a linear function, but instead of a line, it generates a specific shape that fits the data. A neural network is nonlinear, and it can take on very

complex shapes. A neural network can capture features that other machine learning models might miss. Also, since the neural network has the ability to calculate the loss across multiple layers, the neural network could potentially converge to a more optimal solution [Islam et al., 2019].

3.4.1 MLP Classifier as a Neural Network

The neural network, as described in this paper, is a multi-layer perceptron, or simply MLP. The MLP is a supervised learning algorithm which learns some function f by training on a dataset. By providing the MLP a set of features and target values, the MLP can learn a nonlinear function approximator for either classification or regression. As mentioned earlier, the task of our models is binary classification. Therefore, the MLP will act as a classifier. This implies that the MLP will classify an instance as the class with the highest probability based on the approximation [Scikit-learn, nd].

3.4.2 Mathematical Process 4: Neural Network

As mentioned earlier, a neural network is just a 'fancy squiggle fitting machine'. It is a function like any other function. To get a better idea of how neural networks work, it is important to examine the different concepts on which a neural network relies.

Structure Neural networks are built by nodes that are connected with edges. These nodes act as perceptrons, and it is common for edges to have a weight applied to them. Neural networks consist of different layers: an input layer, output layer, and a (often multiple) hidden layer(s) [Islam et al., 2019]. It is common for a neural network to have multiple hidden layers. The MLP classifier contains 2 hidden layers. The first hidden layer contains 16 nodes, and the second hidden layer contains 8 nodes [Scikit-learn, nd].

Computation Process A neural network initialises with a forward pass. The model processes the input, and at each layer, the model computes a weighted sum of the inputs, adds a bias, and then applies an activation function. The activation function will produce some output. The ReLU activation function is a common activation function, which is also used by the MLP classifier. The ReLU function is very popular, due to its simplicity, and ability to counter vanishing gradients [Kılıçarslan et al., 2021].

$$\text{ReLU}(x) = \max(0, x)$$

After applying some more calculations, and combining all outcomes, the neural network returns a result.

After the forward pass, the network calculates the loss. This is useful for optimising the weights and biases. Since we have a binary classifier, binary cross-entropy loss, or simply log loss, is used [Ho and Wookey, 2020].

$$\text{loss}(q) = -\sum_{x \in X_Pos} \log\, q_x(Pos) \; - \sum_{x \in X_Neg} \log\, q_x(Neg)$$

Utilising the loss function, the MLP Classifier can calculate the loss for each output for some given input using backpropagation. Backpropagation is the key distinction between a neural network and other ML models. It is a very powerful tool to optimise parameters. Using backpropagation, the neural network computes the gradient of the loss function with respect to each of the network's parameters. This is done using the chain rule of calculus, which propagates the error backward through the network [Islam et al., 2019].

3.4.3 Hyperparamaters

Once the gradients are computed, an optimisation algorithm updates the weights and biases in the opposite direction of the gradient. In the case of the MLP classifier, this optimisation algorithm is Adam. Being able to use the Adam optimisation algorithm benefits the performance of the MLP classifier. Adam has the ability to adapt to the learning rate for each parameter using estimates of the first and second moments of the gradients. By combining AdaGrad, which is great with sparse gradients, and RMSProp, which is effective in non-stationary settings, Adam works efficiently with only first-order gradients while requiring little memory [Kingma and Ba, 2014].

4 Results & Discussion

Table 1: Confusion matrix of each model

Model	TP	FP	FN	TN
LR	6937	55	61	6947
L-SVM	6932	110	85	6873
DT	6866	124	132	6878
NN	6953	58	47	6942

4.1 Logistic Regression

Logistic regression (LR) performed well compared to the other models, with only 61 false negatives and 55 false positives. The model achieved an impressive accuracy of 99.17%. The model relied more heavily on specific features, which makes its predictions more straightforward but less flexible in capturing a wider range of patterns.

4.2 Linear Support Vector Machine

The L-SVM maximises the margin between different classes by finding an optimal hyperplane, which in this case is a line due to the binary nature of the data. The model achieved a high accuracy of 98.61%, but its reliance on a dominant feature limited its flexibility in capturing more complex patterns.

The PCA of the L-SVM's total variance explained 98.90% of the data, with the most important feature being age by a large margin of 90%. In this case, the PCA capture tells us that false positives can easily be recognised by verifying the person's age. This does not mean that the prediction should be ignored, rather, reevaluated.

4.3 Decision Tree

The Decision Tree (DT) achieved 98.08% accuracy with Gini Impurity and 98.17% with Information Gain. Unlike LR and L-SVM, which relied heavily on single dominant features, DT considered multiple factors when making predictions. The Gini impurity model has slightly more false positives compared to the information gain model, which means the Gini impurity model might be a bit more prone to incorrectly classifying low-risk patients as high-risk. On the other hand, the Gini Impurity model performs better with fewer false negatives, meaning that this model is better at correctly classifying high-risk patients and minimising missed diagnoses. Therefore, the Gini Impurity model is preferred due to its slightly lower false negatives, which is crucial in the context of heart disease prediction.

4.4 Neural Network

After training the MLP, the test data was utilised to analyse the performance of the MLP. Again, similar to the other ML models, the classifier was near-perfect. The accuracy of the model is 99.25%. In the PCA reduced version of the decision boundary, shown in Figure 5 in the Appendix, the versatility of the Neural Network is visible. This versatility relies on the mathematical capabilities of a Neural Network. As explained earlier in the Mathematical Process of a Neural Network section, the Neural Network can perform complex computations during backpropagation for each individual run.

4.5 Comparative Analysis

All of the models performed well on the data set. With a false negative rate of 61, logistic regression is the second-best model in terms of correctly identifying high-risk patients. The log regression model's accuracy is slightly better than the L-SVM and Decision Tree (DT). The Decision Tree (DT) classifier generally performed similar to other models. Overall, the MLP outperformed the other models with the highest

accuracy of 99.25%. However, all models benefited from the simplicity of the data, leading to consistent strong performance across all models.

While the impact of a false positive in the context of predicting heart disease is less dangerous than a false negative — which could cause a missed diagnosis — false positives could still cause unnecessary stress for patients and additional medical testing with additional costs [Roth et al., 2018]. Given this, the DT and L-SVM models, which had higher FP-counts, are less desirable if we prioritise precision. The log regression model, however, showed the most optimal result in this context with the lowest FP count, followed by the neural network.

Looking at the results, they are not significant enough to accept our hypothesis which stated: The feedforward neural network will outperform a logistic regression model, decision tree, and linear SVM, due to a neural network's ability to model more complex relationships. Although the neural network does in fact outperform the other models. However, since our synthetic data set is highly structured and linearly separable, it is too trivial. This makes the difference between the models negligible.

5 Conclusion

The purpose of this study was to find out whether a neural network would significantly outperform a linear SVM, decision tree, and log regression model. However, the results show negligible differences, which complicates drawing reliable conclusions.

Although our hypothesis does not hold in the context of this paper, it is still reasonable to say that a neural network is likely the most versatile and reliable model when applied to real-word data, as neural networks have the capacity to adapt to more complex patterns.

The mathematical foundations behind each model, such as gradient descent, impurity metrics in decision trees, and backpropagation in neural networks, all show how the models learn and generalise from data. While these foundations may not have significantly affected results on the data set used for this research, these mathematical foundations could still provide an understanding of why certain models outperform others in more complex cases.

In conclusion, the models performed nearly equally well due to the simple structure of the data, and therefore it is difficult to draw a reliable conclusion. However, neural networks appear to be the most flexible and scalable approach for broader, real-world applications. To confirm this, future work should focus on testing these models on more complex real-world data sets.

References

[Cortes and Vapnik, 1995] Cortes, C. and Vapnik, V. (1995). Support vector network. *Machine Learning*, 20:273–297.

[Ghodeshwar et al., 2023] Ghodeshwar, G. K., Dube, A., and Khobragade, D. (2023). Impact of lifestyle modifications on cardiovascular health: A narrative review. *Cureus*, 15(7):e42616.

[HH, 1932] HH, H. (1932). Analysis of complex statistical variables into principal components. *J. Educat. Psychol.*, 24:417–520.

[Ho and Wookey, 2020] Ho, Y. and Wookey, S. (2020). The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling. *IEEE Access*, 8:4806–4813.

[Islam et al., 2019] Islam, M., Chen, G., and Jin, S. (2019). An overview of neural network. *American Journal of Neural Networks and Applications*, 5(1):7–11.

[Kılıçarslan et al., 2021] Kılıçarslan, S., Adem, K., and Çelik, M. (2021). An overview of the activation functions used in deep learning algorithms. *Journal of New Results in Science*, 10(3):75–88.

[Kingma and Ba, 2014] Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

[Mahesh, 2020] Mahesh, B. (2020). Machine learning algorithms - a review. *International Journal of Science and Research (IJSR)*, 9(1).

[Mantovani et al., 2018] Mantovani, R. G., Horváth, T., Cerri, R., Barbon Junior, S., Vanschoren, J., and de Carvalho, A. C. P. L. F. (2018). An empirical study on hyperparameter tuning of decision trees. *arXiv preprint arXiv:1812.0207*.

[Raileanu and Stoffel, 2004] Raileanu, L. E. and Stoffel, K. (2004). Theoretical comparison between the gini index and information gain criteria. *Annals of Mathematics and Artificial Intelligence*, 41(1):77–93.

[Roth et al., 2018] Roth, A. R., Lazris, A., and Ganatra, S. (2018). Overuse of cardiac testing. *American Family Physician*, 98(10):561–563.

[Scikit-learn, nd] Scikit-learn (n.d.). Neural Network Models (Supervised) (Version 1.2.2) [Documentation]. *scikit-learn*. Retrieved March, 2025, from `https://scikit-learn.org/stable/modules/neural_networks_supervised.html`.

[Zou et al., 2019] Zou, X., Hu, Y., Tian, Z., and Shen, K. (2019). Logistic regression model optimization and case analysis. *IEEE Xplore*.

6 Final Report Information

Group number: 128

Authors:

- Sarah Girgis (2823161)
- Sayuri Chan (2821342)
- Jalen Cyrus (2735249)
- Thierno Sylla (2811424)
- Jayden Kashani Maleki (2819712)

6.1 Software Used

For our project, we made use of Python. We have all been working in separate Jupyter Notebooks. We have made use of the packages Pandas, Matplotlib, Numpy, and Scikit-learn. We have integrated different ML models available on Scikit-learn to develop our models.

6.2 Use of AI tools

We have made use of AI tools such as ChatGPT to help us think of new ideas for the project. It can be very helpful to have an AI that can think along on certain topics. For the writing of the code and report itself, no AI tools were used. We emphasise the importance of actually learning something. However, to improve grammar in the report where necessary, we did utilise AI, mainly Grammarly.

6.3 Source Code

`https://github.com/jkama4/project_ml`

7 Appendix

7.1 Data set

https://www.kaggle.com/datasets/mahatiratusher/heart-disease-risk-prediction-dataset

7.2 Visuals

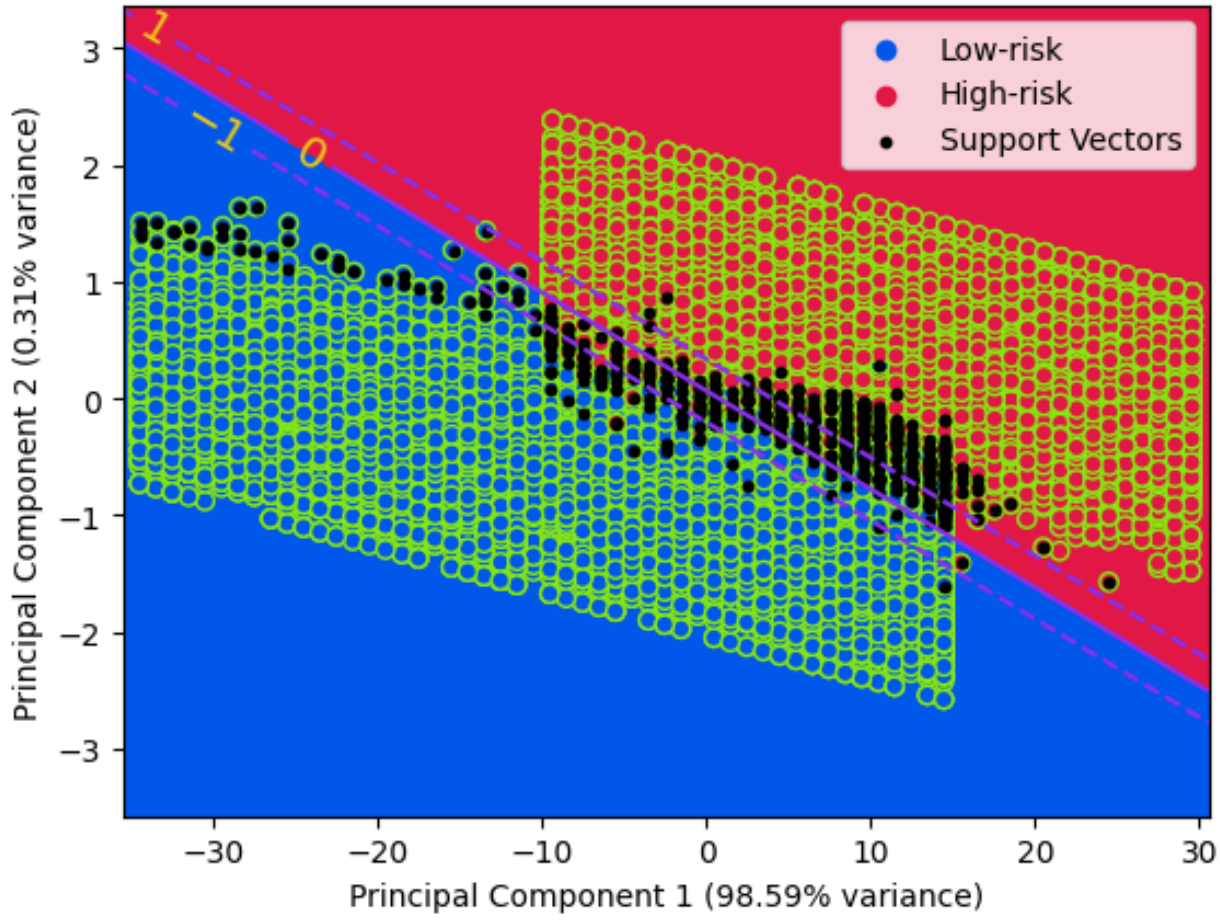

Figure 4: SVM decision boundary with PCA.

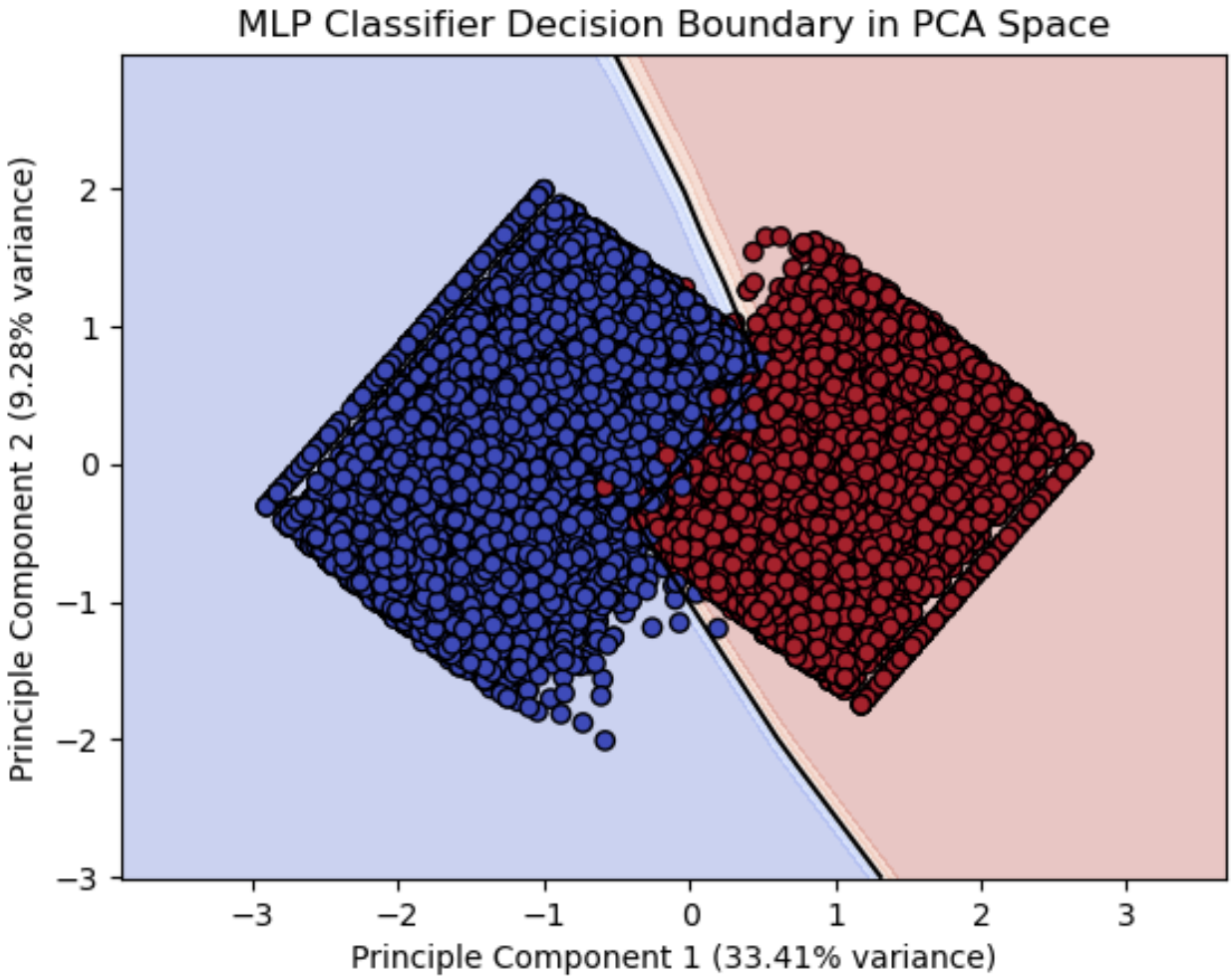

Figure 5: MLP Classifier (Neural Network) decision boundary with PCA.

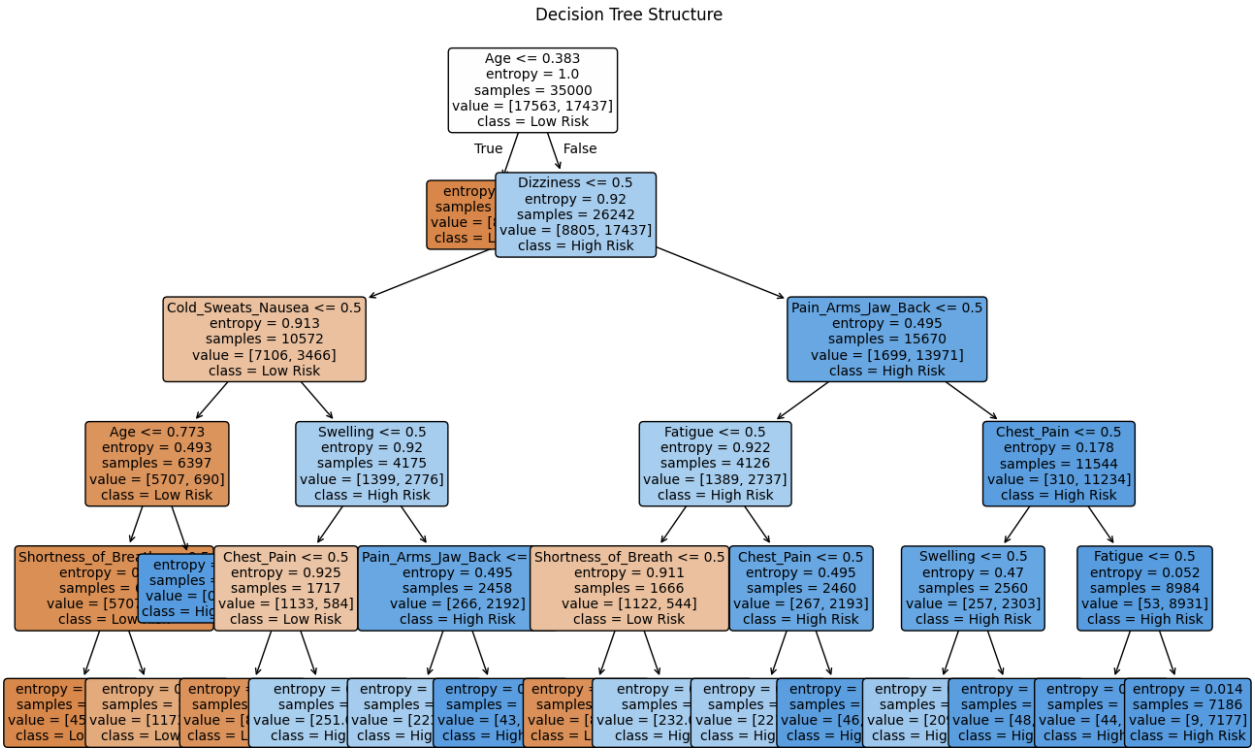

Figure 6: Decision Tree Classifier Visualised

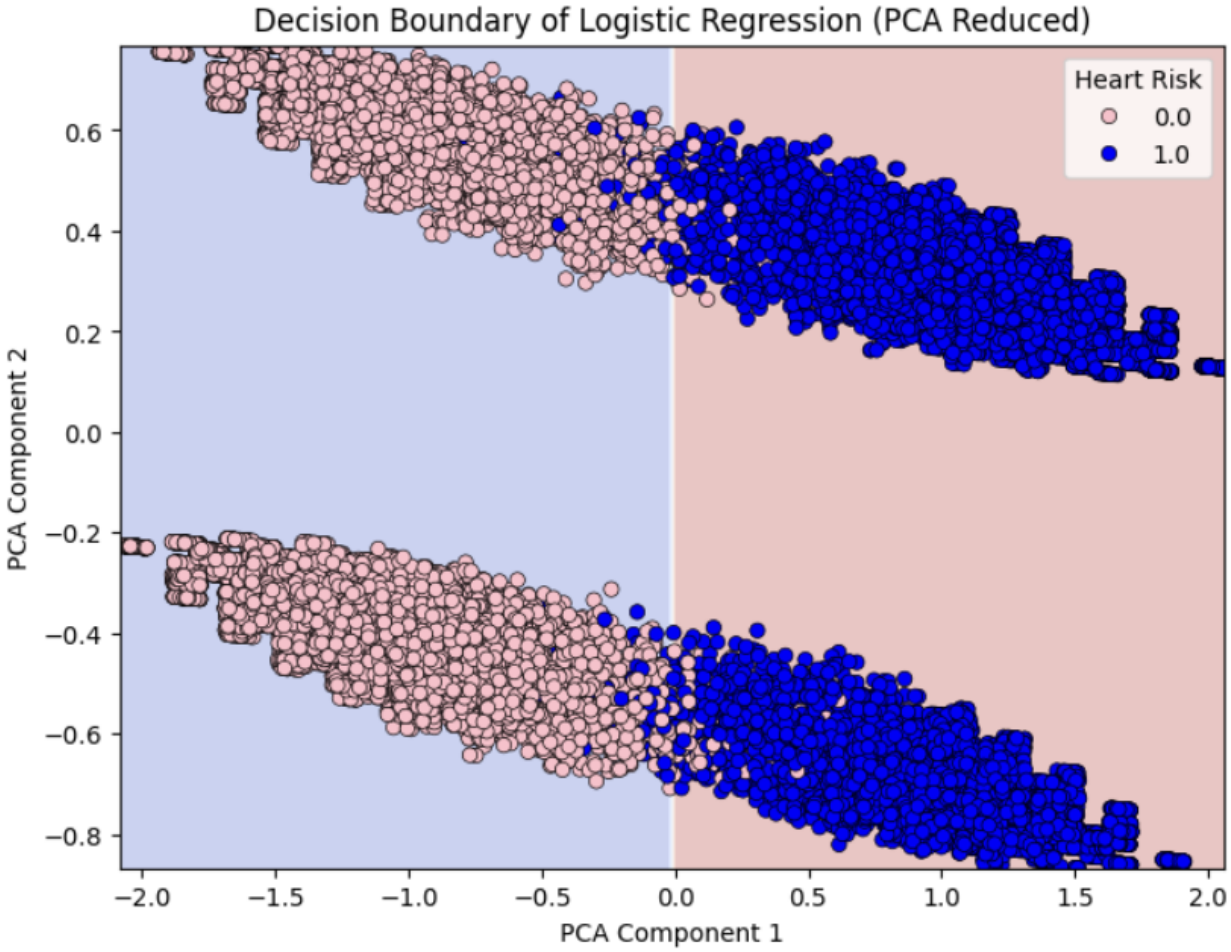

Figure 7: Logistic Regression decision boundary with PCA.