A Comparative Study on Coronary Heart Disease Prediction Through Multilayer Perceptrons and Support Vector Machines

Tasdeed Haq
taz.haq@city.ac.uk

1. Introduction

Cardiovascular Disease (CVD) relates to the conditions affecting the heart or the surrounding blood vessels, in other words, the circulatory system, in which blood is not efficiently pumped properly throughout the body. Examples of CVD amongst others include, coronary heart disease (CHD) and strokes, in which for the latter, the heart cannot receive an adequate amount of blood, which is widely understood to be attributed to the build-up of fatty deposits within the walls of the arteries (atherosclerosis) [1]; these can be caused by many varied factors explained in the following paragraph.

The standard process for diagnosing CVD involves medical practitioners examining available clinical data alongside patient family histories [1][2]. This can be costly, especially where many tests are conducted for example electrocardiograms, stress tests and MRI scans etc., which is therefore also inefficient and slow. Furthermore, decisions made by the practitioner can be potentially subjective causing human error [4]. Where CVDs are the leading cause of death globally, accounting for 32% of deaths in 2019, of which 42% of these deaths were caused by CHD [5][7], it is clear that there is a global need for the accurate and early detection of CVD, resulting in early specialised treatment, thus improving patient care and reduced mortality rates. Unfortunately, accurate diagnosis of CVD is not an easy task given the variety of health factors such as blood pressure and cholesterol levels etc.,[3] as well as demographical factors such as age and gender [3] and symptomatic factors such as if a patient has previously had a stroke [6]. This is where deep learning comes in, to aide in developing a preliminary prediction model to detect patterns in the above risk factors, supporting medical practitioners in making the increasingly complicated diagnosis decision simpler.

The purpose of this paper is to critically evaluate two models with the aim of determining whether a patient has a 10-year risk of future coronary heart disease (CHD), in particular, via the Multilayer Perceptron (MLP) and the Support Vector Machine (SVM) models. We meticulously investigate a wide range of different configurations for each model considering various metrics before selecting a suitable model for this problem.

2. Model Summary

2.1 Multilayer Perceptrons (MLP)

The Multilayer Perceptron is a supervised (initially unsupervised) learning classifier which falls under a class of feedforward neural networks inspired by the biological model of the human brain. It consists of at least three layers: a single input and output layer, and one or more hidden layers which are fully connected through a network of weights and biases between the neurons in each layer, where each neuron contains an activation function. Through training, the hidden layers learn how to extract useful information through a common systematic practice called backpropagation, which combined with optimization techniques such as gradient descent, suitably adjust weights and biases in a way that minimises a specified loss function.

Advantages

Multilayer Perceptron's are universal approximators in the sense that they can approximate any function, no matter how complex, provided a sufficient amount of hidden layers and activation functions, thus making it applicable for linearly and non-linearly separable problems. This combined with the fact MLPs do not make any assumptions of the underlying distribution of the data means that they have a strong capacity to generalise well on unseen data thereby boosting model performance. Multilayer Perceptrons are also highly customizable in terms of additional techniques that can be implemented for example, weight decay and momentum for error terrain exploration, dropout layers as a means to protect against overfitting as well as batch normalization methods to help improve model training.

Disadvantages

It is often the case that MLPs in the presence of hidden layers, that their loss function associated to it is non-convex and as a result, there exists more than one local minimum, some of which are sub-optimal. Hence, multiple different random weight initialisations are required, which whilst time-consuming, also leads to different validation accuracies. MLPs are also prone to overfitting on the training set especially when using complex networks to model a simple phenomenon or using too many epochs during training, hence the need for methods to protect against this such as the early-stopping criterion. In larger networks, the network can be difficult to optimise since the tuning of hyperparameters become computationally expensive, which proves further problematic when dealing with a non-convex loss function.

2.2 Support Vector Machines (SVM)

Support Vector Machines are non-probabilistic, binary linear classifiers, a class of supervised learning methods. Given a set of training examples, SVMs map these examples to points in space in a way that maximises the width of the gap between the two classes. Formally speaking, the aim of SVMs is to develop a hyperplane in high dimensional space, corresponding to the dimensions of the data, in which it is as far away from the nearest training points of each class as possible. These points are called the support vectors, and this distance is called the functional margin – through these conditions, we therefore minimize the generalization error of the classifier as much as possible. New samples are classified depending on what side of the hyperplane they fall.

Advantages
SVMs are relatively memory efficient so as a result, they are good to start with when one has no idea on the data. SVMs naturally work best where there is a clear margin of separation between the classes and tend to be highly effective in high dimensions. The risk of overfitting is also less. One major advantage of SVMs is that they can be extended to cases where classes are not linearly separable in space, like much of the data available nowadays – by mapping the original dimensional space to a higher-dimensional space in a way that makes the separation easier, using the kernel trick, allowing SVMs to be applicable to a wider array of problems. Interestingly, in certain cases, SVMs can outperform Artificial Neural Networks, which we will evaluate here.

Disadvantages
There are many kernel functions available, hence finding the perfect kernel function is not easily done, so methods such as grid searches need to be implemented – furthermore the relevant hyperparameters can prove difficult to fine tune. SVMs also often take time to fit, especially for larger datasets – and since SVMs are not probabilistic models, we cannot explain the classifications in terms of probability like other binary classifier models such as Logistic Regression and ANNs (with a SoftMax function in the outer layer).

3. Dataset

The dataset used is the Framingham Heart Study, extracted from Kaggle [8], which is an ongoing population-based, observational cohort study to investigate the risk factors affecting cardiovascular disease, on residents of the town of Framingham, Massachusetts. Consisting of 4133 patients and fifteen predictors, closely relating to those in Section 1, the aim of the study is to effectively identify the most important risk factors and thereby classify whether the patient has a 10-year risk of developing future coronary heart disease (CHD). Table 1 highlights some summary statistics relating to the dataset once cleaned using techniques in the following paragraph – which highlights a naturally strong imbalance with the ratio of 85:15 of not being diagnosed with CHD to being diagnosed in the next 10 years, a binary output.

An extensive data analysis of the original data was implemented which included but was not limited to exploratory data analysis, outlier analysis, missing value analysis (and imputation using MICE, as this is real-life patient data so smart imputation was necessary) and feature engineering via one-hot encoding; through this rigorous process, this led to the removal of four features including level of education and heart rate in which the latter can be seen as naturally irrelevant.

Some interesting findings were that many features did not have direct relationships to the response i.e., being diagnosed with CHD in the future, other than the naturally well-known predictors such as blood pressure measurements and age. However, for much of these remaining features, they showed substantial changes in the joint distributions with other features when accounting the response. This means that whilst much of our predictors are not important individually, there are deemed important when considered in combination with the other features in the dataset, hence why we have retained many of these original predictors.

feature	kind	count	mean	std	min	25%	50%	75%	max
male	binary	4132.0	0.427396	0.494760	0.00	0.00	0.000	1.0000	1.0
age	continuous	4132.0	49.555421	8.563832	32.00	42.00	49.000	56.0000	70.0
currentSmoker	binary	4132.0	0.494918	0.500035	0.00	0.00	0.000	1.0000	1.0
cigsG0	binary	4132.0	0.505082	0.500035	0.00	0.00	1.000	1.0000	1.0
cigsG1	binary	4132.0	0.151016	0.358108	0.00	0.00	0.000	0.0000	1.0
cigsG2	binary	4132.0	0.249032	0.432505	0.00	0.00	0.000	0.0000	1.0
cigsG3	binary	4132.0	0.094869	0.293070	0.00	0.00	0.000	0.0000	1.0
prevalentStroke	binary	4132.0	0.006050	0.077558	0.00	0.00	0.000	0.0000	1.0
prevalentHyp	binary	4132.0	0.310987	0.462954	0.00	0.00	0.000	1.0000	1.0
diabetes	binary	4132.0	0.025411	0.157390	0.00	0.00	0.000	0.0000	1.0
totChol	continuous	4132.0	236.510411	43.319394	107.00	206.00	234.000	262.0000	439.0
sysBP	continuous	4132.0	132.357575	22.075671	83.50	117.00	128.000	143.5000	295.0
diaBP	continuous	4132.0	82.866893	11.948695	48.00	75.00	82.000	89.5000	142.5
BMI	continuous	4132.0	25.776576	4.076840	15.54	23.06	25.375	27.9825	56.8
glucG0	binary	4132.0	0.213214	0.409627	0.00	0.00	0.000	0.0000	1.0
glucG1	binary	4132.0	0.777832	0.415754	0.00	1.00	1.000	1.0000	1.0
glucG2	binary	4132.0	0.008955	0.094215	0.00	0.00	0.000	0.0000	1.0
TenYearCHD	binary	4132.0	0.151500	0.358579	0.00	0.00	0.000	0.0000	1.0

Table 1 - Summary of Cleaned Dataset. 'cigsG0-cigsG3' and 'glucG0-glucg2' refer to the one-hot encoded versions of the original 'cigarettes smoked per day' and 'glucose levels' features which were grouped due to their noisiness. This is before scaling.

4. Methods

4.1. Data Preparation and Methodology

The dataset will be split into training and test sets in the ratio 80:20 via a stratified split due to the class imbalance. This ensures that the proportions of classes within the test set are representative of the population, since the data based on real patients. In managing the imbalance, the original SMOTE paper suggested random under-sampling on the majority class before applying SMOTE to the minority class. Edited Nearest Neighbours (ENN) will be used as the under-sampling method of choice on the training set to identify and remove points on the class boundary i.e., 'noisy points.' Due to the presence of categorical features, a SMOTE variant denoted SMOTE-NC, will be implemented on the training set, applied to the minority class, leading to a balanced dataset with ratio 50:50. Due to the varying orders of magnitudes in the dataset, a Standard Scaler was deemed appropriate as certain extreme values were kept within the dataset for specific features in Section 3 making the Min-Max Scaler ill-suited. This is all implemented within a pipeline.

In terms of model selection, a grid search will be implemented for the MLP and SVM models as a means to tune hyperparameters. Whilst the data is balanced, one could now use the accuracy as the scoring metric for this, however as the interest is in reducing the false negatives as much as possible (we do not want patients likely to get CHD in the next 10 years to go undiagnosed), the recall was chosen. However, preliminary modelling led to the best models simply diagnosing every patient as having CHD in the future rendering the models useless. Hence the F2 score was used to mitigate this, by putting much more emphasis on the recall whilst still accounting for the precision. For both models, cross-validation will be implemented within the grid search since the size of the dataset would not allow for a validation set, and given time feasibility and computation resources, it was deemed to use 3-fold cross (stratified) cross-validation rather than a 5- or even 10-fold which are more recommended. The hyperparameter combination in which the average validation F2 score is highest will be selected for both models.

Before applying sampling techniques, baseline MLP and SVM models were produced without sampling i.e., with the imbalance, and these were then tuned using the above grid search procedure to produce tuned baseline performance metrics, so that metrics produced by models that implemented sampling could be compared to these. To compare the sampled MLP and SVM models between each other, the chosen models via the grid search are then retrained on the training set, before examining performance on the test set - critically comparing these methods to each other but also to their unsampled counterparts expressed above. For the MLP, an early stopping criterion will be enforced, this ensures that the model keeps training until either the validation loss (validation set comprised of 10% of training data) stops decreasing for a set amount of epochs (via a patience parameter) or it starts to increase. This prevents the model training for too many epochs which could introduce overfitting worsening generalization, but also provides the benefit of allowing the grid search to be completed faster especially where at higher epochs, the validation loss may keep decreasing but by miniscule amounts therefore not improving the model at all.

4.2 MLP Architecture and Parameters

Given that the dataset, once cleaned, has seventeen features, there will be seventeen input nodes, furthermore, since this is a binary classification problem where observations can belong to exactly one class, by default sklearn uses a SoftMax activation on the output nodes. The advantage of this is that the outputs in the output layer add up to one therefore providing a probabilistic decision rule when performing classification. Upon initial model testing, it was found that the data was not linearly separable due to poor metrics (a natural conclusion given the sheer number of predictors), hence it was deemed that hidden layers were required. Typically, for the majority of problems, sufficient performance can be generated using 1 or 2 hidden layers which is what will be assessed here, initially starting with one layer and proceeding to add another layer if performance is inadequate. A baseline model, with sampling showed that the default 100 nodes in a single layered system achieved relatively decent results, however it is often the case that similar or in some cases, better performance can be achieved with less neurons hence a variety of choices were examined through the grid search in multiples of fifties. In terms of activation functions for the hidden layers, with regards to classification, two of the recommended functions are the ReLU and the Sigmoid activations. ReLU overcomes many of the limitations its sigmoid counterpart has, in the sense it is less susceptible to vanishing gradients preventing training; hence this will be considered. The loss function that will be considered is the Cross Entropy Loss, and the optimizer used will be the renowned Adam optimizer in which the learning rate will be tuned. Varying levels of momentum will also be assessed in the hopes that it improves the network by getting out of sub-optimal minima to improve generalizability. Finally, varying batch sizing will be implemented given the fact that it has been shown (empirically) that smaller batch sizes whilst naturally leads to faster training and has better generalization than larger ones.

The full list of hyperparameters being tuned are the learning rate, number of hidden units, batch size and momentum – a model with two hidden layers is done separately.

4.3 SVM Architecture and Parameters

Since initial testing suggested that the data was not linearly separable, it is therefore necessary to employ a wide range of kernels in an attempt to try and successfully separate the data in the context of SVMs. This will be one of our hyperparameters. The second hyper parameter to tune is the regularization parameter, denoted by the letter 'C' or lambda which in a sense tells you how much you want to avoid misclassifying each training example or the 'thickness' between the soft margins. In linearly separable problems, we do not want any misclassifications at all, so we use very narrow distances between the soft margins, however in non-linearly separable cases, we might need to allow for some misclassification i.e., points falling within the soft margins, by decreasing lambda, as this could lead to a better generalization. Furthermore, the higher lambda is, the higher the risk of overfitting since the model chooses more data points as support vectors leading to higher variance and reduced bias, and vice versa, so a balance needs to be found. These are the only major hyperparameters for this model, however there is a special case. Aside from the above two hyperparameters, the Radial Basis Function (RBF) kernel, contains its own hyperparameter, denoted gamma. Roughly speaking, the larger this is, the more linear the decision boundary becomes, increasing the possibility of underfitting, and the closer it gets to zero the more non-linear the boundary becomes which can improve generalization but up to a point, after this it begins to overfit. Thus, this also needs to be tuned when using this kernel.

5. Hypothesis Statement

Our MLP model has many hyperparameters to tune, however the number of different values tested within the grid search is slightly higher to that of the SVM. It is therefore expected that the MLP model will take significantly longer than the SVM and also due to the fact each combination will need to run through multiple epochs, even with early stopping, on each fold. We might further suspect the SVM model to fit more quickly compared to the MLP given the smaller size of the dataset. It is also expected that the optimal SVM model will outperform the optimal single layer MLP, however we expect this to be attributed to the computational cost of not being able to add further combinations in the MLP grid search due to time constraints.

6. Results, Findings and Evaluation

The following table show the results of the grid search and the different values evaluated for each hyperparameter. In total, there were 72 different combinations for the SVM grid search, totalling 216 different fits when implementing 3-fold cross-validation, compared to the MLP which had a larger 270 fits. Chosen hyperparameters are highlighted in bold.

MLP (Single Layered) – Optimizer: Adam, Activation: ReLU, Tolerance: 1e-3, Patience: 8 epochs, Random Seed: 42 (Same Weights + Biases)		**SVM**	
Hyperparameter	**Hyperparameter Ranges**	**Hyperparameter**	**Hyperparameter Ranges**
Learning Rate	[**0.001**, 0.01, 0.1]	Regularization	[0.01, 0.1, **1**, 10, 100, 1000]
Number of Hidden Neurons	[5, 10, 20, 50, **100**, 150]	Kernel	['linear', **'sigmoid'**, 'rbf']
Batch Size	[64, **128**]	Gamma	[0.001, **0.01**, 0.1, 1]
Momentum	[**0**, 0.5, 0.9]		

Table 2 - Grid Search Description and Results, Chosen Hyperparameters in Bold

The time taken for the grid search to complete for the MLP model took 4 minutes and 24 seconds compared to the SVM which took 11 minutes and 40 seconds. This is a surprising result, contradicting our hypothesis, because given the larger number of fits that were required and that each validation fold for each hyperparameter combination would need to go through multiple epochs, it was expected that the times taken for all of these would eventually pile up and surpass that of the SVM. It is suspected that the reasoning for this phenomenon is largely attributed to the use of the early stopping criterion in the MLP model. Even with an increased patience parameter (from 5 to 8, in the event the model falls into a plateau before possibly improving), implying that we train for further epochs before the criterion is enforced, we still get faster time – this suggests that each model in our grid search were fully trained in the early epochs before the tolerance was reached for 8 epochs. Why the models trained so quickly is another question, although there is no clear-cut reason as to why, it could be suspected that this due to a combination of using the Adam optimizer thus quicker convergence at each epoch, a (relatively) smaller dataset and lower batch sizes (relative to 256 and 512).

Table 3 and Figures 4 and 5 highlight relevant performance metrics of the optimal models in comparison to optimal baselines as well as relevant confusion matrices produced on the test set, using our F2 scoring. What we firstly clearly observe are the sheer differences in performance between the unsampled versions of the models compared to the sampled versions. In the unsampled versions, both SVM and MLP models naturally struggled in the classification of patients within the more important minority class, naturally due to the fact that there simply were not enough observations, as reflected by recall and F2 scores. It is clear that our choice of using under sampling in combination with oversampling to repair the imbalances in the data worked well; by introducing artificial points very similar to those within the minority class, we were able to help the model classify this class better.

Model	Class	Precision	Recall	F2-Score	F2-Score (Cross-Validated)
SVM (Unsampled)	0	0.86	0.87	0.87	0.23
	1	0.23	0.22	0.22	
MLP (Unsampled)	0	0.86	0.98	0.95	0.14
	1	0.47	0.12	0.14	
SVM (Sampled)	0	0.92	0.66	0.70	0.50
	1	0.26	0.68	0.51	
MLP (Sampled)	0	0.91	0.68	0.72	0.48
	1	0.26	0.63	0.49	

Table 3 - Performance Metric Comparisons (All Tuned Models) on Test Set – in bold are the most important results (uses SMOTE-NC etc.)

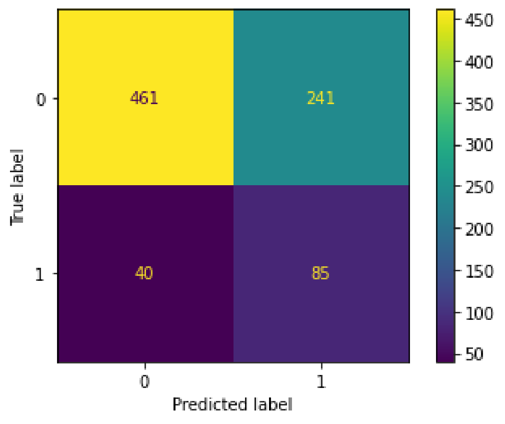


Figure 4 - Tuned SVM Confusion Matrix

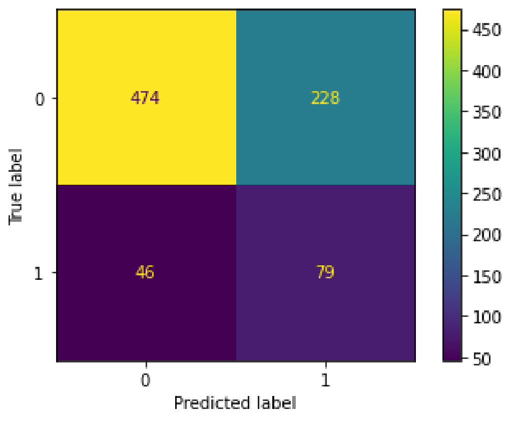


Figure 5 - Tuned MLP Confusion Matrix

Mistakes were initially made whereby when cross validation was implemented on the training set in which sampling methods had been applied on this beforehand, this had the consequence of giving unrepresentative cross validation scores in combination with contradictory and significantly poor results on the test set, this was attributed to data leakage. By implementing a pipeline, this was prevented, and more valid solutions were attained.

Through Table 3, specifically amongst the sampled models, the SVM seemed to outperform the MLP model on not only the minority class but in some cases the majority class. We further see that this is reflected in the Confusion Matrices in Figures 4 and 5, in which for the SVM, we observe more correct classifications on the minority class and more importantly a reduced amount of false negatives which is what we were aiming for. However, for the SVM we have resulted in slightly more false positives though this is not important as firstly, this is fairly marginal in comparison to the MLP model secondly, we deemed this as not as important and thirdly, this is simply the result of the trade-off in minimizing the amount of false negatives. We therefore have one of our hypotheses satisfied.

Metric	SVM	MLP
Training Time	8.34s	1.38s
Prediction Time	0.577s	0.202s

Table 6 - Further Model Metrics

Table 6 considers the times to fit our models as well as predict. What we quickly see is that firstly the SVM model took over 6 times as long to fit compared to the MLP model disproving another one of our hypotheses. This can be attributed to the fact that we used an early stopping criterion for the MLP preventing the training from going on too long, in combination with the fact that the MLP models were extremely quick to fit for reasons to explained earlier. Whilst the times taken to fit are clearly not problematic, further slight adjustments could be made so that the SVM training time could match that of the MLP such as using kernel approximators (e.g., Nystroem), or simply not using all of the training data for example via random under-sampling – since from an SVM perspective, only observations near the decision boundaries are deemed more important. In terms of times to predict, after multiple runs of predicting and measuring the time taken, on average we could see that the MLP model predicted just over twice as fast compared to the SVM model. The speed of prediction for SVM models are affected by two factors, the number of support vectors and the cost of evaluating the kernel matrices - given the fairly small dataset, it could be that there were large amounts of support vectors. The reasoning for why our MLP predicted quickly could possibly be attributed to the simplicity of the network created though no literature could be found supporting alternative reasoning.

We now finally move onto the looking at the ROC curves which consider how well a model is at classifying at different thresholds. These can be seen in Figures 7 and 8. In terms of the AUC score, which is sometimes referred to as a noisy metric, we barely see a difference here. Simply comparing models against this would make them indistinguishable which is unrepresentative. Looking at the ROC Curves, what we notice is that these look too similar to draw anything meaningful from. However, there is one thing that could be seen which is that in Figure 7, for the SVM model, there is a kink which is unique to this plot. At this point, we are able to distinguish some difference between the two, in which we observe that the true positive rate is significantly higher for that given threshold in comparison to the MLP model. Whilst we have noted that the SVM is the better model, at this threshold, we would see further improved performance by this model, although this would of course be marginal. Generally speaking, what we note from these ROC Curves, as could be naturally deduced from previous figures, are that the models do indeed a better job than random guessing which highlights some usefulness in this models, however it must be noted that the models are imperfect, as reflected in these figures which we see are far from perfect or even near perfect models.

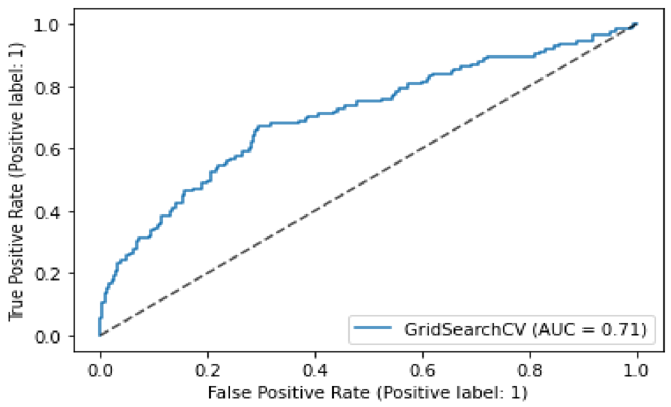


Figure 7 - ROC Curve for SVM Model

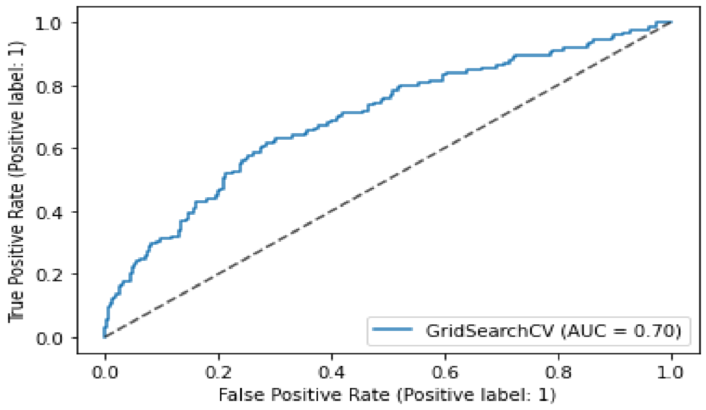


Figure 8 - ROC Curve for MLP Model

As a result, these models would be unsuitable to 'deploy', especially in medical areas. One possible reason why we might observe 'sub-par' performance may be attributed to the fact that whilst we are using the most important features found via deduction, there may be some further underlying factors which also play a role, which would need to be further examined. These additional factors may have helped by increasing the degree of separation between the classes.

Whilst the two tuned model have underperformed in terms of performances required to 'deploy', they have nonetheless outperformed the models that used the data without any sampling A further MLP model was developed which instead contained two hidden layers with the same number of nodes in each layer, this only gave marginal increases in terms of the relevant metrics, with the SVM model interestingly still outperforming this. This is covered in additional implementation details. Based on all of our conclusions, we have deduced that our model of choice is the SVM model.

7. Conclusions, Lessons Learned and Further Study

In this paper, we have evaluated the following two algorithms: MLP and SVM. We concluded with choosing the SVM model as most suited, although both were imperfect and sub-optimal. SVM was chosen given how well in performed across all relevant metrics in comparison to the MLP model, albeit small differences. Whilst the MLP model did do well in certain aspects, these were in fairly minor areas and therefore did not play too much of a role in the decision making.

We reflect on the choice of framework to develop our models on. Whilst sklearn was sufficient for what we planned to do, PyTorch allows more functionality, allowing us to tweak other hyperparameters that were not available in sklearn such as weight decay for our optimizer or even implement additional techniques on the MLP model such as dropout (for regularization) or batch normalization (improving training). This could have changed the scope for the conclusions we found and may have potentially allowed for an even generalizable model - this should be further investigated. This segues into avenues for further analysis. We used a fairly limited grid when hyperparameter tuning in the interest of time feasibility and given computational resources. Adding more entries causes the number of combinations to grow exponentially, as well as the time taken to complete. A similar issue arises with randomized grid searches with large numbers of iterations, hence more efficient applications would be required for effective tuning such as Optuna which uses 'automated early-stopping' methods to prune unpromising trials – this could have led to possibly improved models.

It should be noted that a random seed had been used for the MLP model within the grid search which utilized the same weights and biases for each combination. This allowed for uniformity and fairer comparisons; however, it has been shown that different initializations greatly affect model performance. As such it would be necessary to look at how performance is affected through different initialization methods for example those used by Lecun et al [9]. Other activations functions could be further implemented too, in the MLP model such as those amongst the ELU family, alongside others. The Leaky ReLU does mitigate the 'dying ReLU' issue which can be very problematic especially if there are 'dead' neurons in the first hidden layer, thus a good activation to try next.

References

[1] R. Alizadehsani, M. J. Hosseini, Z. A. Sani, A. Ghandeharioun, and R. Boghrati, 'Diagnosis of Coronary Artery Disease Using Cost-Sensitive Algorithms', in *2012 IEEE 12th International Conference on Data Mining Workshops*, Dec. 2012, pp. 9–16. doi: 10.1109/ICDMW.2012.29.
[2] J. S. Sonawane and D. R. Patil, 'Prediction of heart disease using multilayer perceptron neural network', in *International Conference on Information Communication and Embedded Systems (ICICES2014)*, Feb. 2014, pp. 1–6. doi: 10.1109/ICICES.2014.7033860.
[3] https://www.nhs.uk/conditions/cardiovascular-disease/
[4] V. Sameera, A. Bindra, and G. P. Rath, 'Human errors and their prevention in healthcare', *Journal of Anaesthesiology Clinical Pharmacology*, vol. 37, no. 3, pp. 328–335, Sep. 2021, doi: 10.4103/joacp.JOACP_364_19.
[5] https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)
[6] H. Puri, J. Chaudhary, K. R. Raghavendra, R. Mantri, and K. Bingi, 'Prediction of Heart Stroke Using Support Vector Machine Algorithm', in *2021 8th International Conference on Smart Computing and Communications (ICSCC)*, Jul. 2021, pp. 21–26. doi: 10.1109/ICSCC51209.2021.9528241.
[7] https://www.healthknowledge.org.uk/public-health-textbook/disease-causation-diagnostic/2b-epidemiology-diseases-phs/chronic-diseases/coronary-heart-disease
[8] https://www.kaggle.com/datasets/navink25/framingham
[9] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.

Additional Implementation Details

This report encapsulates nearly the entirely of our implementation details. One thing that was lightly touched upon was the fact that did consider an MLP model with two layers (which was built with the same number of neurons in each layer). Whilst this did yield marginal significant improvements, we still had the SVM model outperforming this, hence no further examination was implemented. Below are the optimal hyperparameters chosen through the grid search as well as similar tables and plots shown in Section 6 such as ROC Curves and Confusion Matrices.

MLP (Two-Layered) – Optimizer: Adam, Activation: ReLU, Tolerance: 1e-3, Patience: 8 epochs, Random Seed: 42 (Same Weights + Biases)	
Hyperparameter	**Hyperparameter Ranges**
Learning Rate	[0.001, 0.01, **0.1**]
Number of Hidden Neurons in Each Layer	[5, 10, 20, 50, **100**, 150]
Batch Size	[64, **128**]
Momentum	[**0**, 0.5, 0.9]

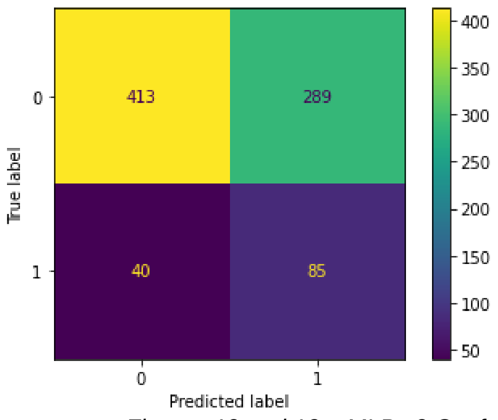
Table 9 – MLP v2 Grid Search Results

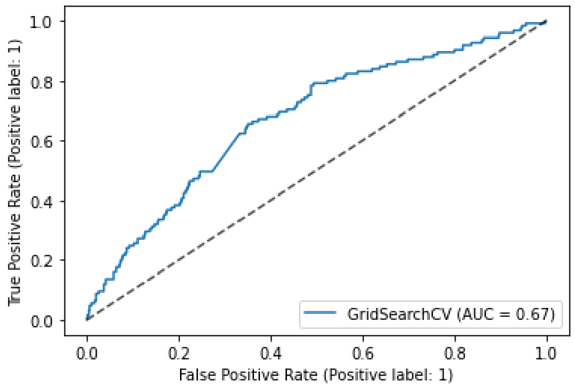
Model	**Class Label**	**Precision**	**Recall**	**F2-Score**	**F2 Score (Cross Validated)**
MLP v2 (Sampled)	0	0.91	0.59	0.63	0.49
	1	0.23	0.68	0.49	

Table 10 – Performance Metrics of Tuned Models on Test Set

Metric	MLP
Training Time	2.98s
Prediction Time	0.213s
Test AUC	0.67

Table 11 - Further Model Metrics





Figures 12 and 13 – MLP v2 Confusion Matrix (Left) and ROC Curve (Right)

(Aside – Some Deductions)

When comparing these metrics to their single layer counterpart, we did see some improved results in minority class in terms of recall, but we further saw a reduction in the recall surprisingly for the majority class too. Training time took twice as long in comparison, which can be attributed to the fact that the model was now more complex, so there were substantial increases in the number of weights and biases trained. Further observation showed that another reason this occurred, was that in comparison, more epochs were implemented during training before the early stopping criterion stopped kicked in, which could be supported naturally by the fact that as model complexity increases, training times take longer. It was surprising to see however, that although the model was more complex, prediction times were still on par with the single layered network, although no literature could be found on why this was the case.

Another worthwhile point to raise is that the grid search took roughly two minutes longer than its single layer version, however this was still less than how long the SVM took, which suggests fast training for previously expressed reasons combined with early stopping.

Taking all of this into account, the marginal increases in performance and the increased training times do not make this two-layered network worthwhile to use, since the single-layered network performed on par with this model whilst also being quicker to train – hence the optimal model to select would still be between the single-layered MLP model and the SVM model – in which we selected the SVM model.

Glossary

- AUC: an evaluation metric that considers all possible classification thresholds, providing the probability that a classifier will be more confident that a randomly chosen positive example is actually positive than that a randomly chosen negative example is positive
- Activation Function: a function, typically non-linear that taken in the weighted sums of all the inputs from the previous layer and produces an output value to the next layer, several examples exist such as those within the ELU family such as Leaky ReLU, ELU and Leaky ReLU alongside others such as sigmoid and hyperbolic tangent
- Backpropagation: the algorithm used to update weights and biases in a neural network when training via gradient descent by propagating the error through the network
- Batch Normalization: a method used to normalize input or output of the activation functions in the hidden layer which stabilizes model training and reduces overfitting
- Cross-Validation: a method used to estimate how well a model would generalize to new data by test one or more non-overlapping subsets withheld from the training set, the number of fold to use are customizable, although a 10-fold is most common
- Early Stopping Criterion: a (regularization) method used to stop model training once the validation set loss begins increasing
- Edited Nearest Neighbours: an under-sampling technique that implements K-nearest neighbours to remove observations close to the decision boundary
- F1-Score: an evaluation metric defined to be the harmonic mean between the precision and recall, this can be modified in cases where precision may be more important (F0.5 score) or recall is more important (F2 score)
- Grid Search – a tuning method that performs an exhaustive search on pre-specified ranges for model hyperparameters in an attempt to find optimal hyperparameters
- Hyperparameters: special model parameters that cannot be optimized using the data, but by the model practitioner which potentially improves model performance, these vary amongst different models, with some container more than others
- Kernel: a function used in SVMs to help solve problems, by providing a short to avoid complex calculations allowing to scale data to high dimensions to enhance separability of classes
- Learning Rate: a scaler used to train models via gradient descent, it provides the step size at each descent iteration. Larger learning rates cause bigger jumps during iterations which could lead to passing the optimal minima
- Loss Function: a function used to determine the error between the output of models and the correct outputs, aim is to minimize this, at each iteration during training, it is expected that the value from the loss function decreases, this value is called the loss
- MICE: (multiple imputation by chained equations) a multiple imputation method used in replacing missing values in data under certain assumptions
- Non-Convex: a function that is neither convex or concave, which can therefore have multiple local minimums and maximums
- Overfitting: a phenomenon where the model matches so closely to the training data, it fails to correctly predict new data
- Precision: an evaluation metric defined to be the ratio between the number of positive samples correctly classified to the total number of samples classified as positive
- Recall: an evaluation metric defined to be the ratio between the number of positive samples correctly classified as positive to the total number of positive samples
- ROC: a curve plotting the true positive rates vs. false positive rates at different thresholds, area under this is the AUC
- SMOTE: an over-sampling techniques used to increase the number of observations in the minority class which utilizes K-Nearest Neighbours
- SMOTE-NC: one of the many variations of SMOTE that is designed to be able to handle both continuous and categorical data
- SoftMax: an activation function used in the outer layer of neural networks that predict a multinomial probability distribution
- Standard Scaler: a method of data standardization that sets a features mean to be zero and scales it to unit variance
- Stratified Split: a method uses when splitting a dataset into training and tests that that preserves the sample proportions of examples in each class as seen in the original dataset
- Tomek Links: an under-sampling technique used to eliminate observations that are close to the decision boundary to help reduce misclassifications
- Weight Decay: a (regularization) technique that adds a penalty term to the loss function leads to the shrinking of the weights during backpropagation