A Comparison of Multilayer Perceptrons and Support Vector Machines Through the Classification of Heart Disease

Fasih Munir
fasih.munir@city.ac.uk

Abstract

This paper will focus on critically evaluating the ability of two machine learning techniques – namely Multilayer Perceptrons (MLP) and Support Vector Machines (SVM) – to effectively classify whether a patient has heart disease or not. Our problem is a binary classification problem with a relatively small number of features available. Four models are compared, two base models and two tuned models (one each for each technique). Base models are generated with simple starting inputs which were randomly selected while the tuned models were selected by tuning select hyperparameters through a grid search method. All models were cross-validated with stratification and results were evaluated by comparing classification reports, confusion matrices and receiver operating curves (ROC). This paper concludes by finding that the Base SVM is better suited for the task

1. Introduction

Cardiovascular diseases (CVD), known as heart disease (HD) from here on, is the global leading cause of death and is one of the most costly family of diseases to treat [1]. Given that those suffering with HD are also at a higher risk of comorbidity [2], being able to diagnose the possibility of HD before it worsens can likely treat patients early and reduce treatment costs.

This paper will compare a Multilayer Perceptron (MLP) to a Support Vector Machine (SVM) using 11 features that are found in a publicly available dataset which we will describe further in section 2. We will test multiple combinations of parameters (tuning) on these 2 models to try and get the best predictions. In our case, we will focus our tuning on precision (or the quality of predictions) which is defined as the ratio of true positives (people with the disease actually predicted to have the disease – correctly predicting the positive class) to total positive predictions (a combination of true positives and false positives). We are using precision because in the scenario of medical conditions, correctly identifying who has the disease is more important than classifying someone without the disease as having it because if the context is reversed, labelling an individual as not having the disease when they actually have it will lead to a worse outcome for them.

We will continue section 1 by introducing the 2 methods used. In section 2 we will discuss the dataset. Section 3 will cover relevant implementation details and methods. Section 4 will evaluate the results and critically reflect on them. Section 5 will conclude this paper on the note that our preferrable model for classifying heart disease is the Base SVM along with learnings and recommendations. The last numbered section will be followed by the references section and the final section will hold the appendices.

1.1. The Multilayer Perceptron (MLP)

MLPs are versatile and can be used for a wide variety of regression and classification tasks. They generally contain 3 main layers, an input layer, a hidden layer (these could be 1 or more) and an output layer. Additionally, each layer holds a number of neurons. A simple perceptron would only have 3 layers [3]. They are also known as feedforward networks due to the nature of how data is processed.

As a summary, the input layer is given our initial data which holds our features that will be used to train the algorithm for prediction. Each feature is neuron. The input layer connects to the hidden layer by passing the input data through with an initial set of weights for them and a bias term. This “forward” pass sums the weights and the hidden layer then applies a non-linear activation function which allows the algorithm to learn more complex patterns. You can think of this as taking points on a straight line and transforming them to fit points on a non-straight line. Data goes through the input layer, through all the hidden layers and then produces an output at the output layer. A loss function is calculated here to measure prediction errors. This error is then

"back propagated" so that the algorithm can go through another forward pass, updating its weights to account for the error. This will continue until we are satisfied by how much loss we want to accept.

While MLPs are extremely versatile and have the ability to learn deep and complex patterns, they do tend to overfit which is a very common problem in machine learning. This is to say that it is able to learn the noise during the training process near perfectly which reduces the generalizability of the model.

1.2. The Support Vector Machine (SVM)

SVMs are also able to be applied on various supervised learning tasks. SVMs use support vectors, hyperplanes and margins. Essentially, we want to find an optimal hyperplane. These are decision boundaries that are created to classify the targets in a dataset (think drawing a line down the middle of a binary dataset). The support vectors are data points that are used to draw this line. The further your data point is from the hyperplane the better, in other words being closer to the hyperplane blurs the decision. This is generally known as maximising the margin [4]. As the dimensions increase (the problem goes from binary class to multi class), visualising this line gets difficult but the concept carries over.

While SVMs are strong with linearly separable data, similar to MLPs, they have the ability to also work with non-linearly separable data by using non-linear mapping functions (known as kernels) such as a polynomial or sigmoid.

SVMs can be applied in many different industries but fall short to scalability issues as the models computation is expensive given that the training matrix scales quadratically to the size of the data [5].

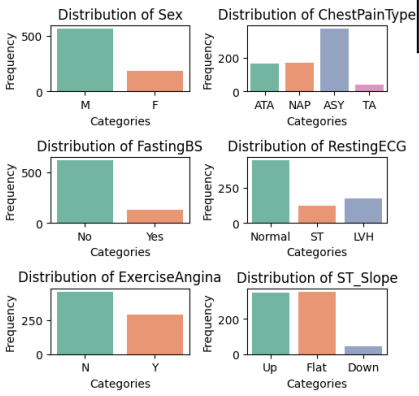
2. The Data

The data that is used in this report is freely available on Kaggle and is known as the "Heart Failure Prediction Dataset" [6]. It is a combination of 5 datasets that are also individually (and freely) available online at the UCI Machine Learning Repository. In total, the data contains 918 unique observations, 11 features (5 continuous and 6 categorical) and 1 target variable ("HeartDisease" – HD). Some inconsistencies were observed in the data, specifically with the features "Oldpeak" and "Cholesterol". Some values for these features were negative and zero respectively. Since these features can not hold these values they were removed from the dataset. One paper [7] suggested imputing the Cholesterol value but this was decided against as that entails introducing fake data which would cause misinterpretation of results. After the removal, we are left with a final sample of 745 observations. Within this sample, we have a relatively balanced target class at 52% (0) to 48% (1). This should negate any fears of bad predictions due to class imbalance, however we will still use stratified k-fold cross validation to ensure model generalizability and maintain our target ratio in each set. Figures 1 and 2 display the variables and basic summaries.

Figure 1

	count	mean	std	min	25%	50%	75%	90%	max
Age	745.0	52.906040	9.489625	28.0	46.0	54.0	59.0	65.0	77.0
RestingBP	745.0	133.036242	17.290451	92.0	120.0	130.0	140.0	158.0	200.0
Cholesterol	745.0	244.747651	59.113689	85.0	208.0	237.0	275.0	310.0	603.0
MaxHR	745.0	140.209396	24.536108	69.0	122.0	140.0	160.0	172.0	202.0
Oldpeak	745.0	0.902953	1.072953	0.0	0.0	0.5	1.5	2.4	6.2

Figure 2



The next steps in exploration were towards understanding outliers. Generally, outliers negatively impact model performance but in the context of healthcare, it is the outliers that are associated with the diseases. Outliers existed for 3 variables, "RestingBP", "Cholesterol" and "Oldpeak". Since we cannot remove them, we need to use a method to minimise their impact and still retain them in the dataset. To achieve this we will use robust scaling that centers data around the median [8]. This scaling will also scale our continuous variables which will help with our chosen models as both are sensitive to different scales.

Our categorical variables have also been scaled which while technically not necessary was done to maintain robustness. The categorical variables were additionally turned into dummy variables with the last dummy of each variable being dropped to combat the problem of multicollinearity. Figure 3 (below) shows the distribution of the continuous variables. Visually we see little correlation between variables which likely means that our SVM may tune to a non-linear kernel. To end our exploration, we found that ratio wise, males are more susceptible to heart disease than females.

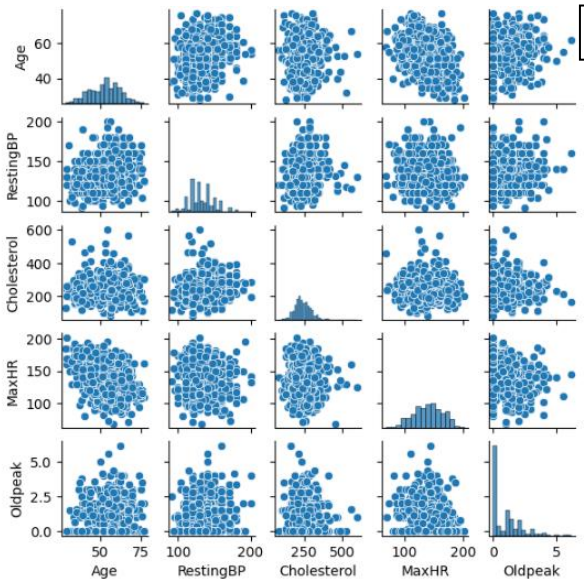


Figure 3

3. Methodology

First we made an initial 15% stratified split of our data to get a train and test set. Using the train set, as described in section 2, we prepared the data by scaling the values since the algorithms are sensitive to scale. These scaling parameters were saved to then be used to scale the final testing data when the time came to test the models. This approach helps avoid any data bleed and keeps the test data completely unseen to the training process. The training data was further split (15%) with stratification into additional train and test (or validation) sets, converted to floats and transformed into arrays to make it easier for the models to use.

Second, we then created 4 models. 2 each for each algorithm that include a Base and a Tuned. The Base models were simpler versions of the tuned models using fewer parameters and randomly selected starting values. The tuned models used a larger set of parameters, the best set being discovered by a gird search. All parameters and values used for the base model were also included here. The models were created in this manner to serve as comparisons for the tuning process. Each tuned model and combination of parameters in the grid search were cross-validated on 5 folds to ensure the models are able to generalize well. The best parameters were chosen on the basis of the best mean precision from the folds. Usually, accuracy is used as a default but in the context of healthcare, since the quality of prediction matters more (as described before) we will use this to pick the best tuned parameters.

Finally, algorithms were trained and then used to get predictions on the test data. Classification reports, confusion matrices and ROC curves were created to compare how well the models did when used on unseen data. We did not test to see if the difference in results were statistically significant.

Random seeds for all models were set as, especially in the case of MLP, the random initialization of weights was greatly influencing the outcome so for this research a seed was set for reproducibility. Note that in production settings this would not be the case as instead you would want to test over a number of random initializations and average across.

Figure 4 shows a summary of the parameters that were tuned. Parameters in brackets were used for the base models.

MLP	
Parameter	**Values**
Hidden Layers (1)	1, 2, 4
Hidden Layer Size (10)	10, 20, 40, 60
Dropout Rate	0.0, 0.1, 0.2
Activation Function (ReLU)	ReLU, Tanh
Learning Rate (0.01)	0.1, 0.01, 0.001, 0.0001

SVM	
Parameter	**Values**
C	0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10
Kernel (linear)	linear, rbg, poly, sigmoid
Gamma	scale, auto, 0.001, 0.01, 0.02, 0.1, 1, 5
Degree	2, 3, 4

Figure 4

3.1. MLP Specifics

We first created a simple 1 hidden layer net for the Base model using a small list of parameters whose values were selected at random. Our input layer consisted of 15 neurons (1 for each feature), the hidden layer had a size of 10 and used a ReLU activation function [11]. Maximum epochs was left at 50 (AUC scores were already coming out to be much higher with a small number of passes through the data so was decided against using a large number of epochs to save on compute cost) and a learning rate of 0.01 was chosen. Given our task was binary, we only had 1 neuron in the output layer that was then passed through a sigmoid layer to get our classification. The Adam optimizer was used to help the model converge [9] and Binary Cross Entropy loss (BCE Loss) was the loss function chosen to minimise during fitting. BCE with Logit Loss could have also been used but given that we have specified a sigmoid layer, we did not use this [10].

During tuning, we decided to focus on the number of hidden layers, the size of the hidden layers, different activation functions, varied learning rates and multiple dropout rates [12][13]. These would not only help with learning speed but also generalizability. In the tuning phase we also adjusted the net slightly. Dropout layers were added after the activation functions and the structure of the hidden layer was kept the same but the number was updated depending on the tuning.

For all models early stopping was used based on the validation loss not improving after 10 epochs. The problem of local minima is reduced using Adam (since it uses momentum), dropout rates and early stopping.

3.2. SVM Specifics

SVMs have fewer parameters to work with. Many regularization terms were tested (C) along with different kernels. The regularization term helps combat overfitting as they work closely to finding support vectors that build the hyperplane. The higher this value the lower the chance of misclassification. Given our data did not show much linearity it is likely that tuning will lead to a non-linear kernel selection although this does not mean that a linear kernel will not preform well. Alongside the kernels, values of gamma and degree are chosen but both are not applied at the same time as they are dependent on the kernel and affect their decision boundaries [14]. Note that degree is only used on a poly kernel and the code will ignore any gamma values in that case.

4. Results and Analysis

Figure 5 below, shows the classification reports of the test (or validation) sets during the training phase. 0 is the negative class (no Heart Disease) and 1 is the positive class (Heart Disease).

Figure 5

Test Classification Report Base MLP:

	precision	recall	f1-score	support
0.0	0.85	0.82	0.84	50
1.0	0.81	0.84	0.83	45
accuracy			0.83	95
macro avg	0.83	0.83	0.83	95
weighted avg	0.83	0.83	0.83	95

Test Classification Report Final Tuned MLP:

	precision	recall	f1-score	support
0.0	0.86	0.86	0.86	50
1.0	0.84	0.84	0.84	45
accuracy			0.85	95
macro avg	0.85	0.85	0.85	95
weighted avg	0.85	0.85	0.85	95

Test Classification Report Base SVM:

	precision	recall	f1-score	support
0.0	0.87	0.80	0.83	50
1.0	0.80	0.87	0.83	45
accuracy			0.83	95
macro avg	0.83	0.83	0.83	95
weighted avg	0.83	0.83	0.83	95

Test Classification Report Final Tuned SVM:

	precision	recall	f1-score	support
0.0	0.57	0.98	0.72	50
1.0	0.89	0.18	0.30	45
accuracy			0.60	95
macro avg	0.73	0.58	0.51	95
weighted avg	0.72	0.60	0.52	95

We can first observe that both base models do surprisingly well with strong overall accuracy, 83% for both models. The results are quite comparable across all point metrics with slight differences. The base models were also simpler to setup and did not require extensive compute such as grid searching making these initial results promising.

However, we see surprising results during the tuning phase. As we can see in Figure 5, the Tuned MLP although improved, does not provide drastically better gains. The F1 score (86% and 84%) suggests a better balance between precision and recall compared to the 2 base models that achieved scores approximately 2 percentage points less. Even though we did not see a greater improvement, given our healthcare context, even a minor improvement is welcomed. The grid search resulted in the best parameter combination on mean precision to be a learning rate of 0.001, a Tanh activation function, a dropout rate of 0.1, a hidden layer size

of 20 and 1 hidden layer. This seems to coincide with what was suggested by Yu et al (2019) [12] that a shallower network of 1 layer outperforms deeper layers given a large layer size. Although the difference for us is that we have a smaller layer size selected by the grid search. This might have fared better if we allowed for searching over greater than 60 hidden layer size.

What is even more surprising is that the Tuned SVM preformed the worst out of all the models even though the Base SVM kernel was passed to the grid search. The initial parameters passed were resulting in a precision of 1 which signalled serious over fitting. To combat this, more values of C were added (which is why there are so many). With the updated parameters as seen in Figure 4, the model's overfitting was reduced but the point metrics worsened dramatically. The high recall of the negative class, high precision of the positive class and low recall of the positive class suggests that the model is being conservative in positive class predictions and is biased towards predicting the negative class more frequently. Given that the AUC scores of all models at this stage were at 90% or more, and the sklearn libraries using a default threshold of 0.5 for point metrics suggests that even though at the default threshold the Tuned SVM is worse, there is a threshold at which it can preform better since the AUC is tested across multiple thresholds. The grid search evaluation suggested the best parameters for the Tuned SVM to be a value of C at 0.1, a gamma of 1 and the rbf (radial basis function) kernel. The results might also happen due to our choice to tune for precision. Please see appendix for results that use a different combination of parameters that were passed in the same grid search that resulted in better performance making it all the more odd why these parameters were chosen*.

The final testing results on unseen data can be seen in figures 6 and 7 below.

Test Classification Report Base MLP:

	precision	recall	f1-score	support
0.0	0.92	0.84	0.88	58
1.0	0.85	0.93	0.88	54
accuracy			0.88	112
macro avg	0.89	0.89	0.88	112
weighted avg	0.89	0.88	0.88	112

Test Classification Report Final Tuned MLP:

	precision	recall	f1-score	support
0.0	0.86	0.84	0.85	58
1.0	0.84	0.85	0.84	54
accuracy			0.85	112
macro avg	0.85	0.85	0.85	112
weighted avg	0.85	0.85	0.85	112

Test Classification Report Base SVM:

	precision	recall	f1-score	support
0.0	0.91	0.84	0.88	58
1.0	0.84	0.91	0.88	54
accuracy			0.88	112
macro avg	0.88	0.88	0.88	112
weighted avg	0.88	0.88	0.88	112

Test Classification Report Final Tuned SVM:

	precision	recall	f1-score	support
0.0	0.57	1.00	0.73	58
1.0	1.00	0.20	0.34	54
accuracy			0.62	112
macro avg	0.79	0.60	0.53	112
weighted avg	0.78	0.62	0.54	112

Figure 6

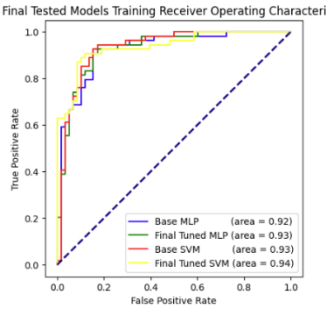


Figure 7

Once again, AUC scores (Figure 7) are high suggesting an optimal threshold could be found which is not the base threshold of 0.5. Looking at the at the classification reports (Figure 6), we see another surprising result where our models have preformed better on the unseen data than in the training phase. Generally, the performance is better during training because the models have "seen" the data, however these results seem to suggest something more about the dataset and its curation.

Firstly the target classes being relatively balanced along with a stratified cross validation strategy seem to have mitigated problems that might arise due to imbalance. Additionally, effort was taken to combat overfitting in the tuned models however the Base models were left without these efforts and have still outperformed on unseen data. Once again pointing towards data that is representative and discernible. This further points towards a lack of underfitting especially since training results were not significantly worse.

Given these results, our two best models are the Base SVM and the Base MLP with the Base SVM being preferred due to its simpler set up.

5. Conclusion

Both, the MLP and SVM (for our context) are favourable algorithms. Given the high possibility that the data is representative, these models have preformed well during both training and testing. Given the simple set up nature of the Base SVM, we would prefer that model over the others as it strikes a good balance between compute cost and results.

However, this study has also highlighted the importance of randomised seeds and chosen parameters. We have learnt that for MLPs, initial weights are important. since weights are randomly initialized, the results, every time the model was run, were different. This prompted us to control the seed for reproducibility. In real world production settings however, this control should likely not be done as you would want to allow for this randomness to improve the robustness of the model. Additionally, the parameter space is extremely important

given the improvement we saw during the MLP tuning and the deterioration during the SVM tuning (even though a better combination existed).

While we kept our parameter space small, given the results, tuning across a larger space is likely better. For this we would then recommend to move away from grid search and use random search instead which will save the computing power and provide a close enough estimate.

For future iterations of this work, we would further recommend to tune the threshold value for the point metrics to improve results and consider a larger dataset than the one we have used to reduce any bias towards the chosen sample and improve reliability of results.

6. References

1. E. O. Lopez and A. Jan, “Cardiovascular Disease,” National Library of Medicine, Aug. 22, 2023. https://www.ncbi.nlm.nih.gov/books/NBK535419/
2. C. Kendir, M. van den Akker, R. Vos, and J. Metsemakers, “Cardiovascular disease patients have increased risk for comorbidity: A cross-sectional study in the Netherlands,” European Journal of General Practice, vol. 24, no. 1, pp. 45–50, Nov. 2017, doi: https://doi.org/10.1080/13814788.2017.1398318.
3. F. Murtagh, “Multilayer perceptrons for classification and regression,” Neurocomputing, vol. 2, no. 5–6, pp. 183–197, Jul. 1991, doi: https://doi.org/10.1016/0925-2312(91)90023-5.
4. A. Shmilovici, “Support Vector Machines,” Data Mining and Knowledge Discovery Handbook, pp. 257–276, 2005, doi: https://doi.org/10.1007/0-387-25465-x_12.
5. J. Cervantes, F. Garcia-Lamont, L. Rodríguez-Mazahua, and A. Lopez, “A comprehensive survey on support vector machine classification: Applications, challenges and trends,” Neurocomputing, vol. 408, no. 1, pp. 189–215, Sep. 2020, doi: https://doi.org/10.1016/j.neucom.2019.10.118.
6. “Heart Failure Prediction Dataset,” www.kaggle.com. https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction/data
7. N. A. Baghdadi, S. Mohammed, A. Malki, I. Gad, A. A. Ewis, and El-Sayed Atlam, “Advanced machine learning techniques for cardiovascular disease early detection and diagnosis,” Journal of Big Data, vol. 10, no. 1, Sep. 2023, doi: https://doi.org/10.1186/s40537-023-00817-1.
8. L. B. V. de Amorim, G. D. C. Cavalcanti, and R. M. O. Cruz, “The choice of scaling technique matters for classification performance,” Applied Soft Computing, vol. 133, p. 109924, Jan. 2023, doi: https://doi.org/10.1016/j.asoc.2022.109924.
9. R. Abdulkadirov, P. Lyakhov, and N. Nagornov, “Survey of Optimization Algorithms in Modern Neural Networks,” Mathematics, vol. 11, no. 11, p. 2466, Jan. 2023, doi: https://doi.org/10.3390/math11112466.
10. B. Shan and Y. Fang, “A Cross Entropy Based Deep Neural Network Model for Road Extraction from Satellite Images,” Entropy, vol. 22, no. 5, p. 535, May 2020, doi: https://doi.org/10.3390/e22050535.
11. Szandała, Tomasz. (2020). Review and Comparison of Commonly Used Activation Functions for Deep Neural Networks. https://www.researchgate.net/publication/344757203_Review_and_Comparison_of_Commonly_Used_Activation_Functions_for_Deep_Neural_Networks
12. H. Yu, D. C. Samuels, Y. Zhao, and Y. Guo, “Architectures and accuracy of artificial neural network for disease classification from omics data,” BMC Genomics, vol. 20, no. 1, Mar. 2019, doi: https://doi.org/10.1186/s12864-019-5546-z.
13. A. Piotrowski, J. J. Napiorkowski, and Agnieszka Piotrowska, “Impact of deep learning-based dropout on shallow neural networks applied to stream temperature modelling,” Earth-Science Reviews, vol. 201, pp. 103076–103076, Feb. 2020, doi: https://doi.org/10.1016/j.earscirev.2019.103076.
14. Shadeed, Intisar & Alwan, Jwan & Abd, Dhafar. (2020). The effect of gamma value on support vector machine performance with different kernels. International Journal of Electrical and Computer Engineering (IJECE). 10. 5497. 10.11591/ijece.v10i5.pp5497-5506.

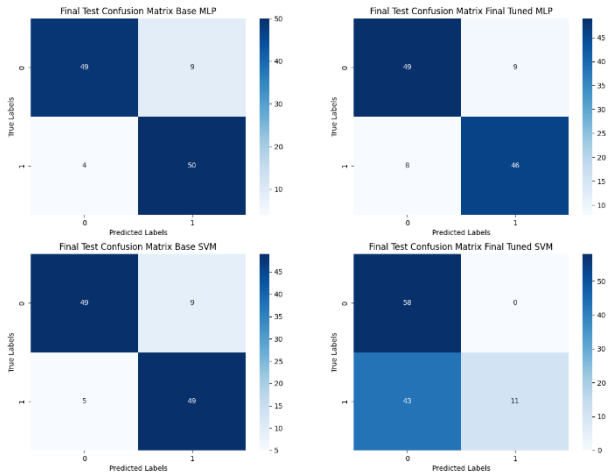
Appendix 1 – Glossary

Term	Definition
Perceptron	Single layer neural network for supervised machine learning
Support Vector	Data points that create and support a decision boundary
Hyperparameter	Parameters set to control the learning process
Grid Search	Method to find the best hyperparameters
Cross Validate	Uses multiple data subsets to evaluate models
Receiver Operating Curve	Graph that shows the false positives against true positives
Precision	True positives to total predicted positives
Feedforward	Data moves in a forward direction from the input in a neural net
Activation Function	Transforms inputs in a neural net to non linear forms for the next layer
Back Propagation	Gradients and loss used to adjust weights back through neural nets
Generalizability	Ability to perform on unseen data
Hyperplane	A line/plane that separates different classes
Decision Boundary	Another word for hyperplane
Kernel	Functions that transform data to higher dimensions. Egs: poly, sigmoid
Gamma	For specific kernels it defines the influence of support vectors
Degree	For specific kernels it defines the influence of support vectors
C	Regularization parameter that balances classification errors
Train Data	Data used to train an algorithm. This data is seen
Test Data	Data used to test an algorithm. This data is unseen
Epoch	One complete pass through the training data in a neural net
Area Under the Curve	A metric tested on multiple thresholds to distinguish between classes
Learning Rate	Used to adjust how much the model weights in a neural net change
Optimizer	Algorithm that minimises a loss function. Eg Adam Optimiser
Loss Function	Measures prediction error. Eg Binary Cross Entropy Loss
Binary Cross Entropy Loss	Specifically measures errors between 2 classes
Hidden Layer	The middles layers in a neural net using activation functions
Dropout Rate	Neurons ignored to avoid overfitting in neural nets while training
Local Minima	A point lower than other nearby points but not the lowest point
Early Stopping	Stopping training if performance does not improve
Momentum	Helps gradient descent algorithms by pushing in relevant direction
Overfitting	Models learning training data very well
Underfitting	Models learning training data not very well
Accuracy	Total predictions correctly classified
F1	Harmonic mean of precision and recall. Represents their balance
Recall	Correctly identified positives versus all actual positives

Appendix 2 – Intermediate Details

- Random seeds were set for reproducibility else results were changing too much
- Had trouble implanting cross validation
    - Initially I was trying to manually create this through a loop but the function was just returning the output for the first fold
    - I then switched to using cross validate function in scikit learn library to make the work easier
- Had trouble building the first perceptron as I was trying to use BCE with Logit Loss. But this meant I lose the ability to get the probabilities as it is applying a sigmoid in its calculation directly which meant I could not calculate an AUC
    - I switched to normal BCE and manually added the sigmoid layer but unsure if I was doing something wrong in the code
    - I also then found that skorch allows for multiple metrics in the call backs in which I could have calculated AUC but I did not test this with BCE with Logit Loss
- I used both cross_validate and cross_val_predict functions but I was ending up with different answers for metrics
    - Upon further reading this has something to do with how they are built
    - Since I needed the classification reports I stuck with cross_val_predict

- Models were converging very fast and did not need a large number of epochs
    - Seeing this I limited the number and did not tune for this
- I had to adjust the parameter grids for MLP multiple times
    - Initially it was because of the time it was taking to train which was crossing into multiple hours
    - Removing the epochs here also helped
        - Best parameters found: {'lr': 0.01, 'module__mlp_tuned_activation_function': <class 'torch.nn.modules.activation.Tanh'>, 'module__mlp_tuned_dropout_rate': 0.1, 'module__mlp_tuned_hidden_layer_size': 40, 'module__mlp_tuned_hidden_layers': 1}
    - After seeing the above I added higher learning rates and more layer sizes to see if these might be preferred
        - Best parameters found: {'lr': 0.0001, 'module__mlp_tuned_activation_function': <class 'torch.nn.modules.activation.ReLU'>, 'module__mlp_tuned_dropout_rate': 0.1, 'module__mlp_tuned_hidden_layer_size': 40, 'module__mlp_tuned_hidden_layers': 1}
    - Not much change in preference was seen apart from activation function and learning rate
    - Final adjustments to the parameter grid lead to the results we see in the main report
- I was also early stopping on validation precision to control even more for precision tuning however this was leading to worse models. I then switched to the loss function mentioned above and improved the models a lot more
- *SVM tuning was quite odd
    - The parameters initially were giving a training precision of 1.0 which signals very serious over fitting
    - It can also predict only 1 class ie the positive class perfectly - which is not the worst thing since we want true positives
    - To combat the over fitting I played with the value of C (the regularization) multiple times
    - C = 0.01 was a very curious case as whenever I left that in the parameter grid the model chose that value and gave imbalanced predictions
    - Similarly a gamma of 1 also tanked the classification report with severe imbalance in the predictions - 1 class predicted much better than the other
    - Not including the above gave the least over fitted result (sensible as it chose a harsher regularization) but the tuning did not improve over the base
    - In fact while the mlp improved slightly, the svm became slightly worse
    - From the same grid a better combination is as follows (also in the training notebook)
        - appendix_final_tuned_svm = SVC(C=0.1, kernel='poly', gamma=1, degree=2, probability=True, random_state=9)
        - this could be due to us calculating precision during cross validation but unsure
        - achieved accuracy of 83% during training



Final Confusion Matrices on Testing Data