

Exploring the effects of hyperparameter changes in a Multi-Layer Perceptron model and building a best model for predicting Parkinson's Disease

Kenny Cai - Assignment 2 ZZSC8536 Data Mining and Machine Learning

Hexamester 5 2020

I. INTRODUCTION

This report explores the effects of altering parameters of a multi-layer perception (MLP) model on the UCI Parkinson's Speech dataset, and outlines and evaluates a *best model*. Due to reasons such as increased computing power and the explosion of data, artificial neural networks (ANNs) have been revived in recent history to solve a variety of complex machine learning problems [1][2].

This report will focus on using shallow MLPs to explore the effects of changing hyperparameters such as the number of hidden layers, number of neurons in the hidden layers, learning rate, different optimizers, and neuron dropout in the hidden layers. The UCI Parkinson's Speech dataset is used to illustrate these changes and the purpose of the *best model* is to develop a model that can predict if a person has Parkinson's Disease based on speech data.

It is important to note that when experiments involving exploring hyperparameters are conducted, the metrics reported are dependant on the data set used. For example, if adding an extra hidden layer in the neural network improves the accuracy of the test set, this may not generalise to a different dataset [3][4].

II. THE DATA SET

The original data taken from the UCI Machine Learning Repository contains speech data from 20 participants with PD and 20 without. For each participant, 26 rows of data were recorded, each for a different task – sounding out a certain vowel, or counting to 10 etc. The dataset also contains 29 columns. The first column being 'Subject ID', 2-27 are speech features, 28 'UPDRS', and 29 being the class information. All the data was numerical¹.

Table 1: Data column breakdown

Columns	Labels
1	Subject ID
2-6	Speech jitter
7-12	Shimmer
13-15	AC, NTH, HTN
16-20	Pitch
21-24	Number of pulses and periods
25-27	Voice breaks
28	UPDRS
29	Class information

Examining the data using pair plots showed many of the features were not good predictors of class (0 negative and 1 positive for Parkinson's Disease). For example, the following pair plots examining "shimmer" for both classes showed major overlapping in "shimmer" in its distributions. There was a slightly higher spread of shimmer for negative classes but overall, the distributions were very similar as seen in the diagonal plots in figure 1.

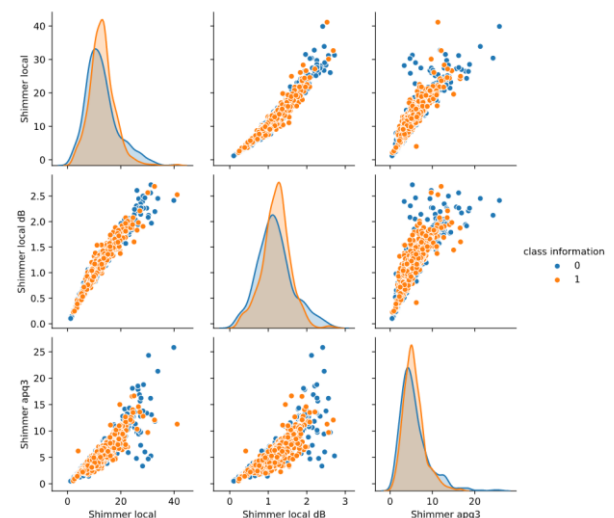


Figure 1: Shimmer and Class distribution

¹ More in-depth information about the dataset can be found at <https://archive.ics.uci.edu/ml/datasets/Parkinson+Speech+Data+et+with++Multiple+Types+of+Sound+Recordings>

Another feature of interest was “pitch”. Participants with PD on average, had lower pitches than their healthy counterparts as seen in figure 2 where the orange dots are grouped much closer together, towards the lower end.

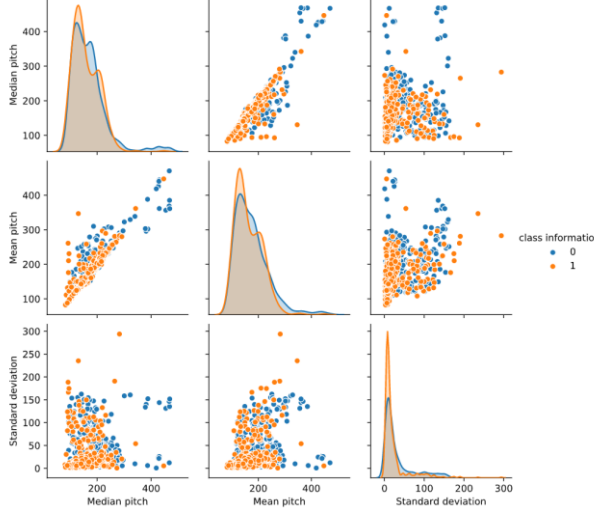


Figure 2: Pitch and class distribution

III. DATA PRE-PROCESSING

In preparation for the experiments, columns 1 “Subject ID” and 28 “UPDRS” were removed from the dataset. UPDRS was dropped because it represented a score that the patient was likely to have PD based on an expert physician. Including this in the dataset would have made our model too accurate and not fit for the purpose of the prediction.

All features were also scaled between 0 and 1. This is one strategy to scale all features around the same number to ensure the weights of the features are not unbalanced. It also helps with the MLP training process given that the value ranges in our features vary considerably [5].

IV. EXPLORING PARAMETERS

A. Methodology

Each row of data was treated as an independent case as it was given an assigned class, as opposed to keeping 26 rows together for each patient. This is done to test whether a single trial measuring the feature set was enough to provide an adequate prediction of PD. Another option not performed here was to combine the 26 rows and provide a single feature value for each patient, then use this for the experiment.

The experiment methodology explored the effects of changing parameters had on the model. Each

combination of parameters was fitted 10 times using the same seed and a mean and 95% confidence interval was reported for the following metrics: accuracy, loss, recall, precision, and AUC. The same seed was used to reduce the variability of the results, which would provide more information as to whether the change in hyperparameter did indeed change the result [6].

In fitting the MLP, the data was split into a 60% and 40% train test split, with the 60% training set being split further into a 60% train and 40% validation set. The 40% test set was used as a hold-out set. The metrics reported were conducted on the test set and training set.

The experiments were conducted in the following order

1. Comparing the optimizers: Adaptive moment estimation (Adam) and Stochastic gradient descent (SGD).
2. Changing the learning rate and momentum rates.
3. Using different numbers of hidden layers.
4. Using different numbers of neurons in each hidden layer.
5. Adding dropouts to the hidden layers.

Finally, the above methodology leads to a search for the *best model* to predict if a patient as PD. The best model is then evaluated using its metrics, confusion matrix, the receiver operating characteristic (ROC) curve and area under the curve (AUC).

B. Results

- i) Changing the optimizer – Adam vs SGD

To illustrate the effects of changing the optimizer, all other parameters were kept constant: number of hidden layers = 1, neurons in hidden layers = 50, learning rate = 0.01. Number of epochs were set to 500 in the SGD experiment to show convergence. See table 2 for results.

Table 2: Results - Optimizers Adam and SGD

Parameters	Mean	Confidence intervals	Observations (Single plot)
Optimizer: Adam Epochs: 200	Accuracy = 0.647 Loss = 0.622 Recall = 0.657 Precision = 0.661 AUC = 0.713	Accuracy = (0.637, 0.657) Loss = (0.616, 0.628) Recall = (0.634, 0.68) Precision = (0.654, 0.668) AUC = (0.708, 0.718)	
Optimizer: SGD Epochs: 500	Accuracy = 0.624 Loss = 0.638 Recall = 0.633 Precision = 0.639 AUC = 0.677	Accuracy = (0.619, 0.629) Loss = (0.636, 0.64) Recall = (0.624, 0.642) Precision = (0.635, 0.643) AUC = (0.672, 0.681)	

Observations:

The results of changing the optimizers showed how quickly the Adam optimizer trained the model to its optimum accuracy. This was done after about 75 epochs whereas the SGD optimizer was a lot slower and trained to its optimal accuracy after about 200 epochs. In the Adam plot, observe that the training accuracy and validation accuracy separate from each other after 50 epochs which indicate that the model begins to overfit. Metrics on

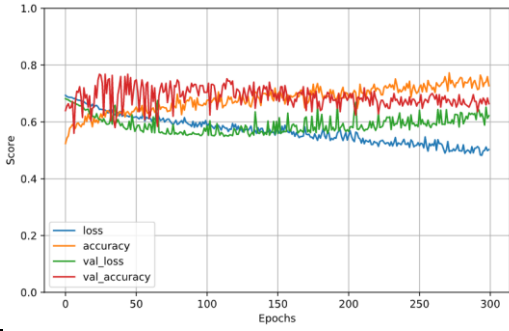
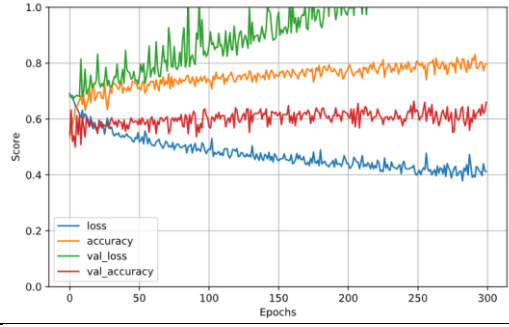
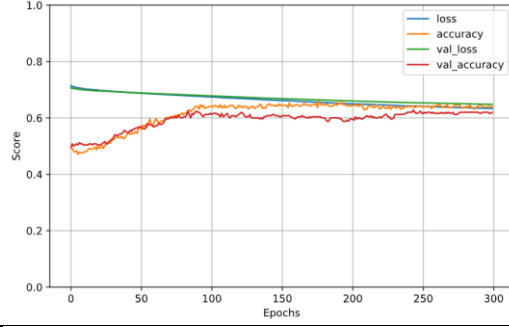
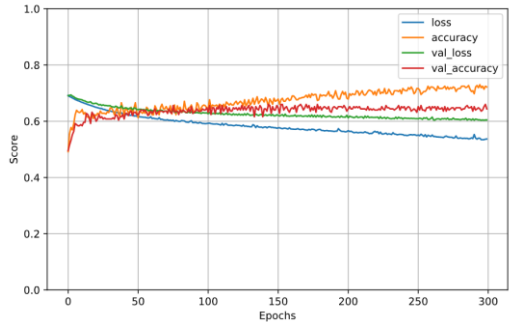
the Adam optimizer were also higher for every metric except the loss.

ii) Changing the learning rate and momentum

For the experiments involving changing the learning rate and momentum, the following parameters were left unchanged: number of hidden layers = 1, neurons in each layer = 50, optimizer = SGD, and epochs = 300. See table 3 for results.

Table 3: Results - Learning and Momentum rates

Parameters	Mean	Confidence intervals	Observations (Single plot)
Learning rate: default (0.01) Momentum rate: 0	Accuracy = 0.596 Loss = 0.665 Recall = 0.628 Precision = 0.578 AUC = 0.646	Accuracy = (0.586, 0.607) Loss = (0.66, 0.67) Recall = (0.611, 0.645) Precision = (0.566, 0.59) AUC = (0.634, 0.658)	

Learning rate: 0.3 Momentum rate: 0	Accuracy = 0.625 Loss = 0.7 Recall = 0.677 Precision = 0.625 AUC = 0.686	Accuracy = (0.602, 0.647) Loss = (0.641, 0.758) Recall = (0.546, 0.809) Precision = (0.609, 0.642) AUC = (0.672, 0.7)	
Learning rate: 0.6 Momentum rate: 0 Epochs: 300	Accuracy = 0.649 Loss = 0.863 Recall = 0.743 Precision = 0.63 AUC = 0.713	Accuracy = (0.631, 0.667) Loss = (0.814, 0.912) Recall = (0.643, 0.844) Precision = (0.605, 0.655) AUC = (0.7, 0.726)	
Learning rate: 0.01 Momentum rate: 0.3	Accuracy = 0.655 Loss = 0.636 Recall = 0.717 Precision = 0.615 AUC = 0.705	Accuracy = (0.649, 0.661) Loss = (0.633, 0.639) Recall = (0.703, 0.731) Precision = (0.608, 0.623) AUC = (0.699, 0.711)	
Learning rate: 0.01 Momentum rate: 0.9	Accuracy = 0.617 Loss = 0.648 Recall = 0.631 Precision = 0.631 AUC = 0.662	Accuracy = (0.61, 0.623) Loss = (0.645, 0.652) Recall = (0.623, 0.64) Precision = (0.625, 0.637) AUC = (0.656, 0.668)	

Observations:

The higher the learning rate, the more chaotic the model training is, as is seen in table 3. This is consistent with theory as the higher the learning rate, the more likely optimising algorithm will overshoot the minima and become unstable while training [7]. This is seen in the graph with a learning rate of 0.3 and 0.6 as the validation loss begins to increase over epochs. In fact, even a learning rate of 0.1 led to divergence while training. When the learning rate was set very small (0.001), the training was very slow and wouldn't converge until many epochs.

Changing the momentum had a similar effect as changing learning rate – the higher the value, the quicker the training would occur, but above a certain value it led to instability and the divergence of validation loss, and separation of the validation accuracy and training accuracy which meant it was overfitting.

iii) Adding more hidden layers

In exploring the addition of hidden layers, the parameters left unchanged were optimizer = Adam and learning rate = 0.001. Epochs were often

changed to show either convergence or divergence. See table 4 for results.

Table 4: Results - adding more hidden layers

Parameters	Mean	Confidence intervals	Observations (Single plot)
No. of hidden layers: 2 Neurons in hidden layers: 1: 20 2: 10 Epochs: 150	Accuracy = 0.657 Loss = 0.622 Recall = 0.676 Precision = 0.668 AUC = 0.72	Accuracy = (0.646, 0.668) Loss = (0.611, 0.632) Recall = (0.645, 0.708) Precision = (0.657, 0.68) AUC = (0.712, 0.728)	
No. of hidden layers: 2 Neurons in hidden layers: 1: 20 2: 20 Epochs: 150	Accuracy = 0.659 Loss = 0.64 Recall = 0.661 Precision = 0.676 AUC = 0.717	Accuracy = (0.652, 0.666) Loss = (0.63, 0.65) Recall = (0.638, 0.685) Precision = (0.667, 0.685) AUC = (0.71, 0.725)	
No. of hidden layers: 2 Neurons in hidden layers: 1: 20 2: 40 Epochs: 150	Accuracy = 0.653 Loss = 0.647 Recall = 0.673 Precision = 0.664 AUC = 0.714	Accuracy = (0.64, 0.666) Loss = (0.63, 0.664) Recall = (0.636, 0.709) Precision = (0.653, 0.676) AUC = (0.704, 0.724)	
No. of hidden layers: 3 Neurons in hidden layers: 1: 20 2: 20 3: 20 Epochs: 50	Accuracy = 0.644 Loss = 0.624 Recall = 0.658 Precision = 0.657 AUC = 0.711	Accuracy = (0.634, 0.654) Loss = (0.617, 0.631) Recall = (0.621, 0.696) Precision = (0.647, 0.667) AUC = (0.704, 0.719)	

No. of hidden layers: 3 Neurons in hidden layers: 1: 20 2: 40 3: 40 Epochs: 50	Accuracy = 0.669 Loss = 0.601 Recall = 0.702 Precision = 0.643 AUC = 0.736	Accuracy = (0.66, 0.679) Loss = (0.592, 0.61) Recall = (0.66, 0.743) Precision = (0.626, 0.66) AUC = (0.728, 0.744)	
No. of hidden layers: 4 Neurons in hidden layers: 1: 20 2: 40 3: 40 4: 20 Epochs: 50	Accuracy = 0.623 Loss = 0.665 Recall = 0.587 Precision = 0.65 AUC = 0.69	Accuracy = (0.614, 0.631) Loss = (0.653, 0.677) Recall = (0.533, 0.64) Precision = (0.63, 0.671) AUC = (0.675, 0.705)	

Observations:

Adding more layers again gave very similar metric scores. It also led to the overfitting of the model a lot quicker. In table 4, the bottom two models were fitted only with 50 epochs and show the divergence of the training accuracy and validation accuracy. This is also due to the number of neurons in the hidden layers. We know that the more weights and bias added to an ANN model, the more likely it is to overfit. The last graph also shows that model with 4 hidden layers and a combination of neurons as 20, 40, 40, 20 causes the validation loss becomes decreases, then begins to increase. This means the

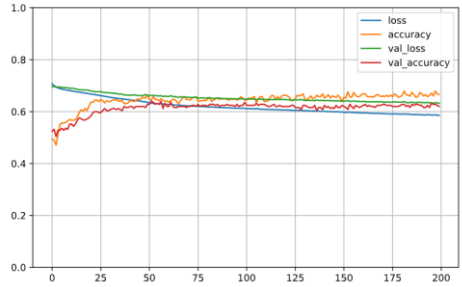
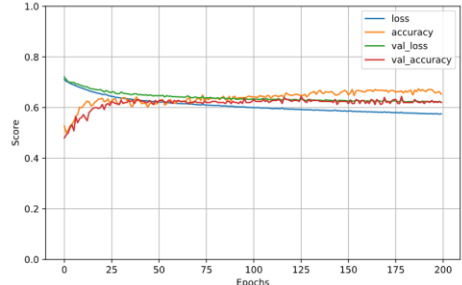
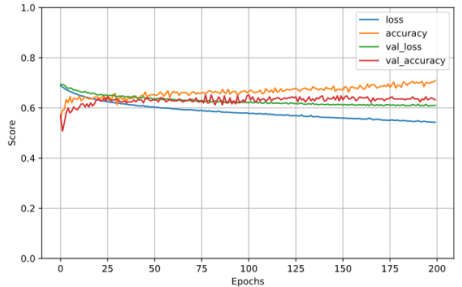
model has become unstable over epochs and is beginning to overfit the data. The combination of adding more layers and the number of hidden neurons is a delicate balance.

iv) *Number of neurons in the hidden layer*

The results in table 5 show the effects of changing the number of neurons in the hidden layer. The parameters that remained constant were epochs = 200, optimizer = Adam, learning rate = 0.001, and number of hidden layers = 1.

Table 5: Results - No. of neurons in the hidden layer

Neurons in hidden layer	Means	Confidence intervals	Observations (Single plot)
5	Accuracy = 0.639 Loss = 0.633 Recall = 0.669 Precision = 0.648 AUC = 0.686	Accuracy = (0.633, 0.645) Loss = (0.63, 0.636) Recall = (0.649, 0.689) Precision = (0.639, 0.657) AUC = (0.682, 0.698)	

10	Accuracy = 0.638 Loss = 0.631 Recall = 0.634 Precision = 0.657 AUC = 0.695	Accuracy = (0.63, 0.646) Loss = (0.626, 0.636) Recall = (0.616, 0.652) Precision = (0.65, 0.664) AUC = (0.689, 0.7)	
20	Accuracy = 0.646 Loss = 0.622 Recall = 0.645 Precision = 0.663 AUC = 0.709	Accuracy = (0.639, 0.653) Loss = (0.616, 0.627) Recall = (0.628, 0.663) Precision = (0.659, 0.668) AUC = (0.704, 0.714)	
50	Accuracy = 0.647 Loss = 0.622 Recall = 0.657 Precision = 0.661 AUC = 0.713	Accuracy = (0.637, 0.657) Loss = (0.616, 0.628) Recall = (0.634, 0.68) Precision = (0.654, 0.668) AUC = (0.708, 0.718)	

Observations:

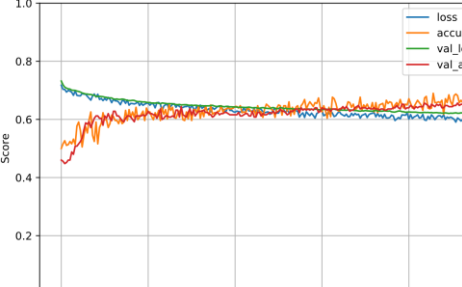
The means for the metrics and 95% confidence intervals were all very similar. The more neurons in the hidden layer, the more quickly the model would train and obtain an optimal accuracy value with the 50 hidden neurons training to optimal validation accuracy at around 25 epochs. As the epochs increased, the models with a higher number of neurons can be seen to overfit the training data, as

evident from the separation between the train accuracy and validation accuracy.

v) Dropouts

To keep the dropout experiments consistent, the parameters that remained constant were neurons in hidden layers = 20, optimizer = Adam, and learning rate = 0.001. The dropout experiments were only undertaken with either 1 or 2 hidden layers. See table 6 for results.

Table 6: Results - Dropouts

Parameters	Mean	Confidence intervals	Observations
No. of hidden layers: 1 Dropout: 0.5 Epochs: 250	Accuracy = 0.642 Loss = 0.619 Recall = 0.682 Precision = 0.648 AUC = 0.705	Accuracy = (0.636, 0.648) Loss = (0.616, 0.622) Recall = (0.671, 0.693) Precision = (0.641, 0.654) AUC = (0.7, 0.709)	

No. of hidden layers: 1 Dropout: 0.2 Epochs: 200	Accuracy = 0.649 Loss = 0.613 Recall = 0.725 Precision = 0.587 AUC = 0.721	Accuracy = (0.644, 0.654) Loss = (0.608, 0.617) Recall = (0.71, 0.74) Precision = (0.582, 0.592) AUC = (0.715, 0.726)	
No. of hidden layers: 2 Dropout: 1: 0.5 2: 0.5 Epochs: 100	Accuracy = 0.648 Loss = 0.627 Recall = 0.743 Precision = 0.629 AUC = 0.704	Accuracy = (0.64, 0.656) Loss = (0.621, 0.633) Recall = (0.754, 0.787) Precision = (0.621, 0.637) AUC = (0.696, 0.711)	

Observations:

Dropouts again had similar metric scores but on average, had higher scores than the other parameters. Even though the model fits were slightly chaotic, the validation and training accuracies remained quite similar which meant the models were not overfitting. This makes sense as dropouts are used as regularisation and may prevent overfitting [8]. Adding dropouts also increased the recall of the model quite significantly – with an increase of approximately 7% above other models.

V. BEST MODEL

Using the lessons learned from the experimental results, a *best model* was chosen given the following criteria: 1) having the highest metric scores, 2) enough epochs needed not to overfit the data, 3) a reasonable number of epochs as to not take up too much computational resources.

Since this is a model for predicting the onset of Parkinson’s Disease, an additional criterion that was considered for the *best model* was the optimisation for recall to reduce the number of false negatives. It would prove practical to detect early onset of PD such that proper prevention and intervention strategies could be implemented, thus prioritising false negatives over false positives.

Refer to table 7 for the hyperparameters and metrics of the *best model*.

Table 7: Best Model

Parameters	Optimizer: Adam Learning rate: 0.0005 No. of hidden layers: 2 Neurons in each hidden layer: 1: 25, 2:15 Dropouts: 1: 0.5, 2: 0.5
Mean	Accuracy = 0.651 Loss = 0.62 Recall = 0.718 Precision = 0.649 AUC = 0.705
95% Confidence interval	Accuracy = (0.645, 0.658) Loss = (0.614, 0.626) Recall = (0.697, 0.738) Precision = (0.64, 0.657) AUC = (0.699, 0.71)

A. Best Model Discussion and Evaluation

i) Rationale behind the hyperparameters

From the experimental results, the Adam optimizer was clearly the better of the two since it trained much faster. The number of neurons in the layers were chosen such that the number of neurons did not exceed the prior layer [9] and the number of

layers were chosen such that the model didn't overfit or underfit. The dropouts were included in each layer to boost the recall score – as evident in our results section, and finally the learning rate was chosen to be half the default Adam learning rate to smoothen the training process.

ii) Confusion Matrix and Classification report

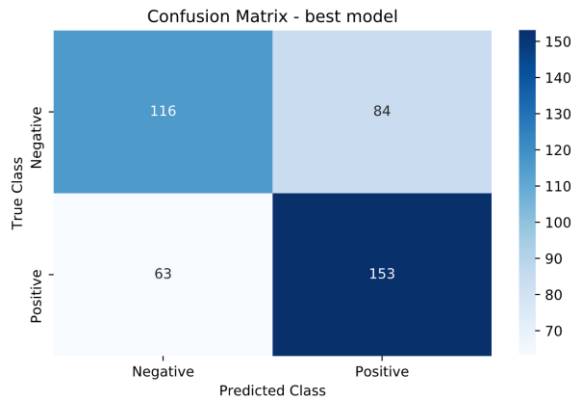


Figure 3: Best model Confusion Matrix

The *best model* had an accuracy of 65%, a recall of 72% and a precision of 65%. The confusion matrix shows that 63 rows of positively classed data was classified as negative (false negatives) by our model, and 84 rows of negatively classed data was classified as positive (false positives). The model performed particularly bad on correctly classifying negative cases getting 42% incorrect, whereas predicting positive cases the model classified 29% incorrectly.

Table 8: Classification report

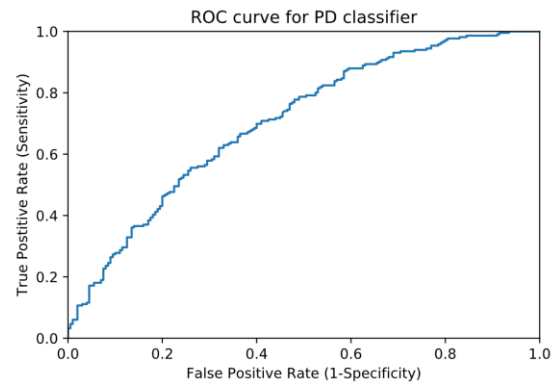
	precision	recall	f1-score	support
1	0.645570	0.708333	0.675497	216.000000
accuracy	0.646635	0.646635	0.646635	0.646635
weighted avg	0.646760	0.646635	0.645035	416.000000
macro avg	0.646807	0.644167	0.643817	416.000000
0	0.648045	0.580000	0.612137	200.000000

iii) ROC Curve and Area Under the Curve (AUC)

The receiver operating characteristic (ROC) curve plots the true positive rate (otherwise known as sensitivity/recall) against the false positive rate. The false positive rate is the ratio of negative classes that are incorrectly predicted as positive, or otherwise defined as $1 - \text{specificity}$ [10]. The ROC curve is provided to enable us to choose thresholds that balance sensitivity and specificity for the

required problem, for example, a sensitivity(recall) of 0.8 gives a specificity of approximately 0.55. Since our problems require us to optimise for some recall, a recall of 0.72 is adequate and the threshold is left unchanged.

Figure 4: ROC curve best model



The AUC Score for the *best model* is 0.705 and represents the ratio of the area under the ROC curve. A perfect model has an AUC score of 1.0 and a purely random classifier would have an AUC of 0.5, so given that our AUC is 0.705, this shows that our model is an improvement over a purely random model.

VI. CONCLUSION AND FURTHER RECOMMENDATIONS

This report explored many effects of changing hyperparameters of an MLP model on a binary classification problem. Training an MLP model has proven to be a delicate balance between all the hyperparameters in its aim to provide the most accurate model for predicting Parkinson's Disease from speech data. The experiment of changing the hyperparameters showed how the models underfit, overfit and became unstable over iterations of training. Even though there wasn't a large improvement in changing most hyperparameters, the best model chosen had an accuracy of 65.1% and an AUC of 70.5% - with the recall of 72% exceeding all the other models.

Further recommendations and improvements to this experiment on the dataset include: running trials using different types of data scaling - for example standardisation of features, or scaling from -1 to 1; exploring other ways to process the data since each person in the dataset used 26 rows and not 1 row as in most other datasets; researching other ways to plot hyperparameter changes; further exploration with using different train test split ratios; and lastly, a more efficient way to find the *best model* –

possibly using sci-kit learn's GridSearchCV, RandomizedCV, and other Keras hyperparameter tuning methods.

REFERENCES

- [1] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*, O'Reilly Media, 2019.
- [2] X. Yao, "Evolving artificial neural networks", *Proceedings of the IEEE*, 87(9), 1423-1447, 1999.
- [3] J. De Villiers, and E. Barnard, "Backpropagation neural nets with one and two hidden layers", *IEEE transactions on neural networks*, 4(1), 136-141, 1999.
- [4] D. Stathakis, "How many hidden layers and nodes?", *International Journal of Remote Sensing*, 30(8), 2133-2147, 2009.
- [5] S. Laskhmanan, *How, When, and why should you Normalise / Standardize / Rescale your data?*, May 16, 2019. Accessed on: September 29, 2020: [Online]. Available: <https://towardsai.net/p/data-science/how-when-and-why-should-you-normalize-standardize-rescale-your-data-3f083def38ff>
- [6] ODSC – Open Data Science, *Properly Setting the Random Seed in ML Experiments. Not as Simple as You Might Imagine*, May 8, 2019. Accessed on: September 30, 2020: [Online]. Available: <https://medium.com/@ODSC/properly-setting-the-random-seed-in-ml-experiments-not-as-simple-as-you-might-imagine-219969c84752>
- [7] L. N. Smith, "A disciplined approach to neural network hyper-parameters: Part 1--learning rate, batch size, momentum, and weight decay", *arXiv preprint arXiv:1803.09820*, 2018.
- [8] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting", *The journal of machine learning research*, 15(1), 1929-1958, 2014
- [9] J. Heaton, *Volume 3 Deep Learning and Neural Networks*, Artificial Intelligence for Humans; Heaton Research: St. Louis, MO, USA, 2015.
- [10] K. Woods, and K. W. Bowyer, "Generating ROC curves for artificial neural networks", *IEEE Transactions on medical imaging*, 16(3), 329-337, 1997.