COL774
Machine Learning
Assignment 3

Atif Anwer
2020EE10479

October 30, 2022

Contents

Libraries Used

- numpy
- scipy
- matplotlib
- nltk
- sklearn
- sys
- pandas
- os
- re
- time
- tqdm
- multiprocessing
- xgboost
- lightgbm

1 Decision Trees, Random Forests and Gradient Boosted Trees

1.1 Mammographic mass lesion severity prediction

Dataset Description

The data consisted of 4 attributes, Age, Shape, Margin and Density of mammographic mass lesions, for which a machine learning model was to be trained to classify them as benign/malignant. The data consisted of missing values at some attributes, which were handled accordingly.

1.1.1 Decision tree classifier

A decision tree provided by scikit-learn was trained on the training split and the following results were obtained.

- *Depth* : 20
- *Training Accuracy* : 92.5275%
- *Validation Accuracy* : 75.2067%
- *Test Accuracy* : 69.9605%

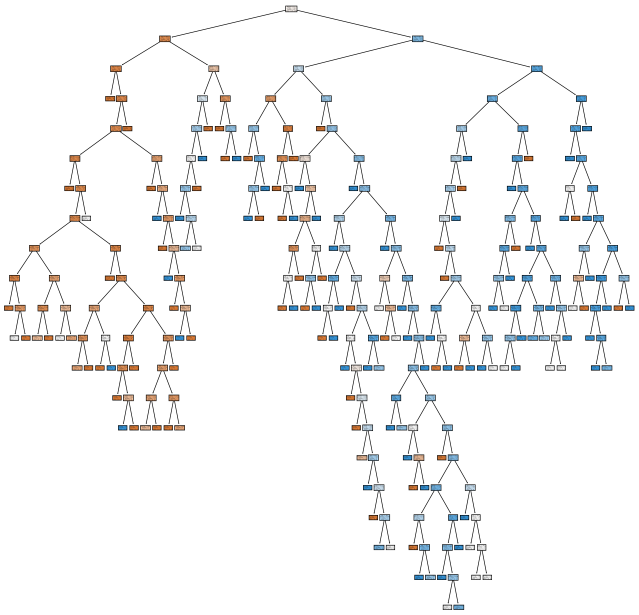


Figure 1: Decision Tree Classifier

The model clearly depicts an overfit on the training set, resulting in a high training accuracy but poor performance on the test set.

1.1.2 Decision tree parameters grid search

Grid search was performed using scikit-learn's *GridSearchCV* for the decision tree classifier on the hyperparameters `max_depth`, `min_samples_split` and `min_samples_leaf`.

Best estimator results

- `max_depth` : 3
- `min_samples_split` : 2
- `min_samples_leaf` : 19
- *Training Accuracy* : 81.0989%
- *Validation Accuracy* : 89.2562%
- *Test Accuracy* : 75.8893%

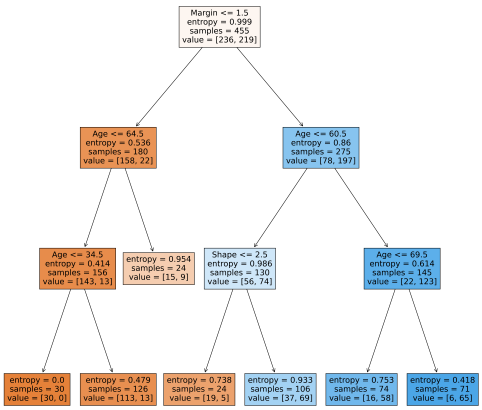


Figure 2: Decision Tree Classifier Grid Search

With the above grid search, the tree's depth has been controlled from overfitting(which was the case in section 1.1.1) on the training data and hence improving in its test performance.

1.1.3 Cost complexity pruning of decision tree

Pruning was performed on the default decision tree using scikit-learn to obtain a range of `ccp_alpha` parameters and corresponding impurities. The variation in parameters with `ccp_alpha` was observed and plotted. The last ccp_alpha value corresponding to a trivial single-node tree was excluded for analysis.

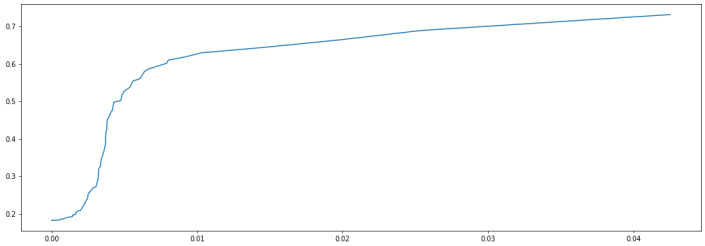


Figure 3: `ccp_alpha` vs. Total impurity on training data

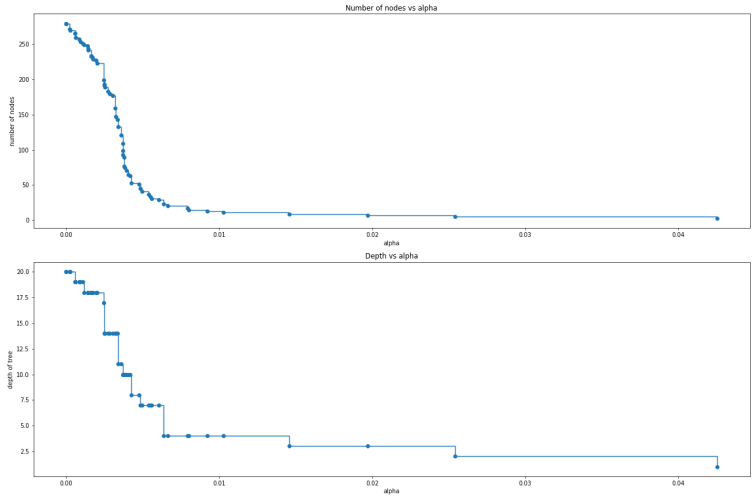


Figure 4: Variation of tree node count and depth with `ccp_alpha`

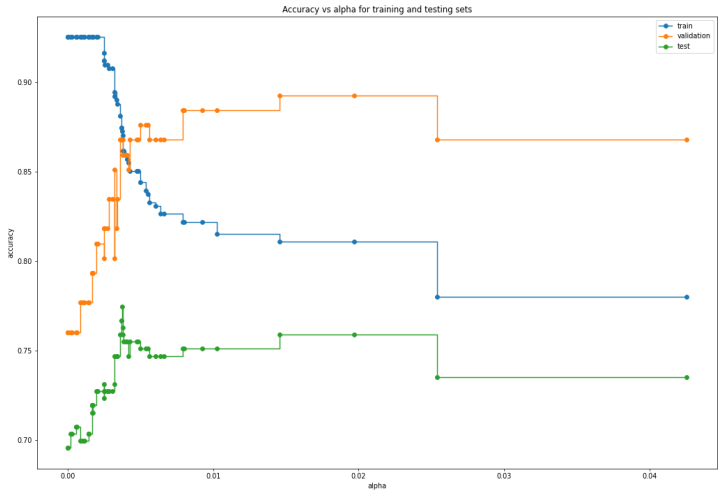


Figure 5: Variation of accuracy with `ccp_alpha`

The best classifier was selected by performance on validation set and the following results were obtained.

- `ccp_alpha` : 0.019707381977522254
- *Depth* : 3
- *Training Accuracy* : 81.0989%
- *Validation Accuracy* : 89.2562%
- *Test Accuracy* : 75.8893%

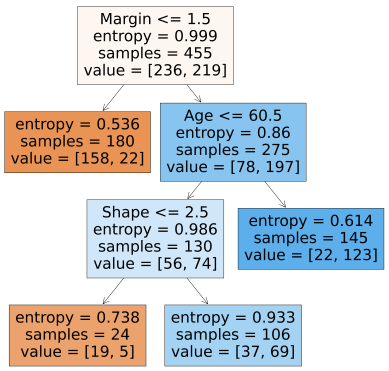


Figure 6: Optimally pruned decision tree

Figure 5 shows the effect of pruning on the validation/test set performance, where the model is prevented from overfitting by impurity-based pruning of the tree, reducing the training accuracy while increasing on test set performance. For higher values of alpha, the model begins to underfit and hence drops in all three accuracies. The model is obviously better than 1.1.1, and has the same metrics as 1.1.2

1.1.4 Random forest

For the optimal random forest model, a grid search was performed on the hyperparameters `n_estimators`, `min_samples_split` and `max_features`.

Best estimator results

- `n_estimators` : 1000
- `max_features` : 2
- `min_samples_split` : 20
- *Training Accuracy* : 83.7362%

- *Out-of-bag Accuracy* : 80.2198%
- *Validation Accuracy* : 87.6033%
- *Test Accuracy* : 77.8656%

Owing to the robustness of random forest and reduction in variance, the model performs better than the decision trees discussed earlier.

1.1.5 Missing Data Imputation

Imputation was performed on the training data using scikit-learn's *SimpleImputer* to fill up the missing data based on median and mode.

Imputation by median

Default decision tree :

- *Depth* : 16
- *Training Accuracy* : 91.8063%
- *Validation Accuracy* : 77.6859%
- *Test Accuracy* : 71.9368%

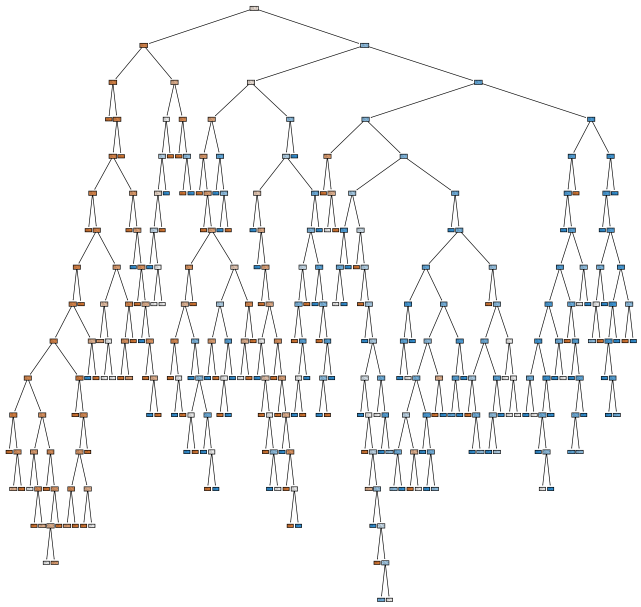


Figure 7: Decision Tree Classifier

Decision tree grid search :

Best estimator results

- `max_depth` : 3
- `min_samples_split` : 2
- `min_samples_leaf` : 12
- *Training Accuracy* : 80.0745%
- *Validation Accuracy* : 87.6033%
- *Test Accuracy* : 77.0751%

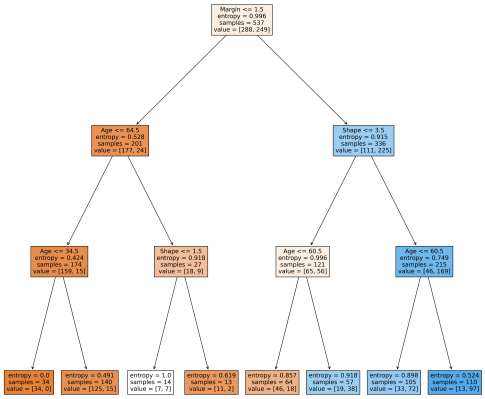


Figure 8: Decision Tree Classifier Grid Search

Cost complexity pruning of default decision tree :

Best estimator results

- `ccp_alpha` : 0.0037673884288719553
- *Depth* : 11
- *Training Accuracy* : 85.8473%
- *Validation Accuracy* : 88.4297%
- *Test Accuracy* : 78.6561%

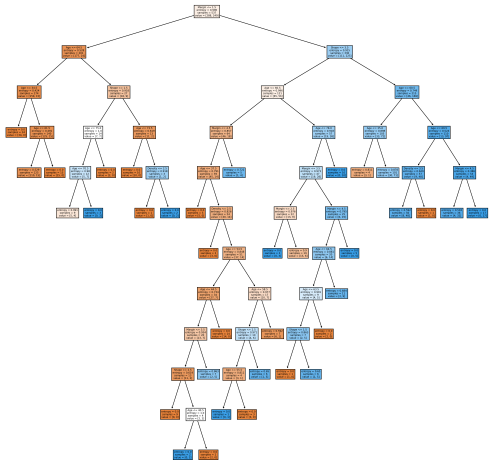


Figure 9: Optimally pruned decision tree

Random forest classifier grid search :

Best estimator results

- `n_estimators` : 1000
- `max_features` : 4
- `min_samples_split` : 15
- *Training Accuracy* : 84.9162%
- *Out-of-bag Accuracy* : 77.8398%
- *Validation Accuracy* : 85.9504%
- *Test Accuracy* : 76.2846%

With the above results, it is clear that with imputation by median, the model learns more about the trend of the data and hence performs significantly better than the case where these data points were ignored.

Imputation by mode

Default decision tree :

- *Depth* : 18
- *Training Accuracy* : 90.6890%
- *Validation Accuracy* : 75.2066%
- *Test Accuracy* : 71.1462%

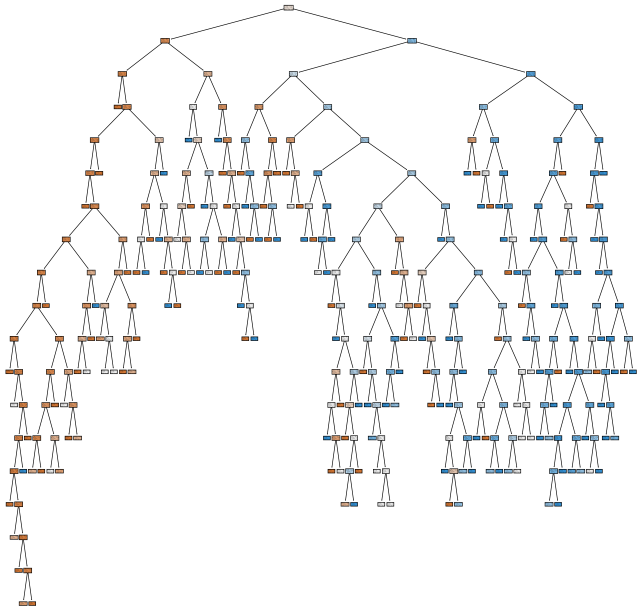


Figure 10: Decision Tree Classifier

Decision tree grid search :

Best estimator results

- `max_depth` : 3
- `min_samples_split` : 2
- `min_samples_leaf` : 19
- *Training Accuracy* : 79.7020%
- *Validation Accuracy* : 89.2562%
- *Test Accuracy* : 75.8893%

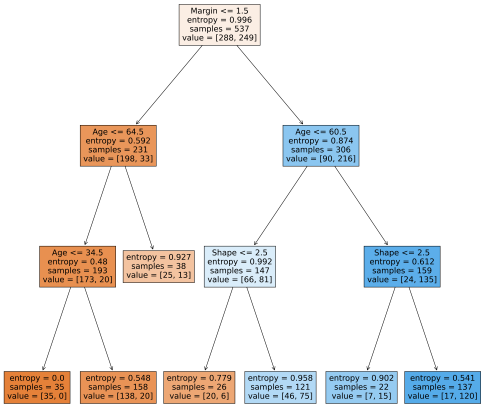


Figure 11: Decision Tree Classifier Grid Search

Cost complexity pruning of default decision tree :

Best estimator results

- `ccp_alpha` : 0.0031882073652631823
- *Depth* : 9
- *Training Accuracy* : 83.4264%
- *Validation Accuracy* : 90.0826%
- *Test Accuracy* : 73.9130%

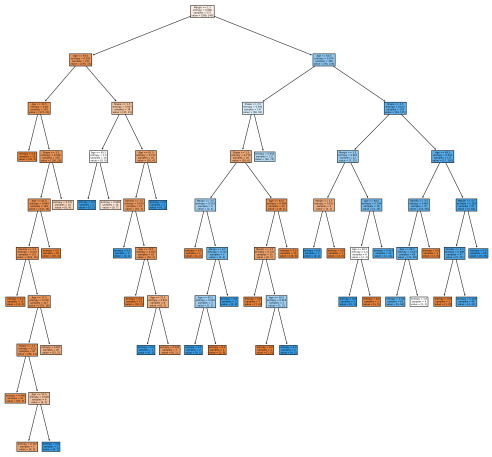


Figure 12: Optimally pruned decision tree

Random forest classifier grid search :
Best estimator results

- `n_estimators` : 1000
- `max_features` : 2
- `min_samples_split` : 15
- *Training Accuracy* : 83.9851%
- *Out-of-bag Accuracy* : 77.2812%
- *Validation Accuracy* : 86.7769%
- *Test Accuracy* : 77.4704%

Once again, with imputation the model learns more about the data and performs better than ignoring the data points with missing attributes. The median imputation results in better performance on the test set in all cases except the random forest, where mode imputation wins.

1.1.6 Gradient boosted trees : XGBoost

Extreme gradient boosting based estimators were constructued using the *xgoost* package and grid search was performed to tune it's hyperparameters.
Best estimator results :

- `n_estimators` : 10
- `max_depth` : 10

- `subsample` : 0.5
- *Training Accuracy* : 83.61266%
- *Out-of-bag Accuracy* : 85.9504%
- *Test Accuracy* : 75.4941%

1.2 Drug rating prediction

Dataset Description

The data consisted of 2 text attributes, review of a drug and the corresponding condition of the patient and 3 numerical attributes, date and useful count. a machine learning model was to be trained to predict the user rating for the drug. The text attributes were converted to numerical data using scikit-learn's *CountVectorizer* after removal of stop-words.

1.2.1 Decision tree classifier

A decision tree provided by scikit-learn was trained on the training split and the following results were obtained.

- *Depth* : 119
- *Training Accuracy* : 100%
- *Validation Accuracy* : 57.3333%
- *Test Accuracy* : 57.1049%
- *Training time* : 109.16s

It can easily be deduced that this is a strong overfit on the training data, causing a low validation and test accuracy.

1.2.2 Decision tree parameters grid search

Grid search was performed using scikit-learn's *GridSearchCV* for the decision tree classifier on the hyperparameters `max_depth`, `min_samples_split` and `min_samples_leaf`.
Best estimator results

- `max_depth` : 120
- `min_samples_split` : 2
- `min_samples_leaf` : 1
- *Training Accuracy* : 100%
- *Validation Accuracy* : 57.1049%
- *Test Accuracy* : 117.6766%

Performing grid search on these hyperparameters made no difference on the resulting decision tree. This suggests that this might be the best that can be done with one decision tree.

1.2.3 Cost complexity pruning of decision tree

Pruning was performed on the default decision tree using scikit-learn to obtain a range of `ccp_alpha` parameters and corresponding impurities. The variation in parameters with `ccp_alpha` was observed and plotted. The last ccp_alpha value corresponding to a trivial single-node tree was excluded for analysis. For the selection of the best model, every 50^{th} value in `ccp_alpha` was used to train and validate considering the huge number of values.

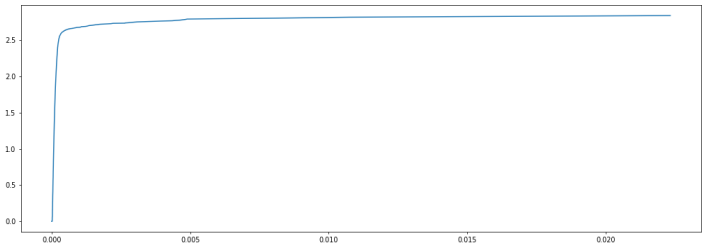


Figure 13: `ccp_alpha` vs. Total impurity on training data

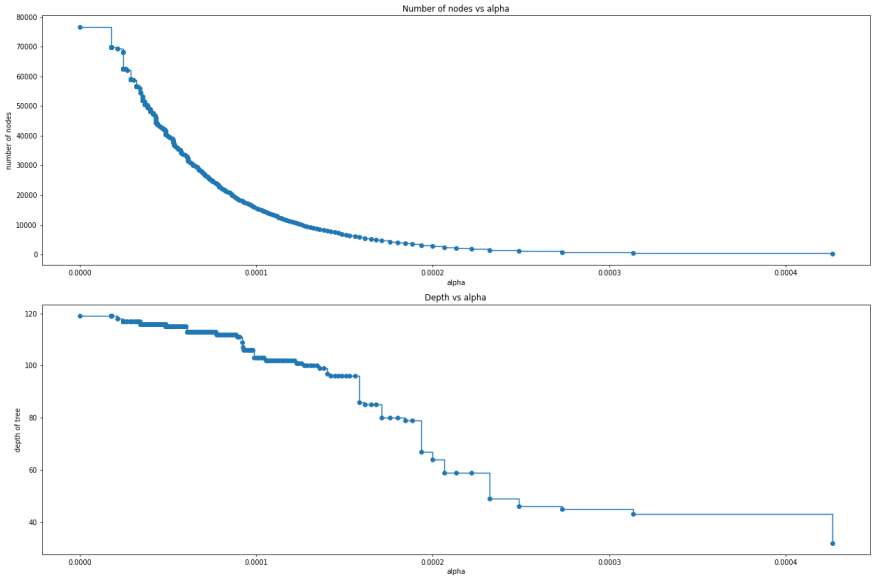


Figure 14: Variation of tree node count and depth with `ccp_alpha`

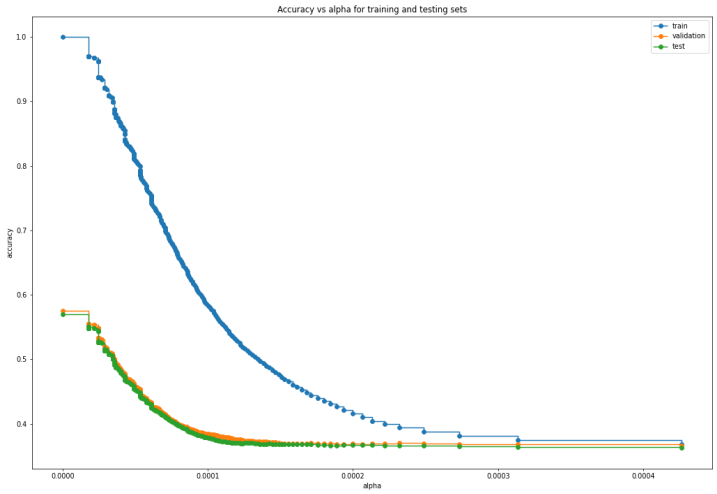


Figure 15: Variation of accuracy with `ccp_alpha`

The best classifier was selected by performance on validation set and the following results were obtained.

- `ccp_alpha` : 0.0
- *Depth* : 119
- *Training Accuracy* : 100.0%
- *Validation Accuracy* : 57.4717%
- *Test Accuracy* : 57.01559%
- *Training time* : 119.69s

The results above allow us to deduce that the decision tree's performance is very sensitive to depth, with training,validation and test sets all following the same trend in view of depth of the tree and a monotonic decrease is observed in all three accuracies with pruning.

1.2.4 Random forest

For the optimal random forest model, a grid search was performed on the hyperparameters `n_estimators`, `min_samples_split` and `max_features`.

Best estimator results

- `n_estimators` : 450
- `max_features` : 0.8
- `min_samples_split` : 2

- *Training Accuracy* : 100.0%
- *Out-of-bag Accuracy* : 64.5889%
- *Validation Accuracy* : 64.3204%
- *Test Accuracy* : 64.1223%
- *Training time* : 652.28s

Moving on to a random forest instead of a single decision tree, we observe a significant rise in test accuracy owing to the reduction in variance in random forests, hence their robustness. The training accuracy still indicates an overfit which might be improved with gradient boosting.

1.2.5 Gradient boosted trees : XGBoost

Extreme gradient boosting based estimators were constructued using the *xgoost* package and grid search was performed to tune it's hyperparameters.

Best estimator results :

- `n_estimators` : 450
- `max_depth` : 40
- `subsample` : 0.4
- *Training Accuracy* : 76.4667%
- *Validation Accuracy* : 53.4502%
- *Test Accuracy* : 53.0521%
- *Training time* : 429.46s

The loss in training accuracy in gradient boosted trees here strongly suggest that the given grid for grid search has an insufficient amount of trees(`n_estimators`).

1.2.6 Gradient boosted machines : LightGBM

Gradient boosted machines based estimators were constructued using the *lightgbm* package and grid search was performed to tune it's hyperparameters.
Best estimator parameters :

Best estimator results :

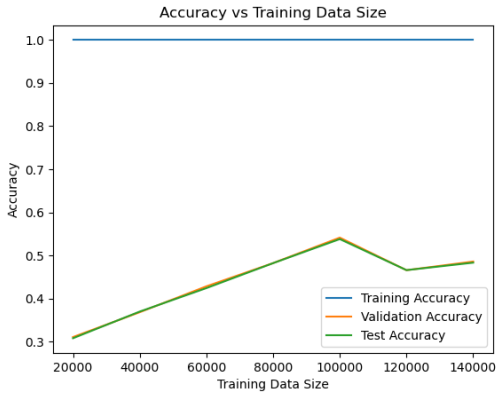
- `n_estimators` : 2000
- `max_depth` : 40
- `subsample` : 0.4
- *Training Accuracy* : 97.1269%
- *Validation Accuracy* : 64.4775%

- *Test Accuracy* : 64.1855%
- *Training time* : 185.71s

Owing to the scalability to large datasets of gradient boosted machines, LightGBM perferms the best among all the previous models, with a visible reduction in overfit on training set when compared to the random forest model in section 1.2.4.
Comparing the training times of the models discussed above, the decision tree models obviously train faster, with pruning and parameter fixing taking closely the same times, while taking a few more seconds than the default decision tree. Among the multi-estimator models, random forest performs the slowest with parameter fixing while XGBoost performs slightly better. LightGBM achieves training times close to decision tree while performing better on the test data too.

1.2.7 Training with varying amount of data

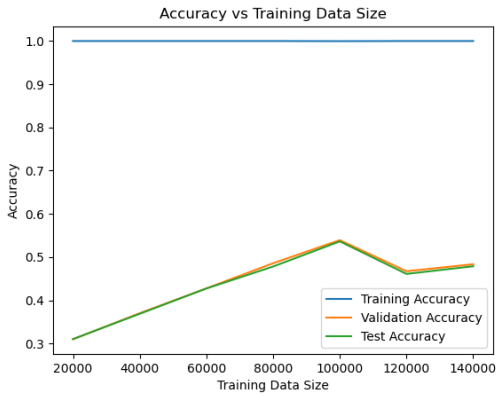
The models discussed earlier were trained with different training data sizes, each synthesized by randomly sampling data points for the required size(with replacement for sizes above the available training data) from the training set. The variation in accuracy and training time was observed as follows.



(a) Accuracy variance

(b) Training time variance

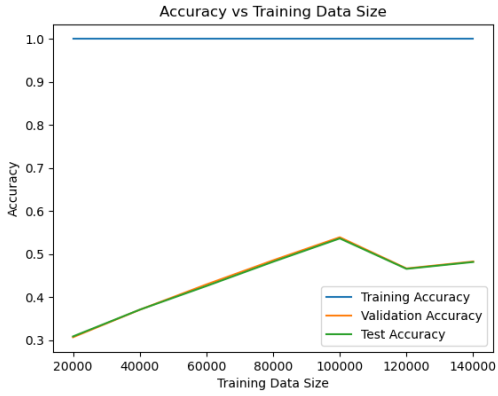
Figure 16: Decision tree classifier



(a) Accuracy variance

(b) Training time variance

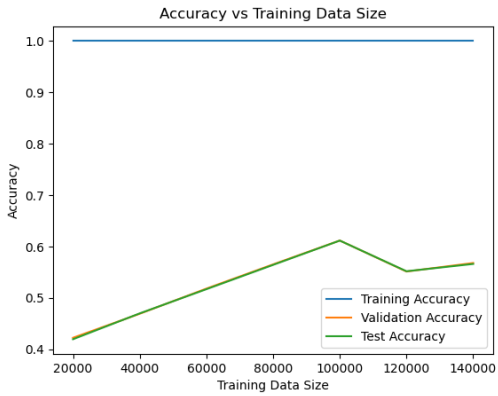
Figure 17: Decision tree classifier parameter grid search



(a) Accuracy variance

(b) Training time variance

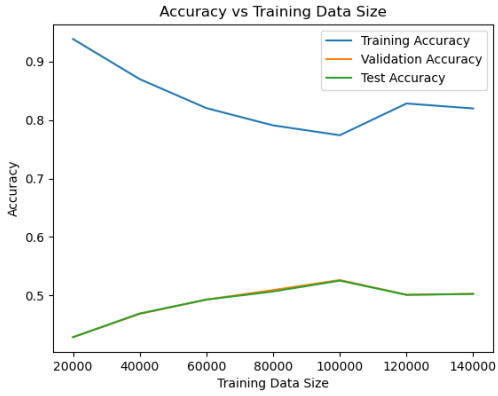
Figure 18: Decision tree classifier cost complexity pruning



(a) Accuracy variance

(b) Training time variance

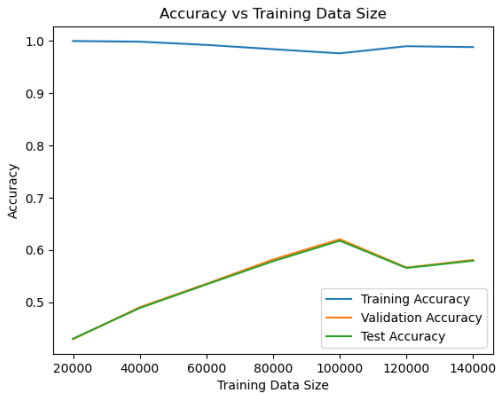
Figure 19: Random forest classifier grid search



(a) Accuracy variance

(b) Training time variance

Figure 20: XGBoost classifier grid search



(a) Accuracy variance

(b) Training time variance

Figure 21: LightGBM classifier grid search

With the above observations, it is clear that the models follow the same trend with the sampling described. It can be gererally said that any model gains more information with the amount of data and hence performs better on the test data. The trade off here is obviously the time take to train on large datasets, bringing it down to the computational strength one holds.

2 Neural Networks

Dataset description

The Fasion MNIST data consisted of 70000 28×28 images flattened and split into training and test sets. A neural network had to be constructed to classify the samples into 10 classes.

2.1 Neural network backend

- A python class `neuralnetwork` was constructed to facilitate training and prediction of neural networks of the desired architecture, activation function(sigmoid/ReLU), learning rate, batch size and the number of classes.
- The initalization of weights and biases were performed using Xavier's initialization in case of sigmoid activation and using He's initialization for ReLU activation.
- The learning rate was allowed to be either constant/adaptive with the desired base/starting value.
- In the training phase, backpropagation is performed on by stochastic gradient descent on the objective/loss function, which could either be mean square error or binary cross entropy error.
- A universal stopping criterion was kept to be the minimum change in loss(MSE/BCE) averaged over an epoch as 10^{-6} and a cap on number of epochs as 1000, whichever happens first.

2.2 Single hidden layer network with sigmoid activation

Neural networks with single hidden layer were trained and tested for different number of hidden neurons.

5 hidden layer units

- *Training accuracy* : 86.2167%
- *Test accuracy* : 82.84%
- *Training time* : 343.24s
- *Training epochs* : 1000
- *Test data confusion matrix* :

$$\text{confusion matrix} = \begin{pmatrix} 787 & 3 & 17 & 75 & 5 & 1 & 90 & 0 & 20 & 2 \\ 2 & 940 & 12 & 36 & 2 & 0 & 1 & 0 & 4 & 3 \\ 33 & 2 & 707 & 12 & 147 & 0 & 87 & 0 & 12 & 0 \\ 37 & 19 & 10 & 836 & 29 & 0 & 56 & 1 & 9 & 3 \\ 1 & 2 & 100 & 36 & 745 & 2 & 106 & 0 & 8 & 0 \\ 3 & 2 & 0 & 0 & 0 & 902 & 0 & 48 & 14 & 31 \\ 135 & 4 & 109 & 47 & 102 & 0 & 569 & 0 & 32 & 2 \\ 0 & 0 & 0 & 0 & 0 & 34 & 0 & 926 & 0 & 40 \\ 11 & 0 & 4 & 8 & 13 & 6 & 18 & 3 & 934 & 3 \\ 2 & 0 & 0 & 0 & 0 & 8 & 0 & 37 & 5 & 948 \end{pmatrix}$$

10 hidden layer units

- *Training accuracy* : 89.15%
- *Test accuracy* : 84.58%
- *Training time* : 378.39s
- *Training epochs* : 1000
- *Test data confusion matrix* :

$$\text{confusion matrix} = \begin{pmatrix} 819 & 1 & 13 & 47 & 5 & 0 & 97 & 0 & 16 & 2 \\ 4 & 950 & 8 & 25 & 6 & 0 & 4 & 0 & 3 & 0 \\ 39 & 2 & 754 & 9 & 121 & 1 & 68 & 0 & 6 & 0 \\ 38 & 15 & 19 & 844 & 40 & 1 & 38 & 1 & 4 & 0 \\ 2 & 2 & 107 & 38 & 761 & 0 & 83 & 0 & 7 & 0 \\ 1 & 1 & 0 & 0 & 0 & 923 & 1 & 38 & 5 & 31 \\ 151 & 1 & 108 & 36 & 108 & 0 & 579 & 0 & 17 & 0 \\ 0 & 0 & 0 & 0 & 0 & 28 & 0 & 939 & 0 & 33 \\ 3 & 1 & 6 & 7 & 5 & 3 & 23 & 5 & 947 & 0 \\ 1 & 0 & 0 & 0 & 0 & 16 & 0 & 40 & 1 & 942 \end{pmatrix}$$

15 hidden layer units

- *Training accuracy* : 90.9517%
- *Test accuracy* : 86.37%
- *Training time* : 460.23s
- *Training epochs* : 1000
- *Test data confusion matrix* :

$$\text{confusion matrix} = \begin{pmatrix} 828 & 3 & 11 & 36 & 3 & 1 & 105 & 1 & 12 & 0 \\ 7 & 955 & 6 & 25 & 5 & 0 & 1 & 0 & 1 & 0 \\ 21 & 4 & 778 & 8 & 104 & 1 & 71 & 1 & 12 & 0 \\ 31 & 12 & 13 & 853 & 48 & 0 & 34 & 3 & 6 & 0 \\ 1 & 1 & 97 & 33 & 797 & 3 & 63 & 0 & 5 & 0 \\ 1 & 1 & 0 & 1 & 0 & 939 & 0 & 35 & 4 & 19 \\ 138 & 3 & 99 & 35 & 76 & 0 & 627 & 0 & 22 & 0 \\ 0 & 0 & 0 & 0 & 0 & 24 & 0 & 944 & 1 & 31 \\ 2 & 1 & 6 & 5 & 6 & 4 & 9 & 6 & 961 & 0 \\ 0 & 0 & 0 & 0 & 0 & 11 & 1 & 33 & 0 & 955 \end{pmatrix}$$

20 hidden layer units

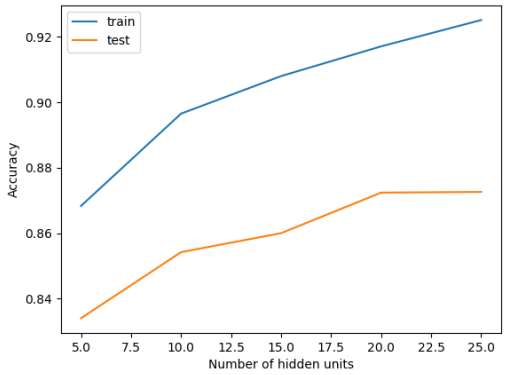
- *Training accuracy* : 91.805%
- *Test accuracy* : 87.15%
- *Training time* : 441.03s
- *Training epochs* : 1000
- *Test data confusion matrix* :

$$\text{confusion matrix} = \begin{pmatrix} 857 & 4 & 12 & 39 & 1 & 3 & 67 & 1 & 15 & 1 \\ 4 & 958 & 3 & 26 & 6 & 0 & 2 & 0 & 1 & 0 \\ 18 & 2 & 769 & 14 & 109 & 3 & 77 & 1 & 7 & 0 \\ 26 & 11 & 11 & 884 & 33 & 1 & 26 & 1 & 7 & 0 \\ 2 & 2 & 89 & 38 & 811 & 1 & 52 & 0 & 5 & 0 \\ 0 & 0 & 0 & 2 & 0 & 943 & 1 & 32 & 4 & 18 \\ 149 & 0 & 84 & 40 & 77 & 2 & 632 & 1 & 15 & 0 \\ 0 & 0 & 0 & 0 & 0 & 20 & 0 & 947 & 1 & 32 \\ 3 & 3 & 5 & 10 & 4 & 3 & 9 & 5 & 957 & 1 \\ 0 & 0 & 0 & 0 & 0 & 12 & 1 & 30 & 0 & 957 \end{pmatrix}$$

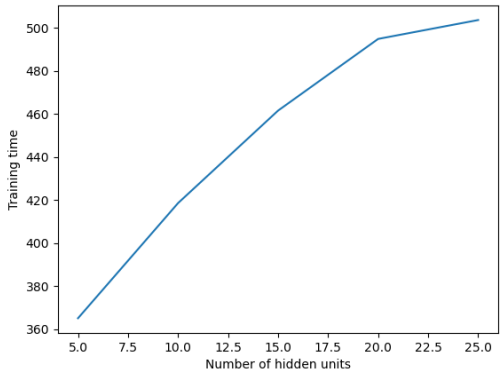
25 hidden layer units

- *Training accuracy* : 92.3917%
- *Test accuracy* : 87.33%
- *Training time* : 476.93
- *Training epochs* : 1000
- *Test data confusion matrix* :

$$\text{confusion matrix} = \begin{pmatrix} 834 & 1 & 15 & 35 & 5 & 2 & 94 & 0 & 13 & 1 \\ 3 & 962 & 2 & 24 & 4 & 0 & 4 & 0 & 1 & 0 \\ 23 & 2 & 780 & 7 & 111 & 0 & 69 & 1 & 7 & 0 \\ 27 & 9 & 10 & 879 & 36 & 0 & 31 & 1 & 6 & 1 \\ 1 & 0 & 92 & 35 & 797 & 1 & 69 & 0 & 5 & 0 \\ 0 & 0 & 0 & 1 & 0 & 951 & 0 & 29 & 3 & 16 \\ 140 & 2 & 81 & 32 & 79 & 0 & 653 & 0 & 13 & 0 \\ 0 & 0 & 0 & 0 & 0 & 24 & 0 & 950 & 0 & 26 \\ 1 & 1 & 3 & 6 & 3 & 3 & 8 & 5 & 970 & 0 \\ 0 & 0 & 0 & 0 & 0 & 11 & 0 & 31 & 1 & 957 \end{pmatrix}$$



(a) Accuracy variance



(b) Training time variance

Figure 22: Single hidden layer neural network

Observations

- The models trained with the above hyperparameters, all run till the 1000^{th} epoch. Hence, the time taken entire depends on the number of units in the hidden layer(Higher the number of units, more the computation in each iteration)
- The training accuracy increases with number of units as expected, causing the model to overfit. The training accuracy increases along with it but a slower pace and also appears to stabilize at higher number of units.

2.3 Adaptive Learning Rate

The neural network was trained on the same hyperparameters as above using the following adaptive learning rate setting :

$$\eta_t = \frac{\eta_0}{\sqrt{t}}, \eta_0 = 0.1$$

5 hidden layer units

- *Training accuracy* : 76.9067%
- *Test accuracy* : 76.02%
- *Training time* : 426.17s
- *Training epochs* : 1000

- *Test data confusion matrix* :

$$\text{confusion matrix} = \begin{pmatrix} 808 & 3 & 6 & 71 & 9 & 2 & 79 & 0 & 22 & 0 \\ 2 & 936 & 2 & 34 & 11 & 0 & 10 & 0 & 5 & 0 \\ 38 & 1 & 173 & 14 & 507 & 0 & 255 & 0 & 12 & 0 \\ 70 & 28 & 6 & 830 & 35 & 0 & 24 & 0 & 7 & 0 \\ 4 & 6 & 99 & 38 & 736 & 0 & 98 & 0 & 19 & 0 \\ 0 & 0 & 0 & 1 & 0 & 864 & 0 & 76 & 8 & 51 \\ 210 & 5 & 85 & 43 & 118 & 0 & 503 & 0 & 36 & 0 \\ 0 & 0 & 0 & 0 & 0 & 42 & 0 & 905 & 0 & 53 \\ 17 & 4 & 1 & 4 & 2 & 2 & 38 & 6 & 926 & 0 \\ 0 & 0 & 0 & 0 & 0 & 26 & 2 & 51 & 0 & 921 \end{pmatrix}$$

10 hidden layer units

- *Training accuracy* : 85.2167%
- *Test accuracy* : 83.65%
- *Training time* : 457.06s
- *Training epochs* : 1000
- *Test data confusion matrix* :

$$\text{confusion matrix} = \begin{pmatrix} 801 & 0 & 24 & 60 & 7 & 2 & 92 & 0 & 13 & 1 \\ 2 & 948 & 11 & 29 & 6 & 0 & 2 & 0 & 2 & 0 \\ 28 & 2 & 737 & 8 & 143 & 1 & 71 & 0 & 10 & 0 \\ 34 & 12 & 18 & 863 & 29 & 1 & 37 & 2 & 4 & 0 \\ 0 & 2 & 98 & 37 & 769 & 0 & 83 & 1 & 9 & 1 \\ 0 & 0 & 0 & 0 & 0 & 904 & 0 & 61 & 8 & 27 \\ 161 & 0 & 126 & 40 & 116 & 0 & 526 & 2 & 29 & 0 \\ 0 & 0 & 0 & 0 & 0 & 31 & 0 & 925 & 0 & 44 \\ 0 & 1 & 7 & 10 & 6 & 5 & 20 & 4 & 947 & 0 \\ 0 & 0 & 0 & 0 & 0 & 13 & 0 & 41 & 1 & 945 \end{pmatrix}$$

15 hidden layer units

- *Training accuracy* : 86.0817%
- *Test accuracy* : 84.23%
- *Training time* : 466.42s
- *Training epochs* : 1000

- *Test data confusion matrix* :

$$\text{confusion matrix} = \begin{pmatrix} 825 & 3 & 13 & 43 & 5 & 2 & 89 & 0 & 20 & 0 \\ 5 & 953 & 9 & 27 & 4 & 0 & 0 & 0 & 2 & 0 \\ 18 & 2 & 725 & 16 & 151 & 1 & 74 & 0 & 13 & 0 \\ 32 & 13 & 18 & 864 & 34 & 0 & 34 & 0 & 5 & 0 \\ 0 & 2 & 103 & 40 & 769 & 2 & 74 & 0 & 10 & 0 \\ 0 & 0 & 0 & 1 & 0 & 905 & 0 & 56 & 6 & 32 \\ 160 & 2 & 119 & 34 & 99 & 2 & 558 & 0 & 26 & 0 \\ 0 & 0 & 0 & 0 & 0 & 30 & 0 & 925 & 1 & 44 \\ 2 & 1 & 11 & 11 & 2 & 3 & 11 & 5 & 953 & 1 \\ 0 & 0 & 0 & 0 & 0 & 11 & 0 & 42 & 1 & 946 \end{pmatrix}$$

20 hidden layer units

- *Training accuracy* : 86.3217%
- *Test accuracy* : 84.59%
- *Training time* : 444.33s
- *Training epochs* : 1000
- *Test data confusion matrix* :

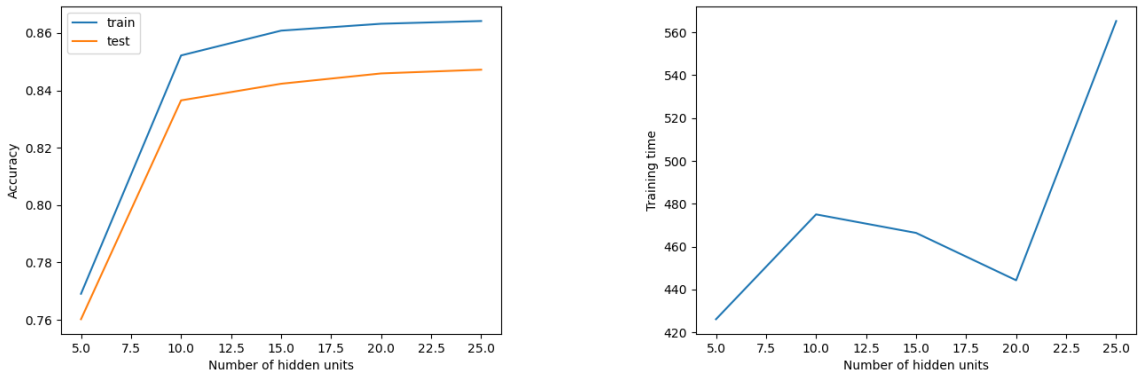
$$\text{confusion matrix} = \begin{pmatrix} 818 & 2 & 11 & 48 & 4 & 5 & 97 & 0 & 15 & 0 \\ 5 & 948 & 10 & 26 & 5 & 0 & 5 & 0 & 1 & 0 \\ 22 & 3 & 761 & 11 & 130 & 2 & 61 & 0 & 10 & 0 \\ 29 & 15 & 16 & 867 & 34 & 1 & 34 & 0 & 3 & 1 \\ 0 & 0 & 105 & 38 & 779 & 0 & 71 & 0 & 7 & 0 \\ 0 & 0 & 0 & 2 & 0 & 912 & 0 & 53 & 7 & 26 \\ 154 & 1 & 123 & 34 & 103 & 2 & 549 & 1 & 33 & 0 \\ 0 & 0 & 0 & 0 & 0 & 31 & 0 & 929 & 0 & 40 \\ 2 & 2 & 10 & 6 & 4 & 3 & 15 & 5 & 953 & 0 \\ 0 & 0 & 0 & 0 & 0 & 11 & 0 & 45 & 1 & 943 \end{pmatrix}$$

25 hidden layer units

- *Training accuracy* : 86.4167%
- *Test accuracy* : 84.72%
- *Training time* : 565.2328s
- *Training epochs* : 1000

- *Test data confusion matrix* :

$$\text{confusion matrix} = \begin{pmatrix} 837 & 4 & 15 & 41 & 6 & 3 & 80 & 0 & 14 & 0 \\ 3 & 951 & 7 & 29 & 6 & 0 & 2 & 0 & 2 & 0 \\ 19 & 2 & 755 & 13 & 130 & 2 & 66 & 1 & 12 & 0 \\ 27 & 14 & 18 & 869 & 36 & 1 & 32 & 0 & 3 & 0 \\ 0 & 1 & 112 & 42 & 766 & 0 & 74 & 0 & 5 & 0 \\ 1 & 0 & 0 & 2 & 0 & 909 & 0 & 52 & 8 & 28 \\ 151 & 2 & 117 & 37 & 103 & 1 & 556 & 0 & 33 & 0 \\ 0 & 0 & 0 & 0 & 0 & 33 & 0 & 928 & 0 & 39 \\ 2 & 2 & 9 & 6 & 3 & 4 & 15 & 4 & 955 & 0 \\ 0 & 0 & 0 & 0 & 0 & 11 & 0 & 42 & 1 & 946 \end{pmatrix}$$



(a) Accuracy variance

(b) Training time variance

Figure 23: Single hidden layer neural network with adptive learning rate

Observations

- The adaptive learning rate may result in a better fit, but also results in a slow learning process. This creates the requirement of a higher number of epochs than a model with constant learning rate.
- In the above models, the observed reduction in accuracies from the models in section 2.2 is an example of underfitting due to the lack of training epochs and requires more time to fit on the data properly.

2.4 ReLU activation

The sigmoid activation was compared with the ReLU activation in the same hidden layer architecture : [100,100] with an adaptive learning rate.

Sigmoid Activation

- *Training accuracy* : 87.1117%
- *Test accuracy* : 85.29%
- *Training time* : 1648.58s
- *Training epochs* : 1000
- *Test data confusion matrix* :

$$\text{confusion matrix} = \begin{pmatrix} 826 & 3 & 14 & 48 & 5 & 4 & 86 & 0 & 14 & 0 \\ 7 & 954 & 4 & 27 & 3 & 0 & 4 & 0 & 1 & 0 \\ 17 & 2 & 776 & 13 & 121 & 1 & 62 & 0 & 8 & 0 \\ 31 & 13 & 16 & 866 & 39 & 0 & 30 & 0 & 4 & 1 \\ 0 & 3 & 108 & 40 & 773 & 1 & 68 & 0 & 7 & 0 \\ 1 & 0 & 0 & 2 & 0 & 919 & 0 & 46 & 6 & 26 \\ 140 & 1 & 111 & 45 & 86 & 0 & 586 & 0 & 31 & 0 \\ 0 & 0 & 0 & 0 & 0 & 31 & 0 & 928 & 1 & 40 \\ 0 & 1 & 7 & 8 & 3 & 3 & 14 & 4 & 960 & 0 \\ 0 & 0 & 0 & 0 & 0 & 14 & 0 & 44 & 1 & 941 \end{pmatrix}$$

ReLU Activation

- *Training accuracy* : 94.0917%
- *Test accuracy* : 88.46%
- *Training time* : 1408.95s
- *Training epochs* : 1000
- *Test data confusion matrix* :

$$\text{confusion matrix} = \begin{pmatrix} 867 & 3 & 11 & 26 & 6 & 0 & 76 & 0 & 11 & 0 \\ 5 & 966 & 3 & 20 & 3 & 0 & 2 & 0 & 1 & 0 \\ 18 & 2 & 807 & 13 & 94 & 1 & 64 & 0 & 1 & 0 \\ 29 & 7 & 14 & 886 & 36 & 0 & 23 & 0 & 4 & 1 \\ 1 & 1 & 78 & 37 & 827 & 0 & 52 & 0 & 4 & 0 \\ 1 & 0 & 0 & 1 & 0 & 959 & 1 & 26 & 2 & 10 \\ 142 & 1 & 76 & 31 & 74 & 0 & 663 & 0 & 13 & 0 \\ 0 & 0 & 0 & 0 & 0 & 19 & 0 & 956 & 0 & 25 \\ 7 & 2 & 4 & 5 & 7 & 2 & 10 & 5 & 958 & 0 \\ 0 & 0 & 1 & 0 & 0 & 9 & 0 & 32 & 1 & 957 \end{pmatrix}$$

Observation

- The above comparison clearly indicates the superiority of the ReLU activation function of the sigmoid activation function, with ReLU performing better in test accuracy by 3.17%.
- The training time in case of ReLU activation is significantly lesser than the sigmoid activation at the same number of epochs, indicating the lesser computation required per iteration.

2.5 Multi-layer networks

Keeping the number of units in each hidden layer set to 50, observations were made on the effect of number of hidden layers on the test performance and training time.

Sigmoid Activation

2 Hidden layers

- *Training accuracy* : 86.6733%
- *Test accuracy* : 84.92%
- *Training time* : 922.92s
- *Training epochs* : 1000
- *Test data confusion matrix* :

$$\text{confusion matrix} = \begin{pmatrix} 823 & 4 & 15 & 45 & 4 & 4 & 90 & 0 & 14 & 1 \\ 6 & 946 & 8 & 28 & 7 & 0 & 3 & 0 & 2 & 0 \\ 21 & 1 & 771 & 12 & 118 & 1 & 69 & 0 & 7 & 0 \\ 31 & 10 & 14 & 869 & 39 & 0 & 32 & 0 & 5 & 0 \\ 0 & 0 & 110 & 37 & 765 & 1 & 81 & 0 & 6 & 0 \\ 0 & 0 & 0 & 1 & 0 & 912 & 0 & 52 & 7 & 28 \\ 146 & 2 & 118 & 33 & 97 & 1 & 574 & 0 & 29 & 0 \\ 0 & 0 & 0 & 0 & 0 & 35 & 0 & 928 & 0 & 37 \\ 0 & 1 & 11 & 7 & 2 & 5 & 12 & 5 & 957 & 0 \\ 0 & 0 & 0 & 0 & 0 & 11 & 0 & 41 & 1 & 947 \end{pmatrix}$$

3 Hidden layers

- *Training accuracy* : 86.4533%
- *Test accuracy* : 84.74%
- *Training time* : 1219.32s
- *Training epochs* : 1000
- *Test data confusion matrix* :

$$\text{confusion matrix} = \begin{pmatrix} 808 & 4 & 20 & 44 & 4 & 1 & 103 & 0 & 16 & 0 \\ 3 & 940 & 11 & 37 & 5 & 0 & 3 & 0 & 1 & 0 \\ 23 & 2 & 772 & 10 & 123 & 1 & 60 & 0 & 9 & 0 \\ 30 & 9 & 17 & 859 & 41 & 0 & 40 & 0 & 4 & 0 \\ 2 & 2 & 98 & 31 & 778 & 1 & 81 & 0 & 7 & 0 \\ 0 & 0 & 0 & 1 & 0 & 921 & 0 & 45 & 5 & 28 \\ 160 & 2 & 120 & 35 & 97 & 1 & 558 & 0 & 27 & 0 \\ 0 & 0 & 0 & 0 & 0 & 31 & 0 & 933 & 0 & 36 \\ 0 & 2 & 6 & 9 & 2 & 3 & 15 & 6 & 957 & 0 \\ 0 & 0 & 0 & 0 & 0 & 11 & 0 & 40 & 1 & 948 \end{pmatrix}$$

4 Hidden layers

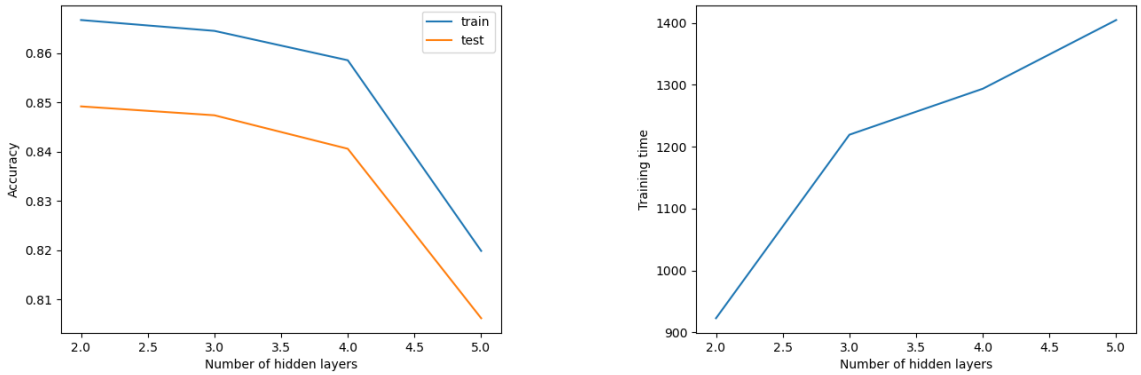
- *Training accuracy* : 85.8567%
- *Test accuracy* : 84.06%
- *Training time* : 1293.59s
- *Training epochs* : 1000
- *Test data confusion matrix* :

$$\text{confusion matrix} = \begin{pmatrix} 806 & 4 & 10 & 44 & 9 & 4 & 110 & 0 & 12 & 1 \\ 6 & 936 & 16 & 34 & 5 & 0 & 2 & 0 & 1 & 0 \\ 18 & 1 & 754 & 9 & 128 & 3 & 80 & 0 & 7 & 0 \\ 29 & 16 & 8 & 851 & 37 & 2 & 53 & 0 & 4 & 0 \\ 1 & 0 & 104 & 36 & 770 & 0 & 82 & 0 & 7 & 0 \\ 0 & 0 & 0 & 1 & 0 & 919 & 0 & 46 & 6 & 28 \\ 163 & 1 & 128 & 33 & 94 & 3 & 556 & 0 & 22 & 0 \\ 0 & 0 & 0 & 0 & 0 & 33 & 0 & 931 & 0 & 36 \\ 0 & 1 & 9 & 7 & 9 & 13 & 14 & 6 & 939 & 2 \\ 0 & 0 & 1 & 0 & 0 & 15 & 0 & 39 & 1 & 944 \end{pmatrix}$$

5 Hidden layers

- *Training accuracy* : 81.9867%
- *Test accuracy* : 80.62%
- *Training time* : 1404.54s
- *Training epochs* : 1000
- *Test data confusion matrix* :

$$\text{confusion matrix} = \begin{pmatrix} 832 & 5 & 30 & 53 & 9 & 7 & 53 & 0 & 11 & 0 \\ 9 & 937 & 11 & 33 & 5 & 0 & 4 & 0 & 1 & 0 \\ 17 & 1 & 782 & 9 & 146 & 0 & 36 & 0 & 9 & 0 \\ 39 & 13 & 14 & 846 & 46 & 1 & 35 & 0 & 6 & 0 \\ 4 & 0 & 118 & 38 & 803 & 1 & 28 & 0 & 8 & 0 \\ 0 & 0 & 0 & 1 & 0 & 905 & 0 & 50 & 6 & 38 \\ 205 & 0 & 256 & 59 & 298 & 6 & 151 & 0 & 25 & 0 \\ 0 & 0 & 0 & 0 & 0 & 34 & 0 & 933 & 0 & 33 \\ 0 & 1 & 8 & 9 & 12 & 11 & 15 & 3 & 936 & 5 \\ 0 & 0 & 0 & 2 & 0 & 15 & 0 & 45 & 1 & 937 \end{pmatrix}$$



(a) Accuracy variance **(b)** Training time variance

Figure 24: Multi Layer neural network - Sigmoid activation

ReLU Activation

2 Hidden layers

- *Training accuracy* : 92.9883%
- *Test accuracy* : 88.16%
- *Training time* : 815.54s
- *Training epochs* : 1000
- *Test data confusion matrix* :

$$\text{confusion matrix} = \begin{pmatrix} 861 & 2 & 16 & 24 & 6 & 1 & 78 & 0 & 11 & 1 \\ 6 & 966 & 2 & 19 & 3 & 0 & 3 & 0 & 1 & 0 \\ 27 & 0 & 805 & 11 & 93 & 0 & 61 & 0 & 3 & 0 \\ 32 & 8 & 15 & 882 & 40 & 0 & 20 & 0 & 3 & 0 \\ 1 & 2 & 91 & 31 & 812 & 0 & 56 & 0 & 7 & 0 \\ 0 & 0 & 1 & 1 & 0 & 955 & 0 & 29 & 1 & 13 \\ 138 & 1 & 77 & 28 & 76 & 0 & 665 & 0 & 15 & 0 \\ 0 & 0 & 0 & 0 & 0 & 22 & 0 & 951 & 0 & 27 \\ 2 & 1 & 6 & 6 & 7 & 2 & 8 & 4 & 964 & 0 \\ 0 & 1 & 0 & 0 & 0 & 8 & 0 & 36 & 0 & 955 \end{pmatrix}$$

3 Hidden layers

- *Training accuracy* : 94.8983%
- *Test accuracy* : 87.85%
- *Training time* : 988.30s
- *Training epochs* : 1000

- *Test data confusion matrix* :

$$\text{confusion matrix} = \begin{pmatrix} 842 & 0 & 11 & 22 & 8 & 1 & 103 & 0 & 13 & 0 \\ 4 & 960 & 5 & 20 & 6 & 0 & 4 & 0 & 0 & 1 \\ 20 & 1 & 798 & 17 & 92 & 0 & 64 & 2 & 5 & 1 \\ 30 & 8 & 12 & 884 & 35 & 0 & 27 & 1 & 3 & 0 \\ 2 & 0 & 85 & 29 & 810 & 1 & 68 & 1 & 4 & 0 \\ 0 & 0 & 0 & 1 & 0 & 954 & 0 & 30 & 2 & 13 \\ 130 & 2 & 73 & 35 & 73 & 0 & 674 & 0 & 13 & 0 \\ 0 & 0 & 0 & 0 & 0 & 31 & 0 & 942 & 0 & 27 \\ 2 & 1 & 7 & 4 & 4 & 2 & 8 & 6 & 966 & 0 \\ 0 & 0 & 0 & 0 & 0 & 9 & 1 & 35 & 0 & 955 \end{pmatrix}$$

4 Hidden layers

- *Training accuracy* : 95.865%
- *Test accuracy* : 87.56%
- *Training time* : 1026.08s
- *Training epochs* : 1000
- *Test data confusion matrix* :

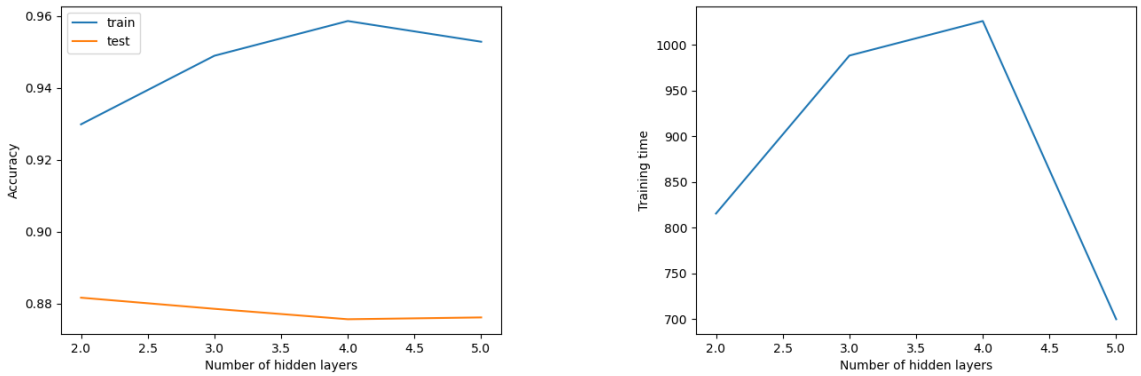
$$\text{confusion matrix} = \begin{pmatrix} 830 & 2 & 17 & 29 & 8 & 1 & 98 & 0 & 15 & 0 \\ 4 & 963 & 6 & 22 & 3 & 0 & 2 & 0 & 0 & 0 \\ 15 & 3 & 789 & 12 & 92 & 0 & 85 & 0 & 3 & 1 \\ 25 & 9 & 13 & 892 & 31 & 1 & 24 & 0 & 5 & 0 \\ 2 & 0 & 95 & 35 & 794 & 1 & 67 & 1 & 5 & 0 \\ 1 & 0 & 0 & 0 & 0 & 952 & 0 & 24 & 2 & 21 \\ 140 & 1 & 84 & 28 & 61 & 2 & 674 & 0 & 10 & 0 \\ 1 & 0 & 0 & 0 & 0 & 26 & 0 & 944 & 0 & 29 \\ 2 & 0 & 6 & 7 & 7 & 4 & 14 & 6 & 954 & 0 \\ 0 & 0 & 0 & 0 & 0 & 8 & 0 & 27 & 1 & 964 \end{pmatrix}$$

5 Hidden layers

- *Training accuracy* : 95.2883%
- *Test accuracy* : 87.61%
- *Training time* : 699.91s
- *Training epochs* : 604

- *Test data confusion matrix* :

$$\text{confusion matrix} = \begin{pmatrix} 835 & 2 & 22 & 32 & 9 & 1 & 86 & 0 & 13 & 0 \\ 5 & 963 & 6 & 20 & 5 & 0 & 0 & 0 & 1 & 0 \\ 18 & 0 & 787 & 21 & 92 & 0 & 79 & 0 & 3 & 0 \\ 27 & 12 & 16 & 887 & 28 & 0 & 23 & 0 & 7 & 0 \\ 1 & 1 & 98 & 40 & 801 & 0 & 56 & 0 & 3 & 0 \\ 0 & 0 & 1 & 1 & 0 & 957 & 0 & 20 & 2 & 19 \\ 130 & 0 & 82 & 33 & 68 & 0 & 675 & 0 & 12 & 0 \\ 0 & 0 & 0 & 0 & 0 & 25 & 0 & 929 & 0 & 46 \\ 5 & 0 & 4 & 7 & 6 & 2 & 8 & 3 & 965 & 0 \\ 0 & 1 & 0 & 1 & 0 & 7 & 0 & 27 & 2 & 962 \end{pmatrix}$$



(a) Accuracy variance **(b)** Training time variance

Figure 25: Multi Layer neural network - ReLU activation

Observation

- The decay in training and test accuracy in case of multi-layer models with sigmoid activation indicates underfitting due to the lack of training epochs. This allows us to deduce that an increase in number of hidden layers introduces a higher demand of training epochs to properly fit on the training data.

- Unlike the models with sigmoid activation function, the models the ReLU activation show an increase in training accuracy till 4 layers, while the test accuracy decreases. This clearly indicates that with the same number of epochs, a model with ReLU activation is able to train to overfitting, while the sigmoid activation models underfit. This allows us to deduce that a model with ReLU activation function requires significantly lesser number of epochs than a model with sigmoid activation to acheive the same fit. The 5-layer model with ReLU activation shows a drop in training accuracy, with a slight increase in test accuracy, indicating a relative underfit than the 4-layer network. This can also be due to an unexpected early stop caused by vanishing gradients considering the low number of training epochs.

Experimenting with layers : Hunt for the best architecture

ReLU activated [25,50] architecture

- *Training accuracy* : 90.1253%
- *Test accuracy* : 87.91%

ReLU activated [50,25] architecture

- *Training accuracy* : 89.9883%
- *Test accuracy* : 86.75%

ReLU activated [50,75] architecture

- *Training accuracy* : 93.3742%
- *Test accuracy* : 88.51%

ReLU activated [25,50,25] architecture

- *Training accuracy* : 91.5254%
- *Test accuracy* : 87.2342%

2.6 Binary cross entropy loss

Considering the backpropagation was programmed to implement a stochastic gradient descent on a desired cost function J, we can formulate the following :

Let the output of neurons be $\mathcal{O}_j$ and ψ be the activation function.

$$\delta_j = \frac{\partial J}{\partial \mathcal{O}_j}\frac{\partial \mathcal{O}_j}{\partial net_j} = \begin{cases} \frac{\partial J(\mathcal{O}_j,y)}{\partial \mathcal{O}_j}\frac{\partial \psi(net_j)}{\partial net_j} & \text{if } j \text{ is and output neuron} \\ (\sum_{l\in L} w_{jl}\delta_l)\frac{\partial \psi(net_j)}{\partial net_j} & \text{if } j \text{ is and inner neuron} \end{cases}$$

The only change to implement gradient descent on binary cross entropy loss will be in the δ_j calculation at the output layer, where $\partial J(\mathcal{O}_j, y)/\partial \mathcal{O}_j$ will now be evaluated as follows.

$$\begin{aligned} J(\mathcal{O}, y) &= -\sum_{j=1}^{C} y_j log(\mathcal{O}_j) + (1-y_j)log(1-\mathcal{O}_j) \\ \frac{\partial J(\mathcal{O}, y)}{\partial \mathcal{O}_j} &= \frac{\mathcal{O}_j - y_j}{\mathcal{O}_j(1-\mathcal{O}_j)} \\ \frac{\partial J(\mathcal{O}_j, y)}{\partial \mathcal{O}_j}\frac{\partial \psi(net_j)}{\partial net_j} &= \mathcal{O}_j - y_j \end{aligned}$$

Implementing the above with 3 hidden layers with 50 units each(as it performed the best in section 2.5) with ReLU activation and adaptive learning rate, the following observations were made.

- *Training accuracy* : 96.8233%
- *Test accuracy* : 87.4%
- *Training time* : 824.8647s
- *Training epochs* : 1000
- *Test data confusion matrix* :

$$\text{confusion matrix} = \begin{pmatrix} 847 & 4 & 19 & 37 & 4 & 1 & 76 & 1 & 11 & 0 \\ 4 & 969 & 3 & 11 & 7 & 0 & 3 & 0 & 3 & 0 \\ 22 & 1 & 808 & 15 & 83 & 1 & 67 & 0 & 3 & 0 \\ 35 & 13 & 15 & 886 & 33 & 1 & 12 & 0 & 5 & 0 \\ 0 & 0 & 113 & 41 & 788 & 0 & 53 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 960 & 0 & 25 & 1 & 14 \\ 149 & 2 & 85 & 36 & 77 & 0 & 635 & 0 & 14 & 2 \\ 0 & 0 & 0 & 0 & 0 & 28 & 0 & 929 & 0 & 43 \\ 12 & 0 & 4 & 8 & 6 & 5 & 7 & 2 & 956 & 0 \\ 0 & 1 & 0 & 0 & 1 & 10 & 1 & 25 & 0 & 962 \end{pmatrix}$$

2.7 MLPClassifier by scikit-learn

The multi-layer perceptron classifier *MLPClassifier* by scikit-learn was used to train(SGD) a neural network provide with 2 hidden layers (100,100) with ReLU activation. MLPClassifier by default uses binary cross entropy loss, hence can be compare to the model in section 2.6. The results of the training were as follows :

- *Training accuracy* : 99.395%
- *Test accuracy* : 87.59%
- *Training time* : 991.32s
- *Test data confusion matrix* :

$$\text{confusion matrix} = \begin{pmatrix} 839 & 3 & 16 & 33 & 5 & 3 & 94 & 0 & 7 & 0 \\ 9 & 963 & 2 & 18 & 2 & 0 & 6 & 0 & 0 & 0 \\ 29 & 3 & 813 & 14 & 79 & 0 & 59 & 0 & 3 & 0 \\ 37 & 11 & 19 & 870 & 26 & 0 & 31 & 0 & 6 & 0 \\ 1 & 1 & 87 & 44 & 801 & 0 & 55 & 0 & 11 & 0 \\ 0 & 0 & 1 & 2 & 0 & 947 & 0 & 26 & 3 & 21 \\ 124 & 1 & 79 & 33 & 81 & 0 & 670 & 0 & 11 & 1 \\ 0 & 0 & 0 & 0 & 0 & 17 & 0 & 954 & 0 & 29 \\ 11 & 0 & 5 & 6 & 6 & 3 & 9 & 5 & 955 & 0 \\ 0 & 1 & 0 & 0 & 0 & 8 & 1 & 43 & 0 & 947 \end{pmatrix}$$

Observations

- The MLPClassifier model takes longer to train and performs slightly better than the model built from scratch, with both implementing a stochastic gradient descent on binary cross entropy loss with the same neural network architecture and activation.
- The models appear to have a very similar approach/behavior on the training and test data which can be seen from the similarity of the confusion matrices(misclassifications are high/low at the exact same rows and columns).