COL774 Assignment3

Aniket Gupta
Entry No. - 2019CS10327

30 October 2021

1 Decision Trees (and Random Forests)

1.1 Decision Tree Construction

In this part, I constructed a decision tree using the given data to predict clients who subscribed a term deposit. At each node, I selected an attribute which led to maximum decrease in entropy on the training data. For a numerical attribute, a two way split is performed by calculating the median value for the data at that node. For categorical attribute, we have used two approaches - either doing a multi-way split or doing two way split after one-hot encoding the categorical attributes. The variation of accuracy as we expand the decision tree for these two approaches are shown below.

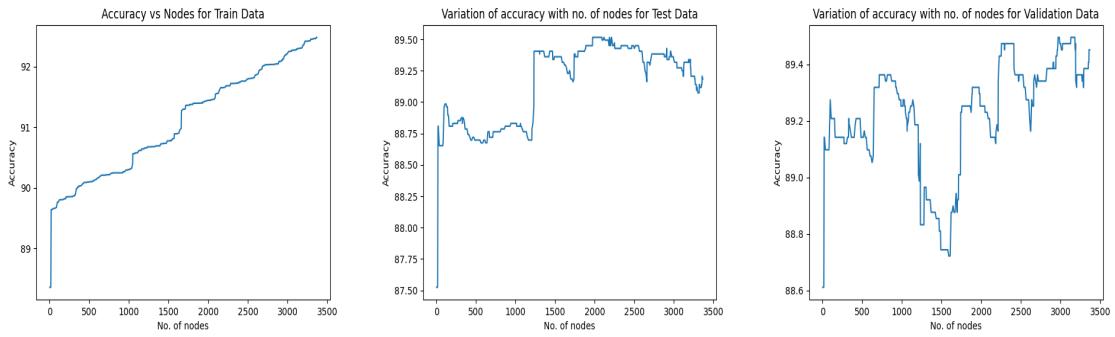

Figure 1: Variation of accuracy with no. of nodes in the Decision tree on multi-way split for categorical attributes

On doing multi-way split for categorical attributes:

- The accuracy increases monotonically on the training data as the number of nodes increases as the split on a node always increases the accuracy on the training data.
- The accuracy does not increase monotonically on test as well as validation data as the decision tree is trained on another set of training data which may have different distribution of samples from that in test and validation data.

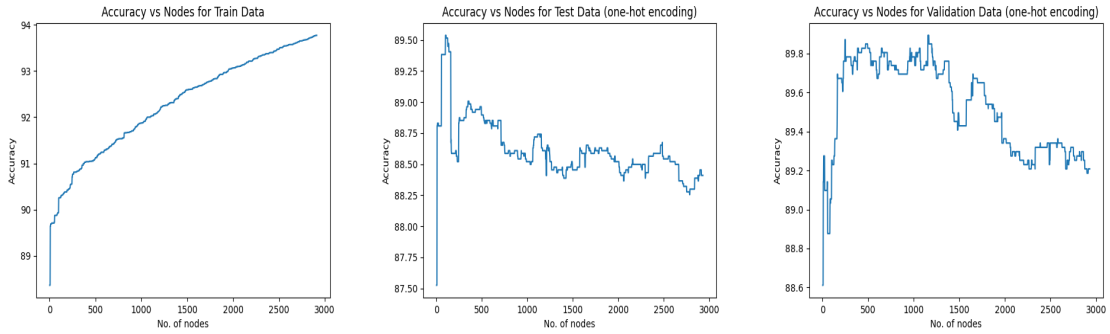

Figure 2: Variation of accuracy with no. of nodes in the Decision tree on two-way split after one-hot encoding categorical attributes

On doing two-way splits for categorical attributes after one-hot encoding them:

- The accuracy increases monotonically on training data due to the same reason as explained in multi-way split.
- The accuracy first shows an increasing trend and then decreases a bit on the test as well as validation data. This may be due to the overfitting of decision tree model on the training data.

On comparing multi-way split with two-way split (after one-hot encoding), we can see from the plots abive that both the methods gives almost same accuracy on all training, test and validattion data.

1.2 Decision Tree Post Pruning

In this part, I post-pruned the decision tree to reduce its overfitting on the training data. I iteratively picked a node pruning which leads to maximum increase in accuracy on the validation data.

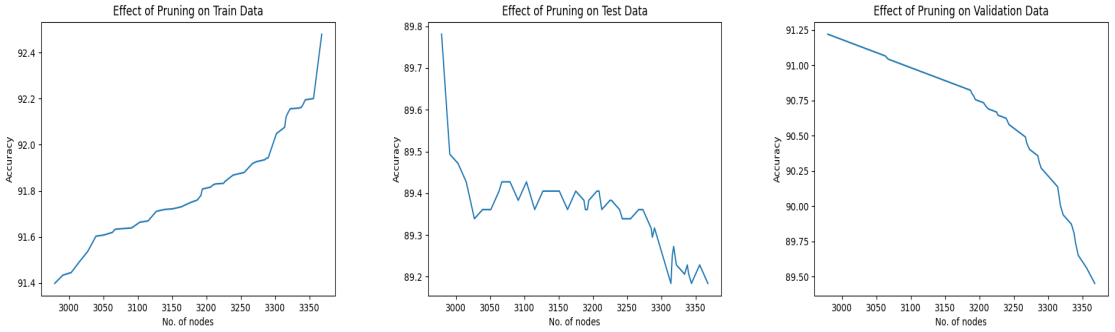

Figure 3: Effect of post-pruning on accuracy (for the decision tree based on multi-way split on categorical attributes)

Accuracies obtained after post-pruning the first decision tree:

- Training Data - 91.39 %

- Test Data - 89.78 %
- Validation Data - 91.22 %

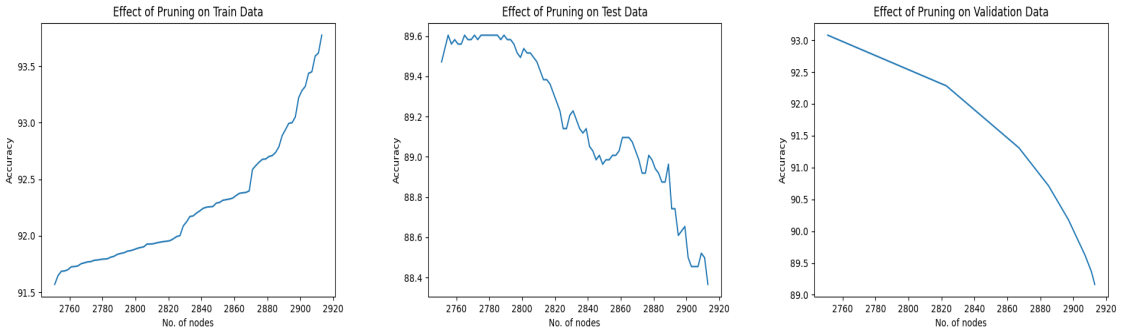

Figure 4: Effect of post-pruning on accuracy (for the decision tree based on two-way split on categorical attributes after one-hot encoding)

Accuracies obtained after post-pruning second decision tree:

- Training Data - 91.56 %
- Test Data - 89.47 %
- Validation Data - 93.07 %

Observations:

- For both the decision trees, we find the similar observations on post pruning.
- The accuracy on the validation set increases as the no. of nodes decreases in the decision tree during post-pruning. This is because post-pruning method used here iteratively prunes as long as there are chances of increase in accuracy on the validation data.
- The accuracy on the training data decreases as the no. of nodes decreases in the decision tree during post-pruning. This is because during the decision tree building, the accuracy increased on every split of the node. Now, pruning reverses that increase on the training data.
- The accuracy shows an increasing trend as the number of nodes decreases due to pruning. This is because post-pruning reduce the overfit of the decision tree models on the training data and thus the model gives more accurate result on new unseen data.

1.3 Random Forests

In this part, I used the Random Forest approach. In Random Forest, we grow multiple decision trees in parallel on bootstrapped constructed from original training data. I implemented this using scikit-learn library. I experimented with different values of parameters by performing a grid search over the following parameters.

- n estimators (50 to 450 in range of 100)

- max features (0.1 to 1.0 in range of 0.2)
- min samples split (2 to 10 in range of 2)

To find the best parameter, I found the parameters which gave the best out-of-bag accuracy. Also, since scikit assume integer values as ordinal, we cannot encode the categorical attributes by integers. I used one-hot encoding to encode the dataset. **Results Obtained:**

- Optimal set of parameters
  - n_estimator = 350
  - max_features = 0.7
  - min_sample_split = 10
- Out-of-bag accuracy with optimal parameters = 90.81 %
- Accuracy on training data with optimal parameters = 98.52 %
- Accuracy on test data with optimal parameters = 90.00 %
- Accuracy on validation data with optimal parameters = 90.49 %

Comparison with results obtained in part b:

Model Used	Accuracy on Training-Data	Accuracy on Test-Data	Accuracy on Validation Data
Decision Tree Post Pruning	91.56 %	89.47 %	93.07 %
Random Forest	98.52 %	90.00 %	90.49 %

Observations:

- In case of Decision Tree with Post Pruning, accuracy is maximum on validation data as the pruning step was trying to increase the accuracy on validation data.
- In case of Random Forests, the accuracy is maximum on training data as the model trained tries to increase its accuracy on training data during learning.
- The accuracy on training data is greater in case of Random Forest since there is no pruning to avoid overfit in this case.
- The accuracy on test data is marginally greater for Random Forest whereas the accuracy on Validation data is greater in case of Decision Tree post pruning.

1.4 Random Forests - Parameter Sensitivity Analysis

In the previous part, we obtained the optimal set of parameters for Random Forests. In this part, I varied one of the parameters while fixing others to their optimum.

Variation in accuracy on varying the n_estimator:

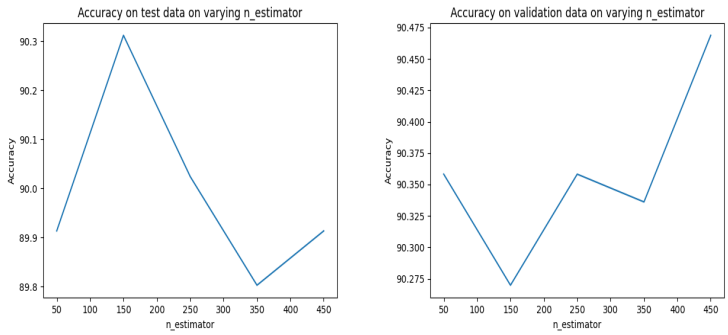

Figure 5: Accuracies on changing n_estimators

Variation in accuracy on varying the max_features:

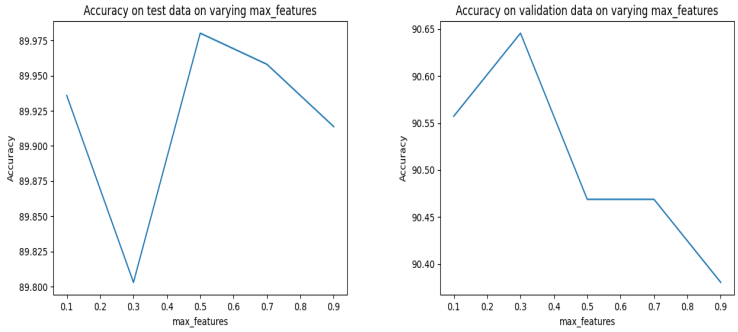

Figure 6: Accuracies on changing max_features

Variation in accuracy on varying the min_samples_split:

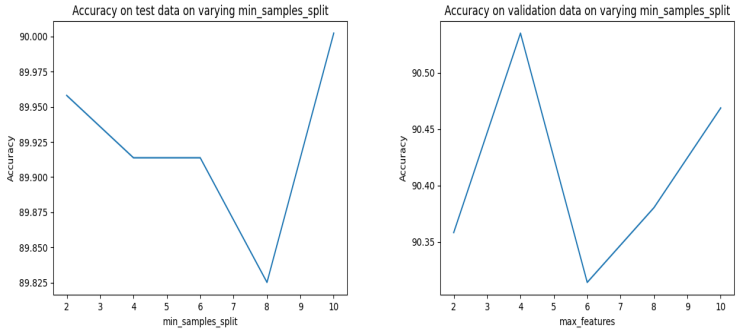

Figure 7: Accuracies on changing min_samples_split

Observations:

- On varying n_estimators, the maximum accuracy for test data is obtained at 150 whereas for the validation data, it is maximum at 450.
- On varying max_features, the maximum accuracy for test data is obtained at 0.5 whereas for the validation data, it is maximum at 0.3.
- On varying min_samples_split, the maximum accuracy for test data is obtained at 10 whereas for the validation data, it is maximum at 4.
- n_estimators appear to be the most sensitive based on test and validation data, as the range of variation of accuracy is maximum in this case.

2 Neural Networks

2.1 One-hot Encoding

In this part, I one-hot encoded the training as well as test dataset. One-hot encoding converted the categorical features to binary. The initial dataset has 10 categorical attributes. On one-hot encoding, the new data has 85 binary attributes. The transformed data has 10 output classes. For one-hot encoding, I iteratively traversed and one-hot encoded each sample.

2.2 Generic Neural Network Model

In this part, I implemented a generic neural network architecture to learn a model for multi-class classification using one-hot encoding as described above. The model implemented is generic to create an architecture based on Mini-batch Size, Number of features, Hidden layer architecture and Number of target classes. I used mini-batch Stochastic Gradient Descent method to train the network. The Mean Squared error over each mini-batch is used as the loss function.

$$J^b(\theta) = \frac{1}{2M} \sum_{i=(b-1)M}^{bM} \sum_{l=1}^{r} (y_l^{(i)} - o_l^{(i)})^2$$

I used the He initialisation for the weight parameters of each layer. In this, we multiply a random number with

$$\sqrt{\frac{2}{size_{l-1}}}$$

where $size_{l-1}$ is no. of units in the previous layer.

2.3 Varying no. of hidden layer units

In this part, I experimented with a neural network having a single hidden layer by varying the number of hidden layer units from the set {5, 10, 15, 20, 25}. The learning rate is set at 0.1 and mini-batch size of 100 examples.

Stopping Criteria

The average error is calculated over all the mini-batches in a single epoch. When the difference in average error over consecutive epochs become less than ϵ = 1e-8, the stopping criteria is achieved. In case the number of epochs increases more than 1000, we stop the learning in that case as well.

Results Obtained:

- No. of Hidden Layer Units: 5
  - Accuracy on Training set: 58.41 %
  - Accuracy on Test set: 55.70 %
  - Time taken to train the model: 42.50 sec
  - Confusion Matrix Obtained:

$$\begin{bmatrix} 362980 & 138229 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 228408 & 194090 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 19890 & 27732 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 6256 & 14865 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1575 & 2310 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1611 & 385 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 299 & 1125 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 27 & 203 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 10 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

- No. of Hidden Layer Units: 10
  - Accuracy on Training set: 66.72 %
  - Accuracy on Test set: 63.65 %
  - Time taken to train the model: 46.43 sec
  - Confusion Matrix Obtained:

$$\begin{bmatrix} 399004 & 102205 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 184961 & 237537 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 9807 & 37815 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3931 & 17190 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1247 & 2638 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1794 & 202 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 126 & 1298 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 9 & 221 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

- No. of Hidden Layer Units: 15
  - Accuracy on Training set: 63.42 %
  - Accuracy on Test set: 60.50 %
  - Time taken to train the model: 56.78 sec
  - Confusion Matrix Obtained:

$$\begin{bmatrix} 391501 & 109708 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 208932 & 213566 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 14909 & 32713 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3530 & 17591 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2570 & 1315 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1862 & 134 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 194 & 1230 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 228 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

- No. of Hidden Layer Units: 20
  - Accuracy on Training set: 67.40 %
  - Accuracy on Test set: 64.4 %
  - Time taken to train the model: 52.24 sec
  - Confusion Matrix Obtained:

$$\begin{bmatrix} 427768 & 73441 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 206266 & 216232 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 11055 & 36567 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4045 & 17076 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1953 & 1932 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1921 & 75 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 76 & 1348 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 10 & 220 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 11 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

- No. of Hidden Layer Units: 25
  - Accuracy on Training set: 63.72 %
  - Accuracy on Test set: 60.71 %
  - Time taken to train the model: 62.82 sec
  - Confusion Matrix Obtained:

$$\begin{bmatrix} 385790 & 115419 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 201186 & 221312 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 14846 & 32776 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2730 & 18391 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2594 & 1291 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1847 & 149 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 197 & 1227 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 225 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 8 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

As we vary the number of hidden layer units, the accuracy on test and training dataset and time taken to train the model varies as shown below:

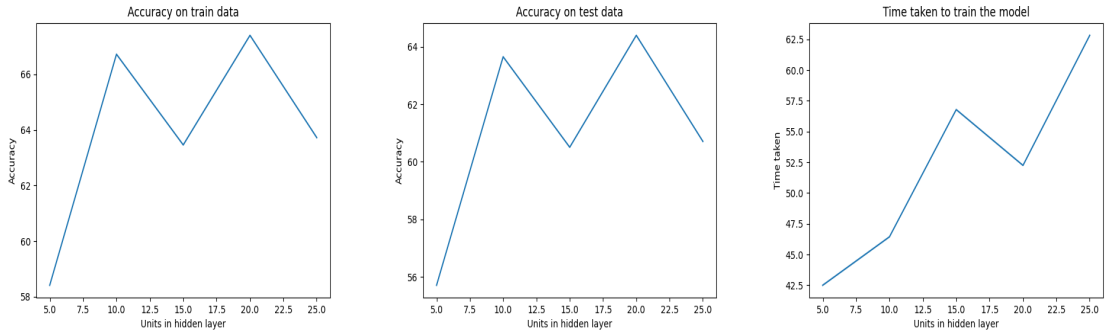

Figure 8: Change in different metrics as the number of units in hidden layer increases

Observations:

- The model is mostly predicting only two classes out of total 10 classes. This is due to the disproportionate distribution of the samples in different classes in the training dataset.
- The time taken to train the model shows an increasing trend as we increase the number of units in the hidden layer. This is expected as the computation time increases as the complexity of the architecture increases.
- The accuracy on test as well as training data is minimum for 5 hidden layer units. It shows an almost increasing trend with the increase in number oh hidden layer units.

2.4 Adaptive learning

In this part, I used adaptive learning rate inversely proportional to the number of epochs i.e. $\eta = \frac{\eta_0}{\sqrt{e}}$ where $\eta_0 = 0.1$ is the seed value and e is the current epoch number.

Stopping Criteria:
The average error is calculated over all the mini-batches in a single epoch. When the difference in average error over consecutive epochs become less than ϵ = 1e-8 or the number of epochs become greater than 3000, the stopping criteria is achieved. I needed to increase the maximum no. of epochs with respect to the non-adaptive case because in adaptive learning the the models learns lesser in each iteration as compared to the non-adaptive case as the learning rate decreases after each epoch.

Results Obtained:

- No. of Hidden Layer Units: 5
  - Accuracy on Training set: 49.85 %
  - Accuracy on Test set: 49.94 %
  - Time taken to train the model: 129.69 sec
  - Confusion Matrix Obtained:

$$\begin{bmatrix} 493806 & 7403 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 416859 & 5639 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 47087 & 535 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 20887 & 234 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3786 & 99 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1927 & 69 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1415 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 229 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

- No. of Hidden Layer Units: 10
  - Accuracy on Training set: 49.85 %
  - Accuracy on Test set: 49.79 %
  - Time taken to train the model: 140.86 sec
  - Confusion Matrix Obtained:
$$\begin{bmatrix} 489248 & 11961 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 413789 & 8709 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 46792 & 830 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 20806 & 315 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3758 & 127 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1885 & 111 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1411 & 13 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 229 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

- No. of Hidden Layer Units: 15
  - Accuracy on Training set: 50.0 %
  - Accuracy on Test set: 50.008 %
  - Time taken to train the model: 145.99 sec
  - Confusion Matrix Obtained:
$$\begin{bmatrix} 495832 & 5377 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 418247 & 4251 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 47176 & 446 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 47176 & 446 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 20946 & 175 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3814 & 71 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1924 & 72 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1414 & 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 11 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

- No. of Hidden Layer Units: 20
  - Accuracy on Training set: 49.85 %
  - Accuracy on Test set: 49.75 %

  - Time taken to train the model: 156.42 sec
  - Confusion Matrix Obtained:

$$\begin{bmatrix} 485930 & 15279 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 410910 & 11588 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 46471 & 1151 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 20692 & 429 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3733 & 152 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1843 & 153 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1398 & 26 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 229 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 11 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

- No. of Hidden Layer Units: 25
  - Accuracy on Training set: 49.90 %
  - Accuracy on Test set: 49.67 %
  - Time taken to train the model: 171.43 sec
  - Confusion Matrix Obtained:

$$\begin{bmatrix} 480840 & 20369 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 406596 & 15902 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 46020 & 1602 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 20494 & 627 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3677 & 208 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1843 & 153 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1393 & 31 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 225 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 10 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

As we vary the number of hidden layer units, the accuracy on test and training dataset and time taken to train the model varies as shown below:

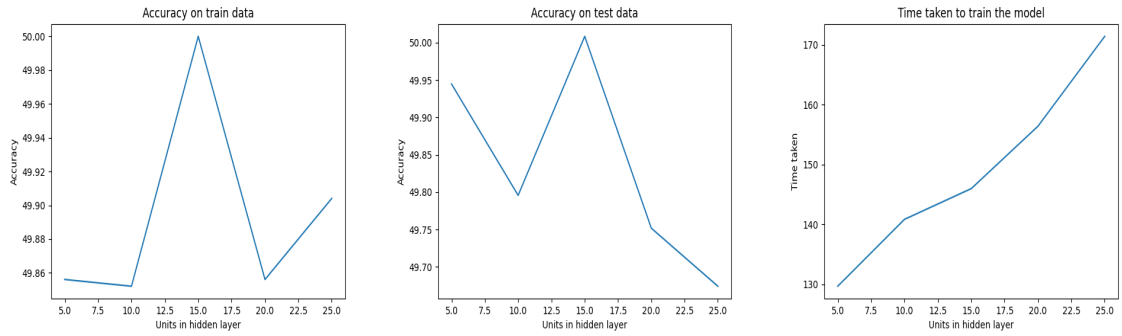

Figure 9: Change in different metrics as the number of units in hidden layer increases in case of adaptive learning rate

Observations:

- The accuracy on test and training dataset almost remains constant as we vary the number of hidden layer units.
- The time taken to train the model increase as we increase the number of hidden layer units.
- The accuracy obtained using adaptive learning rate is smaller as compared to non adaptive learning rate. This is because in case of adaptive learning rate, the model learns comparatively slower in each iteration as the learning rate decreases with each epoch and the maximum epoch limit is reached before the model actually converges.
- Also, the time taken to train the model is greater in this case as compared to non-adaptive case as model learns slower in each iteration and the convergence takes longer time and also the max epoch limit was increased in this case.

2.5 ReLU Activation Unit

In this part, I learnt a model with 2 hidden layers with 100 units in each layer using both sigmoid and ReLU activation units. ReLU is defined using the function: g(z) = max(0, z). The results obtained in both cases are shown below:

Results obtained using Sigmoid Activation unit:

- Training Data accuracy = 49.95 %
- Test Data accuracy = 50.12 %
- Confusion Matrix:

$$\begin{bmatrix} 501209 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 422498 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 47622 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 21121 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3885 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1996 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1424 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 230 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Results obtained using ReLU Activation unit:

- Training Data accuracy = 92.32 %
- Test Data accuracy = 91.77 %
- Confusion Matrix:

$$\begin{bmatrix} 499518 & 1691 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4234 & 418264 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 47622 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 21121 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3821 & 64 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1995 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1424 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 230 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Observations:

- We can observe that using ReLU function increases the accuracy significantly over both the training as well as test data.
- The Sigmoid activation predicted all the samples to be of same class 0 whereas ReLU activation unit predicts samples to be in both class 0 and class 1.

2.6 MLPClassifier

In this part, I implemented the same architecture as in part 2(e) with ReLU activation unit for hidden layers and Sigmoid for output layer using scikit-learn library. The results obtained are shown below:

- Accuracy on Training dataset: 99.96 %
- Accuracy on Test dataset: 97.49 %
- Time taken to train the model: 137.43 sec

Observations:

- We can see that using scikit library gives better accuracy on both train as well as test dataset.
- The time taken to train the model is also very less as compared to that taken by the model in part 2(e) above.