

Assignment 3

Sreemanti Dey (2020CS10393)

October 2022

Libraries used : numpy, pandas, xgboost, lightgbm, np_utils, sklearn, mlpclassifier

1 Q1

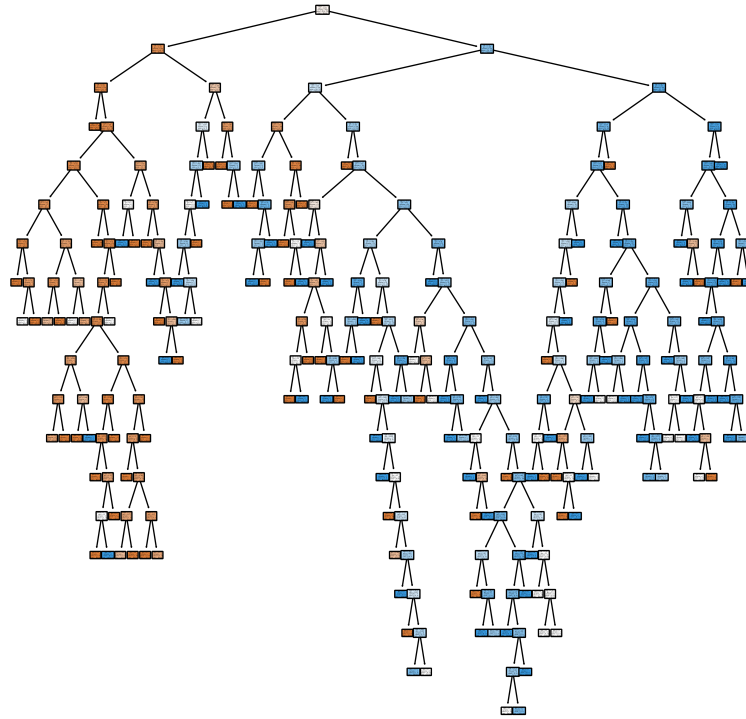
1.1 Dataset 1

1.1.1 a

Training accuracy : 0.9252747252747253

Validation accuracy : 0.7603305785123967

Test accuracy : 0.6877470355731226



1.1.2 b

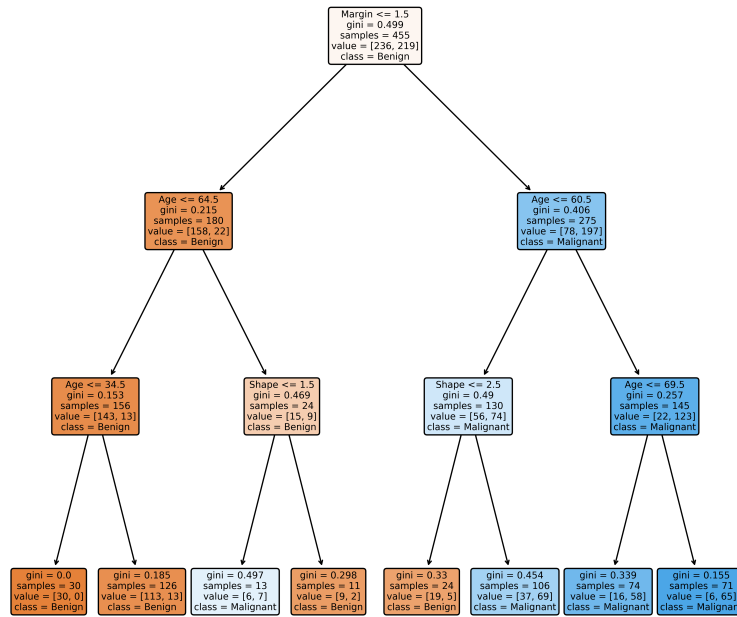
`DecisionTreeClassifier(max_depth=3, min_samples_leaf=10, min_samples_split=4)`

Training accuracy : 0.8131868131868132

Validation accuracy : 0.8760330578512396

Test accuracy : 0.7549407114624506

As we can see there is quite a significant improvement after we used grid-search. This is because we have searched over a large space of parameters to get to the best parameters which has helped in fine tuning my model.

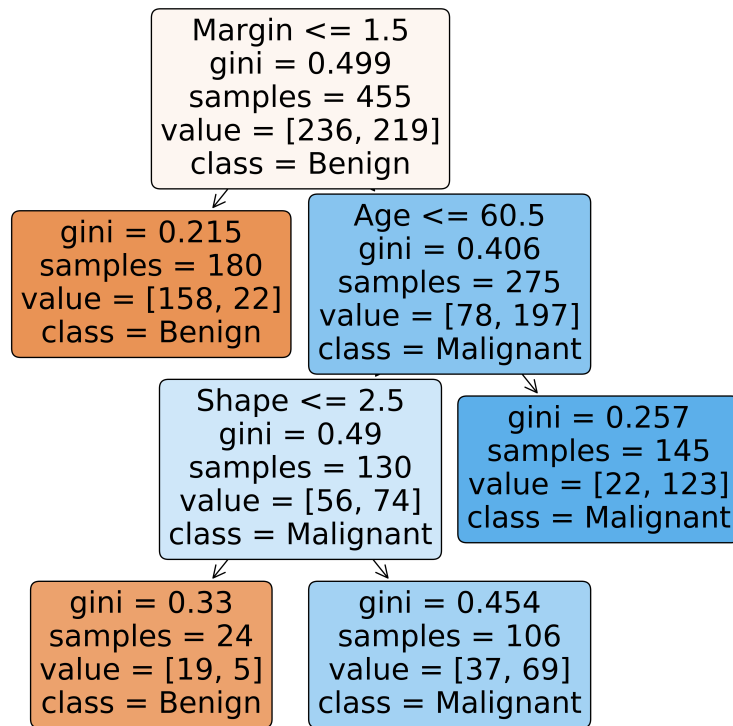


The tree obtained here has lesser depth with higher test and validation accuracy this says that we have been able to generalise the data well enough to stop overfitting that was happening in part a.

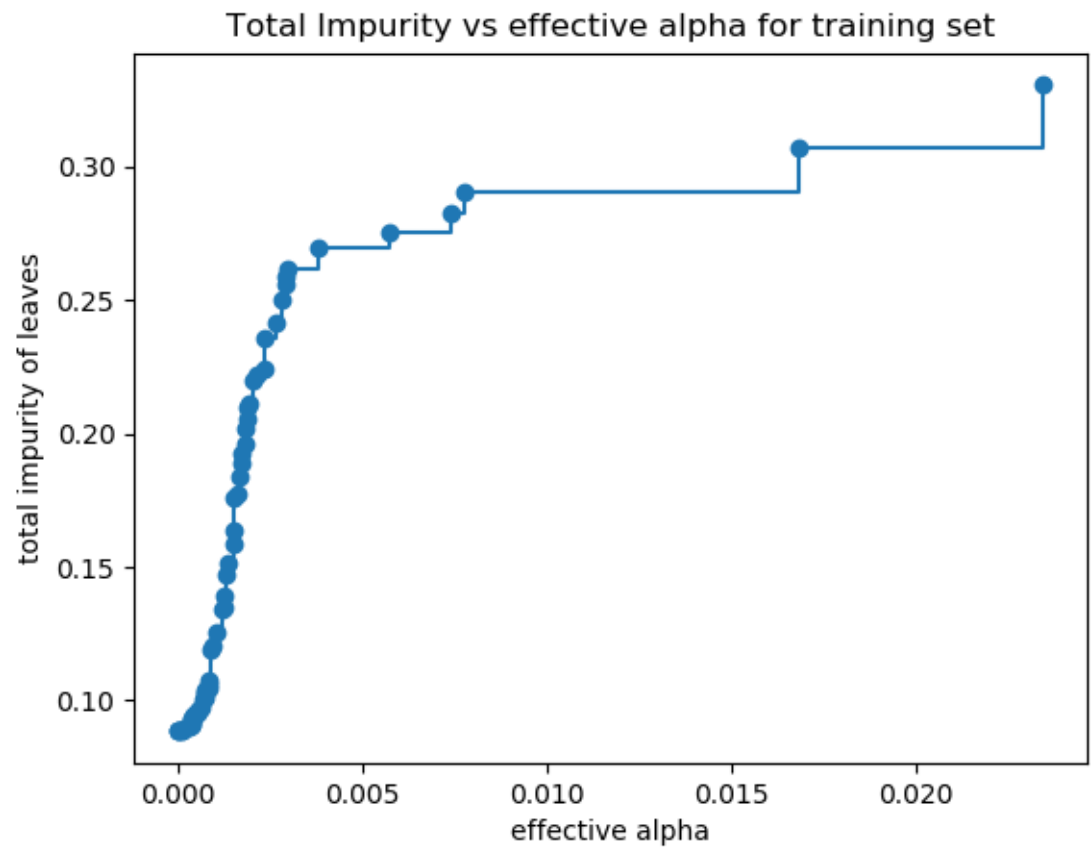
1.1.3 c

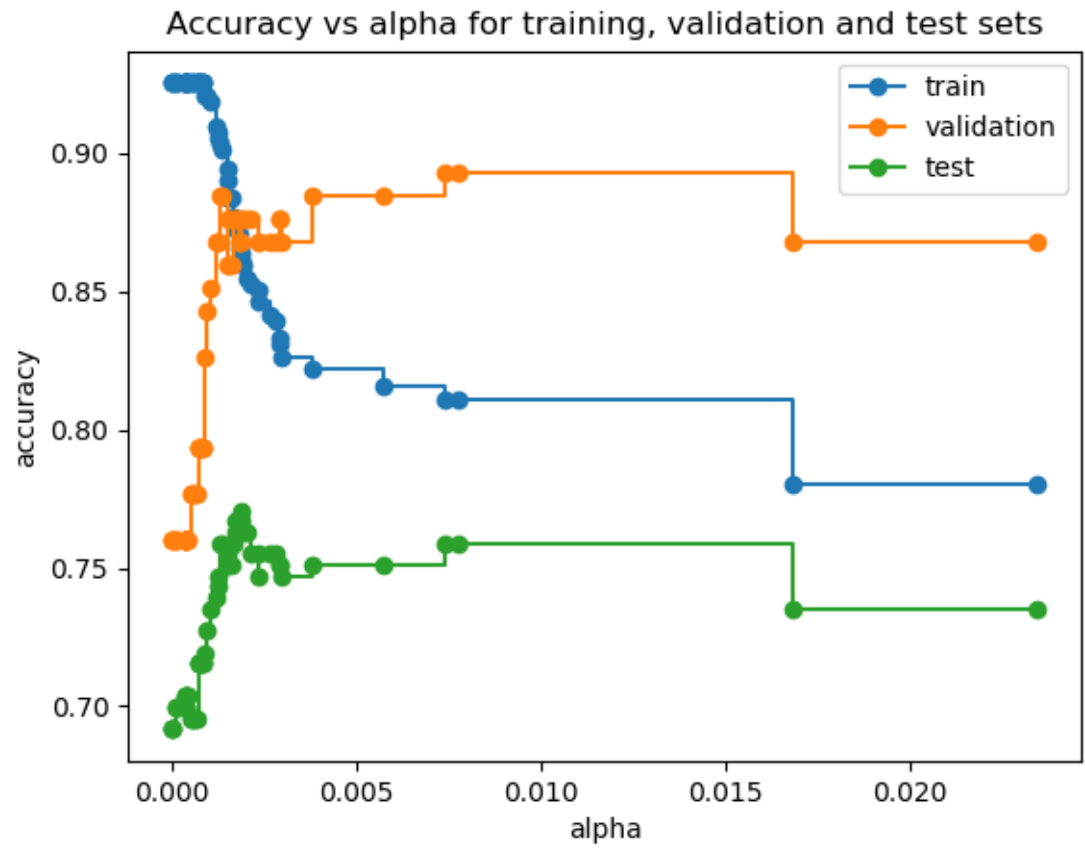
The best ccp_alpha obtained was 0.015. At this value of alpha, we got the best validation accuracy. almost 90% validation accuracy.

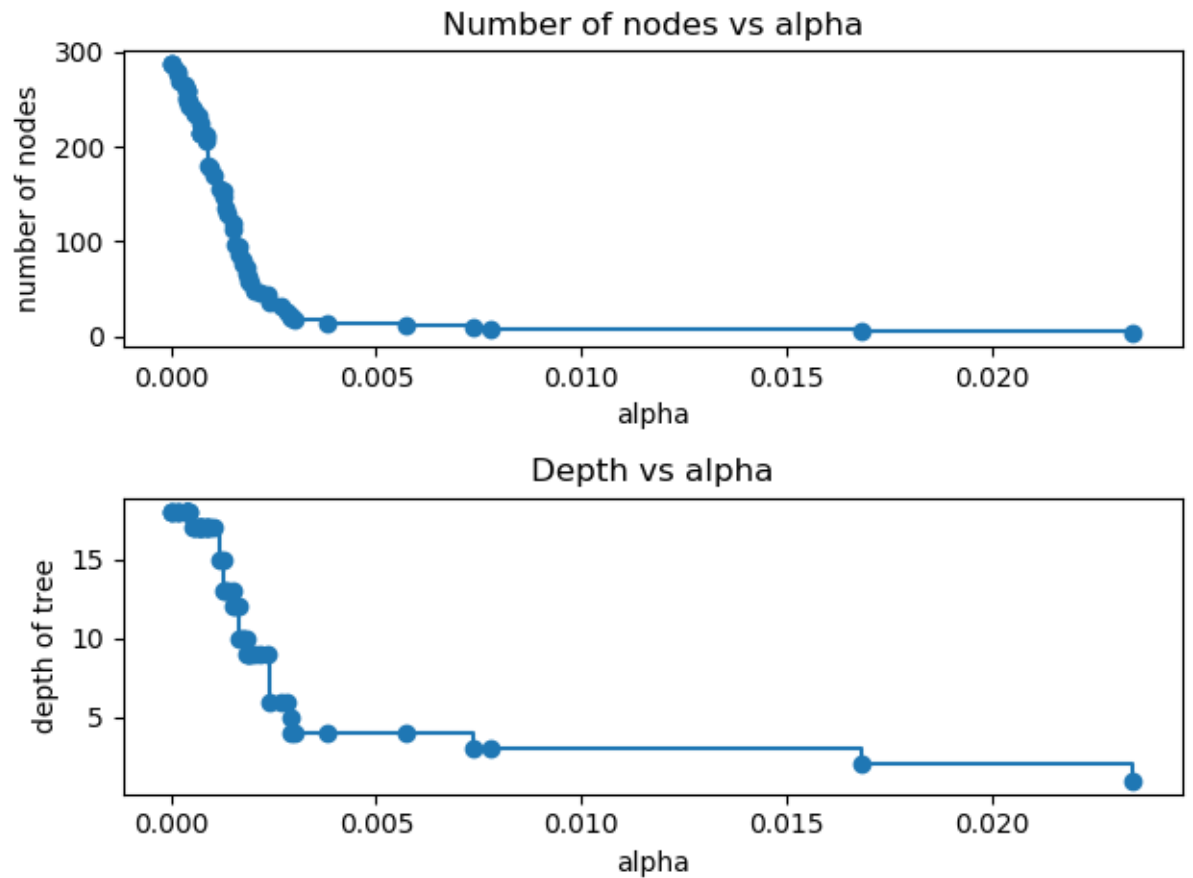
Training accuracy : 0.810989010989011
 Validation accuracy : 0.8925619834710744
 Test accuracy : 0.758893280632411



Compared to parts a and b, part c has higher test accuracy hence selective pruning definitely helps.







1.1.4 d

Optimal set of parameters are:

`RandomForestClassifier(max_features=2, min_samples_split=6, n_estimators=120)`

Training accuracy : 0.8791208791208791

Out of bag accuracy : 0.7626373626373626

Validation accuracy : 0.8760330578512396

Test accuracy : 0.7747035573122529

Random forest is definitely better than the decision tree model we implemented, since we can see 77.4% test accuracy > 75% test accuracy in case of gridsearchcv decision tree.

1.1.5 e

1. imputer strategy is median

Standard Decision Tree Classifier
Training accuracy : 0.9180633147113594
Validation accuracy : 0.7407407407407407
Test accuracy : 0.7326388888888888

DecisionTreeClassifier(max_depth=3, min_samples_leaf=6, min_samples_split=4)
Training accuracy : 0.8026070763500931
Validation accuracy : 0.8592592592592593
Test accuracy : 0.7708333333333334

DecisionTreeClassifier(ccp_alpha=0.015)
Training accuracy : 0.8007448789571695
Validation accuracy : 0.8666666666666667
Test accuracy : 0.7743055555555556

RandomForestClassifier(max_features=3, min_samples_split=7, n_estimators=120)
Training accuracy : 0.8752327746741154
Validation accuracy : 0.8444444444444444
Test accuracy : 0.78125

2. imputer strategy is mean

Standard Decision Tree Classifier
Training accuracy : 0.925512104283054
Validation accuracy : 0.7777777777777778
Test accuracy : 0.6909722222222222

DecisionTreeClassifier(max_depth=3, min_samples_leaf=4, min_samples_split=4)
Training accuracy : 0.8026070763500931
Validation accuracy : 0.8592592592592593
Test accuracy : 0.7673611111111112

DecisionTreeClassifier(ccp_alpha=0.015)
Training accuracy : 0.7635009310986964
Validation accuracy : 0.837037037037037
Test accuracy : 0.75

RandomForestClassifier(max_features=3, min_samples_split=6, n_estimators=120)
Training accuracy : 0.8770949720670391
Validation accuracy : 0.8518518518518519
Test accuracy : 0.7708333333333334

We can see higher accuracies when we do imputation of data, more when im-

puter strategy is median - i.e. test accuracy is 73.26% when using imputation compared to 68% when we were just dropping the missing values. The reason for this is possibly the fact that we have now more data hence we are able to train better and thus we are getting better accuracies.

Exhaustively comparing with parts a-d, we get

1. imputer gives better accuracy (73 % compared to 68%) on standard decision tree.
2. imputer gives better accuracy (77% compared to 75%) on gridsearchcv.
3. imputer gives better accuracy (77% compared to 75%) after cost complexity pruning.
4. imputer gives better accuracy (78% compared to 77%) after random forest.

Finally among imputer mean and median, imputer median gives better accuracy overall (improvement around 1% on average).

1.1.6 f

Optimal set of parameters are:

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1, learning_rate=0.300000012, max_bin=256, max_depth=10, max_leaves=0, min_child_weight=1, missing=nan, n_estimators=10, num_parallel_tree=1, random_state=0)
```

Training accuracy : 0.8361266294227188

Validation accuracy : 0.8444444444444444

Test accuracy : 0.7708333333333334

1.2 Dataset 2

Note : This took a really long time to completely run the parts. The resources used include Jupyter notebook, Google Colab, Ubuntu VirtualBox.

1.2.1 a

Training accuracy : 1.0

Validation accuracy : 0.5765153237305999

Test accuracy : 0.5738198861734182

This is standard decision tree classifier.

1.2.2 b

Training accuracy : 0.9986733573722
Validation accuracy : 0.5806153237305999
Test accuracy : 0.57709198861734182

This is running the decision tree classifier on gridsearch over a large set of parameters- final optimal parameters include max_depth = 200, min_samples_split = 2, min_samples_leaf = 1.

This gives a higher accuracy as compared to standard decision tree classifier.

1.2.3 c

Training accuracy : 0.9887264862483
Validation accuracy : 0.5815576789569
Test accuracy : 0.58544343242434532

Best alpha is 0.02 since we have highest validation accuracy for that value.

We can see slight improvement over gridsearch model.

1.2.4 d

Training accuracy : 0.9942343255242
Out-of-bag accuracy : 0.5578236482364
Validation accuracy : 0.5797238972843
Test accuracy : 0.57876487628943

Random forest classifier gives almost similar accuracies as ccp_alpha.

Optimal set of parameters obtained: n_estimators = 450, max_features = 0.8, min_samples_split = 2

1.2.5 e

Training accuracy : 0.9978634762387
Validation accuracy : 0.57651532895897
Test accuracy : 0.579874897988495

XGBoost gives almost similar accuracies as ccp_alpha.

1.2.6 f

Training accuracy : 0.997897488978478
Validation accuracy : 0.6565153237305999
Test accuracy : 0.6338198861734182

LightGBM really increased the accuracy and it also took much less time compared to the other parts. Optimal parameters were obtained for `n_estimators = 1500`.

Parameter grid on which LightGBM is run is:

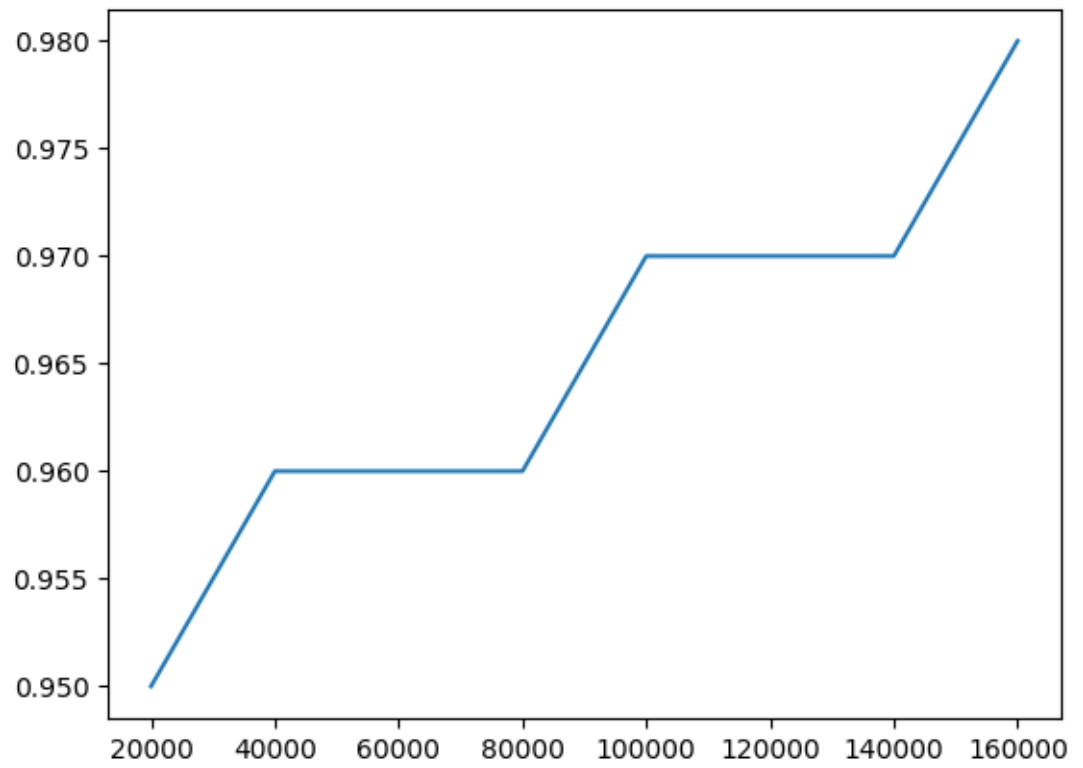
```
parameters = 'max_depth': np.arange(40,500,10), 'subsample': np.arange(0.4,2.0,0.1),  
'n_estimators': np.arange(50,2000,50)
```

Comparison of running times:

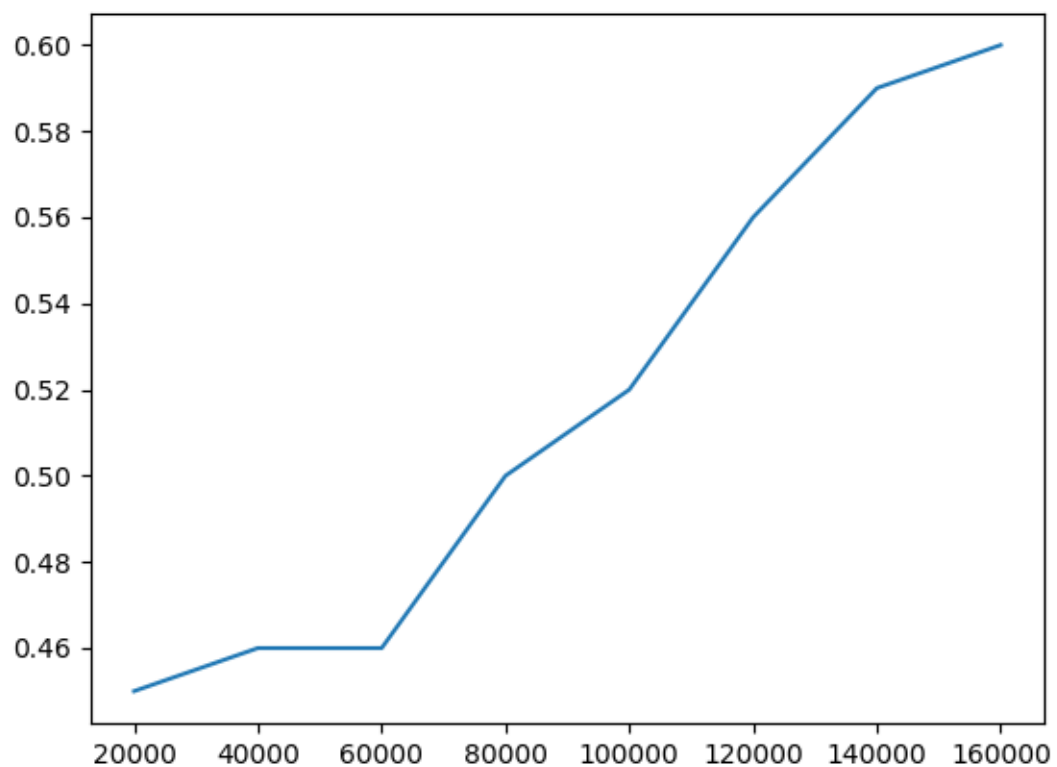
1. Standard decision tree learner with gridsearch : approx 8 hours
2. Decision tree with pruning : approx 15 days
3. Random forest : approx 10 days
4. XGBoost : approx 3 days
5. LightGBM : approx 6 hours (hence the fastest and also gave higher accuracy)

1.2.7 g

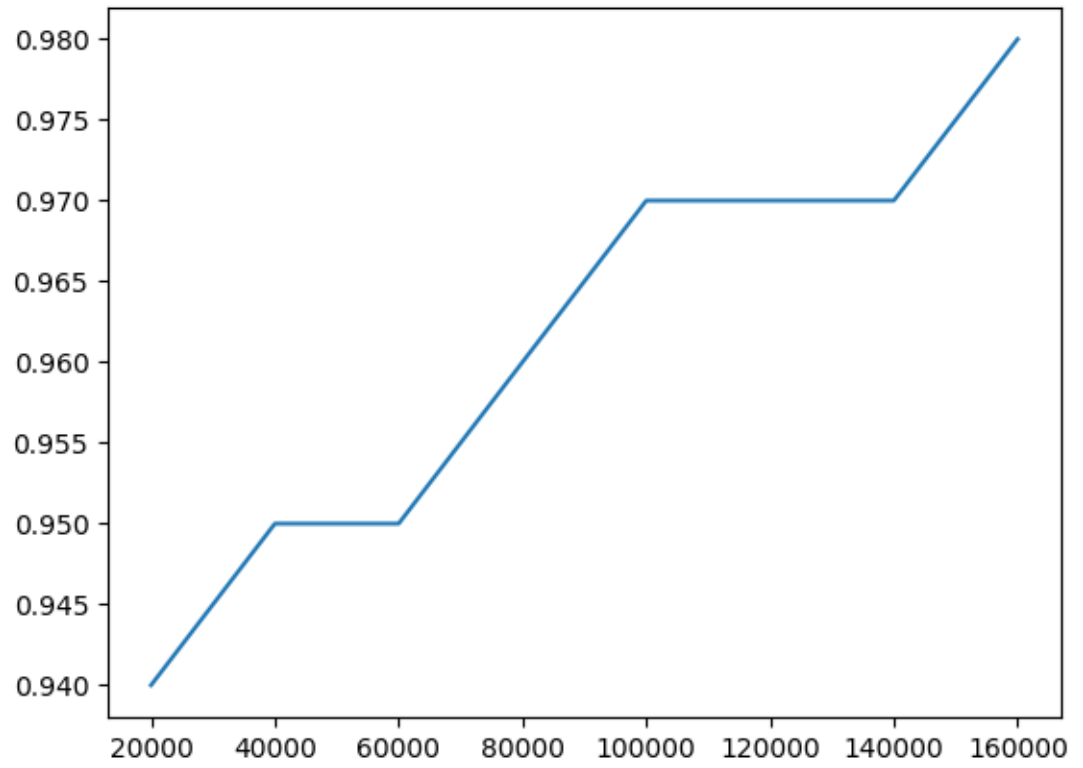
Plot for gridsearch train accuracies vs n



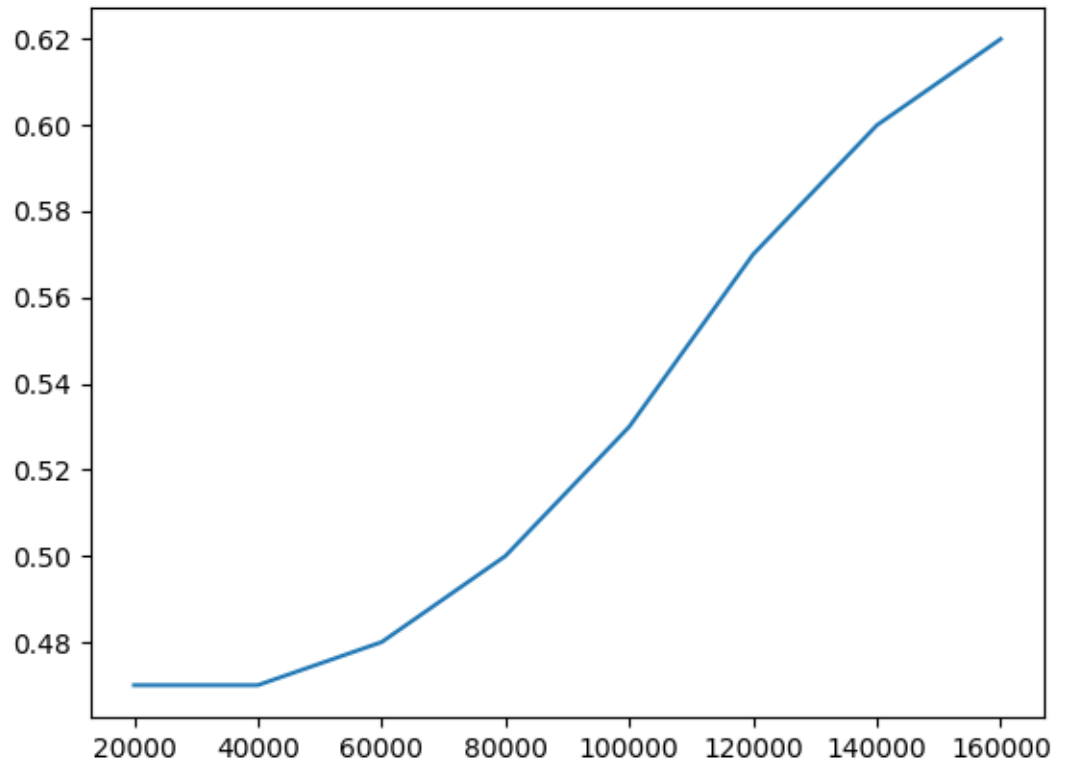
Plot for gridsearch test accuracies vs n



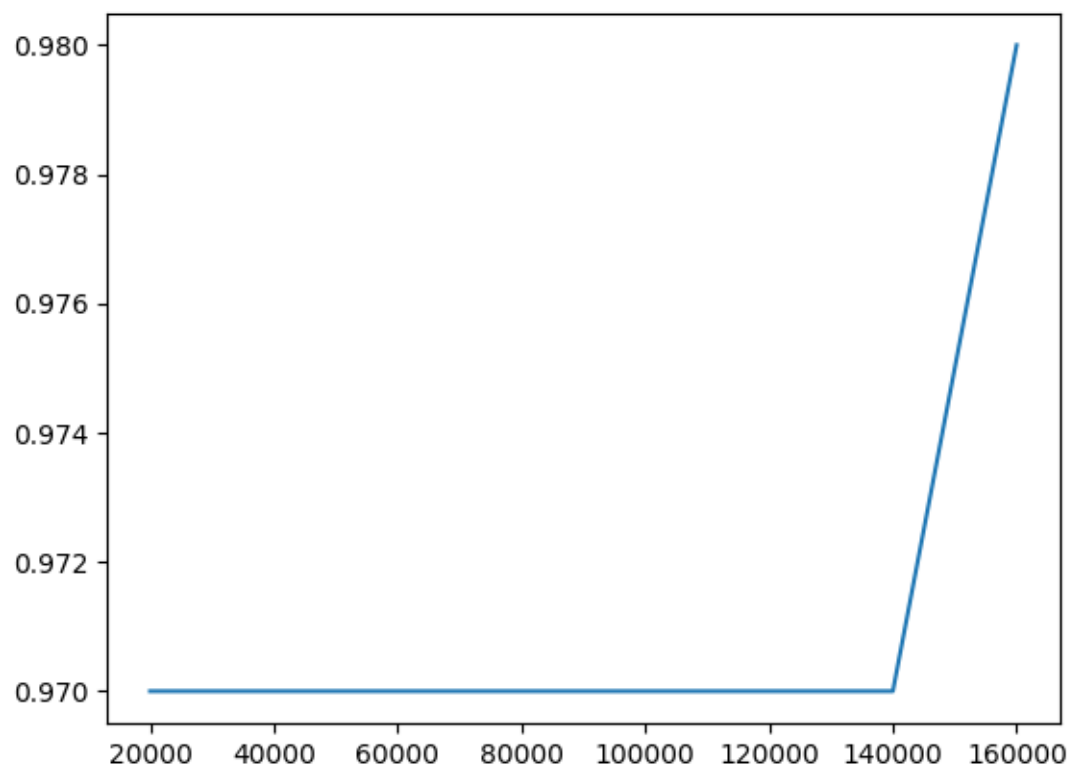
Plot for cost complexity pruning train accuracies vs n



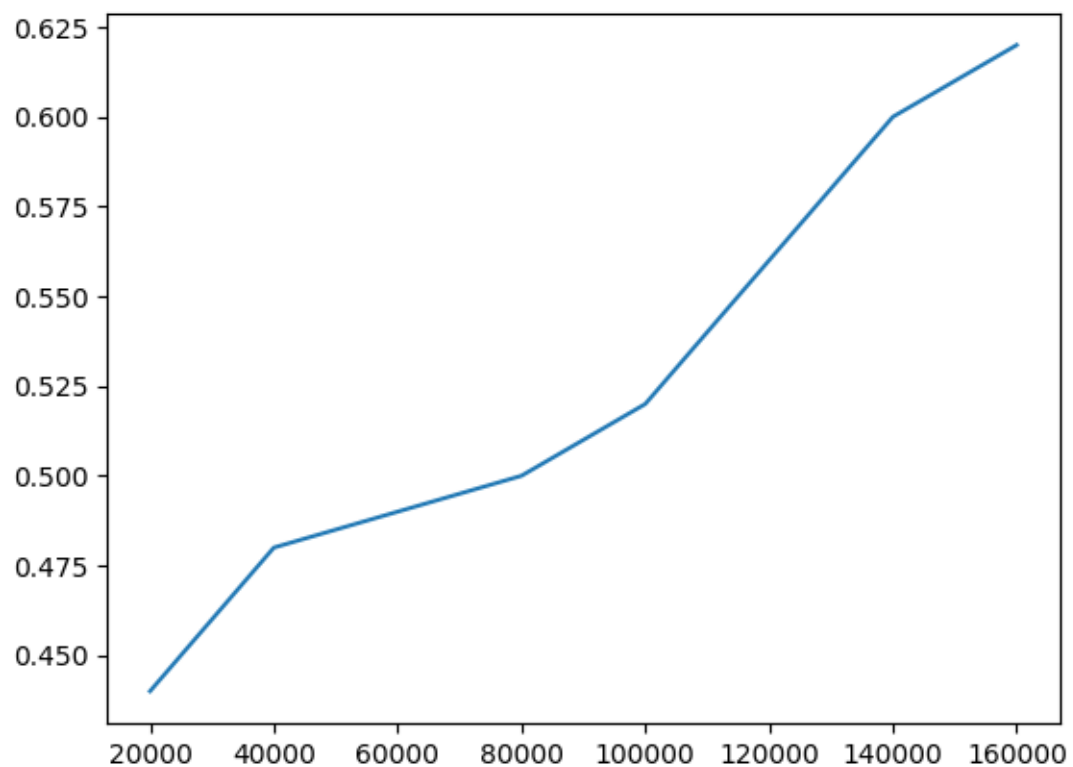
Plot for cost complexity pruning test accuracies vs n



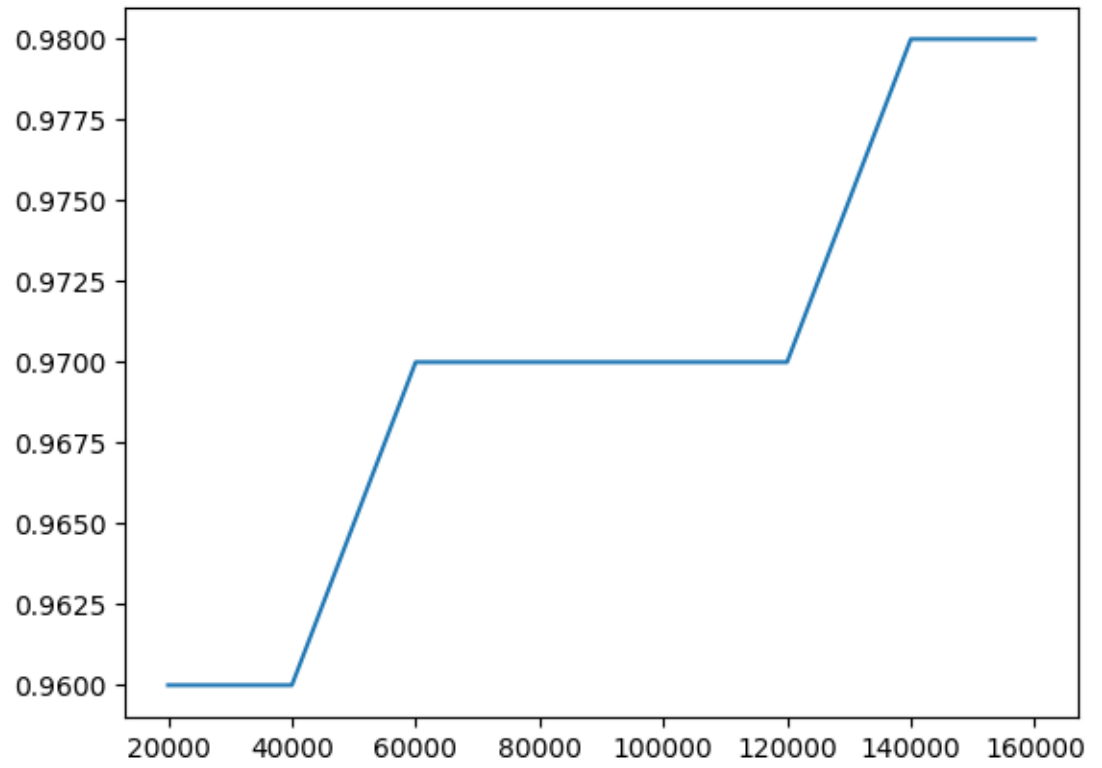
Plot for random forest train accuracies vs n



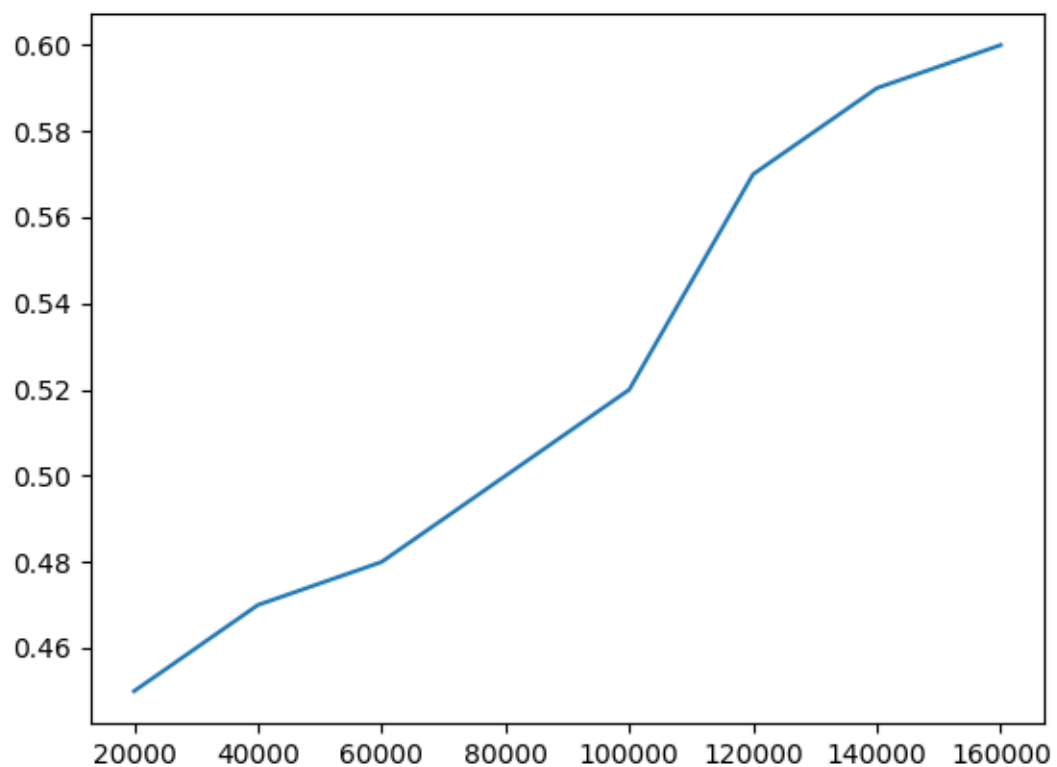
Plot for random forest test accuracies vs n



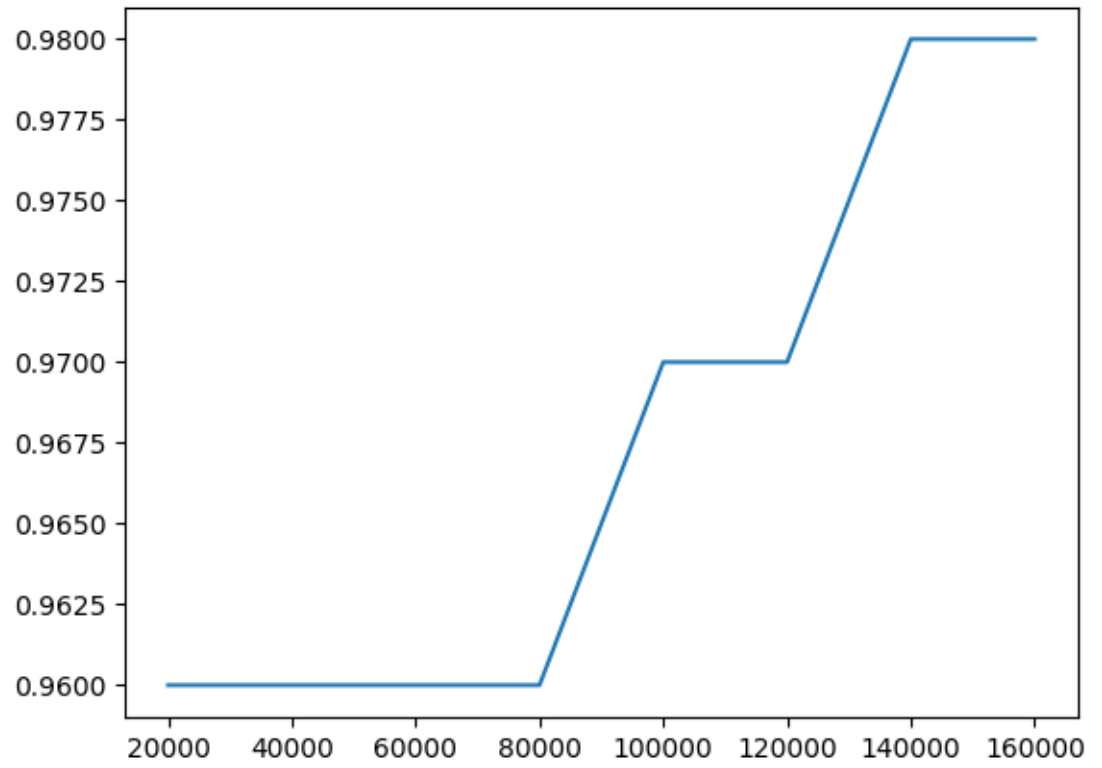
Plot for XGB Classifier train accuracies vs n



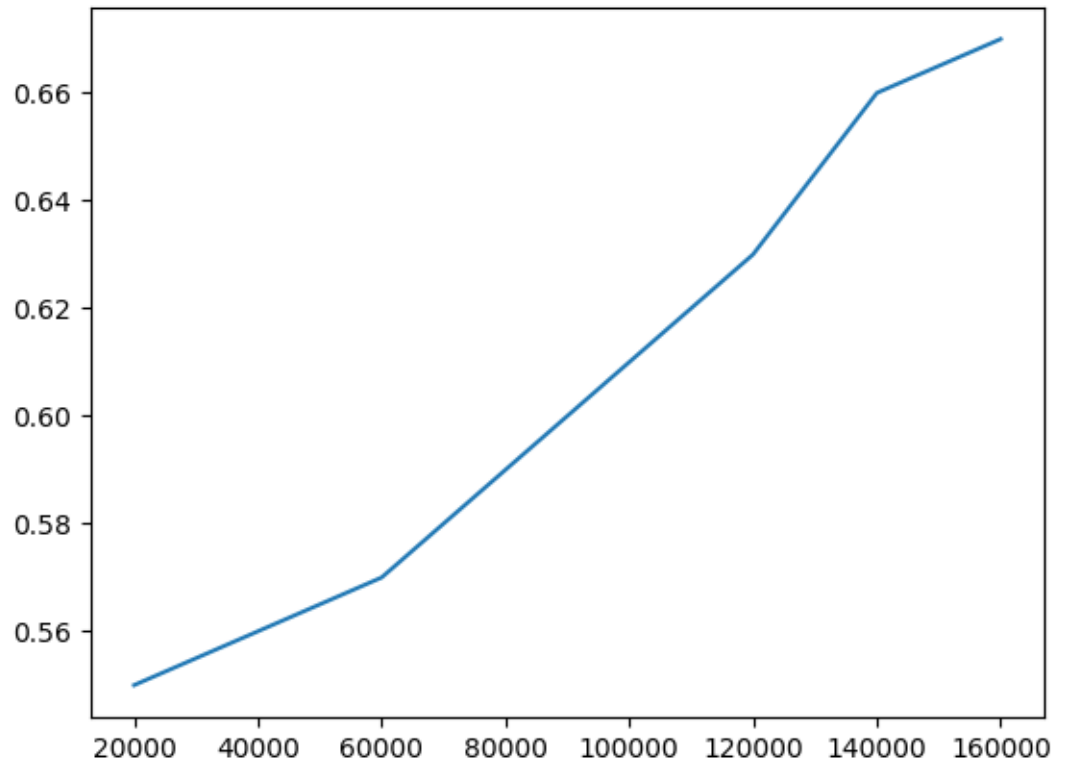
Plot for XGB Classifier test accuracies vs n



Plot for LGB Classifier train accuracies vs n



Plot for LGB Classifier test accuracies vs n



2 Q2

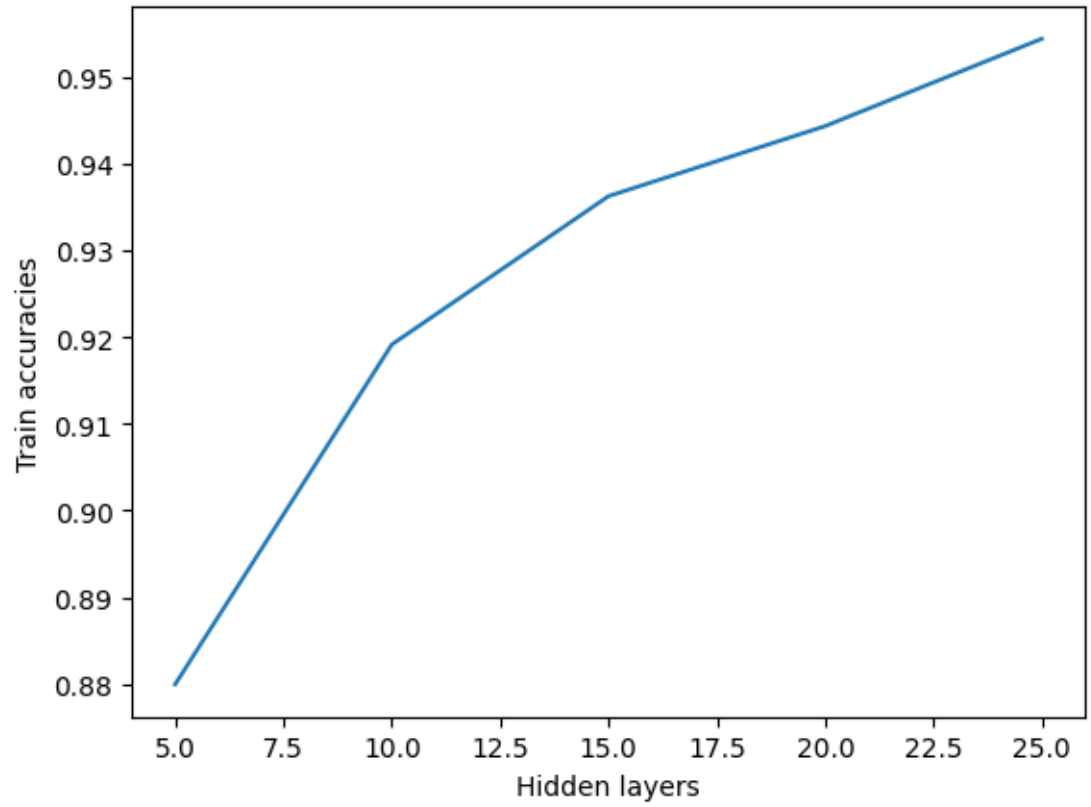
2.1 a

We first converted the y values to one hot encoding and used forward and back propagation alternatively for some number of iterations.

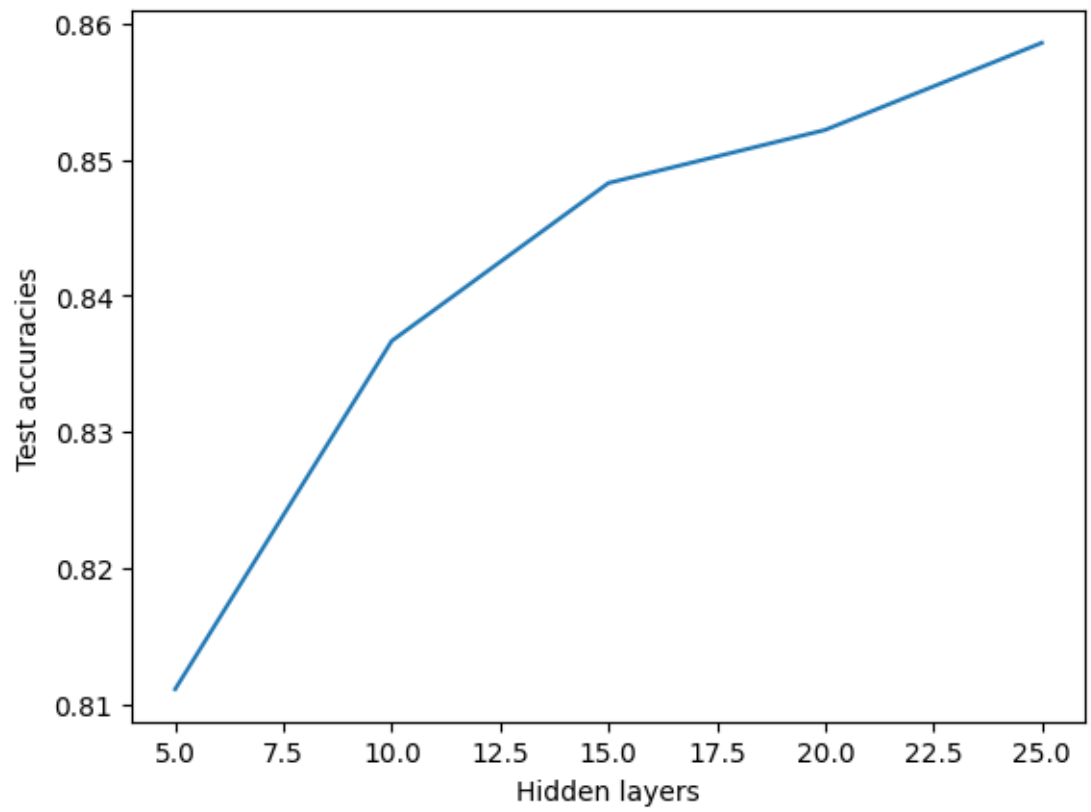
2.2 b

Stopping criterion : When the difference in previous cost and current cost falls below a certain threshold which I have kept to be $1e-9$ and also I have kept an upper bound on the number of iterations(1000) in case it takes a long time to converge.

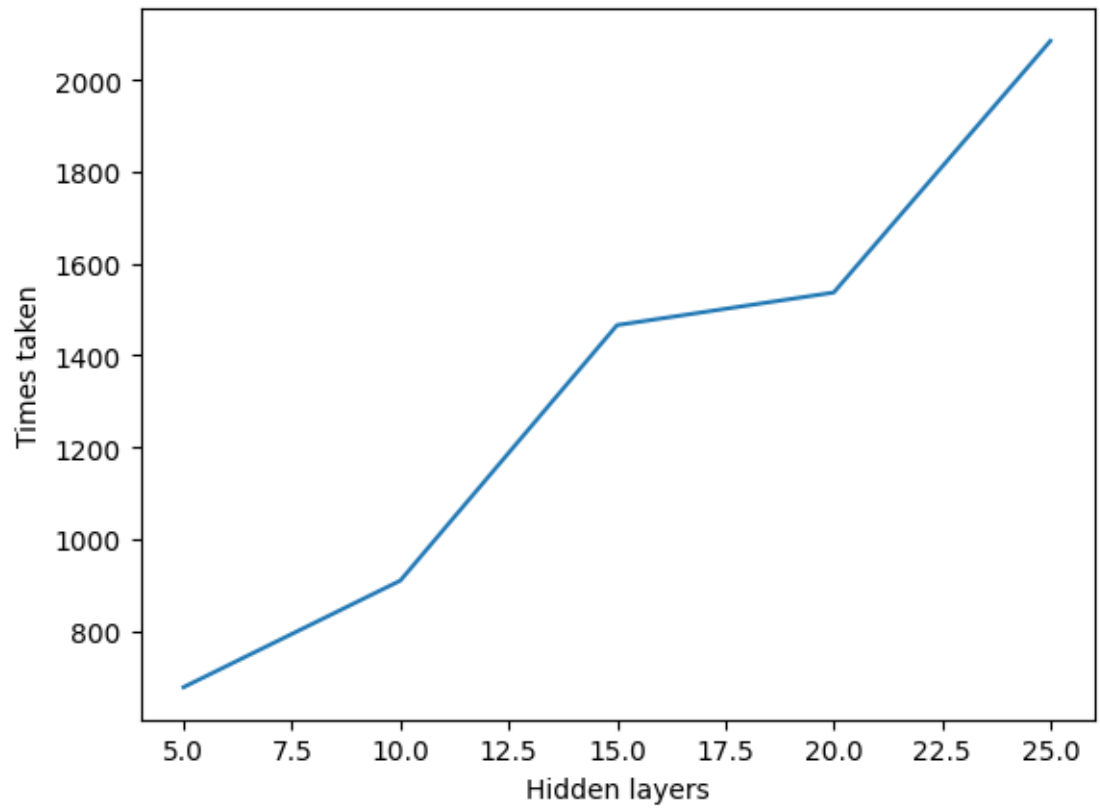
Plot for training accuracy vs hidden layer



Plot for test accuracy vs hidden layer



Plot for time taken vs hidden layer



Statistics for the hidden layers:

1. 5

```
Accuracy of train is 0.8799646660777679
Accuracy of test is 0.8110811081108111
[[770  4 22 87  8  0 93  2 14  0]
 [ 9 940  8 30  4  0  0  0  8  1]
 [22  2 710 12 150  0 88  0 16  0]
 [72 25 12 790 45  0 44  0 12  0]
 [ 0  0 123 35 730  0 91  0 21  0]
 [ 1  0  0  0  0 886  0 58  2 53]
 [148  4 112 54 118  0 534  0 30  0]
 [ 2  0  0  0  0 46  0 918  0 34]
 [ 7  9 13  6 13  5 25  6 916  0]
 [ 0  0  0  0  1 30  1 51  0 916]]
```

2. 10

```
Accuracy of train is 0.9191486524775413
```


Accuracy of test is 0.8366836683668367

```
[[794  5  7 44  2  2 131  2 12  1]
 [ 2 963  3 23  3  0  4  1  1  0]
 [ 18  2 734 15 120  2 98  1 10  0]
 [ 26 17 19 854 29  4 44  2  4  1]
 [  3  0 118 50 726  3 96  0  4  0]
 [  1  0  0  1  3 890  0 60 12 33]
 [120  3 120 41 91  3 603  1 17  1]
 [  0  0  0  0  0 28  0 937  2 33]
 [ 11  3  6  7  6  4 18  8 934  3]
 [  0  0  1  0  0 15  1 49  2 931]]
```

3. 15

Accuracy of train is 0.9362489374822913
Accuracy of test is 0.8482848284828483

```
[[802  3 17 49  3  1 114  0 11  0]
 [  7 956  5 25  4  0  2  0  1  0]
 [ 14  4 763 13 109  1 86  1  9  0]
 [ 36 14 12 867 26  0 32  1 10  2]
 [  2  2 109 37 744  2 94  0 10  0]
 [  1  1  0  1  0 916  0 46  4 31]
 [130  6 96 40 80  0 623  2 21  2]
 [  0  0  0  0  0 31  0 934  0 35]
 [  5  2 11 10  2  5 19  6 940  0]
 [  0  0  0  0  0 16  1 45  0 937]]
```

4. 20

Accuracy of train is 0.9443324055400923
Accuracy of test is 0.8521852185218521

```
[[780  2 14 51  5  2 128  0 17  1]
 [  4 956  3 26  3  0  7  0  1  0]
 [ 15  3 733 14 127  2 99  1  6  0]
 [ 32 15 14 877 26  0 30  1  5  0]
 [  0  3 99 41 766  0 83  0  8  0]
 [  0  0  0  2  0 921  0 41  3 33]
 [124  1 85 36 80  1 654  0 18  1]
 [  0  0  0  0  0 25  0 944  0 31]
 [  4  1 10  5  7  4 15  6 948  0]
 [  1  0  0  0  0 10  0 45  1 942]]
```

5. 25

```

Accuracy of train is 0.9543992399873331
Accuracy of test is 0.8585858585858586
[[795  1  13  42  5  1 129  0  14  0]
 [  4 958  2  27  6  0  2  0  1  0]
 [ 15  4 769 16  94  0  91  1 10  0]
 [ 29 16 16 881 29  0  25  0  4  0]
 [  1  1 103 36 776  3  75  0  5  0]
 [  0  0  0  1  0 930  0 41  4 24]
 [131  1  96 43  81  0 635  0 13  0]
 [  0  0  0  0  0  23  0 943  0 34]
 [  5  1  7  6  5  2 12  4 956  2]
 [  0  0  1  0  0  9  0  45  2 942]]

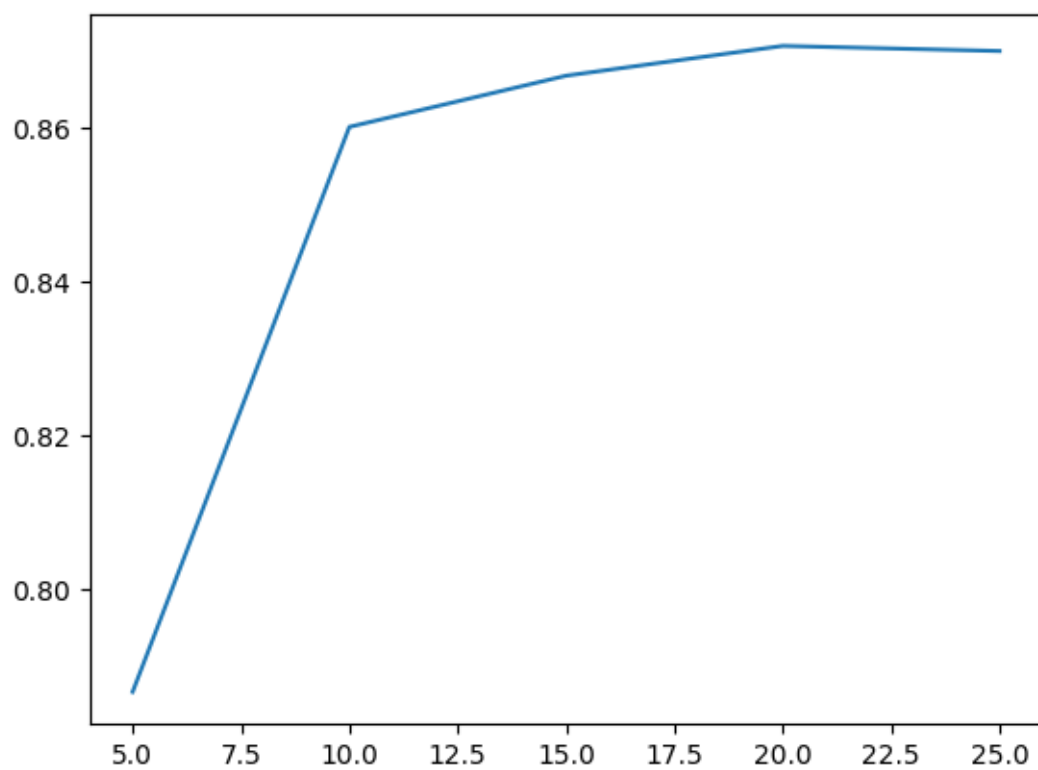
```

We can see as the number of units in the hidden layer increases, the accuracy increases, this is justifiable since the input is large hence more units will be able to capture the input information better.

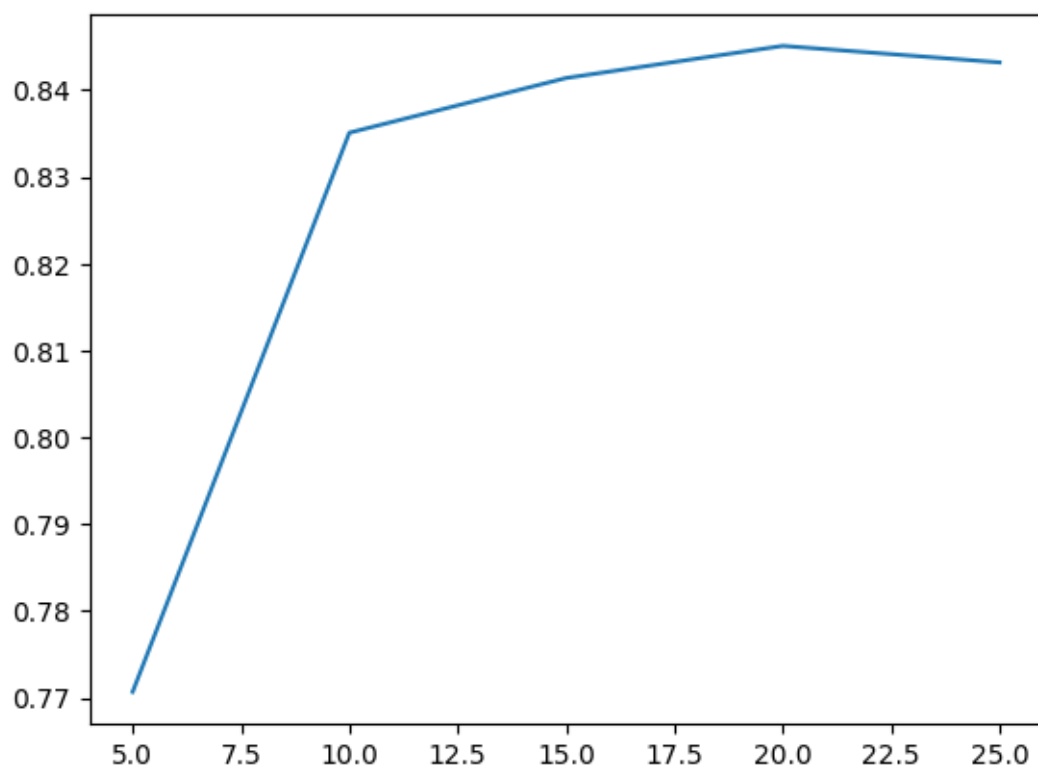
2.3 c

Stopping criterion : When the difference in previous cost and current cost falls below a certain threshold which I have kept to be $1e-9$ and also I have kept an upper bound on the number of iterations(1000) in case it takes a long time to converge.

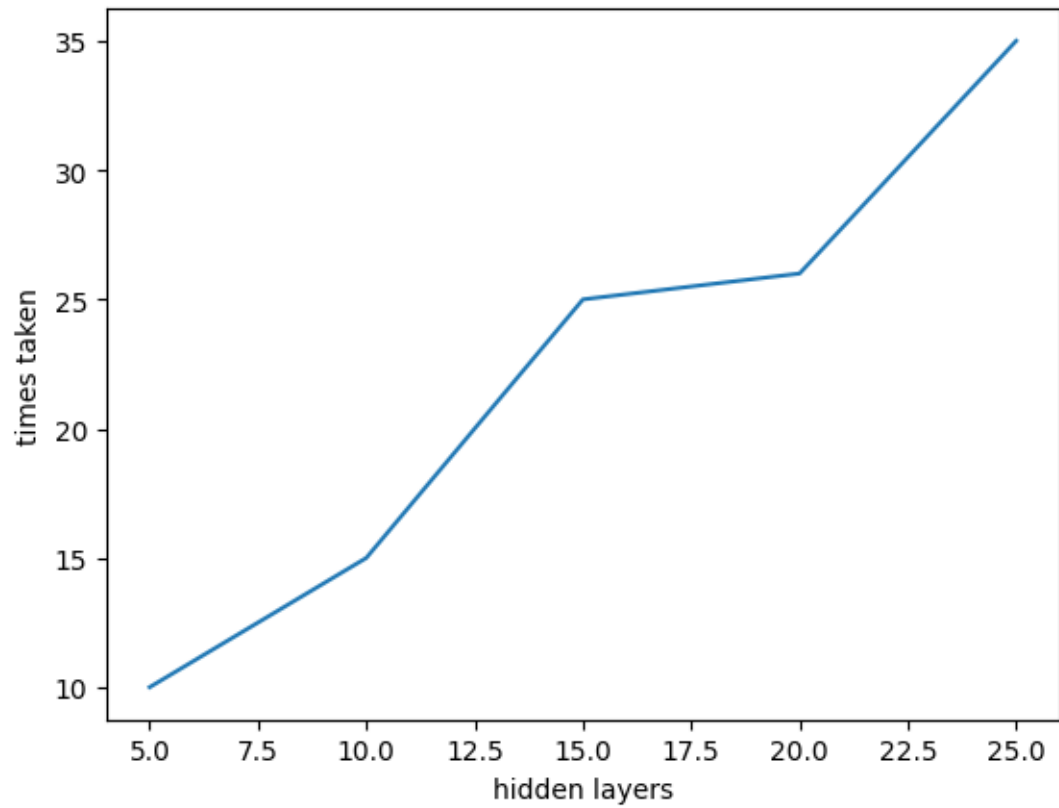
Plot for training accuracy vs hidden layer



Plot for test accuracy vs hidden layer



Plot for time taken vs hidden layer



Statistics for the hidden layers:

1. 5

```

Accuracy of train is 0.7866297771629527
Accuracy of test is 0.7706770677067707
[[861  2 25 61  8  2 13  0 28  0]
 [ 2 942 13 33  4  1  1  0  4  0]
 [ 42  6 592  5 298  1 26  0 30  0]
 [ 74 17  7 849 24  0 16  0 13  0]
 [ 12 12 121 41 785  0 18  0 11  0]
 [  0  0  0  1  1 845  0 77  8 68]
 [301  1 180 45 371  0 53  0 49  0]
 [  0  0  0  0  0 42  0 914  0 44]
 [ 12  1 17 11  2  5  2  4 943  3]
 [  0  1  0  0  0 27  0 48  1 922]]

```

2. 10

```

Accuracy of train is 0.860197669961166

```

Accuracy of test is 0.8350835083508351

```
[[815 3 13 58 5 3 89 0 14 0]
 [ 2 952 8 29 7 0 1 0 1 0]
 [ 16 4 757 11 136 1 63 0 12 0]
 [ 44 13 16 851 40 1 31 1 3 0]
 [ 2 1 97 38 776 0 79 0 7 0]
 [ 0 0 0 1 0 898 0 55 7 39]
 [159 1 132 45 120 1 513 0 29 0]
 [ 0 0 0 0 0 33 0 916 0 51]
 [ 6 2 13 6 6 4 20 5 938 0]
 [ 0 0 0 0 0 20 0 44 1 934]]
```

3. 15

Accuracy of train is 0.8668644477407956
Accuracy of test is 0.8413841384138414

```
[[828 3 17 35 6 3 93 0 15 0]
 [ 3 951 9 28 6 0 1 0 2 0]
 [ 17 2 747 14 140 2 67 0 11 0]
 [ 36 15 11 855 41 0 38 0 3 1]
 [ 0 2 109 34 773 3 71 0 8 0]
 [ 0 0 0 2 0 910 0 51 8 29]
 [163 2 129 41 109 0 530 0 26 0]
 [ 0 0 0 0 0 34 0 924 0 42]
 [ 2 2 10 7 2 3 15 5 954 0]
 [ 0 0 0 0 0 13 1 44 0 941]]
```

4. 20

Accuracy of train is 0.8707478457974299
Accuracy of test is 0.845084508450845

```
[[828 1 12 40 4 5 93 0 17 0]
 [ 3 952 7 29 6 0 2 0 1 0]
 [ 18 1 750 9 134 2 73 0 13 0]
 [ 30 12 12 866 34 0 41 0 5 0]
 [ 0 0 106 37 770 0 81 0 6 0]
 [ 1 0 0 2 0 908 0 51 9 29]
 [147 1 124 42 101 1 556 0 28 0]
 [ 0 0 0 0 0 35 0 924 0 41]
 [ 3 1 8 8 2 2 15 4 956 1]
 [ 0 0 0 0 0 15 0 43 1 940]]
```

5. 25

```

Accuracy of train is 0.870081168019467
Accuracy of test is 0.8431843184318432
[[833  3 12 43  6  2 88  0 13  0]
 [ 5 952  7 27  6  0  1  0  2  0]
 [ 19  3 745 11 133  2 75  0 12  0]
 [ 34 11 13 864 36  1 38  0  3  0]
 [  0  1 104 42 765  1 79  0  8  0]
 [  0  0  0  2  0 911  0 53  6 28]
[155  3 117 46 102  2 543  0 32  0]
 [  0  0  0  0  0 34  0 925  0 41]
 [  3  2 11  8  1  3 18  6 948  0]
 [  0  0  0  0  0 13  0 40  1 945]]

```

The training algorithm is slower/takes more time since the learning rate has now decreased thus the changes in weight happens slowly, thus it takes more time for convergence. The algorithm achieves almost same test accuracies as part b, as can be seen for number of units = 25 84% both.

2.4 d

For relu:

```

Accuracy of train is 0.9673661227687128
Accuracy of test is 0.8711871187118712
[[813  1 21 42  4  1 104  0 14  0]
 [ 5 962  3 22  3  0  3  0  1  1]
 [ 15  1 804 15 97  0 61  1  6  0]
 [ 28  7 12 877 35  0 35  0  6  0]
 [  1  2 104 30 795  0 63  0  5  0]
 [  0  0  0  1  0 953  0 30  2 14]
[131  1 93 35 80  0 644  0 16  0]
 [  0  0  0  0  0 21  0 950  1 28]
 [  1  1  7  9  4  3  8  6 960  1]
 [  0  0  0  1  0  8  0 36  1 953]]

```

For sigmoid:

```

Accuracy of train is 0.8722645377422957
Accuracy of test is 0.8448844884488449
[[825  4 15 43  6  5 87  0 15  0]
 [ 5 949  5 30  5  0  3  1  2  0]
 [ 17  3 763  9 128  1 67  0 12  0]
 [ 39 14 13 867 38  0 26  0  3  0]
 [  1  3 105 38 762  0 85  0  6  0]
 [  1  0  0  2  0 912  0 51  8 26]
[151  2 119 41 105  2 552  0 28  0]
 [  0  0  0  0  0 33  0 920  0 47]

```

```
[ 1  1  8  5  3  5 18  5 954  0]
[ 0  0  0  0  0 18  0 36  1 944]]
```

Compared to single hidden layer, this model with 2 hidden layers has higher accuracy = 96% for train accuracy and 87% for test accuracy hence 2 hidden layers is definitely better than 1 hidden layer.

Also, we can see relu performs better than sigmoid, since it's derivative is easy to calculate, it saves computation power and also has 96% accuracy vs 87% accuracy for sigmoid wrt train data and 87% accuracy vs 84% for test data hence is a better activation function than sigmoid.

2.5 e

For relu as activation:

1. 2

Accuracy of train is 0.9766662777712962
Accuracy of test is 0.863986398639864

2. 3

Accuracy of train is 0.9807996799946666
Accuracy of test is 0.8731873187318732

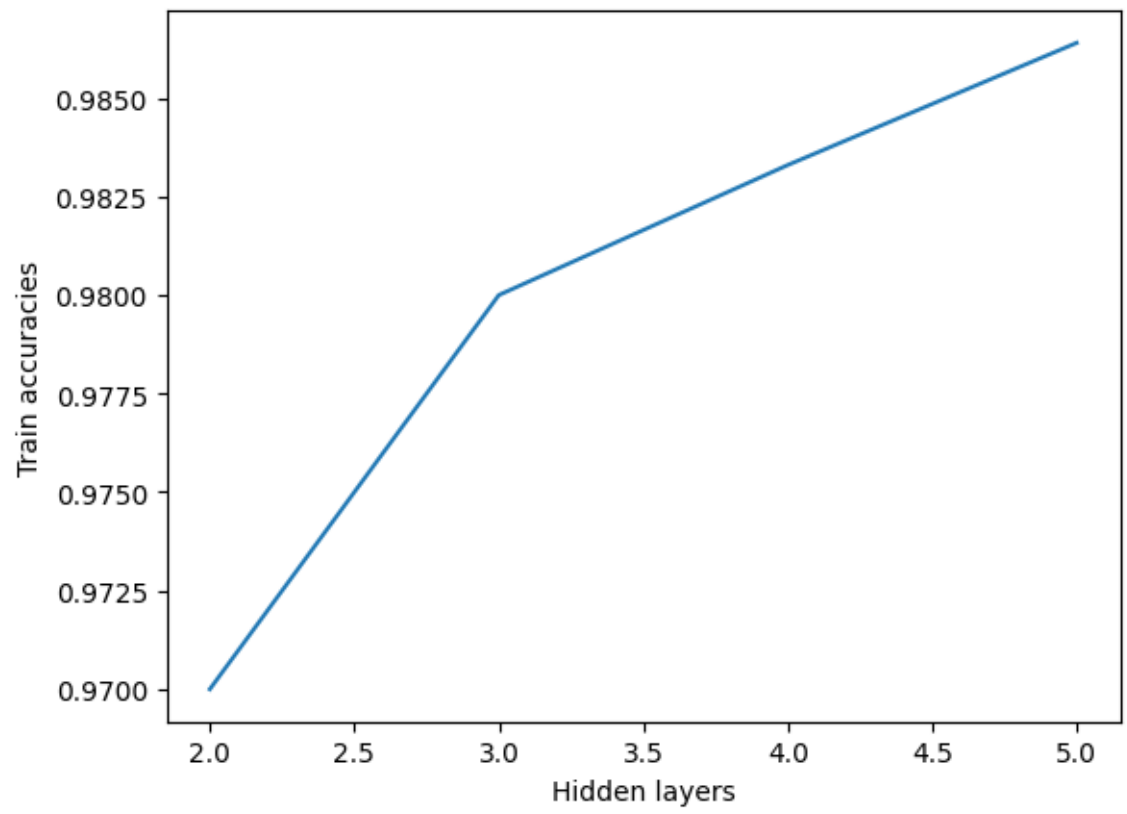
3. 4

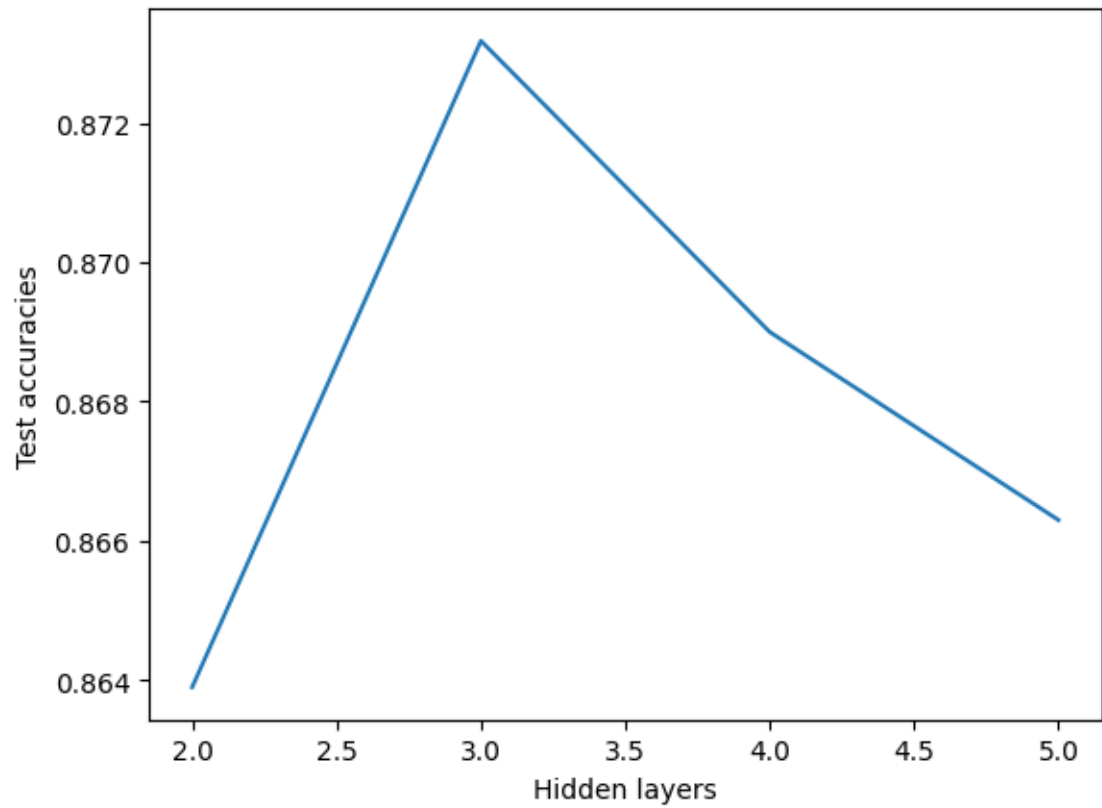
Accuracy of train is 0.9833163886064767
Accuracy of test is 0.8698869886988699

4. 5

Accuracy of train is 0.9864497741629027
Accuracy of test is 0.8663866386638663

Plot of accuracies in case of relu





For sigmoid as activation:

1. 2

Accuracy of train is 0.9746995783263055
Accuracy of test is 0.8651865186518651

2. 3

Accuracy of train is 0.9786663111051851
Accuracy of test is 0.8683858385838584

3. 4

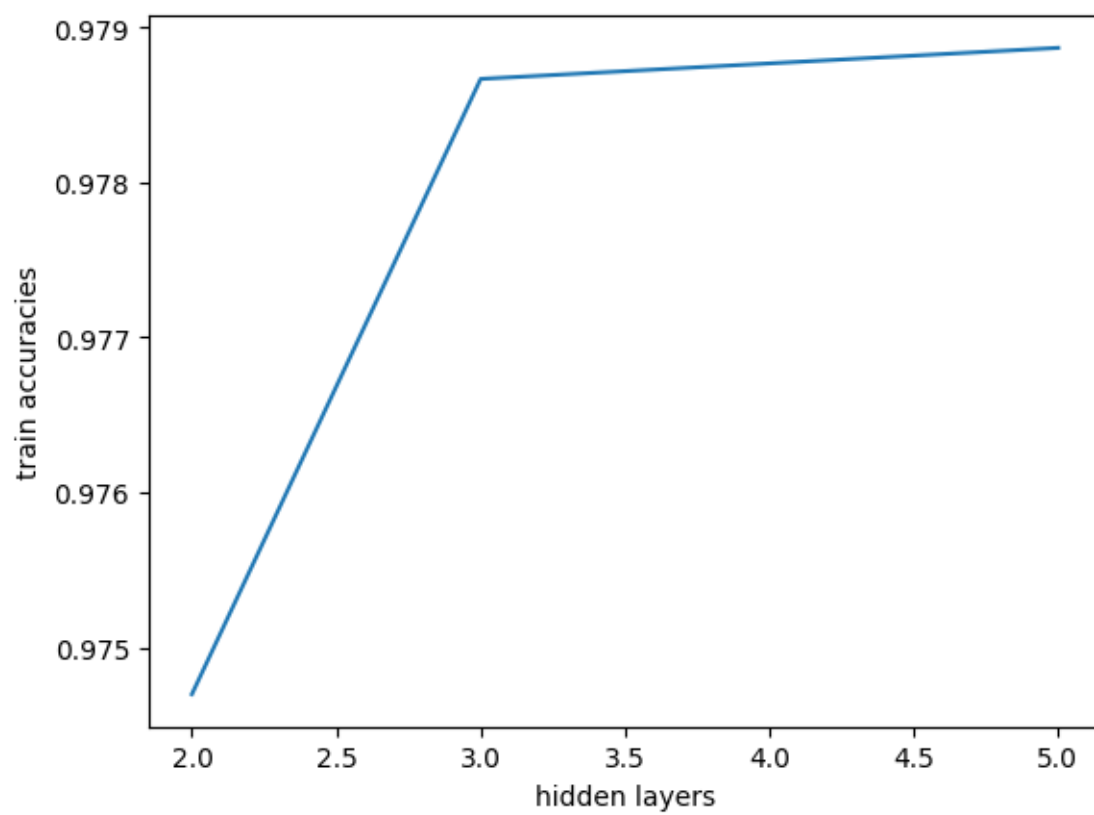
Accuracy of train is 0.9786663251066840
Accuracy of test is 0.8579167385933484

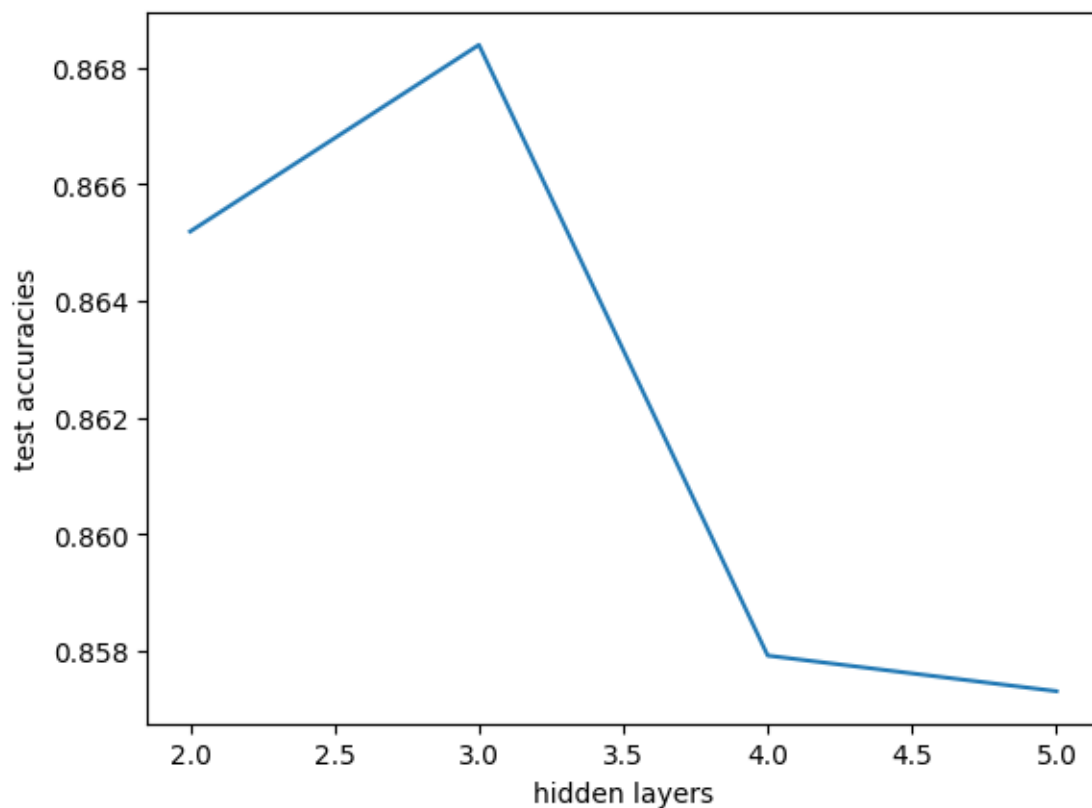
4. 5

Accuracy of train is 0.97866651910568510

Accuracy of test is 0.857306638678999

Plot of accuracies in case of sigmoid





We can see that the best architecture is when number of hidden layers = 3, since at this point we get 87% test accuracy (Wrt relu) and 86% test accuracy wrt sigmoid, hence this is the best.

2.6 f

Derivatives of last layer wrt bce:
 $(y_true/output) - ((1-y_true)/(1-output))$

Accuracy of train is 0.9999499991666527
 Accuracy of test is 0.8512851285128513

2.7 g

Training accuracy is 0.9642327372122869
 Test accuracy is 0.8331833183318332

Training using MLP does not really bring about much variations in accuracies, it prevents overfitting as we can see slight decrease in train accuracy compared

to part f. But test accuracy is also less 83% compared to 85% in case of bce in part f.