ASSIGNMENT 3

BUAN 6341: Applied Machine Learning

Madhav Sampath Gubbala (MSG170330)

Introduction:

The goal here is to implement Artificial neural networks (ANN) and K Nearest Neighbors. This report includes data analysis, model building and experimentation by tuning the parameters. Through the experimentation the best model is identified.

In this assignment we use 2 data sets, the description of the respective data sets is given below:

GPU Run Time:

This data set measures the running time of a matrix-matrix product A*B = C, where all matrices have size 2048 x 2048, using a parameterizable SGEMM GPU kernel with 241600 possible parameter combinations.
This dataset is used for implementing ANN and K Nearest Neighbors which is downloaded from UCI Machine Learning repository. The details about the dataset are mentioned below:

- Dataset consists of 241600 observations on 14 variables.
- Dependent variable is the average of Run1 (ms), Run2 (ms), Run3 (ms), Run4 (ms) columns which is 'avg_med'.
- Data split is done by 70:30 ratio for training and testing data set with no missing values.

Mobile Price Classification:

Link: https://www.kaggle.com/iabhishekofficial/mobile-price-classification
This data set measures the Price of the mobile phone with different features with 2500 possible parameter combinations. The price of data set is classified into two factors.
This dataset is used for implementing the ANN and K Nearest Neighbors algorithms is downloaded from Kaggle. The details about the dataset are mentioned below:

- Dataset consists of 2500 observations on 21 variables
- Dependent variable is the price of the mobile phone.
- Data split is done by 70:30 ratio for training and testing data set with no missing values.

Correlation: In SVM and Decision Tree algorithms correlation is important factor. For SVM, the effect is similar to that of multicollinearity in linear regression for linear kernel. For decision trees, no assumptions will be made by the algorithm on relation between features. If highly correlated information gain will be less.

	MWG	NWG	KWG	MDIMC	NDIMC	MDIMA	NDIMB	KWI	VWM	VWN	avg_med
MWG	1	-0.33	-0.26	-0.12	-0.095	0.14	-0.14	-0.0068	0.26	-0.14	-0.0072
NWG	-0.33	1	0.11	0.022	0.13	-0.036	0.19	0.0022	-0.085	0.38	-0.16
KWG	-0.26	0.11	1	0.093	0.24	-0.066	0.038	0.0018	-0.073	0.0093	0.14
MDIMC	-0.12	0.022	0.093	1	-0.16	0.15	0.026	-0.0023	-0.34	0.038	0.25
NDIMC	-0.095	0.13	0.24	-0.16	1	0.066	0.17	0.0012	0.0007	-0.12	-0.018
MDIMA	0.14	-0.036	-0.066	0.15	0.066	1	0.03	0.0013	-0.39	-0.033	0.063
NDIMB	-0.14	0.19	0.038	0.026	0.17	0.03	1	0.0015	-0.051	-0.17	-0.073
KWI	-0.0068	0.0022	0.0018	-0.0023	0.0012	0.0013	0.0015	1	-0.0025	0.0007	0.002
VWM	0.26	-0.085	-0.073	-0.34	0.0007	-0.39	-0.051	-0.0025	1	-0.041	-0.19
VWN	-0.14	0.38	0.0093	0.038	-0.12	-0.033	-0.17	0.0007	-0.041	1	-0.28
avg_med	-0.0072	-0.16	0.14	0.25	-0.018	0.063	-0.073	0.002	-0.19	-0.28	1

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	sc_h
battery_power	1	0.011	0.011	-0.042	0.033	0.016	-0.004	0.034	0.0018	-0.03	-0.03
blue	0.011	1	0.021	0.035	0.0036	0.013	0.041	0.004	-0.0086	0.036	-0.003
clock_speed	0.011	0.021	1	-0.0013	-0.0004	-0.043	0.0065	-0.014	0.012	-0.0057	-0.029
dual_sim	-0.042	0.035	-0.0013	1	-0.029	0.0032	-0.016	-0.022	-0.009	-0.025	-0.012
fc	0.033	0.0036	-0.0004	-0.029	1	-0.017	-0.029	-0.0018	0.024	-0.013	-0.011
four_g	0.016	0.013	-0.043	0.0032	-0.017	1	0.0087	-0.0018	-0.017	-0.03	0.027
int_memory	-0.004	0.041	0.0065	-0.016	-0.029	0.0087	1	0.0069	-0.034	-0.028	0.038
m_dep	0.034	0.004	-0.014	-0.022	-0.0018	-0.0018	0.0069	1	0.022	-0.0035	-0.025
mobile_wt	0.0018	-0.0086	0.012	-0.009	0.024	-0.017	-0.034	0.022	1	-0.019	-0.034
n_cores	-0.03	0.036	-0.0057	-0.025	-0.013	-0.03	-0.028	-0.0035	-0.019	1	-0.0003
sc_h	-0.03	-0.003	-0.029	-0.012	-0.011	0.027	0.038	-0.025	-0.034	-0.0003	1

Task 1:

Downloaded and used MLPClassifier package to classify classification problems. Experimented with number of layers, number of nodes, activation functions and may some other parameters.

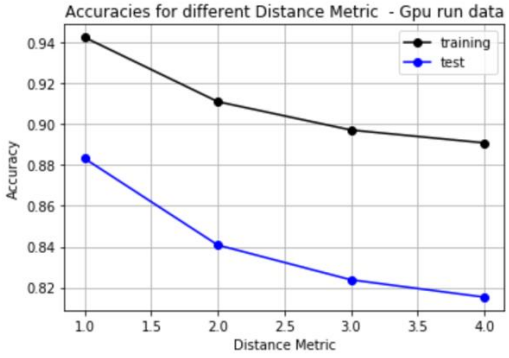
Task 2:

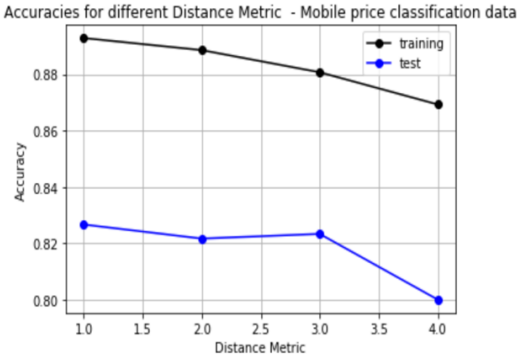
Downloaded and used KNeighborsClassifier to classification problems. Experiment with number of neighbors.

Experiment 1 - Experimented with different p values (Distance Metric):

Distance metric uses distance function which provides a relationship metric between elements in the dataset. If the distance is zero, then elements are equivalent else they are different from each other.

If p = 1(Manhattan Distance), p = 2(Euclidean Distance), p = ∞ (Chebyshev Distance).





GPU-Data

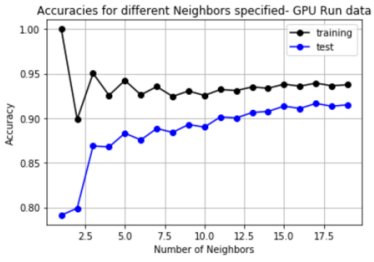
I had chosen 'p-value' to be '1' because it has high accuracy for both train and test data. Distance metric for **p='1'** is "Manhattan Distance method". I had taken the Manhattan Distance metric for the K-NN algorithm.

Mobile-Price Classification Data

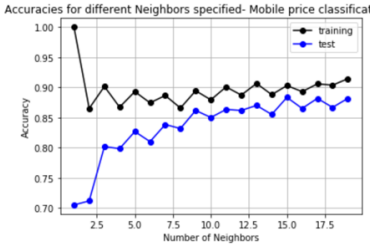
I had chosen 'p-value' to be '1' because of its high accuracy for both train and test data. Although p=3 also had high accuracy but slightly less than p=1.Distance metric for **p='1'** is "Manhattan Distance method". I had taken the Manhattan Distance metric for the K-NN algorithm.

Experiment 2 - Experimented with the different K values (n_neighbors):

'*n_neighbors* 'are the number of neighbors that will vote for the class of the target point. We choose n_neighbors to be odd to avoid draw. Default number chosen is '5'.




```
2 th Iteration with Acc_Score 0.7912
3 th Iteration with Acc_Score 0.799
4 th Iteration with Acc_Score 0.8689333333333333
5 th Iteration with Acc_Score 0.8677333333333334
6 th Iteration with Acc_Score 0.8831333333333333
7 th Iteration with Acc_Score 0.8756
8 th Iteration with Acc_Score 0.8884
9 th Iteration with Acc_Score 0.8840666666666667
10 th Iteration with Acc_Score 0.8928666666666667
11 th Iteration with Acc_Score 0.89
12 th Iteration with Acc_Score 0.9014
13 th Iteration with Acc_Score 0.9002666666666667
14 th Iteration with Acc_Score 0.9063333333333333
15 th Iteration with Acc_Score 0.9076
16 th Iteration with Acc_Score 0.9134666666666666
17 th Iteration with Acc_Score 0.9109333333333334
18 th Iteration with Acc_Score 0.9165333333333333
19 th Iteration with Acc_Score 0.9134
20 th Iteration with Acc_Score 0.9150666666666667
```




```
2 th Iteration with Acc_Score 0.705
3 th Iteration with Acc_Score 0.7116666666666667
4 th Iteration with Acc_Score 0.8016666666666666
5 th Iteration with Acc_Score 0.7983333333333333
6 th Iteration with Acc_Score 0.8266666666666667
7 th Iteration with Acc_Score 0.81
8 th Iteration with Acc_Score 0.8383333333333334
9 th Iteration with Acc_Score 0.8316666666666667
10 th Iteration with Acc_Score 0.8616666666666667
11 th Iteration with Acc_Score 0.85
12 th Iteration with Acc_Score 0.8633333333333333
13 th Iteration with Acc_Score 0.8616666666666667
14 th Iteration with Acc_Score 0.87
15 th Iteration with Acc_Score 0.855
16 th Iteration with Acc_Score 0.8833333333333333
17 th Iteration with Acc_Score 0.865
18 th Iteration with Acc_Score 0.8816666666666667
19 th Iteration with Acc_Score 0.8666666666666667
20 th Iteration with Acc_Score 0.8816666666666667
```

GPU-Data

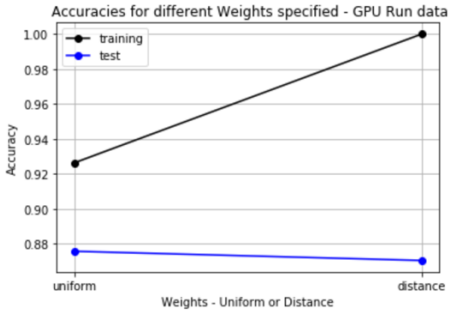
*n_neighbors=1 had max accuracy but overfits dataset as test accuracy is very low. As n_neighbors value increases the data set looks balanced with proper bias and variance. I had chosen **n_neighbors=18** as it has the highest accuracy(91.65%).*

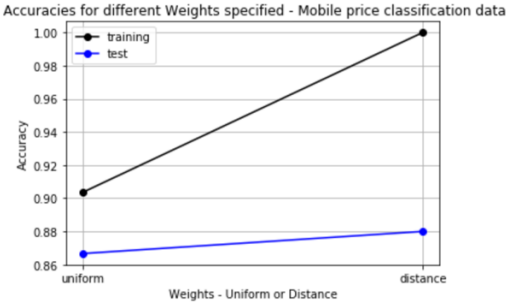
Mobile-Price Classification Data

n_neighbors=1 had max accuracy but overfits dataset as test accuracy is very low. As n_neighbors value increases the data set looks balanced with proper bias and variance. I had chosen ***n_neighbors=16*** *as it has the highest accuracy(86.13%).*

Experiment 3 – Experimented with changing "weight" parameter (weights):

'weight *'parameter has '*uniform*',* 'distance *'and 'callable'. For '*uniform *'weight, each of the k-neighbors has equal vote whatever its distance from the target point and for '*distance *'weight voting weightage varies by inverse of distance. Points which are nearest to the target point have greater influence than those which are far. For 'callable' weight a user can define function which accepts an array of distances and returns an array of the same shape.*





GPU-Data:

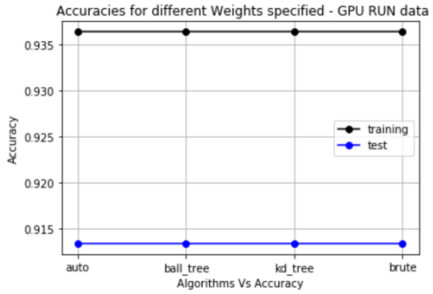
We could see that UNIFORM had performed better when compared with distance with less error difference between the training and test set. If the number of neighbors had been high, then the DISTANCE would have least error differences between training and test set. From this, we can conclude that **UNIFORM** performs better with good accuracy and also least error difference.

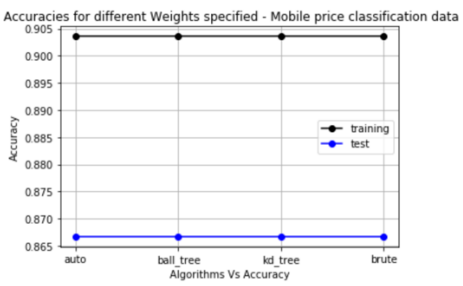
Mobile-Price Classification Data:

We could see that UNIFORM had performed better when compared with distance with less error difference between the training and test set. If the number of neighbors had been high, then the DISTANCE would have least error differences between training and test set. From this, we can conclude that **UNIFORM** performs better with good accuracy and also least error difference.

Experiment 4 – Experimented with various "algorithms" (algorithm):

Algorithms has 4 different types such as 'auto', 'kd_tree', 'ball_tree', 'brute'. For this, I have made the n_neighbors value to be "18" for GPU run time and "16" for Mobile price classification, distance metric p to be "1". The results are plotted below with accuracies for the various algorithms. There is no much variation in the accuracies for both the data's.





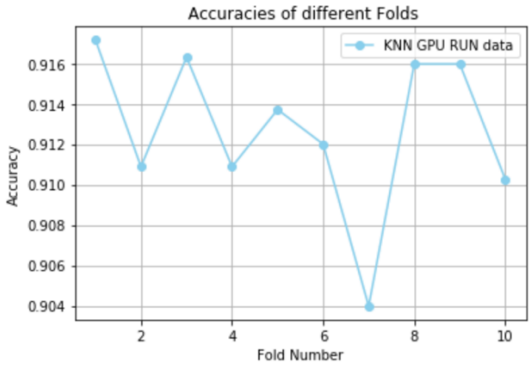
Final Algorithm GPU Run Data:

Using the identified values for the different parameters the final model was created for the GPU Run dataset using K-NN algorithm. The values used for building the model is p=1 , weight = 'uniform' and n_neighbors = 18 .
The Accuracy score and confusion matrix of the models is mentioned below.

```
Accuracy score is 0.9134
[[7068  446]
 [ 853 6633]]
```

The accuracy obtained from cross validation's mean accuracy is given below. They are almost same. The accuracies of K-Fold cross validation and accuracy of test set is almost the same, which means that the training data was able to capture most of the variation of the entire dataset and hence model is more generalized.

```
K-fold accuracy score is  0.9127140269504161
```



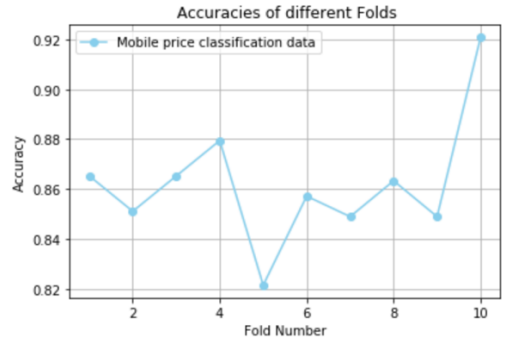
Final Algorithm Mobile Price Classifiaction:

Using the identified values for the different parameters the final model was created for the Mobile price Classification dataset using K-NN algorithm. The values used for building the model is p=1 , weight = 'uniform' and n_neighbors = 16 .
The Accuracy score and confusion matrix of the models is mentioned below.

```
Accuracy score is 0.865
[[265  21]
 [ 60 254]]
```

The accuracy obtained from cross validation's mean accuracy is given below. They are almost same. The accuracies of K-Fold cross validation and accuracy of test set is almost the same, which means that the training data was able to capture most of the variation of the entire dataset and hence model is more generalized.

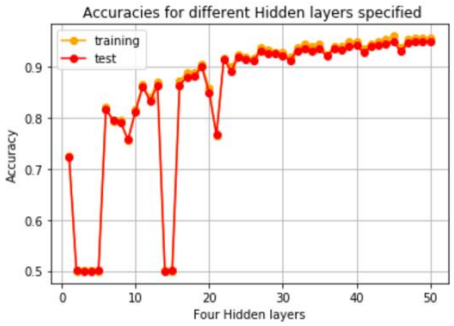
```
K-fold accuracy score is  0.8621578724862056
```

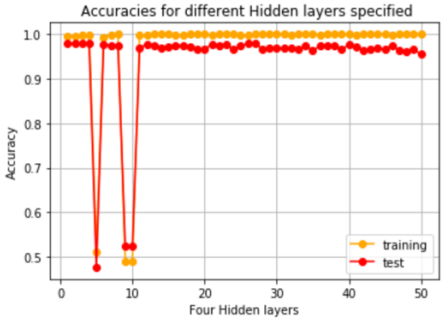


Artificial Neural Network(ANN)

Experiment 1 -Experimented with Number of neurons and hidden layer:

Experimented with different hidden layers and number of neurons in each hidden layer's. As we could see and analyze from the below graphs that with the increase in the hidden layer count and neurons in a particular layer, we yield a good accuracy . But as we keep on increasing the number of hidden layers , the complexity of the model will increase and this returns to overfitting of model . Also, the computational time and cost is also gets affected.If data set is huge then we will restrict number of hidden layers and number of neurons.



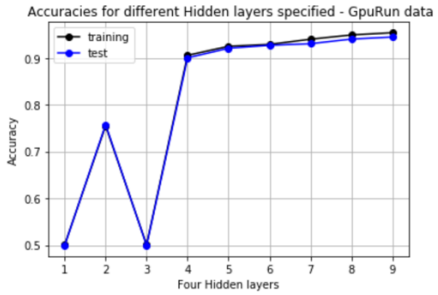


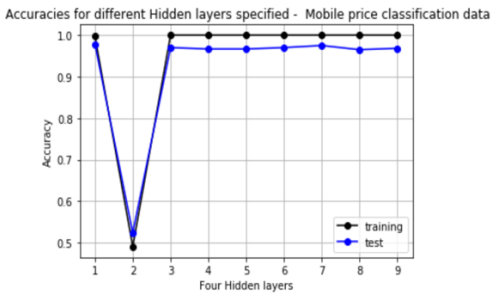
GPU and Mobile-Price Classification Data:

From the above results I had considered that **4-layer** neural network with 4 neurons in each to be a precise model fpr both of the data sets and which doesn't overfit the data and also produces good accuracy and least error between training and test set.

Experiment 2 -Experimented with Number of neurons in hidden layer(hidden_layer_sizes):

Here I experimented on the number of neurons by considering fixed hidden layer's. From the previous experiments 1,i came to the conclusion that 4 hidden layers yield to the better results.





GPU-Data:

From the above graph, for 4,5 and 6 hidden layers there is more accuracy and least error between the training set and test set.Due to complexity issues and easy compilation I had chosen **5** as my count of neurons in each hidden layer.

Mobile-Price Classification Data:

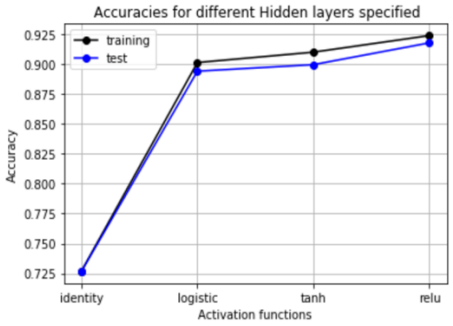
From the above graph, Except 2 all the other hidden layers had almost accuracy.While considering the least error between the training set and test set,I had chosen 7 as my count of neurons in each hidden layer considering complexity issues and easy compilation.

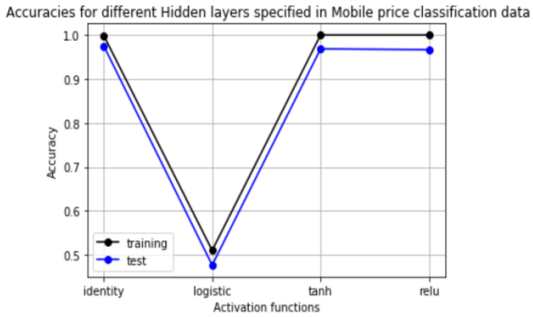
Experiment 3 -Experimented with different Activation layers(activation):

The main purpose of Activation layer is to convert a input signal of a node in a A-NN to an output signal. *That output signal now is used as a input in the next layer in the stack. A Neural Network without Activation function would simply be a* ***Linear regression Model,*** *which has limited power. There are majorly 4 activation layers which are 'identity', 'logistic', 'tanh', 'relu'. Depending on the type of function used activation function is chosen.*

GPU-Data:

From the below graph ***'relu'*** *activation function had highest accuracy and* least error between the training set and test set.I had chosen "**relu**" as my activation function.





Mobile-Price Classification Data:

*From the above graph **'relu','tanh'** and **'identify'** activation function had highest accuracy and* least error between the training set and test set. Although all had similar accuracy I had chosen "**relu**" as my activation function considering other factors.

Experiment 4 -Experimented with different Solver types(solver):

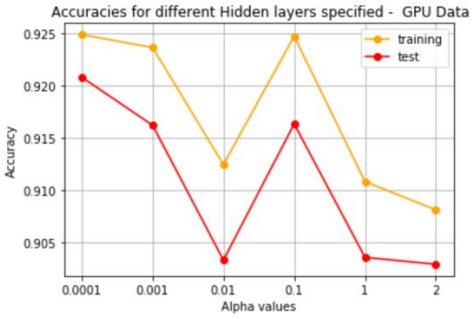
Solver parameter specifies the algorithm for weight optimization across the nodes. We have 3 solver types namely 'lbfgs', 'sgd', 'adam' .

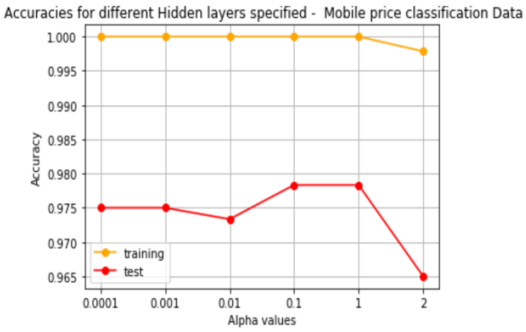
GPU and Mobile-Price Classification Data:

From the above plot, we could observe that the "**Adam**" solver has good accuracy and also the least error between the training set and test set for both the data sets. So, by tradeoff we will be choosing the "Adam" solver for our final ANN model build.

Experiment 5 – Experimented with different values of penalty regularization term(Alpha):

Alpha is a parameter for regularization term which combats overfitting by constraining the size of the weights. Increasing alpha may fix high variance which is a sign of overfitting.



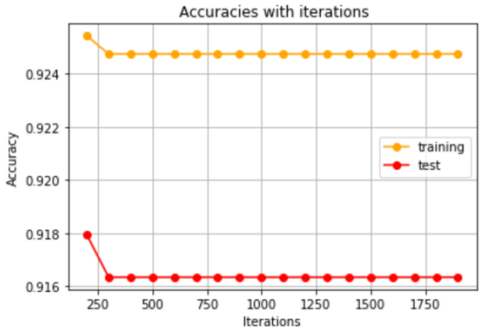


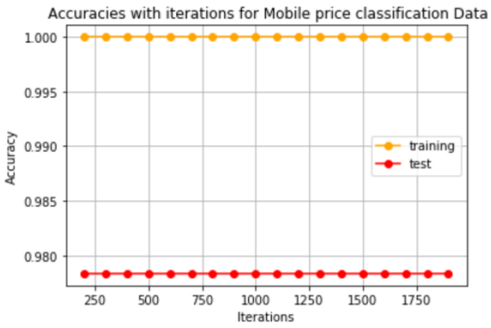
GPU-Data:

Alpha value of 0.0001 and 0.1 had more accuracy but 0.0001 had less error between train set and test set.So I had chosen alpha value to be **0.0001.**

Mobile-Price Classification Data:
Alpha value of 0.0001 and 0.1 had more accuracy for train data but 0.1 had less error between train set and test set. So I had chosen alpha value to be **0.1.**

Experiment 6 – Experimented with different number of iterations(max_iter):





GPU-Data:
From the above graph after the certain threshold even though if we increase the number of iterations there is no much difference. In order to get space for solver to converging ,I had chosen the no of iterations as **500**.

Mobile-Price Classification Data:
From the above graph there is no much difference between accuracy with the increase in the number of iterations.there is no much difference. In order to get space for solver to converging ,I had chosen the no of iterations to be **500**.

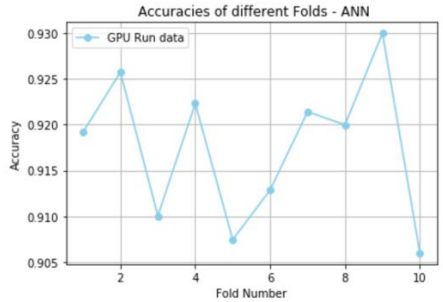
Final Algorithm for GPU Run data:

With the identified number of hidden layers (4 layers), number of neurons (5 in each hidden layer) , activation function(relu) , solver methods and alpha values(0.0001), we will be creating our final model using MLPClassifier . The confusion matrix for the same is also given below:

```
Accuracy is 0.9208
[[6940  574]
 [ 614 6872]]
```

The Mean accuracy is found from the 10 folds of the training set. We take cross validation when we have very few data points and also to make sure that the variance of the entire dataset is captured . If we spilt the dataset by regular methods, our training set may not totally capture the variation explained by the full dataset. So we do cross validation and by doing this ,we will find how accuracy changes for each fold that is being trained. The accuracies of the different Folds are represented below. They seem to change because of variation in data based on individual folds.

```
0.9174857171801751
```



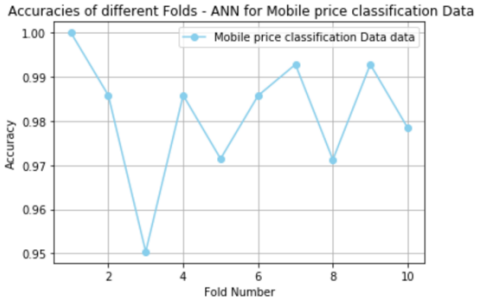
Final Algorithm for Mobile price classification:

With the identified number of hidden layers (4 layers), number of neurons (7 in each hidden layer) , activation function(relu) , solver methods and alpha values(0.1), we will be creating our final model using MLPClassifier . The confusion matrix for the same is also given below:

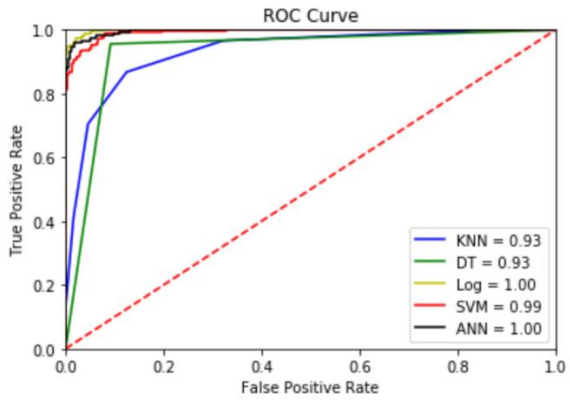
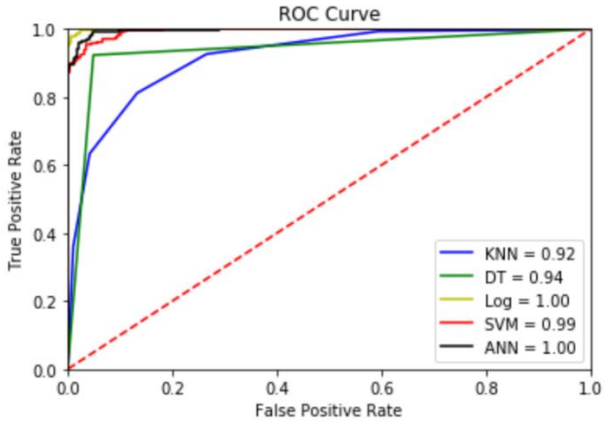
```
Accuracy is 0.9783333333333334
[[281   5]
 [  8 306]]
```

The Mean accuracy is found from the 10 folds of the training set. We take cross validation when we have very few data points and also to make sure that the variance of the entire dataset is captured .

```
Mobile price classification accuracy is 0.9814380471306846
```



Model Comparision with AUC:

GPU Run-time Data	Mobile Price Classification Data
	
❖ The best model is decided based on the lower false positive rate or true negative rate etc. ❖ For the sake of experimentation, the considered lower false positive rate as the best model ❖ From the above plot, we can observe that ANN is the best model with AUC of 1	❖ The best model for mobile price classification data is ANN with again AUC of 1.