Name: Het Piyush Sheth **GTID:** 903393644

CS 7641-HW1

Introduction:

In the homework, I worked with five different Supervised Learning algorithms and analyzed their training and testing errors, performance with respect to time, number of iterations, learning rates, etc. In order to improve the quality of my analysis, I have included results obtained by running the models on two different datasets. I have implemented all the algorithms using the scikit-learn package in Python.

Describing the Classification Problem (Dataset description):

The two datasets that I have used for this homework are:

- Adult Income Data
- Wisconsin Breast Cancer Data

I downloaded the first dataset from the UCI Machine Learning Repository. It contains 14 attributes with 30,000 instances (approx) and using these attributes, the aim is to classify whether a person's income is below or above $50,000 per annum. What I find particularly interesting about this dataset is that it has a balanced mix of categorical and numeric attributes and that none of the attributes require domain specific knowledge. This very factor, although trivial for many, has actually helped improve the quality of my analysis as I already have an understanding of how the attributes could affect the classification results.

As for the second dataset, it contains 10 numeric attributes with 700 instances (approx.) that are used to classify if a cancerous cell is malignant or benign. What's interesting about this dataset is that since all the attributes are different measures of the same cancerous cell, there would be a lot of correlation between them and thus this should reflect in the results. At the same time, positive correlation amongst the various attributes would also help strengthen the binary classification model.

Evaluation strategy:

- For testing the model, I have split the data into 75% training and 25% testing. I have deviated from the standard norm of 70/30 to improve my analysis of the models.
- For cross-validation, I have used the K-fold cross-validation method with 5 folds, which I implemented on the training set. Thus, I split the training data further into 80% learning and 20% validation.
- I have used the accuracy as a measure to evaluate the performance rather than RMSE as the datasets I am trying to solve a classification problem and accuracy is a better measure for that.

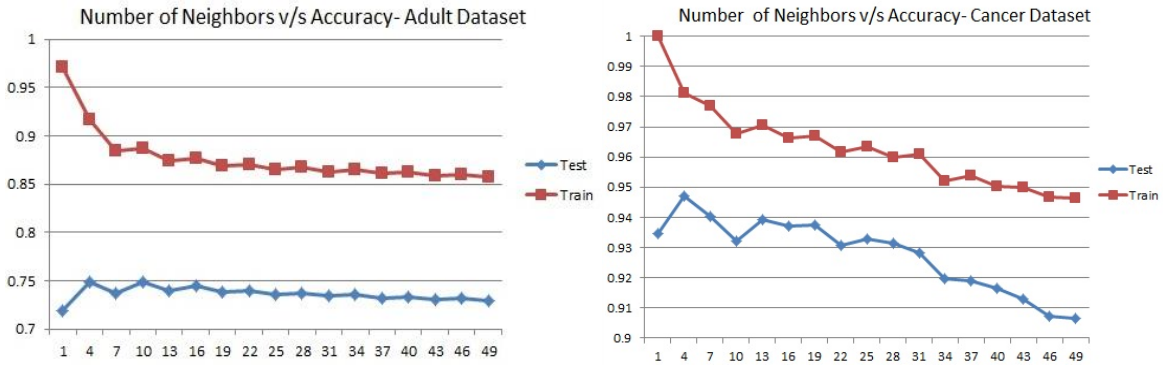
K- Nearest Neighbors:

KNN is one of the simplest Machine Learning algorithms. It is an instance based learning algorithm that predicts a class based on the k-closest training instances in the feature space. The major parameters that can affect the result of the KNN algorithm are:

- K: The number of neighbors that the algorithm uses as a threshold for classification.
- Distance metric: The nearest neighbors are estimated using the distance between a given data point and training points. This parameter specifies how to measure the distance. (I used: Manhattan, Euclidean, Chebyshev)

- Weights: This specifies whether the training points are to be considered equal or should higher weights be assigned to closer points while estimating the class for a new data point. (I used: Uniform, Distance)

I wanted to see how the performance differed for the two datasets based on the number of neighbors chosen. The results were as follows:



- One thing common amongst the training samples for both the datasets is that it starts with almost 100% accuracy and decreases as we increase the number of neighbors. This is because k=1 means that each sample is using itself as a reference for classification. Thus, this would count as an overfitting case.
- Second observation is for the trend in the Test accuracy of the two datasets. For the adult dataset, the accuracy seems to increase as we increase the number of neighbors and stays constant for a while after that before slightly decreasing. Whereas, for the Cancer dataset, it steeply decreases as we increase the number of neighbors. This variation is because of the difference in the size of the datasets. The adult dataset has around 30,000 instances and thus several training points in the plane to calculate the distance for higher k values, whereas for the cancer dataset with only 700 instances, if we keep a high k then it will end up accepting the training points that belong to the other class as well and thus misclassify several results.

On analyzing the performance with respect to the other parameters, I got the following results:

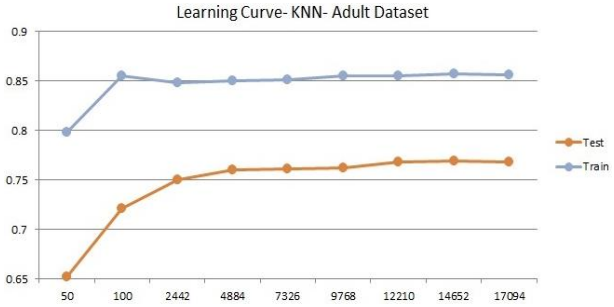
Adult			Cancer		
Weights	Train	Test	Weights	Train	Test
Distance	0.979676	0.734124	Distance	1.000000	0.937108
Uniform	0.774379	0.738252	Uniform	0.926506	0.918532

- We can see that both the training and testing accuracy is higher for a weighted model for both the datasets. It is because the data points are sparsely located for both the data sets and giving them equal importance (as in uniform measure) would not make sense.

Adult			Cancer		
Metric	Train	Test	Metric	Train	Test
Chebyshev	0.870262	0.723264	Chebyshev	0.947896	0.905607
Euclidean	0.880810	0.746744	Euclidean	0.972538	0.939132
Manhattan	0.880011	0.743556	Manhattan	0.971325	0.938722

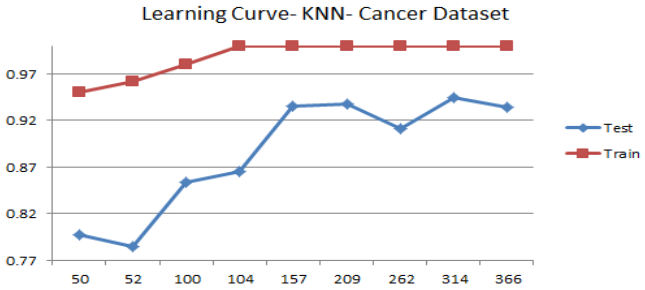
- Euclidean and Manhattan distance give a similar accuracy while Chebyshev is slightly lower. This is because of the way the distances are calculated. While the first two are quite similar to the normal distance measurement in a coordinate plane, the last one uses a vector plane. This indicates that for the given dataset, measuring the distance over a coordinate plane is more beneficial.

Performance on Adult Dataset:



- As we increase the size of the dataset, initially both the train and test accuracies steeply increase and then they stabilize thereafter. The increment is only for the initial period because the dataset isn't that wide (number of attributes=14) and when the length (number of tuples) of the dataset balances out the width, then the accuracy starts stabilizing.
- Another factor that has led to the early stabilization is that the above problem is that of a binary classification. Thus, the model can effectively learn the attributes with lesser training size. If the number of classes increased, then the model would require more training samples to stabilize.

Performance on Cancer Dataset:



- Here, the accuracy for the training curve is near 100%.This is obvious considering it is being trained on an instance based learning method.
- Whereas, for the testing accuracy, the climb is very steep throughout because it is a small dataset and thus for every tens of training samples, the model obtains a large amount of information (this is relatively large and significant because of the overall size of the dataset).

Comparing the performance of the two Datasets:

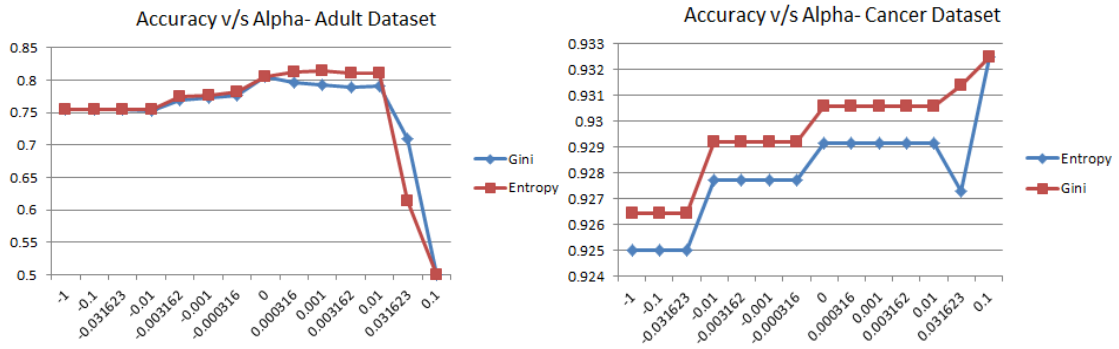
- The difference between the numbers of attributes of the two datasets isn't that much (1.4 times) and the difference between the instances is huge (35 times). What this implies is that the learning curve of the test set should be the same for both the datasets for the first few absolute number of training samples (as the attributes are similar in number).
- This is observed in the above graphs as the cancer dataset's curve is steep for all of its training samples and so is the adult dataset (first 500 samples).

Decision Tree:

A decision tree is a Machine Learning algorithm that uses a tree like structure to learn from the data. It tries to approximate a sine curve by using if-else conditions. The deeper the tree, the fitter the network. The two major parameters that can affect the outcome of a Decision Tree algorithm are:

- Alpha: The value of alpha determines the pruning level of a Decision tree. Pruning is done in order to reduce the complexity of the tree.
- Criterion: It determines the quality of the split, i.e., it tells us what the split should be based on. The two values I have used are (Gini and Entropy)

Below are the results obtained based when I compared the alpha values with the accuracy of gini and entropy measure respectively. Here, I have used the test accuracy for each.



- Based on the first graph, it is obvious that the accuracy decreases when we increase the value of alpha. What this, means is that when we prune the tree further, it tends to misclassify. Thus, from the graph, the best value of alpha is around 0.001. Also, notice that initially there is an increase in the accuracy, before the decrease. This is because, some pruning is beneficial, but pruning excessively can damage the model.
- On the other hand, a similar trend is not noticed for the cancer dataset. For the cancer dataset, the accuracy seems to be increasing rather than decreasing. This is because of the fact that most of the attributes of the cancer dataset have a high positive correlation. This is because they are different measures of the same cancerous cell. Thus, what is means in the context of a decision tree is that extra pruning would not affect the performance of the model when learnt on this dataset.

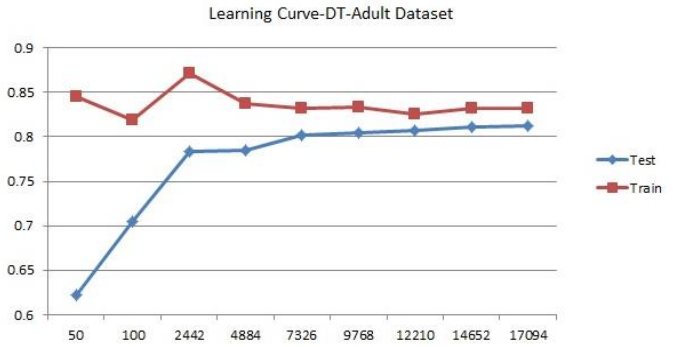
The second factor that I analyzed is the split criterion. The results are as follows:

Adult			Cancer		
Criterion	Train	Test	Criterion	Train	Test
Entropy	0.833513	0.751366	Entropy	0.950494	0.929945
Gini	0.823313	0.751277	Gini	0.949370	0.929484

- For both datasets, the Gini and Entropy criterion have a similar result. This is not out of the blue because they both are supposed to perform in a similar fashion. The major difference is that Entropy uses a logarithmic function in its calculation.
- Using a log makes entropy computationally extensive and I observed this by analyzing the time it took for both the criterion to run.

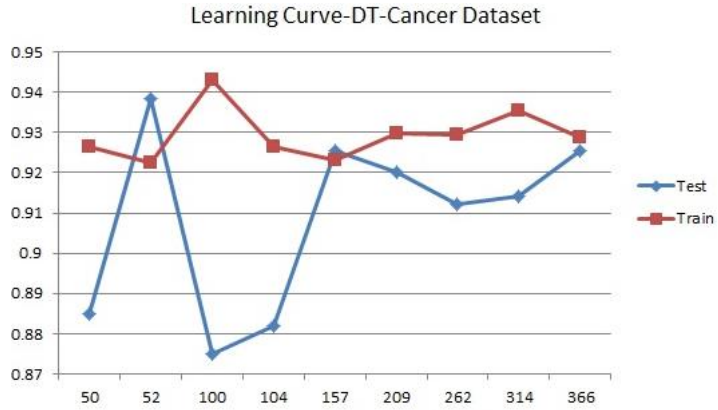
Adult			Cancer		
Criterion	Mean fit time	Mean score time	Criterion	Mean fit time	Mean score time
Entropy	7.135248	0.048795	Entropy	0.093464	0.013189
Gini	6.130830	0.015120	Gini	0.072547	0.006594

Performance on Adult Dataset:



- What we can observe from the above graph is that for the test data, there is a significant increase in the accuracy early on and then it just stays the same. The reason this happens is because Decision Trees improve by making the tree structure complex. Now, the complexity solely depends on the number of attributes because they only serve as the nodes within the trees. Thus, the initial steep is because the tree is incorporating the information of the various attributes. Once it has that, there is only little difference in the curve. Thus, we can say that only 3000 instances are required to learn all the attributes.

Performance on Cancer Dataset:



- In the above graph, we can see that there is a lot of fluctuation in accuracies. It is because of the tree structure of the graph. There are only finite nodes in the tree for finite attributes and if some of the training instances have a strong bias towards one direction then it may change the orientation of a particular node. Thus, eventually towards the end of the graph, when a majority of the data is trained, the accuracies become stable.

Comparing the Performance of the two Datasets:

- Although the LC curves of both the datasets look different, they both are doing the same thing in the sense that initially, while the curve of the cancer dataset fluctuates, the adult dataset does too. It's just that in context of the large training size it has and thus the length of the axis, it doesn't look very evident. Eventually, both the curves stabilize when a majority of the training set is used.

Support Vector Machines (SVM):

SVM is a supervised Machine Learning algorithm that given labeled training data; it creates a hyperplane separating the classes and categorizes the new data. For a two dimensional space, this hyperplane would be a line.

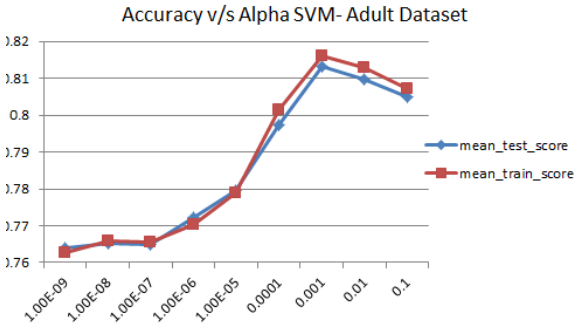
But what if the data cannot be separated linearly? In such a case a different kind of hyperplane can be created. What If the data points when plotted on a two dimensional plane take the form of concentric circles? In such a case, a radial SVM can be used. The different kinds of hyperplanes can be specified using the kernel parameter in scikit-learn.

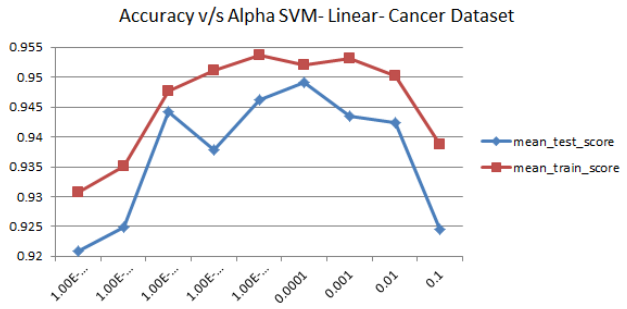
The various parameters used in my implementation are:

- Kernel: Described above (Linear, RBF)
- Alpha: It is the constant that is multiplied to the regularization term.
- Gamma (Kernel Coefficient): It defines how far the influence of a single training example reaches. (This is used only for the RBF Kernel in my implementation.)

1) Linear SVM:

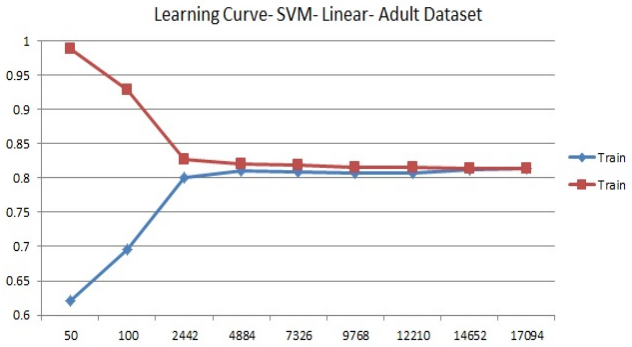
In Linear SVM, I have used only one parameter, alpha. The trend of the accuracy v/s alpha is as follows:





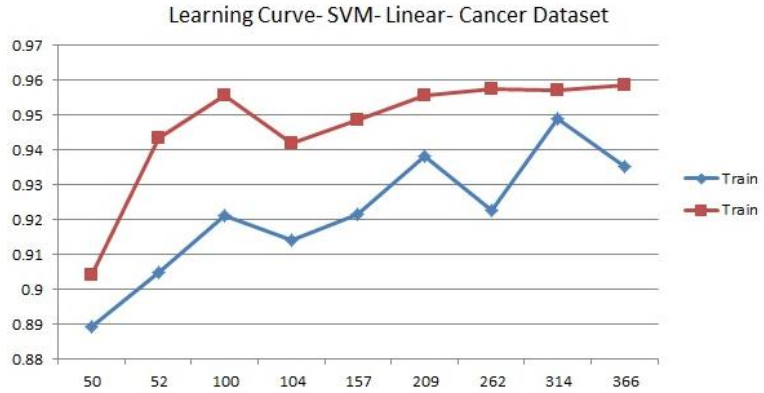
- As we know, alpha is a multiplier to the regularization constant. And a higher regularization constant means that the model avoids misclassifying. Thus we can say that a higher alpha can lead to overfitting.
- In the above graphs, initially as we increase alpha, the hyperplane modifies accordingly and at the highest point, overfitting occurs.
- Now, the dip at the end in both the graphs is because of the fact that the hyperplane is of a very small margin and thus it leads to the other class training point falling in the region of the current class. (Due to the small margin.)

Performance on Adult Dataset for Linear SVM:



- SVM classifies data by making a hyperplane between them. Thus, initially, there is a very high training accuracy because there are very few data points and they can be easily classified by a hyperplane. But this only particular to the training set and is not efficient otherwise. Thus, the test accuracy is very low initially.
- The merge towards the end of the train and test set just goes to show that once the hyperplane has seen all the possibilities, then SVM is a very good binary classifier.

Performance on Cancer Dataset for Linear SVM:



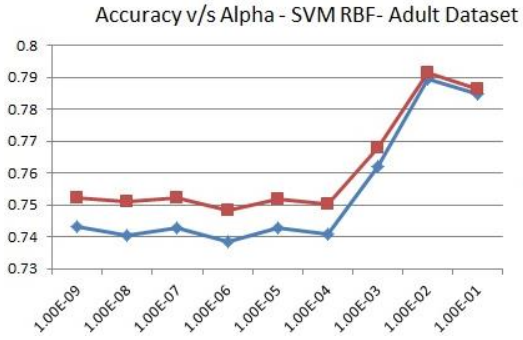
- Here, both the train and test are gradually increasing with an increase in the number of training samples. This is because the hyperplane fails to classify the data initially and the reason for that being the high correlation amongst the carious attributes. Eventually, the hyperplane learns the attributes and the accuracy increases.

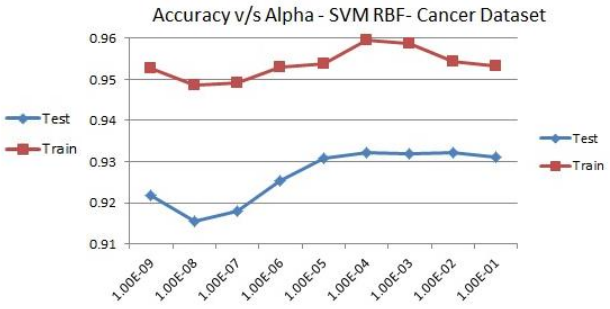
Comparing the Performance of the two Datasets for Linear SVM:

- The difference in the performance of the model on two datasets is a result of the difference in the ratio of instances to attributes in both the datasets. While for adult, the ratio is very high and for cancer, it is low.

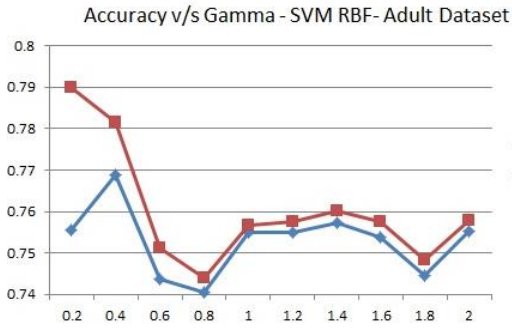
2) RBF SVM:

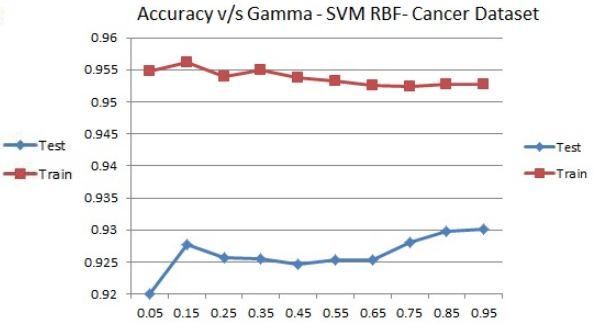
For the RBF SVM, I have used two different parameters, alpha and gamma.





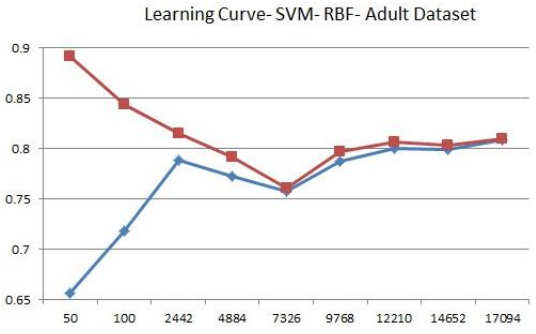
- The trend followed by the alpha values in both the datasets is the same as that for the Linear SVM. This is because changing the kernel doesn't affect the function that alpha performs in SVM

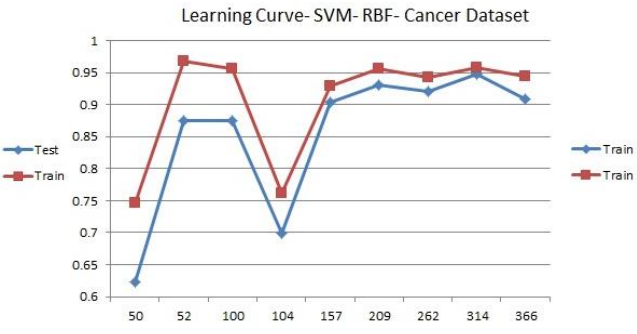




- The dip in the accuracy for the adult dataset with an increase in gamma is because now a faraway training sample from the hyperplane also affects the hyperplane structure. This is quite prominent in the adult dataset and not so in the cancer dataset because of the number of training samples in the adult dataset v/s cancer dataset. More samples mean a greater influence.

Performance on Adult and Cancer Dataset for RBF SVM:





- For the adult dataset, the graph is similar to that of Linear SVM and it indicates that the position of the training points is such that a linear as well as a radial hyperplane can separate it with the same accuracy.
- As for the curve in the Cancer dataset, there is very little difference between the training and test results, indicating that the radial hyperplane is better suited for the cancer dataset.

Comparison between Linear and RBF SVM:

- The type of kernel doesn't seem to affect the trend with the alpha values.
- For the adult dataset, Linear and RBF SVM have a very similar performance. This indicates that the training points are oriented in such a way that it can be split using a linear and a radial hyperplane. It is also because of the fact that there are way too many training points per attribute.
- As for the cancer dataset, based on the graphs, the RBF model would be better suited for a greater amount of data but if we had lesser data, then the linear model performs better.

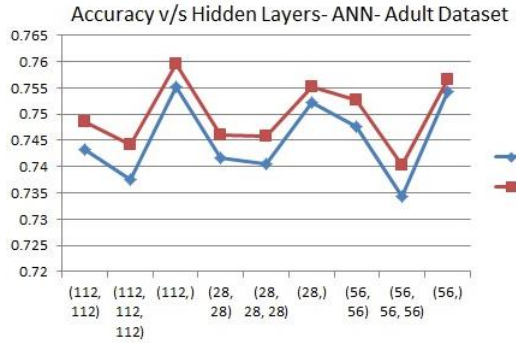
Artificial Neural Networks:

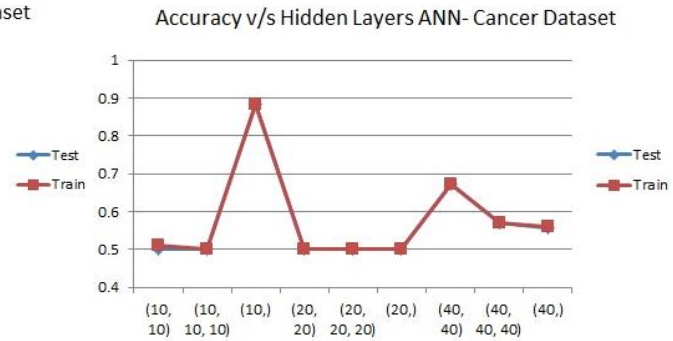
Artificial Neural Networks is a learning algorithm that mimics the functioning of the human brain. It has a layer architecture where neurons from one layer are connected to all neurons in the next layer. For the purposes of this assignment, I have used a Multi-Layer Perceptron (MLP). The parameters that I have tuned are:

- Activation function
- Hidden layer size
- Alpha (Regularization constant)

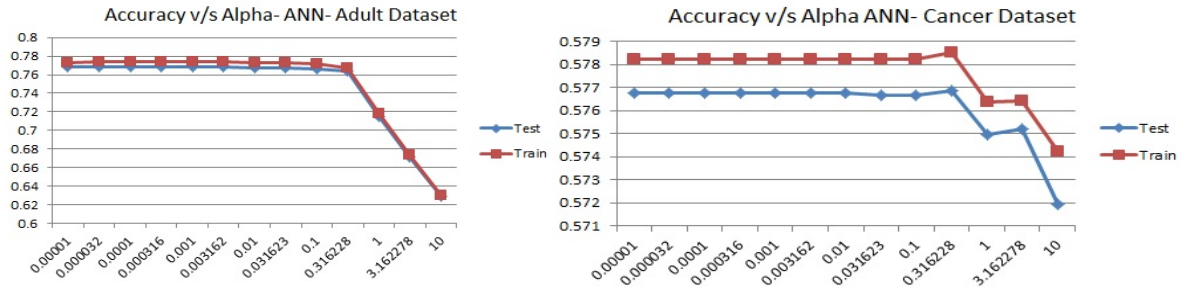
Adult			Cancer		
Activation	Train	Test	Activation	Train	Test
ReLu	0.618678	0.615441	ReLu	0.774341	0.766423
Logistic	0.537607	0.537821	Logistic	0.725384	0.723956

- From the above table, we can say that the model performs better with ReLu activation function. This is because ReLu handles sparsity in the data better.



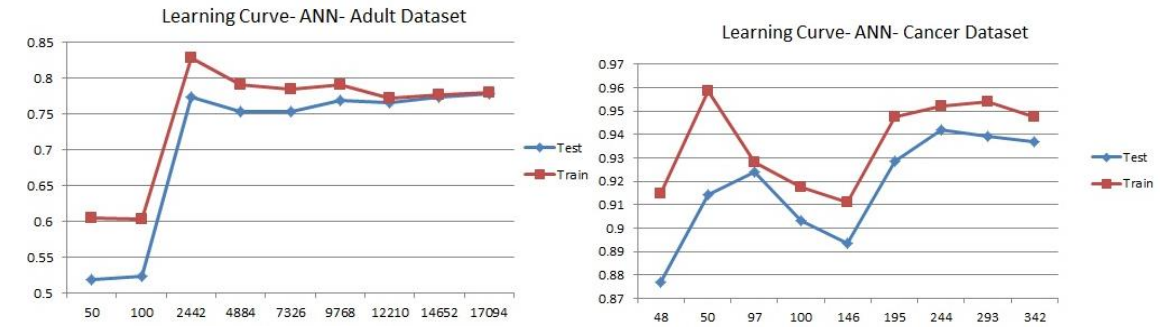


- From the above graph, we can say that the accuracy reduces as we reduce the number of units in the hidden layers while the number of hidden layers does not seem to affect the accuracy.
- A similar trend is seen for the graph of the cancer dataset but the training and testing accuracies are similar. And this is due to the lower number of training size available.



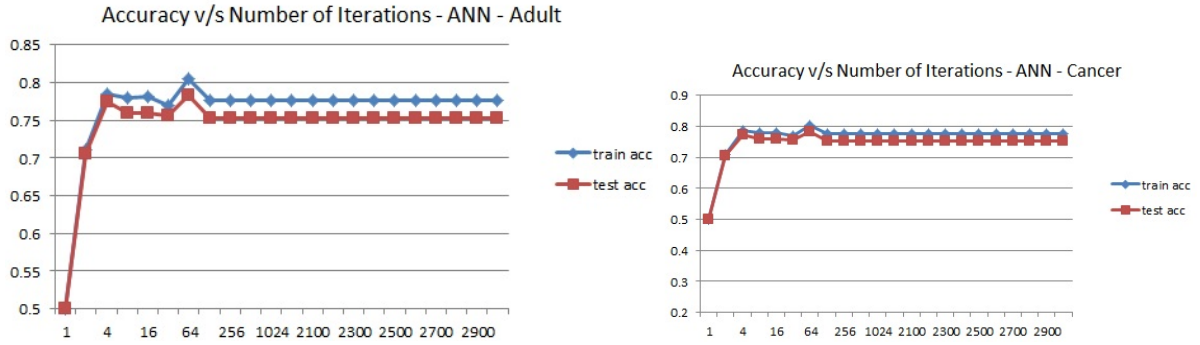
- We observe that the accuracy remains stable for a wide range of alpha values and steeply decreases after a certain threshold. This is due to the way the learning rate is calculated after we increase alpha to a certain threshold and the negative component of the learning rate increases, thus decrease in the accuracy.

Performance on Adult and Cancer datasets of ANN:



- We can see that for the Adult dataset, the learning rate increases steeply and remains constant after that whereas it wavers for the Cancer dataset.
- This is because the weights of the hidden layers stabilize after seeing a greater amount of training samples and thus this explains the increment in the Adult dataset and since the training size of the Cancer dataset is small, thus the learning rate does not stabilize in the given data size.

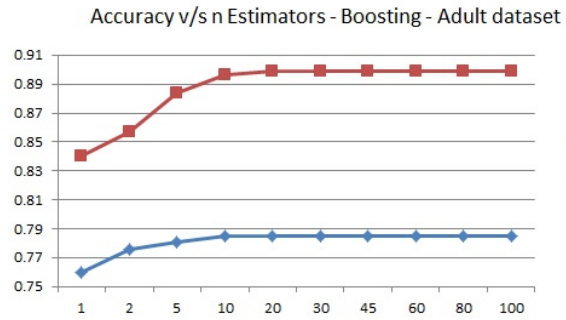
Variation of the ANN accuracy with respect to the number of iterations:

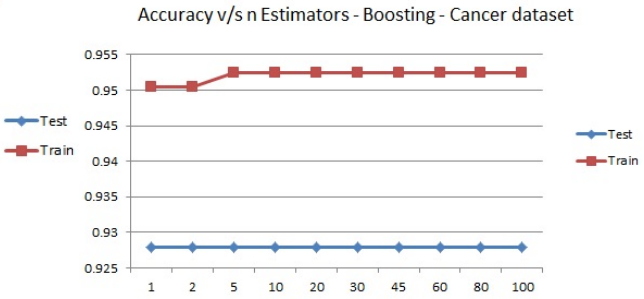


- For both the datasets, the accuracy steeply increases with the increase in the number of iterations indicating that there is no overfitting in the model.

Boosting:

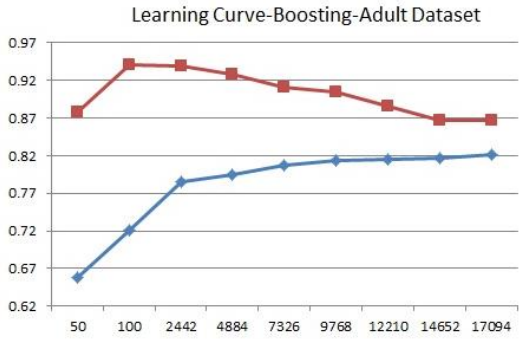
Boosting is an iterative process that learns the information gain and applies it on a subset of the data. Boosting is generally used in combination with other machine learning algorithms to improve the performance. I have implemented boosting with decision tree classifier. It improves the performance of the decision tree by creating a copy of the classification model and adjusting weights in subsequent iterations. The parameters I have used are base estimator from which the boosted ensemble is built and the N estimator at which the boosting is terminated.

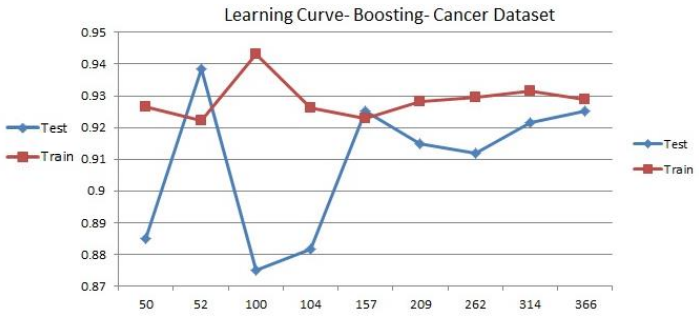




- For the Adult dataset, the accuracy slightly increases upon increasing the number of estimators and stays constant indicating that the model is not overfitting.
- For the Cancer dataset, the accuracy majorly remains constant confirming that the model is not overfitting and that the number of estimators do not influence the accuracy of the dataset with smaller training size. This is obvious considering boosting is an iterative algorithm.

Performance of Boosting on Adult and Cancer datasets:





- The performance shown by the Adult dataset validates the logic of boosting as the accuracy is constantly increasing as we increase the number of training samples. This is because with more training samples, the iterative elimination in boosting works efficiently.
- Whereas the similar trend is not followed in the cancer dataset because the number of training samples are too small for an iterative process like boosting to work which is why there is high fluctuation initially and slight stabilization as the number of samples increase.

Please Turn Over

Summary:

1) What results did you get?

   The smaller dataset gives better classification accuracy and better model fitting.

2) Comparison between different algorithms:

Algorithm	Adult		Cancer	
	Train	Test	Train	Test
KNN	85.2346	78.1357	96.5346	94.5123
Decision tree	83.4317	80.9316	93.6111	92.3128
ANN	81.2356	79.2123	92.0379	91.4712
SVM	83.6312	83.0126	94.3606	93.7882
Boosting	87.9942	82.7653	93.3792	92.8538

3) How long did each algorithm take to run?

   SVM – RBF took the longest to run followed by ANN and boosting. The rest were not time intensive as they are simpler machine learning algorithms.

4) Did cross-validation help?

   Cross-validation minorly improved the performance of the Adult dataset but majorly affected the performance of Cancer dataset. This is because cancer dataset was smaller in size and upon using cross-validation; the use of every single sample in training and testing was promoted.

5) Performance based on problem you chose:

   Overall, the models performed better for the simpler classification problem (Cancer).

6) Performance based on various parameters:

   Described within individual algorithms sections above.

Best Algorithm:

- For the cancer dataset, KNN performed the best. This is because KNN uses a simple distance measure between the data points in order to classify the new data. The positively correlated attributes of the dataset make KNN effective for classifying the data. Moreover there are fewer misclassifications because we are performing only binary classification.
- For the adult dataset, boosting performs the best. This is because the Adult dataset had a huge number of training samples while a relatively small number of attributes. While a decision tree structure works perfectly well with the given shape of the dataset, iterating over the falsely classified samples using boosting helps make the decision tree structure complex, thus improving the accuracy.