**Syed Hashir Ali -** SXA180067

**Applied Machine Learning**-Assignment 2

**Scope:**

In this report two datasets are used to classify two different outcomes. The major classification algorithms used in this assignment are decision trees and support vector machines. There are various experiments for parameter tests and model augmentations (ensembles) carried out to improve model performance. The purpose is to correctly classify “male” and “female” voices from Dataset 1, and” high” and “low” GPU run-time from dataset 2.

**Key Observations:**

* Trees made through both the; tree package and Scikit-learn are unstable and full trees tend to overfit. The DecisionTreeClassifier made more accurate trees through default parameters.
* Cross-validation helps overcome the effect of overfitting and getting closer to best parameters.
* Radial support vector machines tend to overfit, sigmoid kernels do not perform well on either dataset. Hence, if the data is linearly separable, linear kernels should be used.
* “MWG”, “NWG” and “MDMC”, and “meanfun”, “IQR” and “sd” are the most influential features for classification of GPU and Voice recognition data for gradient boosting of trees.
* Boosted trees always perform better than the original fully grown trees.

**Dataset 1: GPU Performance**

This dataset measures the performance of GPU kernel/s. There are 241600 rows in this dataset. There are 14 measurement variables and 4 run time or performance variables. For the purposes of this report, the 4 run time variables are averaged (RunF) to create another binary variable (RunF Binary). If the variable’s value is greater than the average of RunF, the variable is coded 1 or “high”, it is otherwise coded as 0 or “low”. Further details and explanations of variables can be found at <https://archive.ics.uci.edu/ml/datasets/SGEMM+GPU+kernel+performance>.

**Dataset 2: Voice Recognition**

This is a dataset labeled by gender as “male” or 1, and “female” or “0” as response variable. The dataset contains 20 other input variables. Each variable is a different type of statistical measure of the frequency of a voice. This dataset is for voice recognition by gender. The dataset contains an equal number of male and female labels. The dataset contains of 3168 rows and no null values.

**Experiment 1 – Tree vs Scikit-Learn**

In this experiment decision trees of the same data are modeled with the tree function from tree package in R and with the DecisionTreeClassifier in Scikit-learn from Python. To draw a full decision tree for the GPU dataset, an 80-20 train-test split is carried out. Fecision trees are generally not considered to be very powerful algorithms, hence a bigger training set. Both the R and Python packages have different default parameters and slightly different cost functions. For this experiment, the measure of impurity for each node is Gini. Gini impurity measures the degree or probability of a variable being wrongly classified when it is randomly chosen at a node. As our goal is to minimize wrongly classified variables, we use “Gini as our splitting criteria.

For the GPU dataset, tree in R and DecisionTreeClassifier in Scikit-learn both return similar number of branches number of trees. R tree returns a tree of 21 branches while the tree formed using Scikit-learn has 22 branches. The accuracy measures of the trees are given below.

The DecisionTreeClassifier has a much higher accuracy and lower train and test misclassification. However, as we can see below it has a training accuracy of 100%. This can mean that although the full decision tree is fitting the training and even the test set well, it might not perform as well with out of sample data.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Predicted** | **R Tree** | | |  | **Predicted** | **Scikit-learn Decision\_Tree\_Classifier** | | |
| **GPU Train tree** | | |  | **GPU Train tree** | | |
| **Actual** | | |  | **Actual** | | |
|  | High | Low |  |  | High | Low |
| High | 136390 | 13808 |  | High | 142769 | 0 |
| Low | 6570 | 36512 |  | Low | 0 | 50511 |
|  | **Accuracy** | 89.46% |  |  | **Accuracy** | 100% |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Predicted** | **R Tree** | | |  | **Predicted** | **Scikit-learn Decision\_Tree\_Classifier** | | |
| **GPU Test tree** | | |  | **GPU Test tree** | | |
| **Actual** | | |  | **Actual** | | |
|  | High | Low |  |  | High | Low |
| High | 33854 | 3532 |  | High | 35564 | 77 |
| Low | 1639 | 9295 |  | Low | 90 | 12589 |
|  | **Accuracy** | 89.30% |  |  | **Accuracy** | 99.65% |

Despite having the same split criteria as “gini”, the two algorithms return different tree on the same seed values. The trees are unstable and largely depend upon randomization, node splitting criterion entropy vs gini, the test-train split, and the package used.

For the voice detection dataset, R-tree returns a tree with 9 nodes while DecisionTreeClassifier from Scikit-learn returns 10 nodes. The accuracy matrices are given below.

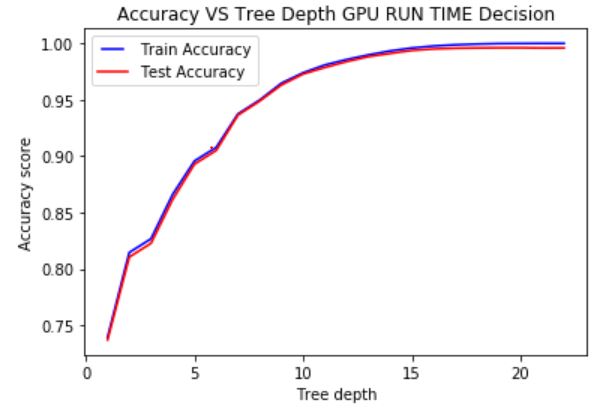
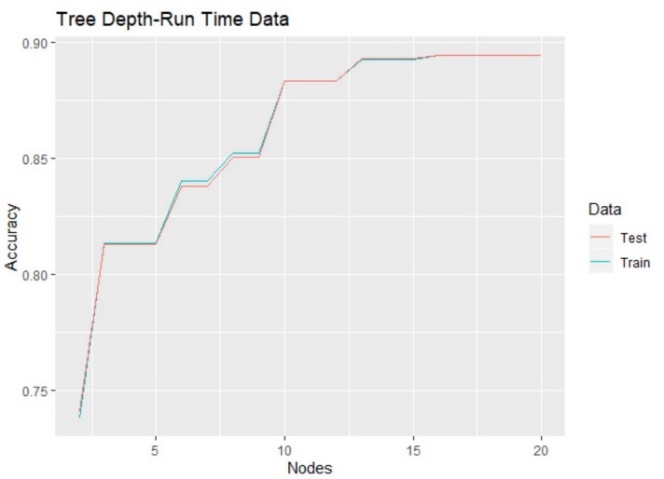
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Predicted** | **R Tree** | | |  | **Predicted** | **Scikit-learn Decision\_Tree\_Classifier** | | |
| **Voice Detection Train** | | |  | **Voice Detection Train** | | |
| **Actual** | | |  | **Actual** | | |
|  | Female | Male |  |  | Female | Male |
| Female | 1246 | 39 |  | Female | 957 | 0 |
| Male | 18 | 1231 |  | Male | 0 | 943 |
|  | **Accuracy** | 97.75% |  |  | **Accuracy** | 100.00% |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Predicted** | **R Tree** | | |  | **Predicted** | **Scikit-learn Decision\_Tree\_Classifier** | | |
| **Voice Detection Test** | | |  | **Voice Detection Test** | | |
| **Actual** | | |  | **Actual** | | |
|  | Female | Male |  |  | Female | Male |
| Female | 308 | 14 |  | Female | 303 | 13 |
| Male | 12 | 300 |  | Male | 15 | 303 |
|  | **Accuracy** | 95.90% |  |  | **Accuracy** | 94.16% |

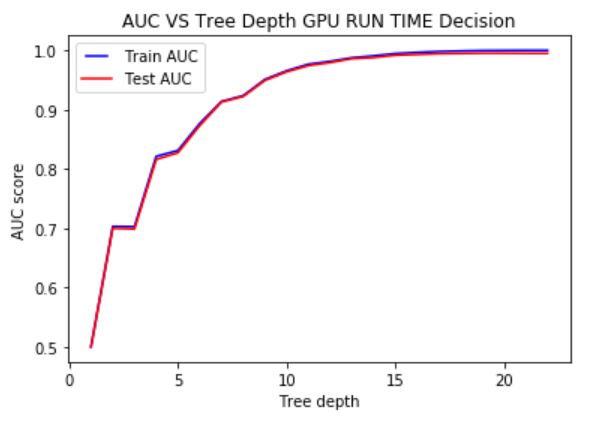
We can conclude that although, Scikit-learn’s DecisionTreeClassifier with its default parameters has a higher accuracy, it is also more prone to high variance as compared to R’s tree. R’s tree on the other hand is more prone to bias and hence, lower accuracy. Both the single trees however remain unstable.

**Experiment 2: Post-pruning vs Pre-pruning**

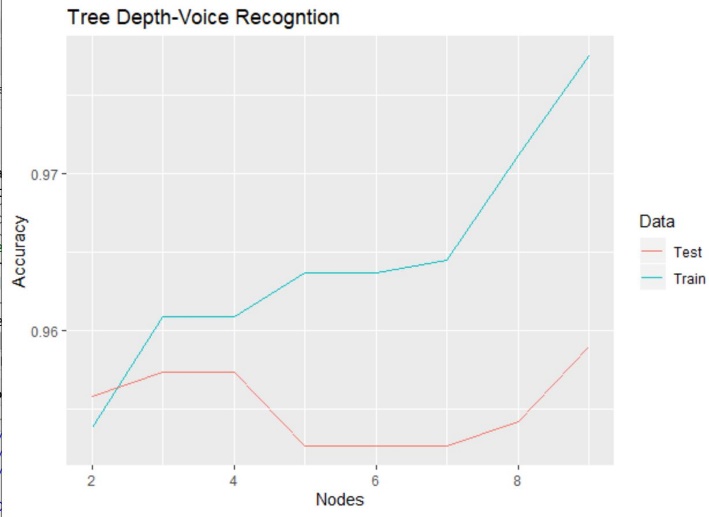
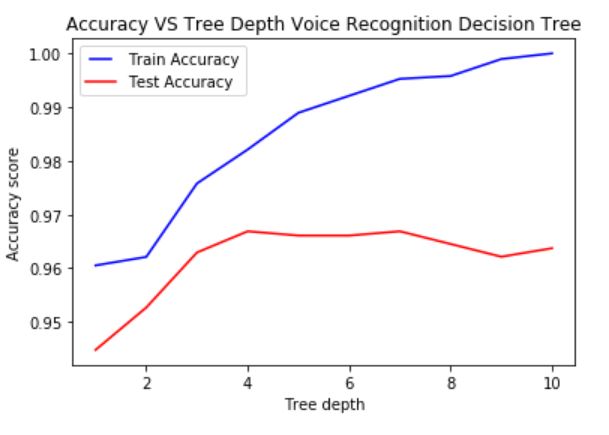
Two approaches of pruning are discussed here; post-pruning and pre-pruning. **Pre-pruning** stops growing the tree earlier, before it perfectly classifies the training set. **Post-pruning** allows the tree to perfectly classify the training set, and then post prune the tree. In the case of high variance, pre-pruning works well, as the tree is pruned before it begins to model the noise. Post-pruning works better when growing the whole tree does not overfit the training data. For the sake of brevity, the R-trees are post-pruned and the DecisionTreeClassifier trees are pre-pruned as they tend to overfit when allowed to perfectly classify the training set.



**GPU Run Time Data. Left: R- Tree Post-pruned. Right: Scikit-learn DecisionTreeClassifier Pre-pruned**

The pre-pruned and post-pruned trees have similar performances. The bias-variance trade-off concluded earlier still holds true. For both the algorithms, a tree with 15 branches seems a good choice as the rate of improvemof further complexity is very small.

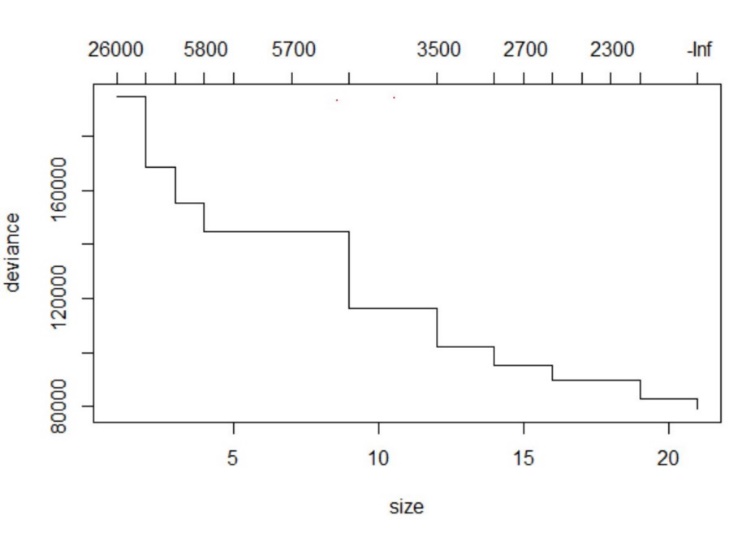
The AUC score strictly follows the accuracy learning curve. This is true for all models as AUC is a derived from proportion of classified outcomes. This trend repeats for all algorithms and is hence only shown once.

**Voice Recognition Data. Left: R- Tree Post-pruned. Right: Scikit-learn DecisionTreeClassifier Pre-pruned**

In the gender recognition data, the bias variance trade-off is evident. As the depth of the tree or number of nodes increases, the model starts fitting the noise in the training set. For both post-pruning and pre-pruning the depth of 4 seems reasonable. After the depth of 4 the testing error begins to fall. We should hence, cross-validate, boost and re-prune the trees.

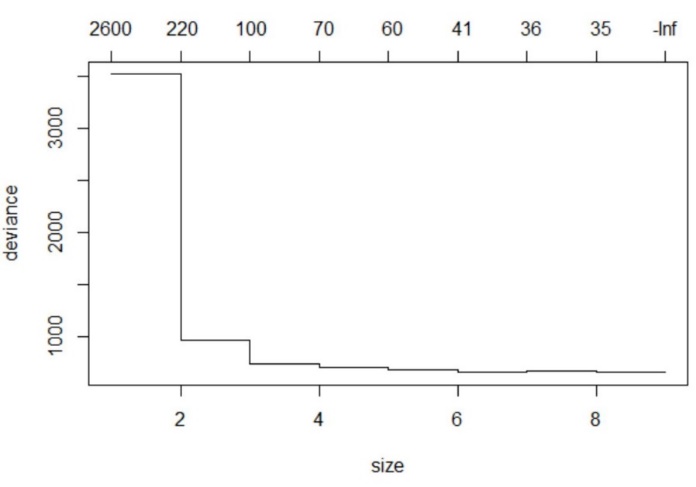
**Experiment 3. Cross-validation of R trees**

From this experiment onwards, there will not be further comparison between the trees created through R and Python libraries. Sine R-trees have a greater room for improvement and to avoid redundancy, only trees created through R have been experimented with.

The R-tree for the **GPU Run Time data** has been cross-validated with **K=10** i-e the data is split into 10 parts for training and testing. The graph measures deviance against influence at every additional node/branch.

**Deviance** measures the difference in "fit" of a candidate model and that of the saturated model. A saturated model is a model that fits exactly or has as many features as variables. Hence, the lower the deviance, the better the fit of the model. Deviance falls as we add more features/nodes to the tree. However, the complexity of the tree rises as we add that node.

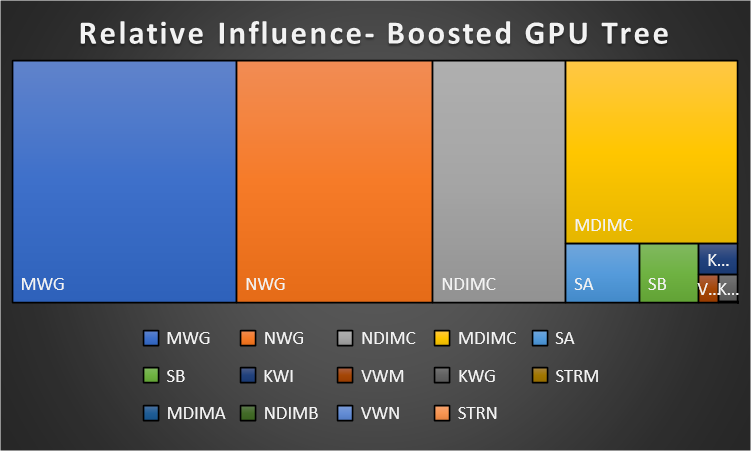
**Influence** of a node measures its contribution in improving the fit. This would be discussed further when these trees are boosted. At a glance, it seems that trees with 12-15 branches fit the data well, after cross-validation.



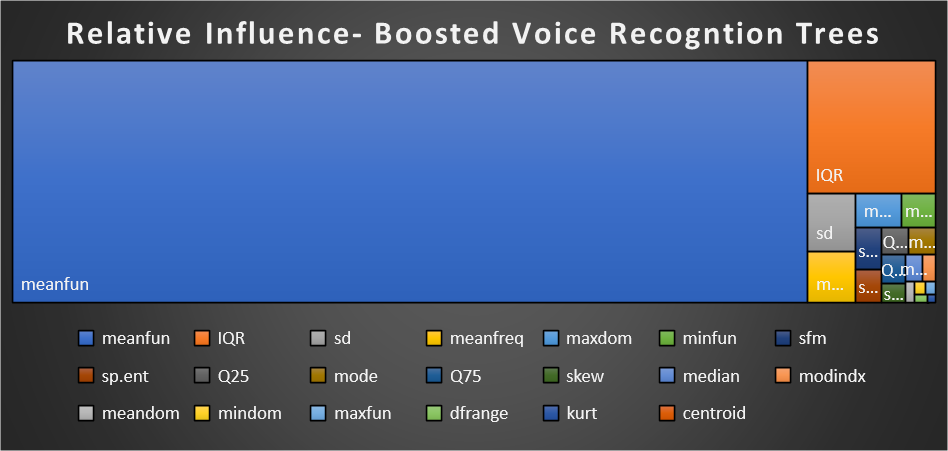
The cross-validated tree for the **voice recognition** data is a good place to explain that the more influential a tree node is, the greater the change in deviance and vice versa. For this tree, there is little improvement in reducing deviance after the first four nodes. Once again, **K=10**.

**Experiment 4: Boosted Trees- GPU Run Time Data**

The tree is boosted using the Gradient Boosting Method in the Cart package in R. The goal is to see if an ensemble of simpler trees would perform better than the tree allowed to grow fully or the best pruning choice tree.

This tree map shows the influence of different variables in improving the fit and reducing deviance for the generalization of the model.

The tree map shows that MWG, followed by NWG, NDIMC and MDIMC have a far larger influence than the other 11 variables

The tuning parameter 'shrinkage' was held constant at a value of 0.1 ROC was used to select the optimal model using the largest value. The final value used for the number of trees was 150, and the depth of each tree was 3. In the accuracy matrix on the left, it can be noted that the boosted tree performs better than both the original fully-grown tree and its pruned version. There is a 5.24% improvement from the original tree in terms of test accuracy.

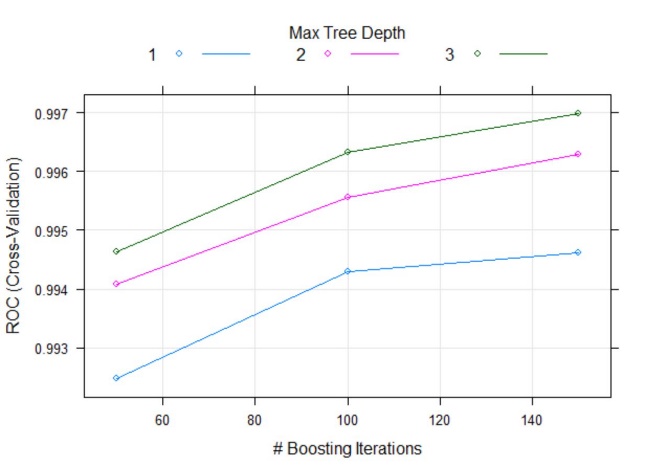
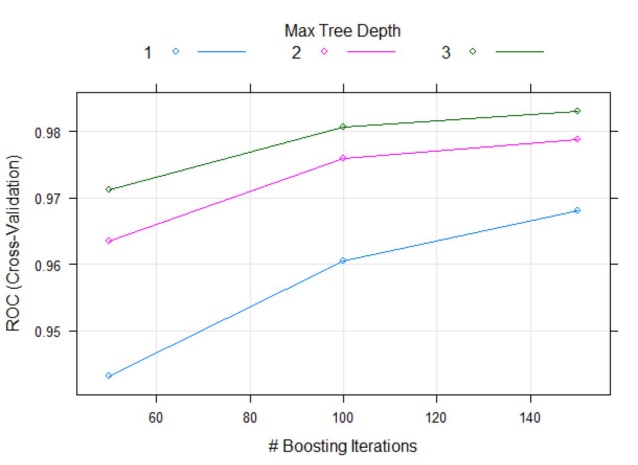
For the voice recognition dataset, meanfun plays a huge influence in the classification of male or female voices. This followed by Interquartile Range and other variables.



Once again, the number of trees

The depth of tree is fixed at 3. The 3 most influential variables are used. There is a 1.18% increase in test accuracy after boosting the tree.

**Experiment 5 Pruning the boosted trees**



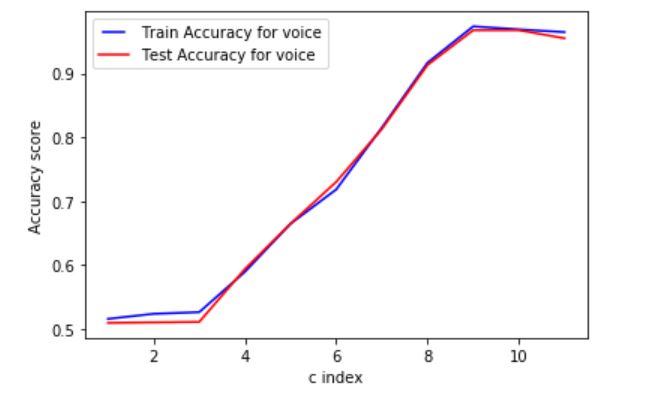
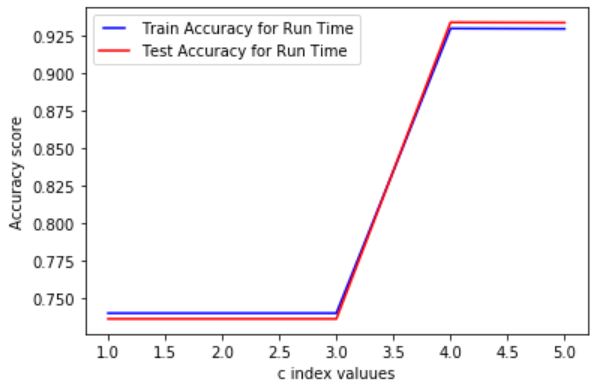
**Left: Boosted tree at each depth for GPU data. Right Boosted tree at each depth for Voice Recognition data.**

For both the datasets, the maximum AUC is achieved with tree depth as 3. The number of trees can be changed however. There is a decline in increase in AUC after 100 iterations. This however is a design decision.

**Experiment 6 & 7: Different Kernels and Parameters for Support Vector Machines**

In this experiment, the **linear**, **sigmoid** and the **radial** kernels are used for support vector machines. Certain elements of the SVC function in the scikit-learn library in Python are left at default. The **c values** are experimented with. The data has been normalized for GPU dataset to optimize the computing time of the algorithms. The C parameter tells the SVM optimization how much to avoid misclassifying each training observation. For large values of C, SVM chooses smaller-margin hyperplanes if that hyperplane does a better job of getting all the training points classified correctly, and vice versa. Tree-based algorithms tend to perform better on non-normalized data, so it was not scaled before.

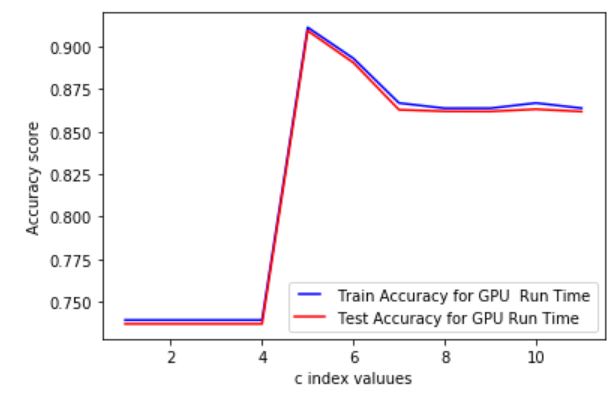
**Linear Kernel**

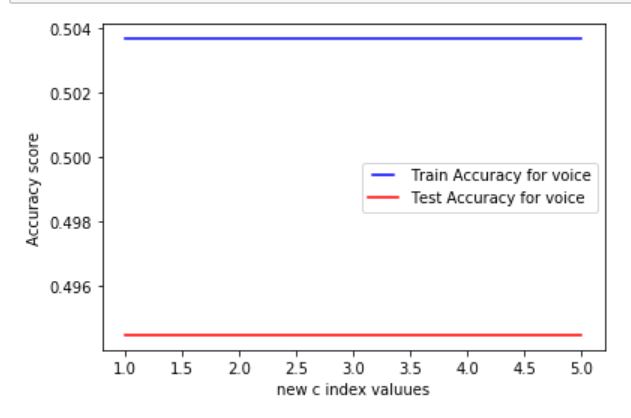
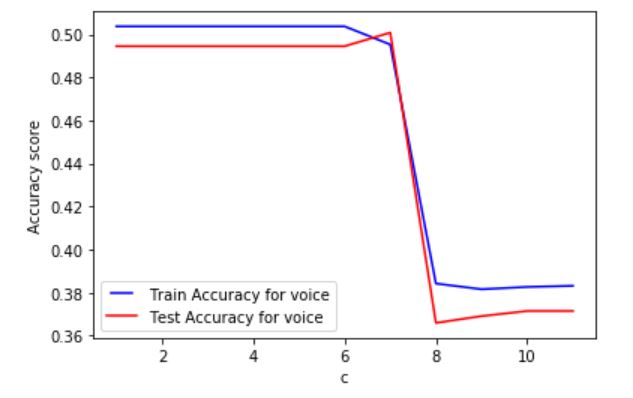


**linear GPU c= 10^-30, 10^-20,10^-10,10^-1,10^0 SVM linear Voice c= 0.0000001, 0.000001, 0.00001, 0.0001, ……………………………………………………………………………………………………………………………… … 0.001,0.01,0.1,1,10,100,1000**

The Linear Kernel performs well for both the GPU and Voice recognition datasets. For the GPU dataset, the optimal choice of the hyperplane is between 3rd and 4th value of the index. It increases at c= 10^-10 till c=10^-1. In the last experiment, cross-validation would allow getting closer to a better range of answers. For the Voice Recognition dataset, the optimal c value exists between the 8th and the 10th index i-e c=10 and c=100.

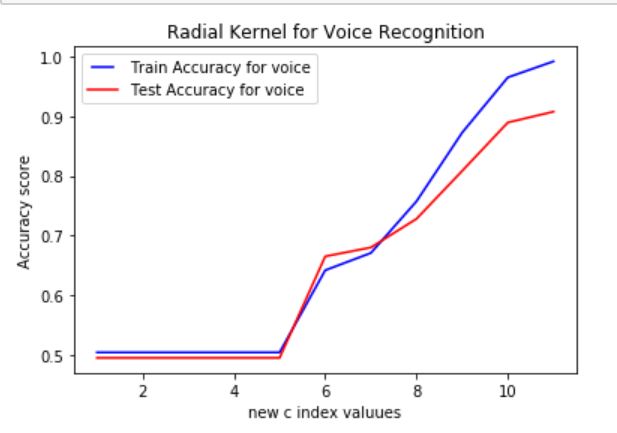
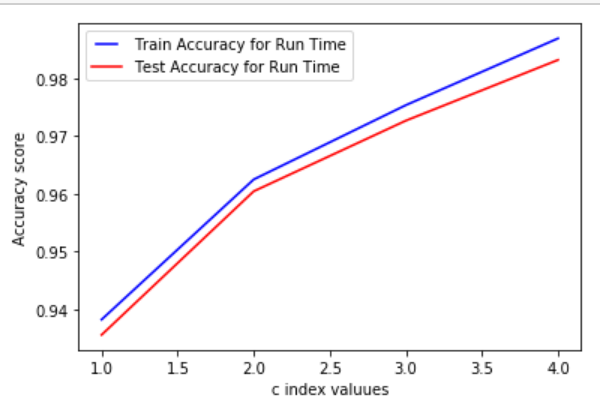
**Sigmoid Kernel**

This graph shows the accuracy at different levels c for the GPU data. The different values of C = 0.0000001, 0.000001, 0.00001, 0.0001, 0.001,0.01,0.1,1,10,100,1000. The optimal value of C is between the 4th and 5th index so between 0.0001 and 0.001. The sigmoid kernel has a lower accuracy than the linear kernel.



Both the graphs above show that the sigmoid kernel did not perform well for the Voice Recognition dataset at other default parameters. There is even a decline in accuracy which stays constant at the new range of c values. The first range of c values was c=0.0000001, 0.000001, 0.00001, 0.0001, 0.001,0.01,0.1,1,10,100,1000. Seeing the upward trend around c=1000, the new range of c was c=10^1015, 10^-20,10^-25,10^30.

**Radial Kernel ‘rbf’**

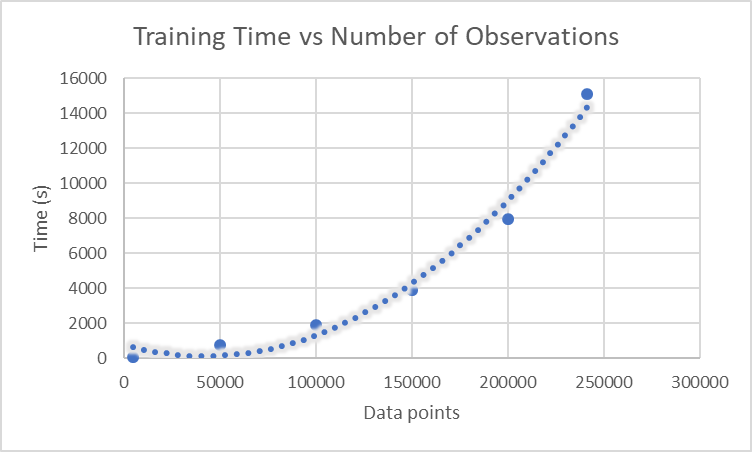


**Left rbf kernel for GPU Data with c= 10^-2,10^-1,10^0,10^1. Right rbf kernel for voice data with c=0.0000001, 0.000001, 0.00001, 0.0001, 0.001,0.01,0.1,1,10,100,1000**

RBF is the most complex kernel trained on so far. It is expensive in terms of training time but performs better than the other two kernels for GPU data. For both datasets, rbf tends to begin to overfit at larger values of c. This is depicted by the growing difference between training and testing sets as c increases. Cross-validation is required to avoid variance and reach a better value of c.

**Experiment 8. Support Vector Machine Training Time.**

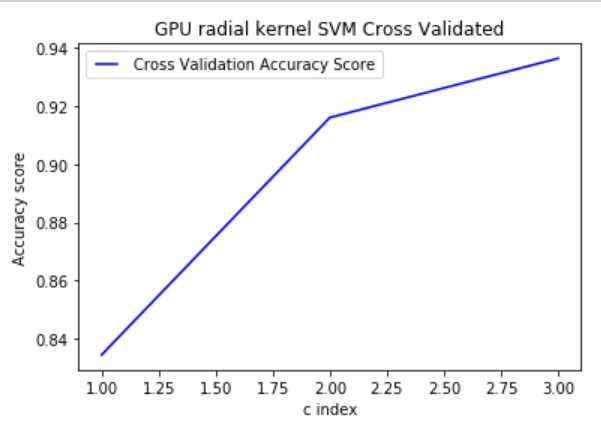
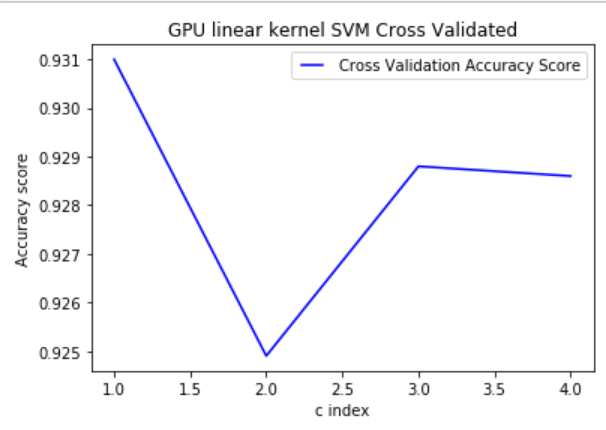
It is normal for support vector machine algorithms to take a lot of time. 2 hours on a typical OS is normal for an algorithm However, once the kernel is tested on different kernels and parameters, it can take exponentially longer. The simplest kernel in terms of complexity and hence training time is the linear kernel. It trains the Voice recognition in seconds. In this experiment, the linear kernel is tested on different subsets of the GPU dataset that originally has 241600 observations. This is shown in the graph below.

 The graph shows that the time taken to train the kernel increases exponentially. Therefore, a k-fold cross-validation or a parameter test with 10 values can take an entire day for complex kernels on the GPU data. On the other hand, it may take an hour to do similar task on the voice recognition data. In the interest of time, this experiment was not extended to more complex kernels.

**Experiment 9: Cross-Validating Support Vector Machine**

As seen in the previous experiments, certain kernels are prone to overfitting. The purpose behind this experiment is to improve the test accuracy for the different kernels of support vector machines through cross-validation. Different parameters of c i-e the penalty for misclassification have already been experimented with. Another purpose of cross-validation is to reach a value of c that is closer to its optimal value in terms of maximum accuracy.

**GPU Run Time Dataset:**



**SVM Linear GPU c=0.01, 0.03, 0.05,0.07 c=10^-1,10^0,10^1**

As discussed in **Experiment 8**, SVM takes a lot of time to run on the GPU dataset. Several values of k for **k-fold cross-validation** and model parameters were tried. To reduce the computing time of the SVMs, two procedures are carried out. Firstly, the x-variables or features of the GPU dataset are scaled. SVMs that still took several hours to cross-validate were cross-validated on a sample of the data.

**Linear Kernel:**

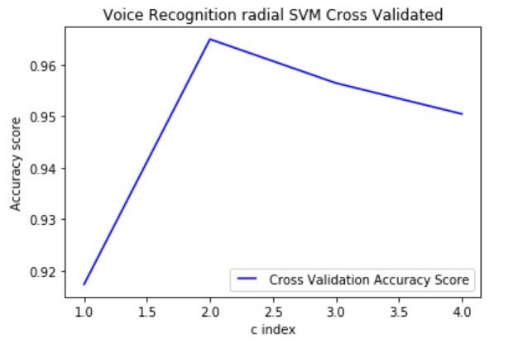
In **Experiment** **6 and 7,** the linear kernel for GPU data showed a steady increase in accuracy in the range c=10^-10 and 10^-1. This is still a very wide range depending on how precisely the misclassification penalty must be classified. For this experiment, the range of c=0.01 to 0.07 was chosen. The best accuracy value comes out at c=0.01. The test accuracy does not improve, as compared to the original linear kernel for this c value.

**Sigmoid Kernel:** Interestingly, the sigmoid kernel had **test accuracy over 90%** in the single split SVM for the c values in the range of **0.0001 to 0.001 in Experiment 6 and 7**. It now has a test accuracy = 0.74 for all values of c. Since this is k-fold cross-validation with k=5 instead of a single split, these results are more reliable.

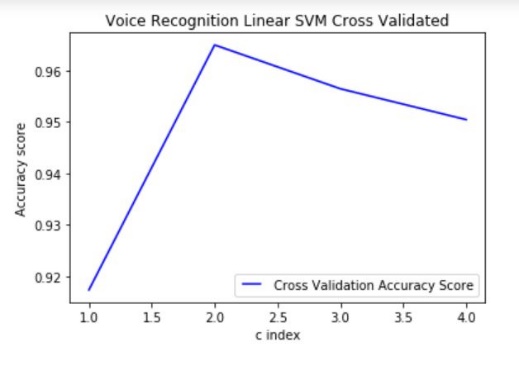
**Radial Kernel:** This kernel has a tendency of overfitting. The max cross-validated accuracy value is at c=10 and it is 0.936. Although this is lower than the original radial kernel on GPU data, it is a more consistent measure of accuracy for the test set.

**Voice Recognition Dataset:**

In experiment 6 and 7, sigmoid kernel did not work for the Voice Recognition dataset. Despite several trails of the C parameter, accuracy did not improve. Hence, in this experiment, sigmoid kernel is omitted from cross-validation. Below are the results for the linear kernel and the radial kernel respectively.

**Linear Kernel**   
In experiment 6 and 7, the accuracy increased mainly in the range **c=1 and c=1000**  for the linearkernel**.** In crossvalidation with **k=5**, this narrower range is used for c values. A peak can be seen at the second index for c. This is **c=10**. The test accuracy for the cross-validated ‘rbf’ kernel has also cross-validated test accuracy for the kernel is **0.964**.

**SVM linear Voice CV c=1,10,100,1000**

**Radial Kernel**  
In experiment 6 and 7, the accuracy increased mainly in the range **c=1 and c=1000** for the radial kernel on Voice Recognition data**.** Incrossvalidation with **k=5**, a this narrower range is used for c values. A peak can be seen at the second index for c. This is **c=10**. The test accuracy for the cross-validated ‘rbf’ kernel has also **improved from 0.9077 to 0.964**

**SVM RBF Voice CV c=1,10,100,1000**

**Future Considerations:**

* A deep breakdown of the tree and DecisionTreeClassifier to understand why the algorithms produce different accuracies. Documentation on both is available [here](https://www.rdocumentation.org/packages/tree/versions/1.0-40/topics/tree) and [here](https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html).
* Increase access to computing power to try polynomial kernel for SVM.
* Cross-validation does not improve the test accuracy on all SVMs. The value of k for k-fold cross-validation can be tested.
* The radial kernel tends to overfit. The value of gamma parameter (the radial influence of each data point) in rbf kernel svm can also be experimented with.
* If resoureces allow, implement a super ensemble of the trees and the SVMs.