**Syed Hashir Ali**

**Sxa180067**

**Assignment 4**

**Scope**

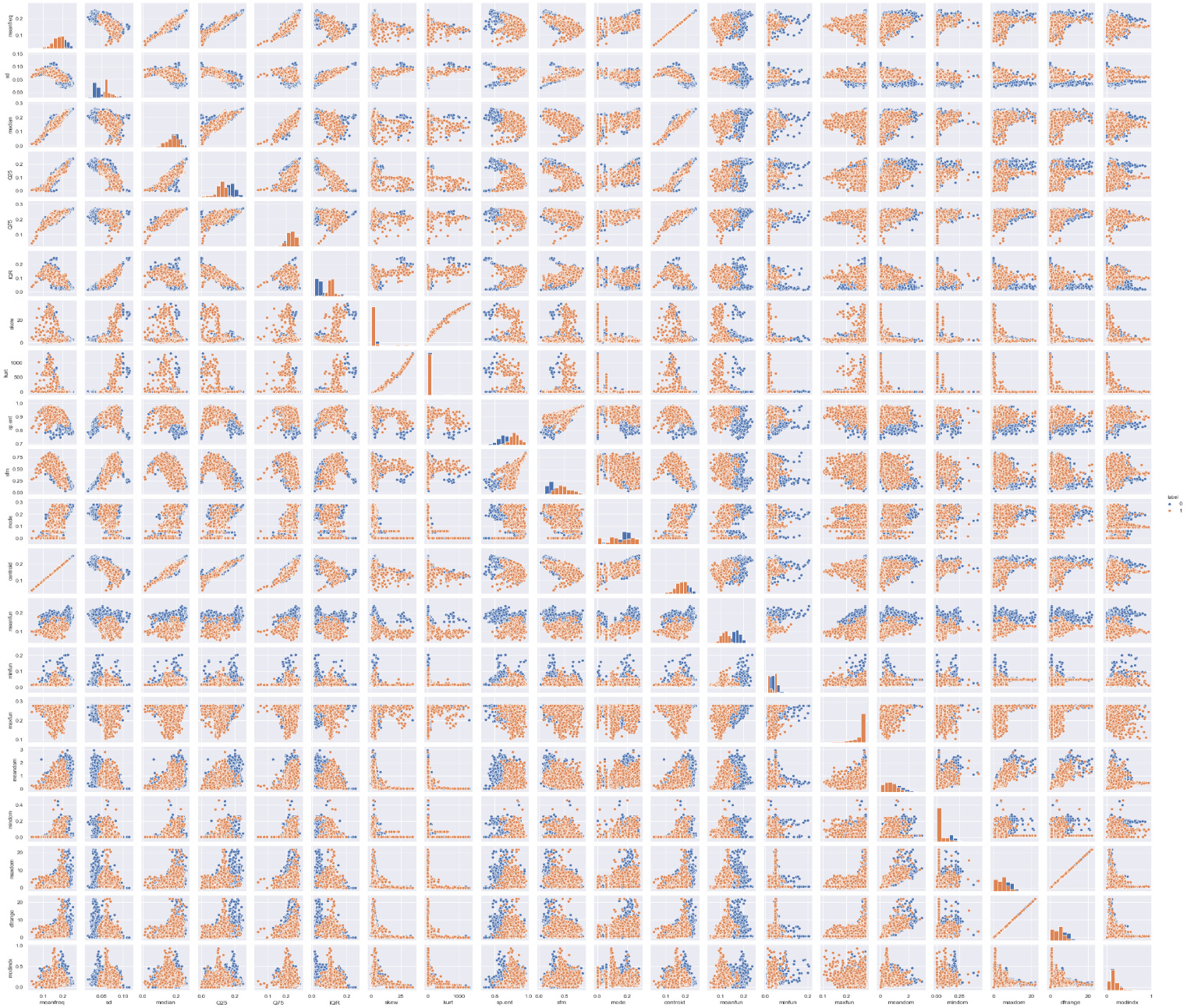
In this report two datasets are preprocessed, clustered, and later classified. This report is the 4th part of an ongoing research on 2 classification problems; Voice Recognition and GPU Run Time Prediction. This part of the research explores clusters in the input features through K-means and Gaussian Mixture Models. The input features are then transformed into Principal Components, Randomized Projections and Independent Components to reduce dimensions, create further clusters and eventually classify the data through Artificial Neural networks.

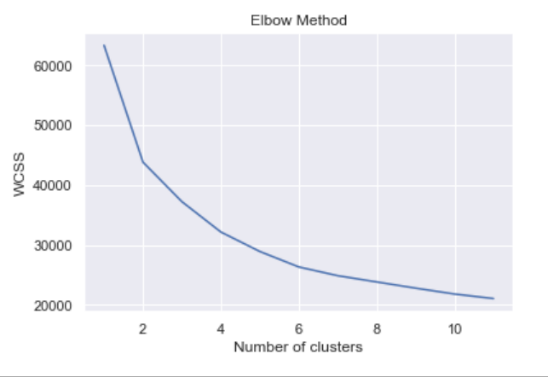
**Key Observations**

* Even when K-means clustering results in the same number of clusters as classes, the clusters may not be the same as classes.
* K-means and GMM clusters can outline new patterns in the data. These patterns can be interesting features that may have a high correlation with the outcome/feature of interest.
* As we move away from the high concentration areas, and closer to the cluster boundary, the probability of belonging to a class decreases.
* Voice Recognition clusters have a **good Silhouette score (0.291) but bad Adjusted Rand Score (0.07)**. These clusters show good patterns on their own but are not the same segregation as the original labels.
* The GMM clusters for both Voice Recognition and GPU Run Time Prediction have similarity to the K-means cluster assignments with an **Adjusted Rand Score of 0.78 and 0.93 respectively.**
* Filtering the features makes cluster assignments 4-5 times more similar to original labels for Voice Recognition. This experiment is irrelevant for GPU as it has a different number of clusters than the true labels.
* Varying the number of PCA, RP and ICA components used in clustering, varies the similarity of the resulting clusters with the actual labels, noted by a change in Adjusted Rand Score.

**Data and Features**

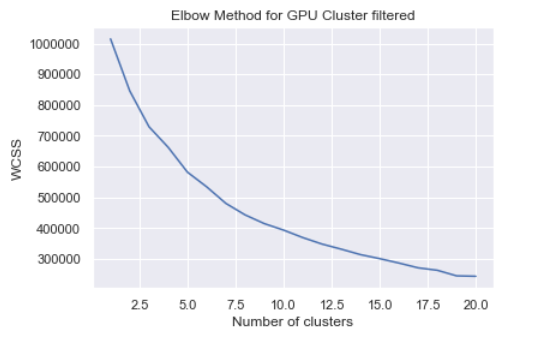
The report is based on 2 datasets. The data used to predict GPU Run Time has 14 features and 4 continuous output variables. These 4 output variables are averaged to create a final binary label for high and low Run Time. The features are numerical but are not distributed continuously. This can be seen in the correlation plot (partially shown on the left) that is color coded by the binary output variable. The data does not have very high variation but the segregation of the binary variable can still be noticed in some features. The data for voice recognition can be seen in contrast. The plot on the next page shows how the features for Voice Recognition have much more variation. Several variables in this data show the segregation between observations associated with male and female voices.

****

**Clustering**

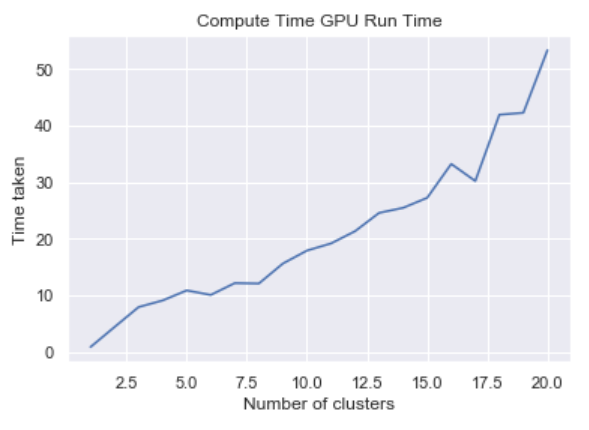
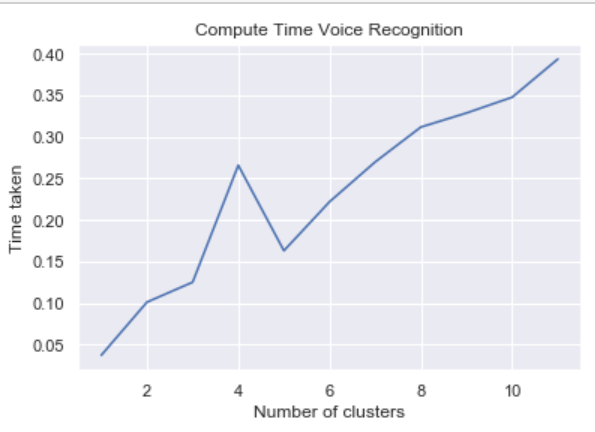
**K-Means Clustering**

The ground truth for both datasets is known. Voice Recognition has a natural binary outcome as male or female, but GPU Run Time only has a binary outcome that was artificially engineered for convenience. After normalizing the features, Kmeans- a Euclidean distance based clustering algorithm is applied to these two datasets.

The elbow curve on the left shows with-in group sum of squares (WCSS for the Voice Recognition data. As expected, this dataset has a clear elbow **(or drop in WCSS) at k=2**. 

This data is separable into two clusters just like the original labels (male/ female).   
On the contrary, the data for GPU Run Time Prediction does not show the same pattern.

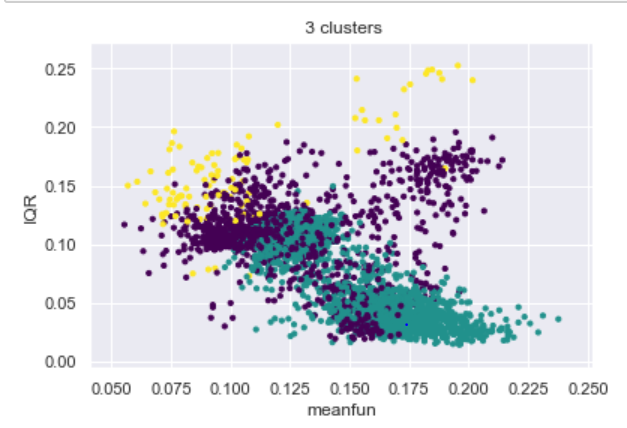
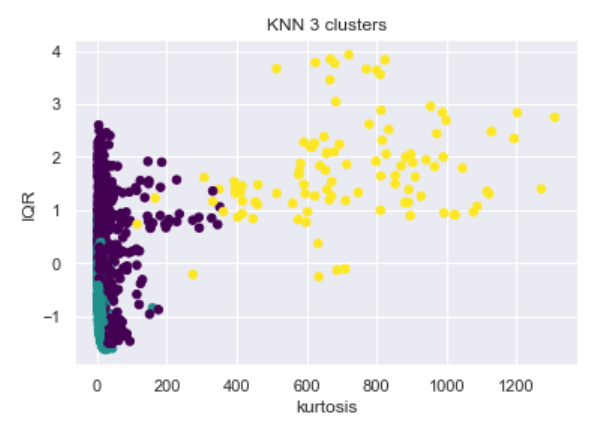
Running Kmeans clustering shows that the data can be grouped into **8 clusters** with a sharp decline in WCSS.



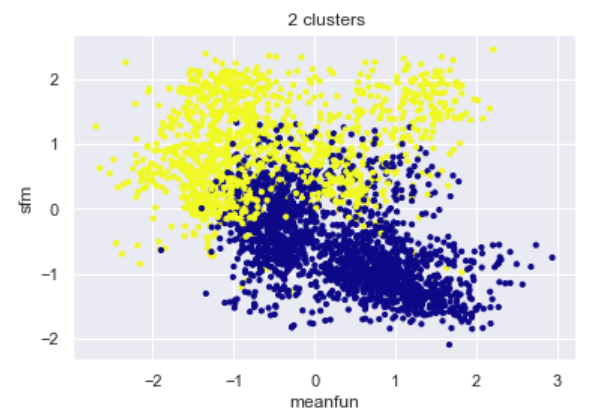
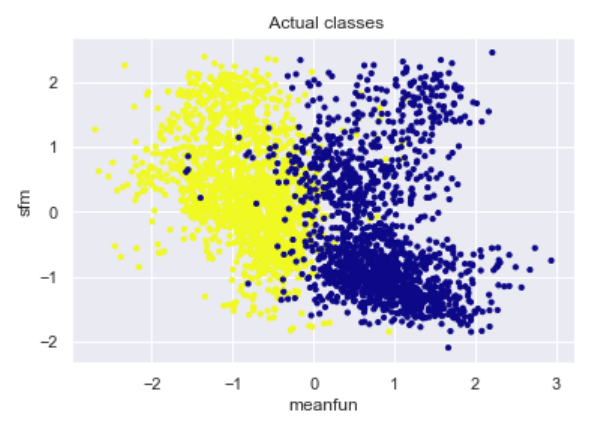
Kmeans is among the faster algorithms for clustering when k is low. However, compute time can quickly increase as the number of clusters and observations rise. 2 clusters for Voice Recognition and 8 clusters for GPU Run Time have been selected because there is a **sharp decrease in WCSS** but **not a sharp increase in computing time** at these cluster levels.

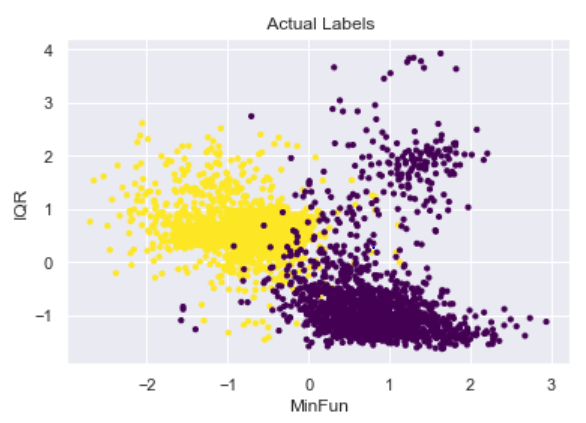
As discussed earlier, GPU Run Time does not have a lot of variation and there are several “pseudo continuous” observations that occur in batches. This leads to clusters that look disparate but may not be good.

The **Silouhette Score** is repeatedly reported in this research to meassure the quality of clusters in this research. The Silhouette Coefficient is calculated using the mean intra-cluster distance (a) and the mean nearest-cluster distance (b) for each sample. Silouhette scores near +1 indicate that the sample is far away from the neighboring clusters. A value of 0 indicates that the sample is on or very close to the decision boundary between two neighboring clusters and negative values indicate that those samples might have been assigned to the wrong cluster. The Silhouette Score for these 8 clusters is **0.0924.** This is a low score (near 0) and signifies that the clusters maybe close to each other.

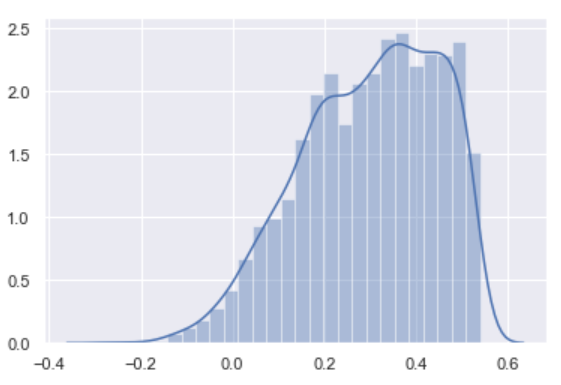
 

On the contrary, Voice recognition shows a more scattered variation that can be visualized with k =3 and k=2. K-means with k=2 can be contrasted with the actual labels.

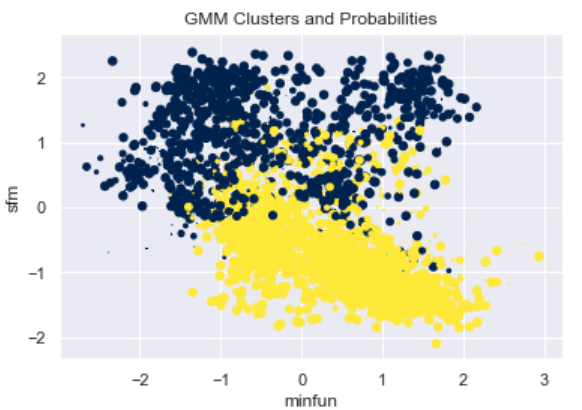
 

Plots on the **“most influential features”** (more on this later) show that there is a clear separation between clusters, but a lot of overlap can still be seen. Although the number of clusters is equal to the number of classes, the groupings are not the same. K-Means clustering may have explored new patterns in the data. These patterns are different from the original features. Rand Index is used to measure the similarity between the cluster assignments of K-Means, Actual labels and GMM (later to come). The **Adjusted Rand Score** computes a similarity measure between two cluster assignments by considering all pairs of samples and counting pairs that are assigned in the same or different clusters in the predicted and comparison labels. The adjusted Rand index is thus ensured to have a value close to 0.0 for random labeling independently of the number of clusters and samples and exactly 1.0 when the cluster assignments are identical The Adjusted Rand Score for K-means clustering for Voice Recognition clusters and the original labels is 0.07. These clusters have a **good Silhouette score (0.291) but bad Adjusted Rand Score (0.07)**. These clusters hence show good patterns on their own but are not the same segregation as the original labels.

As discussed earlier, the Silhouette Score for Voice Recognition Data is much higher than GPU Run Time Data. The figure on the left shows Silhouette Scores on the x-axis and relative frequencies of different samples on the y-axis for Voice Recognition data. The left-skewed distribution shows that the clusters are generally disparate and good. The **Mean Silhouette Score across all samples is 0.291.**

**Gaussian Mixed Models**

This algorithm allows for easy evaluation of, sampling from, and maximum-likelihood estimation of the parameters of a GMM distribution. GMM resulted in similar clusters to K-means.

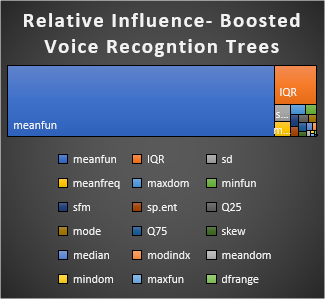
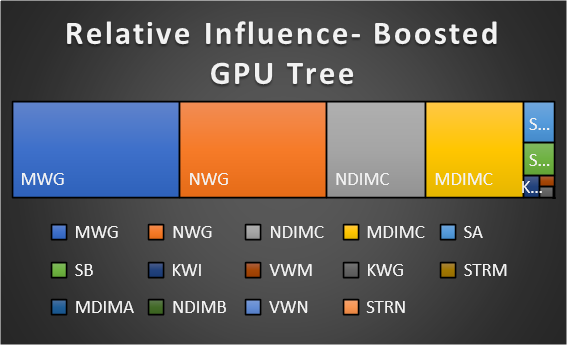


The graph shown above for Voice Recognition depicts the two clusters according to GMM. The colors represent the two clusters while the size of an observation represents the probability of belonging to that cluster. As we move away from the high concentration areas, and closer to the cluster boundary, the probability of belonging to a class decreases.

The **Silhouette Score for the Voice Recognition data is 0.299.** GMM improves the score by 0.008. GMM clusters for Voice Recognition are very different from the original labels. The **Adjusted Rand Score for these cluster assignments is 0.05.** This is even lower than the K-means cluster assignments. On the other hand, the Silhouette score for the **GMM clusters in GPU Run Time prediction stays roughly the same as K-means at 0.0923**. The GMM clusters for both Voice Recognition and GPU Run Time Prediction have similarity to the K-means cluster assignments with an **Adjusted Rand Score of 0.78 and 0.93 respectively.**

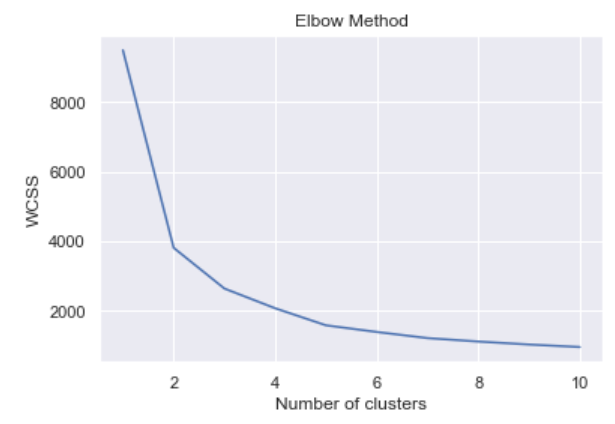
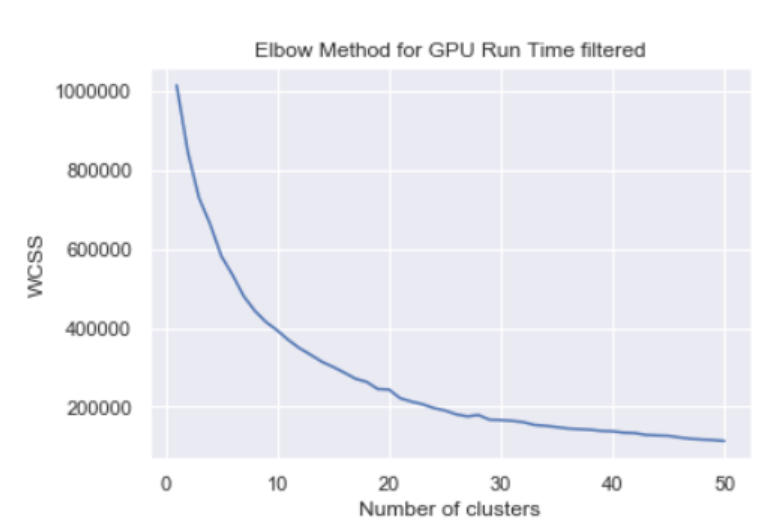
**Transformation and Dimensionality Reduction**

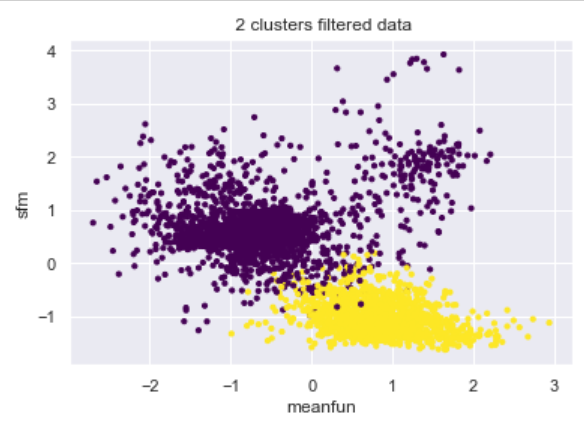
**Filtering:** In other parts of this research, Decision Trees are used to find the relative influence of different features in the 2 datasets.

These tree charts for boosted trees show the **Relative Influence** in branching out the data to its classes. Relative influence can be seen as a measure of contribution to information gain. Voice Recognition features are filtered to only keep meanfun and IQR. GPU Run Time Prediction features are filtered to keep only MWG, NWG, NDIMC, MDIMC, SA and SB.

The clustering algorithms are run on both the datasets again. For the sake of brevity, the elbow curves for the algorithms are only shown once.

The Voice Recognition Data shows a clear loss in WCSS at 2 even after filtering. This can be the bases for selecting k in both K-means and GMM. The GPU Run Time Prediction data does not show the same results. The elbow is not as clear as before. There is a rapid decline in WCSS in the k=7 to k=10 range. For the sake of consistency, future experiments with GPU Run Time ave been conducted with k=8.

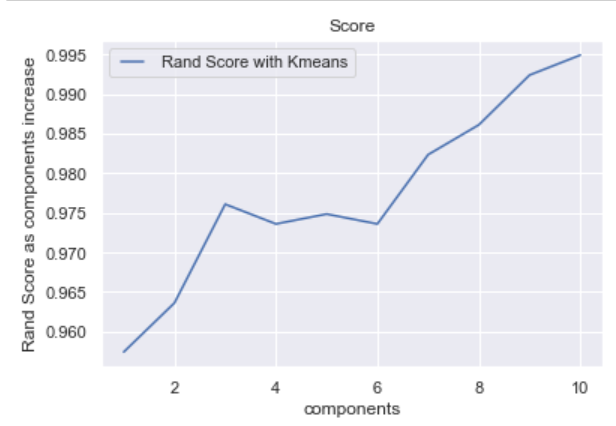
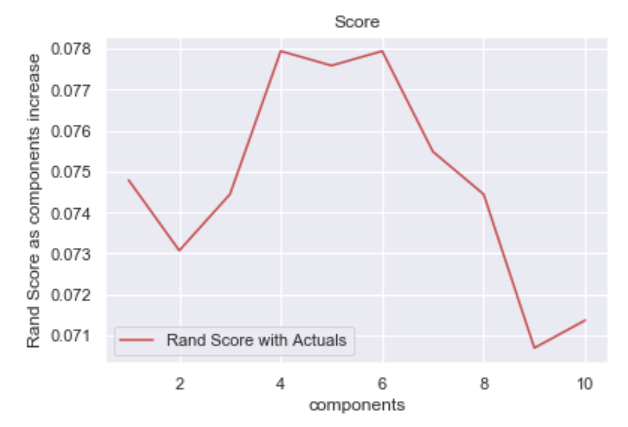
We can contrast this ploto the ones shown for the original labels and the cluster assignments before filtering. The plot shows different cluster assignments then before. There is also a visual improvement in the separation of the clusters. The **Adjusted Rand Score for Voice Recognition Data with the original assignments improves to 0.25**. This is a huge improvement from 0.07 (K-means) and 0.05.

Filtering the features has a positive impact on clustering in terms of similarity with original cluster assignments.

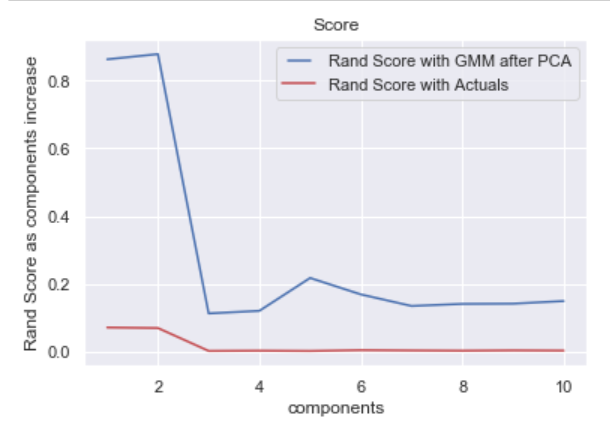
**Principal Component Analysis**

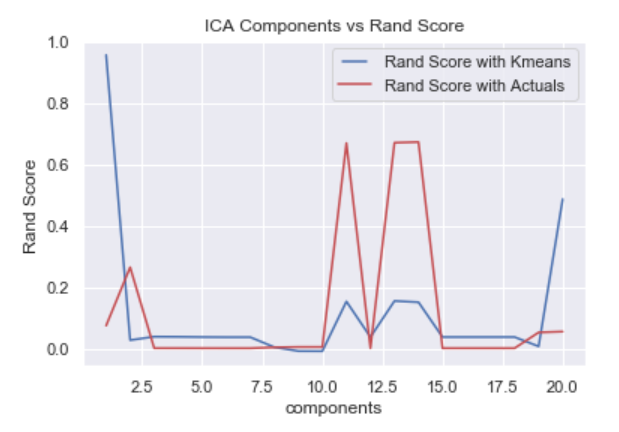
Principal Component Analysis is used to transform Voice Recognition to 13 components from 20 features and GPU Run Time to 10 components from 14 features. Clustering algorithms are run on the transformed components that capture the variation of the original data as follows.

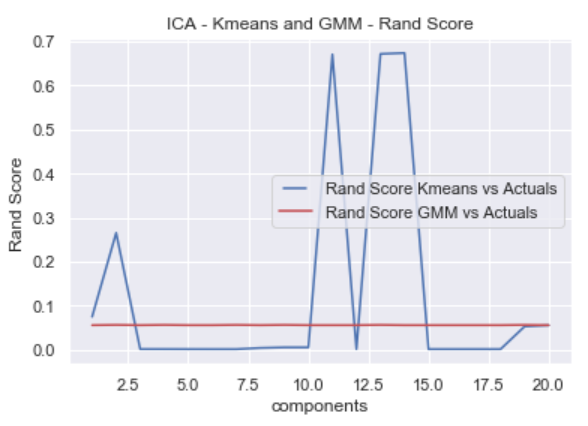
After running the clustering algorithms on all the components of GPU data, we see a **Silhouette Score of 0.08.** This is a lower than before PCA for both K-means and GMM.

These plot on the left shows that as we **increase the number of PCA components used for K-means clustering, the clustering assignments become more similar to the clustering** **assignments before PCA**. The second plot shows that as we increase the number of componets to a certain point (6), the similarity of the clustering assignments based on PCA components to the true/actual labels increases. However, it decreases after that.

The plot on the left shows the same information for the GMM algorithm. Interestingly, the similarity with actuals shows the same trend. However, the Rand Score before and after GMM does not show the same trend. Here, **fewer componets yield more similar results to GMM clustering before PCA.**

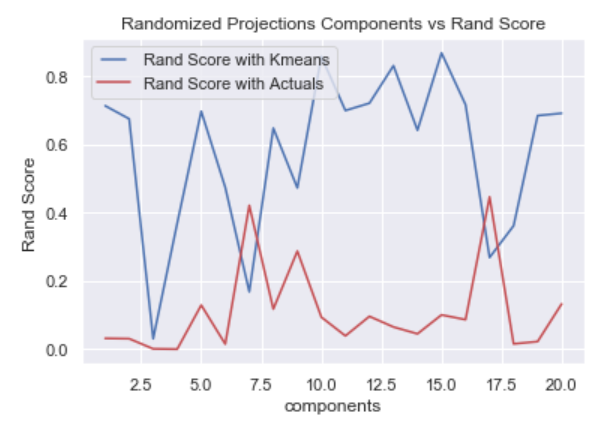
**Independent Component Analysis**



ICA is used to divide the data into 20 independent components. Variations of these components are then used for clustering algorithms. Although ICA can be used for feature elimination, it is better explained as a feature separation technique.

The plot (left) show that we get very different clusters from the K-means algorithm run on entire features. **The before/after ICA similarity measured by Adjusted Rand Score increases as we use almost all (18+) components.** Moreover, using **11, 13 or 14 components makes our clusters very similar to the original labels.** This is the **highest similarity with the original labels seen so far.**  The plot on the right shows the same information about GMM algorithm. The trend is the same for GMM however, it can be noted that the **similarity of GMM cluster assignments after ICA to the GMM cluster assignments before ICA is even lower** (lower Adjusted Rand Score).

**Randomized Projections**

The dimensions and distribution of random projections matrices are controlled to preserve the pairwise distances between any two samples of the dataset.  
The graph shows that clustering on random projections with **6 and 16 components has the highest similarity with the original labels.** Moreover, the similarity with clustering before the transformation remains generally high (blue line). However, **when 6 or 16 components are used, cluster assignments are less similar to cluster assignments before feature transformation.** This is true for voice recognition data features only.

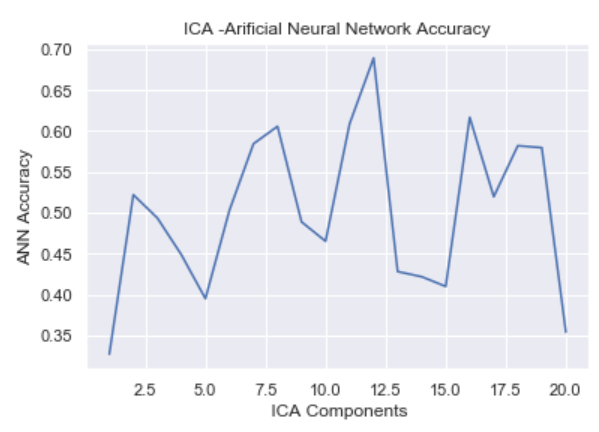
**Artificial Neural Networks (ANN)**

In previous research, the best performing ANN for Voice Recognition had, logistic activation, lgbdfs solver, 3 layers and 10 nodes in each layer. Similarly, for GPU Run Time Prediction it had relu solver, adam activation, 5 layers with 6 nodes in each. These 2 ANNs are now used on the best performing dimensionality reduction techniques seen above. These are Filtering and ICA.

These accuracy matrices show that the ANN perform well for the filtered features for both data.

**While the GPU dataset loses just 2% accuracy after dropping 10 features, Voice prediction does not lose much**.

Although ICA showed promising results in terms of Adjusted Rand Score with the original label, the otherwise best performing ANN for features fails on the transformed independent components. The error rate is over 46% and the ANN performs just better than a naïve approach for prediction.

  
 The high error can be seen in the accuracy matrix on the left. The graph on the right shows how using a different number of independent components can increase the accuracy of this ANN. The **highest accuracy 68% is seen when using 12 independent components.**

**Clustering Results as Features**

For each of the 2 datasets, cluster results from K-means and GMM were used as features and the ANN was used for predictions. The accuracy matrix on the left shows that with just K-means and GMM original cluster assignments as features, the same **ANN performs with an accuracy of almost 64%**. If we add the cluster assignments of KNN on Filtered features as an **additional feature, the accuracy jumps to 89.1%**.

For GPU Run Time the best performing ANN failed with just K-means and GMM clusters in the input layer. However, once the cluster assignments from filtering were added to the input layer of the ANN, the accuracy jumped to 81%.

Neither of these results mean that ANN can not perform on the cluster outcomes as features. Perhaps, tuning the number of layers, nodes, activation functions and components can help achieve better accuracy. This can be a future consideration in further research.