# UNSUPERVISED LEARNING

Aayush Kumar

T-SQUARE ID: akumar372

## Introduction

### Datasets

To explore Unsupervised Learning and Dimensionality Reduction, I chose two datasets such that one that offered a handful of features and arguably had related patterns in the labels discernible to human observation and intuition, while the other one offered more features and far more intricate relations that governed the labels of each instance. As a result, for the former I settled on a gym dataset from Kaggle that documents the crowdedness of campus gyms over the course of a year, a real world situation that I have domain knowledge of and wanted to see how well and quickly each algorithm picked up on certain patterns that we students observe along with those which we perhaps overlook. For the second, I chose a song dataset from the UCI ML Repository that archived the timbre of songs since the mid 1900s and could be used to predict which year a song was released in based on its sound characteristics.

| | GYM DATASET | SONG DATASET |
|---|---|---|
| Total Instances: | 62184 | 515345 |
| Total Features: | 10 | 90 |
| Used Instances | 20000 (15000 Train + 5000 Test) | 75000 (52500 Train + 22500 Test) |
| Used Features | 8 | 12 |

To account for incomparable units in our dataset, I normalized the features to zero mean and unit variance- this ensured that later on in my case study I would not differentiate clusters along certain attributes solely because they had larger variance while giving bias to attributes with smaller variance for forming individual clusters.

### *Hypotheses*

Starting out I had very low expectations that my datasets would be good candidates for unsupervised learning. There are multiple specific combinations of conditions that warrant the same level of crowdedness in a campus gym, and thus using clustering as an approach is inherently at a disadvantage. Similarly, the style of music that is released in every decade is not inherently similar in all attributes. We see that different genres evolve in different ways throughout years, and thus the different timbre of each genre would develop in its own way throughout decades. Thus, two songs of different genres, with very different but reasonable timbres, can be from the same year but would be nearly impossible to cluster in high dimensional space. This all proved to be true, as a I got some frankly terrible accuracies, but I was still quite interested in how effective the Dimensionality Reduction algorithms would play into Supervised learning on both datasets!

## Measuring Performance

### *Clustering*

In order to gauge how well my unsupervised learning algorithms were performing, I used a measure of Adjusted Mutual Information (AMI) between two clusterings, which while relatively slower when compared to other accuracy measures provides several benefits. Namely, AMI is a variation of Mutual Information, which we discussed in class as a measure of the mutual dependence between two variables, where chance is also accounted for. Because we have no idea of how many instances we will receive per corresponding cluster, AMI is able to take into account varying numbers of clusters and purely assess the amount of information shared. In addition, "a permutation of the class or cluster label values won't change the score value in any way."[1] Lastly, it enables symmetry across both inputs where switching the ground truth labels with the predicted labels will yield the same score value.

### *Dimensionality Reduction*

In order to easily evaluate how well each of the dimensionality reduction algorithms performed, I had to derive some way of consistently testing all of them. Therefore, for it's quick and surprisingly respectable performance, I chose to use a K-Nearest Neighbor algorithm on the transformed features to see which algorithms helped contribute to supervised classification the best. I held the number of neighbors, k, to be 10, as it was my best performing parameter after our Supervised Learning case study.

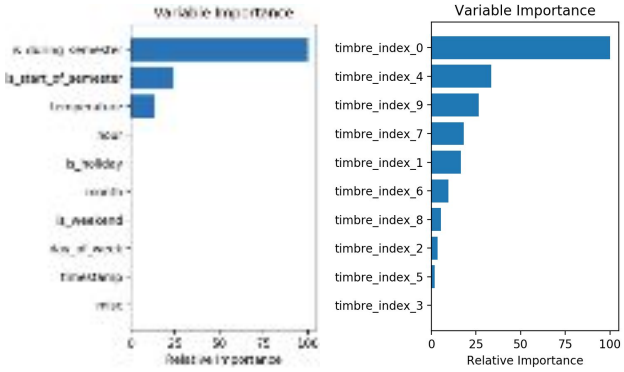
### *Choosing K Centers/Components*

To choose K, I simply ran grid search on the algorithms with different configurations of K, ensuring that I maintained at least 2 features because no single attribute could completely capture the labelings in both datasets whatsoever. Of course, I capped my K right below the number of features for both datasets to ensure that I did not exceed the maximum, but also to see if every feature was truly needed.

[1] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.adjusted_mutual_info_score.html

## Visualization of Clusters

Whenever possible, I wanted to visualize clusters in some way to give some graphic intuition as to how they were forming. Therefore, I chose to use a three dimensional scatter plot where the color of the point indicates which class it belongs to. These colors are part of an overarching color scale to help understand which clusters are somewhat related to one another, and overall helps contribute to the identifiable scatterplot clusterings below. Perhaps the most interesting part of this process was choosing the best three axes to plot each data point along. In either case, I had to choose 3 out of about 10 features, and this resulted in having to repeatedly evaluate which features really mattered. To do so, I trained a decision tree classifier on both datasets and then evaluated feature importance via where and how each attribute was split upon. Those feature importances are visualized here for the Gym Dataset, (below, left-hand) and the Song Dataset (below, right-hand). I used this largely, along with some trial and error for modifying the third axis, to govern which attributes I would plot each dataset's corresponding scatter plots according to.

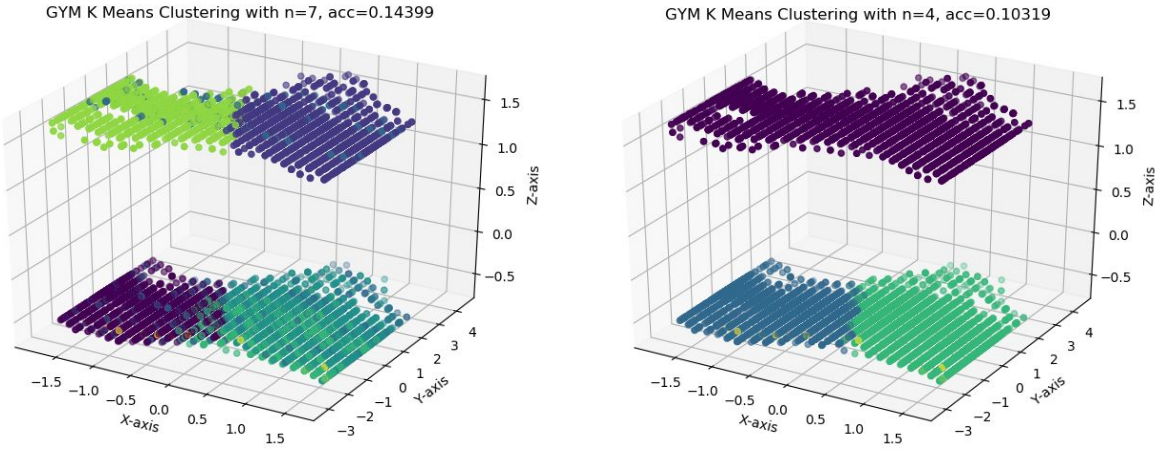


# Clustering

## 1. K-MEANS

### Gym Dataset

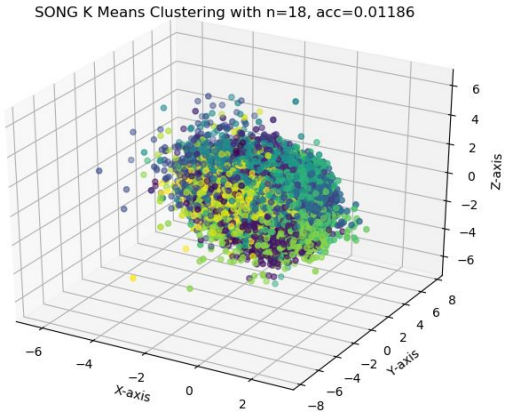
With the gym dataset, I immediately noticed that there was a fairly well formed clustering with 2 different parameter inputs. The dataset itself has 8 different classes, so I assumed that the best configuration would come from using k=8 to form 8 individual clusters. However, to my surprise I got better results from k=7 and k=4, which I was not quite sure about:




The visualizations you see above are 3D scatter plots which I mentioned briefly in my introduction- where the X-axis is the hour of the day, the Y-axis is the temperature, and the Z-axis is the is_weekend boolean attribute. (Note: Due to feature scaling as part of my preprocessing stage, the numbers visible on the axes do not correspond with intuitive values for each attribute.) With k=7, we see 4 somewhat distinct clusterings because there is the notion of a color scale. Seeing how the data is inclined to be partitioned into 4 groups, I manually went to identify what each of the groupings represented. To do so I define levels of crowdedness on a 4 point scale where 1 is the most empty and 4 is the most crowded. Starting with the k=7 visualization, I observed that the lime green cluster (weekend morning) was mostly crowdedness level 1, the purple cluster (weekday morning) was crowdedness level 2, the indigo cluster (weekend afternoon/evening) was a crowdedness level of 4, and the teal cluster (weekday afternoon) was a crowdedness level 3. This aligns pretty closely with my own assumptions about when the gym here at Georgia Tech is crowded, where afternoon are on average more crowded than morning times because of most students schedules. To further see whether the idea that our data is naturally better seperated into 4 groups, I attempted a run of K-means with k=4, and that yielded only slightly worse results, and seemingly appears to only really describe 3 clusters, something we will touch upon later.

### Song Dataset

Unlike the Gym Dataset, the Song dataset showed little to no success with being clustered by the K-means algorithm regardless of the k chosen. In the end, the best candidate was setting k to 18 centers, but even then we are not able to identify very confident clusters:

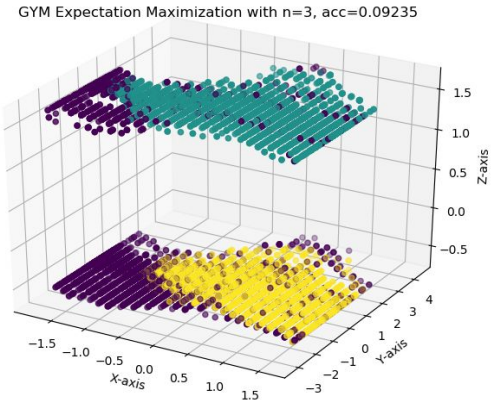



Again, the visualization you see above are 3D scatter plots which I mentioned briefly in my introduction- where in this case the X-axis is 0th index of the timbre vector, the Y-axis is 4th index, and the Z-axis is the 9th index. (Note: Due to feature scaling as part of my preprocessing stage, the numbers visible on the axes do not correspond with intuitive values for each attribute.)

## 2. EXPECTATION MAXIMUM

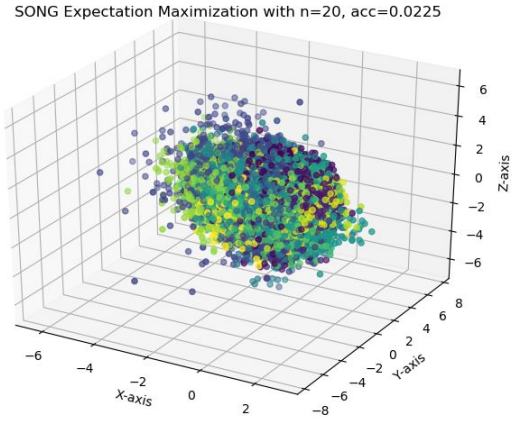
As opposed to K Means, which has a monotonically non-increasing error with each iteration and promise of convergence, the EM backed Gaussian Mixture Model has no promise of convergence due to its ability to explore continuous space. However, this tradeoff of higher complexity also allows it to often be more accurate.

### Gym Dataset




To assess the effectiveness of Expectation Maximization with my datasets, I used a Gaussian Mixture (GM) algorithm trained with Expectation Maximization, where we assume that our data could have been generated by gaussians in n-space. We see that the EM algorithm performs better on the unaltered Gym Dataset, and in face begins to explain the 3 clusters we saw when using K-Means with k=4; perhaps K-Means, due to its requirement to make discrete choices, adds an extra cluster to help account for a boundary decision while the GM algorithm optimized by Expectation Maximization is able to inherently describe uncertainty between clusters. This fundamentally helps explain why EM is usually the better performer.

### *Song Dataset*

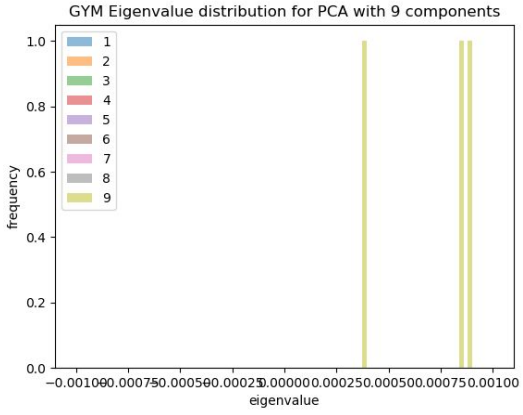


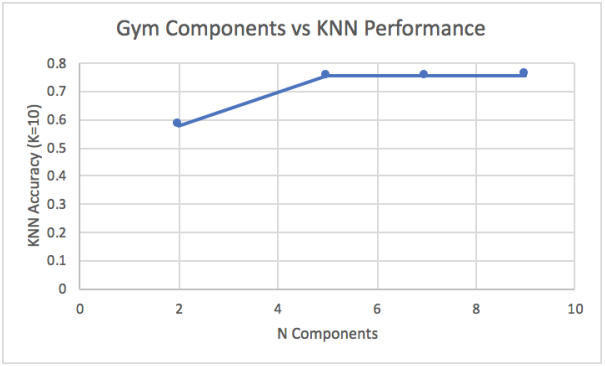

As seen previously with K Means, the song dataset performed quite poorly as well. When it comes to having many classes with varying corresponding feature vectors in the case of my converted regression problem, it is very difficult to cluster with any reasonable confidence and showing how terribly unsuited these problems are for unsupervised learning. Nevertheless, EM did perform relatively much better than K Means, achieving a 9% accuracy on the Gym Dataset and a 2% accuracy on the Song Dataset while K Means achieved a staggering 1.4% accuracy on the Gym Dataset and 1.1% accuracy on the Song Dataset.

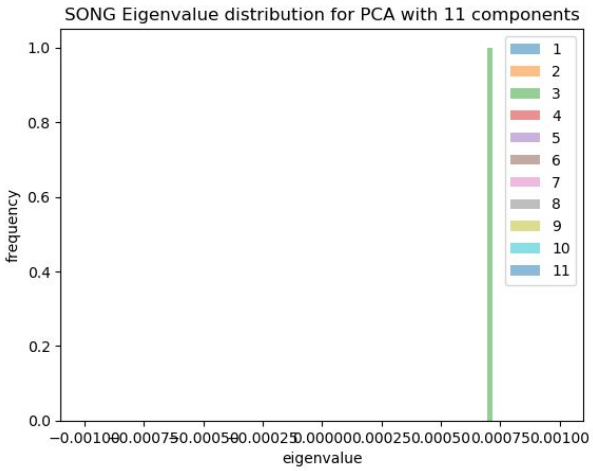
# Dimensionality Reduction

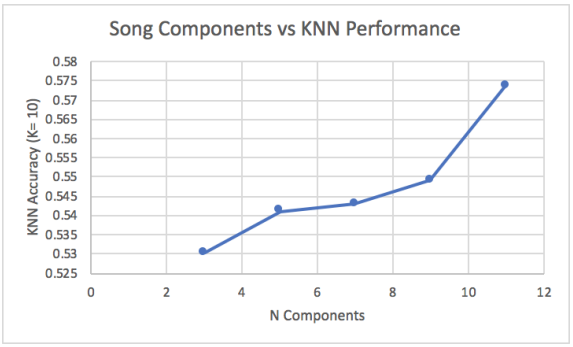
## 3. Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a linear dimensionality reduction using the linear algebra matrix operation of Singular Value Decomposition to project input features into a lower dimensional space. The process involves extrapolating eigenvectors of the covariance matrix and choosing them greedily to maximize information learned from the data.



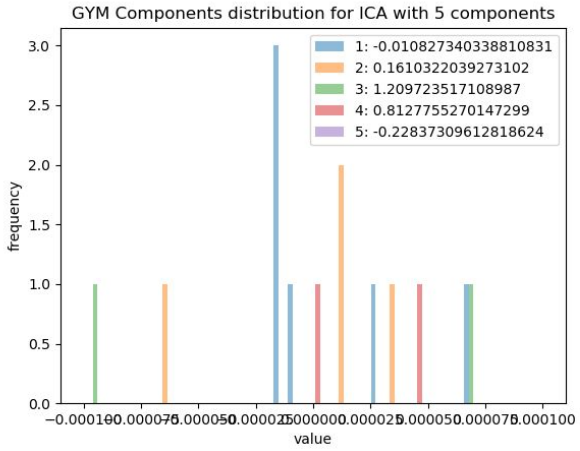


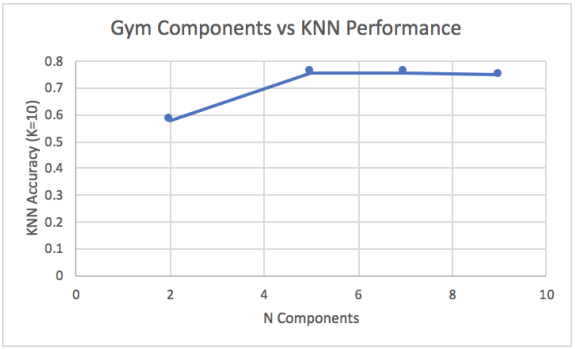



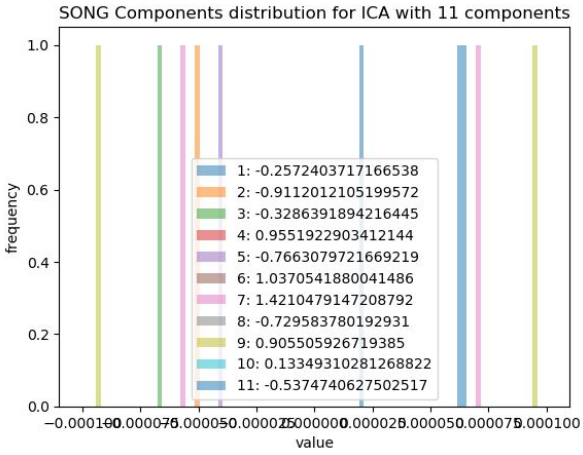

In regards to the amount of information learned with each additional component, we see that the Gym dataset, although technically monotonically increasing, largely plateaus after 5 components. Each additional component can be seen to add diminishing returns thereafter. However, the Song dataset shows a very different situation where, although also monotonically increasing, shows that each added component adds notable information necessary to the classification of each instance.

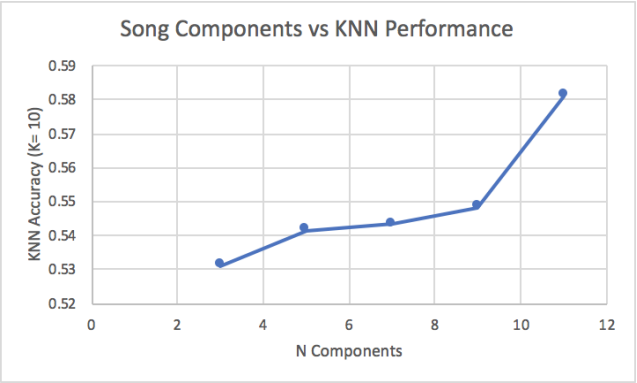
## 4. Independent Component Analysis (ICA)

Independent Component Analysis aims to maximize the statistical independence of the subcomponents of the signal that is a culmination of all the features present in the dataset. While normally used for separating superimposed signals, ICA can be modified with whitening, which is the transformation of random variable vectors into a set of new vectors whose covariance is the identity matrix, a characteristic of normal signals.



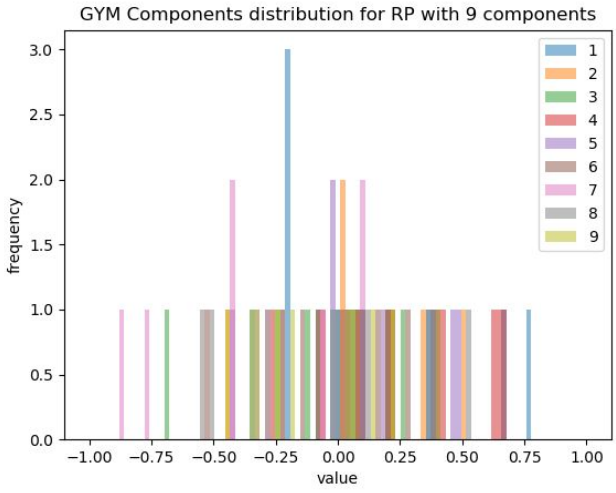


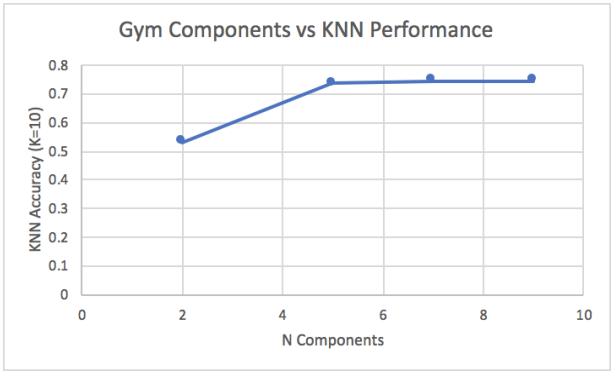



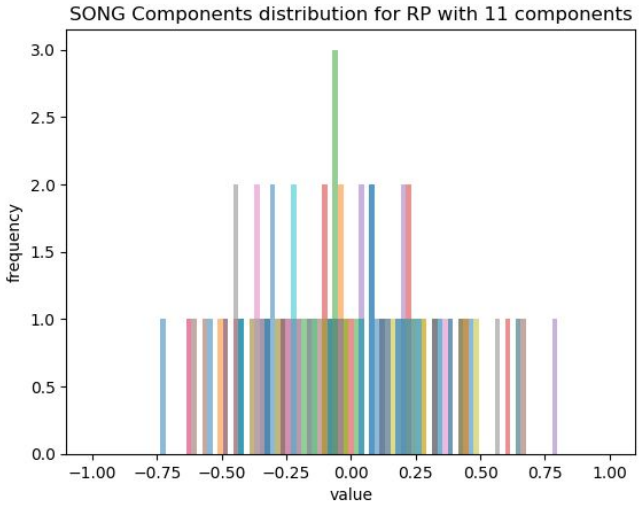

Once again, we see that the Gym Dataset shows notable improvement until having added five components using ICA while the Song Dataset consistently shows improvement in information representation with incorporating every single attribute of the given timbre vector.

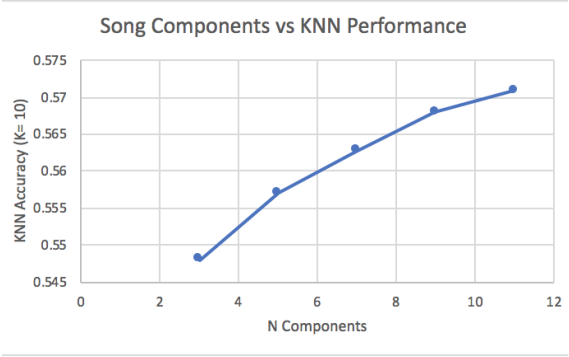
## 5. RANDOMIZED PROJECTIONS

Randomized Projections (RP) takes a random approach to calculating projections, thus offering benefits in much better speed, lesser data dependencies, and exploration of non-linear subspaces as opposed to the methods above. Due to the uniformly random projections generated, the distributions gradually begin to approach a Gaussian with more and more components. This results in a tradeoff where we are required to run many many iterations in order to get promising results, and results are not deterministic simply due to this random nature.



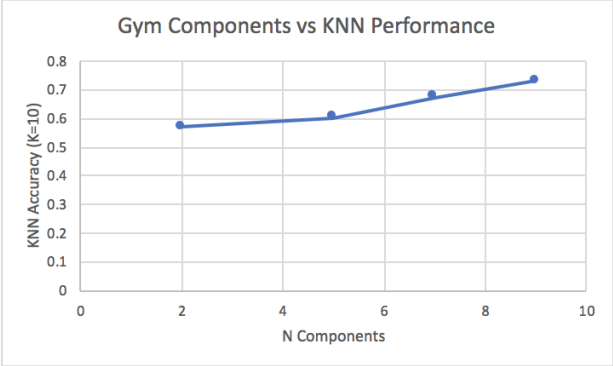



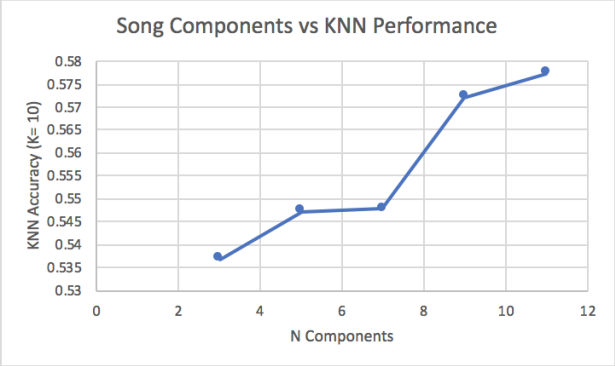



## 6. SELECT K BEST

The Select K Best algorithm is a much more simple approach in that it reduces input features by subsampling them one by one based on how much information each one provides using univariate feature selection.
Because there is no heavy mathematically transformation of the data, it often ran faster. In order to do so, it needs a univariate scoring function and p-values (a statistical indication of significance). For my experiments, I chose to use the F-Regression scoring function, which is more suited to my regressional problems and uses measures of correlation which is then converted into an F-Score and a P-Value.

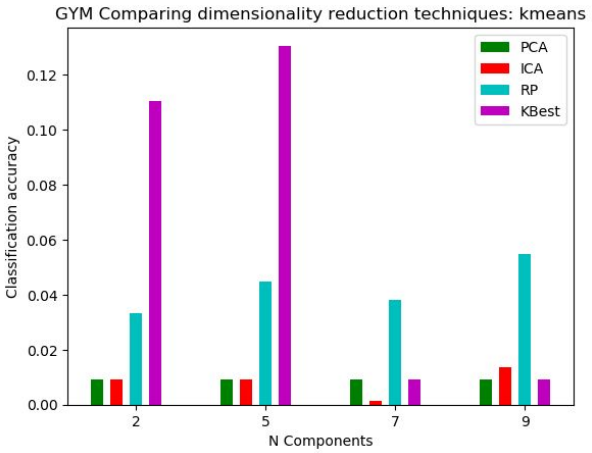


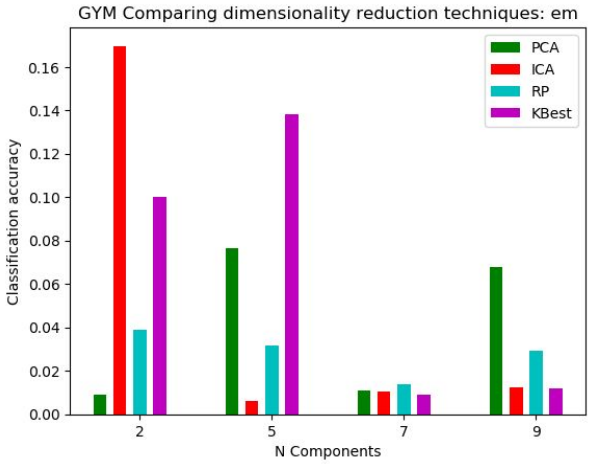



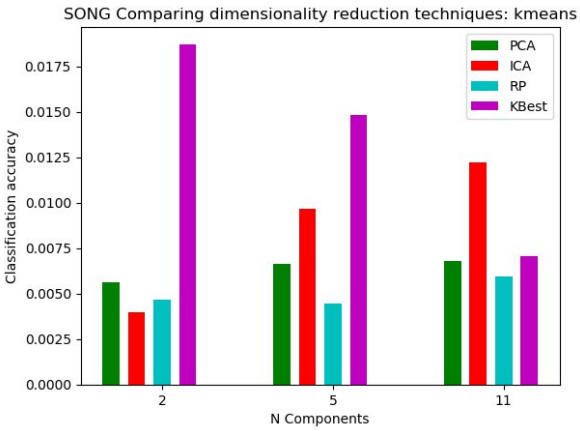

Here we see that

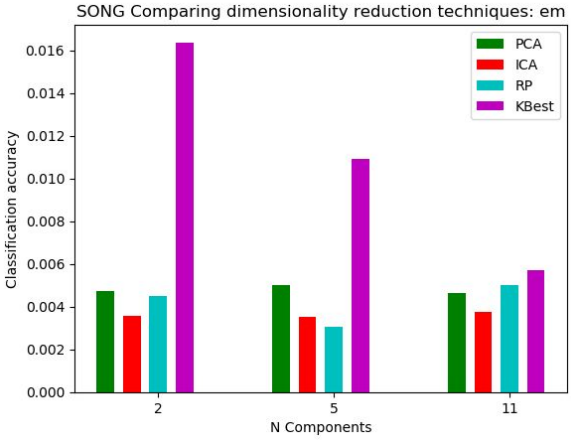
# Dimensionality Reduction Applications

## Dimensionality Reduction + Clustering



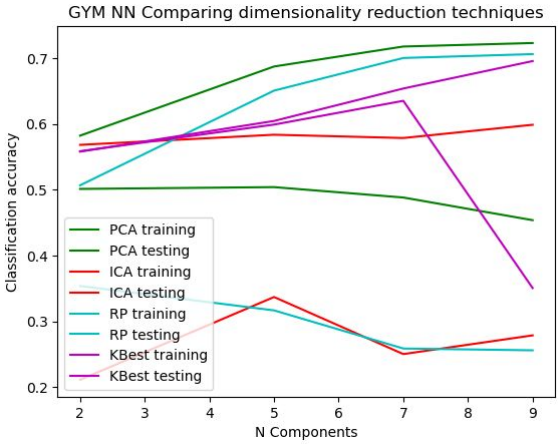



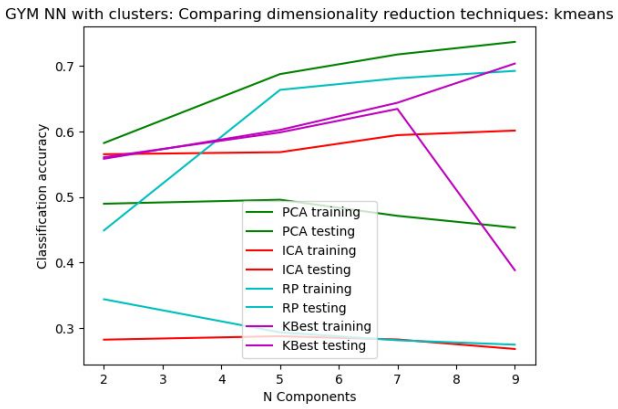

We see that for K-Means, despite being reasonably incapable of tackling this problem inherently, features reduced by Select K Best were consistently much better the unsupervised learning algorithm. I assume that because we have a situation where a many different types of conditions for the gym and genres of music for a song entail the same respective labels, so understanding which attributes to really focus on with a supervised dimensionality reduction helps immensely. On the other hand, it did not perform as well when compared to ICA when acting as a 2-component preprocessing step for an EM backed Gaussian Mixture Model. In the case of EM on the Gym Dataset, we see interesting behavior with the unsupervised accuracy where less components results in a higher accuracy; a little above 16% as opposed to the 9% achieved without any dimensionality reduction. While still low, this is an impressive increase in accuracy relatively speaking with such few components.

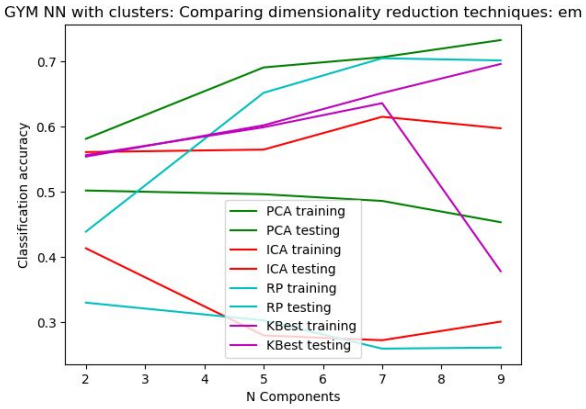
## Dimensionality Reduction + Neural Net (GYM)




We see that PCA, SelectKBest, and RP are the better candidates for dimensionality reduction for the Gym Dataset when it comes to training with a Neural Network. However, I am surprised that we do not see much of an increase from the previous performance of the same Neural Network without any dimensionality reduction, which yielded an accuracy of 69.04%. This comes as somewhat of a surprise, as NN are known for being good at handling noisiness and potentially corrupted data, which is most likely reduced by Dimensionality reduction yet yields no notable improvement. This could imply that the data itself is quite reliable and precise, which would make sense since many attributes in the Gym dataset are simply either boolean attributes that are quite easily measured or counts of people which even if off by a little bit can still teach valuable insight.

## Dimensionality Reduction + Clustering Features + Neural Net (GYM)




When using unsupervised learning to generate new attributes to add onto the reduced dimensions of the dataset, we did not see all that much difference in the performance. However, it does seem that on average, all the neural nets with each variation of dimensionality reduction did a tiny bit better with EM clusters as opposed to K Means clusters, which we can attribute to how EM is usually a better performer. When compared to without clustering, we see that some of the bias originally in our model disappeared, which shows the benefit of the added feature!

# Conclusion

To little surprise, my hypothesis regarding the poor performance of unsupervised learning on my datasets turned out to be true. Across the board, I saw mostly sub 10% accuracies and this is simply an inherent result of the algorithms assumption that instances with the same label are automatically close together in the feature space. However, the uses of dimensionality reduction are most definitely evident in this case study. Across the board, the neural nets ran at least 1.3 to 1.5x faster than they did before, without much loss in accuracy. It was interesting to see how dominant a Supervised Dimensionality Reduction Algorithm like SelectKBest outperformed the other Unsupervised Dimensionality Reduction Algorithms, making it clear how important it was to learn from the labelings of both my datasets.

(References in Readme)