Neural Networks and Clustering for Classification Problems

Zavier Andrianarivo[1]

[1]**zoa215@nyu.edu - N14173755**

ABSTRACT

This short report provides a detailed analysis of the performance of a neural network model used to classify song genres from the Spotify API. The report goes in detail on preprocessing, clustering algorithms used, and the models created to assist with the classification task. In short, the base model achieved an accuracy score of 60%, the model with KMeans cluster labels achieved an accuracy score of 14%, the GMM labels produced an accuracy score of 14%, and running PCA with the labels provided yielded a 35% accuracy.

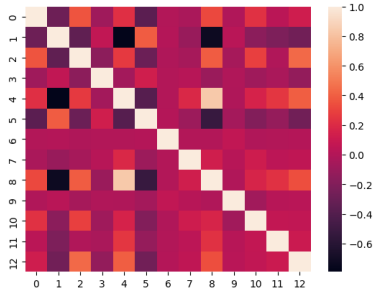
Keywords: classification, networks, supervised, unsupervised

INTRODUCTION

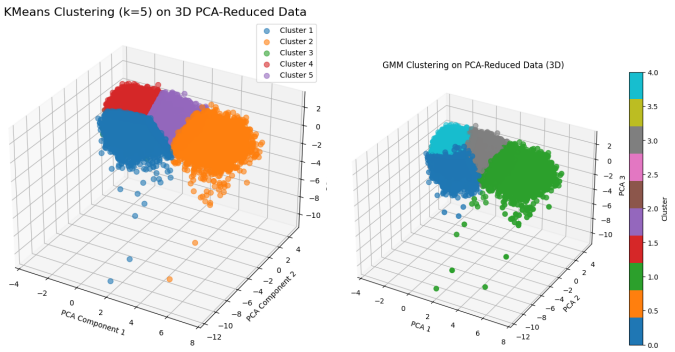
Given a classification problem, and a dataset provided by the Spotify API, a series of preprocessing as well as deep models were implemented to assist in best predicting the genre of a song given the features in the dataset. A number of preprocessing techniques were applied to clean and normalize the data, ensuring that the models could perform effectively. The project explores multiple supervised and unsupervised machine learning algorithms, compares their performance, and analyzes which features contribute most significantly to genre prediction. More information on the preprocessing techniques and models can be found in later sections of this report.

DATASET AND PREPROCESSING

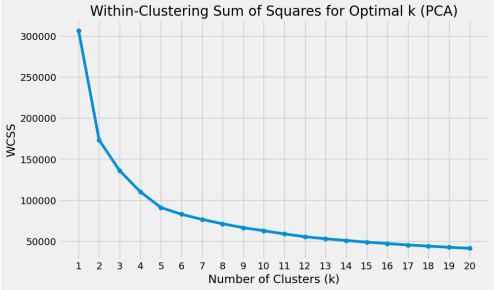
The dataset provided was retrieved from via the Spotify API (as noted on our capstone project sheet). There was a lot done and computed that went into preprocessing this dataset. For preprocessing, I evaluated the data through computing correlation coefficients to see if any linear dependencies existed. And it turns out there were a few features that were very highly dependent on each other - those being *energy* and *loudness*. With the 13 features, after computing the covariance matrix though and viewing the heatmap, we could see that some sort of dimensionality reduction was necessary.



To perform dimensionality reduction, the dataset was normalized, giving every feature a mean of 0 and a standard deviation of 1. Features that were dropped included *artist_name*, *song_name*, *instance_id*, and *obtained_date*. These features were deemed not relevant to the model performance. However, one thought I am having now is how certain artists could be clustered into one genre - which is something I plan to explore more this summer with this project. Below are the two clustering algorithms being ran on PCA-reduced data:



We can see our elbow gives us an optimal *k*-value of 5 here, but I think you can argue that 3 would be optimal as well - if not, even more:

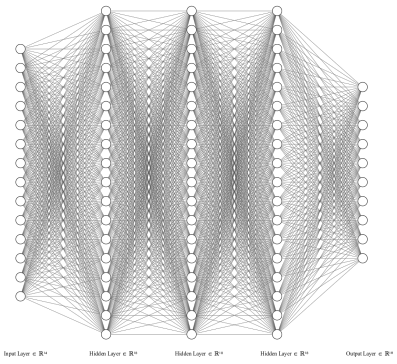


Seeing the clusters, we can see that there is a lot of overlap between them. This indicates that more information was probably lost than we wanted. We can see that in the table showing the the model's performance on the different datasets it was trained on.

MODELS AND ALGORITHMS

After performing dimensionality reduction with PCA on the dataset, KMeans with Lloyd's Algorithm and Gaussian Mixture Models were performed to create clusters within the reduced dataset. The labels from these clustering algorithms were imputed to the original dataset and trained the neural network on the data.

For the model implemented, I implemented a deep neural network. The design architecture is as follows:



Though a simplification of the actual architecture, the model consists of *n*-features as the size of the input, 3 hidden layers consisting of 64 dense nodes/layer, with the output layer consisting of 10 nodes, 10 classes - the 10 genres we are trying to predict. The activation functions used were ReLU() for all of the forward-pass layers until our output, using Softmax() as our activation function to turn the output into

a probability distribution of the most-probable label. Computing loss was done with cross-entropy and optimization was done with Adam.

A neural network was chosen to be used for this dataset because while evaluating the visuals of the reduced datasets, the clusters were really dense and contained a lot of overlap. In addition to losing more data than we wanted, we also can't classify this dataset with a soft-margin based classifier – as there is no real linearly separable set of clusters that would allow that. Therefore, it made sense to use a neural network in this situation.

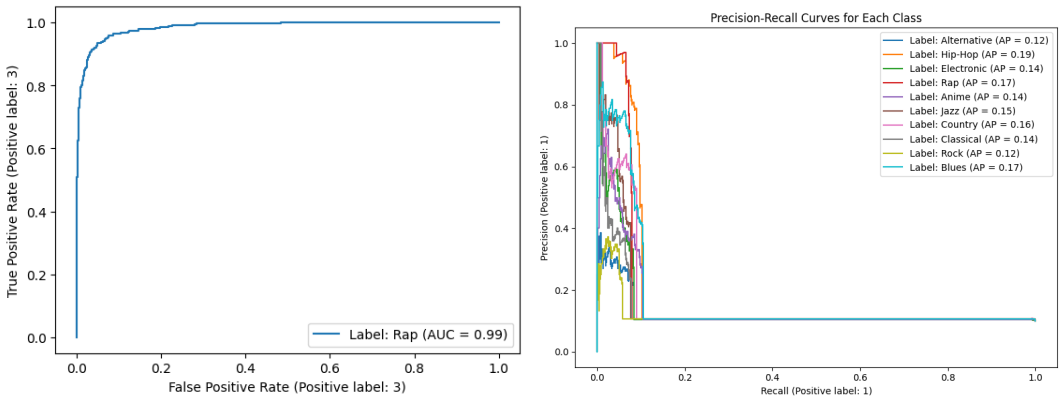
Figures and Tables

We can compare some of the metrics of the models below:

Model	Accuracy	Precision	Recall	F1-Score	MCC
Base Model (no cluster imputation)	0.58	0.59	0.58	0.58	0.53
PCA w/ Labels	0.35	0.32	0.35	0.32	0.28
KMeans Imputation	0.14	0.39	0.14	0.09	0.08
GMM Imputation	0.14	0.50	0.14	0.10	0.11

Table 1. Metrics of the model performance, which can also be seen in the jupyter notebook

Here, we can see look at a few of the graphs of AUROC and AUPRC for the models performance.



OBSERVATIONS

Looking at the metrics, we can see that the model produced a very high AUC with just the base dataset, with an AUC of **0.99** - on the *rap* genre with an average AUC of **0.94**. Here, we can see that the GMM-imputed model produced an average AUC of **0.148**. However, the highest AUCs for all models were under the *racp* genre, indicating the model performs really nicely on classifying the rap genres.

Something that came unexpected to me was the lack of correlation between certain features such as popularity and dancability, energy and tempo, etc. Treating the clusters as a method of finding any non-linear patterns in the data by imputing the cluster labels to the respective feature is something I thought would work, but it only led to really bad performance on the dataset. Given some of the data was dependent, I think what determined the success of the base model was the lack of dependence - making us lose a lot of information during dimensionality reduction.

CONCLUSION

To conclude, the simpler the better for this project. Getting the best results for classification through the least-preprocessed dataset was not expected. However, a lot more could be done on this dataset. For example, for data imputation, I would like to handle data imputation using variational autoencoders to synthetically create data data (such as the tempo of a song). To have to deal with missing data by dropping it does not feel authentic to the true dataset. Something else I would like to explore is the correlation between artist name and genre. Clusters could group certain artists together to find the relation between artists and genres. In all, a learning experience, with a lot of potential for more exploring.