Classification Project Report

yl7789, Yuheng Lu, May 2022

Data Processing

Problems:

1. There are missing values in the dataset.
2. There are linguistic features that are not useful for this project.
3. "accousticness" feature is not normally distributed.
4. "key" feature is in string format, which needs to be transformed into numerical data.
5. "music_genre," the target, is in categorical format, which is not model-friendly and needs to be transformed into numerical data.
6. "obtained_date" feature is in categorical format, which needs to be transformed to dummy variables.

Solutions:

The bulletin number of the following solutions are corresponding to the bulletin number of problems; that is to say, solution 1 is for problem 1, etc.

1. First, there are five rows containing missing values, NaN. In total, there are 50005 rows. By simply excluding those five rows, the remaining 50000 rows still satisfy the requirement that for each music genre, there should be 5000 rows and a 4500:500 train-test split. Thus, those five rows containing NaN are directly dropped. Second, there are missing values represented as '?' in tempo column which should be considered as missing values. However, if rows containing '?' in tempo are simply dropped, the requirement of 5000 rows and 4500:500 train-test split could not be satisfied. Therefore, these values have to be filled in. The fill-in work should be done after train-test split as we do not want any leakage between test set and training set. After train-test splitting, these missing values are filled in by mean values of tempo column. Since there are not too many missing values, I believe this fill-in-na method is solid and will have more positive impact than simply drop them.
2. Linguistic features are not very useful. They are simply dropped. They are instance_id, artist_name, and track_name. artist_name and track_name are obviously linguistic features. instance_id in my opinion is identifiers of songs that has no meanings. It should be considered as linguistic features as well.
3. Since "accousticness" feature is not normally distributed, MinMaxScaler() instead of the conventional StandardScaler() method is used for this column, as MinMaxScaler() will normalize the data and keep the original shape of distribution.
4. Since "key" feature is in string format, from "A" to "G#", they are transformed into 1 to 12 in number, as "A" to "G#" is from low to high. They are used numerically instead of categorically as well, since "A" to "G#" describes the low or high key of a song which could somehow be intepreted numerically.
5. "music_genre" is transformed into numerical data to be model-friendly.
6. "obtained_date" feature is in categorical format. An interesting observation is that there is only one row having '0/4' as value in its 'obtained_date' column. This value is confusing, and making only one value into a dummy variable will provide very minor information to the later on models but consume extra time. Therefore, it is simply replaced by the most common existing value in the "obtained_date" feature. Then, "obtained_date" feature is transformed into dummy variables.

Train, Test Split

Operations:

1. Since for each music genre, a 4500:500 train-test split needs to be performed. I did this seperately for each music genre. That is to say, data frame is divided by different music genres and train-test split is performed 10 times as there are 10 genres. Then, training sets and test sets are concatenated with previous training sets and test sets. This could ensure the overall train-test split is 45000:5000, and for each genre, train-test split is 4500:500.
2. Standardizations, normalizations, and fill-in-na operations would be done after train-test split to prevent any leakage between training set and test set.

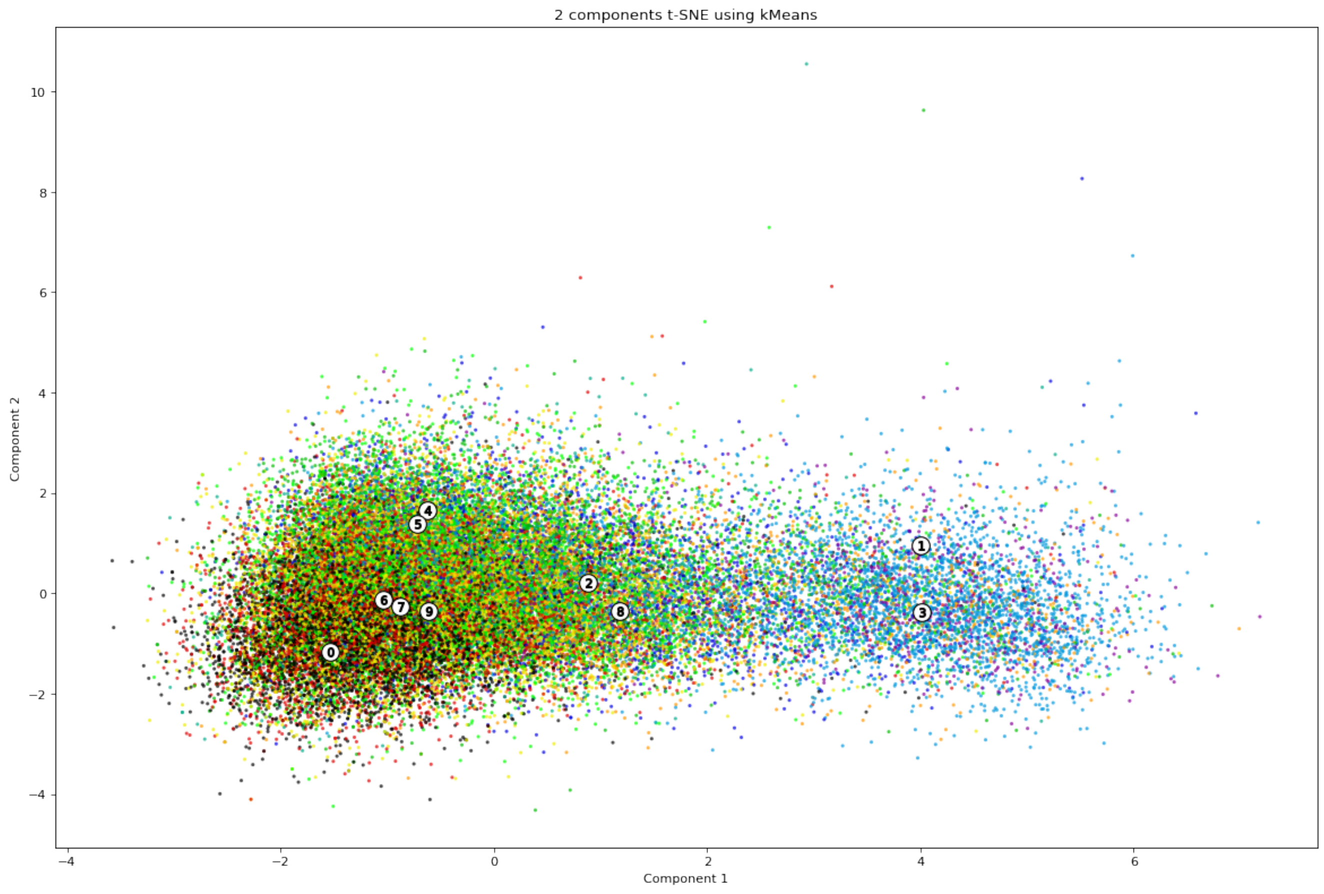
Dimensionality Reduction

Operations:

1. PCA is performed to reduce dimension of the dataset. The top columns that can explain 95% of the dataset are kept.
2. Visualization is done by KMeans clustering using first two componenet of the dimension-reduced dataset.

Obeservations:

1. After PCA, 10 columns left compared to the original 18 columns.
2. KMeans clustering looks like the picture below. The patterns of clusters are not obvious for human. I would say this dataset is not two dimensional sparse dataset. But there are some valuable observations that can somehow differeniate clusters. For example, clusters 1 and 3, clusters 2 and 8, clusters 4 and 5, and the rest of the clusters' centroids are distanced further compared to them pairwisely. But I still will say, plotting the dimensionality result onto a 2D space is not very informative and barely helps. The dimension's reduction from 18 to 10 is much more meaningful and useful compared to the plotting.



Clustering

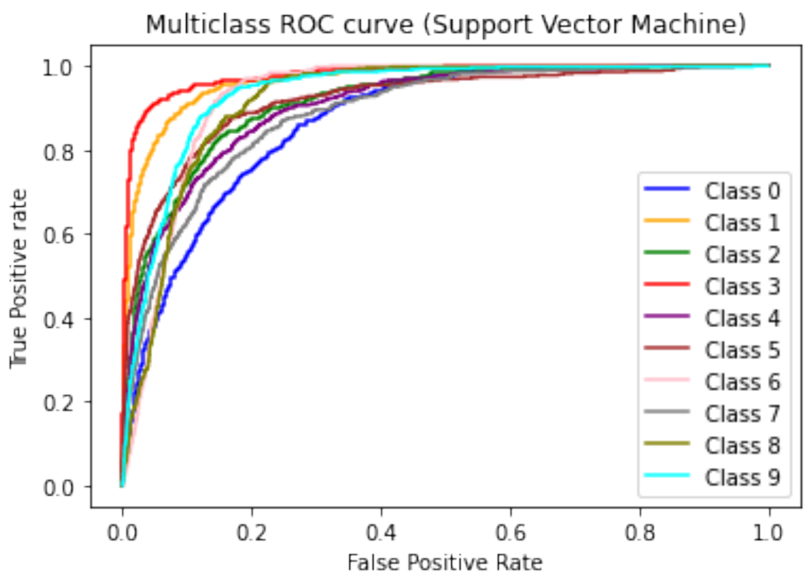
Operations:

1. KMeans has been performed in the previous dimensionality reduction section, and we found it barely helps with plotting. However, KMeans could be used as features to boost the dataset's performance for future models.
2. The results of clustering of KMeans are extracted, as numerical data that which song belongs to which centroid. These numerical data actually tell the categorical property of each row. Therefore, they are transformed to dummy variables and added into the datasets after dimensionality reduction.

Classification

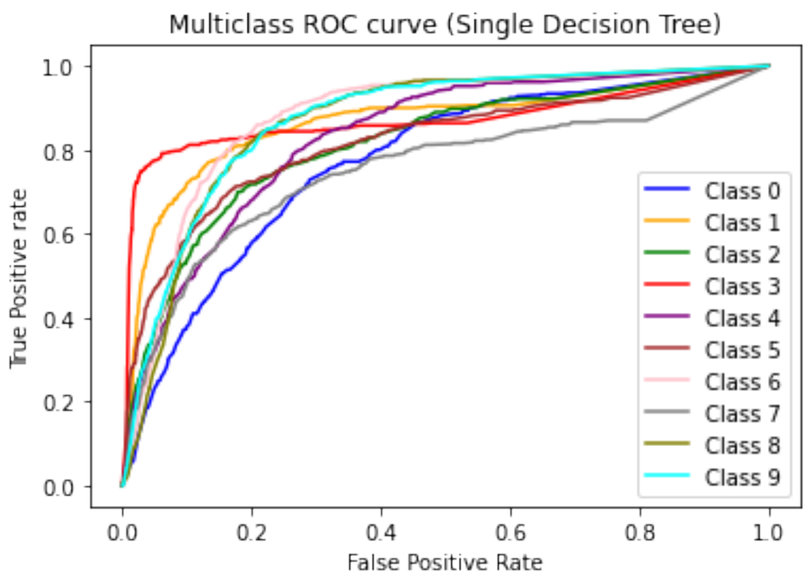
Support Vector Machine:

1. Multi-class support vector machine is built. Not too much hyperparameters need to be tuned, as the AUC turns out pretty good.
2. AUC of Multi-Class Support Vector Machine is: 0.9217242222222225.
3. Its ROC curve looks like the one below. Since this is a multi-class classification. There will n-class lines in the ROC curve plot. Each line represents nth-class vs the rest of classes.



Single Decision Tree:

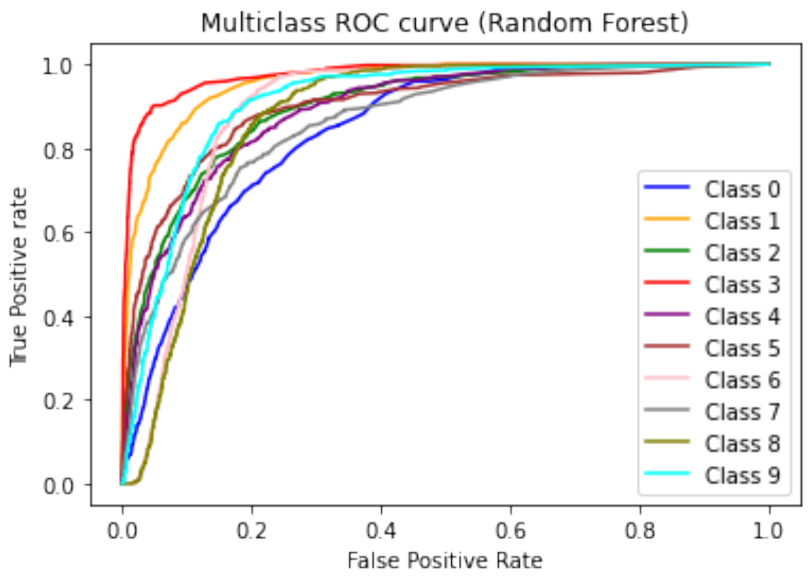
1. Single Decision Tree is built. The max_depth parameter is tuned to an appropraite number instead of letting the tree grows to the end, since two problems might happen: the tree overfits the data and performs badly; no probability of each leaf will be returned.
2. AUC of Single Decision Tree is: 0.8332484888888894.
3. Its ROC curve looks like the one below. Since this is a multi-class classification. There will n-class lines in the ROC curve plot. Each line represents nth-class vs the rest of classes.



Random Forest

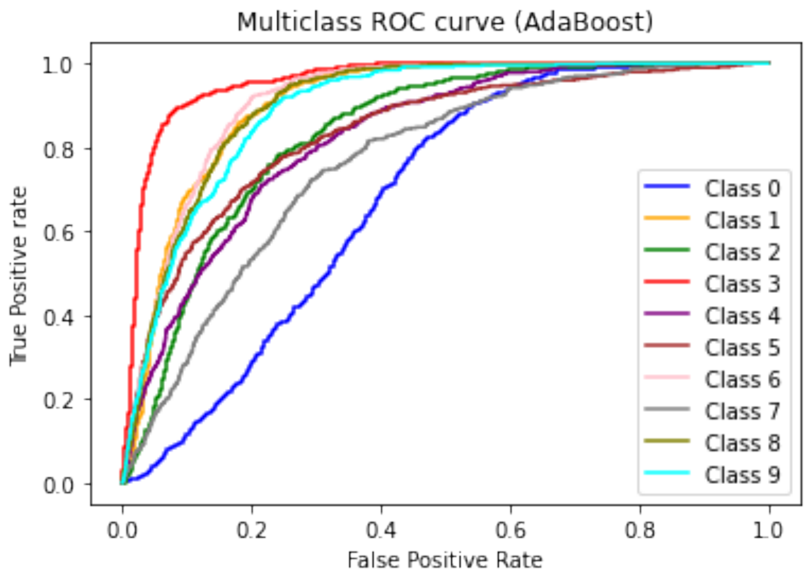
1. Random Forest is built. n_estimators is set to 200. If n_estimators was too high, the model might be overfitted and take an extra long time to be built. If n_estimators was too low, the model might be underfitted and have low AUC or accuracy. Max_features and max_samples are adjusted to make the model finish in a reasonable time and have a acceptable performance.
2. AUC of Random Forest is: 0.900935866666667.

3. Its ROC curve looks like the one below. Since this is a multi-class classification. There will n-class lines in the ROC curve plot. Each line represents nth-class vs the rest of classes.



AdaBoost

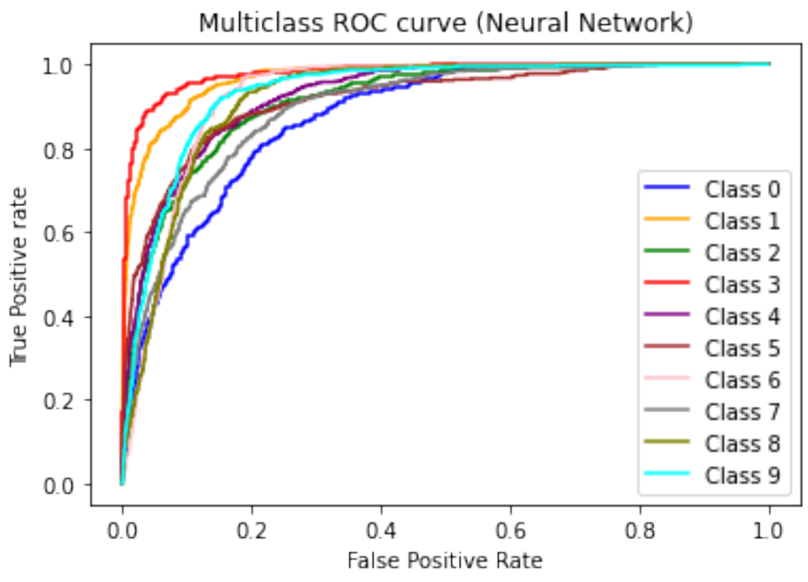
1. AdaBoost is built.
2. AUC of AdaBoost is 0.8508239555555561.
3. Its ROC curve looks like the one below. Since this is a multi-class classification. There will n-class lines in the ROC curve plot. Each line represents nth-class vs the rest of classes.



Neural Network

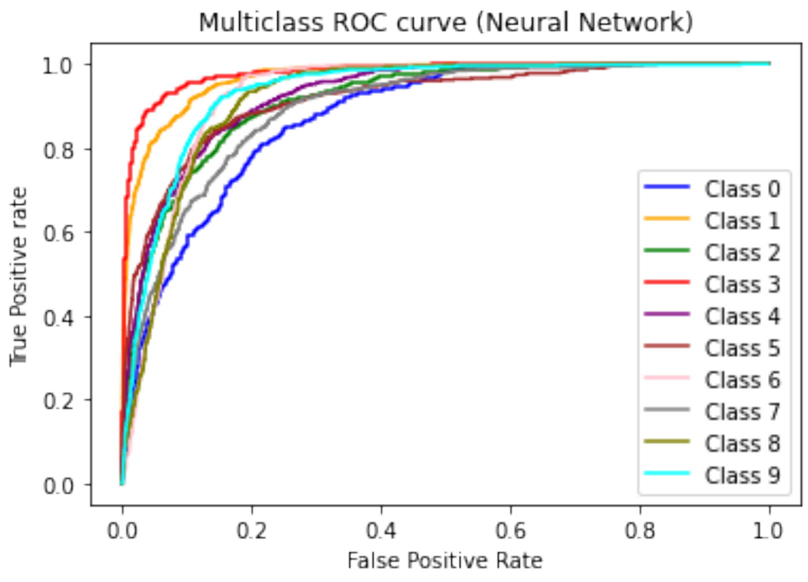
1. Neural Network is built. There is one ReLU hidden layer as ReLU performs better than Sigmoid, Tanh, etc. in this dataset. There is only one hidden layer since by experimenting, adding more hidden layers no matter with what kind of combination with other activation functions makes the model overfit and yield a poor AUC score. The number of nodes in hidden layer is set to 40 as this number makes the model performs the best.
2. AUC of Neural Network is 0.9255717111111116.

3. Its ROC curve looks like the one below. Since this is a multi-class classification. There will n-class lines in the ROC curve plot. Each line represents nth-class vs the rest of classes.



Conclusion

1. Among all the models I tried my best to make, Neural Nework has the best AUC score, 0.9255717111111116.
2. Again, this is the ROC curve of my Neural Network model:



3. I initially tried to build Neural Network models without dimensionality reduction and clustering data. The AUC is around 88%. After applying these two method, the performance boosted around 4%. This could be a big improvement, as it is improved from 88%, a not bad score. However, this improvement, 4%, is not as big as I thought. I would say the main function of dimensionality reduction for this dataset is to make the dataset smaller and speed up the training process. And clustering for this dataset helps mainly with the visualization. However, PCA will lose some information as well which negatively influence the model such as accuracy and precision. We need to evaluate this trade of to choose if do PCA or not.

4. The most important factor underlies the classification success is still model selection and hyperparameter tuning. As an approriate model leads to more than 10% improvement among all of my models.

Extra Credit

1. Among all these lines in different plots, there is one thing in common: Class 3, the red line always has the best score, and Class 0, the blue line always has the worst score. This indicates that for Class 3, it is more distinguishable compared to other genres; for Class 0, it is less distinguishable compared to other genres.
2. Even though the AUC for Neural Network model is pretty good, its accuracy is around 60%, which is not a good score.