Report of Machine Learning Capstone Project

NetID:zs2182 Name:Zhihao SHAN

1 Overview

The task is to classify music genres using a couple of features given, including popularity, acousticness, etc. The dataset is perfectly balanced after dropping rows with missing labels. I first cleaned the dataset, then fed it to the Logistic Regression mode, Adaboost model, and neural network model. Moreover, I dimension-reducted the variables to 2 dimensions using t-SNE, to 2, 8, and 16 dimensions using PCA, and fed each of them to the best model I got in the previous step. To get the most important factor, I excluded each column of X and get AUROC for each class & overall.

2 Data Preprocessing

The first step is to read data from the file and check if there are missing values. There are null values in our label 'music_genre' and I dropped these rolls. Though there are no null values in other variables, there are '?' values in the column 'tempo' and -1 in the column 'duration_ms', which are invalid. I grouped the data using our label and used the mean value of tempo (or duration) within the group to replace invalid values.

We have two variables whose datatype is string, key and mode. For key, though it appears to have an alphabetical sequence, from the perspective of music theory, A key and B key are not more similar than A key and D key. In fact, A key and D key are more similar. As a result, I treated 'key' as a categorical variable and one-hot encoded it. For mode, it is clear that there are only two values, minor and major. It is for sure categorical and therefore I also one-hot encoded it.

I normalized other continuous variables to make the data on the same scale. Since there are 50000 rows of data, it is time-consuming to do it everytime I run the program, so I store the cleaned data in csv format. I also included a histogram for each variable to check their distribution.

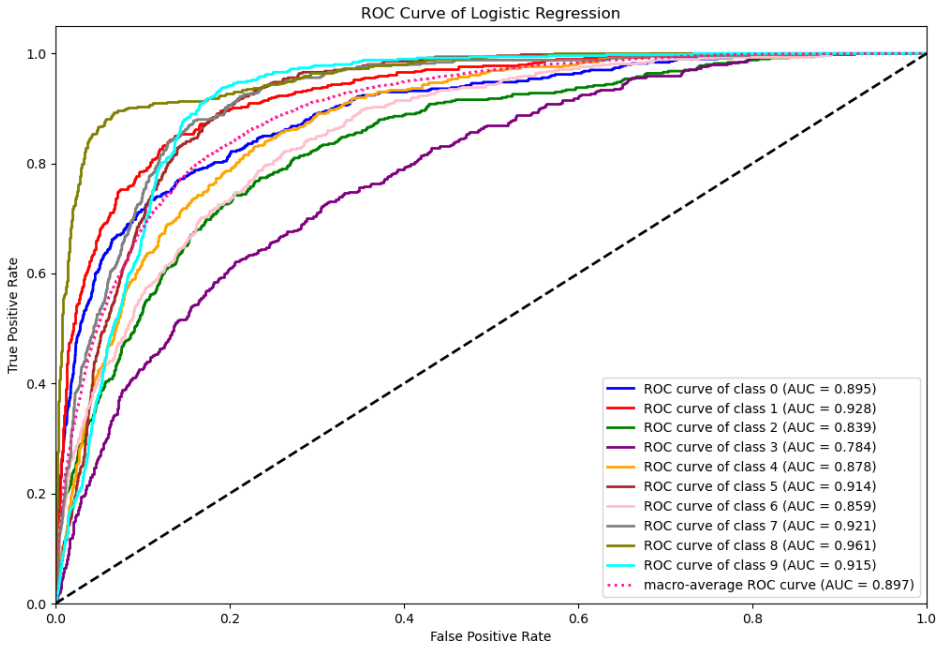
The final step is to do a train-test split. I set the test size as 5000 as required, and stratified with 'music_genre' to make all sets balanced.

3 Logistic Regression Model

Logistic Regression has a straightforward loss function and is fast to train. The way to do multi-class logistic regression is to do a binary logistic regression for each class and combine the results. Fortunately, logistic regression in scikit-learn can automatically do his process.

I fit the multi-class logistic regression model to the training data and derive accuracy and roc curve using test data. The accuracy score is 50.82%, and the overall AUROC is 0.897. The detailed ROC curve and AUROC for each class is in the figure below.

I also tried Adaboost, it can achieve an accuracy of 56.67% and AUROC of 0.922 using a base estimator decision tree with depth 10 and is much slower to train. In fact, it is even slower than training the following neural network for 20 epochs.



ROC Curve for Logistic Regression Model

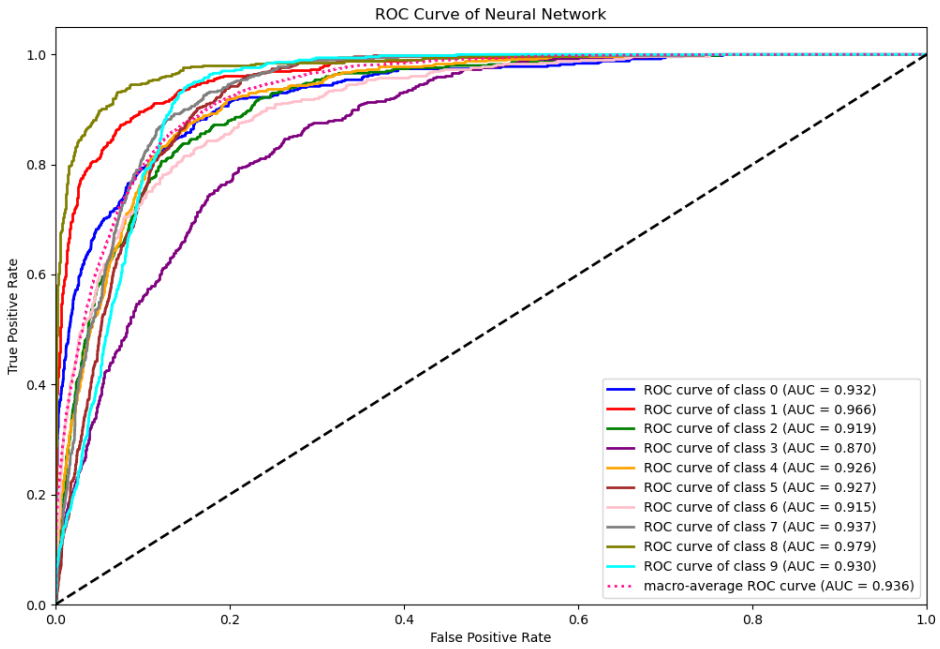
4 Neural Network Model

I tried several structures of feedforward neural networks, including different activation functions (sigmoid, tanh, and relu) and different numbers of hidden layers (1, 2, and 3). I also included drop-out to decrease the probability of overfitting for models with multiple hidden

layers. Using 2 hidden layers and relu as the activation function gives the best result, with an accuracy of 57.84 and an overall AUROC of 0.936. The detailed AUROC curve is below.

The training process also matters. Using Adam as the optimizer gives far better results than SGD and Adagrad. It might be caused by the large number of local minimums of the loss function. I used 20 epochs of training. When plotting the loss against epochs, there are lots of spikes and not always a trend to converge, but using 20 epochs gives very close answers when trying it multiple times.

This model gives the best AUROC I can get, so I used it to test the importance of each factor. I use a loop to exclude each column of X, train the neural network, and evaluate its performance. For the overall AUROC, the most important factor is 'popularity'. Without this variable, the overall AUROC would be 0.883.



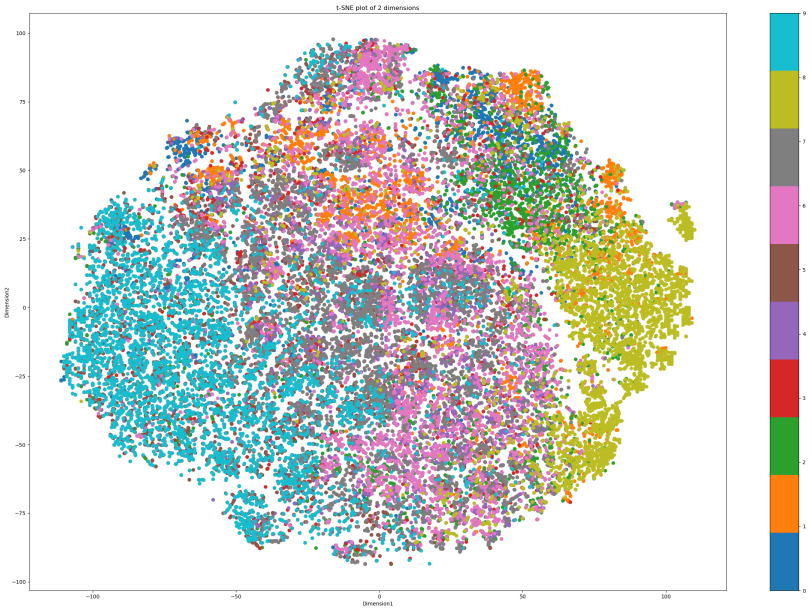
ROC Curve for Neural Network Model

5 Dimension Reduction

I used 2 dimension reduction techniques, PCA for linear and t-SNE for non-linear. I first did t-SNE since it may solve the crowding problem. The result of t-SNE is in the figure below. The light blue dots are for hip-hop and the yellow dots are for classical music. These two genres seem to be classifiable after the dimension reduction. All the other classes are crowded together. The reason might be that the variables are not Gaussian distributed, and

not even t-distributed. I then fed the dimension-reduced data to the neural network model. As I expected, the performance is much worse than the original data, with an accuracy score of around 40. I also did a k-Means clustering to compare its results to the label, it seems the results make no sense.

For PCA, I decided to set ‘n_components‘ to 2, 8, and 16 dimensions and feed them to the neural network model. They all perform worse than the original data. As a result, dimension reduction might not be useful in this situation.

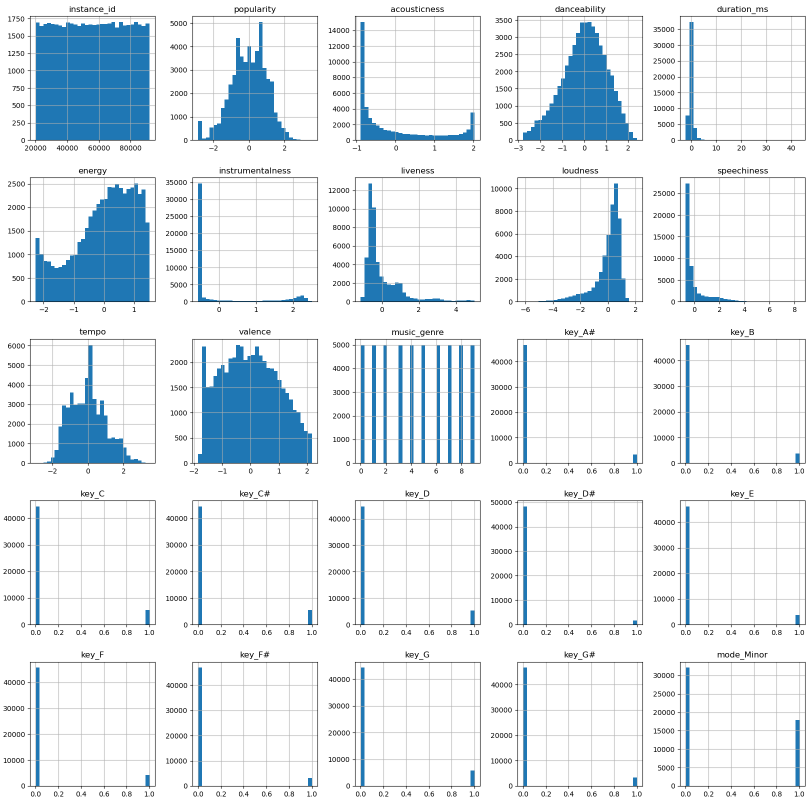


ROC Curve for Neural Network Model

6 Extra Findings

I tested the importance of each factor on identifying each class. For all classes except ’classical music’, the most important factor is ’popularity’. For ’classical music’, the most important factor is ’speechiness’, which is quite reasonable since most classical music does not have lyrics.

For the t-SNE plot, the x-axis in the t-SNE plot might be how melodic the music is since hip-hop mostly locates in the left half of the plot and classical music locates on the right half. The y-axis might be popularity, as all classes have some points up above, and some points down below.



Histogram for each variable