# Music Genre Classification using Machine Learning based on Audio Features and Lyrics

Xuhui Wang

November 2020

## Introduction

In this project, firstly, I introduce the performance comparisons between different methods for pitch extraction, including zero-crossing, magnitude spectrum, autocorrelation, AMDF, Yin and PYin. I also introduce the timebral feature extraction method using spectral centroid and give the visualization of the different musical genres in 2D using the spectral centroid representation. The evaluations show that spectral centroid is a good feature used for music genre classification tasks based on machine learning methods. Then I respectively introduce the supervised and unsupervised machine learning methods used for music genre classification tasks based on audio features and lyrics and experiment these methods on different dataset. The classification results show that these methods work well with the accuracy up to 87% for audio feature and 70% for lyrics, for this challenging task.

## Pitch Extraction

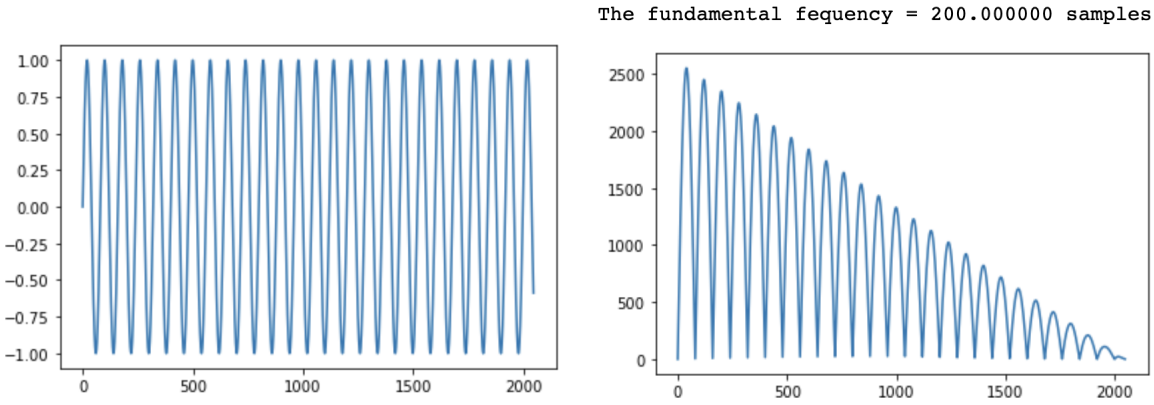
### Methods

(Note that, in this report, the wordings that describe the questions, algorithms and dataset are all from the course material by Dr. Shier and I will give the appropriate citation at the end of each corresponding paragraph. On the other hand, all figures, methods implementation, statistics, analysis, explanations and conclusions are all from my personal work.) In this part, we explore the use of the many important methods for pitch extraction starting from average magnitude difference function (AMDF). The average magnitude difference function is defined as

$$AMDF(m) = \sum_{n=0}^{N=1} |x[n] - x[n+m]|$$

where $N$ denotes the size of the input frame and $m$ denotes the lag number (we could compute this for multiple lag values $m \in 0, 1, 2, 3, 4, N-1$). The pitch is found as the location of the first valley in the AMDF function. The AMDF function takes in an audio signal and the sampling rate, and return the fundamental frequency estimation in Hertz. [1]

Firstly we check that the AMDF function works correctly using as input a sinusoidal signal with a frequency of 200Hz at a sampling rate of 16kHz using a frame size of 2048 and checking the output of your function as shown in Figure 1, which shows that the AMDF works perfectly.




(a) original sinusoidal

(b) AMDF result of the original sinusoidal signal

Figure 1: Input and Smoothed Meshes

Here we use the Yin and PYin pitch estimators to create a pitch track for an existing audio file, of which the curve plot is shown in Figure 2. We plot the pitch track and label the y-axis with frequency in hertz and the x-axis with time in seconds for better visualization in Figure 3. Specifically, we use a frame size of 1024 and a hop-size of 512.

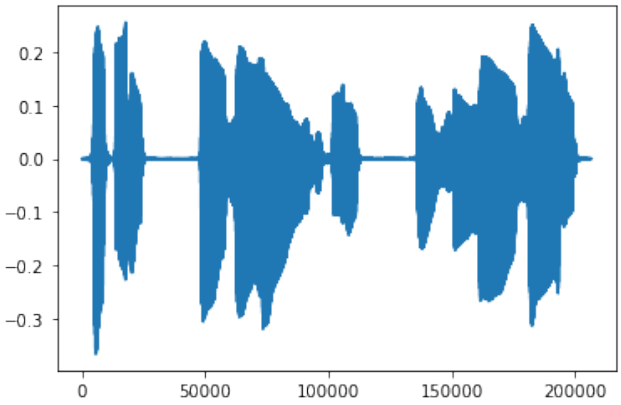
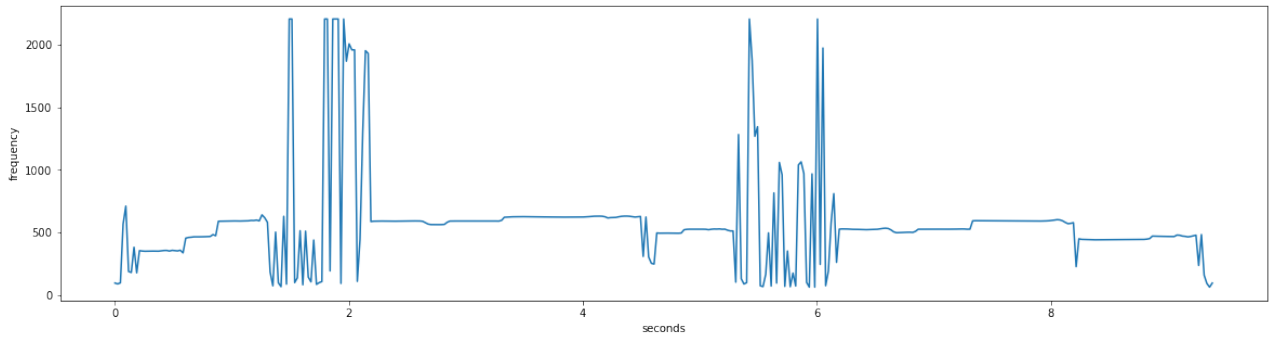
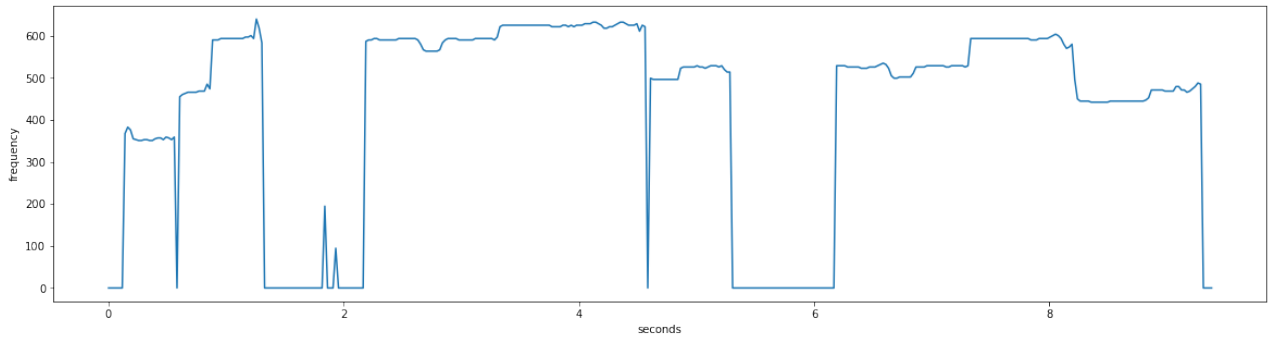



Figure 2: sinusoidal signal of the sax audio




(a) Yin



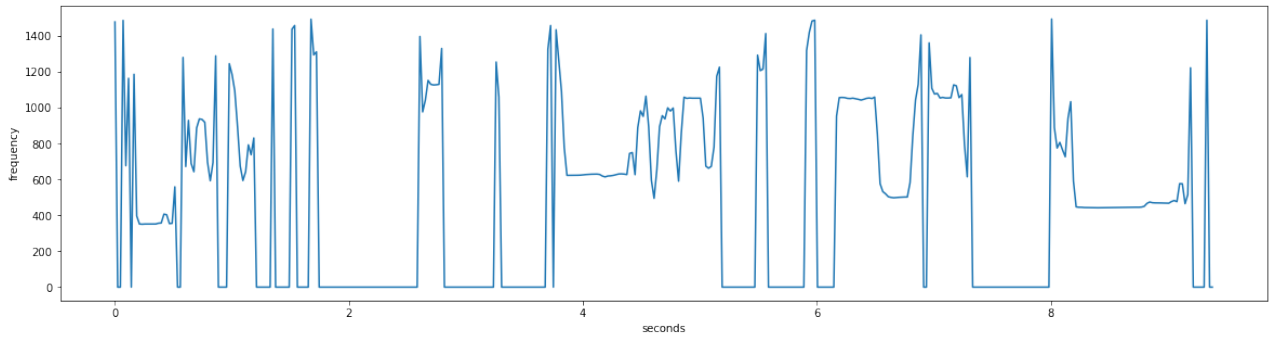

(b) Pyin

Figure 3: pitch track

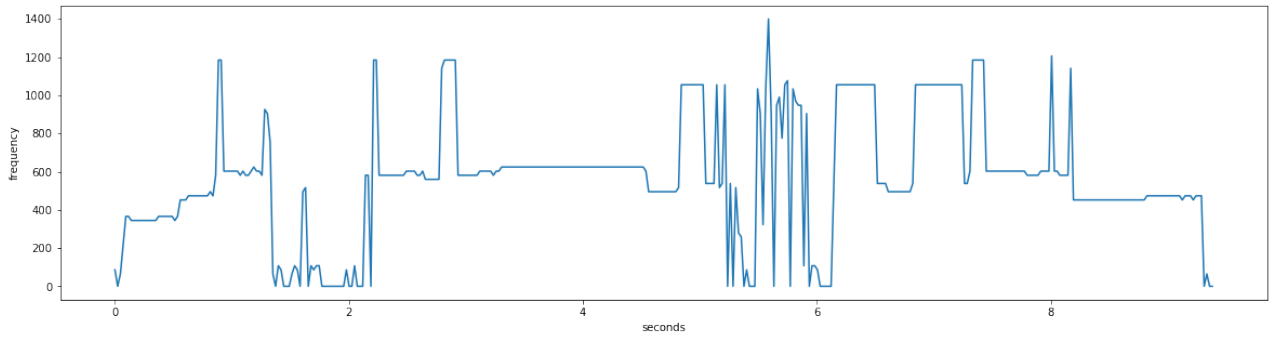
It could be seen that from the above figure, though the result of Yin has some much high frequancy at the time around 2 and 6, their main fluctuation remains similar with the result of PYin.

We also perform pitch tracks using zero-crossing rate, magnitude spectrum, auto-correlation, as well as AMDF on the same audio file. We use a frame (window) size of 1024 and a hop-size of 512 and then plot the resulting pitch tracks. Sonify all the results could provide us the generated audios for the comparison with the original sax audio. In this part, we would use just-in-time (jit) compilation using numba to accelerate those algorithms.[1]

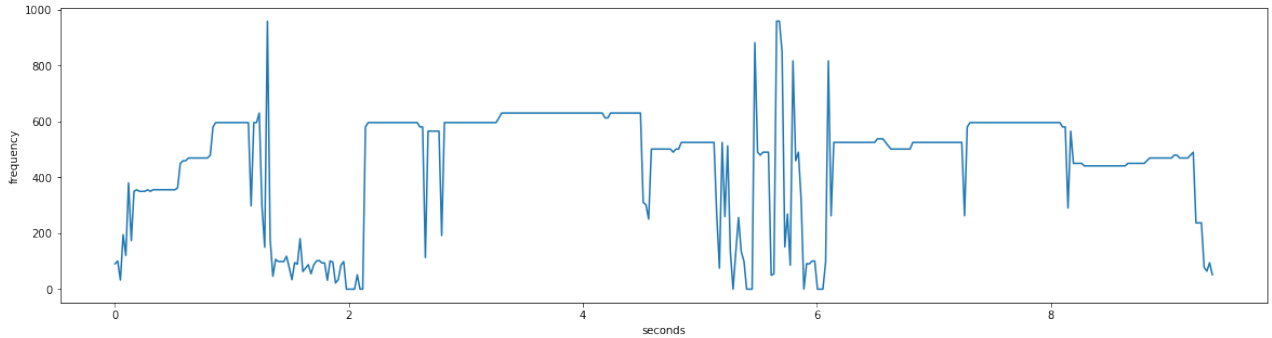
Again, we plot the resulting pitch tracks for all these methods and label the y-axis with frequency in hertz and the x-axis with time in seconds for better visualization in Figure 4.



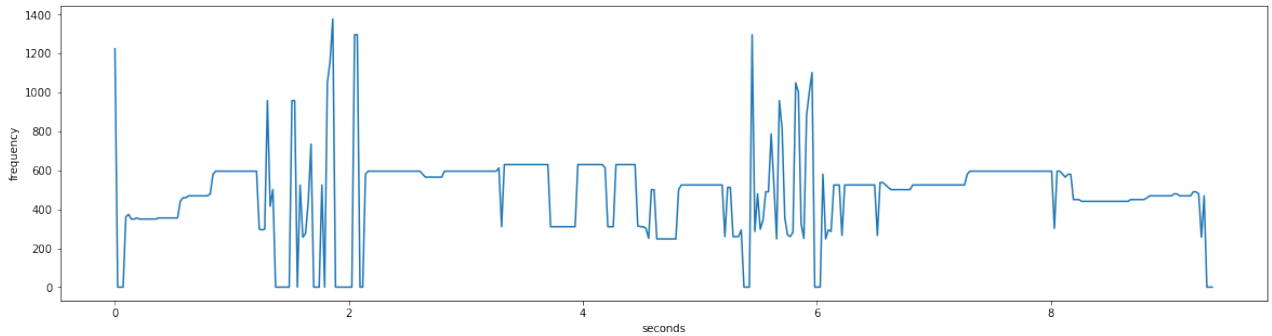

(a) zero-crossing rate




(b) magnitude spectrum




(c) auto-correlation




(d) AMDF

Figure 4: pitch track

Upon listening tests and the camparison between Figure 3 and 4, we could conclude that Zero-crossing is quite sensitive to noise, so it gives a not very ideal result. Although the result of Magnitude spectrum is mixed by some high frequency signals when there's no note playing, we can generally recognize that the sonified melody is considerably identical to the original audio. Though the result of auto-correlation is mixed by some high

frequency signals when there's no note playing, we can recognize that the sonified meody is highly identical to the original audio. The results of AMDF and Yin are also highly identical to the original audio. Finally, the result of PYin is perfectly identical to the original audio, which proves that auto-PYin is an excellent method of pitch detection. Meanwhile PYin is the only method that solves the problem of high frequency signals between different notes. We could use the result of PYin as important feature when performing some classification tasks based on pitch variability.

## Evaluation

In this part, we perform an evaluation of these pitch extraction algorithms using the part of MAST melody dataset. These audios are melodies being played on a piano as well as being sung. Also, there are text files that have the ground truth frequency to compare the results of each algorithm against.[1]

We need to calculate the mean-square error (MSE) between the pitch estimation from each algorithm with the ground truth for both examples and compare the results. We will run each pitch estimation algorithm with the correct hop size so that we get the correct number of pitch estimations to match up with the ground truth data. Specifically, we should convert the pitch tracks to fractional MIDI note numbers before calculating error.[1]

Because the frequency of chromatic music notes spaces logarithmically instead of linearly, calculating MSE using frequency would result in disorder and inaccuracy of the error measurement in different pitch levels. For MSE to make more sence, we convert the pitch tracks to fractional MIDI note numbers so that we can measure the errors linearly. Therefore the error measurement over different pitch levels is consistent and accurate.

After calculating the MSEs for the piano melodies and voice melodies, we give the table showing the comparisons between different methods in Figure 5.

| Methods | Piano melody | voice melody |
|---|---|---|
| zero-crossing | 533.82776 | 1282.0443 |
| magnitude spectrum | 343.73898 | 826.604 |
| autocorrelation | 146.14526 | 257.95825 |
| AMDF | 172.55042 | 913.74695 |
| Yin | 149.96696 | 225.2246 |
| PYin | 96.74928 | 244.87141 |

Figure 5: Table of the MSE results

When we perform these algorithms over the piano melody where there is only instrument playing, these tools accomplished better detection than in human voice environment.

To be specific, We can observe that PYin achieves the best performance in pitch detection. Not as good as PYin, Yin, AMDF and Autocorrelation still give small MSE values and almost identical melodies to the original melody. Magnitude spectrum, despite of big MSE values, generate considerably identical melody to the original melody using sonify function. In contrast, Zero-crossing is a theoretically scientific method but it always gives not very ideal results due to the interference of noise.

On the other hand, when performing these algorithms over the voice melody, the experiment shows that given noise and multiple frequency waves(human voice environment), these pitch detection tools could not remain as effective as in purer environments where only instruments playing. We need more powerful tools in such a complicated situation.

Specifically, PYin and Yin still make good approximation to the original melody. Though generative melody of Autocorrelation using sonify function could be arguably recognized identical to the original melody, this algorithm gives quite large MSE values. By comparison, the performances of AMDF, Magnitude spectrum

and Zero-crossing are not ideal because complicated environment and noise have negative impacts on the ways they work. The evaluation results again show that we could use the result of PYin as important feature when performing some classification tasks based on pitch variability.

# Timbral Feature Extraction

## Methods

In this part, we examine timbral feature extraction using spectral centroid and use it to visualize different musical genres. Spectral centroid is define as

$$C = \frac{\sum_{k=0}^{N-1} |X[k]|k}{\sum_{k=0}^{N-1} |X[k]|}$$

where $k$ is the DFT bin number, $|X[k]|$ is the magnitude for bin $k$ for the frame being analyzed, and $N$ is the number of bins - we use only positive frequency bins here, so if the input signal is length $L$, then $N = L/2 + 1$ for even $L$. Through testing, the spectral centroid result using a sinusoid at 440Hz, 2048 samples long, with a sample rate of 16kHz is much close to 440Hz.[1]

However, if we test the spectral centroid method using an instrumental sound at the same pitch as the 440Hz sine wave, it will give a different result as 1523.479415 Hz. Because flute is an instrument that outputs multiple frequency waves, it is reasonable that the estimation of spectral centroid would be different from that of single frequency wave. The additional frequency waves would result in the increase of the spectral centroid measurement. By looking at the standard FFT of 1-second flute audio we can observe how multiple frequency waves have impacts on the spectral centroid measurement as shown in Fugure 6.

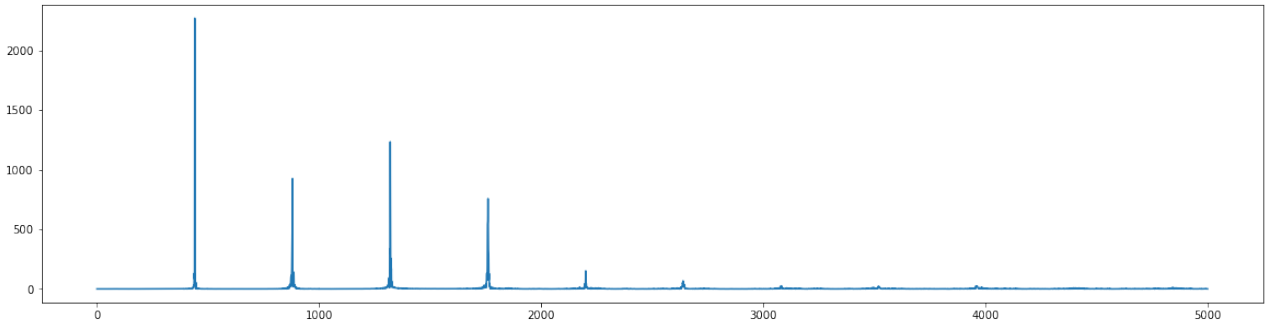



Figure 6: standard FFT of 1-second flute audio

Then, we will explore plotting and sonifying the spectral centroid and compare it to the pitch of a melody. We choose a piano melody audio file used in the previous section, create a pitch track using one of the Yin extraction methods from the previous section, and also create a spectral centroid track using frame-by-frame computation of the spectral centroid.[1] As comparison, we create a plot of the pitch track and the spectral centroid together. Also the y-axis is in Hertz and the x-axis is time in seconds as shown in Figure 7.

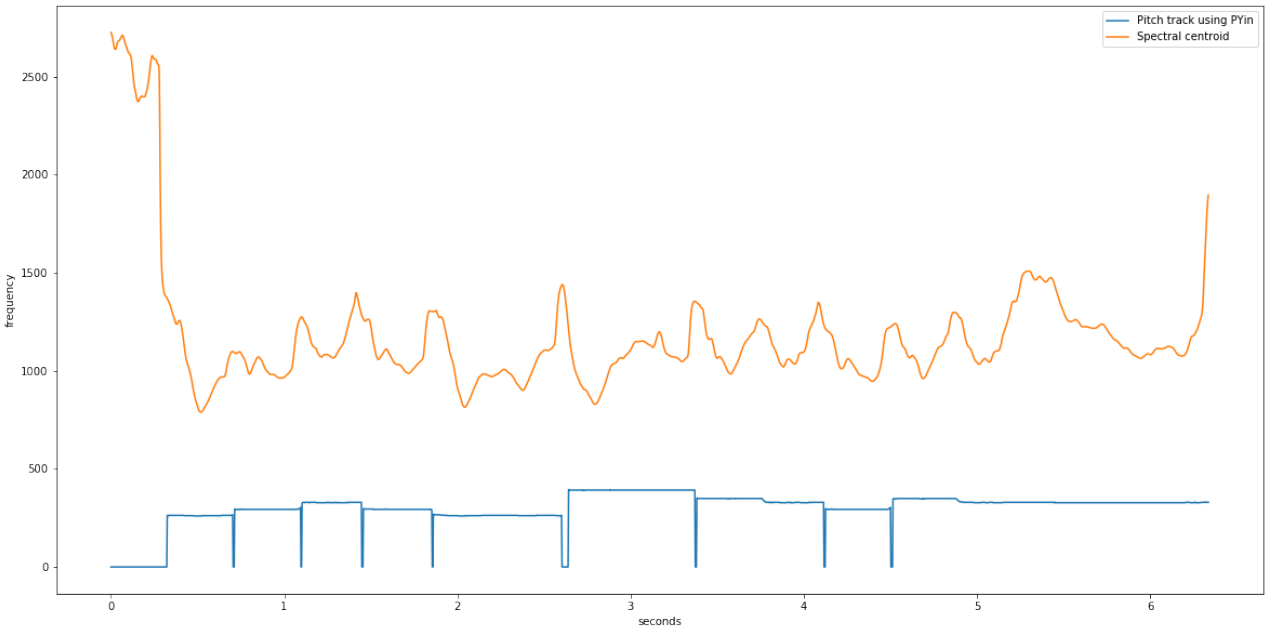



Figure 7: pitch track

## Music Plot using Spectral Centroid

Finally, we will visualize a few different musical genres in 2D using the spectral centroid. A selection of 30 second snippets from three different genres (classical, hip-hop, and blues) are selcted from the GTZAN dataset. 100 examples from each genre are included. We load all of these audio files and compute the spectral centroid on them individually using a frame size of 2048 and a hop length of 512. Then we perform song-level time summarization by calculating the mean and standard deviation of the frame-by-frame results for each audio file. Finally, we plot each sample using a scatter plot with the mean on the x-axis and the standard deviation on the y-axis.[1] The resulted classification plot is shown in Figure 8.

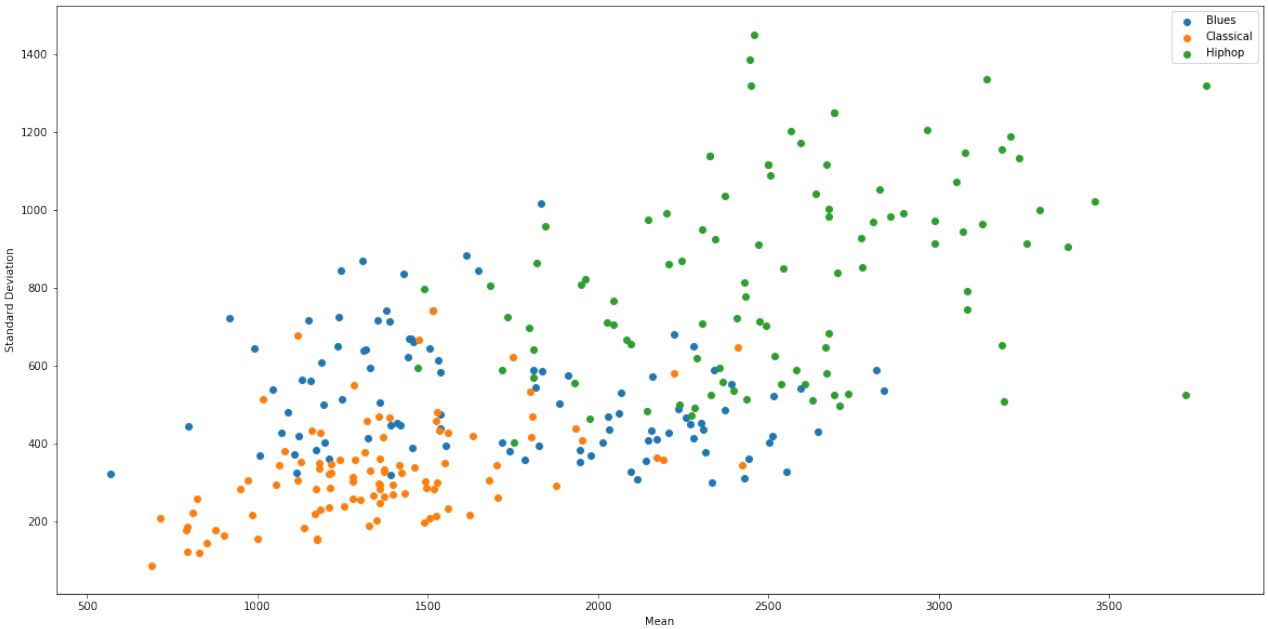



Figure 8: plot of each sample using a scatter plot

From the above figure, we conclude that these different genres of music have different features when their means and standard deviation are spaced on the coordinates.

Classical music always has lower spectral centroid than blues and hip-hops, because classical instruments, like piano, do not generate multiple waves with too high frequencies and there is no complicated human voice and too heavy drums, so that the spectral centroid would not be too high. And, classical music has relatively small standard deviation, which means that classical music has more stable spectrum. When we listen to classical music, we tend to not expect too heavy drums and abruptly unharmonious notes. The whole melody goes always as smooth as water flows. There is no big gap between pitches of different notes.

In comparison, blues music has higher spectral centroid than classical music because the accompaniment of blues music have more drums and different instruments, mixed with the complicated human voice, resulting in the higher spectral centroid. But we can observe that the standard deviation of blues music is also not very big, which means that blues music also has relatively stable spectrum. When we listen to blues music, we can feel the whole tone set of the melody is always within a fixed range and there are no very drastic ups and downs with the notes. These characteristics considerably correspond to the statistics in the picture.

The statistical dots of hip-hop music are discretely spaced on the half right area of the coordinates. It is reasonable that hip-hop music has the highest spectral centroid and most unstable standard deviation. In hip-hop music there are lots of heavy drums and electronic accompaniments with high frequencies waves, along with the complicated human voice, resulting in the highest spectral centroid. Meanwhile the least stable deviation shows that there is frequent pitch variation of music notes when the music playing. When we listen to hip-hop music, we look forward to the sensory stimulation brought by these elements. The evaluation results again show that we could use the result of spectral centroid as important feature when performing some classification tasks based on pitch variability.

## Genre Classification using Supervised Learning based on Audio Features

In this part, we again build on the audio feature extraction using spectral centroid from another dataset that consist of other genres of musics. We will use these results for audio classification. We'll perform experiments on three different genres: classical, disco, and reggae. There are 300 audio files in total, 100 for each genre. These audio files are available in the GTZAN dataset.[1] we computes the mean and standard deviation of the spectral centroid for each track and plots on a scatter plot as shown in Figure 9.

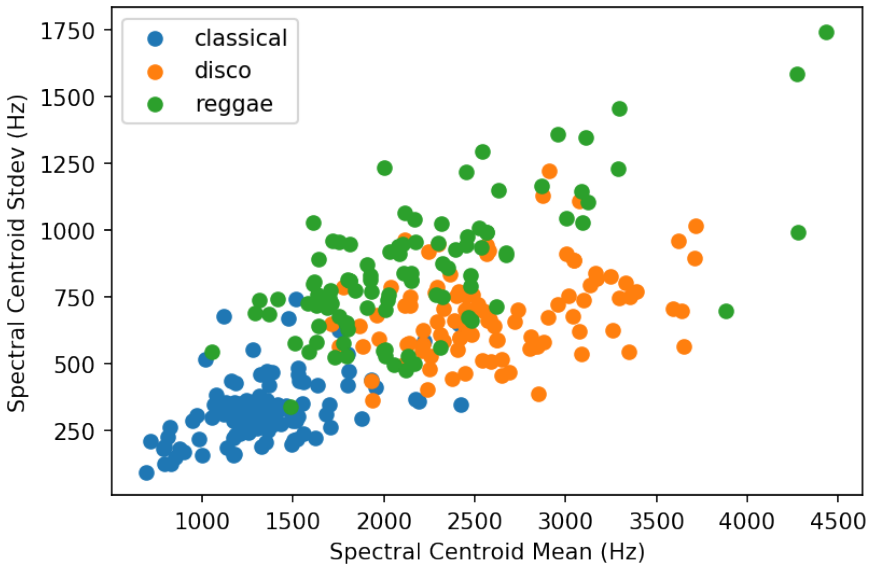



Figure 9: plot of each sample using a scatter plot

## Mean Centroid and Standard Centroid

We report the 10-fold cross-validation classification accuracy for a linear support vector machine and a naive bayes classifier trained on the two features calculated above (mean centroid and std centroid) to predict the three genres. Then we show the confusion matrix for each case.Firstly, we preprocess the data for the convenience of the classification task. Standardization gives the boxplot as shown in Figure 10.

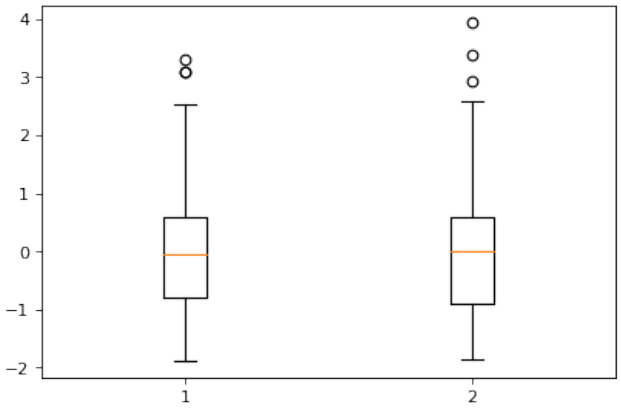



Figure 10: The boxplot of the data after standardization

We report the 10-fold cross-validation classification accuracy for a linear support vector machine, a naive Bayes classifier and their confusion matrices as shown in Figure 11.

```
The confusion matrix is:
[[90  5  5]
 [ 2 76 22]
 [ 5 14 81]]
The testing accuracy: 82.33%
```

(a) SVM

```
The confusion matrix is:
[[90  5  5]
 [ 2 79 19]
 [ 6 21 73]]
The testing accuracy: 80.67%
```

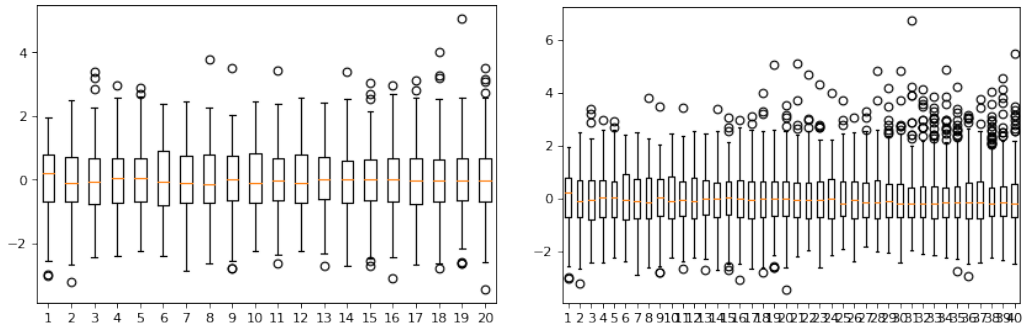
(b) naive Bayes

Figure 11: classification accuracy and confusion matrices

## MFCCs

We compute the MFCCs for each recording using the default settings of librosa. Then we summarize the entire recording by taking the mean of the MFCCs across the recording as well as the mean and standard deviation across each recording. The resulting configurations will be just the mean (20 features per recording) and the mean and std (40 features per recording).[1]

We report on the 10-fold cross-validation classification accuracy and confusion matrix for these two configurations using the linear support vector machine and naive bayes classifier. The box plots of mean and std data after standardization are shown in Figure 12. 10-fold cross-validation classification accuracy for these two configurations using a linear support vector machine and a naive bayes classifier and their confusion matrices are shown in Figure 13 and Figure 14.



(a) mean of the MFCCs (b) mean and standard deviation of the MFCCs

Figure 12: box plots of MFCCs

```
The confusion matrix is:
[[95  1  4]
 [ 3 83 14]
 [ 6 14 80]]
The testing accuracy: 86.00%
```

(a) linear support vector machine

```
The confusion matrix is:
[[91  2  7]
 [ 4 53 43]
 [ 5 23 72]]
The testing accuracy: 72.00%
```

(b) naive bayes classifier

Figure 13: classification accuracy and confusion matrices for mean configuration

```
The confusion matrix is:
[[97  0  3]
 [ 1 84 15]
 [ 2 17 81]]
The testing accuracy: 87.33%
```

(a) linear support vector machine

```
The confusion matrix is:
[[98  2  0]
 [ 1 80 19]
 [ 2 22 76]]
The testing accuracy: 84.67%
```

(b) naive bayes classifier

Figure 14: classification accuracy and confusion matrices for mean and std configuration

## Dimensionality Reduction

We reduce the dimensionality of the 300 by 40 feature matrix of mean and std mfccs to a 300 by 2 feature matrix, and then visualize the corresponding scatter plot with coloring of the points based on genre. Again, we run the 10-fold cross-validation classification accuaracy and show the confusion matrix for the same configurations as the full feature set, but now using only the 2 dimensions returned from t-SNE.[1]

t-SNE converts similarities between data points to joint probabilities and tries to minimize the Kullback-Leibler divergence between the joint probabilities of the low-dimensional embedding and the high-dimensional data. As shown in Figure 15, we can observe in the scatter plot that the separations and overlaps are all more distinctly recognizable compared to those in the scatter plot of just the centroid statistics.

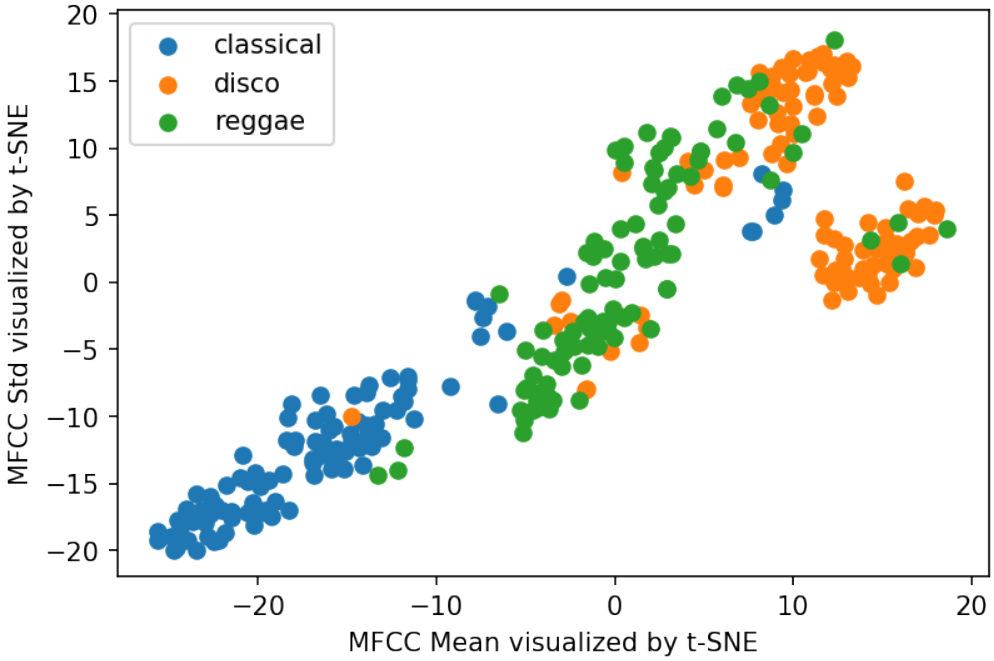



Figure 15: scatter plot after dimensionality reduction

10-fold cross-validation classification accuracy for the embedded mean and std configuration using a linear support vector machine and the confusion matrix is shown in Figure 16(a). We just lost a litttle accuracy (about 3%) compared to the previous prediction. But the amount of loss is tolerable given we reduced a feature matrix with very high dimensionality to one with much lower dimensionality.

10-fold cross-validation classification accuracy for the embedded mean and std configuration using naive bayes classifier and the confusion matrix is shown in Figure 16(b). We just lost a litttle accuracy (about 4%) compared to the previous prediction. But the amount of loss is tolerable given we reduced a feature matrix with very high dimensionality to one with much lower dimensionality.

```
The confusion matrix is:
[[97  0  3]
 [ 1 84 15]
 [ 2 17 81]]
The testing accuracy: 87.33%
```

(a) linear support vector machine

```
The confusion matrix is:
[[98  2  0]
 [ 1 80 19]
 [ 2 22 76]]
The testing accuracy: 84.67%
```

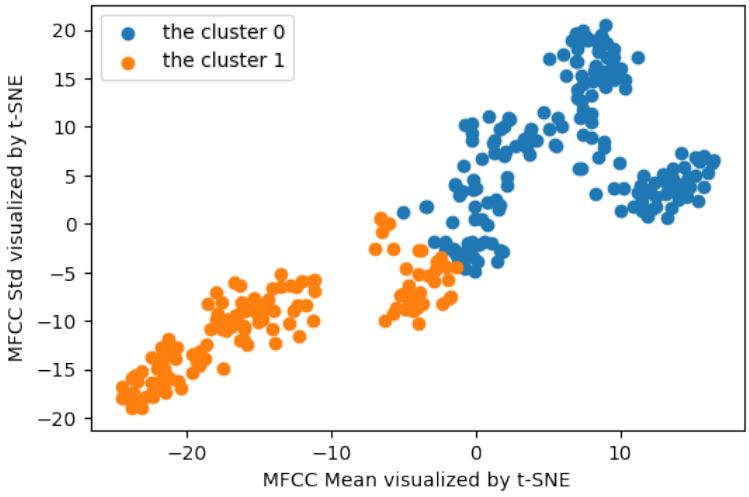
(b) naive bayes classifier

Figure 16: classification accuracy and confusion matrices for the embedded mean and std configuration

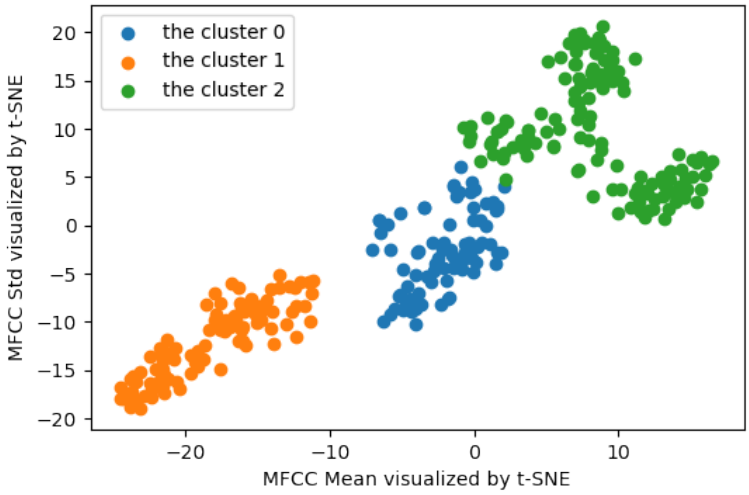
## Genre Classification using Unsupervised Learning based on Audio Features

Let's forget about the genre labels and do some unsupervised learning. In this part we will perform clustering on the two-dimensional results from t-SNE using K-Means.[1] We plot the results of clustering using three different choices for the number of clusters.

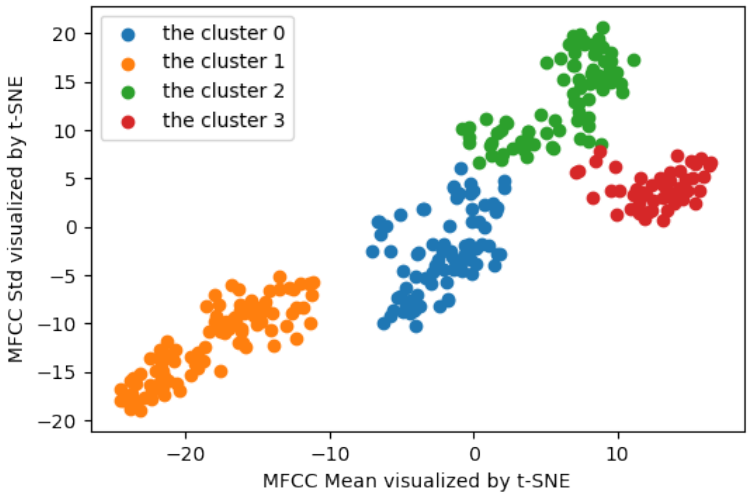
We choose 2,3,4 for the number of clusters to plot as shown in Figure 17. Clearly we can observe that 3 clusters give the best result that is considerably identical to the plot from the previous question in spite of the lack of some overlaps shown in the previous one.




(a) 2 cluster




(b) 3 clusters




(c) 4 clusters

Figure 17: clustering on the two-dimensional results

We respectively select three points randomly from 2 clusters out of the result of unsupervised learning with the number of clusters equal 3, and we play the associated audio files (6 in total). We can notice that the audio files from the same cluster sound similar, which proves that they may be from the same genre. Furthermore, we can also see that they have the similar filenames that all include the same genre names. So the model has separated them into different clusters in terms of their own genres.

## Genre Classification based on Song Lyrics

In this section, we will look at Naive Bayes classification with song lyrics. Our goal will be to build a simple Naive Bayes classifier for the MSD dataset, which uses lyrics to classify music into genres. More complicated approaches using term frequency and inverse document frequency weighting and many more words are possible but the basic concepts are the same.[1]

We are going to use the musicXmatch dataset which is a large collection of song lyrics in bag-of-words format for some of the tracks contained in the Million Song dataset (MSD). The corresponding genre annotations, for some of the song in the musicXmatch dataset, is provided by the MSD Allmusic Genre Dataset. For demonstration purpose, we use a reduced version of the musicXmatch dataset. Three genres are considered, namely: "Rap", "Pop Rock", and "Country". The resulting genre annotated dataset is obtained by an intersection of musicXmatch and MAGD, where we select 1000 instances of each genre, such that the three classes are balanced and easy to handle. In addition, we also reduce the cardinality of the dictionary of words used for the bag-of words lyrics representation (originally equal to 5000), to the 10 best words for each genre. Intuitively, the best words are the most frequent words for a particular genre that are not frequent among all the genres. The resulting dictionary of words is presented in 18.[1]

| Genre | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| rap | de | hood | ya | und | yall | ich | fuck | shit | yo | bitch |
| rock | end | wait | again | light | eye | noth | lie | fall | our | away |
| country | gone | good | night | blue | home | long | littl | well | heart | old |

Figure 18: The resulting dictionary of words

Firstly, we calculate the probabilities for each dictionary word given the genre. For the purposes of demonstration we are considering only the tracks belonging to the three genres: Rap, Rock/Pop, Country. We use add-one additive smoothing to handle the case that there is no instance of a particular word in a genre. Specifically, we calculate the conditional probability $P(word|genre)$ using additive smoothing. A zero in a track indicates that the word does not exist in a song and a one indicates that a word appears in a song (regardless of how many times it appeared). So we set every element in a track that is >= 1 to 1, for the convenience of further calculation. And we use a $30 \times 3$ matrix to represent the conditional probability matrix of $P(word|genre)$. The rows represent different word and the colomns represent Rap, Rock/Pop, Country respectively, as shown in Fugure 19.

```
[[0.01392625 0.00882285 0.00145048]
 [0.02943504 0.00162526 0.00082884]
 [0.06963127 0.01068029 0.01077497]
 [0.00996993 0.00742977 0.00020721]
 [0.04478557 0.00162526 0.00414422]
 [0.00917867 0.00626886 0.00020721]
 [0.06535844 0.02043186 0.0018649 ]
 [0.08055072 0.00951939 0.00248653]
 [0.06520019 0.00534014 0.00269374]
 [0.04953315 0.00441142 0.00124327]
 [0.02848552 0.04643603 0.02983838]
 [0.01851559 0.04411423 0.02900953]
 [0.0272195  0.05131182 0.0435143 ]
 [0.03117582 0.04643603 0.03937008]
 [0.03687292 0.07174367 0.05428927]
 [0.0191486  0.04457859 0.02590137]
 [0.01772432 0.04318551 0.01989225]
 [0.02247191 0.05200836 0.03543307]
 [0.03402437 0.05525888 0.04289266]
 [0.02579522 0.07452984 0.05594695]
 [0.027536   0.03575575 0.04227103]
 [0.04272828 0.03668447 0.0567758 ]
 [0.0356069  0.06152775 0.07749689]
 [0.01519228 0.01485953 0.03336096]
 [0.03006805 0.03738101 0.05325321]
 [0.02911853 0.04156025 0.06527145]
 [0.0382972  0.03436267 0.06464981]
 [0.03386612 0.04573949 0.06651471]
 [0.02611173 0.06059902 0.07708247]
 [0.02247191 0.025772   0.06133444]]
```

Figure 19: conditional probability matrix of $P(word|genre)$

We can consider the naive Bayes classifier as a generative model that can generate binary feature vectors using the associated probabilities from the training data. The idea is similar to how we do direct sampling in Bayesian Networks and depends on generating random number from a discrete distribution (the unifying underlying theme of this assignment question). Here we describe how we would generate random genre "lyrics" consisting solely of the words from the dictionary using your model.

A good way to think about that is to treat each word as a random variable, and then sample each word given the probability it exists for a genre. Then a genre includes it if it comes out as being included, and a genre doesn't include it otherwise.

For 30 words we create 30 random variable with two values $[included, unincluded]$ and the corresponding conditional probabilities. When generating lyrics, for every random variable we do sampling for one time. If the value returned is 1, then we can say that the song has this word as lyrics. We show 5 examples of randomly generated tracks for each of the three genres: Rap, Rock pop, and Country; each example consists of a subset of the words in the dictionary in Figure 20.

```
The generative Rap lyrics for the 1 time:
hood shit end
The generative Rap lyrics for the 2 time:
shit our well
The generative Rap lyrics for the 3 time:
shit
The generative Rap lyrics for the 4 time:
yall
The generative Rap lyrics for the 5 time:
long
```

(a) generative Rap lyrics

```
The generative Rock lyrics for the 1 time:
light our long
The generative Rock lyrics for the 2 time:
our
The generative Rock lyrics for the 3 time:
end fall away gone
The generative Rock lyrics for the 4 time:
blue old
The generative Rock lyrics for the 5 time:
fall
```

(b) generative Rock lyrics

```
The generative Country lyrics for the 1 time:
littl
The generative Country lyrics for the 2 time:
long old
The generative Country lyrics for the 3 time:
home
The generative Country lyrics for the 4 time:
blue
The generative Country lyrics for the 5 time:
again noth littl
```

(c) generative Country lyrics

Figure 20: generative lyrics

We need to explain how these probability estimates can be combined to form a Naive Bayes classifier.[1] We calculate the classification accuracy and confusion matrix that we would obtain using the whole data set for both training and testing.

We already have the conditional probability $P(word|genre)$ matrix which is the prior probabilities. Given the formula of Naive Bayes theorem,

$$P(genre|word_1, word_2, ...) = \frac{P(word_1|genre)P(word_2|genre)..... \times P(genres)}{P(word_1, word_2, .....)},$$

we can calculate the probabilities of different combination of words shown given a genre. And, because the deterministic values $P(genres)$ and $P(word_1, word_2, .....)$ are calculated over training set and $P(Rap) = P(Rock) = P(Country)$, we can form a Naive Bayes classifier through just comparing the prior conditional probabilities using the conditional probability matrix.

Having introduced the previous content, we make the prediction using the entire data as the testing set, report the prediction accuracy and show the confusion matrix in Figure 21.

```
The confusion matrix is:
[[556. 327. 117.]
 [223. 738.  39.]
 [107.  88. 805.]]
The testing accuracy: 69.97%
```

Figure 21: prediction accuracy and the confusion matrix

Finally, we calculate the classification accuracy and confusion matrix using the k-fold cross-validation, where k = 10. Note that we generate our own splits. We firstly shuffle the dataset, split it into 10 folds and calculate the classification accuracy and confusion matrix using the 10-fold cross-validation. From the Figure 22 we can observe that the results using 10fold cross-validation are identical to the previous prediction.

```
The confusion matrix is:
[[550. 332. 118.]
 [228. 731.  41.]
 [110.  88. 802.]]
The testing accuracy: 69.43%
```

Figure 22: prediction accuracy and the confusion matrix

# Conclusion

In this project, firstly, I firstly introduced the performance comparisons between different methods for pitch extraction, including zero-crossing, magnitude spectrum, autocorrelation, AMDF, Yin and PYin. I also introduced the timebral feature extraction method using spectral centroid and gave the visualization of the different musical genres in 2D using the spectral centroid representation. The evaluations show that spectral centroid is a good feature used for music genre classification tasks based on machine learning methods. Then I respectively introduced the supervised and unsupervised machine learning methods used for music genre classification tasks based on audio features and lyrics and experiment these methods on different dataset. The classification results show that these methods work well with the accuracy up to 87% for audio feature and 70% for lyrics, for this challenging task.

# References

[1] J. Shier, *Csc475: Music retrieval techniques course materials.*