# Music Genre Classification

Clark Brown, Sam Cochran, Daniel Swingle

October 26, 2020

[1]:
```python
import numpy as np
import pandas as pd
from pandas.plotting import parallel_coordinates
from models import logistic_regression
from matplotlib import pyplot as plt
```

[2]:
```python
%matplotlib inline
plt.rcParams['figure.dpi'] = 96
```

## 1 Introduction

Our work seeks to curate audio features to train a music genre classifier.

### 1.1 Motivation

It is a somewhat simple task for a trained musician or musicologist to listen to a work of music and label its genre. What do we need to help a computer complete the same task? Questions we want to answer:

1. What features of music make it a part of its genre?

### 1.2 Related Work

There have been many studies in the area of genre classification in machine learning. Traditionally models have used learning algorithms for SVM and KNN and have relied heavily on commone spectral features including the MFCCs (1). The state of the art has improved over time with most classical machine learning classifiers managing 60-70% accuracy. This is similar to human capabilities with short song intervals according to some human trials (2). In more recent years, neural networks have been able to make more accurate predictions near 80-90% accuracy in some cases.

# 2 Data

Our data comes from the Free Music Archive ([https://github.com/mdeff/fma]) reated by Michaël Defferrard, Kirell Benzi, Pierre Vandergheynst, Xavier Bresson. International Society for Music Information Retrieval Conference (ISMIR), 2017.

We use the audio files and genre tags, but build our own features. We also use the small data set composed of 8000 30-second songs (8 GB in `.mp3` fromat). We convert each file to a `.wav` for simplicity. Each song is designated by a `track_id` and labeled with one of eight genres: Hip-Hop, Pop, Folk, Experimental, Rock, International, Electronic, and Instrumental. There songs are distributed evenly across genres with 1000 songs per genre.

## 2.1 Potential Issues

One potential issue with our data is that the dataset is composed entirely of free music (creative commons), and therefore our model may have difficulty analyzing other kinds of music, which may be quite different.

Specifically, we have reason to believe that the genre definitions, quality, and style of a free music database may differ from commercial music, so we will have to find a way to evaluate how well a model trained on a free music database can generalize to samples of commercial music

## 2.2 Missing Data

The dataset is fairly robust, but of the 8000 tracks, there are 6 that are not actually 30 seconds long. We ignore these tracks from our analysis.

## 2.3 Ethical Concerns and Implications

The music used in our work comes from the Creative Commons and is liscensed for this kind of use. We see no privacy concerns with the collection of this data. As music genre does not make a serious impact on the commercialization of music or the daily lives of non-musicians, we do not anticipate and negative repercussions from our work. The lines around genre are vague enough to ensure that professors of music theory and music history need not worry that they shall be out of a job.

# 3 Feature Engineering

Since our original data was made up only of track IDs corresponding to wav files, and their genre labels, our feature extraction makes up all of our useful data. We created a dataframe that has the following features as its columns. In the next section, we discuss the meaning of each added feature column.

### 3.1 Feature Descriptions and Reasoning

**Track ID**: each wav file corresponds to a number, and we have a function that generates the file path to access each track if needed. Genre Code: We have encoded our eight genres by a 1:1 mapping to integers 0-7.

**Zero Crossing Rate**: Indicates the average rate at which the sign of the signal changes. Higher zero crossing rates match with higher percussiveness in the song. We added this feature because genres often have a certain feel relative to beat and percussive sound.

**Frequency Range**: The max and min frequency the audio ignoring the top 20% and bottom 20%. Clipping the top and bottom was important because almost all of our audio files go from 10 Hz to 10000 Hz. But seeing the range in where most of the sound of a song is seems to be connected to genre. Some genres have greater ranges while others are in a small range.

**Key and Tonality**: We used the Krumhansl-Schmuckler algorithm to estimate the most likely key that the audio sample is in, and whether the key is major or minor. We chose this because even though most genres have songs in different keys, knowing the key will aid in normalizing pitch information for other features.

**Mel Frequency Cepstral Coefficients (MFCCs)**: Represents the short term power spectrum of the sound. Aligns closely with the human auditory system's reception of sound. These 6 (for now) coefficients describe the sound of a song in a human way. MFCCs are being used more and more in Music Information Retrieval specifically with genre tasks because they encapsulate the human experience of sound. We feel this will improve accuracy.

**Spectral Rolloff**: The frequency below which a certain percent of the total spectral energy (pitches) are contained. When audio signals are noisy, the highest and lowest pitches present do not convey much information. What is more useful is knowing the frequency range that 99% of the signal is contained in, which is what the spectral rolloff represents.

**The Three Highest Tempo Autocorrelation Peaks**: Indicative of what we would guess the average BPM will be for this audio file (3 columns). This is a way of summing up the entire tempogram array in just a few numbers so that comparing tempo features for each track is tractable.

**Average Tonnetz over all Time**: The mean and variance of the x and y dimensions of the tonal centers for the major and minor thirds, as well as the fifths (this ends up being 6 means and 6 variances for a total of 12 columns). Here we take the means and variances to reduce the information down from a 6xt matrix (where t is the number of time values, about 1200) to just 12 numbers that sum up that matrix for each track.

## 4 Visualization and Analysis

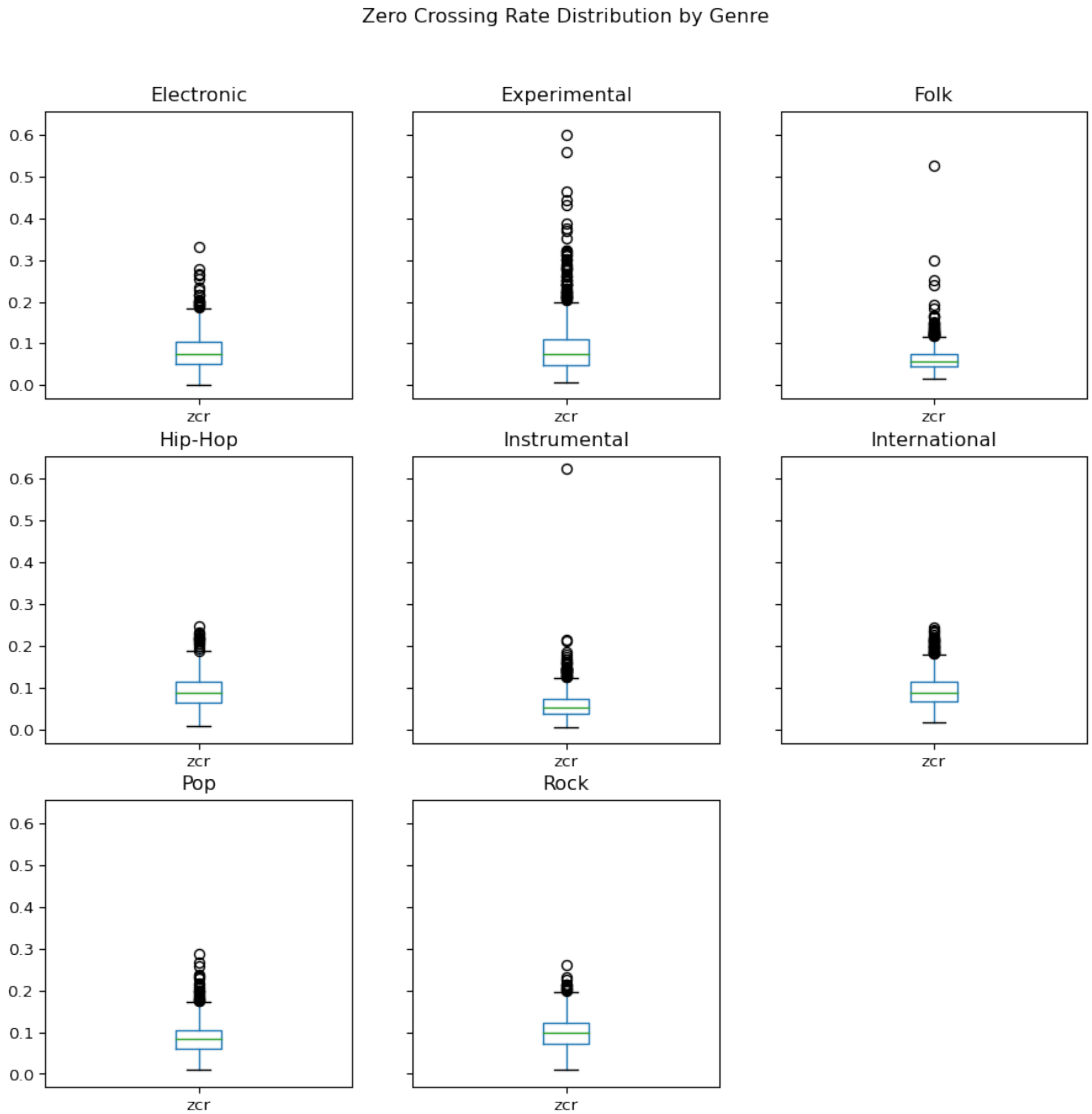
### 4.1 Visualization

For now we present visualizations of most of these features. We will eventually be more selective.

[3]:
```
genres = [
    "Hip-Hop",
    "Pop",
```

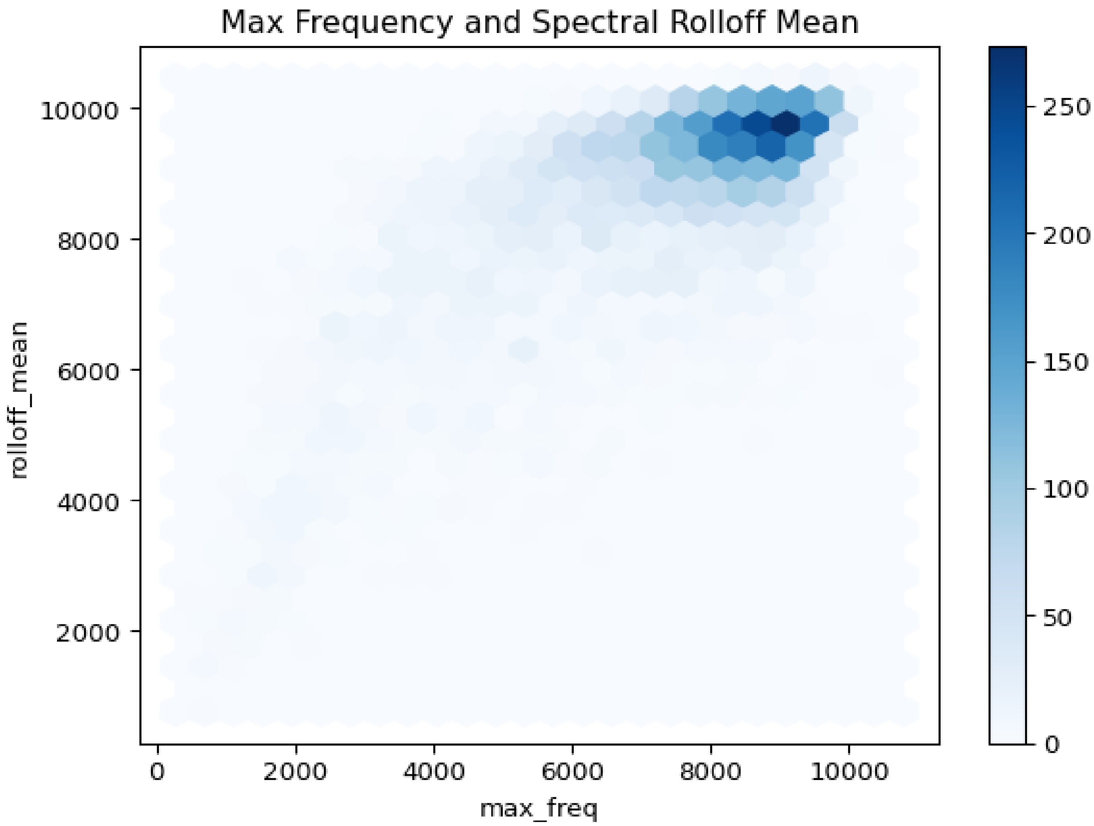
```
    "Folk",
    "Experimental",
    "Rock",
    "International",
    "Electronic",
    "Instrumental",
]

df = pd.read_csv('../data/features.csv', header=0, index_col=0)
df['genre'] = df.genre_code.apply(lambda x : genres[x])
```

[4]:
```
df[['zcr', 'genre']].groupby('genre').boxplot(column='zcr', grid=False,␣
 ↪figsize=(11,11))
plt.suptitle('Zero Crossing Rate Distribution by Genre')
plt.show()
```



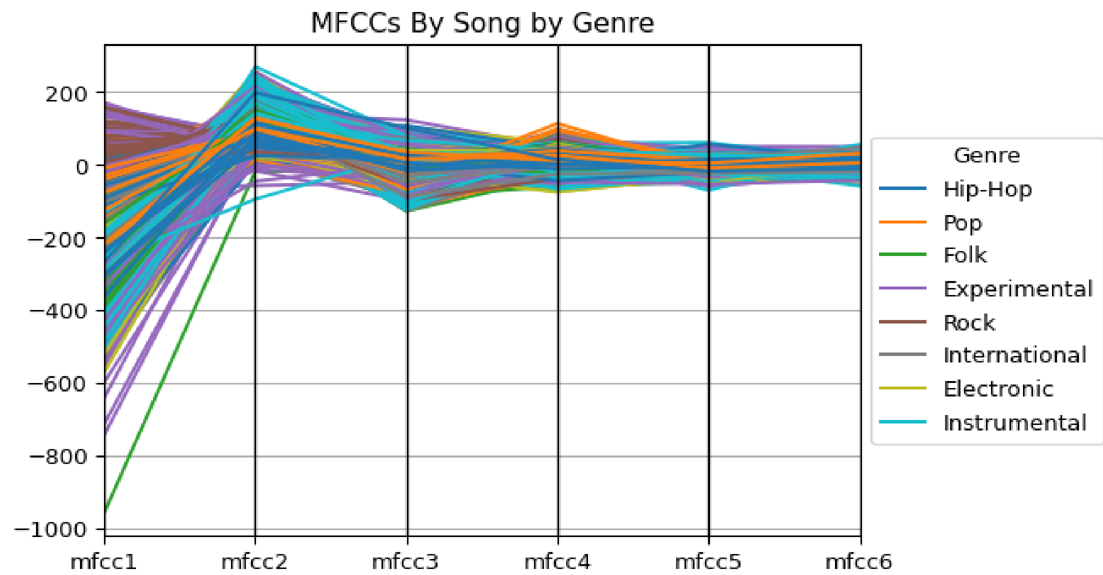

These boxplots fore each genre show the Zero Crossing Rate distribution by genre. ZCR is usually thought of as a good measure to include when doing a genre analysis because it conveys something of the percusiveness of the song. We see that the distributions differ enought to justify including it, but some genres are more drastic than others.

[5]:
```
df.plot(kind='hexbin', x='max_freq', y='rolloff_mean', gridsize=25, figsize=(7,␣
 ↪5), cmap='Blues', sharex=False)
plt.title('Max Frequency and Spectral Rolloff Mean')
plt.show()
```




The hexbin plot compares the max frequency and the spectrall rolloff mean. Because the spectrall rolloff mean is the mean frequency greater than 99% of a time frame's frequencies, it make sense that it may be redundant information or colinear with max_frequency.

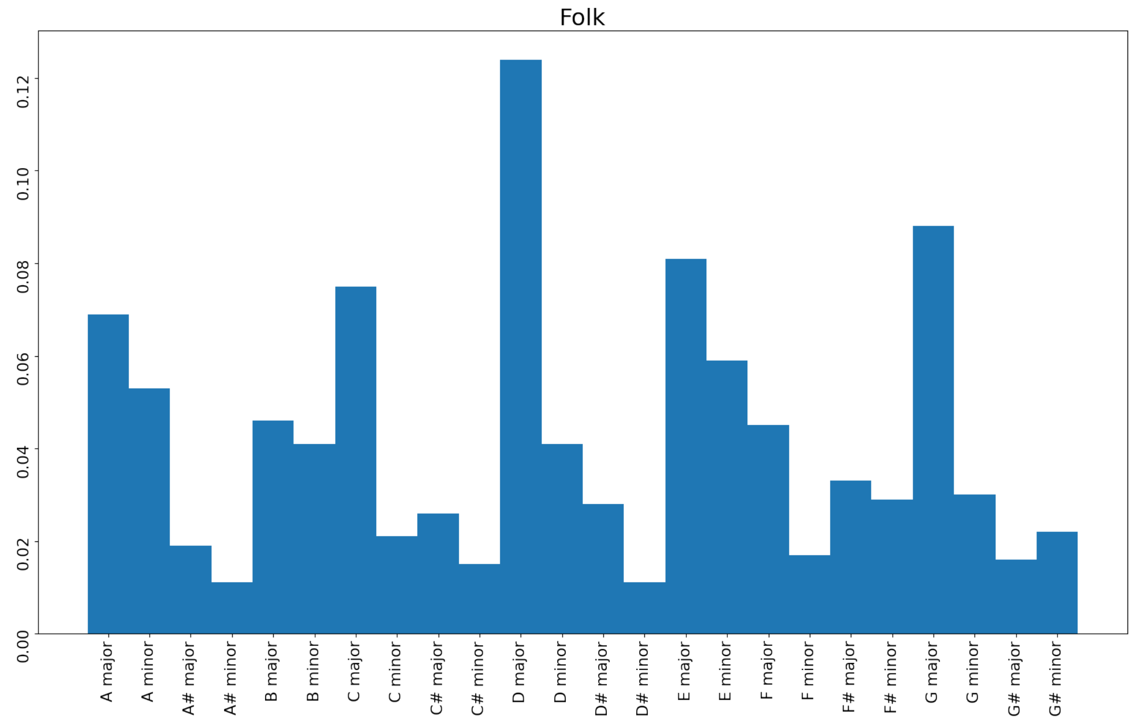
[6]:
```
mfcc_cols = [f'mfcc{i}' for i in range(1,7)]
plt.figure()
parallel_coordinates(df[mfcc_cols + ['genre']], 'genre', colormap='tab10')
plt.title('MFCCs By Song by Genre')
plt.legend(title='Genre', loc='center left', bbox_to_anchor=(1.0, 0.5))
plt.show()
```

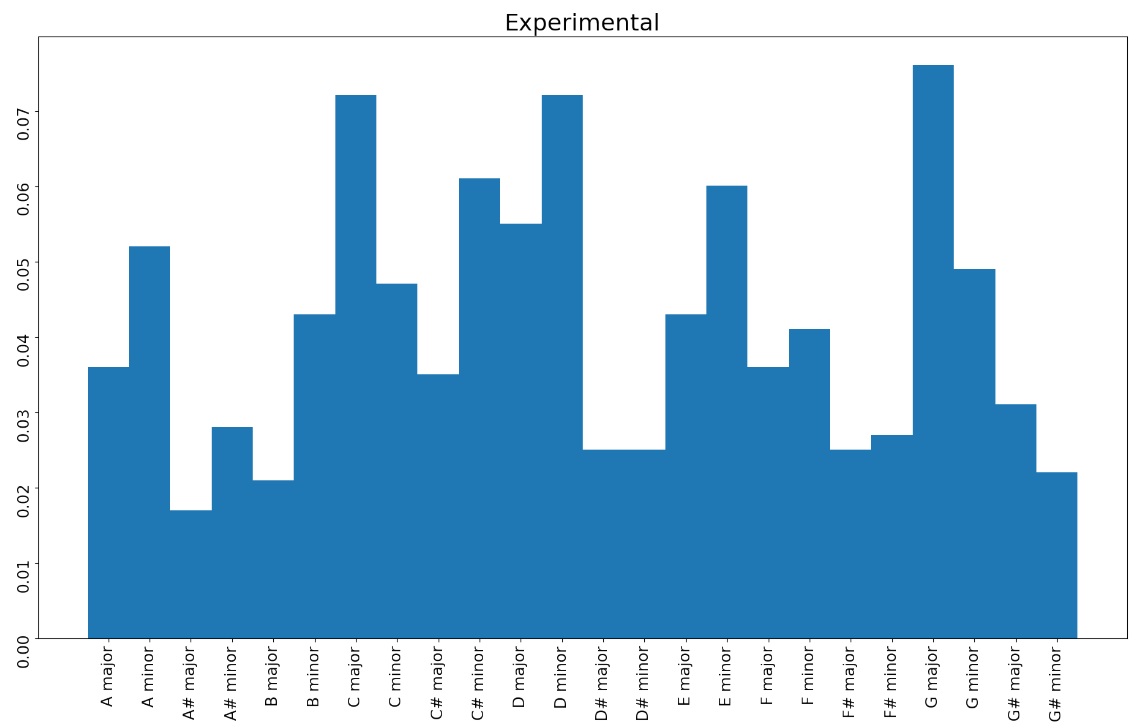



The above graph shows each songs average MFCC for each of the six band filters. It is color coded by genre. This shows that the genres do seem to have some groupings in the MFCCs. There is greater variety in the first than in the others. We may expand the number of MFCCs we compute and then be more selective about which we include.

[7]:
```python
#map an index 0-11 to a key
keys = ["C", "C#", "D", "D#", "E", "F", "F#", "G", "G#", "A", "A#", "B"]

def map_key(is_major: bool, index: int) -> str:
    return keys[index] + " " + ("major" if is_major else "minor")
```

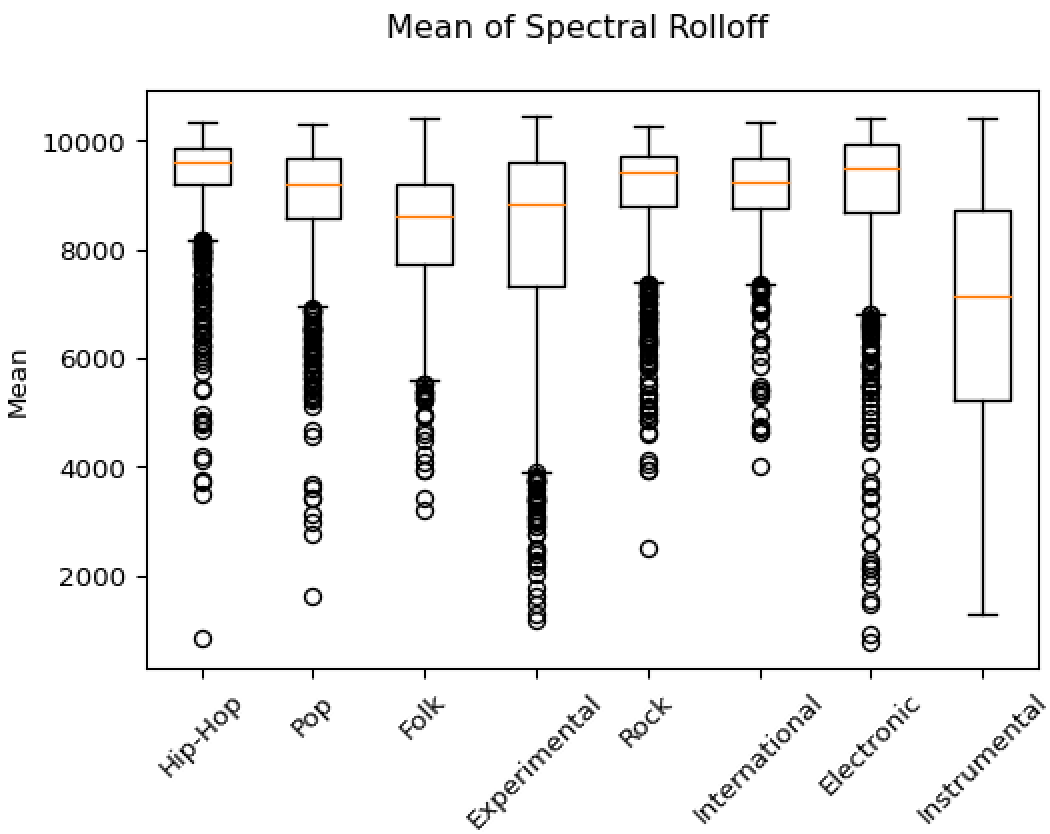
[8]:
```python
fig, ax = plt.subplots(2, 1, figsize=(20, 25))
ax = ax.ravel()
bins = np.arange(25) - 0.5

for i in range(2):
    features = df[df["genre_code"] == (i+2)]

    key = features["key"]
    major = features["major"]
    sig = np.array([
        map_key(bool(m), int(k)) for m, k in zip(major, key)
    ])
    sig = np.sort(sig)

    ax[i].hist(sig, density=True, bins=bins)
    ax[i].set_title(genres[i+2], size=24)
    ax[i].tick_params(labelrotation=90, labelsize=16)
```




The histograms above contrast the key signatures of folk and experimental songs. The plots suggest that folks songs tend to mostly be written in major keys, and of those major keys, a few are predominant (D, G, A, and C major). In contrast, experimental music tends to be written more in minor key signatures. Additionally, each key is more evenly represented.

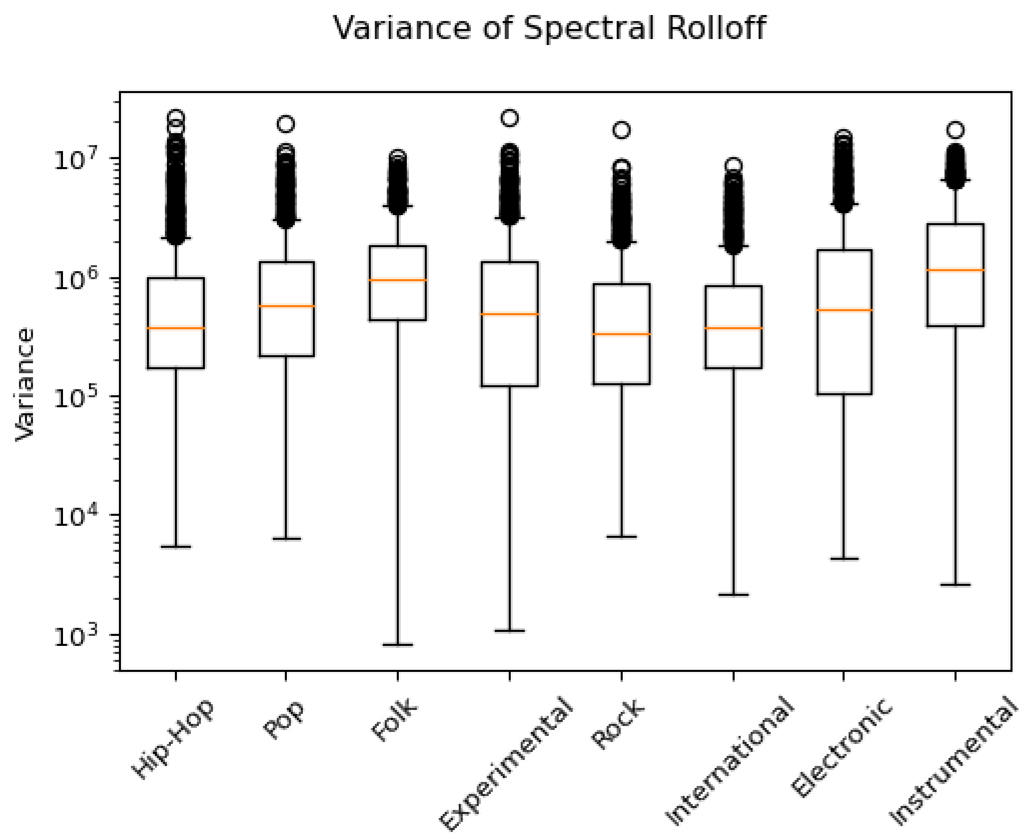
[9]:
```python
import warnings
warnings.filterwarnings("ignore",
                        category=np.VisibleDeprecationWarning)

rolloff_mean = df["rolloff_mean"]

plt.boxplot([
    rolloff_mean[df["genre_code"] == i] for i in range(len(genres))
], labels=genres)
plt.suptitle("Mean of Spectral Rolloff")
plt.ylabel("Mean")
plt.xticks(rotation=45)
plt.show()
```




A couple things to note from the plot above are the distributions of the mean spectral rolloff of experimental and instrumental music, which tend to be skewed lower than for other genres.

[10]:
```python
rolloff_var = df["rolloff_var"]

plt.yscale("log")
```

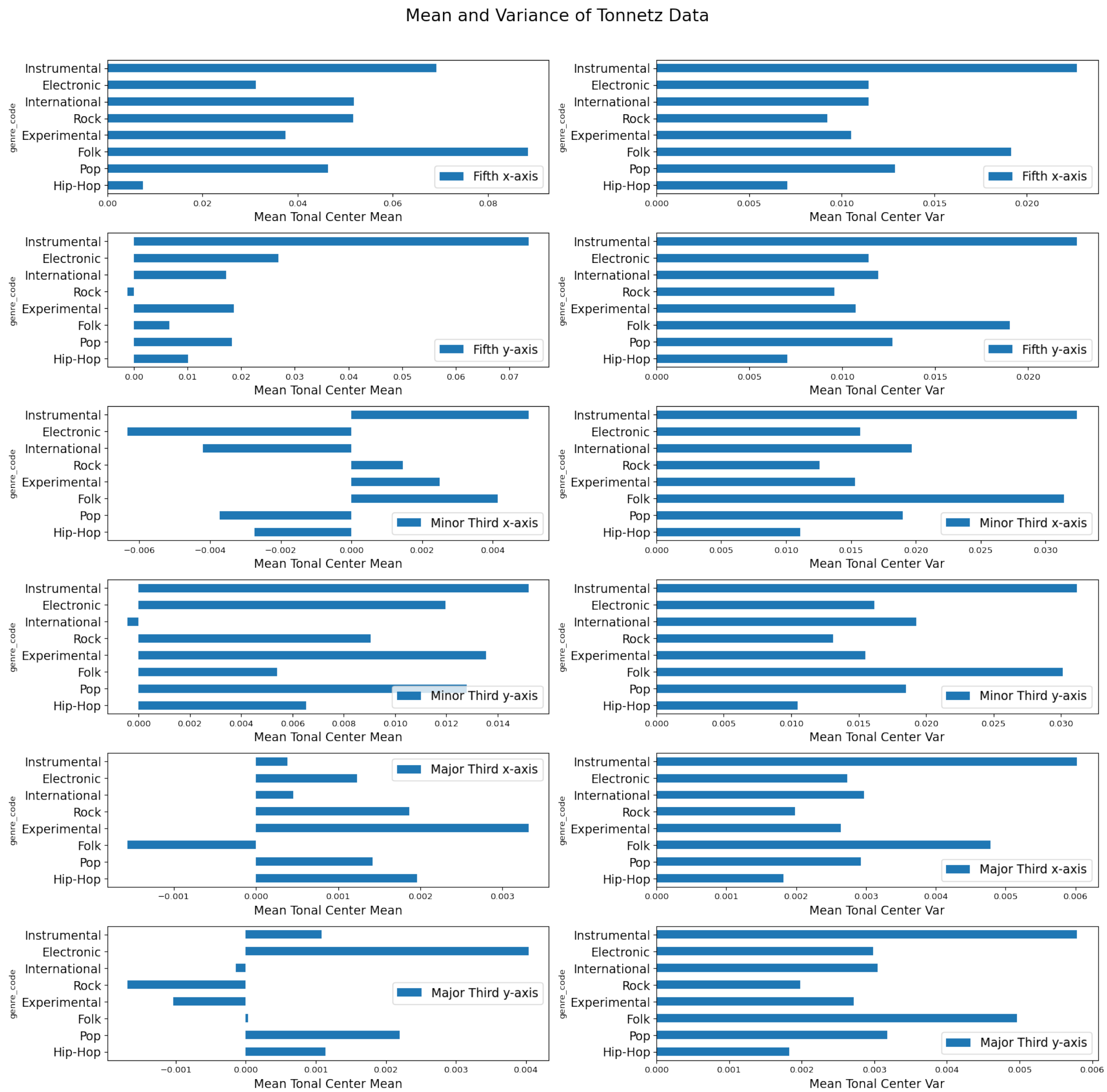
```
plt.boxplot([
    rolloff_var[df["genre_code"] == i] for i in range(len(genres))
], labels=genres)
plt.suptitle("Variance of Spectral Rolloff")
plt.ylabel("Variance")
plt.xticks(rotation=45)
plt.show()
```




The above plot suggests that the variance of the spectral rolloff may not be a useful variable to consider when differentiating between genres; the distributions for each genre are very similar.

[11]:
```
from matplotlib import rcParams
rcParams['figure.figsize'] = 18, 18

# Load the data and get the labels
data = pd.read_csv('./../data/features.csv')

# Save the genre labels
genre_labels = ["Hip-Hop", "Pop", "Folk", "Experimental", "Rock",␣
 ↪"International", "Electronic", "Instrumental"]
```

```python
tonnetz_labels = ['Fifth x-axis', 'Fifth y-axis', 'Minor Third x-axis', 'Minor␣
↪Third y-axis', 'Major Third x-axis', 'Major Third y-axis']

# Get the tonnetz features in their own dataframe and group by genre
tonnetz_features = data[['genre_code', 'tonnetz1', 'tonnetz2', 'tonnetz3',␣
↪'tonnetz4', 'tonnetz5', 'tonnetz6', 'tonnetz7', 'tonnetz8', 'tonnetz9',␣
↪'tonnetz10', 'tonnetz11', 'tonnetz12']]
group = tonnetz_features.groupby('genre_code')

# Make some bar plots
fig1, axes1 = plt.subplots(6, 2)
for k in range(12):
    group.mean()['tonnetz' + str(k+1)].plot(kind='barh', ax=axes1.
↪reshape(-1)[k], legend=True)
    axes1.reshape(-1)[k].set_yticks([x for x in range(8)])
    axes1.reshape(-1)[k].set_yticklabels(genre_labels, fontsize=14)
    if k % 2 == 0:
        axes1.reshape(-1)[k].set_xlabel('Mean Tonal Center Mean', fontsize=14)
        axes1.reshape(-1)[k].legend([tonnetz_labels[k//2]], fontsize=14)
    else:
        axes1.reshape(-1)[k].set_xlabel('Mean Tonal Center Var', fontsize=14)
        axes1.reshape(-1)[k].legend([tonnetz_labels[k//2]], fontsize=14)

plt.suptitle('Mean and Variance of Tonnetz Data\n\n', fontsize=20)
plt.tight_layout()
plt.show()
```




The above plots show the mean mean and mean variance of each of several tones across the eight genres.

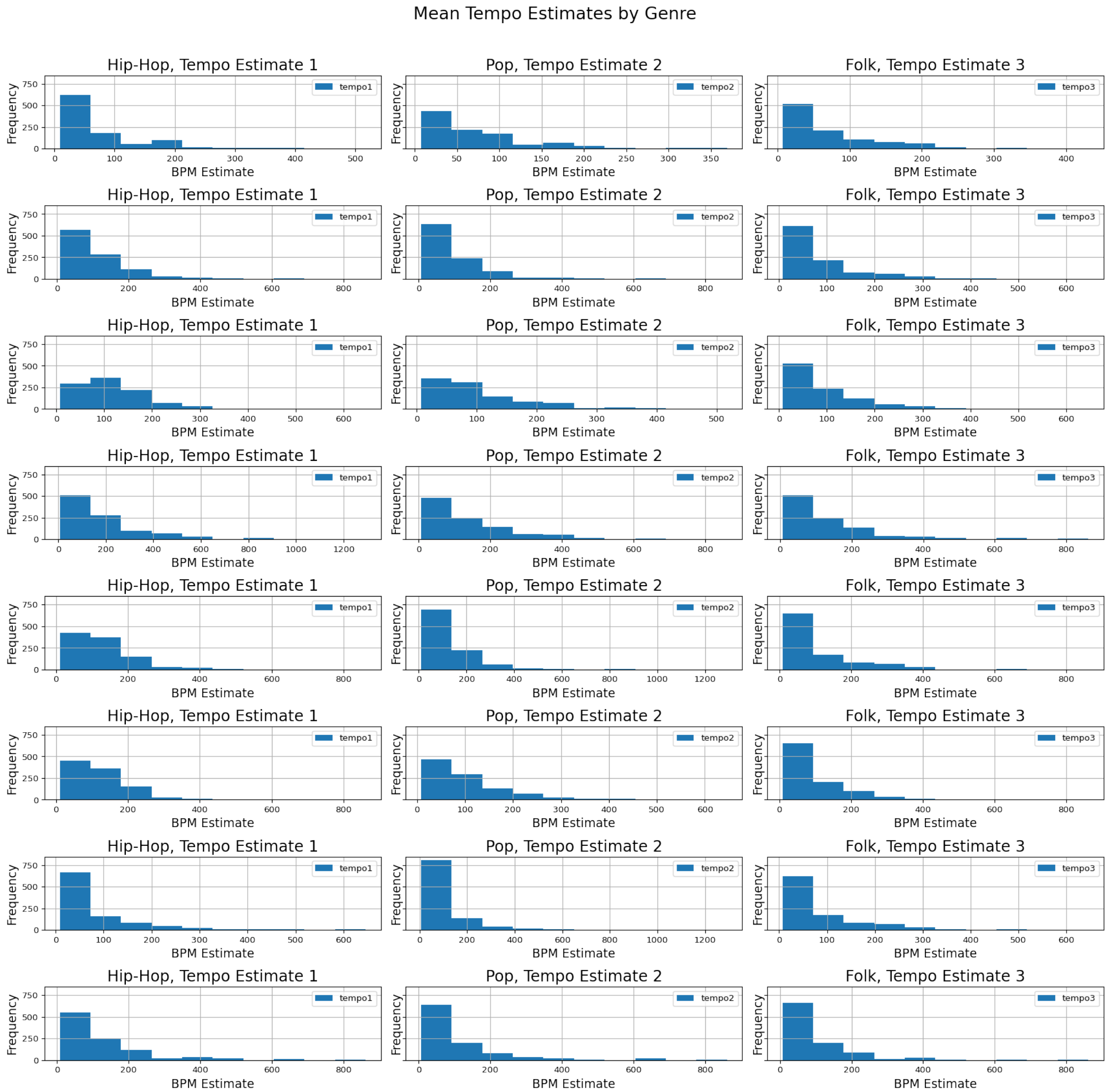
From these plots, we see first of all that the mean of the variances for each genre are almost identical for each of the different tones. This indicates that our data will be collinear here, and we would probably do well to drop all (or all but one) of the variances from our dataset.

For the means, we note that for many of these tones, the average tonal center is negative for some genres, and positive for others. Which tones are positive and negative changes for each tone, indicating that the mean tonal center data could be useful in making decisions between genres.

For example, for the major third x-axis mean tonal centers, folk music is strongly negative, while the rest of the genres are positive. Thus, this particular feature could be useful for classifying folk vs. not folk, and a similar idea could be used to interpret the other tonal features.

[12]:

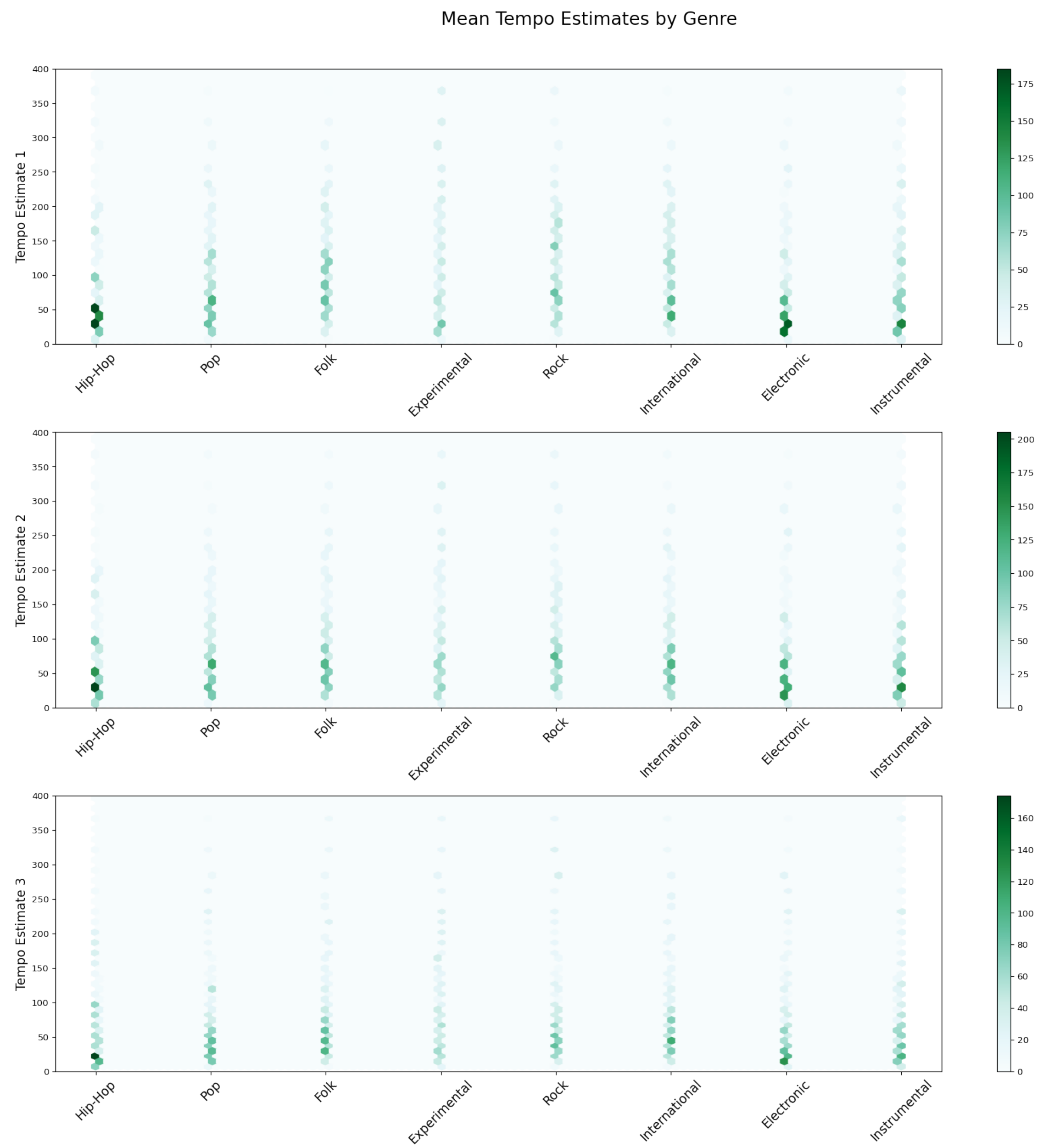
```
# Get the tempo features in their own dataframe
tempo_features = data[['genre_code', 'tempo1', 'tempo2', 'tempo3']]

# Group by genre
group = tempo_features.groupby('genre_code')

# Make a density plot
fig2, axes2 = plt.subplots(8, 3, sharey=True)
for j in range(8):
    for k in range(3):
        group.get_group(j).hist(column='tempo'+str(k+1), ax=axes2[j, k],␣
↪legend=True)
        axes2[j, k].set_xlabel('BPM Estimate', fontsize=14)
        axes2[j, k].set_ylabel('Frequency', fontsize=14)
        axes2[j, k].set_title(genre_labels[k] + ', Tempo Estimate ' + str(k+1),␣
↪fontsize=18)

plt.suptitle('Mean Tempo Estimates by Genre\n\n', fontsize=20)
plt.tight_layout()
plt.show()
```




The above plots show histograms for top three tempo estimates for each genre. These correspond to the three highest peaks on the tempo autocorrelation curve found using the librosa algorithms. For a given song, the first estimate, corresponding to the highest autocorrelation curve, is the guess for the average BPM for that song. The second and third highest peaks serve to provide more information about the tempo data for the song, and are especially useful if they are close to the height of the highest peak.

Though the histograms for each genre and estimate have roughly the same shape, we see some variation from genre to genre, and even from estimate to estimate. While it would appear that these tempo features are not by any means the most useful for classifying by genre, they at least show some variation. We will likely conduct some principal component analysis/dimension reduction before arriving at our final model, and that will help us decide if the tempo data visualized above is worth keeping.

[13]:
```
# Make a hexbin
fig3, axes3 = plt.subplots(3, 1)
for k in range(3):
    tempo_features.plot(ylim=[0, 400], x='genre_code', y='tempo'+str(k+1),␣
 ↪kind='hexbin', ax=axes3[k])
    axes3[k].set_xticks([x for x in range(8)])
    axes3[k].set_xticklabels(genre_labels, rotation=45, fontsize=14)
    axes3[k].set_ylabel('Tempo Estimate ' + str(k+1), fontsize=14)
    axes3[k].set_xlabel('')

plt.suptitle('Mean Tempo Estimates by Genre\n\n', fontsize=20)
plt.tight_layout()
plt.show()
```



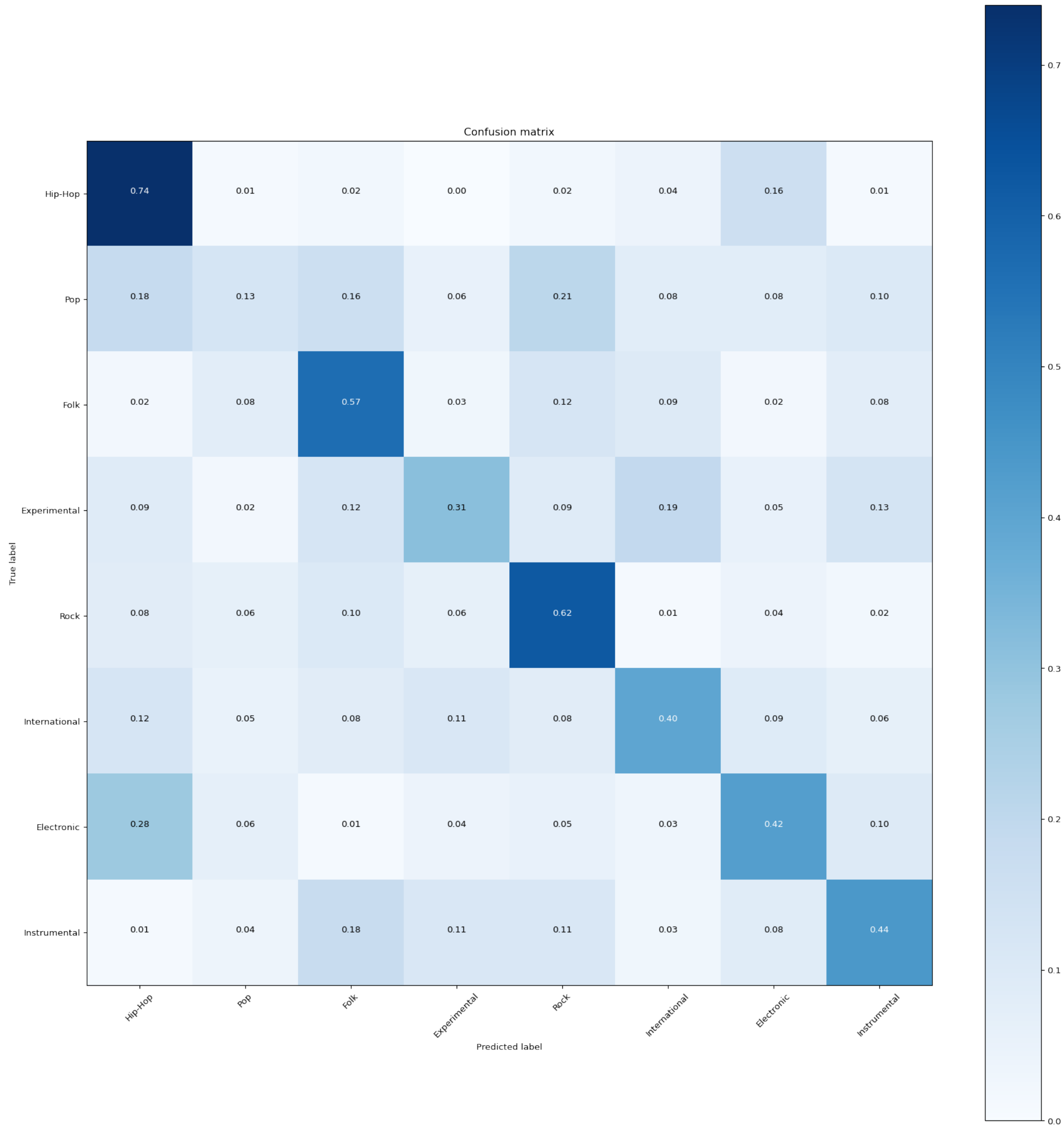

This is just another visualization of the tempo data, and is perhaps more interprative than the histograms above. Here, we see that instrumental, electronic, and hip-hop songs appear to have a stronger clustering of tempo estimates at the lower/slower end of the spectrum, which could indicate that the tempo data may be useful for classification.

We may also have some collinearities here, and it might be beneficial to remove the second and third tempo estimates from the dataset to clean up our model.

## 4.2 Models

```
[14]: logistic_regression(plot_matrix=True, test_size=.1, normalize=True)
```

```
Accuracy: 0.455
```




```
[14]: LogisticRegression()
```

## 5 Conclusion

Music classification is hard. Deep learning may be useful.

## 6 Bibliography


(1) G. Tzanetakis and P. Cook. Musical genre classification of audio signals. IEEE Transactions on Speech and Audio Processing, 10(5):293–302, July 2002.

(2) D. Perrot and R. Gjerdigen, “Scanning the dial: An exploration of factors in identification of musical style,” in Proc. Soc. Music Perception Cognition, 1999, p. 88

(3) Mingwen Dong. Convolutional neural network achieves human-level accuracy in music genre classification. CoRR, abs/1802.09697, 2018