Homework 3: Music Genre Identification

Eunji Lindsey Lee
(*e-mail:* elee0025@uw.edu)

Abstract

For this homework, we discussed the performance of classifying a given piece of music by sampling a five-second music clip, using supervised learning classification. This paper explores visual musical sound analysis by plotting wave-plots and spectrograms, and perform various algorithms of supervised learning classification. It then evaluates each methods through its accuracy scores on training and testing sets of the data.

Contents

1 Introduction and Overview

Our brains are outperforming at instantly recognizing the music artist (assuming that we know the artist) or its music genre upon hearing a new piece of music. Although song genres are quite subjective and sometimes the genres of the song are mixed and there exist multiple sub-genres, we believe there always exists one top genre that dominates the others.

Music genre classification is a popular topic in machine learning field and its techniques are still developing. For this assignment, we will explore the basics of music genre classification by performing three main tests on 5-second clips. First test is to choose three bands of different genres, and use machine learning to train sets and be able to correctly identify ”new” 5-second clips of music from three chosen bands. For second test, similar classification is done on different three bands of choice, where three chosen bands are of the same genre. Lastly, choose various bands from three chosen genres to do the genre

classification.

To start with this project, choosing the appropriate data set was critical. Unlike popularity of the topic, there were not much data sets together with a metadata openly accessible to the public for music analysis, largely due to legal issues. We chose a dataset previously collected from Free Music Archive (FMA), an interactive library of high-quality, legal audio downloads, by the researcher [**3**]. Instead of using its full dataset which consists of 106,572 untrimmed tracks of 161 genres, we used the small subset of it, which only consists of 8000 mp3 files of 30 seconds and of 8 genres. Moreover, its metadata includes tracks information and track IDs, which are the file names, so the use of metadata makes it easier for us to extract our songs of choice from the dataset.

Analyses done for the project used python and other data analysis tools such as *panda* and *scikit-learn*, a machine learning library for Python. The use of such free software tools made the overall analyses easier and accurate.

2 Theoretical Background

2.1 Audio And Musical Sound Analysis

Music genre recognition and classification is the process of extracting useful information from the audio signals and lyrics, then identifying their patterns from a large amount of data which differentiates one genre to the other. Although musical genre recognition and classification through lyrics is an increasing research field, especially in Natural Language Processing field (NLP), for this particular project, it only focuses on the audio signals.

Audio signals in a digital format is a sequence of bits whose values correspond to the sound-pressure levels in a waveform. [**4**] Wave plots and spectrograms visualize such digitized sound signals. From such audio signals of our dataset, we needed to extract useful features for an analysis. Instead of simply doing a Singular Value Decomposition (SVD) on a collection of vectors of audio signals and use a decomposed matrix of V as a feature matrix (A collection of vectors form a matrix, which can be decomposed into $U\Sigma V^*$), we used several critical features of musical sound to extract them specifically, out of audio signals [**2**]:

- mfcc: Mel-frequency cepstral coefficients, which is used to approximate human perception of non-linearly spaced sound bands. [**4**]
- chorma-stft: a chromagram from a waveform or power spectrogram, which captures 12 distinct pitches. [**5**]
- spectral contrast: a spectral contrast, which represents the musical octaves.
- tonnetz: a tonal centroid features, for musical tones.

Like SVD, such a feature extraction is critical for preprocessing techniques as it removes a lot of redundant information. Furthermore, the process reduces a dimensionality of the dataset and makes incomparable raw dataset into a collection of comparable feature vectors for classification.

2.2 Supervised Learning: Classification

Classification is a type of supervised learning, which is to train labeled dataset to predict the label for a new set of data. In order to build an accurately predictive model, the model has to be generalized to an independent dataset. Hence, cross-validation technique, which splits the dataset into training and testing such that model is trained and already tested (i.e. validated) with labeled data, is critical. Cross validation significantly reduces the danger of overfitting, which is usually the main problem in building an accurate machine learning model. There are several classification algorithms to explore.

2.2.1 K-nearest Neighbours (KNN)

The K-nearest Neighbours classification is to simply classify new data based on its closeness to k nearest instances of data. Usually Euclidean or Mahalanobis distance metrics are used to calculate the closeness between all pairs of data points, and then new data points are classified to the most common class label. KNN algorithm is one of the simplest algorithms, but it can suffer from skewed distributions if a testing dataset includes a certain label that is abnormally frequent.

2.2.2 Naive Bayes (NB)

Naive Bayes is to classify new data based on Bayes theorem, which assumes that data features are independent of one another. Naive Bayes classification weighs all features equally in making decisions and works surprisingly well in high dimensions. However due to its underlying assumption, if the strong independence assumption is not met, then it performs badly.

2.2.3 Support Vector Machines (SVM)

Support Vector Classification performs by finding a hyper-plane that optimally classifies a higher-dimensional feature space into two categories. Thus, it categorizes one target label from other labels by finding an optimal hyper-plane, which has the greatest possible margin between the hyper-plane and training data points. SVM work efficiently with smaller datasets and is sensitive to noises.

2.2.4 Decision Trees

Decision Tree classification is a binary tree classification method that consistently splits training dataset, which starts as a root node, according to an attribute that gives the maximal information gain from its sub-tree until leaf node (labels) is reached.

2.2.5 Adaptive Boosting or Adaboost

Adaboost is one of Boosting methods, which creates a strong classifier by combining several weak learners. Adaboost is a binary classification, and each weak learners can be combined using weighted averages. For each iterations of Adaboost, it weighs more on incorrectly predicted data points to learn more from them.

3 Algorithm Implementation and Development

To retrieve audio signals from mp3 files, we used *librosa* python library for music and audio analysis, which reads audio signals and returns a waveform as a vector from the designated duration of 5 seconds. For each tests, we collected waveform vectors and extracted features from *librosa* library for each song. While collecting feature vectors for each song, we encoded the label for each genres or bands separately as integer values in a label vector. For example, for test 1, we used ”0” to label band, Phemale, used ”1” to label band, Big Blood and ”2” to label band, Black Sea Hotel.

Before performing classification, we wanted to visualize wave plots and spectrograms to observe if there are obvious differences between genres or between bands. Hence, we plotted five different songs from each bands for each plots. We also plotted PCA graph to see how each categories are distributed in 2-dimensional graph.

Now, from these collection of features and labels, we performed supervised learning. Before training feature and label data to do the music classification, we used cross validation to separate data into 75 percent for training and 25 percent for testing set. Then we trained the model with the training set with various classification algorithms such as Support Vector Machines (SVM) and Naive Bayes and explored so as to come up with the best predictive model. All machine learning algorithms used are from *sklearn* library, a famous machine learning library for python programming used by many machine learning engineers nowadays.

4 Computational Results

For test 1, we randomly chose classification algorithms to do the classification and adjusted their parameters carefully to yield the best accuracy score on testing data. Among support vector classification, naive bayes classification and k-nearest neighbor classification, support vector classification resulted in the highest accuracy scores on our testing dataset, which was optained from cross-validation. As observed from the confuction matrix, the band prediction for the band ”Big Blood” outperforms the other two bands. This result contracdicts our expection when we were observing wave plots and spectrograms of bands. We expected to have a better performance of classification for all bands, since unique patterns for each bands were observed. One of the reasons that ”Big Blood” outperforms other two bands might be due to the large number of dataset available for classification for the band ”Big Blood” compared to other bands.

For test 2, we also randomly chose classification algorithms to do the classification and adjusted their parameters carefully to yield the best accuracy score on testing data. Among support vector classification, naive bayes classification and k-nearest neighbor classification, support vector classification resulted in the highest accuracy scores on our testing dataset, which was optained from cross-validation. As observed from the confuction matrix, the band prediction for the band ”Big Blood” outperforms the other two bands. This result contracdicts our expection when we were observing wave plots and spectro-

grams of bands. We expected to have a better performance of classification for all bands, since unique patterns for each bands were observed. One of the reasons that "Big Blood" outperforms other two bands might be due to the large number of dataset available for classification for the band "Big Blood" compared to other bands.

Lastly, for test 3, an accuracy score of 0.91 for the gradient boosting classification was the highest. Overall, there are very little mis-predictions from all classification algorithms, and the accuracy scores are generally high for all classification algorithms. Test 3 has the most datasets used for genre classification training and testing. Larger datasets resulted in more precise and accurate classification model.

5 Summary and Conclusions

Classification model we built for tests 1, 2 and 3 were pretty accurate in predicting new 5-second clips of music. We randomly chose about three classification algorithms and compared them. Parameters used for each classification algorithms were adjusted carefully so that it produces the highest accuracy score for each test sets. Interestingly, there were different classification algorithms that work better in making predictions for each tests. Generally, the more datasets we have, the more accurate classification model we have obtained (though having too large dataset would not make a large effect).

From these analyses, we learned that every bands of different genres or same genres have their own unique musical sound features and they can be identified even just with first five-seconds of their music. One of the most interesting findings from this paper was that the classification of music from bands of same genres were more accurate than bands of different genres. We expected that the bands of same genres would be more difficult to be distinguished from one another. Above all, the best classification prediction model was music genre classification. This paper supports that there are distinguishing musical features among different musical genres, especially among genres of our choice–Pop, Folk and International. We believe this paper can be explored further to build more extensive classification model for more variety of music (of many genres and bands).

6 Appendix

6.1 Jupyter Notebook for codings

Computational Results

In [1]:
```
%matplotlib inline

import os

import IPython.display as ipd
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import librosa
import librosa.display
import sklearn.svm
from sklearn.decomposition import PCA
from sklearn.cross_validation import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, fbeta_score

# Import supplementary utils.py for an easy access to dataset
import utils
```

```
/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-packages/sklearn/cross_validatio
n.py:41: DeprecationWarning: This module was deprecated in version 0.18 in favor of the model_selection mo
dule into which all the refactored classes and functions are moved. Also note that the interface of the new CV iter
ators are different from that of this module. This module will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
```

Load Metadata

In [2]:
```
# Directory where mp3 are stored.
AUDIO_DIR = '/Users/eunji/Downloads/Winter2018/AMATH582/HW3/fma_small'

# Load metadata and features.
tracks = utils.load('tracks.csv')
```

```
/Users/eunji/Downloads/Winter2018/AMATH582/HW3/utils.py:213: FutureWarning: specifying 'categories' or 'ord
ered' in .astype() is deprecated; pass a CategoricalDtype instead
  'category', categories=SUBSETS, ordered=True)
```

Select Small Dataset

These small subset of data contains 8000 mp3 files of 30 seconds and of 8 genres. We will use metadata ('track.csv' file) to easily access to music clips and its information.

In [3]:
```
small = tracks[tracks['set', 'subset']=='small']
small.shape
```

Out[3]:
```
(8000, 52)
```

In [129]:
```
ipd.display(small['track'].head())
```

	bit_rate	comments	composer	date_created	date_recorded	duration	favorites	genre_top	genres	genres_all	informati
track_id											
2	256000	0	NaN	2008-11-26 01:48:12	2008-11-26	168	2	Hip-Hop	[21]	[21]	NaN
5	256000	0	NaN	2008-11-26 01:48:20	2008-11-26	206	6	Hip-Hop	[21]	[21]	NaN
10	192000	0	Kurt Vile	2008-11-25 17:49:06	2008-11-26	161	178	Pop	[10]	[10]	NaN
140	128000	0	NaN	2008-11-26 01:44:07	2008-11-26	253	5	Folk	[17]	[17]	NaN
141	128000	0	NaN	2008-11-26 01:44:10	2008-11-26	182	1	Folk	[17]	[17]	NaN

Get all genres from Small Dataset

These are the 8 genres of music contained in the small subset of data

In [4]:
```
TopGenres = pd.Series(small['track','genre_____top'].unique())
print(TopGenres)
```

```
0       Hip-Hop
1           Pop
2          Folk
3  Experimental
4          Rock
5  International
6    Electronic
7    Instrumental
dtype: category
Categories (8, object): [Hip-Hop, Pop, Folk, Experimental, Rock, International, Electronic, Instrumental]
```

(test 1) Band Classification

Consider three different bands of your choosing and of different genres. For instance, one could pick Michael Jackson, Soundgarden, and Beethoven. By taking 5-second clips from a variety of each of their music, i.e. building training sets, see if you can build a statistical testing algorithm capable of accurately identifying ”new” 5-second clips of music from the three chosen bands.

In [268]:
```
# Return the number of non-existent file from the given list
def checkIfFilesExist(list):
    size = 0;
    for i in range(len(list)):
        filename = utils.get_____audio_____path(AUDIO_____DIR, list[i])
        if (not os.path.isfile(filename)):
            size += 1
    return size
```

Get all lists of track ids for one Pop band

We chose the band "Phemale", and we have 41 music clips of theirs.

In [4]:
```
print(small[small['track','genre_____top']=='Pop']['artist','name'].value_____counts().head(5))
phemaleLists = small[small['artist','name']=='Phemale']
phemaleLists = phemaleLists.index.tolist()
```

```
Phemale         41
Garmisch        29
Fierbinteanu    28
Maxwell Powers  24
Plushgoolash    17
Name: (artist, name), dtype: int64
```

Get all lists of track ids for one Folk band

We chose "Big Blood". They have 91 music clips in our dataset.

In [5]:
```
print(small[small['track','genre_____top']=='Folk']['artist','name'].value_____counts().head(5))
BigBloodLists = small[small['artist','name']=='Big Blood']
BigBloodLists = BigBloodLists.index.tolist()
```

```
Big Blood       91
Derek Clegg     45
Josh Woodward   24
Plusplus        19
Kelly Latimore  19
Name: (artist, name), dtype: int64
```

Get all lists of track ids for one International band

We chose the band "Black Sea Hotel", who contains 23 music clips in our dataset.

In [6]:
```
print(small[small['track','genre_____top']=='International']['artist','name'].value_____counts().head(5))
BlackSeaHotelLists = small[small['artist','name']=='Black Sea Hotel']
BlackSeaHotelLists = BlackSeaHotelLists.index.tolist()
```

```
Turku, Nomads of the Silk Road   29
Sláinte                          27
Black Sea Hotel                  23
The Sounds of Taraab             21
Gogofski                         19
Name: (artist, name), dtype: int64
```

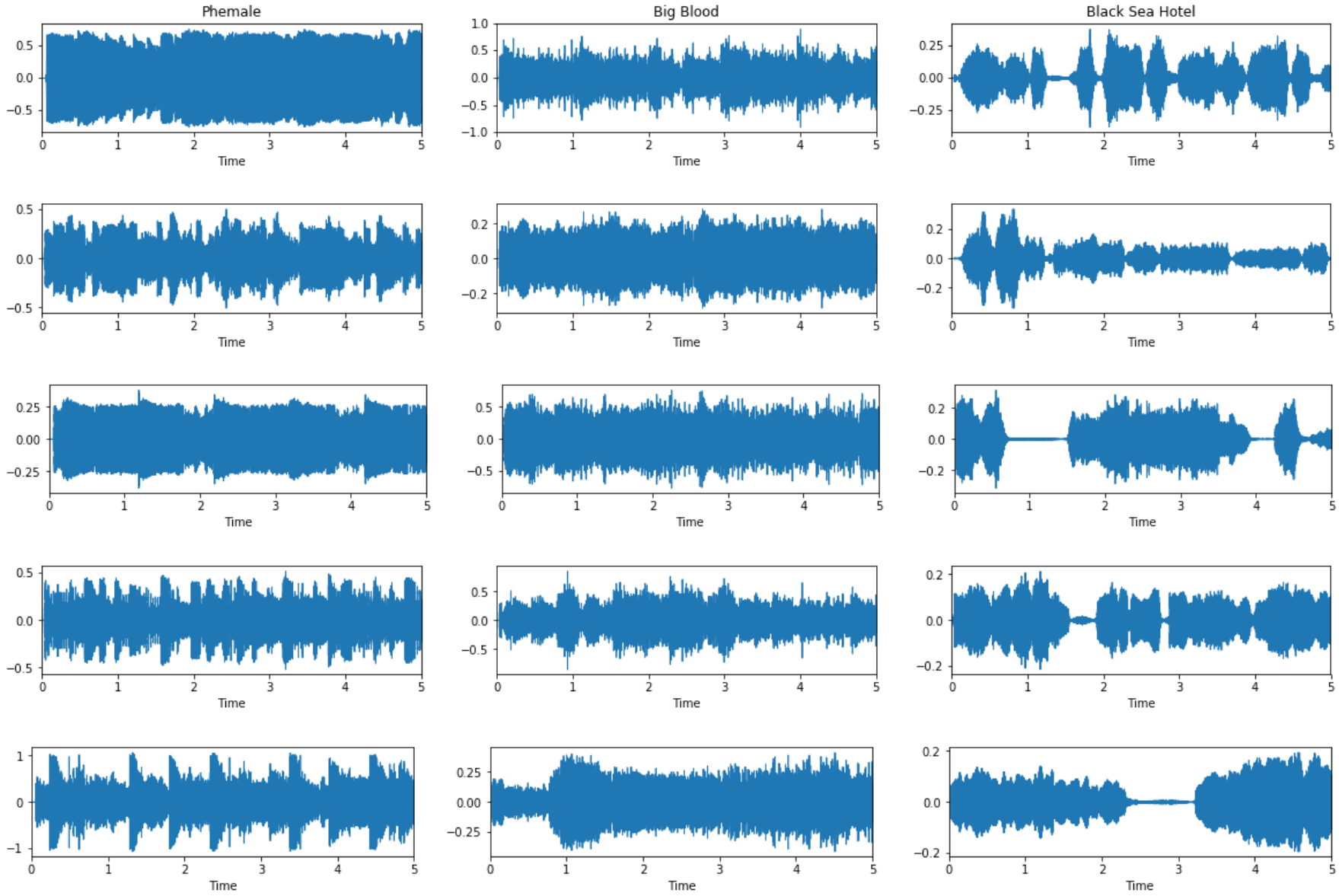
Retrieve Audio

In [179]:
```
# Get 5-sec audio arrays
def getX(id):
    filename = utils.get_____audio_____path(AUDIO_____DIR, id)
    # x = audio time series, sr = sampling rate of 'x'
    x, sr = librosa.load(filename, duration=5)
    return x

# default value
sr = 22050
```

VISUALIZATION: Waveplots & Spectrograms

In [136]:
```
def waveplot(x):
    librosa.display.waveplot(x)

def plot_____specgram(x):
    plt.specgram(np.array(x), Fs=sr)
    plt.colorbar(format='%+2.0f dB')

def plot_____log_____power_____specgram(x):
    D = librosa.logamplitude(np.abs(librosa.stft(x))**2, ref_____power=np.max)
    librosa.display.specshow(D, x_____axis='time', y_____axis='log')
    plt.colorbar(format='%+2.0f dB')
```

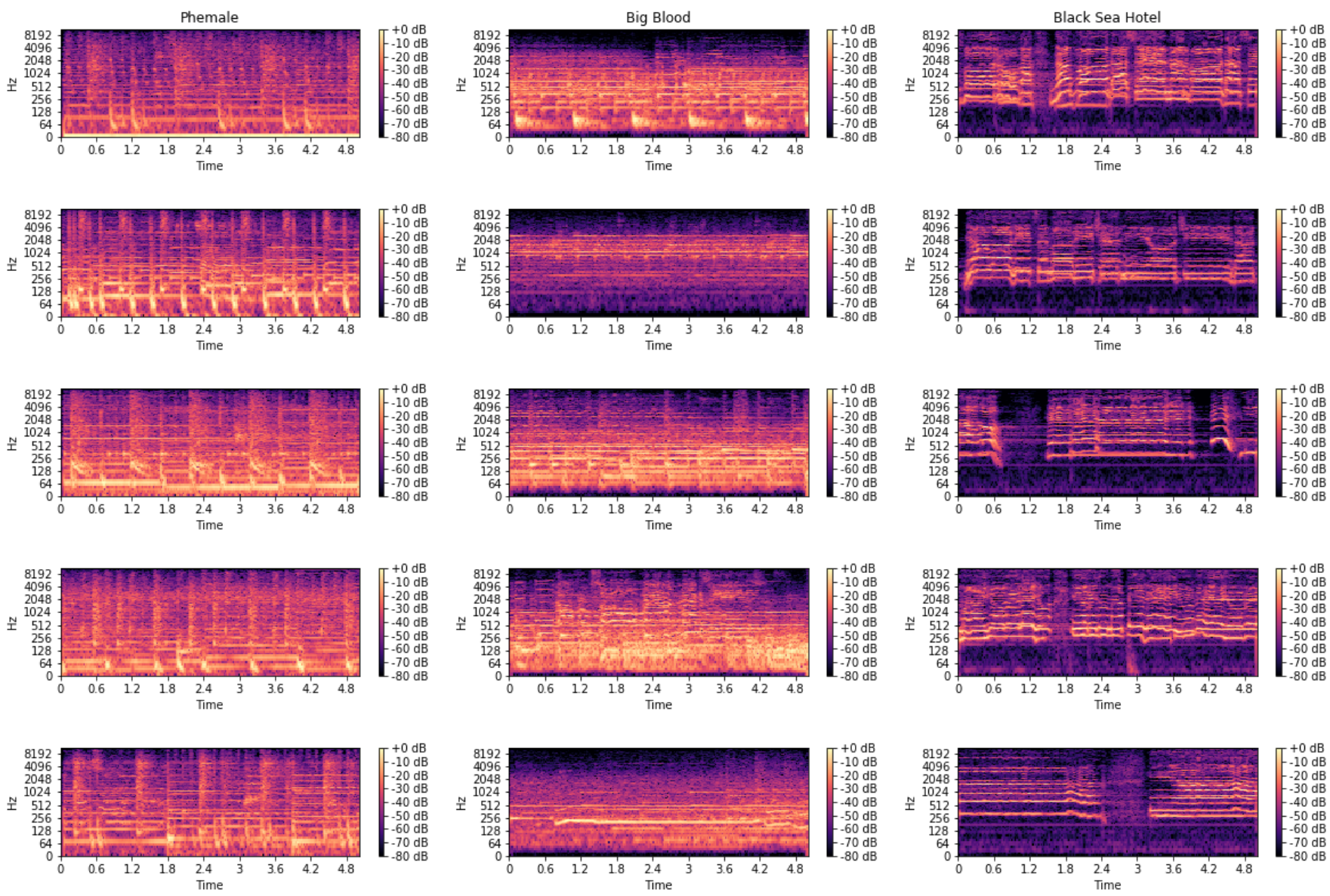
In [521]:
```
# Wave Plots
for i in range(5):
   plt.figure(figsize=(20,10))
   plt.subplot(5,3,i*3+1), waveplot(getX(phemaleLists[i])), plt.title('Phemale') if i == 0 else 0
   plt.subplot(5,3,i*3+2), waveplot(getX(BigBloodLists[i])), plt.title('Big Blood') if i == 0 else 0
   plt.subplot(5,3,i*3+3), waveplot(getX(BlackSeaHotelLists[i])), plt.title('Black Sea Hotel') if i == 0 else 0
```



In [520]:
```
# Spectrograms
for i in range(5):
   plt.figure(figsize=(20,10))
   plt.subplot(5,3,i*3+1), plot_____specgram(getX(phemaleLists[i])), plt.title('Phemale') if i == 0 else 0
   plt.subplot(5,3,i*3+2), plot_____specgram(getX(BigBloodLists[i])), plt.title('Big Blood') if i == 0 else 0
   plt.subplot(5,3,i*3+3), plot_____specgram(getX(BlackSeaHotelLists[i])), plt.title('Black Sea Hotel') if i == 0 else 0
plt.show()
```

```
/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-packages/matplotlib/axes/_____axes.py:7221: RuntimeWarning: divide by zero encountered in log10
  Z = 10. * np.log10(spec)
```

In [519]:
```
# Log-Power Spectrograms
for i in range(5):
   plt.figure(figsize=(20,10))
   plt.subplot(5,3,i*3+1), plot_____log_____power_____specgram(getX(phemaleLists[i])), plt.title('Phemale') if i == 0
else 0
   plt.subplot(5,3,i*3+2), plot_____log_____power_____specgram(getX(BigBloodLists[i])), plt.title('Big Blood') if i ==
0 else 0
   plt.subplot(5,3,i*3+3), plot_____log_____power_____specgram(getX(BlackSeaHotelLists[i])), plt.title('Black Sea
 Hotel') if i == 0 else 0
plt.show()
```



Each columns of plots represents different bands, "Phemale", "Big Blood" and "Black Sea Hotel". There are some distinguishing patterns observed for each bands. We expect to have a fairly accurate classifiation among bands of different genres, according to these plots above.

Extract Features

In [73]:
```
def extract_____features(X):
   stft = np.abs(librosa.stft(X))
   mfccs = np.mean(librosa.feature.mfcc(y=X, sr=sr, n_____mfcc=10).T,axis=0)
   chroma = np.mean(librosa.feature.chroma_____stft(S=stft, sr=sr).T,axis=0)
   mel = np.mean(librosa.feature.melspectrogram(X, sr=sr).T,axis=0)
   contrast = np.mean(librosa.feature.spectral_____contrast(S=stft, sr=sr).T,axis=0)
   tonnetz = np.mean(librosa.feature.tonnetz(y=librosa.effects.harmonic(X),
               sr=sr).T,axis=0)
   return mfccs,chroma,mel,contrast,tonnetz
```

Feature Matrix for all bands

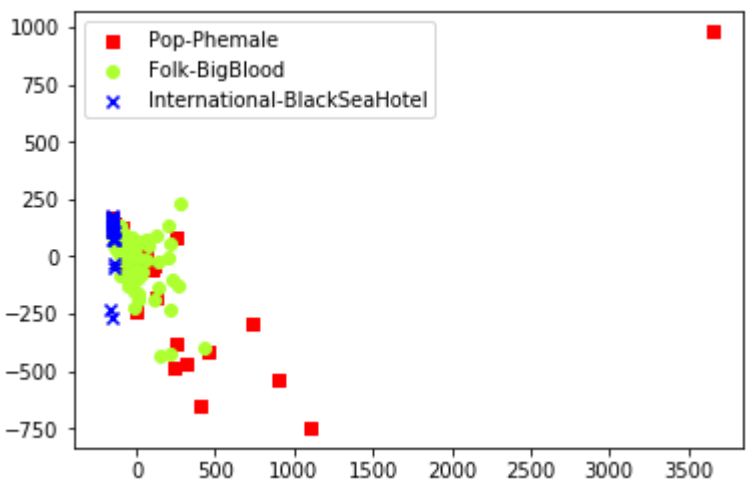
In [369]:
```
features1, labels1 = [], []
def parseList(featurelist, labellist, lists, code):
   for i in range(len(lists)):
      mfccs, chroma, mel, contrast,tonnetz = extract_____features(getX(lists[i]))
      ext_____features = np.hstack([mfccs,chroma,mel,contrast,tonnetz])
      featurelist.append(ext_____features)
      labellist.append(code)

parseList(features1, labels1, phemaleLists, 0)
parseList(features1, labels1, BigBloodLists, 1)
parseList(features1, labels1, BlackSeaHotelLists, 2)

features1 = np.array(features1)
labels1 = np.array(labels1)
```

PCA

The following PCA graph plots each bands' feature matrix in 2-dimensional space. There are some overlaps among bands on the PCA graph.

In [159]:
```
pca = PCA(n_____components=2).fit(features1)
pca_____2d = pca.transform(features1)
colormarkers = [ ['red','s'], ['greenyellow','o'], ['blue','x']]
for i in range(len(colormarkers)):
   px = pca_____2d[:, 0][labels1 == i]
   py = pca_____2d[:, 1][labels1 == i]
   plt.scatter(px, py, c=colormarkers[i][0], marker=colormarkers[i][1])
plt.legend(['Pop-Phemale','Folk-BigBlood','International-BlackSeaHotel'])
```

Out[159]: <matplotlib.legend.Legend at 0x122fa9978>



Classification

In [370]:
```
# Cross validation
# Split into a training and testing set
X1_____train, X1_____test, y1_____train, y1_____test = train_____test_____split(features1, labels1, test_____size=0.2
5, random_____state=1)
print ("Training set has {} samples.".format(X1_____train.shape[0]))
print ("Testing set has {} samples.".format(X1_____test.shape[0]))
```

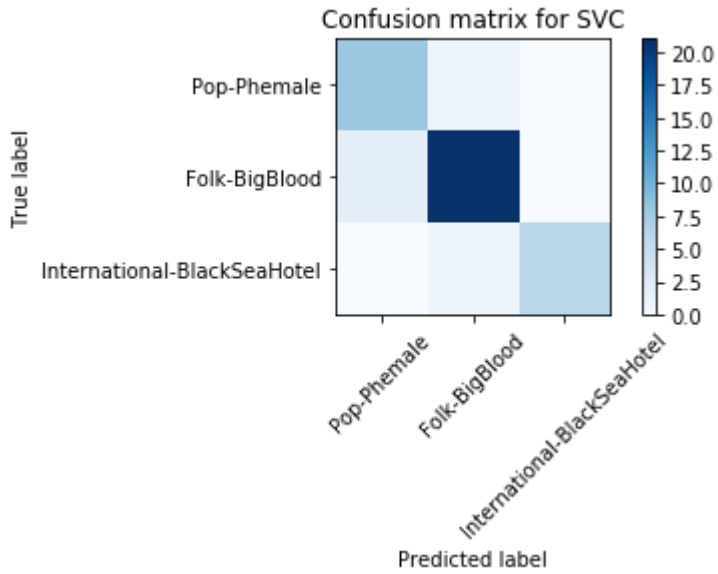
Training set has 116 samples.
Testing set has 39 samples.

In [213]:
```
## The confusion matrix provides more detailed information about Classifier performance.
def plot_____confusion_____matrix(cm, title, lists, cmap=plt.cm.Blues):
   plt.imshow(cm, interpolation='nearest', cmap=cmap)
   plt.title(title)
   plt.colorbar()
   tick_____marks = np.arange(len(lists))
   plt.xticks(tick_____marks, lists, rotation=45)
   plt.yticks(tick_____marks, lists)
   plt.tight_____layout()
   plt.ylabel('True label')
   plt.xlabel('Predicted label')
```

Support Vector Classification

In [371]:
```
clf = svm.SVC(C=1, kernel='poly')
clf.fit(X1_____train,y1_____train)

y1_____pred = clf.predict(X1_____test)
train_____acc = accuracy_____score(y1_____train,clf.predict(X1_____train))

print ("Accuracy score on training data: {:.4f}".format(train_____acc))
print ("Accuracy score on testing data: {:.4f}".format(accuracy_____score(y1_____test, y1_____pred)))

plt.figure()
cm = confusion_____matrix(y1_____test, y1_____pred)
bandsList = ['Pop-Phemale','Folk-BigBlood','International-BlackSeaHotel']
plot_____confusion_____matrix(cm, 'Confusion matrix for SVC', bandsList)
```

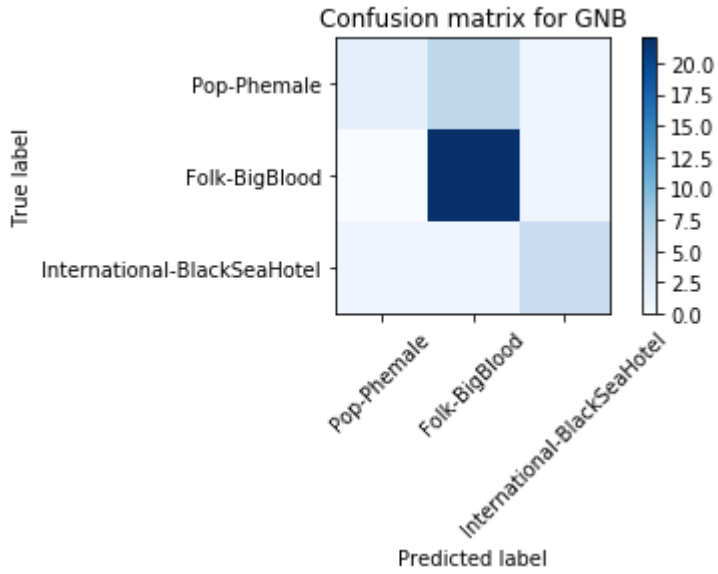
Accuracy score on training data: 1.0000
Accuracy score on testing data: 0.8974



Naive Bayes Classification

In [372]:
```
model = GaussianNB().fit(X1_____train, y1_____train)

# Make predictions
y1_____pred = model.predict(X1_____test)
train_____acc = accuracy_____score(y1_____train,clf.predict(X1_____train))

print ("Accuracy score on training data: {:.4f}".format(train_____acc))
print ("Accuracy score on testing data: {:.4f}".format(accuracy_____score(y1_____test, y1_____pred)))

plt.figure()
cm = confusion_____matrix(y1_____test, y1_____pred)
plot_____confusion_____matrix(cm, 'Confusion matrix for GNB', bandsList)
```

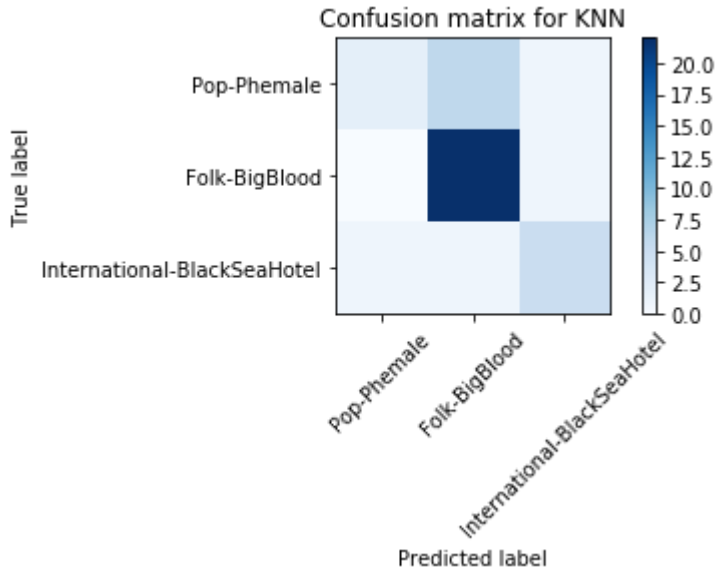
Accuracy score on training data: 1.0000
Accuracy score on testing data: 0.7436



KNearest Neighbors Classification

In [373]:
```
neigh = KNeighborsClassifier(n_____neighbors=3).fit(X1_____train, y1_____train)

# Make predictions
y1_____pred = model.predict(X1_____test)
train_____acc = accuracy_____score(y1_____train,clf.predict(X1_____train))

print ("Accuracy score on training data: {:.4f}".format(train_____acc))
print ("Accuracy score on testing data: {:.4f}".format(accuracy_____score(y1_____test, y1_____pred)))

plt.figure()
cm = confusion_____matrix(y1_____test, y1_____pred)
plot_____confusion_____matrix(cm, 'Confusion matrix for KNN', bandsList)
```

```
Accuracy score on training data: 1.0000
Accuracy score on testing data: 0.7436
```



Result

We randomly chose classification algorithms to do the classification and adjusted their parameters carefully to yield the best accuracy score on testing data. Among support vector classification, naive bayes classification and k-nearest neighbor classification, support vector classification resulted in the highest accuracy scores on our testing dataset, which was optained from cross-validation. As observed from the confuction matrix, the band prediction for the band "Big Blood" outperforms the other two bands. This result contracdicts our expection when we were observing wave plots and spectrograms of bands. We expected to have a better performance of classification for all bands, since unique patterns for each bands were observed. One of the reasons that "Big Blood" outperforms other two bands might be due to the large number of dataset available for classification for the band "Big Blood" compared to other bands.

(test 2) The Case for Seattle

Repeat the above experiment, but with three bands from within the same genre. This makes the testing and separation much more challenging. For instance, one could focus on the late 90s Seattle grunge bands: Soundgarden, Alice in Chains, and Pearl Jam. What is your accuracy in correctly classifying a 5-second sound clip? Compare this with the first experiment with bands of different genres.

Get all lists of track ids for the other two Pop bands

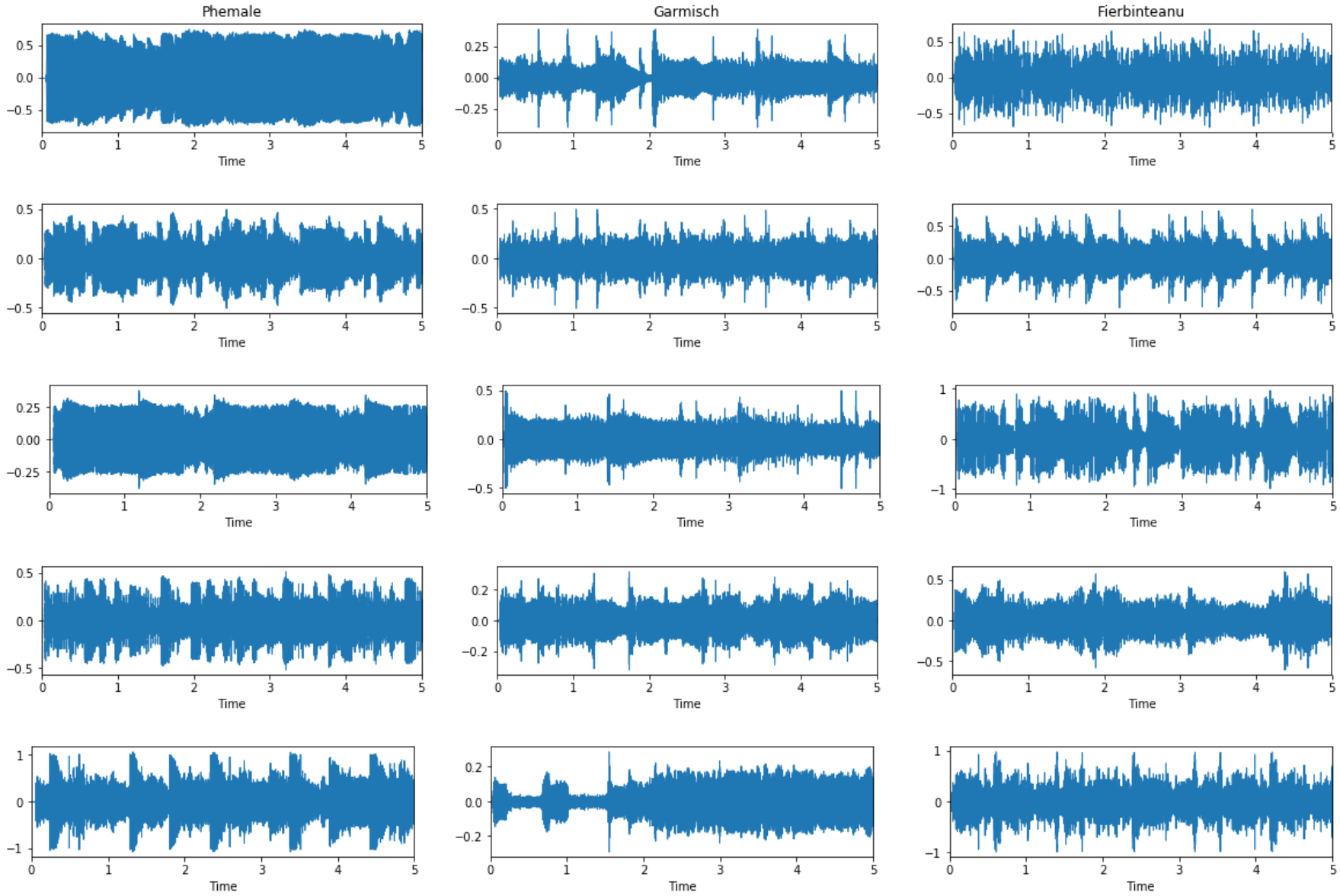
Other than the Pop band chosen earlier, "Phemale", we chose "Garmisch" and "Fierbinteanu" for other two Pop bands.

In [419]:
```
print(small[small['track','genre_____top']=='Pop']['artist','name'].value_____counts().head(5))
GarmischLists = small[small['artist','name']=='Garmisch']
GarmischLists = GarmischLists.index.tolist()
print(checkIfFilesExist(GarmischLists))

FierbinteanuLists = small[small['artist','name']=='Fierbinteanu']
FierbinteanuLists = FierbinteanuLists.index.tolist()
print(checkIfFilesExist(FierbinteanuLists))
```

```
Phemale         41
Garmisch        29
Fierbinteanu    28
Maxwell Powers  24
Plushgoolash    17
Name: (artist, name), dtype: int64
0
0
```

Visualization: Waveplots and Spectrograms

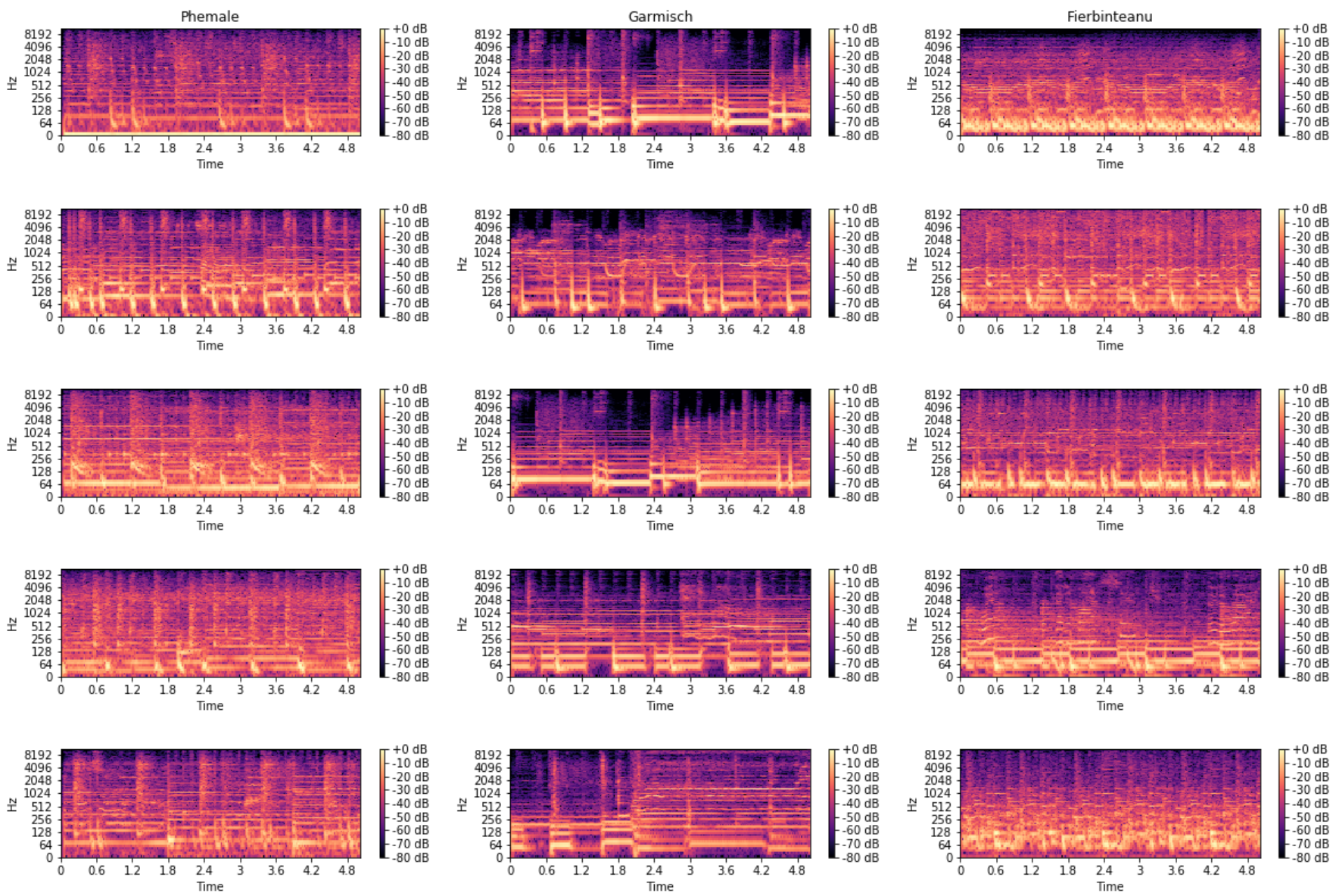
In [518]:
```
# Wave Plots
for i in range(5):
   plt.figure(figsize=(20,10))
   plt.subplot(5,3,i*3+1), waveplot(getX(phemaleLists[i])), plt.title('Phemale') if i == 0 else 0
   plt.subplot(5,3,i*3+2), waveplot(getX(GarmischLists[i])), plt.title('Garmisch') if i == 0 else 0
   plt.subplot(5,3,i*3+3), waveplot(getX(FierbinteanuLists[i])), plt.title('Fierbinteanu') if i == 0 else 0
```



In [517]:
```
# Spectrograms
for i in range(5):
   plt.figure(figsize=(20,10))
   plt.subplot(5,3,i*3+1), plot_____specgram(getX(phemaleLists[i])), plt.title('Phemale') if i == 0 else 0
   plt.subplot(5,3,i*3+2), plot_____specgram(getX(GarmischLists[i])), plt.title('Garmisch') if i == 0 else 0
   plt.subplot(5,3,i*3+3), plot_____specgram(getX(FierbinteanuLists[i])), plt.title('Fierbinteanu') if i == 0 else 0
plt.show()
```

```
/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-packages/matplotlib/axes/_____axes.p
y:7221: RuntimeWarning: divide by zero encountered in log10
  Z = 10. * np.log10(spec)
```

In [516]:
```
# Log-Power Spectrogram
for i in range(5):
    plt.figure(figsize=(20,10))
    plt.subplot(5,3,i*3+1), plot_____log_____power_____specgram(getX(phemaleLists[i])), plt.title('Phemale') if i == 0
else 0
    plt.subplot(5,3,i*3+2), plot_____log_____power_____specgram(getX(GarmischLists[i])), plt.title('Garmisch') if i ==
 0 else 0
    plt.subplot(5,3,i*3+3), plot_____log_____power_____specgram(getX(FierbinteanuLists[i])), plt.title('Fierbintean
u') if i == 0 else 0
plt.show()
```



Some distinguishing patterns are observed for each bands. Especially band "Garmisch", compared with the other two Pop bands, has some differentiable pattern from their log-power spectrograms.

Extract Features: calculate feature matrix

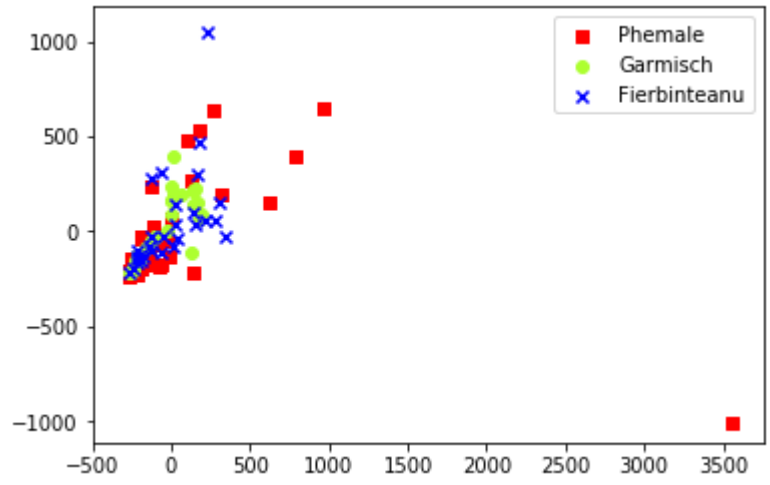
In [435]:
```
features2, labels2 = [], []

parseList(features2, labels2, phemaleLists, 0)
parseList(features2, labels2, GarmischLists, 1)
parseList(features2, labels2, FierbinteanuLists, 2)

features2 = np.array(features2)
labels2 = np.array(labels2)
```

PCA

There are lots of overlaps among bands, on the PCA graph below.

In [440]:
```
pca = PCA(n_____components=2).fit(features2)
pca_____2d = pca.transform(features2)
colormarkers = [ ['red','s'], ['greenyellow','o'], ['blue','x']]
for i in range(len(colormarkers)):
    px = pca_____2d[:, 0][labels2 == i]
    py = pca_____2d[:, 1][labels2 == i]
    plt.scatter(px, py, c=colormarkers[i][0], marker=colormarkers[i][1])
plt.legend(['Phemale','Garmisch','Fierbinteanu'])
```

Out[440]: <matplotlib.legend.Legend at 0x123a3bfd0>



Classification

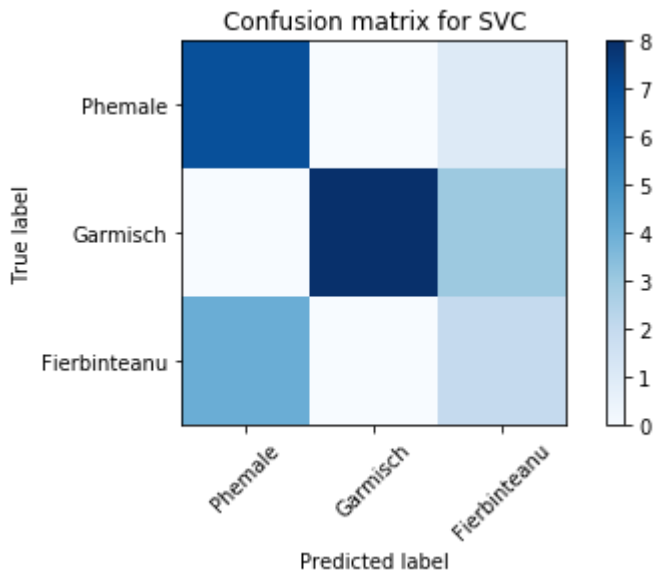
In [436]:
```
# Cross validation
# Split into a training and testing set
X2_____train, X2_____test, y2_____train, y2_____test = train_____test_____split(features2, labels2, test_____size=
0.25, random_____state=1)
print ("Training set has {} samples.".format(X2_____train.shape[0]))
print ("Testing set has {} samples.".format(X2_____test.shape[0]))
```

```
Training set has 73 samples.
Testing set has 25 samples.
```

Support Vector Classification

In [437]:
```
clf = svm.SVC(kernel='poly',degree=2)
clf.fit(X2_____train,y2_____train)

y2_____pred = clf.predict(X2_____test)
train_____acc = accuracy_____score(y2_____train,clf.predict(X2_____train))

print ("Accuracy score on training data: {:.4f}".format(train_____acc))
print ("Accuracy score on testing data: {:.4f}".format(accuracy_____score(y2_____test, y2_____pred)))

plt.figure()
popBandsList = ['Phemale','Garmisch','Fierbinteanu']
cm = confusion_____matrix(y2_____test, y2_____pred)
plot_____confusion_____matrix(cm, 'Confusion matrix for SVC', popBandsList)
```

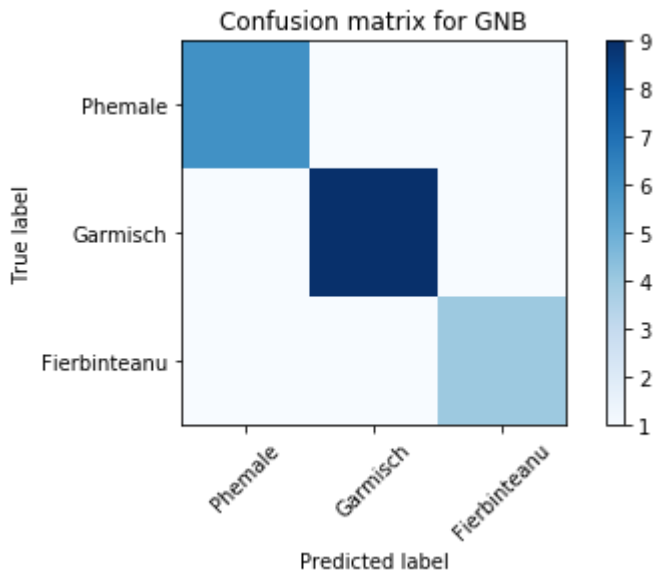
```
Accuracy score on training data: 1.0000
Accuracy score on testing data: 0.6800
```



Gradient Boosting Classification

In [438]:
```
model = GradientBoostingClassifier(learning_____rate=0.5, max_____depth=3, random_____state=0).fit(X2_____train, y2_____train)

# Make predictions
y2_____pred = model.predict(X2_____test)
train_____acc = accuracy_____score(y2_____train,clf.predict(X2_____train))

print ("Accuracy score on training data: {:.4f}".format(train_____acc))
print ("Accuracy score on testing data: {:.4f}".format(accuracy_____score(y2_____test, y2_____pred)))

plt.figure()
cm = confusion_____matrix(y2_____test, y2_____pred)
plot_____confusion_____matrix(cm, 'Confusion matrix for GNB', popBandsList)
```

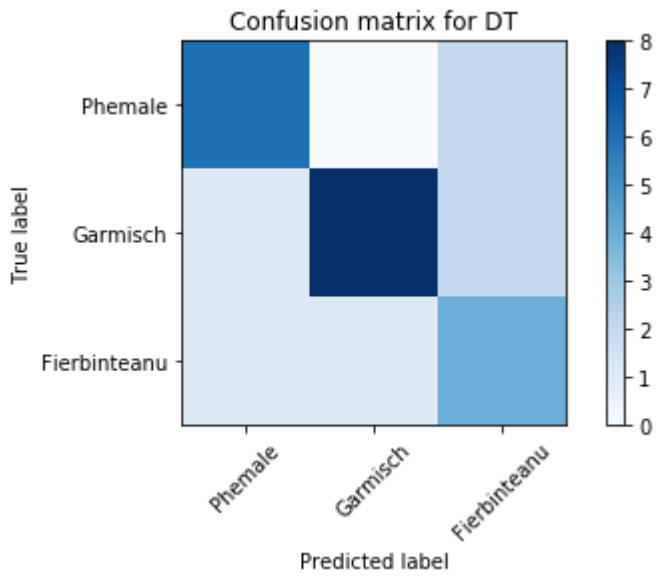
```
Accuracy score on training data: 1.0000
Accuracy score on testing data: 0.7600
```



Decision Tree Classification

In [439]:
```
model = DecisionTreeClassifier(max_____depth=3).fit(X2_____train, y2_____train)

# Make predictions
y2_____pred = model.predict(X2_____test)
train_____acc = accuracy_____score(y2_____train,clf.predict(X2_____train))

print ("Accuracy score on training data: {:.4f}".format(train_____acc))
print ("Accuracy score on testing data: {:.4f}".format(accuracy_____score(y2_____test, y2_____pred)))

plt.figure()
cm = confusion_____matrix(y2_____test, y2_____pred)
plot_____confusion_____matrix(cm, 'Confusion matrix for DT', popBandsList)
```

```
Accuracy score on training data: 1.0000
Accuracy score on testing data: 0.7200
```



Gradient boosting classificaiton has the best classification performance. Support vector classificaion in this case has made some wrong predictions, which might be caused by over-fitting of data. Based on the confusion matrix above, SVC often predicted the band "Fierbinteanu" as "Phemale", and its predictions for "Fierbinteanu" is pretty bad. Decision tree classification method also has made some inaccurate predictions from time to time, but its predictions for each bands is accurate overall. Whereas gradient boosting classification has made almost no wrong predictions, which increases the credibility of this method of classification. Although accuracy score for test 1 classification was higher than test 2, overall predictions of bands in test 2 seem more accurate.

(test 3) Genre Classification:

One could also use the above algorithms to simplify broadly classify songs as jazz, rock, classical etc. In this case, the training sets should be various bands within each genre. For instance, classic rock bands could be classified using sounds clips from Zep, AC/DC, Floyd, etc. while classical could be classified using Mozart, Beethoven, Bach, etc. Perhaps you can limit your results to three genres, for instance, rock, jazz, classical.

In [425]:
```
popLists = phemaleLists + GarmischLists + FierbinteanuLists
```

Get all lists of track ids for the other Folk bands

In [406]:
```
print(small[small['track','genre_____top']=='Folk']['artist','name'].value_____counts().head(6))
BigBloodLists = small[small['artist','name']=='Big Blood']
BigBloodLists = BigBloodLists.index.tolist()

KellyLatimoreLists = small[small['artist','name']=='Kelly Latimore']
KellyLatimoreLists = KellyLatimoreLists.index.tolist()

MichaelChapmanLists = small[small['artist','name']=='Michael Chapman']
MichaelChapmanLists = MichaelChapmanLists.index.tolist()

folkLists = BigBloodLists + KellyLatimoreLists + MichaelChapmanLists
```

```
Big Blood        91
Derek Clegg      45
Josh Woodward    24
Plusplus         19
Kelly Latimore   19
Michael Chapman  18
Name: (artist, name), dtype: int64
```

Get all lists of track ids for the other International bands

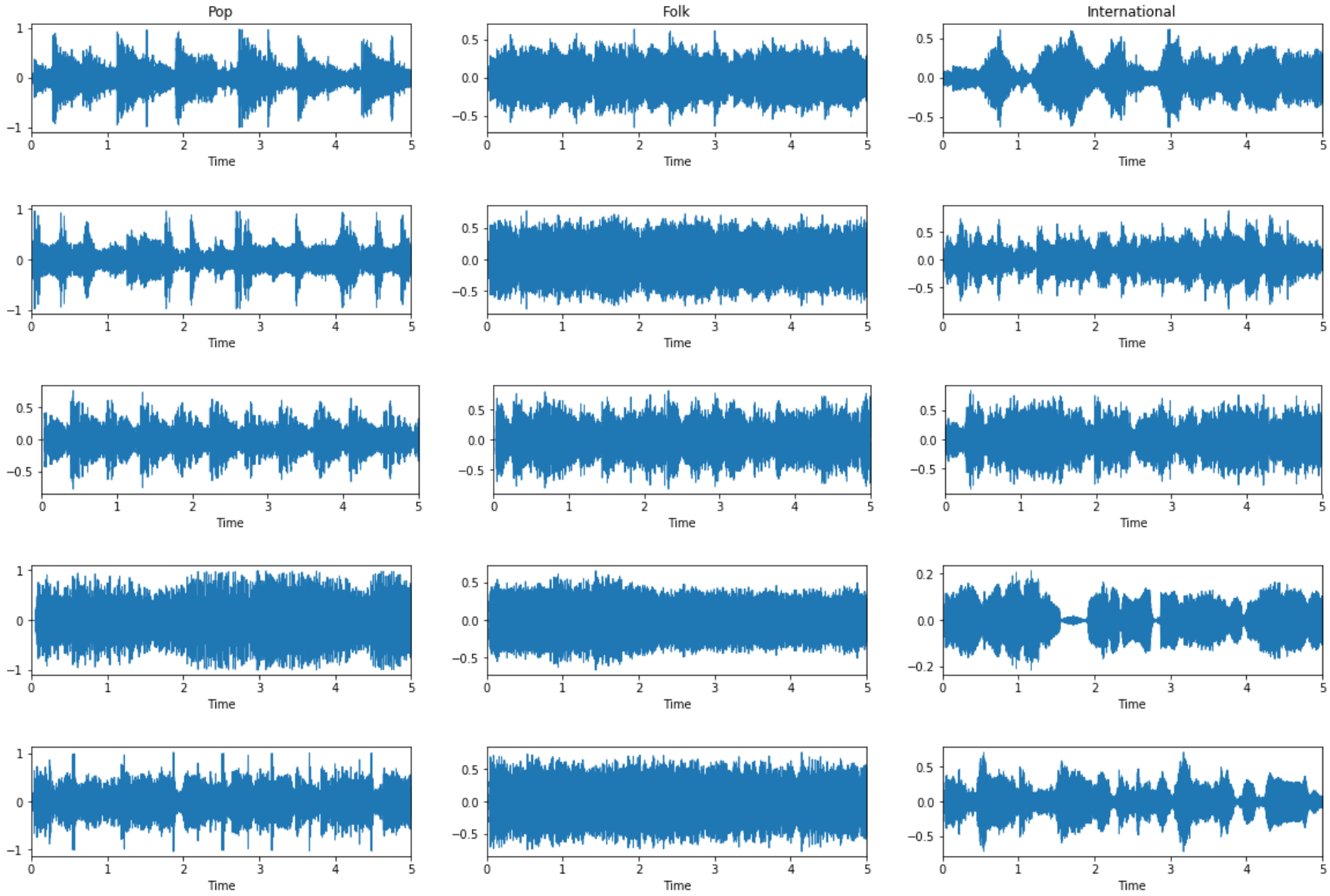
In [417]:
```
print(small[small['track','genre_____top']=='International']['artist','name'].value_____counts().head(5))
SláinteLists = small[small['artist','name']=='Sláinte']
SláinteLists = SláinteLists.index.tolist()

BlackSeaHotelLists = small[small['artist','name']=='Black Sea Hotel']
BlackSeaHotelLists = BlackSeaHotelLists.index.tolist()

TheSoundsOfTaraabLists = small[small['artist','name']=='The Sounds of Taraab']
TheSoundsOfTaraabLists = TheSoundsOfTaraabLists.index.tolist()
checkIfFilesExist(TheSoundsOfTaraabLists)

intlLists = BlackSeaHotelLists + SláinteLists + TheSoundsOfTaraabLists
```

```
Turku, Nomads of the Silk Road    29
Sláinte                           27
Black Sea Hotel                   23
The Sounds of Taraab              21
Gogofski                          19
Name: (artist, name), dtype: int64
```

Visualization: Waveplots and Spectrograms

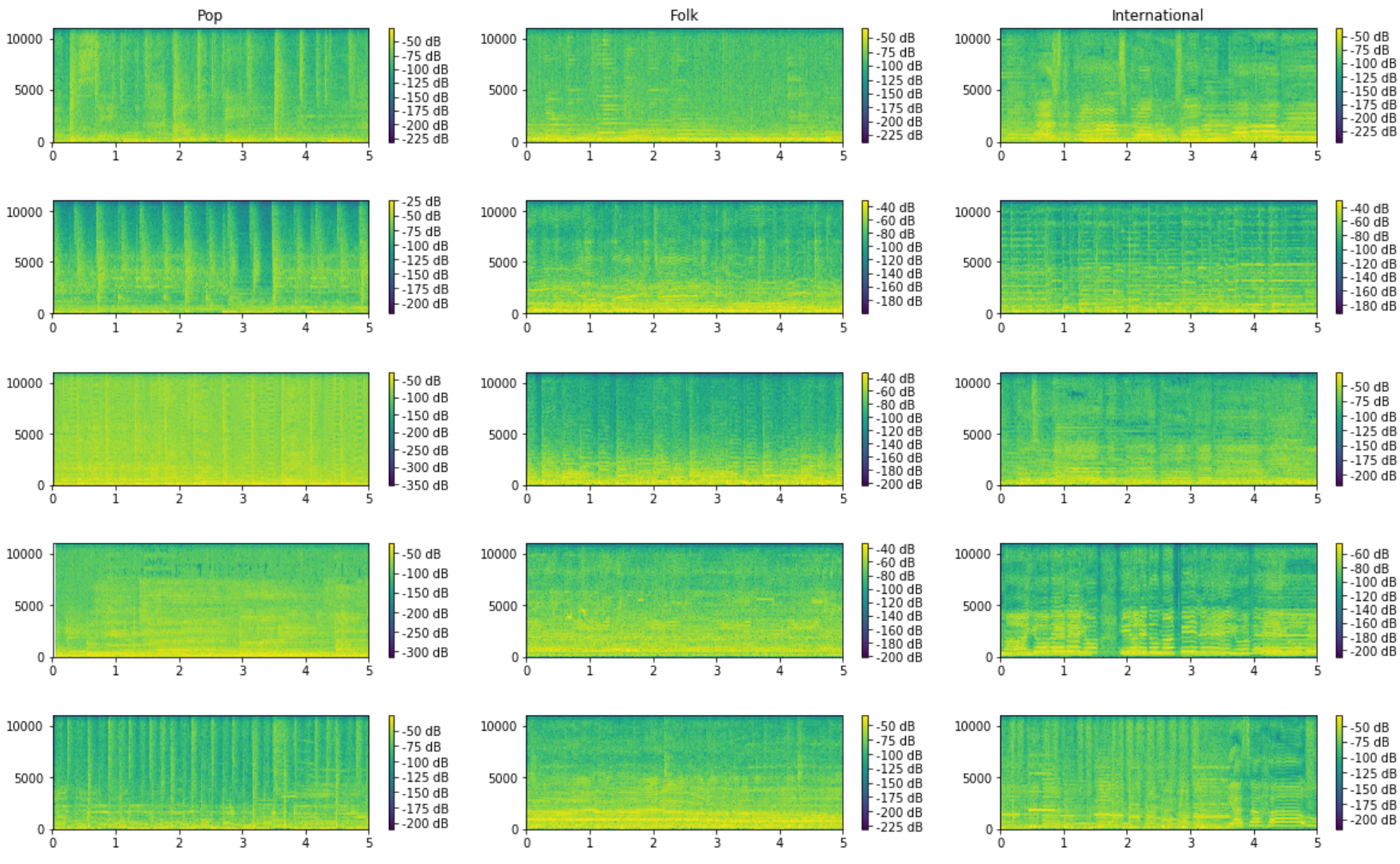
In [428]:
```
# Shuffle lists for more random representations
from random import shuffle
shuffle(popLists)
shuffle(folkLists)
shuffle(intlLists)
```

In [514]:
```
# Wave Plots
for i in range(5):
   plt.figure(figsize=(20,10))
   plt.subplot(5,3,i*3+1), waveplot(getX(popLists[i])), plt.title('Pop') if i == 0 else 0
   plt.subplot(5,3,i*3+2), waveplot(getX(folkLists[i])), plt.title('Folk') if i == 0 else 0
   plt.subplot(5,3,i*3+3), waveplot(getX(intlLists[i])), plt.title('International') if i == 0 else 0
```

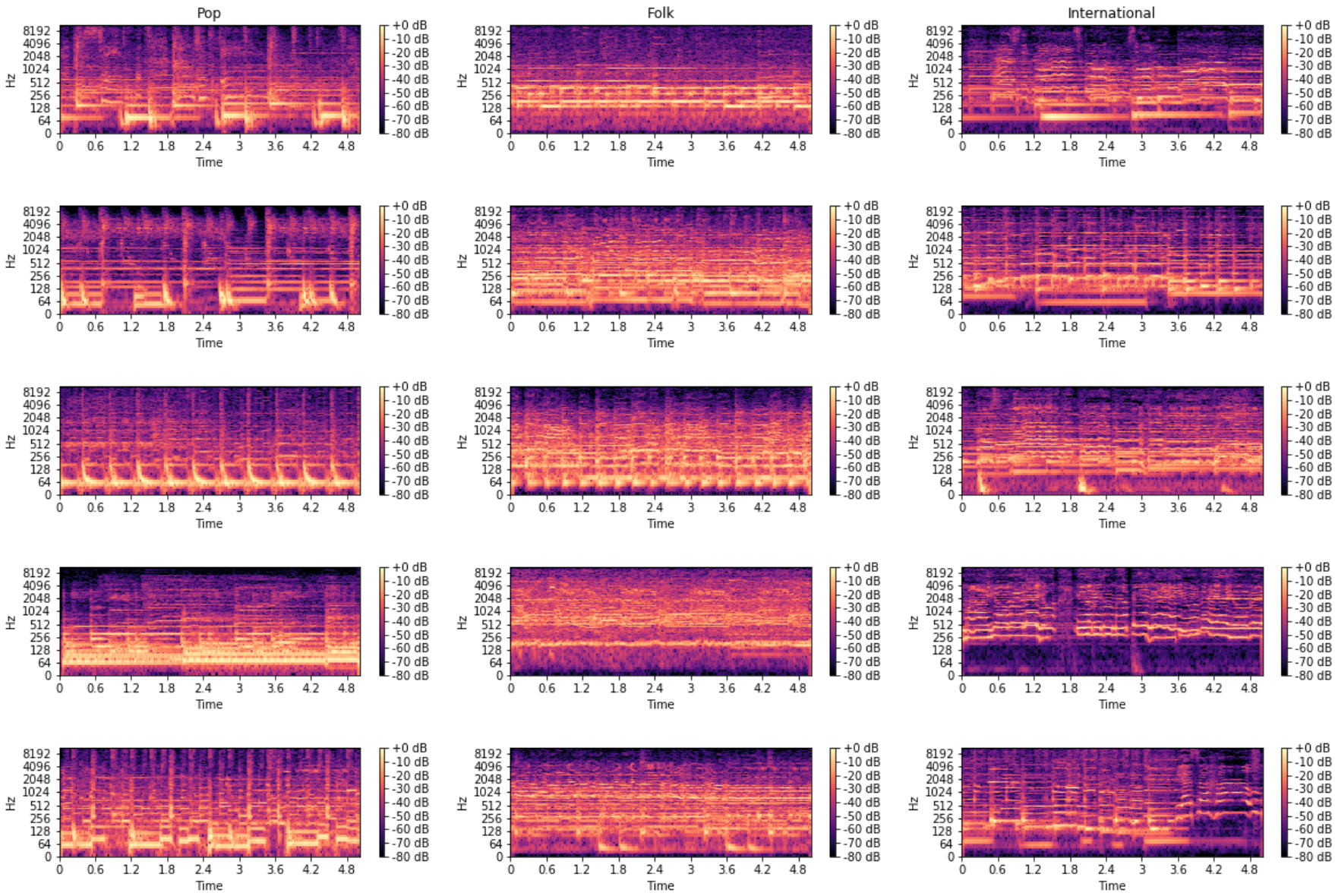


In [511]:
```
# Spectrograms
for i in range(5):
   plt.figure(figsize=(20,10))
   plt.subplot(5,3,i*3+1), plot_____specgram(getX(popLists[i])), plt.title('Pop') if i == 0 else 0
   plt.subplot(5,3,i*3+2), plot_____specgram(getX(folkLists[i])), plt.title('Folk') if i == 0 else 0
   plt.subplot(5,3,i*3+3), plot_____specgram(getX(intlLists[i])), plt.title('International') if i == 0 else 0
plt.show()
```

/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-packages/matplotlib/axes/_____axes.py:7221: RuntimeWarning: divide by zero encountered in log10
 Z = 10. * np.log10(spec)



In [510]:
```
# Log-Power Spectrogram
for i in range(5):
    plt.figure(figsize=(20,10))
    plt.subplot(5,3,i*3+1), plot_____log_____power_____specgram(getX(popLists[i])), plt.title('Pop') if i == 0 else 0
    plt.subplot(5,3,i*3+2), plot_____log_____power_____specgram(getX(folkLists[i])), plt.title('Folk') if i == 0 else 0
    plt.subplot(5,3,i*3+3), plot_____log_____power_____specgram(getX(intlLists[i])), plt.title('International') if i == 0
else 0
plt.show()
```



Plots observed for each genres (Pop, Folk, International) for some of the music clips are differentiable and are quite differentiable. However visualizations of wave plots and spectrograms alone is not easy to classify genres.

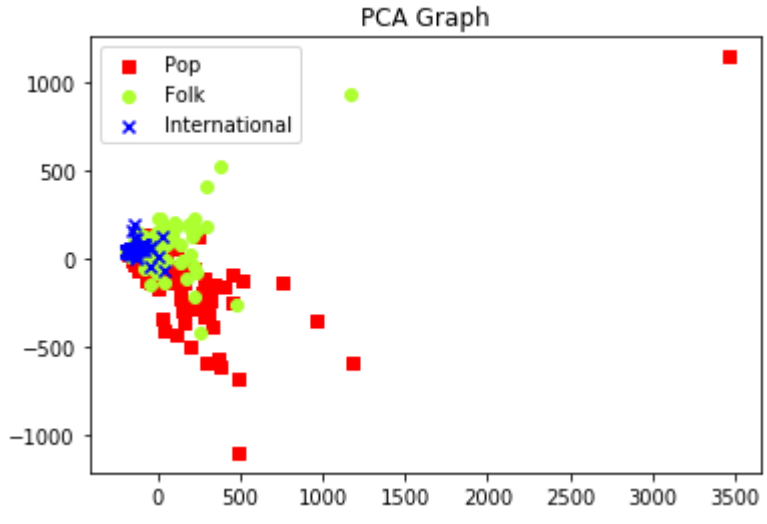
Extract Features: calculate feature matrix

In [441]:
```
features3, labels3 = [], []

parseList(features3, labels3, popLists, 0)
parseList(features3, labels3, folkLists, 1)
parseList(features3, labels3, intlLists, 2)

features3 = np.array(features3)
labels3 = np.array(labels3)
```

PCA

In [490]:
```
pca = PCA(n_____components=2).fit(features3)
pca_____2d = pca.transform(features3)
colormarkers = [ ['red','s'], ['greenyellow','o'], ['blue','x']]
for i in range(len(colormarkers)):
    px = pca_____2d[:, 0][labels3 == i]
    py = pca_____2d[:, 1][labels3 == i]
    plt.scatter(px, py, c=colormarkers[i][0], marker=colormarkers[i][1])
plt.title('PCA Graph')
plt.legend(['Pop','Folk','International'])
```

Out[490]: <matplotlib.legend.Legend at 0x12cd94cc0>



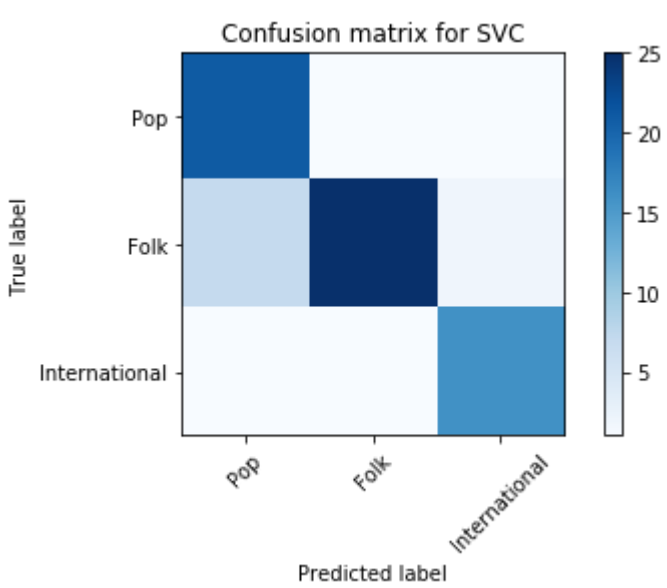
Classification

In [488]:
```
# Cross validation
# Split into a training and testing set
X3_____train, X3_____test, y3_____train, y3_____test = train_____test_____split(features3, labels3, test_____size=0.
25, random_____state=1)
print ("Training set has {} samples.".format(X3_____train.shape[0]))
print ("Testing set has {} samples.".format(X3_____test.shape[0]))
```

Training set has 222 samples.
Testing set has 75 samples.

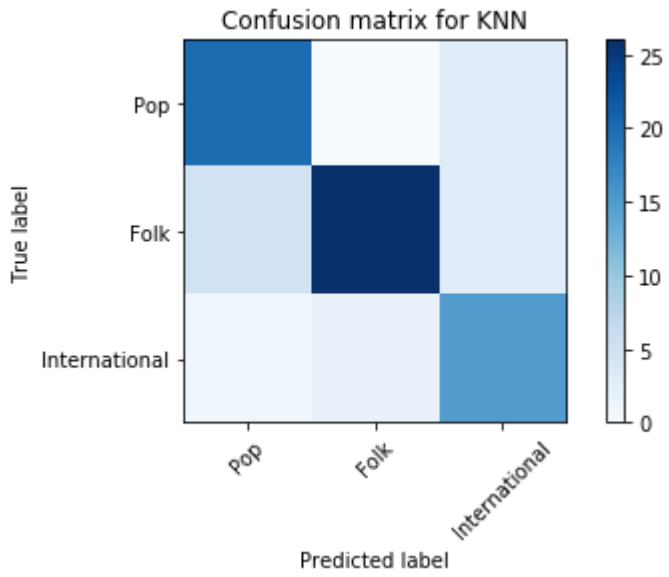
In [479]:
```
clf = svm.SVC(kernel='poly',degree=2).fit(X3_____train,y3_____train)

y3_____pred = clf.predict(X3_____test)
train_____acc = accuracy_____score(y3_____train,clf.predict(X3_____train))

print ("Accuracy score on training data: {:.4f}".format(train_____acc))
print ("Accuracy score on testing data: {:.4f}".format(accuracy_____score(y3_____test, y3_____pred)))

plt.figure()
genresList = ['Pop','Folk','International']
cm = confusion_____matrix(y3_____test, y3_____pred)
plot_____confusion_____matrix(cm, 'Confusion matrix for SVC', genresList)
```

Accuracy score on training data: 1.0000
Accuracy score on testing data: 0.8267



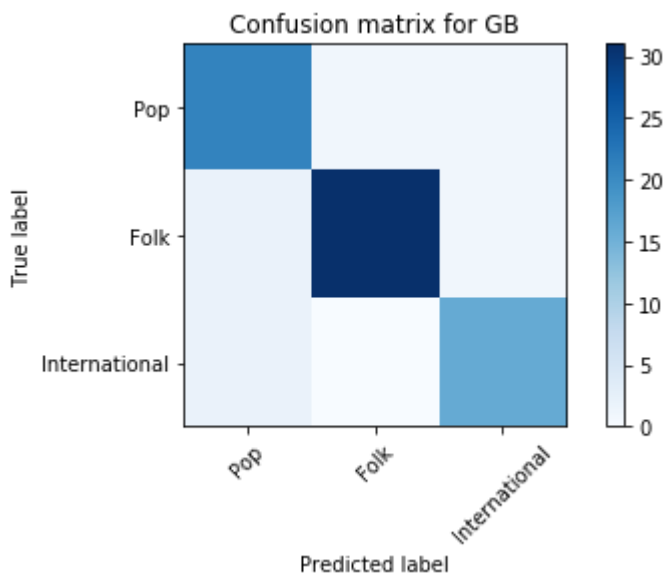
In [494]:
```
clf = KNeighborsClassifier(n_____neighbors=5).fit(X3_____train,y3_____train)

y3_____pred = clf.predict(X3_____test)
train_____acc = accuracy_____score(y3_____train,clf.predict(X3_____train))

print ("Accuracy score on training data: {:.4f}".format(train_____acc))
print ("Accuracy score on testing data: {:.4f}".format(accuracy_____score(y3_____test, y3_____pred)))

plt.figure()
cm = confusion_____matrix(y3_____test, y3_____pred)
plot_____confusion_____matrix(cm, 'Confusion matrix for KNN', genresList)
```

Accuracy score on training data: 0.7613
Accuracy score on testing data: 0.8133



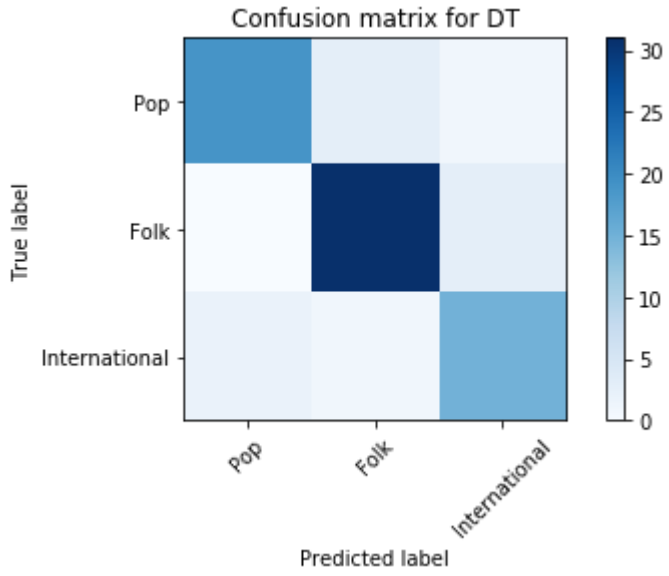
In [481]:
```
clf = GradientBoostingClassifier(learning_____rate=0.5, max_____depth=3, random_____state=0).fit(X3_____train
,y3_____train)

y3_____pred = clf.predict(X3_____test)
train_____acc = accuracy_____score(y3_____train,clf.predict(X3_____train))

print ("Accuracy score on training data: {:.4f}".format(train_____acc))
print ("Accuracy score on testing data: {:.4f}".format(accuracy_____score(y3_____test, y3_____pred)))

plt.figure()
cm = confusion_____matrix(y3_____test, y3_____pred)
plot_____confusion_____matrix(cm, 'Confusion matrix for GB', genresList)
```

Accuracy score on training data: 1.0000
Accuracy score on testing data: 0.9067



In [487]:
```
clf = DecisionTreeClassifier(max_____depth=2).fit(X3_____train,y3_____train)

y3_____pred = clf.predict(X3_____test)
train_____acc = accuracy_____score(y3_____train,clf.predict(X3_____train))

print ("Accuracy score on training data: {:.4f}".format(train_____acc))
print ("Accuracy score on testing data: {:.4f}".format(accuracy_____score(y3_____test, y3_____pred)))

plt.figure()
cm = confusion_____matrix(y3_____test, y3_____pred)
plot_____confusion_____matrix(cm, 'Confusion matrix for DT", genresList)
```

Accuracy score on training data: 0.8423
Accuracy score on testing data: 0.8667



Accuracy score of 0.91 for the gradient boosting classification was the highest. Overall, there are very little mis-predictions from all classification algorithms, and the accuracy scores are generally high for all classification algorithms. Test 3 has the most datasets used for genre classification training and testing. Larger datasets resulted in more precise and accurate classification model.

References

[1] KrisYu. KrisYu/MusicGenreClassification. GitHub, 11 May 2016, `github.com/KrisYu/Music_Genre_Classification`.

[2] Saeed, Aaqib. Urban Sound Classification, Part 1. Aaqib Saeed, `aqibsaeed.github.io/2016-09-03-urban-sound-classification-part-1/`.

[3] Michal Defferrard. FMA: A Dataset For Music Analysis. GitHub, 17 Jan. 2018, `github.com/mdeff/fma`.

[4] Kosina, K. 2002. Music genre recognition. Diploma thesis. Technical College of Hagenberg, Austria.

[5] LibROSA. LibROSA - librosa 0.6.0 documentation, `librosa.github.io/librosa/index.html`.