In [1]:

```
import numpy as np
import pandas as pd
import scipy.io.wavfile as wav
from python_speech_features import mfcc
from tempfile import TemporaryFile
import os
import math
import pickle
import random
import operator
```

In [2]:

```
def distance(instance1, instance2, k):
    distance = 0
    mm1 = instance1[0]
    cm1 = instance1[1]
    mm2 = instance2[0]
    cm2 = instance2[1]
    distance = np.trace(np.dot(np.linalg.inv(cm2), cm1))
    distance += (np.dot(np.dot((mm2-mm1).transpose(), np.linalg.inv(cm2)), mm2-mm1))
    distance += np.log(np.linalg.det(cm2)) - np.log(np.linalg.det(cm1))
    distance -= k
    return distance
```

In [3]:

```
#define a function to get distance between feature vectors and find neighbors
def getNeighbors(trainingset, instance, k):
    distances = []
    for x in range(len(trainingset)):
        dist = distance(trainingset[x], instance, k) + distance(instance,trainingset[x],k)
        distances.append((trainingset[x][2], dist))
    distances.sort(key=operator.itemgetter(1))
    neighbors = []
    for x in range(k):
        neighbors.append(distances[x][0])
    return neighbors
```

In [4]:

```
#function to identify the nearest neighbors
def nearestclass(neighbors):
    classVote = {}

    for x in range(len(neighbors)):
        response = neighbors[x]
        if response in classVote:
            classVote[response] += 1
        else:
            classVote[response] = 1

    sorter = sorted(classVote.items(), key=operator.itemgetter(1), reverse=True)
    return sorter[0][0]
```

In [5]:

```
# define a function that will evaluate a model
def getAccuracy(testSet, prediction):
    correct = 0
    for x in range(len(testSet)):
        if testSet[x][-1] == prediction[x]:
            correct += 1
    return 1.0 * correct / len(testSet)
```

In [10]:

```
import librosa, IPython
import librosa.display
file = 'C:/Users/srira/Desktop/music_genre_classification/Data/genres_original/disco/disco.00041.wav'
signal, sr = librosa.load(file , sr = 22050)
IPython.display.Audio(signal, rate=sr)
```

Out[10]:

▶ 0:00 / 0:30

In [7]:

```
file = 'C:/Users/srira/Desktop/danger_zone.mp3'
signal, sr = librosa.load(file , sr = 22050)
IPython.display.Audio(signal, rate=sr)
```

Out[7]:

▶ 0:00 / 0:16

In [71]:

```
directory = 'C:/Users/srira/Desktop/music_genre_classification/Data/genres_original'
f = open("my.dat", "wb")
i = 0
for folder in os.listdir(directory):
    print(folder)
    i += 1
    if i == 10:
        break
    for file in os.listdir(directory+"/"+folder):
        #print(file)
        try:
            (rate, sig) = wav.read(directory+"/"+folder+"/"+file)
            mfcc_feat = mfcc(sig, rate, winlen = 0.020, appendEnergy=False)
            covariance = np.cov(np.matrix.transpose(mfcc_feat))
            mean_matrix = mfcc_feat.mean(0)
            feature = (mean_matrix, covariance, i)
            pickle.dump(feature, f)
        except Exception as e:
            print("Got an exception: ", e, 'in folder: ', folder, ' filename: ', file)
f.close()
```

```
blues
classical
country
disco
hiphop
metal
pop
reggae
rock
```

In [62]:

```
#split dataset into train and test set
dataset = []

def loadDataset(filename, split, trset, teset):
    with open('my.dat','rb') as f:
        while True:
            try:
                dataset.append(pickle.load(f))
            except EOFError:
                f.close()
                break
    for x in range(len(dataset)):
        if random.random() < split:
            trset.append(dataset[x])
        else:
            teset.append(dataset[x])

trainingSet = []
testSet = []
loadDataset('my.dat', 0.66, trainingSet, testSet)
```

In [63]:

```
# Make the prediction using KNN(K nearest Neighbors)
length = len(testSet)
predictions = []
for x in range(length):
    predictions.append(nearestclass(getNeighbors(trainingSet, testSet[x], 5)))

accuracy1 = getAccuracy(testSet, predictions)
print(accuracy1)
```

```
0.7222222222222222
```

In [64]:

```
from collections import defaultdict
results = defaultdict(int)

directory = 'C:/Users/srira/Desktop/music_genre_classification/Data/genres_original'

i = 1
for folder in os.listdir(directory):
    results[i] = folder
    i += 1
```

In [68]:

```
pred = nearestclass(getNeighbors(dataset, feature,456))
print(results[pred])
```

```
blues
```