Department of Electrical and Computer Engineering

EEE 490 - Introduction to Big Data

Music Genre Classification

Authors: *Student number:*
Adlet Anarbek ID 201473870
Didar Amangeldiyev ID 201462437

November 23, 2020

Contents

Abstract

In this study, we created a classification model that categorizes music files according to their genre utilizing **KNN** algorithm. Approaches such as different number of k near neighbors, diverse compositions of features, and decreasing number of genres to enhance the classification model have been taken into consideration during the project. The mixture of features that maximizes the performance of the system has been identified. All experiment has been conducted on **GTZAN** dataset and we report an *accuracy* value of **69.33%** for *5-genre* music classification model.

1 Introduction

The problem of automatic genre classification became popular in the middle of 2000s, when most of the people had an access to stable and high data-rate internet connections. Since different people have different musical preferences, it is important to divide the existing playlist into groups according to the genre of the song. Nowadays, there are different machine learning algorithms that implement the classification of music genre, and most of the social networks and musical applications use that algorithms. It helps to create a high-quality playlist and to make a good suggestion according to the person's favorite genres

The aim of the project is to classify music genres using a dataset consisting of short musical fragments of different genres, and the amount of fragments will be evenly distributed according to the genre. Actually, there are several ML methods that can sort musical data by their genre, however, after a thorough literature review **K-nearest neighbor (KNN)** algorithm has been chosen, since it has one of the highest accuracies among ML techniques [1]-[3]. In the current project, we have used GTZAN dataset that can be accessed from https://www.kaggle.com/andradaolteanu/gtzan-dataset-music-genre-classification. Music genre classifier were implemented using MatLab.

The rest of this paper is organized as follows. Section 2 describes the algorithm that was applied and the overview of the dataset. Results and their meaning are reported in Section 3, followed by the conclusions in Section 4.

2 Methodology

2.1 Dataset

High-quality dataset is one of the most important details of high-accuracy ML algorithm. In order to create a music genre classifier, a broad range of musical data is needed and this data can be interpreted in two ways. Firstly, audio tracks can be divided into smaller fragments and converted into spectrograms, and this type of data can be used to train Convolutional Neural Networks (CNN)[4]. Second method is to take small fragments of an audio signal and to calculate their **Mell frequency Cepstral coefficients (MFCC)**in different frequency bands, **Root Mean Square (RMS), Zero Crossing Rate (ZCR), Tempo, Chroma Features, Spectral Centroid and Spectral Roll-off.** This is an approach that was used in our algorithm. Our dataset already consists all of these values in the document called "features30sec.csv" and there are overall of 58 features. This document has 1000 rows that represent the number of songs and there are overall 10 genres of music, each consisting 100 song fragments. Genres are namely, "Blues", "Classical", "Country", "Disco", "Hip-hop", "Jazz", "Metal", "Pop", "Reggae" and "Rock".

2.2 Algorithm

Since our dataset already has all of the necessary values to construct working ML model, no additional mathematical calculations were made. However, if needed, the equations that was described in [4] and [5] can be applied to calculate all of the above-mentioned quantities. For example, in order to check the genre of a particular song that is not included to the given list of 1000 songs, all of the features mentioned in the "features30sec.csv" have to be calculated.

Implementation of the automatic music genre classifier that were proposed in this project is based on the KNN technique. Firstly *.csv* file was taken as an input file to *MatLab* and converted to the *1000x60* matrix (1000 songs and 58 features, the value is 60, since the first and last column is name of the file and genre respectively) and taken also were taken as a cell type. Then, the dataset was split into *training* and *test-*

ing data as 70/30, respectively (70 songs from each genre for training and 30 for testing). **KNN** algorithm were applied both manually and using existing *MatLab* commands and no differences were observed (both codes are represented in appendix). In the manual solution, *Euclidean* distances between neighbor points were calculated and used to find the nearest neighbors. Finally, *accuracy* of the model and *Area Under Curve (AUC)* were calculated. Accuracy was found by dividing the number of correctly classified points by the overall number of testing data and *AUC* was found using special *MatLab* functions.

Designed model was tested for different number of features and nearest neighbors. Moreover, there are two main implementations, for 10 genres and for 5 genres. In *5-genre* classifier, "Blues", "Classical", "Country", "Hip-hop", "Metal" genres were used, due to the tangible difference in sound. Results and differences for *5-genre* and *10-genre* classifiers are represented in the next section of the paper.

3 Results and Discussion

3.1 Results

In this section of the project, the *accuracy* and the *AUC* of the classification model has been investigated with the involvement of different number of K near neighbors and the number of features. At first, the performance of the model with all (58) features has been considered. Then, a classification model with only mean value features has been observed. Lastly, the performances of the classification models were evaluated with features that have been selected to maximize the values of *accuracy* and *AUC* of the model.

Number of Features	Number of K near neighbors	Accuracy	AUC
58	3	22.33%	0.6412
58	5	22.33%	0.6322
58	7	22.00%	0.6315
58	9	21.00%	0.6229
58	11	22.67%	**0.6700**
58	13	**24.67**%	0.6496

Table 1. Accuracy and AUC of 10-genre music classification model with all features included.

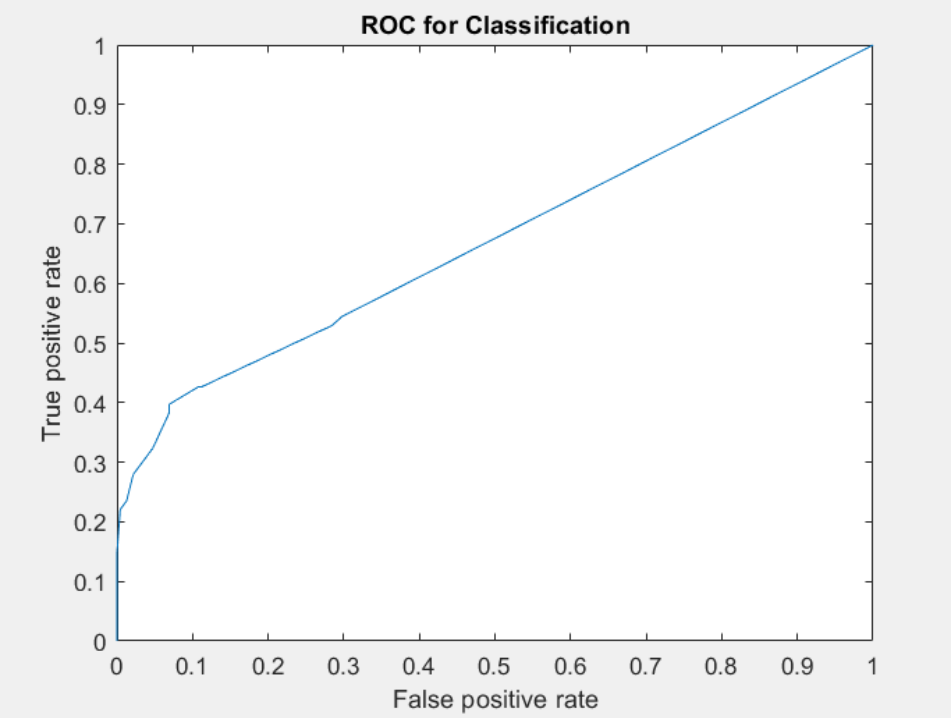

Figure 1: Graph of ROC for maximum AUC from Table 1.

Number of Features	Number of K near neighbors	Accuracy	AUC
58	4	45.33%	0.5851
58	6	40.67%	0.6218
58	8	42.00%	0.6250
58	10	47.33%	**0.6537**
58	12	46.00%	0.6244
58	14	**48.67%**	0.6194

Table 2. Accuracy and AUC of 5-genre music classification model with all features included.

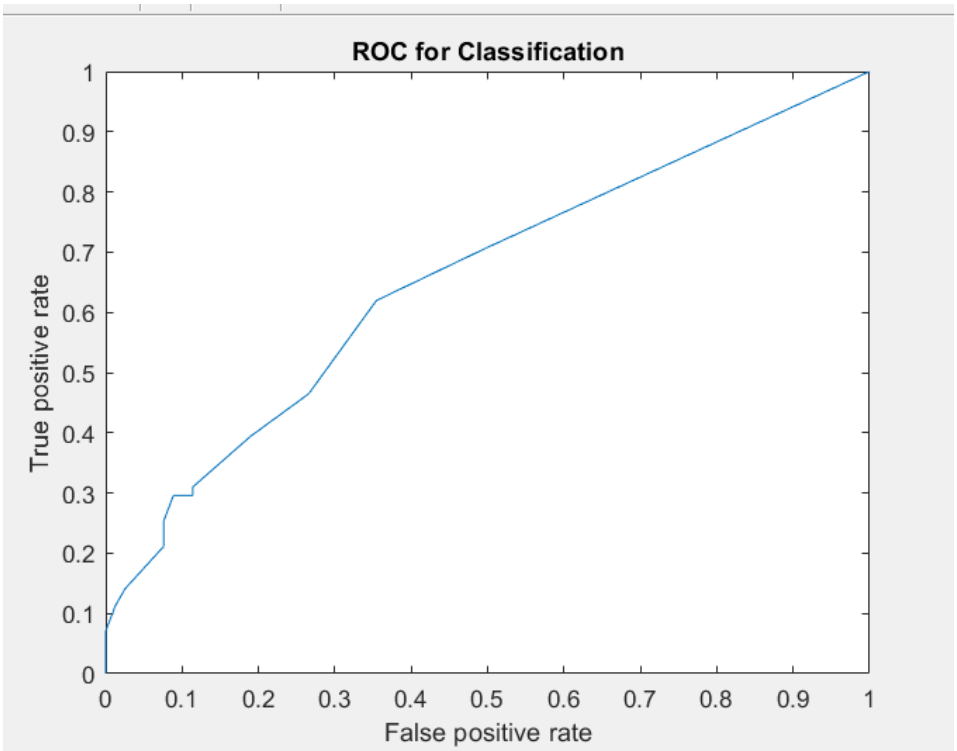

Figure 2: Graph of ROC for maximum AUC from Table 2.

Number of Features	Number of K near neighbors	Accuracy	AUC
28	3	22.67%	0.6033
28	5	**28.67**%	0.5974
28	7	27.33%	0.5890
28	9	28.00%	0.5877
28	11	25.00%	**0.6064**
28	13	26.33%	0.5665

Table 3. Accuracy and AUC of 10-genre music classification model with only mean value features.

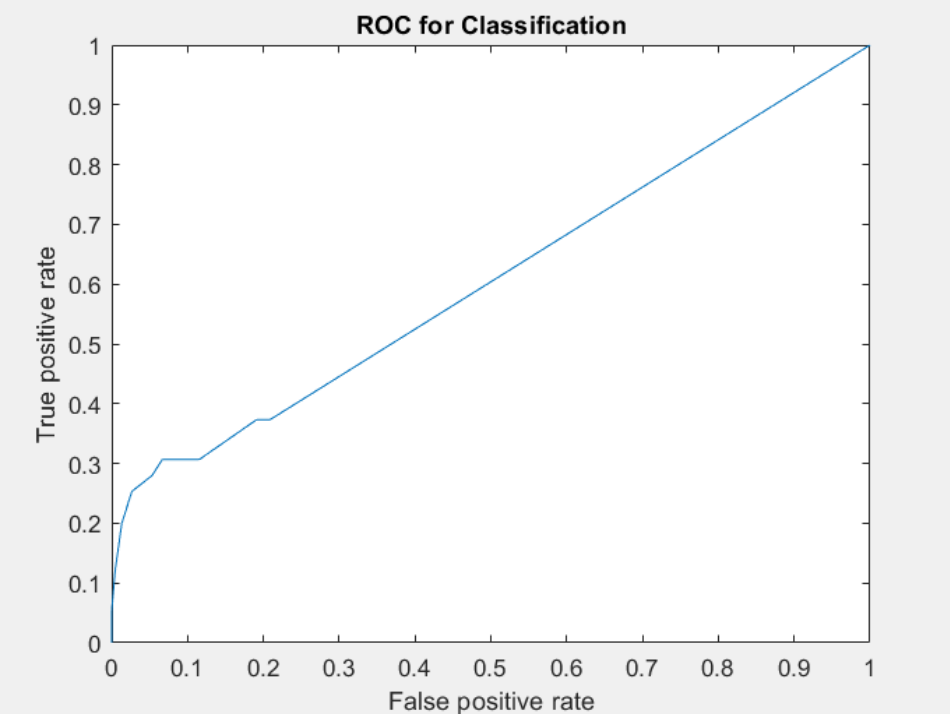

Figure 3: Graph of ROC for maximum AUC from Table 3.

Number of Features	Number of K near neighbors	Accuracy	AUC
28	4	**52.00**%	0.6210
28	6	45.33%	**0.6350**
28	8	47.33%	0.6177
28	10	49.33%	0.6249
28	12	48.00%	0.6124
28	14	50.00%	0.6161

Table 4. Accuracy and AUC of 5-genre music classification model with only mean value features.

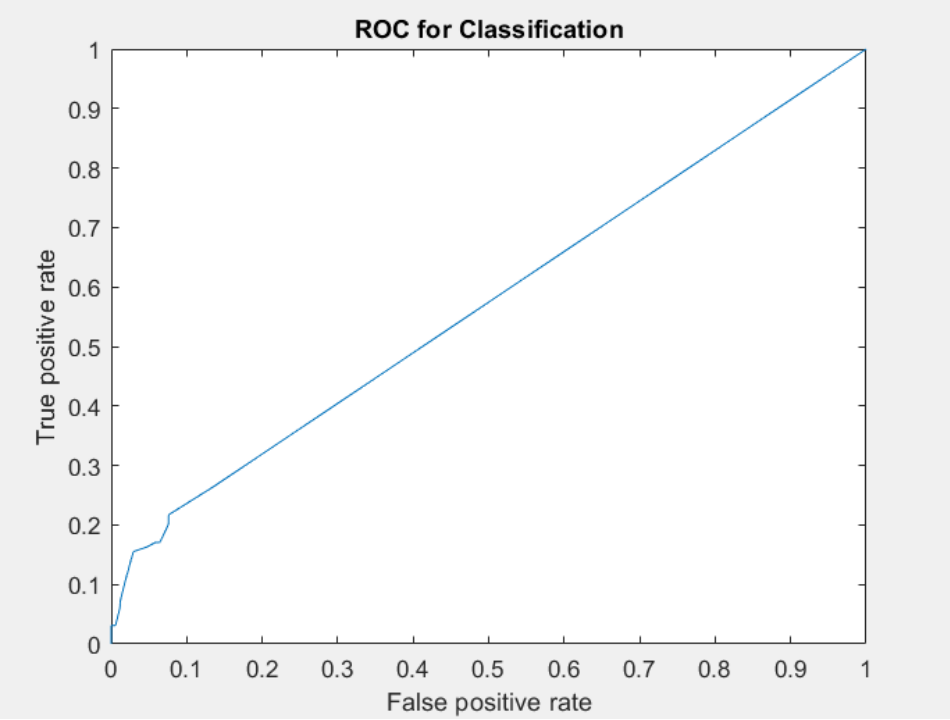

Figure 4: Graph of ROC for maximum AUC from Table 4.

Number of Features	Number of K near neighbors	Accuracy	AUC
7	3	38.67%	0.5694
7	5	39.67%	0.5602
7	7	41.67%	0.5541
7	9	**43.00%**	**0.5712**
7	11	39.67%	0.5631
7	13	40.33%	0.5610

Table 5. Accuracy and AUC of 10-genre music classification model with only selected features.

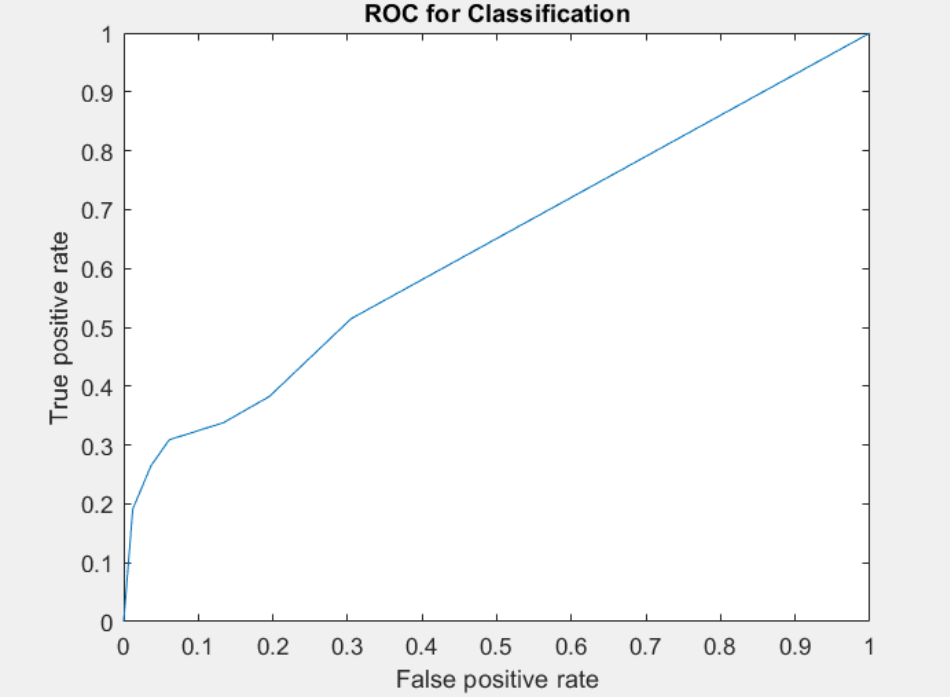

Figure 5: Graph of ROC for maximum AUC from Table 5.

Number of Features	Number of K near neighbors	Accuracy	AUC
7	4	66.67%	0.5912
7	6	68.00%	0.5582
7	8	**69.33%**	0.5870
7	10	67.33%	**0.5966**
7	12	68.67%	0.5693
7	14	65.33%	0.5778

Table 6. Accuracy and AUC of 5-genre music classification model with only selected features.

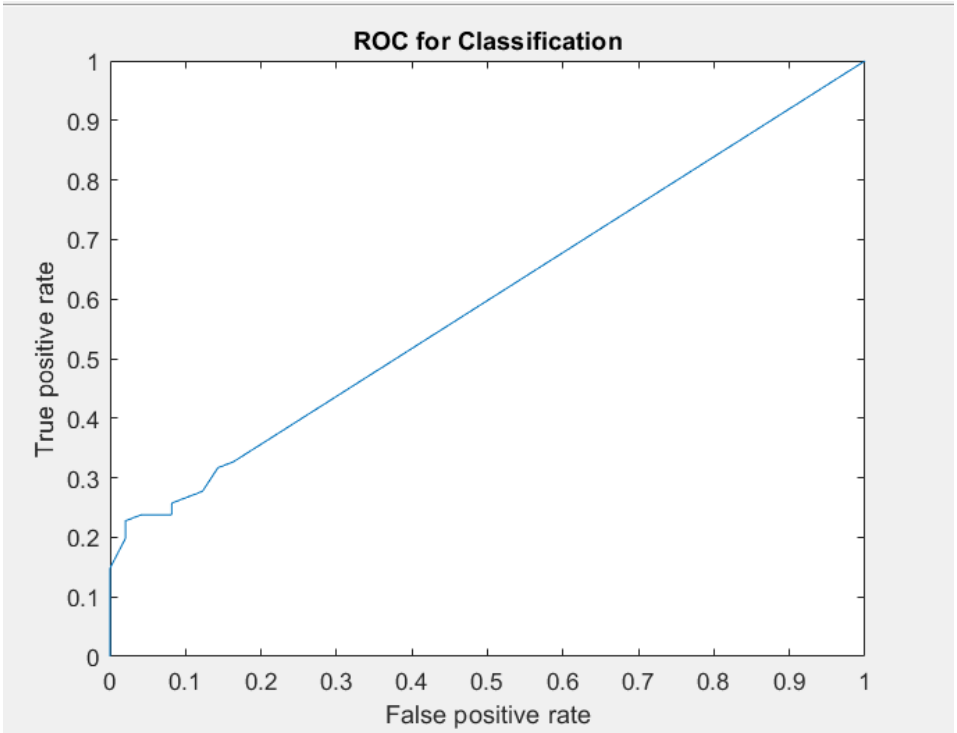

Figure 6: Graph of ROC for maximum AUC from Table 6.

3.2 Discussion

In this part of the project, different modelling approaches will be discussed based on results of the previous section.

During the project, we referred to accuracy as a percentage of correctly classified test samples. As it can be seen from *Tables 1 – 4*, the *accuracy* of these models is not that precise. The reason for this could be using not important features in modelling, because in both cases a lot of features, 58 and 28 features respectively, are being involved in the creation of the classification model. Another cause for the inaccurate results could be making use of *Euclidean norm* for calculating the distances between points in the dataset. Since, the model may require a custom formula for evaluating the distances, which in the case will enhance performance of the model.

On the other hand, the values of accuracy in *Tables 5 – 6* are more precise compared to the results of previous models. In this case, we selected 7 features (*length, mean* and *variance of chroma short-time Fourier transform (STFT), variance of Root Mean Square Energy (RMSE), mean* and

variance of Zero Crossing Rate (ZCR), variance of harmony) to create the classification **KNN** model to maximize the accuracy value. Results display **69.33%** and **43.00%** accuracy for *5-genre* and *10-genre* music classification models accordingly.

Moreover, it is important to mention that, in all modelling cases, the accuracy of *5-genre* models is much higher than *10-genre* music classification model. Therefore, we can assume that decreasing the number of genres substantially improves the accuracy of the appropriate model.

Meanwhile, AUC is defined as the area under the *Receiver Operator Characteristics (ROC)* curve. This estimation criteria are a simple method to evaluate the performance of a multiclass classification system. In the previous section of the project, the peak AUC value for each modelling case has been considered. According to the results, every AUC value of the system is higher than **0.5**, in other words, AUC of the baseline model with equal probability for each class label, which satisfies the requirement for a correctly designed system model.

4 Conclusion

In this work, the task of music genre classification is studied using the **GTZAN** dataset. We propose the method of using **KNN** algorithm by applying different features of an audio file. It was observed that the maximum *accuracy* of the developed algorithm is **69.33%**, however this value can be improved by applying less number of genres. It is to be noted that the dataset that were used in the project were created in the beginning of 2000th, so the *accuracy* with modern music can be a bit lower since the music style changed a lot from that time. On the other hand, a new dataset containing more genres and a large variety of songs can be developed and used in the proposed algorithm. In future studies, *neural network* techniques can be applied in addition to **KNN** to increase the accuracy of the music genre classifier.

Bibliography

[1] M. Wu and X. Liu, "A Double Weighted KNN Algorithm and Its Application in the Music Genre Classification," 2019 6th International Con-

ference on Dependable Systems and Their Applications (DSA), Harbin, China, 2020, pp. 335-340
[2] R. J. M. Quinto, R. O. Atienza and N. M. C. Tiglao, "Jazz music subgenre classification using deep learning," TENCON 2017 - 2017 IEEE Region 10 Conference, Penang, 2017, pp. 3111-3116
[3] B. Liang and M. Gu, "Music Genre Classification Using Transfer Learning," 2020 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR), Shenzhen, Guangdong, China, 2020, pp. 392-393
[4] Bahuleyan, H., "Music Genre Classification using Machine Learning Techniques", arXiv:1804.01149v1, 2018
[5] Aisha Gemala Jondya, Bambang Heru Iswanto, "Indonesian's Traditional Music Clustering Based on Audio Features", Procedia Computer Science, Volume 116, 2017, Pages 174-181, ISSN 1877-0509

Appendix

In this section, MatLab code for **KNN** training algorithm is represented.

Appendix A

Here is the MatLab code for 5 genre classification

```
%% Music Classification with KNN (5 Genres)
clc
clear
%% Reading csv file
A = readmatrix('features_30_sec.csv');
kd = readcell('features_30_sec.csv', 'delimiter', ...
    ',','HeaderLines', 1);

%% Splitting dataset into training and testing
blues = A(1:100,:);
classical = A(101:200,:);
country = A(201:300,:);
disco = A(301:400,:);
hiphop = A(401:500,:);
jazz = A(501:600,:);
metal = A(601:700,:);
pop = A(701:800,:);
reggae =A(801:900,:);
rock = A(901:1000,:);

```

```
blues1 = kd(1:100,:);
classical1 = kd(101:200,:);
country1 = kd(201:300,:);
disco1 = kd(301:400,:);
hiphop1 = kd(401:500,:);
jazz1 = kd(501:600,:);
metal1 = kd(601:700,:);
pop1 = kd(701:800,:);
reggae1 =kd(801:900,:);
rock1 = kd(901:1000,:);

B = [blues; classical; country; hiphop; metal];
C =[blues1; classical1; country1; hiphop1; metal1];
training = [];
testing = [];
features = [2,3,4,6,13,14,16];

Xtrain = [];
Ytrain = [];

Xtest = [];
Ytest = [];

for ii=1:100:500
   training = [training; B(ii:ii+69,features)];
   testing = [testing; B(ii+70:ii+99,features)];

   Xtrain = [Xtrain; C(ii:ii+69,features)];
   Ytrain = [Ytrain; C(ii:ii+69,60)];

   Xtest = [Xtest; C(ii+70:ii+99,features)];
   Ytest = [Ytest; C(ii+70:ii+99,60)];
end

%% Code for counting neighbors and calculating euclidian distance ...
    manually
% dist = [];
% k=8;
% answer = [];
% for ii = 1:150
%     dist = [];
%     for step = 1:350
%         dist = [dist; norm(training(step,:)-testing(ii,:))];
%     end
%     sorted_dist = sort(dist);
%     neighbors_dist = sorted_dist(1:k);
%     neighbors = [];
%     for jj=1:k
%         neighbors = [neighbors find(dist==neighbors_dist(jj),1)];
%     end
%     neighbors = ceil(neighbors/70);
%     answer = [answer mode(neighbors)];
% end
%
% count = 0;
% for ii = 1:150
%    if answer(ii)==ceil(ii/30)
```

```
%        count = count + 1;
%     end
% end
%
% count/150

%% Creaing KNN classification model
resp = [];
Mdl = fitcknn(training,Ytrain,'NumNeighbors',10);

%% Testing the data
[label,score,cost] = predict(Mdl,testing);

%% Calculating accuracy of model
count = 0;
 for ii = 1:150
    if strcmp(label(ii),Ytest{ii,1})
        count=count+1;

    end
 end
resp = strcmp(label,Ytest);
count/150

%% Calculating and plotting AUC
[Xknn,Yknn,Tknn,AUCknn,OPTPROC] = perfcurve(resp,score(:,2),'true');
plot(Xknn,Yknn)
xlabel('False positive rate')
ylabel('True positive rate')
title('ROC for Classification')
AUCknn
```

Appendix B

Here is the MatLab code for 10 genre classification

```
%%  Music Classification with KNN (10 Genres)
clc
clear
%% Reading csv file
A = readmatrix('features_30_sec.csv');
kd = readcell('features_30_sec.csv', 'delimiter', ...
    ',','HeaderLines', 1);
%% Splitting dataset into training and testing
training = [];
testing = [];
features = [2,3,4,6,13,14,16];

Xtrain = [];
Ytrain = [];

```

```
Xtest = [];
Ytest = [];

for ii=1:100:1000

    training = [training; A(ii:ii+69,features)];
    testing = [testing; A(ii+70:ii+99,features)];

    Xtrain = [Xtrain; kd(ii:ii+69,features)];
    Ytrain = [Ytrain; kd(ii:ii+69,60)];

    Xtest= [Xtest; kd(ii+70:ii+99,features)];
    Ytest = [Ytest; kd(ii+70:ii+99,60)];
end

%% Code for counting neighbors and calculating euclidian distance ...
   manually
% dist = [];
% k=11;
% answer = [];
% for ii = 1:300
%     dist = [];
%     for step = 1:700
% %         s = 0;
% %         for ff = 1:length(features)
% %             s = s + (training(step,ff)-testing(ii,ff))^2;
% %         end
%
%         dist = [dist; norm(training(step,:)-testing(ii,:))];
%     end
%     sorted_dist = sort(dist);
%     neighbors_dist = sorted_dist(1:k);
%     neighbors = [];
%     for jj=1:k
%         neighbors = [neighbors find(dist==neighbors_dist(jj),1)];
%     end
%     neighbors = ceil(neighbors/70);
%     answer = [answer mode(neighbors)];
% end
%
% count = 0;
% for ii = 1:300
%    if answer(ii)==ceil(ii/30)
%       count = count + 1;
%    end
% end
% count/300

%% Creaing KNN classification model
resp = [];
Mdl = fitcknn(training,Ytrain,'NumNeighbors',11);

%% Testing the data
[label,score,cost] = predict(Mdl,testing);

%% Calculating accuracy of model
count = 0;
```

```
 for ii = 1:300
    if strcmp(label(ii),Ytest{ii,1})
        count=count+1;

    end
 end
resp =strcmp(label,Ytest);
count/300

%% Calculating and plotting AUC
[Xknn,Yknn,Tknn,AUCknn,OPTPROC] = perfcurve(resp,score(:,2),'true');
plot(Xknn,Yknn)
xlabel('False positive rate')
ylabel('True positive rate')
title('ROC for Classification')
AUCknn
```