CENG3522: Applied Machine Learning

MUSİC GENRE CLASSIFICATION

Project Final Report

07/06/2020

Team Members:

Kadir KARAYAKALI

Merve ÇAĞLARER

Sümeyra ÖZUĞUR

Our Presentation Link:
https://drive.google.com/file/d/1RL0JIBNuhG630LO2NcXQ4KM9oDRkHhjF/view?usp=sharing

1. Description of Data Set

GTZAN genre collection was used as our dataset. The dataset consists of 1000 audio tracks each 30 seconds long. It contains 5 genres(Hip Hop, Jazz, Metal, Pop, Rock), each represented by 100 tracks. Tracks are wav(Waveform Audio File) formats. This tracks will transform digital datas.
Source: http://marsyas.info/downloads/datasets.html

Our digital data columns and descriptions;
Zero Crossing Rate: The zero crossing rate is the rate of sign-changes along a signal, i.e., the rate at which the signal changes from positive to negative or back.
Spectral Centroid: It indicates where the ”centre of mass” for a sound is located and is calculated as the weighted mean of the frequencies present in the sound.
Spectral Rolloff: It is a measure of the shape of the signal. It represents the frequency below which a specified percentage of the total spectral energy, e.g. 85%, lies.
Spectral Bandwidth: It is the wavelength interval in which a radiated spectral quantity is not less than half its maximum value.
Mel-Frequency Cepstral Coefficients: The Mel frequency cepstral coefficients (MFCCs) of a signal are a small set of features (usually about 10–20) which concisely describe the overall shape of a spectral envelope.
Chroma Frequencies: Chroma features are an interesting and powerful representation for music audio in which the entire spectrum is projected onto 12 bins representing the 12 distinct semitones (or chroma) of the musical octave.One of the main features of Chroma features is that they are harmonic and melodic, while being robust against changes in timbre and instrumentation.

2. Data Transformation and Cleaning Steps

This tracks is transformed to digital data by using Python Librosa. Librosa is a Python module to analyze audio signals in general but geared more towards music.
-Audio files were read with librosa library and converted into objects
-Required properties were transferred to variables with librosa library. The properties are generally written to the csv file in averages, as they represent a graphic and have a different value every second.
We use code below;

```
file = open('train.csv', 'w', newline='')
with file:
    writer = csv.writer(file)
    writer.writerow(header)
genres = 'hiphop jazz metal pop rock'.split()
for g in genres:
    for filename in os.listdir(f'./genres/{g}'):
        songname = f'./genres/{g}/{filename}'
        y, sr = librosa.load(songname, mono=True, duration=30)
        chroma_stft = librosa.feature.chroma_stft(y=y, sr=sr)
```

```
        spec_cent = librosa.feature.spectral_centroid(y=y, sr=sr)
        spec_bw = librosa.feature.spectral_bandwidth(y=y, sr=sr)
        rolloff = librosa.feature.spectral_rolloff(y=y, sr=sr)
        zcr = librosa.feature.zero_crossing_rate(y)
        mfcc = librosa.feature.mfcc(y=y, sr=sr)
        to_append = f'{filename} {np.mean(chroma_stft)} {np.mean(spec_cent)}
{np.mean(spec_bw)} {np.mean(rolloff)} {np.mean(zcr)}'
        for e in mfcc:
            to_append += f' {np.mean(e)}'
        file = open('train.csv', 'a', newline='')
        with file:
            writer = csv.writer(file)
            writer.writerow(to_append.split())
```

3. Machine Learning Problem And Analysis

Test data has 5 different music genre like hip hop, rock, pop, jazz, metal. Our goal is predict genres of new musics for some application. Determining music genres is to use classification model so machine learning category is “Classification”.

4. Implementation To Machine Learning Algorithm

K-NN Algorithm

That work is supervised and classification problem. K-Nearest Neighbour algorithms is suitable for that problem because K-NN is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.

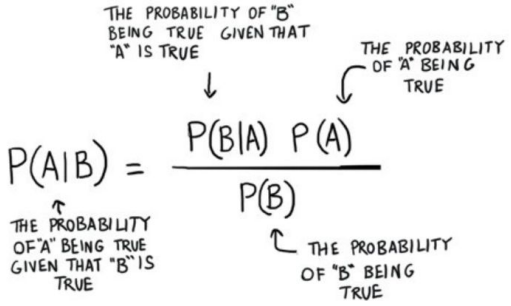
The K-NN working can be explained on the basis of the below algorithm:

Step-1: Select the number K of the neighbours.
Step-2: The distance of new data to be included in the sample data set is calculated individually with euclidean distance based on the existing data.
Step-3: Take the K nearest neighbors as per the calculated Euclidean distance.
Step-4: Among these k neighbors, count the number of the data points in each category.
Step-5: Assign the new data points to that category for which the number of the neighbor is maximum.

```
#KNN Algorithm
def euclidean_distance(x, y):
    summ=0
    for j in range(0,len(x)-1):
        summ+=(x[j]-y[j])**2
    return math.sqrt(summ)

def kNN(trainX,trainY,testX,k=10,clsCount=5):
    distances={}
    for i in range(0,len(trainX)):
        distances[i]=euclidean_distance(trainX[i],testX)
    sortedDistance= {k: v for k, v in sorted(distances.items(), key=lambda item: item[1])}
    sortedDistanceListForK=list(sortedDistance.keys())[:k]
    freqs=[0 for i in range(clsCount)]
    for i in range(len(sortedDistanceListForK)):
      freqs[ trainY[ sortedDistanceListForK[i] ] ] += 1
    print([(i/sum(freqs))*100 for i in freqs])
    maxC=max(freqs)
    return freqs.index(maxC)
```

Naive Bayes

The basis of the Naive Bayes classifier is based on Bayes' theorem. This theorem shows the relationship between conditional probabilities and marginal probabilities within a probability distribution for a random variable.



We can find the probability of A happening, given that B has occurred. Here, B is the evidence and A is the hypothesis. The assumption made here is that the predictors/features are independent. That is presence of one particular feature does not affect the other.

Naive Bayes working can be explained on the basis of the below algorithm:

Step-1: Convert the data set into a frequency table (Aggregation classes by features mean or etc.).
Step-2: Create Likelihood table by finding the probabilities.
Step-3: Using Naive Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction.

SVM (Support Vector Machine)

It is one of the most effective and simple methods used in classification. It is possible to separate the two groups by drawing a border between two groups in a plane for classification. The place where this limit will be drawn should be the most distant part of the members of both groups. SVM determines how this limit is drawn. It tries to find the best line that separates the two classes.
SVM can also classify linear and nonlinear data, but it usually tries to classify the data linearly.

Neural Network

Artificial neural networks are an information technology developed by inspiring the information processing technique of the human brain. Artificial Neural Networks mimic the way the simple biological nervous system works. It is the digital modeling of biological neuron cells and the synaptic bond that these cells establish with each other.

Neural Network is a structure built in layers. The first layer input is called the last layer output. The middle layers are called hidden layers. Each layer contains a certain number of 'Neuron'. These neurons are connected to each other with synapse.

Synapses contain a coefficient. These coefficients say how important the information in the neuron to which they are connected is. The value of a neuron is found by multiplying inputs with coefficients and then adding them to them. This found result is put into an activation function. According to the result of the function, it is decided whether or not that neuron will be fired.

```
def neuralN(trainX,trainY,testX):
    model = models.Sequential()
    model.add(layers.Dense(256, activation='relu', input_shape=(trainX.shape[1],)))

    model.add(layers.Dense(128, activation='relu'))

    model.add(layers.Dense(64, activation='relu'))

    model.add(layers.Dense(5, activation='softmax'))
```

```
model.compile(optimizer='Nadam',
              loss='sparse_categorical_crossentropy')
model.fit(trainX,
          trainY,
          epochs=25,
          batch_size=256)

y_pred=model.predict([testX])
y_pred = y_pred.tolist()[0]
y_prob=model.predict_proba([testX])

y_prob = y_prob.tolist()[0]
y_prob2 = [float("{:.3f}".format(float(i))) for i in y_prob]
y_prob2 = [(i/sum(y_prob2))*100 for i in y_prob2]
y_prob2 = [float("{:.2f}".format(float(i))) for i in y_prob2]
print(y_prob2)
print(y_pred.index(max(y_pred)))
return y_pred.index(max(y_pred))
```

5. Evaluating the Cross Validation and Metrics (Accuracy, Confusion Matrix)

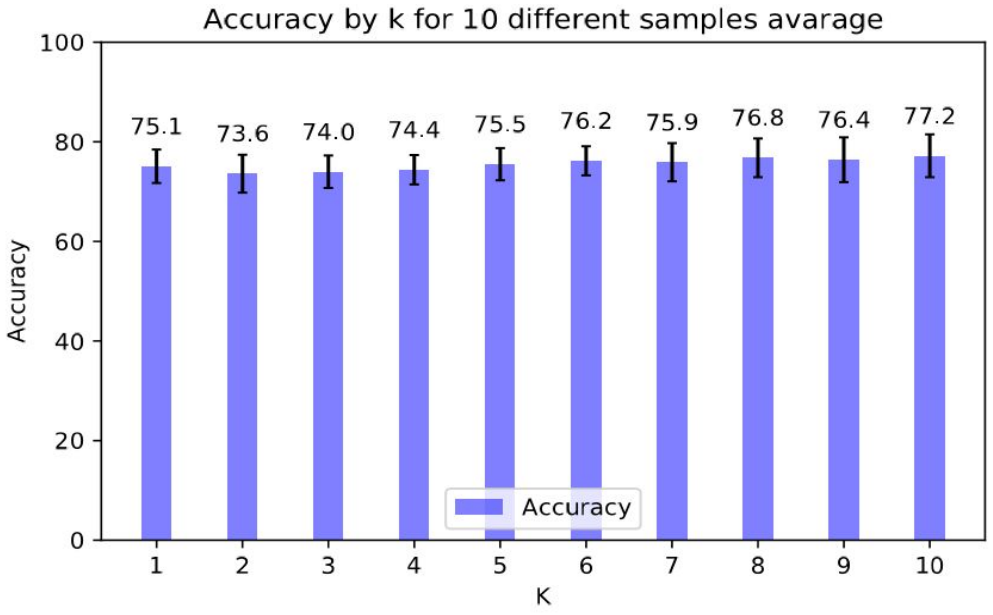
The K-NN Algorithm

Firstly, a k value for the cross validation method was chosen. k value was chosen 10 because it is the most preferred k value is 10. For k = 10, we first divide our data set into 10 equal parts. Our dataset contains 500 records. First of all, we selected 100 test items of dataset and we used the rest for training. Accuracy was calculated according to knn algorithm.

```
#Cross Validation Accuracy Calculation with k-nn for 10 sample
gss = GroupShuffleSplit(n_splits=10, test_size=0.2)

accuracy=[0 for count in range(10)]
for i in range(10):

  train_dataset,test_dataset = next(gss.split(X=X, y=y,
groups=dataSet.index.values))
```

```
 X_train, X_test, y_train, y_test = X[train_dataset], X[test_dataset],
y[train_dataset], y[test_dataset]

 knnAccurarcy=[0 for count in range(10)]
 for k in range(1,11):
  resultsArray=[]
  for j in range(0,len(X_test)):
    resultsArray.append(classifyDict(X_train,y_train,X_test[j],k))
  knnAccurarcy[k-1]=(skl.metrics.accuracy_score(y_test, resultsArray)*100)
 accuracy[i]=knnAccurarcy
```

```
Accuracy for 1 -nn: 75.10 (+/- 3.36)
Accuracy for 2 -nn: 73.60 (+/- 3.80)
Accuracy for 3 -nn: 74.00 (+/- 3.26)
Accuracy for 4 -nn: 74.40 (+/- 2.94)
Accuracy for 5 -nn: 75.50 (+/- 3.23)
Accuracy for 6 -nn: 76.20 (+/- 2.93)
Accuracy for 7 -nn: 75.90 (+/- 3.83)
Accuracy for 8 -nn: 76.80 (+/- 3.89)
Accuracy for 9 -nn: 76.40 (+/- 4.50)
Accuracy for 10 -nn: 77.20 (+/- 4.31)
```



According to the results, the average accuracy values are like this. The increasing k cause a great change in results.

We divided our data into 400 train 100 tests and used it for confusion matrix.

Confusion Matrix is a measurement tool that provides information about the accuracy of the estimates.We have 5 cases(prediction hiphop,prediction jazz,prediction metal, prediction pop,prediction rock) As you would expect, matrix represents a 5 to 5 matrix. 5 columns represent where our algorithm predicts, and rows represent what actually happens.

```
#Confusion Metrics for 1 sample test for 5NN
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

resultsArray=[]
for j in range(0,len(X_test)):
 resultsArray.append(classifyDict(X_train,y_train,X_test[j],5))
```

	hiphop	jazz	metal	pop	rock
hiphop	7	0	5	2	3
jazz	2	16	0	1	1
metal	2	0	17	0	1
pop	1	1	0	23	0
rock	3	2	3	2	8

Metrics Classification Report provides a quick idea of the accuracy of a model using a number of measures and displays the precision, recall, f1-score and support for each class. Lets understand these terms; **Precision** is out of all the positive classes we have predicted correctly, how many are actually positive. **Recall** is out of all the positive classes, how much we predicted correctly. It should be high as possible.**F1-score** helps to measure Recall and Precision at the same time. **Support** is the total value of how many estimates are made from classes.

```
print(skl.metrics.classification_report(y_test, resultsArray,labels=[0, 1, 2, 3, 4]))
```

	precision	recall	f1-score	support
0	0.47	0.41	0.44	17
1	0.84	0.80	0.82	20
2	0.68	0.85	0.76	20
3	0.82	0.92	0.87	25
4	0.62	0.44	0.52	18
accuracy			0.71	100
macro avg	0.69	0.69	0.68	100
weighted avg	0.70	0.71	0.70	100

Naive Bayes

Secondly, we try our dataset with Naive Bayes Classifier using Weka tool.

```
=== Summary ===

Correctly Classified Instances         298               59.6    %
Kappa statistic                          0.495
Mean absolute error                      0.1579
Root mean squared error                  0.3772
Relative absolute error                 49.3444 %
Root relative squared error             94.3025 %
Total Number of Instances              500

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure  MCC      ROC Area  PRC Area  Class
               0,350     0,055     0,614       0,350   0,446      0,371    0,854     0,573     hiphop
               0,640     0,068     0,703       0,640   0,670      0,594    0,919     0,806     jazz
               0,880     0,150     0,595       0,880   0,710      0,640    0,950     0,851     metal
               0,820     0,153     0,573       0,820   0,675      0,591    0,944     0,830     pop
               0,290     0,080     0,475       0,290   0,360      0,257    0,785     0,464     rock
Weighted Avg.  0,596     0,101     0,592       0,596   0,572      0,490    0,890     0,704

=== Confusion Matrix ===

  a  b  c  d  e   <-- classified as
 35  2 28 29  6 |  a = hiphop
  3 64  1 15 17 |  b = jazz
  4  0 88  0  8 |  c = metal
  7 10  0 82  1 |  d = pop
  8 15 31 17 29 |  e = rock
```

Naive Bayes Output show us, data set has the accuracy of the classifier is about 59%.

When using the Naive Bayes algorithm, the rock music genre has a TP (True Positive) Rate of 29%, then the hiphop has a 35% TP Rate, Naive Bayes are not enough for these two genres. The accuracy rate is more reliable in metal, jazz and pop.

Lets understand these terms;

Mean Absolute Error: quantity used to measure how close forecasts or predictions are to the eventual outcomes.
Kappa Statistic is a statistic that is used to measure inter-rater reliability for categorical items.
Root Mean Square Error represents the sample standard deviation of the differences between predicted values and observed values.
Relative Absolute Error is expressed as a ratio, comparing a mean error (residual) to errors produced by a trivial or naive model.
Root Relative Squared Error is relative to what it would have been if a simple predictor had been used. More specifically, this simple predictor is just the average of the actual values.

Support Vector Machine

We try our dataset with Support Vector Classifier using Weka tool.

```
=== Summary ===

Correctly Classified Instances         389               77.8    %
Kappa statistic                          0.7225
Mean absolute error                      0.253
Root mean squared error                  0.3359
Relative absolute error                 79.075  %
Root relative squared error             83.9821 %
Total Number of Instances              500

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure  MCC      ROC Area  PRC Area  Class
               0,750    0,088     0,682      0,750    0,714      0,640    0,868     0,595     hiphop
               0,810    0,033     0,862      0,810    0,835      0,796    0,937     0,810     jazz
               0,850    0,040     0,842      0,850    0,846      0,807    0,959     0,798     metal
               0,870    0,063     0,777      0,870    0,821      0,775    0,949     0,745     pop
               0,610    0,055     0,735      0,610    0,667      0,597    0,837     0,564     rock
Weighted Avg.  0,778    0,056     0,779      0,778    0,777      0,723    0,910     0,703

=== Confusion Matrix ===

  a  b  c  d  e   <-- classified as
 75  2  5 10  8 |  a = hiphop
  9 81  1  4  5 |  b = jazz
 10  0 85  0  5 |  c = metal
  5  4  0 87  4 |  d = pop
 11  7 10 11 61 |  e = rock
```

SVM Output show us, data set has the accuracy of the classifier is about 77.8%.

When using the SVM algorithm, the rock music genre has a TP (True Positive) Rate of 61% is less than SVM accuracy of 77.8%. It seems that SVM is not enough for this type. The accuracy rate is more reliable when it comes to hiphop, metal, jazz and pop.

Neural Network

We try our dataset with Neural Network Classifier using Weka tool.

```
=== Summary ===

Correctly Classified Instances         398               79.6     %
Kappa statistic                          0.745
Mean absolute error                      0.09
Root mean squared error                  0.2601
Relative absolute error                 28.1399 %
Root relative squared error             65.0281 %
Total Number of Instances              500

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,750    0,060    0,758      0,750    0,754      0,693    0,924     0,798     hiphop
                 0,870    0,060    0,784      0,870    0,825      0,780    0,970     0,895     jazz
                 0,840    0,033    0,866      0,840    0,853      0,817    0,976     0,907     metal
                 0,860    0,030    0,878      0,860    0,869      0,836    0,974     0,924     pop
                 0,660    0,073    0,695      0,660    0,677      0,599    0,905     0,723     rock
Weighted Avg.    0,796    0,051    0,796      0,796    0,795      0,745    0,950     0,849

=== Confusion Matrix ===

  a  b  c  d  e   <-- classified as
 75  4  6  4 11 |  a = hiphop
  3 87  1  5  4 |  b = jazz
  8  0 84  0  8 |  c = metal
  4  4  0 86  6 |  d = pop
  9 16  6  3 66 |  e = rock
```

Neural Network output show us, data set has the accuracy of the classifier is about 79.6%

When using the Neural Network algorithm, the rock music genre has a TP (True Positive) Rate of 66% It means that Neural Network is not enough for rock types. The accuracy rate is more reliable when it comes to metal, jazz, hiphop and pop.

6. Conclusion

In conclusion, to get better results for this project, 4 different classifiers have been tried as KNN with Python, Naive Bayes, Support Vector Machine and Neural Network with Weka tool.

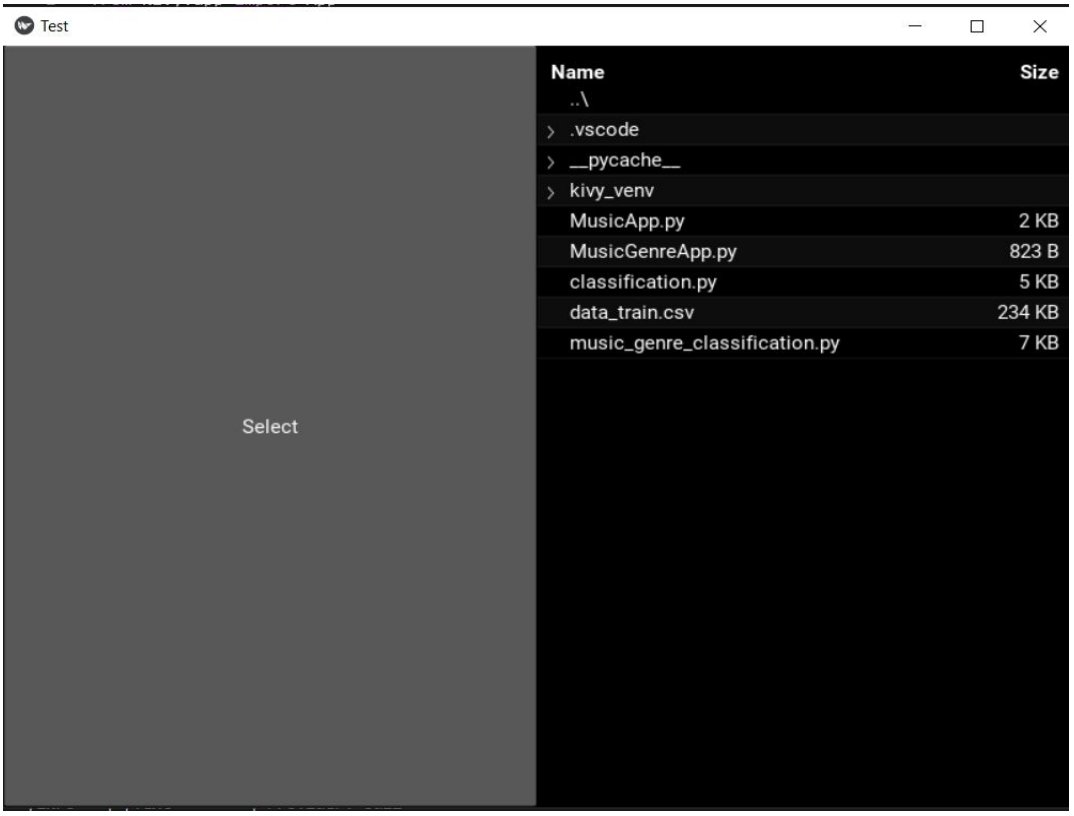
According to our operations, we found an accuracy of 70% in the KNN Algorithm, 50% in the Naive Bayes Algorithm, 77.8% in the Support Vector Machine Algorithm and 79.6% in the Neural Network Algorithm. KNN, Support Vector Machine and Neural Network algorithm results are very close to each other. The accuracy of the Naive Bayes algorithm is lower than others.

- As the Naive Bayes, 2/5 type provides reliability, its reliability rate is very low and for this Music Genre Classification application is not very reliable.
- As the SVM and Neural Network, 4/5 type provides reliability, their reliability rate are high and for this Music Genre Classification application are very reliable.
- The type of music that the methods used make the most false estimation is rock. They generally misprediction rock-type music.

Our project had quite educational aspects for beginners like us: PreProcessing, Feature Selection and Extraction, Algorithms and their properties etc. Project has a wide range of study space and it is still improvable.

7. Music Genre Classification UI Design

Main Screen works to select music from our files.



After selecting the music, algorithms for accuracy calculation are shown and results are given the desired algorithm.