Music Genre Recognition*

Adam Grelewicz[1,*,†], Mateusz Lis[1,†] and Dawid Michalak[1,*,†]

[1]Faculty of Applied Mathematics, Silesian University of Technology, Kaszubska 23, 44-100 Gliwice, POLAND

Abstract

Sound analysis plays a crucial role in identifying various types of defects in collaboration with artificial intelligence system models. To design a well-functioning model, a thorough data analysis is essential. Therefore, this article presents the implementation of the MFCC algorithm for different music genres. The algorithm is supported by high-pass and triangular filters. The recording will be transformed using the discrete Fourier transform (DFT). Then, the correctness of the algorithm will be verified using the KNN classifier and Naive Bayes to check the correct identification of the music genre. The project was conducted on a publicly available dataset. The results for the KNN classifier are very satisfactory. Additionally, this article demonstrates the superiority of the KNN classifier over Bayes for sound analysis.

Keywords

MFCC algorithm, Genre Recognition, Naive Bayes, KNN,

1. Introduction

Sound is a wave that arises from changes in atmospheric pressure caused by vibration [1]. Combined with artificial intelligence systems, it can have broad applications in various fields. In medicine, image recognition using deep learning is utilized. In [2], models are used to help specialists diagnose diseases more quickly. It is worth noting that sound contains a lot of information. Based on sound, certain abnormalities can be detected. In [3], the use of heart sounds for early disease detection is excellently demonstrated, allowing for earlier treatment. In article [12], there is another medical application, namely recognizing people with Parkinson's disease from recorded voice samples. The average accuracy of this method is around 90%.

Therefore sound should be converted into a spectrogram, and then a model for image recognition should be used. A spectrogram is a visual representation of the intensity of a signal over time, with respect to different frequencies present in a given waveform. The evaluation of spectrograms involves transforming the signal from the time domain to the frequency domain using the Fourier transform [4]. In [4], it is shown that sound can also be used in the food industry to identify various food products. In the following articles [8][9][10][11], various techniques utilizing sound recognition are described, such as Environmental Sound Recognition (ESR) and Automatic Sound Recognition (ASR), which can be used in a smart home. A smart home, along with artificial intelligence methods, can provide support for people, reduce exploration costs, and improve energy efficiency [13][14]. Therefore, this field also utilizes sound recognition. This mechanism can be used as one of the biometric security measures for homes [15]. However, the sound processing scheme is the same.

All these articles demonstrate that data analysis is very important for the application of neural networks. In particular, sound must be properly processed. Sound, especially human speech or music, has certain features that can be used for its characterization, such as a unique human voice, communication method-specific noise, or the use of similar instruments in musical pieces of the same genre [1]. Therefore, to extract the most important features of a sound signal in the form of a coefficient matrix, the MFCC algorithm, which will be described in detail in this article. Later in this article, there will be a comparison of two classifiers: KNN and Naive Bayes.

*IVUS2024: Information Society and University Studies 2024, May 17, Kaunas, Lithuania

[1,*]Corresponding author

[†] These author contributed equally.

ml307892@student.polsl.pl (M.Lis); dm307899@student.polsl.pl (D. Michalak); ag307868@student.polsl.pl (A. Grelewicz)

2. Methodology

2.1. The MFCC Algorithm (Mel-frequency cepstral coefficients)

Before describing the MFCC algorithm itself, certain concepts need to be defined:

1. Mel scale - a scale of pitches that measures the perceived frequency of sound, in contrast to the objective frequency scale measured in hertz.

   The function for converting a frequency in hertz to the Mel scale:

   $$m(f) = 1125 \log (1 + \frac{f}{700}) \quad (1)$$

   The inverse function:

   $$f(m) = 700(\exp \frac{m}{1125} - 1) \quad (2)$$

2. Window function is a function that takes non-zero values only within a specified interval. That functions are used to filter signals.

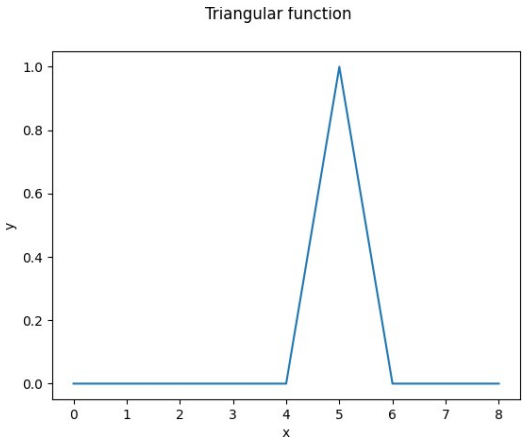

Figure 1: An example of a window function

For the purposes of mathematical description, let's introduce the following notation:

1. $\{...\}^*$ denotes an array, which is a set where elements can repeat and maintain order. An array of arrays is called a matrix.
2. If A is an array, the notation $A[k]$ means the k-th element of A.
3. If A is an array, the notation $A[:k]$ means the first k elements of A.
4. All other operations on arrays work similarly to operations on sets.

Description of the MFCC algorithm:

1. Let:
   - n be the number of samples in the input signal,
   - $X = \{x_0, x_1, ..., x_{n-1}\}^*$ be the input signal,
   - h be the sampling frequency of the input signal in hertz,
   - t be the number of triangular filters,
   - u be the number of numbers transformed by the discrete Fourier transform,
   - l be the length of the window in samples,
   - s be the number of samples by which the window is shifted and $0 < s \leq l$,
   - c be the number of cepstral coefficients.
2. Filters are applied to remove noise. This step is optional, but noise removal improves accuracy, so a high-pass filter is used in the form of:

$$\begin{aligned} &y_k = x_k - 0.97x_{k+1} \\ &k \in \{0, 1, 2, ..., n-2\}^* \end{aligned} \tag{3}$$

3. Triangular filters are created to extract desired features from the input signal while omitting unnecessary ones. These filters are distributed on a frequency scale between 0 and $\frac{h}{2}$. Initially, the boundaries of this scale are converted from frequency in hertz to the Mel scale:

$$\begin{cases} a = m(0) = 0 \\ b = m\left(\frac{h}{2}\right) = 1125 \log\left(1 + \frac{h}{1400}\right) \end{cases} \tag{4}$$

An array of $t+2$ numbers is created, evenly distributed between a and b:

$$\begin{aligned} U &= \{a + k\Delta x : k \in \{0, 1, 2, ..., t+1\}\}^* \\ \Delta x &= \frac{b-a}{t+1} \end{aligned} \tag{5}$$

Another array is created containing the scaled elements of array U, converted from the Mel scale to the frequency scale in hertz and rounded down.

$$\boxtimes = \left\{ \left\lfloor \frac{u+1}{h} f(x) \right\rfloor : x \in U \right\}^* \tag{6}$$

Array $\boxtimes$ contains non-linearly distributed numbers from 0 to $\left\lfloor \frac{u}{2} \right\rfloor$.

Another array is created to contain the triangular filters mentioned at the beginning.

$$F = \{\{g(k, j) : j \in \{0, 1, 2, \ldots, \lfloor \frac{u}{2} \rfloor\}^*\}_* : k \in \{0, 1, 2, \ldots, t-1\}^*\}_* \quad (7)$$

where:

$$g(k, j) = \begin{cases} \frac{j - B[k]}{B[k+1] - B[k]} & , \lfloor \boxtimes[\boxtimes] \rfloor \leq j \wedge \boxtimes < \lfloor \boxtimes[k+1] \rfloor \\ \frac{B[k+2] - j}{B[k+2] - B[k+1]} & , \lfloor B[k+1] \rfloor \leq \boxtimes \wedge \boxtimes < \lfloor \boxtimes[k+2] \rfloor \\ 0 & \text{, for other } \boxtimes \end{cases}$$

4. The input signal is divided into windows, where a window is defined as:

$$W(i) = \{p(\boxtimes) : si \leq \boxtimes < si + l\}^*, i \in \{0, 1, 2, \ldots, \lfloor \frac{n + s - n \bmod s}{s} \rfloor - 1\}^* \quad (8)$$

where:

$$p(k) = \begin{cases} x_k & , 0 \leq k < n \\ 0 & , n \leq k \end{cases}$$

i is the window index.

The power spectrum is calculated, i.e., the discrete Fourier transform (DFT) of the first u elements from the array $W(i)$, then square each number in the resulting array and scale these elements by $\frac{1}{l}$.

$$P(i) = \frac{1}{l} \mathcal{S}(\text{DFT}(W(i), u)) \quad (9)$$

where $P(i)$ denotes the i-th power spectrum for the i-th window $W(i)$ and

$$\mathcal{S}(\{s_0, s_1, \ldots, s_m\}^*) = \{|s_0|^2, |s_1|^2, \ldots, |s_m|^2\}^*$$

Absolute values are required in function $\mathcal{S}$ as s_k can be complex numbers.

The previously calculated filters are then utilized to filter the power spectrum via the matrix product of $P(i)$ and the transpose of matrix F.

$$C(i) = P(i)F^T \quad (10)$$

The final step is to compute the natural logarithm for each element of $C(i)$ and transform these logarithms using the discrete cosine transform (DCT) of type II.

$$R(i) = \text{DCT}(L(C(i)))[: c] \quad (11)$$

where $L(\{x_0, x_1, \ldots, x_m\}^*) = \{\ln x_0, \ln x_1, \ldots, \ln x_m\}^*$.

The result of the algorithm is a matrix:

$$R = \{R(0), R(1), R(2), \ldots, R(\lfloor \frac{n + s - n \bmod s}{s} \rfloor - 1)\}^* \quad (12)$$

2.2. KNN (k-Nearest-Neighbours)

The K-Nearest Neighbors (KNN) algorithm is a classification and regression method that utilizes the similarity between data points. It operates by finding the nearest neighbors (data points) to a new point and uses their information to predict the class or value for that point [5]. Before describing the KNN algorithm itself, certain concepts need to be defined:

Value of k - The number of neighbors to be considered during classification or regression. Mahalanobis distance - It considers the correlations between two vectors x and y with covariance matrix S and scales distances depending on the distribution of data. It is given by the formula:

$$D_S(x, y) = \sqrt{(x - y)^\top S^{-1}(x - y)} \tag{13}$$

2.3. Naive Bayes classifier

The Naive Bayes classifier is a machine learning method used for classifying data into decision classes. Despite its simplicity, it has a wide range of applications in text classification, medical diagnosis, and system performance management. The task of the Bayes classifier is to assign a new case to one of the classes [6][7]. Each training example is described by a set of conditional attributes $\{X_i\}$ and one decision attribute X_d. According to Bayes' theorem, the most probable class to which a new object, described by the values of n-conditional attributes $\langle x_{j1}, x_{j2}, \ldots, x_{jn}\rangle$, belongs is the class d_i that maximizes the conditional probability $P(d_i \mid x_{j1}, x_{j2}, \ldots, x_{jn})$.

$$d' = \arg\max_{d_i \in V_D} P(d_i) \cdot P(x_{j1}, x_{j2}, \ldots, x_{jn} \mid d_i) \tag{14}$$

The probability $P(d_i)$ can be estimated as the ratio of the number of training examples belonging to class d_i to the total number of training examples. To estimate $P(x_{j1}, x_{j2}, \ldots, x_{jn} \mid d_i)$, the Naive Bayes classifier assumes the conditional independence of attributes:

$$P(x_{j1}, x_{j2}, \ldots, x_{jn} \mid d_i) = \prod_{k=1}^{n} P(x_{jk} \mid d_i) \tag{15}$$

The probability $P(x_{jk} \mid d_i)$ can be estimated as the ratio of the number of training examples in class d_i for which the attribute X_{jk} has the value x_{jk} to the total number of training examples in class d_i. Considering this assumption, the class d_{NB} (Naïve Bayes) chosen for a new example is:

$$d_{NB} = \arg\max_{d_i \in V_D} P(d_i) \cdot \prod_{k=1}^{n} P(x_{jk} \mid d_i) \tag{16}$$

3. Experiments

The example of MFCC algorithm will be conducted using the file *classical.00000.wav*.

1. The sampling frequency for this file is $h = 22050$ Hz.
   The length of the file is 30.013 s.

The number of samples is $n = 661794$.
The sound samples are $X = \{x_0, x_1, ..., x_{n-1}\}^*$. The plot of the samples of this file is on Figure 2

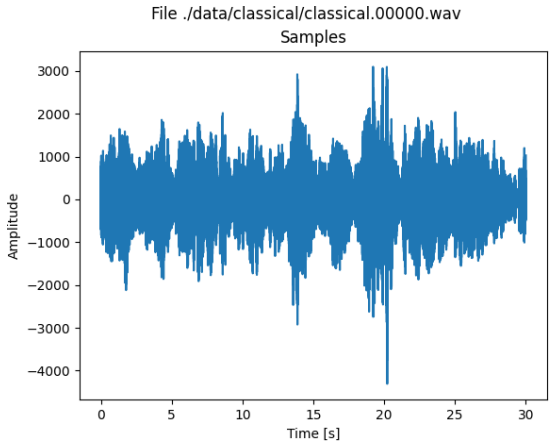

Figure 2: Representation of samples in audio.

The number of triangular filters $t = 26$.
The number of values transformed by the discrete Fourier transform $u = 512$. The length of the window in samples $l = 551$.
The number of samples by which we shift the window $s = 220$. The number of cepstral coefficients $c = 13$.

2. After applying the high-pass filter, the samples look as follows on Figure 3

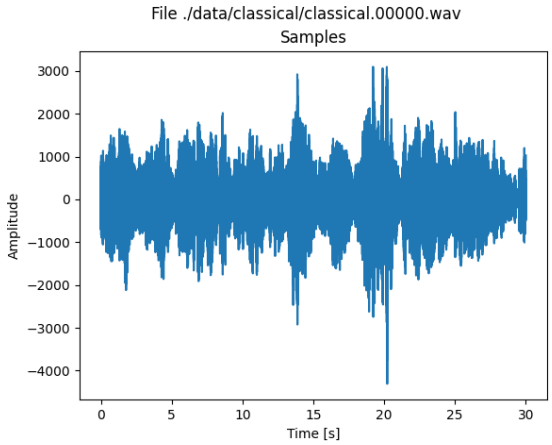

Figure 3: That sample after it was filtered.

The difference between the samples before and after the filter is not visible at first glance,

but applying this filter to each file improved accuracy by about 4%.

3. Triangular filters are created. According to the formulas in the algorithm description, the boundary frequencies are obtained.

$$\begin{cases} a = m(0) = 0 \\ b = m\left(\frac{22050}{2}\right) = 1125 \log\left(1 + \frac{22050}{1400}\right) = \\ \qquad = 1125 \log \frac{67}{4} \end{cases}$$

An array of $t+2$ numbers evenly distributed between a and b is created.

$$\Delta x = \frac{1125 \log \frac{67}{4}}{27} = \frac{125}{3} \log \frac{67}{4}$$

$$U = \{a + k\Delta x : k \in \{0, 1, 2, ..., 27\}\}^*$$

$$\Rightarrow U = \{\frac{125k}{3} \log \frac{67}{4} : k \in \{0, 1, 2, ..., 27\}\}^*$$

$$\Rightarrow U = \{0, \frac{125}{3} \log \frac{67}{4}, \frac{250}{3} \log \frac{67}{4}, ..., \frac{3375}{3} \log \frac{67}{4}\}^*$$

The array ⊠ is created according to the formula in the description.

$$B = \{\left\lfloor \frac{513}{22050} f(x) \right\rfloor : x \in U\}^*$$

$$\Rightarrow B = \{\left\lfloor \frac{513}{22050} f(0) \right\rfloor, \left\lfloor \frac{513}{22050} f(\frac{125}{3} \log \frac{67}{4}) \right\rfloor, ..., \left\lfloor \frac{513}{22050} f(\frac{3375}{3} \log \frac{67}{4}) \right\rfloor\}^*$$

$$\Rightarrow ⊠ = \{\lfloor 0 \rfloor, \left\lfloor \frac{5130}{315} (2^{7 \frac{67}{4}} - 1) \right\rfloor, ..., \lfloor 256.5 \rfloor\}^*$$

$$\Rightarrow B = \{0, 1, ..., 256\}^*$$

The array ⊠ looks as follows:

$$\Rightarrow B = \{0, 1, 3, 5, 8, 11, 14, 17, 21, 25, 29, 35, 40, 46, 53, 61, 70, 79, 90, 102, 115, 129, 145, 163, 183, 205, 229, 256\}^*$$

As can be seen, the array ⊠ contains numbers ranging from 0 to $\lfloor \frac{u}{2} \rfloor = 256$.

As can be seen, these numbers are not evenly distributed, meaning the differences between consecutive numbers increase as the elements progress. This is because the boundary frequencies

a and b were converted from the frequency scale in Hertz to the Mel scale, which is nonlinear. The reason why a change to a nonlinear scale was required will be explained later in the example.

Triangular filters are created, again, according to the formula from the description:

$$F = \{\{g(k, j) : j \in \{0, 1, 2, ..., 256\}^*\}^* : k \in \{0, 1, 2, ..., 25\}^*\}^*$$

For $k = 0$ the function $g(0, j)$ will look like:

$$g(0, j) = \begin{cases} \frac{j - B[0]}{B[1] - B[0]}, & \lfloor \boxtimes[0] \rfloor \leq \boxtimes < \lfloor \boxtimes[1] \rfloor \\ \frac{B[2] - j}{B[2] - B[1]}, & \lfloor \boxtimes[1] \rfloor \leq \boxtimes < \lfloor \boxtimes[2] \rfloor \\ \\ 0, & \text{for other } \boxtimes \end{cases}$$

$$\Rightarrow g(0, j) = \begin{cases} j, & 0 \leq \boxtimes < 1 \\ \\ \frac{3 - j}{2}, & 1 \leq j < 3 \\ 0, & \text{for other } \boxtimes \end{cases}$$

The array $\{g(0, j) : \boxtimes \in \{0, 1, 2, ..., 256\}^*\}^*$ (the first element of F) looks like:

$$\{g(0, j) : j \in \{0, 1, 2, ..., 256\}\}^* = \{0, 1, \frac{1}{2}, 0, ..., 0\}^*$$

Representing this filter on a graph on figure 4.

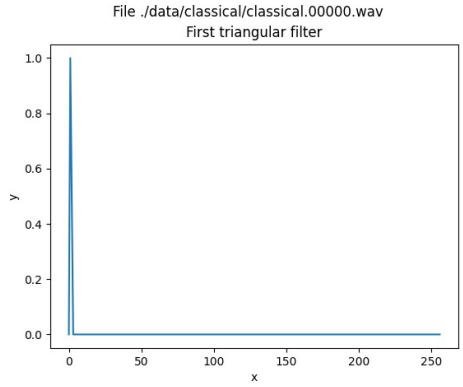

Figure 4: Representation of a triangular filter for that sample

As can be seen, this filter is very narrow. It passes only what is at the beginning and zeroes out the rest.

After calculating all the values $g(\boxtimes, j)$, all $t = 26$ filters can be represented on a graph on figure 5

Due to the application of the Mel scale to distribute these filters, the highest density is at the beginning, and the lowest at the end.

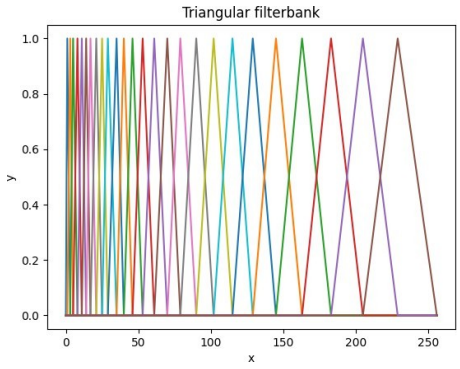

Figure 5: Representation of the triangular filterbank

The reason for needing the Mel scale is that it accurately represents how humans perceive sound. It turns out that most useful information is in the lower frequencies, not the higher ones. Therefore, it makes sense to place more filters at the beginning, which was achieved by converting the frequency scale in Hertz to the Mel scale. Without this, all filters would be evenly distributed across the entire scale.

It was tested what would happen if evenly distributed filters were used, and it degraded the accuracy by about 5%. The input signal is divided into windows.

$$W(i) = \{p(k) : 220i \leq k < 220i + 551\}^*$$
$$, i \in \{0, 1, ..., 3008\}^*$$
$$p(k) = \begin{cases} x_k & , 0 \leq k < 661794 \\ 0 & , 661794 \leq k \end{cases}$$

For the window $i = 0$:

$$W(0) = \{p(k) : 0 \leq k < 551\}^*$$
$$\implies W(0) = \{x_0, x_1, x_2, ..., x_{550}\}^*$$
$$\implies W(0) = \{-102.19, -705.89, -136.54, ..., 123.9\}^*$$

For the window $i = 1$:

$$W(1) = \{p(k) : 220 \leq k < 771\}^*$$
$$\implies W(1) = \{x_{220}, x_{221}, x_{222}, ..., x_{770}\}^*$$

For the window $i = 3008$ (the last window):

$$W(3008) = \{p(k) : 661760 \leq k < 662311\}^*$$
$$\implies W(3008) = \{x_{661760}, x_{661761}, x_{661762}, ...,$$
$$x_{661793}, 0, 0, ..., 0\}^*$$

It is worth noting that for $i = 3008$ the index k goes beyond the input signal, so there are zeros at the end.

Now the power spectral density is calculated:

$$P(i) = \frac{1}{551}\left|\text{DFT}(W(i), 512)\right|$$

For the window $i = 0$:

$$P(0) = \frac{1}{551}\left|\text{DFT}(W(0), 512)\right|$$
$$\Rightarrow P(0) = \frac{1}{551}\left|\text{DFT}(\{x_0, x_1, x_2, ..., x_{550}\}^*, 512)\right|$$
$$\Rightarrow P(0) = \{476.280941, 6620.61200, 8149.37465, ..., 195.371343\}^*$$

Having the power spectral density, the matrix product $P(i)$ and F^F is calculated, which will be the operation of filtering frequencies according to the triangular filters previously established.

$$C(i) = P(i)F^F \tag{17}$$

For $i = 0$ we get:

$$C(0) = P(0)F^F$$
$$\Rightarrow C(0) = \{10695.2993, 31658.4727, 20555.0554, ..., 18245.6147\}^*$$

Finally, the discrete cosine transform of the logarithms of $C(i)$ is calculated, taking only the first $c = 13$ elements:

$$R(i) = \text{DCT}(L(C(i)))[: 13]$$

For $i = 0$ we get:

$$R(0) = \text{DCT}(L(C(0)))[: 13]$$
$$\Rightarrow R(0) = \text{DCT}(L(\{10695.2993, 31658.4727, 20555.0554, ..., 18245.6147\}^*))[: 13]$$
$$\Rightarrow R(0) = \{62.5650537, -2.03586229, -5.32321543, ..., -1.89111160\}^*$$

Finally, the array R is obtained, which is:

$$R = \{R(0), R(1), R(2), ..., R(3008)\}^*$$

Thus, the input signal is represented in the form of a matrix of cepstral coefficients.

Analysis of the results is conducted for 6 classes of abstraction, with the following music genres:

- Classical music,
- Disco,
- Hip-hop,
- Metal,
- Blues,
- Country.

For each genre, there are 100 assigned tracks, each lasting 30 seconds. The split between training and test data is 70:30.

Before conducting a detailed analysis, it is important to determine the most effective value of k for the KNN classifier. According to the Table 2, it can be seen that the most effective value is k = 5, therefore this value should be adopted for the analysis.

k	Accuracy
3	77.78%
4	78.89%
5	80.56%
6	77.78%
7	77.22%
8	77.78%
9	76.67%
10	75.56%

Table 1
Accuracy of the KNN algorithm depending on the number of neighbors.

The next step is to evaluate the obtained matrices with the KNN classifier and Naive Bayes. Performance evaluation metrics such as accuracy, loss, precision, recall, and F1 score will be used to assess the effectiveness of these methods [4]. These metrics are essential for evaluating the performance of machine learning models and are described by the following equations:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{18}$$

$$\text{Loss} = \frac{FP + FN}{TP + TN + FP + FN} \tag{19}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{20}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{21}$$

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{22}$$

where:

- TP (True Positive), which is the number of cases where the model correctly classified positive instances.
- TN (True Negative), which is the number of cases where the model correctly classified negative instances.
- FP (False Positive), which is the number of cases where the model incorrectly classified negative instances as positive.
- FN (False Negative), which is the number of cases where the model incorrectly classified positive instances as negative.

	Class	Disco	Hiphop	Metal	Blues	Country
Accuracy	0.983	0.878	0.939	0.950	0.950	0.911
Sen-Rec	0.926	0.767	0.968	0.865	0.667	0.607
Precision	0.962	0.605	0.750	0.889	1.000	0.773
F1	0.943	0.677	0.845	0.877	0.800	0.680
Specificity	0.993	0.900	0.933	0.972	1.000	0.967

Table 2
Performance evaluation metrics for 6 genres with KNN.

	Class	Disco	Hiphop	Metal	Blues	Country
Accuracy	0.839	0.767	0.867	0.850	0.772	0.639
Sen-Rec	0.185	0.367	0.258	0.594	0.074	0.643
Precision	0.417	0.323	0.889	0.647	0.111	0.247
F1	0.256	0.344	0.400	0.620	0.089	0.356
Specificity	0.954	0.847	0.993	0.916	0.895	0.638

Table 3
Performance evaluation metrics for 6 genres with Naive Bayes .

As we can see from the Table 2, 3, the metric values are very good for KNN with 6 classes of abstraction, whereas Naive Bayes performs significantly worse. In terms of accuracy for the entire test set, KNN achieved 80.56%, while the naive Bayes classifier achieved 36.67%. On the Figure 6, 7 we observe the confusion matrix. An ideal confusion matrix has 100% on the diagonal, and the rest should be 0%. For KNN, the matrix is nearly ideal. The classifier performed worst for the disco music genre. However, for Naive Bayes, the confusion matrix does not resemble the ideal one. Nevertheless, the algorithm performed best for the metal genre.

4. Conclusion

This MFCC algorithm allows for highly efficient classification of music genres with the KNN classifier. The covariance matrix effectively extracted features from the audio signal and could be used for commercial purposes such as in medicine. In the case of classifiers, KNN uses distance metrics that can be very effective in measuring similarities between musical pieces. Additionally, it does not assume any specific form of the classification function, relying instead on local similarities, which is why it worked perfectly here. On the other hand, the advantages of using the naive Bayes classifier are its simplicity and speed compared to KNN. Naive Bayes assumes that the features are independent, which is rarely true for audio data where different features can be strongly correlated. In this project, only one feature was used, namely the mean

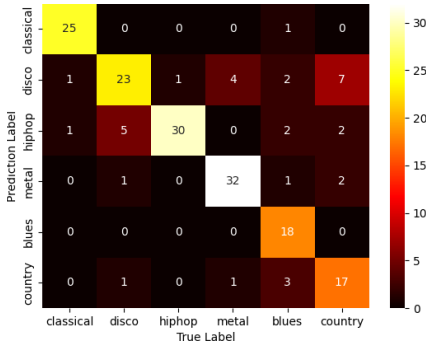

Figure 6: Confusion matrix for the KNN

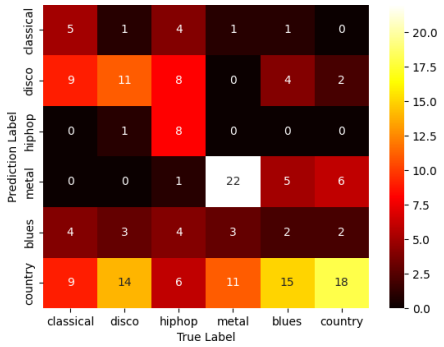

Figure 7: Confusion matrix for the naive Bayes classifier

of the sum of all elements of the matrix, which may have influenced the low accuracy compared to KNN. To achieve high accuracy, more advanced methods such as neural networks like CNN and RNN should be used. In the future, based on the matrix obtained from the MFCC algorithm, a spectrogram can be created and the given algorithm can be tested on more complex models to achieve better results.

References

[1] Malgorzata Przedpelska-Bieniek, "Dzwiek i akustyka. Nauka o dzwieku," 2011.

[2] Marcin Woźniak, Jakub Siłka, Michał Wieczorek, "Deep neural network correlation learning mechanism for CT brain tumor detection," 2021.

[3] Junxin Chen, Zhihuan Guo, Xu Xu, Li-bo Zhang, Yue Teng, Yongyong Chen, Marcin Woźniak, Wei Wang, "A Robust Deep Learning Framework Based on Spectrograms for Heart Sound Classification," 2023.

[4] Yogesh Kumar, Apeksha Koul, Kamini, Marcin Woźniak, Jana Shafi, Muhammad Fazal Ijaz,

"Automated detection and recognition system for chewable food items using advanced deep learning models," 2024.
[5] Bartosz A. Nowak, Robert K. Nowicki, Marcin Woźniak, Christian Napoli, "Multi-class Nearest Neighbour Classifier for Incomplete Data Handling."
[6] I. Rish, "An empirical study of the naive Bayes classifier," 2001.
[7] Harry Zhang, "The Optimality of Naive Bayes," 2004.
[8] Sachin Chachada, C.-C. Jay Kuo, "Environmental sound recognition: a survey," 2014.
[9] Michael Cowling, Renate Sitte, "Comparison of techniques for environmental sound recognition," 2003.
[10] Roneel V. Sharan, Tom J. Moir, "An overview of applications and advancements in automatic sound recognition," 2016.
[11] Jia-Ching Wang, Hsiao-Ping Lee, Jhing-Fa Wang, Cai-Bei Lin, "Robust Environmental Sound Recognition for Home Automation," 2008.
[12] Junxin Chen, Wei Wang, Bo Fang, Yu Liu, Keping Yu, Victor C. M. Leung, Xiping Hu, 'Exploiting Smartphone Voice Recording as a Digital Biomarker for Parkinson's Disease Diagnosis," 2023.
[13] Marcin Woźniak, Dawid Połap, 'Intelligent Home Systems for Ubiquitous User Support by Using Neural Networks and Rule-Based Approach," 2020.
[14] Richard Hauxwell-Baldwin, Charlie Wilson, Tom Hargreaves 'Learning to live in a smart home," 2017.
[15] Jessamyn Dahmen, Brian L. Thomas, Diane J. Cook, Xiaobo Wang, 'Activity Learning as a Foundation for Security Monitoring in Smart Homes," 2017.