# Music Genre Classification

Yiğit Can Akçay[1], Fatih Semirgin[2], Ahmet Başbuğ[3]

[1] Dokuz Eylül University, Faculty of Engineering, Department of Computer Engineering, Izmir, Turkey
[2] Dokuz Eylül University, Faculty of Engineering, Department of Computer Engineering, Izmir, Turkey
[3] Dokuz Eylül University, Faculty of Engineering, Department of Computer Engineering, Izmir, Turkey

**Abstract**

Music occupies an important place in human life. There are many different types of music around the world. Music genre classification plays a critical role in providing personalized recommendations to users and enhancing user experience in various multimedia applications. However, classifying these music individually is a very difficult task without an automated approach. Therefore, in this study, an automatic approach for music genre classification is proposed by utilizing machine learning techniques. Using the GTZAN dataset, various features were obtained from the audio files. Using these obtained features, popular models such as K-Nearest Neighbor (KNN), Random Forest, XGBoost, Support Vector Machine and Artificial Neural Networks (NN) were studied. The classification performance of each model was evaluated and compared through various metrics.

**Keywords:** Music Genre Classification, Deep Learning, Neural Network, Random Forest, XGBoost, KNN, Support Vector Machine.

## 1. Introduction

Music is an art form that is deeply rooted in human life as a form of cultural expression, creating a wide impact on emotional, social and mental levels. Many genres of music from different cultures, geographies and times around the world serve as a means for people to express and share their emotional richness. This diversity stands out as a factor that enriches the music listening experience.

Nowadays, digital music platforms and various multimedia applications allow users to choose from a wide range of music. However, music genre classification has an important role in order to more effectively explore this large music catalog and provide personalized recommendations to users. Music genre classification can help users find music that suits their taste and can also better organize music-related content.

Music genre classification has traditionally been performed by human experts; However, this approach is time consuming and costly. To overcome this challenge, an automatic music genre classification method using machine learning techniques is proposed in this study.

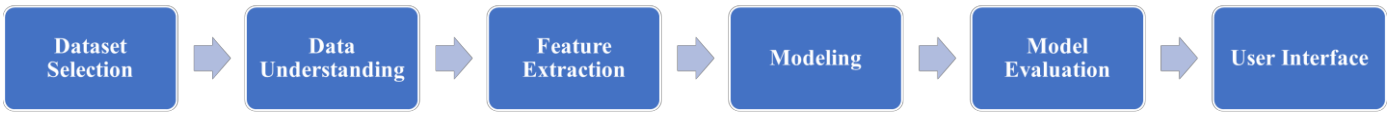
GTZAN dataset is the data source that forms the basis of this study. This dataset contains a total of one thousand 30-second audio files representing ten different musical genres. There are 100 samples from each musical genre, making the dataset diverse and balanced. There is a wide range of music genres in the data set, such as rock, pop, blues, country, jazz, metal, hip-hop, reggae, classical and disco. Each genre is distinguished by its characteristic features, which creates an ideal basis for music genre classification.

In our study, extracting various features from these 30-second audio files is an important step. The feature extraction process involves converting musical features in audio files into numerical values. These features can be fundamental characteristics of music, such as frequency components, rhythm, harmony and spectral features. These extracted features are given as input to machine learning models and music genre classification is performed.

As a result, this study aims to develop an automatic approach for music genre classification using the GTZAN dataset. Machine learning models used include popular models such as K-Nearest Neighbor (KNN), Random Forest, XGBoost, Support Vector Machine, and Artificial Neural Networks (NN). These models were evaluated and compared through various metrics. This study aims to contribute to the development of a music genre classification model that can be used for providing better personalized recommendations to music listeners.

This is the workflow diagram we followed during the execution of this study, as depicted in Figure 1.

*Figure 1. Workflow Diagram*



## 2. Literature Review

Music genre classification is becoming increasingly important today in order to understand the dynamics in the music industry, categorize musical works and reach target audiences more effectively. As a result, there are many studies on this subject in the literature.

Tzanetakis and Cook's study titled "Musical Genre Classification of Audio Signals" is one of the important studies in the literature on this subject. The main aim of the study is to provide effective classification in a wide range of musical genres by evaluating feature sets obtained through the use of audio processing algorithms. Tzanetakis and Cook identified the prominent feature sets as timbral texture, rhythmic content and pitch content. Using the proposed feature sets, a classification success of 61% was achieved for ten music genres (Tzanetakis, G., & Cook, P. (2002)).

In recent years, the use of artificial neural networks (ANNs) in the field of music genre classification has attracted attention. Work by Nikki Pelchat and Craig M. Gelowitz (2020) explores the applications of NNs in this context. The use of spectrograms as a key input feature highlights the importance of extracting meaningful information from audio signals for the accuracy of music genre classification. In this study, a deep CNN is used by making changes in the number of music genres, activation functions and DSP window functions. Their findings show that the accuracy of NN increases when the number of songs per genre is balanced, achieving 85% accuracy on test data (Pelchat, N., & Gelowitz, C. M. (2020)).

Another article introduces an effective automatic classification approach using multilayer support vector machines (SVM) for music genre classification. Various features are used to characterize musical content, such as beat spectrum, linear prediction coefficients, zero crossing rates, time energy, and mel frequency septral coefficients. SVM learning algorithms are used to learn optimal class boundaries between different music genres such as classical/jazz and pop/rock. It is stated that the error rate of the proposed multilayer SVM method is 6.86%, whereas other methods (NN - Nearest Neighbor, GMM - Gaussian Mixture Model, HMM - Hidden Markov Model) have error rates of 20.57%, 12.3% and 11.94%, respectively. These results show that the SVM method performs better than other methods (Xu, C., Maddage, N. C., Shao, X., Cao, F., & Tian, Q. (2003, April)).

Haggblade have evaluated various machine learning algorithms to classify four basic music genres: classical, jazz, metal and pop. They characterized the data using Mel Frequency Cepstral Coefficients (MFCC) and performed music genre classification using algorithms such as k-NN, k-means, multi-class SVM and neural networks. In particular, the use of MFCC has been used to optimize data representation and comparison operations. They have also made an interesting extension by using the Fourier-Mellin 2D transform to match musical genres with visual elements. Overall, k-NN and k-Means algorithms were found to have similar accuracy rates, around 80%. DAG SVM achieved an accuracy of 87% and neural networks achieved an accuracy rate of 96% (Haggblade, M., Hong, Y., & Kao, K. (2011)).

Jitesh Kumar Bhatia, Rishabh Dev Singh and Sanket Kumar also worked on music genre classification. In the tests performed on 1000 music tracks in the GTZAN database, when the accuracy rates are examined, it is seen that the Gaussian Model is 61%, the Logistic Regression Model is 75%, the CNN Model is 88.54%, the CRNN Model is 53.5%, and the CNN-RNN Model is 56.4%. It has been observed that Simple Artificial Neural Network has an accuracy rate of 64.06% and the proposed KNN Model has an accuracy rate of 80.63%. These results reveal that the KNN Model in particular exhibits impressive performance and provides accuracy competitive with other models in the literature (Bhatia, J. K., Singh, R. D., & Kumar, S. (2021, October)).

In a different study, focusing on the deep learning-based Convolutional Long Short-Term Memory deep neural network (CLDNN) architecture for the purpose of music genre classification. An original Turkish Music Database consisting of 200 musical pieces containing various musical genres of Turkish has also been developed. The main purpose of the research is to evaluate the effectiveness of machine learning and deep learning techniques on music genre classification. Common classifiers such as Naive Bayes, K Nearest Neighbors, Random Forests were used in the experiments, and deep learning methods such as LSTM + DNN were examined. The results obtained show that feature selection makes a significant contribution to classification performance and MFCC features offer a distinct advantage in this context. It has been determined that deep learning methods achieve higher accuracy rates compared to traditional classification methods. Research results show that the proposed method offers an effective strategy for music genre classification and 99.5% success is achieved, especially with the LSTM + DNN model (Hızlısoy S., & Tüfekçi, Z. (2021)).

Lastly, a research was carried out on the properties of audio signals using SVM, K-NN and Logistic Regression algorithms for the classification of Turkish music. In the study, the success of the algorithms used in music genre classification under various attributes was subjected to a comparative analysis. Comparative analyzes of the impact of each feature on classification are also presented. The data set, which consists of a total of 600 wav extension music tracks, includes 6 different types of music: arabesque, pop, rock, religious, rap and classical music. According to the results obtained, the SVM algorithm gave the most successful results, reaching the highest accuracy rate of 78.65% (Özbalcı, M. C. (2022)).

# 3. Material and Methods

## 3.1. Dataset Description

GTZAN dataset was used in this study. GTZAN data set is a data set used in the field of music genre recognition (MIR - Music Information Retrieval). This data set was created by George Tzanetakis and Perry Cook at Columbia University. The GTZAN dataset is widely used to evaluate music genre classification algorithms. Table 1 lists the music genres and their numbers in the dataset. Apart from this, all music files are in wav. format.

*Table 1. Music Types*

| ***Music Type*** | ***Number of files*** |
|---|---|
| *Blues* | *100* |
| *Classical* | *100* |
| *Country* | *100* |
| *Disco* | *100* |
| *Hip-hop* | *100* |
| *Jazz* | *100* |
| *Metal* | *100* |
| *Pop* | *100* |
| *Reggae* | *100* |
| *Rock* | *100* |

## 3.2. Feature Extraction

Before moving on to model training, digital features need to be extracted from these sound files. First, each 30 seconds of music in the dataset was divided into 3-second segments. The reason for this is to increase the number of data in the dataset. Various features have been extracted for each section. These features include sound spectrum features (Chroma STFT, RMS, Spectral Centroid, Spectral Bandwidth, Spectral Rolloff, Zero Crossing Rate), harmonic and rhythmic features (Harmony, Percussiveness), Tempogram, Tonnetz, Chroma Cens, MFCC (Mel-frequency cepstral coefficients).) and some other features. These features are explained in detail below.

**Chroma STFT:** Chroma STFT calculates the chromagram of a sound wave or power spectrogram. The chromagram separates the spectral energy of a sound into 12 different pitch classes (or chromatic notes).

**RMS:** It is a term that measures the amplitude or intensity of a sound signal.

**Spectral Centroid:** Spectral centroid is a measure used to characterize a spectrum. It shows where the center of mass of the spectrum.

**Spectral Bandwidth:** Spectral bandwidth is the widt--h of a spectral band. That is, it is the range of wavelengths or frequencies where the magnitude of all spectral components is important.

**Spectral Rolloff:** It is a property that represents the frequency at which the sound spectrum falls by a certain percentage.

**Zero Crossing Rate:** Zero crossing rate determines the moments when a signal reaches zero value.

**Harmony:** Harmonic components are components that generally carry the melodic and tonal characteristics of a sound1. These components generally have a smooth and continuous structure and usually form the melody of a piece of music.

**Percussiveness:** Percussive components are components that generally carry the rhythmic characteristics of a sound. These components generally have an irregular and rapidly changing structure and generally form the rhythm of a musical piece. It represents the percussive component. The term is commonly employed to depict sounds produced through striking or hitting, distinguished by brief, loud, and frequently rhythmic qualities.

**Tempogram:** A tempogram is a representation of an audio signal used to analyze rhythm and tempo changes.

**Tonnetz:** It refers to the tonal center features. I is used to analyze the tonal structure of a piece of music.

**Chroma CENS:** An energy normalized version of chromagram.

**MFCC (Mel-Frequency Cepstral Coefficients):** Mel is a representation of spectral features based on frequency scaling. MFCC is designed to represent sound more effectively and more closely to the human hearing system.

**Spectral Contrast:** Spectral contrast is a feature that evaluates the spectral characteristics of an audio signal. Divides each frame in the spectrogram into subbands. For each subband, the energy contrast is estimated by comparing the average energy in the upper quartile (peak energy) with the average energy in the lower quartile (valley energy). High contrast values generally indicate clear, narrow-band signals, while low contrast values generally indicate broad-band noise. This feature can be used to determine how wide or narrow the frequency content of an audio signal is.

**Tempogram:** A feature that characterizes the tempo changes and local rhythm of a music piece.

**MFCC Delta:** A representation of the temporal changes in Mel Frequency Cepstral Coefficients (MFCC) of a music piece.

**Spectral Flatness:** Spectral flatness is used to determine whether a sound resembles a pure tone or resembles noise. Usually measured in decibels, high spectral flatness values (approaching 1.0) indicate that the spectrum has similar power in all spectral bands. This refers to a sound similar to white noise, and the spectrum graph appears relatively flat and regular. On the other hand, low spectral flatness values (approaching 0.0 for a pure tone) indicate that the spectral power is concentrated in a relatively small number of bands.

While extracting all these features, the Librosa library, which is available in the Python programming language and is especially powerful in sound processing, is used. Librosa is an open source library used to analyze, feature extract and process audio files. This library facilitates audio analysis and feature extraction by examining audio signals in detail in temporal and frequency domains.

Mel-Frequency Cepstral Coefficients (MFCC) is a commonly used property to analyze and represent audio signals. Typically, the number of coefficients MFCC extracts is between 12 and 13. However, in this particular study, 20 coefficients were used in order to obtain a more comprehensive and detailed sound representation. This allows audio signals to be analyzed with more features represented.

Additionally, the delta coefficient of MFCC at five different levels was also calculated to capture changes over time. These delta coefficients were calculated respectively, starting from the first order (order=1) delta coefficients to the fifth order (order=5) delta coefficients. Delta coefficients are used to represent the dynamic properties of audio signals by measuring changes between successive frames of the MFCC. This allows audio analysis to be made more detailed and sensitive to changes over time.

In addition, the mean and variance of all these features were calculated in order to understand the characteristics of each piece of music more comprehensively and to base the training of the model on a more solid foundation. These statistical metrics will help the model learn various aspects of music more precisely by reflecting the overall trend and variability of each feature set. In this way, the generalization ability of the model will increase and it will be able to distinguish different types of music more effectively. The properties of one instance in the resulting dataset are as shown in Table 2.

*Table 2. Dataset After Feature Extraction*

| filename | length | chroma_stft_mean | chroma_stft_var | rms_mean | rms_var | spectral_centroid_mean |
|---|---|---|---|---|---|---|
| blues.00093.wav | 3 | 0.399003 | 0.09466 | 0.056844 | 0.00188 | 640.5312 |
| **spectral_centroid_var** | **spectral_bandwidth_mean** | **spectral_bandwidth_var** | **rolloff_mean** | **rolloff_var** | **zero_crossing_rate_mean** | **zero_crossing_rate_var** |
| 160926.5 | 1022.054 | 104654.7 | 1190.124 | 726941.7 | 0.022397 | 0.000142 |
| **harmony_mean** | **harmony_var** | **perceptr_mean** | **perceptr_var** | **spectral_contrast_mean** | **spectral_contrast_var** | **tonnetz_mean** |
| 1.71E-05 | 0.00436 | -8.9E-05 | 0.00022 | 20.60304 | 27.54738 | 0.015479 |
| **tonnetz_var** | **tempogram_mean** | **tempogram_var** | **chroma_cens_mean** | **chroma_cens_var** | **mfcc_delta1_mean** | **mfcc_delta1_var** |
| 0.047083 | 0.052103 | 0.008829 | 0.193934 | 0.045723 | -0.02248 | 7.224687 |
| **mfcc_delta2_mean** | **mfcc_delta2_var** | **mfcc_delta3_mean** | **mfcc_delta3_var** | **mfcc_delta4_mean** | **mfcc_delta4_var** | **mfcc_delta5_mean** |
| 0.00955 | 2.760068 | -0.00112 | 2.528859 | 0.003421 | 4.775765 | -0.0079 |
| **mfcc_delta5_var** | **spectral_flatness_mean** | **spectral_flatness_var** | **tempo** | **mfcc1_mean** | **mfcc2_mean** | **mfcc3_mean** |
| 16.42404 | 0.00068 | 2.12E-05 | 78.30256 | -350.515 | 2964.59 | 167.4123 |
| **mfcc4_mean** | **mfcc5_mean** | **mfcc6_mean** | **mfcc7_mean** | **mfcc8_mean** | **mfcc9_mean** | **mfcc10_mean** |
| 870.4708 | 19.82976 | 311.1636 | 12.77431 | 286.231 | 35.51379 | 214.5035 |
| **mfcc11_mean** | **mfcc12_mean** | **mfcc13_mean** | **mfcc14_mean** | **mfcc15_mean** | **mfcc16_mean** | **mfcc17_mean** |
| 21.37247 | 41.02737 | 4.857546 | 44.92178 | 11.54639 | 45.32292 | 6.412818 |
| **mfcc18_mean** | **mfcc19_mean** | **mfcc20_mean** | **mfcc1_var** | **mfcc2_var** | **mfcc3_var** | **mfcc4_var** |
| 35.71656 | -0.98271 | 71.35916 | 3.68169 | 28.32964 | 1.817667 | 79.40176 |
| **mfcc5_var** | **mfcc6_var** | **mfcc7_var** | **mfcc8_var** | **mfcc9_var** | **mfcc10_var** | **mfcc11_var** |
| -11.2129 | 37.12267 | -7.73921 | 33.16057 | 0.326726 | 34.5194 | -3.29946 |
| **mfcc12_var** | **mfcc13_var** | **mfcc14_var** | **mfcc15_var** | **mfcc16_var** | **mfcc17_var** | **mfcc18_var** |
| 51.82647 | -7.07854 | 32.02808 | 0.315695 | 44.21118 | 7.07478 | 56.45985 |
| **mfcc19_var** | **mfcc20_var** | **label** | | | | |
| 2.348529 | 45.10616 | blues | | | | |

## 3.3. Modeling

In this study, 5 different models were used for music genre classification: K-Nearest Neighbor (KNN), Random Forest, XGBoost, Support Vector Machine and Artificial Neural Networks (ANN). At the beginning of the analysis, the dataset is divided into 80% training and 20% testing. Then, normalization was performed on the dataset, which ensures that the data are brought to similar scales. In order to obtain more reliable and generalizable results, this study uses the cross validation method. The coefficient to be used for cross-validation was determined as 10. Additionally, when performing a grid search, the CV value was determined as 5.

Additionally, a rigorous hyperparameter tuning process was applied to optimize the parameters used in the training phase of each classification model. The main purpose of this comprehensive optimization was to increase the performance of the model (accuracy, precision, recall, f1-score). It is also aimed to prevent possible problems such as overfitting or underfitting.

Hyperparameter optimization included effective methods such as grid search and random search. Grid search aims to systematically explore a predetermined range of hyperparameters at specified intervals and tests all possible combinations. This approach is valuable for identifying the best hyperparameters within a large search space, but can be time-consuming due to detailed evaluation of many combinations.

On the other hand, random search evaluates randomly selected combinations of hyperparameters. This method can often be more effective because you can try more combinations, thus achieving faster results. This approach is particularly suitable for classification tasks and allows a more flexible and dynamic exploration of the hyperparameter space.

Using these hyperparameter optimization techniques, classification models were refined, achieving optimal performance and ensuring robustness and generalization across diverse datasets.

The metrics used to evaluate the performance of the models are as follows;

**Accuracy:** Accuracy refers to the ratio of correct predictions to the total number of samples. That is, it shows the correct classification rate of the model.

**Precision:** Precision refers to the rate at which samples predicted as positive are actually positive. High precision indicates how reliable the model's positive predictions are.

**Recall:** Recall refers to how many samples it correctly detects that are truly positive. A high recall indicates that the model does not tend to miss positive examples.

**F1-Score:** F1-score is the harmonic mean of precision and recall. This metric strikes a balance between precision and recall and evaluates the overall performance of a model. F1-score is especially useful in datasets with unbalanced classes.

### *3.3.1. K-Nearest Neighbor (KNN)*

The K-NN (k-Nearest Neighbors) Classifier is a simple and effective machine learning algorithm used in a classification problem. The basic idea is to determine the class of a data point based on the class of its k-nearest neighbors around it.

The parameters used during the hyperparameter tuning process are shown with best selected parameters in Table 3.

*Table 3. KNN Hyperparameters*

| Parameter | Values | Best Values |
|---|---|---|
| n_neighbors | {1, 3, 5, 7} | 1 |
| p | {1, 2} | 1 |
| metric | {minkowski} | minkowski |
| algorithm | {auto, ball_tree, kd_tree, brute} | auto |

**n_neighbors:** This parameter determines how many neighbors of each data point will be taken into account during classification.

**p:** Minkowski is a parameter that specifies the distance measure. The Minkowski distance metric is a metric that generalizes Euclidean and Manhattan distances. If **p=1**, Manhattan distance is used, if **p=2**, Euclidean distance is used.

**metric:** It determines the distance metric to be used to evaluate neighborhood relationships. For example, it can take values such as "minkowski", "euclidean", "manhattan".

**algorithm:** Determines the algorithm to be used for neighborhood calculation. It has options such as "auto", "ball_tree", "kd_tree", "brute". The "auto" option automatically selects the most appropriate algorithm based on data size and structures.

The model, determined with the best parameters obtained after grid search, was tested with the cross-validation method for a more robust evaluation. This step was carried out to evaluate the overall performance of the model more reliably. In this way, it can be better understood whether the model encounters problems such as overfitting or underfitting. The results obtained are shown in Table 4.

*Table 4. KNN Results with Cross Validation*

| Fold | Validation Accuracy |
|---|---|
| 1 | 0.9450 |
| 2 | 0.9338 |
| 3 | 0.9550 |
| 4 | 0.9349 |
| 5 | 0.9512 |
| 6 | 0.9312 |
| 7 | 0.9399 |
| 8 | 0.9412 |
| 9 | 0.9374 |
| 10 | 0.9262 |
| | |
| **Average** | 0.9396 |

Table 5 includes the classification report. The model can successfully distinguish different types of music and its overall accuracy rate is 94%, representing a high level of success. The model's high accuracy, recall and F1 scores indicate that it exhibits a reliable and stable performance in the classification task. Achieving high accuracy, especially in classical, metal and hip-hop genres, emphasizes the model's ability to successfully identify these genres.

*Table 5. KNN Classification Report*

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| **blues** | 0.95 | 0.94 | 0.94 | 208 |
| **classical** | 0.98 | 0.98 | 0.98 | 203 |
| **country** | 0.92 | 0.92 | 0.92 | 188 |
| **disco** | 0.93 | 0.93 | 0.93 | 200 |
| **hiphop** | 0.95 | 0.96 | 0.96 | 215 |
| **jazz** | 0.95 | 0.95 | 0.95 | 191 |
| **metal** | 0.98 | 0.97 | 0.97 | 204 |
| **pop** | 0.95 | 0.94 | 0.95 | 181 |
| **reggae** | 0.95 | 0.94 | 0.94 | 210 |
| **rock** | 0.91 | 0.91 | 0.91 | 198 |
| | | | | |
| **accuracy** | | | 0.94 | 1998 |

Finally, confusion matrix can be seen in Table 6.

*Table 6. KNN Confusion Matrix*

| | **blues** | **classical** | **country** | **disco** | **hiphop** | **jazz** | **metal** | **pop** | **reggae** | **rock** |
|---|---|---|---|---|---|---|---|---|---|---|
| **blues** | 196 | 0 | 2 | 0 | 0 | 4 | 0 | 0 | 1 | 5 |
| **classical** | 0 | 199 | 1 | 0 | 0 | 3 | 0 | 0 | 0 | 0 |
| **country** | 1 | 1 | 173 | 5 | 0 | 2 | 0 | 0 | 2 | 4 |
| **disco** | 1 | 2 | 1 | 186 | 4 | 0 | 0 | 0 | 3 | 3 |
| **hiphop** | 0 | 0 | 0 | 0 | 206 | 0 | 1 | 6 | 2 | 0 |
| **jazz** | 4 | 1 | 3 | 0 | 0 | 182 | 0 | 0 | 0 | 1 |
| **metal** | 2 | 0 | 0 | 0 | 1 | 0 | 197 | 0 | 0 | 4 |
| **pop** | 0 | 1 | 1 | 3 | 2 | 0 | 0 | 171 | 2 | 1 |
| **reggae** | 2 | 0 | 2 | 2 | 2 | 1 | 1 | 2 | 197 | 1 |
| **rock** | 1 | 0 | 5 | 5 | 1 | 0 | 3 | 1 | 1 | 181 |

### *3.3.2. Random Forest*

Random Forest is a classification and regression algorithm widely used in machine learning. This algorithm is an ensemble learning method that combines many decision trees. Each decision tree is trained on a randomly selected subset of the data set. This provides diversity across trees and increases the stability of the overall model. Random Forest is used to combine the predictions of these trees to obtain a robust model with high generalization capabilities. Additionally, each tree can identify important features in the dataset by focusing on different features. Therefore, it is an effective method for feature selection. Random Forest is resilient to overfitting problems and can be successfully applied to multi-class classification problems with high-dimensional datasets.

The parameters used during the hyperparameter tuning process are shown with best selected parameters in Table 7.

*Table 7. Random Forest Hyperparameters*

| **Parameter** | **Values** | **Best Values** |
|---|---|---|
| criterion | {gini, entropy} | entropy |
| max_depth | {10, 20} | 20 |
| max_samples_split | {2, 5} | 2 |
| max_samples | {0.8, 1.0} | 1.0 |

**criterion:** It is the function used to measure the quality of a branch. Supported criteria are “gini” for Gini purity, “log_loss” and “entropy” for Shannon information gain, and “poisson” for reduction of Poisson bias.

**max_depth:** It is the maximum depth of the tree. If None, nodes are expanded until all leaves are pure or all leaves contain fewer than min_samples_split samples.

**min_samples_split:** It is the minimum number of instances required to split an internal node. If int, it treats min_samples_split as the minimum number. If float, min_samples_split is a fraction and ceil(min_samples_split * n_samples) is the minimum number of samples for each split.

**max_samples:** Controls the number of samples to use for each tree. If int, max_samples is considered the number of samples. If float, max_samples is a fraction and ceil(max_samples * n_samples) is the number of samples for each tree. If None, a subset of the entire dataset is used if bootstrap=True, the entire dataset is used if bootstrap=False.

The model, constructed with the best parameters determined by grid search, was tested with the cross-validation method on validation data for a more robust evaluation. The results obtained are presented in Table 8.

*Table 8. Random Forest Results with Cross Validation*

| Fold | Validation Accuracy |
|---|---|
| 1 | 0.8813 |
| 2 | 0.8588 |
| 3 | 0.8661 |
| 4 | 0.8673 |
| 5 | 0.8899 |
| 6 | 0.8824 |
| 7 | 0.8686 |
| 8 | 0.8736 |
| 9 | 0.8673 |
| 10 | 0.8761 |
| | |
| **Average** | 0.8731 |

Table 9 includes the classification report. Overall, the performance of the model seems quite good. However, low recall and F1 scores in the "rock" category are notable, indicating further improvements are needed on this category.

*Table 9. Random Forest Classification Report*

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| **blues** | 0.83 | 0.89 | 0.86 | 208 |
| **classical** | 0.95 | 0.99 | 0.97 | 203 |
| **country** | 0.79 | 0.88 | 0.84 | 188 |
| **disco** | 0.83 | 0.83 | 0.83 | 200 |
| **hiphop** | 0.94 | 0.86 | 0.90 | 215 |
| **jazz** | 0.90 | 0.91 | 0.91 | 191 |
| **metal** | 0.91 | 0.94 | 0.92 | 204 |
| **pop** | 0.89 | 0.91 | 0.90 | 181 |
| **reggae** | 0.85 | 0.86 | 0.85 | 210 |
| **rock** | 0.93 | 0.73 | 0.82 | 198 |
| | | | | |
| **accuracy** | | | 0.88 | 1998 |

Finally, confusion matrix can be seen in Table 10.

*Table 10. Random Forest Confusion Matrix*

| | **blues** | **classical** | **country** | **disco** | **hiphop** | **jazz** | **metal** | **pop** | **reggae** | **rock** |
|---|---|---|---|---|---|---|---|---|---|---|
| **blues** | 186 | 1 | 7 | 3 | 0 | 5 | 3 | 0 | 3 | 0 |
| **classical** | 0 | 201 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| **country** | 10 | 0 | 166 | 4 | 0 | 5 | 0 | 0 | 3 | 0 |
| **disco** | 7 | 1 | 2 | 166 | 7 | 2 | 6 | 2 | 6 | 1 |
| **hiphop** | 2 | 0 | 2 | 2 | 185 | 0 | 5 | 11 | 8 | 0 |
| **jazz** | 6 | 5 | 5 | 1 | 0 | 174 | 0 | 0 | 0 | 0 |
| **metal** | 1 | 1 | 1 | 2 | 0 | 0 | 191 | 0 | 2 | 6 |
| **pop** | 0 | 1 | 6 | 4 | 1 | 2 | 0 | 164 | 3 | 0 |
| **reggae** | 4 | 0 | 5 | 3 | 4 | 3 | 1 | 7 | 180 | 3 |
| **rock** | 8 | 1 | 14 | 15 | 0 | 2 | 5 | 1 | 8 | 144 |

### *3.3.3. Support Vector Machine (SVM)*

Support Vector Machines (SVM) is a powerful machine learning algorithm used especially in classification and regression problems. SVM performs classification by creating a hyperplane that separates data points. Its main purpose is to separate data points with maximum margin between classes. One of the distinctive features of SVM is that it uses only a subset of data points to find the optimal location of the bounding hyperplane. This subset is called support vectors and plays an important role in the classification process. The ability of SVM to work effectively in multiple dimensions makes it especially preferred in applications dealing with high-dimensional data sets.

The parameters used during the hyperparameter tuning process are shown with best selected parameters in Table 11.

*Table 11. Support Vector Machine Hyperparameters*

| **Parameter** | **Values** | **Best Values** |
|---|---|---|
| C | {0.1, 1, 10} | 10 |
| kernel | {'linear', 'rbf', 'poly'} | 'rbf' |
| gamma | {'scale', 'auto'} | 'scale' |
| degree | {2, 3} | 2 |

**c (cost parameter):** C is a hyperparameter that controls the regularization of the SVM. This parameter adjusts the balance between bias and variance of the model. A bigger C fits the training data better and creates a more complex model, but can also increase the risk of overfitting. A smaller C increases the amount of trimming and results in a simpler pattern, but may increase the risk of underfitting.

**kernel:** Kernel is a mathematical operation used by Support Vector Machines (SVM) and aims to move low-dimensional, non-linearly separable data sets into high-dimensional space and make them linearly separable there. SVM uses various kernel functions to process datasets, especially those with non-linear boundaries. While the linear kernel processes data sets that can be separated linearly, the RBF (Radial Basis Function) kernel is often preferred in data sets that cannot be separated linearly and controls the domain by using it with the 'gamma' parameter. The polynomial kernel is used to solve data sets that cannot be linearly separated by low-degree polynomials, and the 'degree' parameter determines the degree of the polynomial.

**gamma:** The free parameter of SVM's kernel functions. RBF is used in poly and sigmoid kernels. A small gamma value results in a high variance and a larger domain. A large gamma value results in a low variance and a narrower domain. Therefore, it is important to choose the gamma value carefully.

**degree:** The degree of the polynomial kernel. It is only effective when poly core is used. This value determines the degree of the polynomial kernel. For example, if degree=2, a quadratic polynomial is used.

The model, constructed with the best parameters determined by grid search, was tested with the cross-validation method for a more robust evaluation. The results obtained are presented in Table 12.

*Table 12. Support Vector Machine Results with Cross Validation*

| **Fold** | **Validation Accuracy** |
|---|---|
| 1 | 0.9263 |
| 2 | 0.9400 |
| 3 | 0.9299 |
| 4 | 0.9036 |
| 5 | 0.9362 |
| 6 | 0.9287 |
| 7 | 0.9212 |
| 8 | 0.9337 |
| 9 | 0.9262 |
| 10 | 0.9237 |
| | |
| **Average** | 0.9269 |

Table 13 contains the Support Vector Machine (SVM) classification report. The model can successfully distinguish different types of music, with an overall accuracy rate of 94%, which represents a high level of success. The model's high accuracy, recall, and F1 scores indicate that it exhibits reliable and stable performance in the classification task. Achieving high accuracy, especially in classical, metal and hip-hop genres, highlights the model's ability to successfully identify these genres.

*Table 13. Support Vector Machine Classification Report*

| | **precision** | **recall** | **f1-score** | **support** |
|---|---|---|---|---|
| **blues** | 0.94 | 0.93 | 0.93 | 208 |
| **classical** | 0.95 | 0.99 | 0.97 | 203 |
| **country** | 0.92 | 0.93 | 0.92 | 188 |
| **disco** | 0.91 | 0.92 | 0.91 | 200 |
| **hiphop** | 0.96 | 0.93 | 0.95 | 215 |
| **jazz** | 0.96 | 0.97 | 0.97 | 191 |
| **metal** | 0.96 | 0.97 | 0.96 | 204 |
| **pop** | 0.91 | 0.94 | 0.93 | 181 |
| **reggae** | 0.97 | 0.95 | 0.96 | 210 |
| **rock** | 0.90 | 0.86 | 0.88 | 198 |
| | | | | |
| **accuracy** | | | 0.94 | 1998 |

Finally, the confusion matrix can be seen in Table 14.

*Table 14. Support Vector Machine Confusion Matrix*

| | **blues** | **classical** | **country** | **disco** | **hiphop** | **jazz** | **metal** | **pop** | **reggae** | **rock** |
|---|---|---|---|---|---|---|---|---|---|---|
| **blues** | 194 | 0 | 4 | 1 | 1 | 2 | 1 | 0 | 0 | 5 |
| **classical** | 0 | 200 | 1 | 0 | 0 | 2 | 0 | 0 | 0 | 0 |
| **country** | 6 | 0 | 174 | 1 | 0 | 1 | 0 | 1 | 1 | 4 |
| **disco** | 1 | 3 | 1 | 183 | 1 | 0 | 3 | 2 | 3 | 3 |
| **hiphop** | 0 | 1 | 1 | 4 | 201 | 0 | 1 | 7 | 0 | 0 |
| **jazz** | 2 | 3 | 0 | 0 | 0 | 186 | 0 | 0 | 0 | 0 |
| **metal** | 0 | 1 | 1 | 0 | 1 | 0 | 197 | 0 | 0 | 4 |
| **pop** | 0 | 2 | 2 | 2 | 1 | 1 | 0 | 171 | 1 | 1 |
| **reggae** | 1 | 0 | 0 | 2 | 2 | 0 | 1 | 4 | 199 | 1 |
| **rock** | 3 | 0 | 6 | 9 | 3 | 1 | 2 | 2 | 2 | 170 |

### *3.3.4. XGBoost*

XGBoost is a learning algorithm that has made a strong impact, especially in the field of machine learning. Working on the basis of gradient boosting, this algorithm creates a powerful learner by combining multiple decision trees. During the addition of trees, it continuously improves the performance of the model by focusing on correcting the errors of the previous trees. It prevents overlearning with regularization techniques and exhibits high performance on large datasets thanks to fast, effective calculation methods. It can be adapted to various tasks with its flexible objective functions and loss functions. XGBoost is often the tool of choice to achieve successful results in the competitive field of data science and industry applications.

The parameters used during the hyperparameter tuning process are shown with best selected parameters in Table 15.

*Table 15. XGBoost Hyperparameters*

| **Parameter** | **Values** | **Best Values** |
|---|---|---|
| learning_rate | {0.01, 0.1, 0.2} | 0.2 |
| n_estimators | {50, 100, 150} | 150 |
| max_depth | {3, 4, 5} | 5 |
| min_child_weight | {1, 3, 5} | 1 |
| subsample | {0.8, 0.9, 1.0} | 0.9 |
| colsample_bytree | {0.8, 0.9, 1.0} | 0.9 |
| gamma | {0, 1, 5} | 0 |

**learning_rate:** It is the parameter that determines the extent of the contribution of each tree. A lower learning rate can contribute to making the model more reliable, but it also requires more trees.

**n_estimators:** It determines the total number of trees to be used. More trees can increase the complexity of the model, but can lead to overlearning.

**max_depth**: Determines the maximum depth of a decision tree. Larger values can cause the model to become more complex, but can also lead to overlearning.

**min_child_weight:** Determines the minimum number of eyes under a node. Lower values may cause the model to learn more specific patterns, but may be more susceptible to overlearning.

**subsample:** Determines the random sampling rate of training data for a tree. This allows each tree to be trained on a different subset.

**colsample_bytree:** Determines the proportion of a subset of columns for each tree. This allows each tree to be trained on a different subset of columns.

**gamma:** It is a parameter that controls whether it is appropriate to split a node. Higher gamma values encourage fewer splits.

The model, constructed with the best parameters determined by grid search, was tested with the cross-validation method for a more robust evaluation. The results obtained are presented in Table 16.

*Table 16. XGBoost Results with Cross Validation*

| **Fold** | **Validation Accuracy** |
|---|---|
| 1 | 0.9225 |
| 2 | 0.9164 |
| 3 | 0.9049 |
| 4 | 0.9074 |
| 5 | 0.9299 |
| 6 | 0.9263 |
| 7 | 0.9237 |
| 8 | 0.9174 |
| 9 | 0.9237 |
| 10 | 0.9049 |
| | |
| **Average** | 0.9177 |

Table 17 includes the classification report. The XGBoost model performs very well as it generally has high precision, recall and F1 scores. The high performance of the "classical" category is notable, and the model demonstrates the ability to successfully discriminate various types of music along with the overall accuracy rate.

*Table 17. XGBoost Classification Report*

| | **precision** | **recall** | **f1-score** | **support** |
|---|---|---|---|---|
| **blues** | 0.93 | 0.91 | 0.92 | 208 |
| **classical** | 0.98 | 0.99 | 0.98 | 203 |
| **country** | 0.87 | 0.94 | 0.90 | 188 |
| **disco** | 0.90 | 0.88 | 0.89 | 200 |
| **hiphop** | 0.96 | 0.95 | 0.96 | 215 |
| **jazz** | 0.92 | 0.94 | 0.93 | 191 |
| **metal** | 0.97 | 0.95 | 0.96 | 204 |
| **pop** | 0.90 | 0.92 | 0.91 | 181 |
| **reggae** | 0.94 | 0.91 | 0.92 | 210 |
| **rock** | 0.87 | 0.86 | 0.87 | 198 |
| | | | | |
| **accuracy** | | | 0.92 | 1998 |

Finally, confusion matrix can be seen in Table 18.

*Table 18. XGBoost Confusion Matrix*

| | **blues** | **classical** | **country** | **disco** | **hiphop** | **jazz** | **metal** | **pop** | **reggae** | **rock** |
|---|---|---|---|---|---|---|---|---|---|---|
| **blues** | 189 | 1 | 4 | 5 | 1 | 4 | 2 | 0 | 0 | 2 |
| **classical** | 0 | 200 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 1 |
| **country** | 1 | 0 | 176 | 4 | 0 | 3 | 0 | 1 | 1 | 2 |
| **disco** | 5 | 0 | 3 | 175 | 1 | 1 | 2 | 2 | 5 | 6 |
| **hiphop** | 0 | 0 | 1 | 1 | 205 | 0 | 0 | 6 | 1 | 1 |
| **jazz** | 5 | 2 | 3 | 0 | 0 | 180 | 0 | 0 | 1 | 0 |
| **metal** | 0 | 1 | 2 | 0 | 1 | 0 | 194 | 0 | 1 | 5 |
| **pop** | 0 | 0 | 5 | 0 | 2 | 2 | 0 | 167 | 2 | 3 |
| **reggae** | 0 | 0 | 2 | 4 | 2 | 1 | 0 | 5 | 191 | 5 |
| **rock** | 3 | 0 | 7 | 6 | 2 | 3 | 2 | 4 | 1 | 170 |

### *3.3.5. Artificial Neural Networks (ANN)*

Artificial Neural Networks (ANNs) are artificial intelligence models that simulate information processing capabilities by mimicking biological nervous systems. Artificial Neural Networks have a structure consisting of layers of nodes and weights connecting these nodes. These networks generally consist of input, hidden and output layers. During the training process, backpropagation algorithm is used to optimize the performance of the model.

The parameters used during the hyperparameter tuning process are shown with best selected parameters in Table 19.

*Table 19. ANN Hyperparameters*

| **Parameter** | **Values** | **Best Values** |
|---|---|---|
| units1 | min_value=1024<br>max_value=2048<br>step=128 | 1792 |
| units2 | min_value=1024<br>max_value=2048<br>step=64 | 1920 |
| units3 | min_value=1024<br>max_value=2048<br>step=32 | 1504 |
| dropout_rate | min_value=0.5<br>max_value=0.7<br>step=0.1 | 0.5 |
| learning_rate | 0.001 | 0.001 |

**units1:** Value expressing the number of neurons in the first hidden layer.

**units2:** : Value expressing the number of neurons in the second hidden layer.

**units3:** The value expressing the number of neurons in the third hidden layer.

**dropout:** It is a technique used to combat overfitting. Dropout tries to increase the generalization ability of the network by disabling randomly selected neurons during training (dropout). This can prevent the model from overlearning on specific examples, helping it arrive at a more general model. The dropout_rate parameter determines the rate of neurons to be disabled in a dropout layer. This ratio takes a value between 0 and 1. For example, using dropout_rate=0.2, 20 percent of randomly selected neurons are disabled in each training iteration. Dropout can improve overall model performance by making the network more flexible and reducing dependencies between neurons. However, choosing this ratio too high may limit the model's ability to learn. Therefore, an appropriate dropout rate is usually found through an experimental process.

**learning_rate:** It is an important hyperparameter used in the training process of artificial neural networks. This parameter controls the amount the weights and biases are updated. That is, it determines how much the weights and biases will be changed to minimize the loss of the model at each iteration. When choosing a learning rate, there is a balance to consider. If the learning rate is too small, the model learns very slowly and the training process may take a long time. On the other hand, if the learning rate is too large, the model may fluctuate excessively and learn erratically. Therefore, choosing an appropriate learning rate is important to ensure that the model learns quickly and stably.

The model, constructed with the best parameters determined by grid search, was tested with the cross-validation method for a more robust evaluation. The results obtained are presented in Table 20.

*Table 20. ANN Results with Cross Validation*

| **Fold** | **Validation Accuracy** |
|---|---|
| 1 | 0.9187 |
| 2 | 0.8988 |
| 3 | 0.9149 |
| 4 | 0.9325 |
| 5 | 0.9186 |
| 6 | 0.9324 |
| 7 | 0.9274 |
| 8 | 0.9086 |
| 9 | 0.9274 |
| 10 | 0.9199 |
| | |
| **Average** | 0.9199 |

Table 21 includes the classification report. The model's performance is generally quite good. In particular, the precision, recall and f1-score values are high. This indicates that the model can successfully distinguish between different musical genres. The high performance of the model in the "classical" category is noteworthy. Together with the overall accuracy, the model shows that it can successfully identify various music genres.

*Table 21. ANN Classification Report*

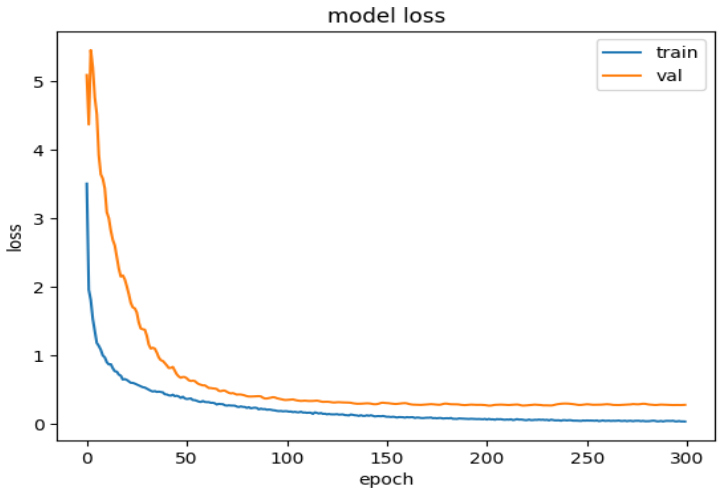
| | precision | recall | f1-score | support |
|---|---|---|---|---|
| **blues** | 0.96 | 0.93 | 0.95 | 208 |
| **classical** | 0.96 | 0.99 | 0.97 | 203 |
| **country** | 0.90 | 0.97 | 0.93 | 188 |
| **disco** | 0.92 | 0.92 | 0.92 | 200 |
| **hiphop** | 0.95 | 0.93 | 0.94 | 215 |
| **jazz** | 0.96 | 0.95 | 0.96 | 191 |
| **metal** | 0.97 | 0.96 | 0.97 | 204 |
| **pop** | 0.90 | 0.93 | 0.92 | 181 |
| **reggae** | 0.95 | 0.91 | 0.93 | 210 |
| **rock** | 0.92 | 0.90 | 0.91 | 198 |
| | | | | |
| **accuracy** | | | 0.94 | 1998 |

Finally, the confusion matrix can be seen in Table 22. In Figure 2, we can see the training and validation loss in each iteration.

*Table 22. ANN Confusion Matrix*

| | blues | classical | country | disco | hiphop | jazz | metal | pop | reggae | rock |
|---|---|---|---|---|---|---|---|---|---|---|
| **blues** | 194 | 0 | 6 | 1 | 0 | 3 | 1 | 0 | 0 | 3 |
| **classical** | 0 | 201 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| **country** | 1 | 0 | 183 | 0 | 0 | 2 | 0 | 0 | 1 | 1 |
| **disco** | 0 | 2 | 0 | 184 | 4 | 0 | 1 | 0 | 5 | 4 |
| **hiphop** | 1 | 0 | 2 | 1 | 200 | 0 | 2 | 8 | 1 | 0 |
| **jazz** | 3 | 5 | 1 | 0 | 0 | 181 | 0 | 0 | 1 | 0 |
| **metal** | 0 | 1 | 2 | 1 | 1 | 0 | 196 | 0 | 0 | 3 |
| **pop** | 0 | 0 | 5 | 2 | 1 | 1 | 0 | 169 | 2 | 1 |
| **reggae** | 1 | 0 | 0 | 5 | 2 | 0 | 1 | 6 | 192 | 3 |
| **rock** | 2 | 1 | 5 | 5 | 2 | 0 | 1 | 4 | 0 | 178 |

*Figure 2. Training/Validation Loss Graph*



# 4. User Interface

*Figure 3. User Interface*


This user interface shown in Figure 3 represents the front end of a Flask web application developed to classify music genres. Flask is a Python-based web application framework and forms the basis of the application in this project. Flask is known for its fast development and easy-to-understand structure. The application allows users to upload music files, process them and view the results using the powerful and flexible structures offered by Flask.

The interface is stylized using the Bootstrap CSS framework to provide a modern and responsive design. The background image and color palette enhance the user experience by giving the app an aesthetic look.

Users can upload music files using the file selection button or by dragging and dropping the file for a more interactive experience. During the file upload process, the name of the uploaded file is visually presented to the user, making it possible to track the user's upload status. The music genre prediction process, which is initiated by clicking the "Find the genre" button, involves extracting the features of the uploaded music file and then making the prediction by the deployed model. Librosa library was used to extract the features of music files. Librosa is a library specifically designed for audio and music processing.

Behind the scenes, extracting the features of the music file is done using parallelization. In particular, simultaneous processing of different segments of the music is achieved using ThreadPoolExecutor. In this way, the process is accelerated, and a faster response is provided to the user.

Additionally, a simple audio player is included in the interface, allowing users to listen to the uploaded music. This feature gives users the chance to evaluate the content of the file they upload. While the prediction process is in progress, a loading indicator is displayed to indicate to the user that the process is in progress.

This user interface attracts users by offering an aesthetically attractive appearance and also effectively performs music genre classification with the technologies used behind it.

# 5. Results, Conclusion and Recommendations

| | Validation Average | F1-Score Average | Test Accuracy |
|---|---|---|---|
| KNN | 0.9396 | 0.94 | 0.945 |
| Random Forest | 0.8731 | 0.88 | 0.879 |
| SVM | 0.9269 | 0.94 | 0.938 |
| XGBoost | 0.9177 | 0.92 | 0.924 |
| ANN | 0.9199 | 0.94 | 0.944 |

This comprehensive music genre classification analysis was conducted using various machine learning models – K-Nearest Neighbors (KNN), Random Forest, Support Vector Machine (SVM), XGBoost and Artificial Neural Networks (ANN) – and detailed the performance and fit of each model. has been examined in some way. Evaluation included K-fold cross-validation, hyperparameter tuning, and optimization processes.

The KNN model performed best, achieving a remarkable 94.5% test success. This high success rate highlights the suitability of KNN for music genre classification problems. The ability to take into account the proximity of data points in feature space has been effective in distinguishing between various types of music. The ANN model, which came right behind, took second place with a test success of 94.4%. The success of ANN highlights its ability to capture complex patterns and relationships within data. The complex neural network structure consisting of input, hidden and output layers allowed the model to understand the complexities in the data. The SVM and XGBoost models performed solidly, achieving test successes of 93.8% and 92.4%, respectively. SVM's ability to deal with high-dimensional datasets and XGBoost's gradient boosting approach proved valuable in distinguishing between different musical genres. It is important to note that although Random Forest performs poorly compared to other models, these results may be due to insufficient model parameters. Further optimization can be done to increase the classification capabilities of the model.

All models achieved approximately 94% success, with balanced success observed in the F1-Score Average metric. This trade-off suggests that the models balance precision and recall across different types of music.

In conclusion, this study provides valuable information about the effectiveness of various machine learning models for music genre classification. The best performing KNN and ANN models reveal the importance of convergence-based and deep learning techniques to address this classification problem.

Future research could focus on further optimizing Random Forest model parameters and potentially improve its performance. Additionally, it is thought that more advanced deep learning architectures and feature engineering techniques can contribute to further improving the models.

## References


Pelchat, N., & Gelowitz, C. M. (2020). Neural network music genre classification. *Canadian Journal of Electrical and Computer Engineering*, *43*(3), 170-173.

Tzanetakis, G., & Cook, P. (2002). Musical genre classification of audio signals. *IEEE Transactions on speech and audio processing*, *10*(5), 293-302.

Xu, C., Maddage, N. C., Shao, X., Cao, F., & Tian, Q. (2003, April). Musical genre classification using support vector machines. In *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03).* (Vol. 5, pp. V-429). IEEE.

Haggblade, M., Hong, Y., & Kao, K. (2011). Music genre classification. *Department of Computer Science, Stanford University*, *131*, 132.

Bhatia, J. K., Singh, R. D., & Kumar, S. (2021, October). Music genre classification. In *2021 5th International Conference on Information Systems and Computer Networks (ISCON)* (pp. 1-4). IEEE.

HIZLISOY, S., & Tüfekci, Z. (2021). Derin Öğrenme İle Türkçe Müziklerden Müzik Türü Sınıflandırması. *Avrupa Bilim ve Teknoloji Dergisi*, (24), 176-183.

Özbalcı, M. C. (2022). *Olasılıksal modeller ile Türkçe müzik türlerinin sınıflandırılması* (Master's thesis).