Musical Genre Classification.

Aleix Francia Albert, Natalia Blaszczyk

Abstract– As the volume of digital audio content continues to grow, there is an increasing demand for robust and efficient techniques to analyze and organize sound. One of the most widely adopted approaches in the field of audio signal processing is the use of Mel Frequency Cepstral Coefficients (MFCCs), which provide a compact representation of the spectral properties of audio in a form that approximates human auditory perception. This paper presents an overview of the MFCC extraction process and explores its applications in speech recognition, music classification, and audio recommendation systems. In addition, we discuss several complementary audio features—including RMS energy, spectral contrast, chroma features, tempo, and rhythm patterns—that can enhance the effectiveness of music information retrieval systems.

Keywords–MFCC, audio signal processing, music information retrieval, speech recognition, spectral features, audio classification, chroma, tempo, rhythm.

1 Introduction

During the course of this project, the team will develop three distinct models aimed at classifying music tracks according to their respective genres. The classification process will primarily leverage *Mel-Frequency Cepstral Coefficients* (MFCCs), a widely adopted set of features in the field of audio analysis, recognized for their effectiveness in capturing perceptually relevant aspects of sound based on the characteristics of the human auditory system.

While MFCCs will serve as the principal descriptor, the analysis may be complemented by additional features, including, but not limited to, root mean square (RMS) energy, spectral contrast, chroma features, tempo, and rhythm patterns. The extraction of these features will be performed utilizing established audio processing libraries such as *Librosa*, which provide robust and efficient tools for audio signal analysis.

By converting raw audio signals into structured and informative feature representations, the project seeks to enable accurate and scalable genre classification. Furthermore, these methodologies have broader applicability in tasks such as content-based retrieval, playlist generation, and music recommendation systems. The remainder of this document presents a detailed overview of the computation and interpretation of MFCCs, an outline of the classification models developed, and a discussion of additional audio descriptors that contribute to a comprehensive analysis of music signals.

• *Contact Email:* afranciaa2501@gmail.com, nataliablszczk@gmail.com
• *Supervised by:* Markus Mayer

2 Dataset

The dataset chosen for this project is **GTZAN Dataset** first presented in a paper by George Tzanetakis and Perry Cook [1]. The dataset contains 1000 audio samples of 30 seconds length. The audio samples are divided into 10 genres of 100 tracks each: blues, classical, country, disco, hip-hop, jazz, pop, metal, rock, reggae.

One file from the jazz genre was found to be corrupted, but it is of no significant difference given the size of the whole dataset, as well as the comparable number of tracks left in the jazz genre and the rest of the genres.

3 Feature Extraction

3.1 Mel-Frequency Cepstral Coefficients (MFCC)

MFCCs are a widely used feature representation of audio signals, especially in speech and sound recognition tasks. They capture the short-term power spectrum of a sound, modeled on human auditory perception. Specifically, MFCCs provide a compact representation of the spectral envelope of an audio signal.

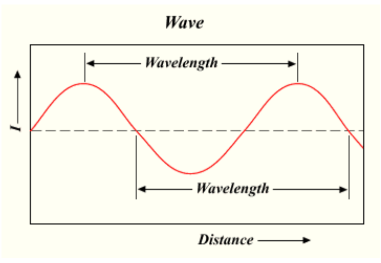

Fig. 1: Sine wave[2].

What do MFCCs represent? MFCCs encode the rate of change in spectral bands. A positive cepstral coefficient suggests that most spectral energy lies in the low-frequency regions, while a negative value indicates concentration in high frequencies. These coefficients are typically averaged across time frames and used as input features for machine learning models.

MFCC Feature Extraction Process: The MFCCs are extracted through the following steps:

1. **Pre-emphasis:** Apply a high-pass filter to amplify higher frequencies and balance the spectrum, compensating for the natural decline of energy at higher frequencies in speech.

2. **Framing:** Split the signal into overlapping short-time frames (e.g., 25 ms with 10 ms overlap). For a 16 kHz signal, this results in 400 samples per frame and a hop length of 160 samples.

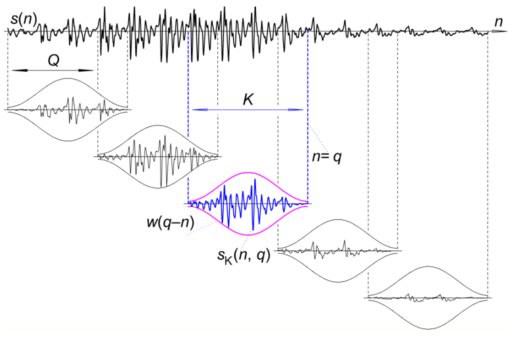

Fig. 2: Framing process[3].

3. **Fast Fourier Transform (FFT):** Convert each frame to the frequency domain using FFT (typically 256 or 512 points), producing a power spectrum that describes energy distribution over frequencies.

   Periodogram: An estimate of the spectral density of a signal.

4. **Mel Filterbank:** Apply triangular filters (20–40) spaced according to the Mel scale to mimic human hearing sensitivity. Each filterbank output represents the energy in a Mel-scaled frequency band.

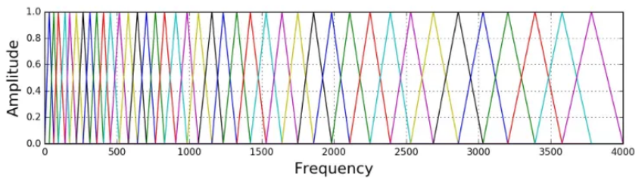

Fig. 3: Mel-scale filterbank[4].

5. **Logarithm:** Take the logarithm of each filterbank energy, simulating the non-linear perception of loudness.

6. **Discrete Cosine Transform (DCT):** Apply DCT to the log filterbank energies to obtain the MFCCs. This step reduces correlation and dimensionality, typically keeping only the first 12–13 coefficients.

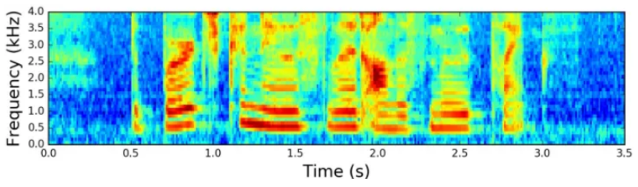

Fig. 4: Spectrogram after applying filterbanks[4].

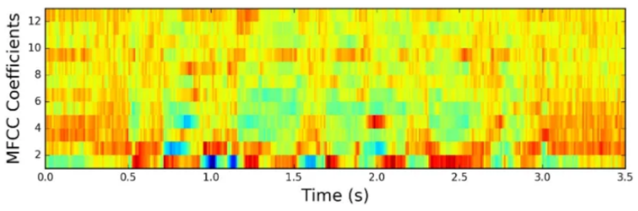

Fig. 5: Final MFCC representation[4].

3.2 Spectral Contrast

Spectral contrast measures the differenece in energy between the peaks(haighest values) and valleys(lowest values) within specific frequency bands. The primary focus is on the differenece or contrast between the most intense and least intense frequencies component within each band.

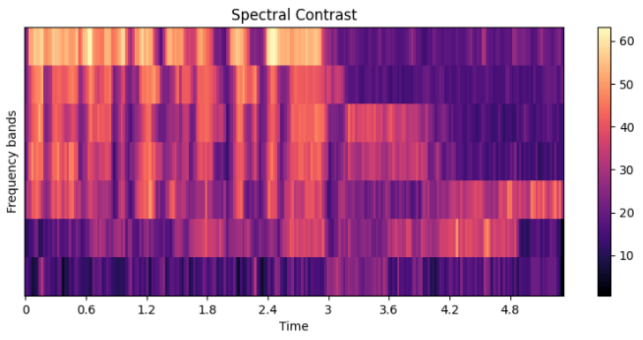

Fig. 6: SpectralContrast[5].

The frequency spectrum is divided into bands, typically on a logarithmic scale to match the human perception of frequency. For each band, the spectral contrast is calculated by comparing the peak and valley energy levels.

By focusing on the peaks and valleys, spectral contrast highlights the most prominent and the least noticeable parts of the signal, witch can indicate dynamic changes in the spectrum.

Also, intermediates are ignored in Spectral Contrast. It can be helpful in applications like music genre classification or audio segmentation, where the extremes play a key role in identifying patterns or characteristics.

1. **Frame the signal:** Split the signal, commonly used window size s for calculation Spectral Contrast, range from 20 milliseconds to 100 milliseconds.

   - **20–40 milliseconds:** Offers a good trade-off between time and frequency resolution; often used for detecting transients and percussive elements.
   - **50 milliseconds:** A standard window size suitable for general-purpose audio analysis across diverse signals.

- **100 milliseconds:** Prioritizes frequency resolution, making it more effective for analyzing stationary or slowly varying sounds.

2. **Windowing:** Window function like a filter. , reducing the amplitude of the signal towards both end of the window, and it causes a reduction in Spectral Leakage when applying FFT. Reduction in amplitude is known as *tapering* or *fading* at the window boudaries.

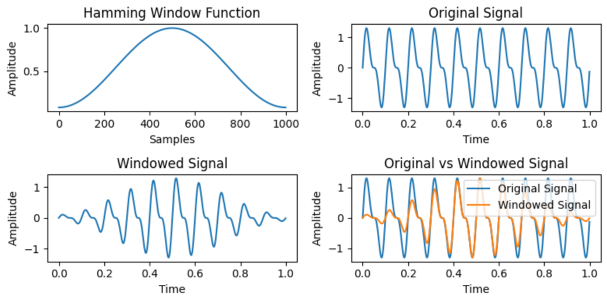

Fig. 7: Window function[6].

Spectral Leakage[6] is a phenomenon that spectrum power 'leak' to nearby range when applying data transform like FFT with finite signals.It happens when transforming unnatullay separated finite signals.

3. **Fourier Transform:** The Discrete Fourier Transform is applied to each windowed frame to obtain the frequency domain representation of the signal.

4. **Frequency Binning:** The frequency range is divided into a seires of frequency bands or bins (typically logarithmic, alined with musical notes or Mel scale).

5. **Calculate Spectral Peaks and Valleys:** For each frequency band, compute the maximum (peak) and minimum (valley) energy with in that band.

6. **Compute Spectral Contrast:** Apply DCT to the log filterbank energies to obtain the MFCCs. This step reduces correlation and dimensionality, typically keeping only the first 12–13 coefficients.

   For each frequency band, spectral contrast is calculated as the ratio between the peak and valley energies (typically measured in decibels, dB). The general formula is:

$$\text{Spectral Contrast} = 10 \times \log_{10}\left(\frac{\text{Peak Value}}{\text{Valley Value}}\right) \quad (1)$$

   Here's a detailed breakdown of each component of the formula:

   - **Peak Value:** The maximum energy point in the spectrum within a specific frequency band.
   - **Valley Value:** The minimum energy point in the same frequency band.
   - **Ratio of Peak and Valley:** This ratio captures the contrast between the most and least intense parts of the spectrum in each band.
   - **Logarithmic Scale:** The use of a logarithm compresses the dynamic range and aligns with the non-linear nature of human auditory perception—similar to how decibels are used to express sound intensity.
   - **Multiplication by 10:** Converting the logarithmic ratio into decibels (dB) involves multiplying by 10, which is a standard practice in signal processing to represent perceived loudness.

3.3 Tools

- **Librosa (Python Library)**: Provides functions for MFCC extraction. Documentation[7].

4 EDA(*Exploratory Data Analysis*)

Once the data preprocessing is completed, an exploratory analysis will be carried out using statistical and visual techniques to understand the nature of the phenomenon, facilitating the subsequent development of the predictive model.

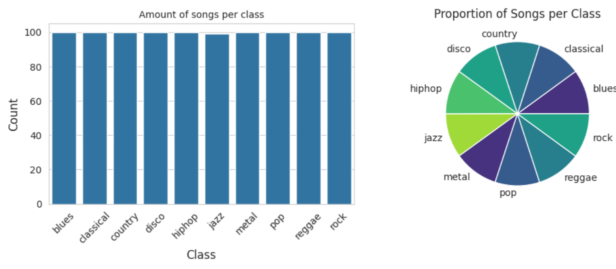

Fig. 8: Distribution of classes.

As EDA shows, the distribution of classes in the classification model is balanced, with roughly equal representation across all classes.

4.1 Influence of independent variables

- **Pearson**: Measures the linear relationship between two variables. The formula for the Pearson correlation coefficient (r) is:

$$r = \frac{\sum(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum(X_i - \bar{X})^2 \sum(Y_i - \bar{Y})^2}}$$

  where X_i and Y_i are the observations of the two variables, and $\bar{X}$ and $\bar{Y}$ are their means.

- **Spearman**: Measures the monotonic relationship (when two variables change in the same direction), but not necessarily in a linear way. The formula for the Spearman correlation coefficient (ρ) is:

$$\rho = 1 - \frac{6\sum d_i^2}{n(n^2-1)}$$

  where d_i is the difference between the ranks of the two variables, and n is the number of observations.

- **Distance Correlation (DCOR)**: Measures all types of dependencies between two variables, whether linear or non-linear. The formula for distance correlation is:

$$\mathrm{DCOR}(X, Y) = \frac{\sqrt{\sum_{i,j}(d(X_i, X_j) - d(Y_i, Y_j))^2}}{n}$$

where $d(X_i, X_j)$ is the distance between the observations of X, and $d(Y_i, Y_j)$ is the distance between the observations of Y, and n is the number of observations.

Feature	Pearson	Spearman	Distance Correlation
MFCC			
mfcc_2	-0.355	-0.347	0.393
mfcc_1	0.340	0.298	0.355
mfcc_9	0.198	0.203	0.259
mfcc_7	0.183	0.174	0.221
mfcc_11	0.141	0.131	0.166
mfcc_8	0.109	0.110	0.164
mfcc_10	0.116	0.113	0.155
mfcc_6	0.081	0.088	0.153
mfcc_4	-0.014	-0.006	0.152
mfcc_3	0.063	0.069	0.147
mfcc_5	0.068	0.067	0.142
Spectral Contrast			
spectral_contrast_5	-0.413	-0.399	0.450
spectral_contrast_4	-0.374	-0.359	0.410
spectral_contrast_6	-0.362	-0.341	0.401
spectral_contrast_3	-0.334	-0.316	0.368
spectral_contrast_2	-0.276	-0.241	0.312
spectral_contrast_7	-0.115	-0.109	0.183
spectral_contrast_1	-0.013	0.022	0.133

TABLE 1: Comparison of Pearson, Spearman, and Distance Correlation for MFCC and Spectral Contrast with the Target Variable

5 *K*-Nearest Neighbors (KNN)

The *k*-nearest neighbors (KNN) algorithm is a non-parametric, supervised learning method primarily used for classification. It classifies data points based on proximity, operating under the assumption that similar instances exist near each other in feature space. In classification, a data point is assigned the label most common among its k nearest neighbors, by majority vote.

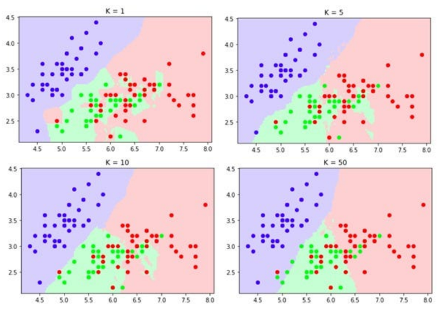

Fig. 9: KNN Model [8].

KNN algorithm, does not perform explicit training, all computations are deferred until prediction time, making it an instance-based or memory-based learning approach..

5.1 Distance Metrics

- **Euclidean distance** ($p = 2$): It calculates the straight-line (L2 norm) distance between the query point and another point, defined as:

$$d(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{n} (x_i - y_i)^2 \right)^{1/2}$$

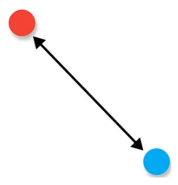

Fig. 10: Euclidean distance[9]

- **Manhattan distance** ($p = 1$): It measures the sum of absolute differences between components:

$$d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} |x_i - y_i|$$

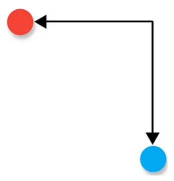

Fig. 11: Manhattan distance[9]

- **Minkowski distance**: A generalized form of both Euclidean and Manhattan distances, parameterized by p:

$$d(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p}$$

When $p = 1$, it reduces to Manhattan distance; when $p = 2$, it becomes Euclidean distance.

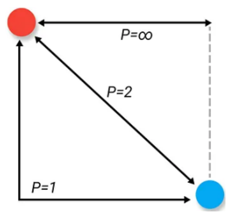

Fig. 12: Minkowski distance[9]

- **Hamming distance**: Counts the number of positions at which corresponding elements differ:

$$d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} \mathbb{1}_{x_i \neq y_i}$$

where $\mathbb{1}_{x_i \neq y_i}$ is the indicator function that returns 1 if $x_i \neq y_i$, and 0 otherwise.

5.2 Results

Table 2 presents the performance of the K-Nearest Neighbors (KNN) classifier across various feature sets, both before and after applying hyperparameter tuning using GridSearchCV.

Feature Set	Accuracy	Cross-Validation Mean Accuracy
Without Tuning		
Original Data	0.64	**0.593**
Custom Data	0.555	0.553
MFCC Data	0.475	0.546
Spectral Data	0.495	0.476
Chroma Data	0.32	0.370
RMS Data	0.21	0.206
Applying GridSearchCV		
Original Data	0.675	**0.638**
Custom Data	0.575	**0.606**
MFCC Data	0.510	0.568
Spectral Data	0.495	0.521
Chroma Data	0.340	0.402
RMS Data	0.225	0.248

TABLE 2: ACCURACY AND CROSS VALIDATION OF KNN BEFORE AND AFTER APPLYING GRIDSEARCHCV

Further analysis of the classifier's behavior is provided in Table 3, which details the precision, recall, and F1-score for each class. The model performed particularly well on the classical and metal genres, with F1-scores of 0.88 and 0.85, respectively. In contrast, genres like disco, country, and blues showed relatively lower classification metrics. The overall accuracy was 0.55, with a macro-averaged F1-score of 0.55.

Class	Precision	Recall	F1-Score	Support
blues	0.39	0.35	0.37	20
classical	0.90	0.86	**0.88**	22
country	0.35	0.45	0.39	20
disco	0.40	0.29	0.33	28
hiphop	0.47	0.47	0.47	17
jazz	0.56	0.43	0.49	21
metal	0.87	0.83	**0.85**	24
pop	0.48	0.92	0.63	12
reggae	0.52	0.80	0.63	15
rock	0.54	0.33	0.41	21
Accuracy		**0.55**		200
Macro Avg	0.55	0.57	0.55	200
Weighted Avg	0.56	0.55	0.54	200

TABLE 3: CLASSIFICATION REPORT FOR KNN ON UPDATED DATA

The plot 13 illustrates the impact of varying the number of neighbors k on the performance of the KNN classifier.

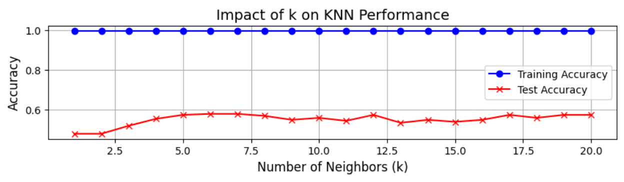

Fig. 13: K Performance.

6 RANDOM FOREST REGRESSOR

The **RandomForestRegressor** is a regression model that uses an ensemble of multiple decision trees to make predictions. This model employs a technique called **bagging** to generate fully-grown decision trees in parallel from random subsets of both the data and the features.

The final prediction is an average of all the predictions made by the different decision trees that were generated during the **bagging** process.

This model is well-suited to the task, as training each tree on different subsets of the data reduces dependency among them, increasing robustness and generalization capability. Moreover, it is capable of handling non-linear relationships effectively and works well with noisy data or outliers.

6.1 Results

Table 4 presents the performance of the RandomForest classifier across various feature sets, both before and after applying hyperparameter tuning using GridSearchCV.

Feature Set	Accuracy	Cross-Validation Mean Accuracy
Without Tuning		
Original Data	0.645	**0.658**
Custom Data	0.585	0.580
MFCC Data	0.54	0.556
Spectral Data	0.55	0.527
Chroma Data	0.39	0.390
RMS Data	0.21	0.218
Applying GridSearchCV		
Original Data	0.62	**0.663**
Custom Data	0.615	**0.600**
MFCC Data	0.54	0.584
Spectral Data	0.545	0.549
Chroma Data	0.39	0.392
RMS Data	0.255	0.239

TABLE 4: ACCURACY AND CROSS VALIDATION OF RANDOMFOREST BEFORE AND AFTER APPLYING GRIDSEARCHCV

Compared to KNN, the Random Forest classifier demonstrated better generalization across the diverse class distribution.

Class	Precision	Recall	F1-Score	Support
blues	0.50	0.40	0.44	20
classical	0.82	0.82	**0.82**	22
country	0.47	0.40	0.43	20
disco	0.71	0.43	0.53	28
hiphop	0.53	0.53	0.53	17
jazz	0.46	0.52	0.49	21
metal	0.81	0.92	**0.86**	24
pop	0.42	0.83	0.56	12
reggae	0.40	0.80	0.53	15
rock	0.83	0.24	0.37	21
Accuracy		**0.57**		200
Macro Avg	0.59	0.59	0.56	200
Weighted Avg	0.62	0.57	0.56	200

TABLE 5: CLASSIFICATION REPORT FOR RANDOMFOREST ON CUSTOM DATA

7 Support Vector Machine (SVM)

A support vector machine (SVM)[10] is a supervised machine learning algorithm designed to classify data by finding an optimal hyperplane that maximizes the margin between classes in an N-dimensional space.

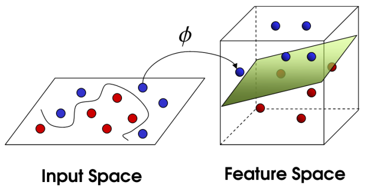

Fig. 14: SVM.

The foundational work on SVMs was developed in the 1990s by Vapnik and colleagues, who introduced the method for function approximation[11], regression estimation, and signal processing.

SVMs operate by identifying the hyperplane that best separates the data into two classes, maximizing the margin between the closest points of opposite classes, these critical data points are known as support vectors.

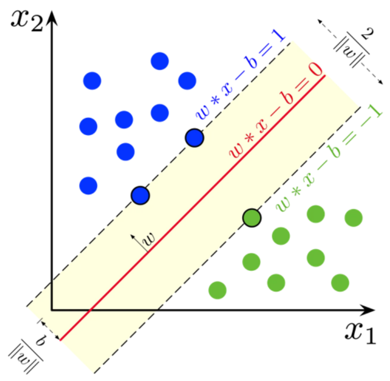

Fig. 15: SVM.

When data is not linearly separable, kernel functions are employed to project the data into a higher-dimensional space where linear separation becomes feasible, a technique referred to as the *kernel trick*.

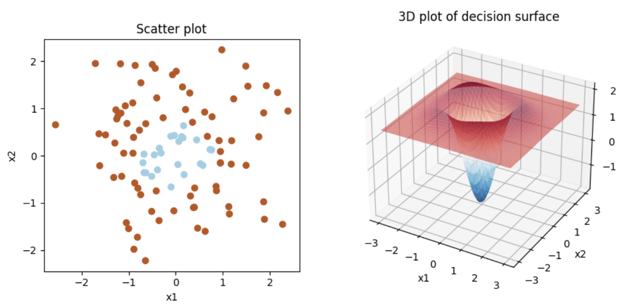

Fig. 16: SVM.

Common kernels include linear, polynomial, radial basis function (RBF), and sigmoid, with the choice depending on the characteristics of the dataset and the problem domain.

This versatility makes SVMs highly effective for both linear and nonlinear classification tasks, enabling robust generalization and accurate predictions.

7.1 Types of SVM Classifiers

- **Linear SVMs**

  Linear SVMs are suitable for linearly separable data, where no transformation is needed to distinguish between classes. The decision boundary and support vectors form what Professor Patrick Winston from MIT refers to as *fitting the widest possible street.* The separating hyperplane is mathematically expressed as:

  $$\mathbf{w} \cdot \mathbf{x} + b = 0$$

  where $\mathbf{w}$ is the weight vector, $\mathbf{x}$ is the input vector, and b is the bias term.

  There are two primary approaches to calculating the margin:

  - **Hard-Margin Classification:** Requires perfect separation of data points outside the support vectors. The constraint is defined as:

    $$(\mathbf{w} \cdot \mathbf{x}_j + b)y_j \geq a$$

    The margin is maximized as:

    $$\max \gamma = \frac{a}{\|\mathbf{w}\|}$$

  - **Soft-Margin Classification:** Allows some misclassifications using slack variables ξ. The hyperparameter C controls the trade-off between margin width and misclassification:

    * Large C: Narrows margin, minimizes misclassifications.
    * Small C: Widens margin, allows more misclassifications.

- **Nonlinear SVMs**

  Nonlinear SVMs handle data that is not linearly separable by mapping input features into a higher-dimensional space using preprocessing techniques. This can lead to overfitting and increased computational cost. The *kernel trick* simplifies this by substituting dot products with kernel functions, enabling efficient computation.

  Common kernel functions include:

  - Polynomial kernel
  - Radial Basis Function (RBF) kernel (also known as Gaussian kernel)
  - Sigmoid kernel

- **Support Vector Regression (SVR)**

  SVR extends SVM to regression tasks, where the target variable is continuous. Like SVMs, SVR seeks to find a hyperplane that maximizes the margin between data points, making it useful for tasks such as time series forecasting.

  Unlike linear regression, SVR does not require prior assumptions about relationships between independent and dependent variables. It learns these relationships directly from the data.

8 Performance of Different Classifiers on the Dataset

To evaluate the performance of various support vector machine (SVM) classifiers, an initial grid search was conducted on the full dataset. The classifiers tested included linear, polynomial, radial basis function (RBF), and sigmoid kernels. This grid search aimed to identify the kernel that yielded the highest classification accuracy without applying any data preprocessing or feature selection techniques.

The results of initial evaluation are summarized in Table 6.

SVM Classifier	Accuracy
Linear	**0.67**
Polynomial Kernel	0.63
Radial Basis Function (RBF) Kernel	**0.71**
Sigmoid Kernel	0.43

TABLE 6: Classification accuracy of SVMs using different kernel functions.

8.1 Sequential Feature Selection (SFS)

To optimize model performance and reduce dimensionality, **Sequential Feature Selection (SFS)** is applied, using both forward and backward approaches with two types of SVM kernels: linear and radial basis function (RBF), the ones with previous better results. This process selects a subset of the most informative features by iteratively adding (forward) or removing (backward) features based on their impact on model accuracy.

In the forward selection, the algorithm starts with no features and adds one feature at a time, selecting the one that improves the model's performance the most at each step. In backward selection, the algorithm starts with all features and removes the least impactful features step-by-step. Both approaches were evaluated using 5-fold cross-validation to ensure generalizability.

The selected features were then used to train SVM classifiers with linear and RBF kernels, and performance was evaluated on a test set of 200 samples. The evaluation metrics include accuracy, precision, recall, and F1-score for each class.

SVM Classifier	Accuracy
Forward Sequential Feature Selector	
Linear	0.63
Radial Basis Function (RBF) Kernel	**0.66**
Backward Sequential Feature Selector	
Linear	0.62
Radial Basis Function (RBF) Kernel	**0.67**

TABLE 7: Accuracy of SVM Classifiers with Forward and Backward Sequential Feature Selection

8.2 Recursive Feature Elimination

To enhance the performance of the Support Vector Machine (SVM) classifiers and reduce dimensionality, we applied two advanced feature selection techniques: Recursive Feature Elimination **(RFE)** and Recursive Feature Elimination with Cross-Validation **(RFECV)**. These methods are used to iteratively select the most important features from the dataset by evaluating their contribution to the model's performance. Both techniques are essential for identifying the most relevant features, ensuring a more efficient and accurate model.

- **RFE (Recursive Feature Elimination)** is a feature selection technique that works by recursively removing the least important features based on their impact on model performance. The procedure involves the following steps:
  - A model is initially trained using all available features.
  - The importance of each feature is evaluated, typically by analyzing the model's coefficients or feature weights.
  - The least important feature(s) are removed from the dataset.
  - The process is repeated, with the model being re-trained and the least important features being eliminated at each step.
  - This continues until the desired number of features is selected, ideally those that contribute most to the model's predictive power.
- For our analysis, we used RFE to select a predefined number of features (27 features) based on their ability to improve the model's accuracy.
- After selecting the most relevant features, the SVM classifiers were trained on the reduced feature set, and performance was evaluated using standard classification metrics such as accuracy, precision, recall, and F1-score.
- **RFECV (Recursive Feature Elimination with Cross-Validation)** is an extension of RFE that incorporates cross-validation to evaluate the performance of different feature subsets. Unlike RFE, which simply selects a fixed number of features, RFECV uses cross-validation to automatically determine the optimal number of features based on the model's performance. The steps involved in RFECV are as follows:
  - A model is trained on the full feature set.
  - Cross-validation is performed to assess the model's performance for each subset of features.
  - Features are eliminated one by one, and cross-validation is repeated for each new subset.
  - The process continues until the optimal number of features is found, which is the one that gives the best performance based on cross-validation results.

- RFECV allows us to select features based not only on their individual importance but also on how well they contribute to the overall model's generalization ability.
- In this study, RFECV helped us determine the most optimal set of features by balancing performance and complexity, ensuring that the selected feature subset performed well across different subsets of the data.

8.3 Results

Table 8 presents the cross-validation mean accuracy scores obtained by Support Vector Machine (SVM) classifiers with different kernel functions (Linear and Radial Basis Function) across four feature selection methods: Forward Sequential Feature Selector, Backward Sequential Feature Selector, Recursive Feature Elimination (RFE), and Recursive Feature Elimination with Cross-Validation (RFECV).

Overall, classifiers using the RBF kernel consistently outperformed their linear counterparts across all selection methods. Among the methods evaluated, RFECV achieved the highest accuracy, with the RBF kernel reaching 0.73 and the linear kernel achieving 0.69. This suggests that the combination of recursive feature elimination and integrated cross-validation provides the most robust feature subset for genre classification in this context.

SVM Classifier	**Cross-Validation Mean Accuracy**
Forward Sequential Feature Selector	
Linear	0.63
Radial Basis Function (RBF) Kernel	**0.66**
Backward Sequential Feature Selector	
Linear	0.62
Radial Basis Function (RBF) Kernel	**0.67**
RFE (Recursive Feature Elimination)	
Linear	0.65
Radial Basis Function (RBF) Kernel	**0.7**
RFECV (Recursive Feature Elimination with Cross-Validation)	
Linear	**0.69**
Radial Basis Function (RBF) Kernel	**0.73**

TABLE 8: ACCURACY OF SVM CLASSIFIERS WITH FORWARD AND BACKWARD SEQUENTIAL FEATURE SELECTION

9 CONCLUSIONS

CONCLUSION

In this study, three machine learning classifiers: Support Vector Machine (SVM), k-Nearest Neighbors (KNN), and Random Forest—were applied to the task of musical genre classification. Feature selection techniques were also explored to assess their impact on model performance.

Among the classifiers, SVM with an RBF kernel in combination with Recursive Feature Elimination and Cross-Validation (RFECV) achieved the highest mean cross-validation accuracy of 0.73, demonstrating the effectiveness of this approach in identifying a relevant and compact feature subset. In contrast, the best result for the KNN classifier was 0.638 when using all available features, and the Random Forest classifier reached a slightly higher performance of 0.663, also using the full feature set.

These findings indicate that while all three models show reasonable predictive capability, the SVM with appropriate feature selection significantly outperforms the rest. This highlights the importance of both model choice and feature selection strategy in classification tasks.

REFERENCES

[1] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, 2002.

[2] Wikipedia contributors. (2025) Sine wave.

[3] Stack Overflow contributors. (2015) How to split speech data on frames and compute mfcc?

[4] A. Tanveer, "Mfccs made easy," 2018.

[5] Y. Mo, "Understanding mfcc: Made easy," 2023.

[6] ——, "Understanding spectral contrast: Made easy," 2023.

[7] B. McFee, C. Raffel, D. Liang, D. P. W. Ellis, J. Richman, Z. Chen, and Others, "librosa: Audio and music signal analysis in python," 2015.

[8] IBM, "What is the k-nearest neighbors (knn) algorithm?" 2023.

[9] A. Chakole, "Distance metrics for data science," 2020.

[10] IBM. (2023, Dec.) What are support vector machines (svms)?

[11] V. N. Vapnik, S. Golowich, and A. Smola, "Support vector method for function approximation, regression estimation, and signal processing," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 9. MIT Press, 1996.

[12] AFAcoding, "Musical genre classification," https://github.com/AFAcoding/Musical-Genre-Classification, 2025.

A APPENDIX

A.1 Appendix Section

The code used in this work is available at the following repository [12]