UNIVERSIDADE D
COIMBRA

Music Genre Classification - Report

Phase 1 - Binary Classification with MDC and feature reduction/selection techniques

Jorge Júnior Rodrigues Martins - 2021207642
Sancho Amaral Simões - 2019217590

Supervised by: Professor César Teixeira

Faculdade de Ciências e Tecnologias Universidade da Universidade de Coimbra

Coimbra March 21, 2023

Contents

1 Introduction

Music genre classification is a challenging task in the field of music information retrieval due to the subjective nature of music and the varying characteristics of different musical genres. One of the most commonly used datasets for music genre classification research is the GTZAN dataset. This dataset contains a diverse set of audio recordings representing ten different musical genres, including blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae, and rock.

The GTZAN dataset provides a wide range of features that can be used for music genre classification. These features include time-domain and frequency-domain features, such as Mel Frequency Cepstral Coefficients (MFCCs), spectral contrast, spectral centroid, and spectral roll-off.

To classify the musical genres in the GTZAN dataset, we will apply pattern recognition techniques, including Mahalanobis Distance Classifier (MDC), Linear Discriminant Analysis Fisher (LDA Fisher), and Principal Component Analysis (PCA). These techniques have been widely used in various fields for classification tasks, including music genre classification.

Overall, the present report aims to provide a comprehensive overview of the GTZAN dataset and the pattern recognition techniques used to classify musical genres. We hope that our findings will contribute to the growing body of research on music genre classification and inform the development of more accurate and reliable music genre classifiers in the future.

2 Goal

The goal of this project is to develop classifiers for genre discrimination using a traditional pattern-recognition pipeline approach. The project aims to explore two scenarios: Scenario A (Binary classification) and Scenario B (Multiclass classification). The dataset used for this project is available at https://www.kaggle.com/datasets/gabrielopecs/gtzan-modified-music-genre-classification and contains 1000 music snippets in ten different genres.

The project will involve splitting the data into two datasets, one for classifier development and the other for testing. The development dataset will be used for training and validation, while the testing dataset will be used for evaluating the classifiers. The data will also be normalized/standardized, and the factors for normalization/standardization will be obtained from the development dataset and applied prospectively in the testing dataset.

The project will also involve feature selection and dimensionality reduction techniques, such as PCA and LDA, to reduce the number of features and improve classifier performance. The classifiers will be designed using the Minimum Distance classifier and Fisher LDA for Scenario A.

The overall goal of the project is to compare the performance of different classifiers and feature selection/reduction techniques in discriminating between different music genres. The results of the project will be presented in this report which includes a detailed analysis of the findings and insights gained through manual inspection of the predictions.

3 GTZAN dataset

The GTZAN dataset is a collection of 1000 audio clips of 30 seconds each, representing 10 different musical genres. The purpose of the dataset is to facilitate research in the area of music genre classification and other related fields. The dataset was created by George Tzanetakis and Perry Cook in 2002 and has since become a widely used benchmark dataset for evaluating the performance of music genre classification algorithms.

Musical genres

The 10 musical genres present in the GTZAN dataset are:

- Blues
- Classical
- Country
- Disco
- Hip-hop
- Jazz
- Metal
- Pop
- Reggae
- Rock

Features

The GTZAN dataset comprises 34 distinct features for each audio clip, which can be broadly categorized into three groups: time-domain, frequency-domain, and spectral features. However, the dimension of the feature matrix is much greater than 34 (i.e., 197), as it includes various statistics extracted from each feature, such as mean, median, and so on.

Time-domain features:

- Zero Crossing Rate (ZCR) - $ZCR = \frac{1}{N-1}\sum_{n=1}^{N-1} |sgn(x[n]) - sgn(x[n-1])|$
- Energy - $E = \sum_{n=1}^{N} x^2[n]$
- Entropy of Energy (EE) - $EE = -\sum_{i=1}^{M} p(i)\log_2(p(i))$
- Spectral centroid - $SC = \frac{\sum_{i=1}^{N} f(i)m(i)}{\sum_{i=1}^{N} m(i)}$
- Spectral spread - $SS = \sqrt{\frac{\sum_{i=1}^{N} (f(i)-SC)^2 m(i)}{\sum_{i=1}^{N} m(i)}}$

- Spectral entropy - $SE = -\sum_{i=1}^{N} p(i) \log_2(p(i))$
- Spectral flux - $SF = \sqrt{\sum_{i=1}^{N} (F(i) - F'(i))^2}$
- Spectral rolloff - $SR = \sum_{i=1}^{k} \frac{\sum_{j=1}^{i} m(j)}{\sum_{j=1}^{N} m(j)}$

Frequency-domain features:

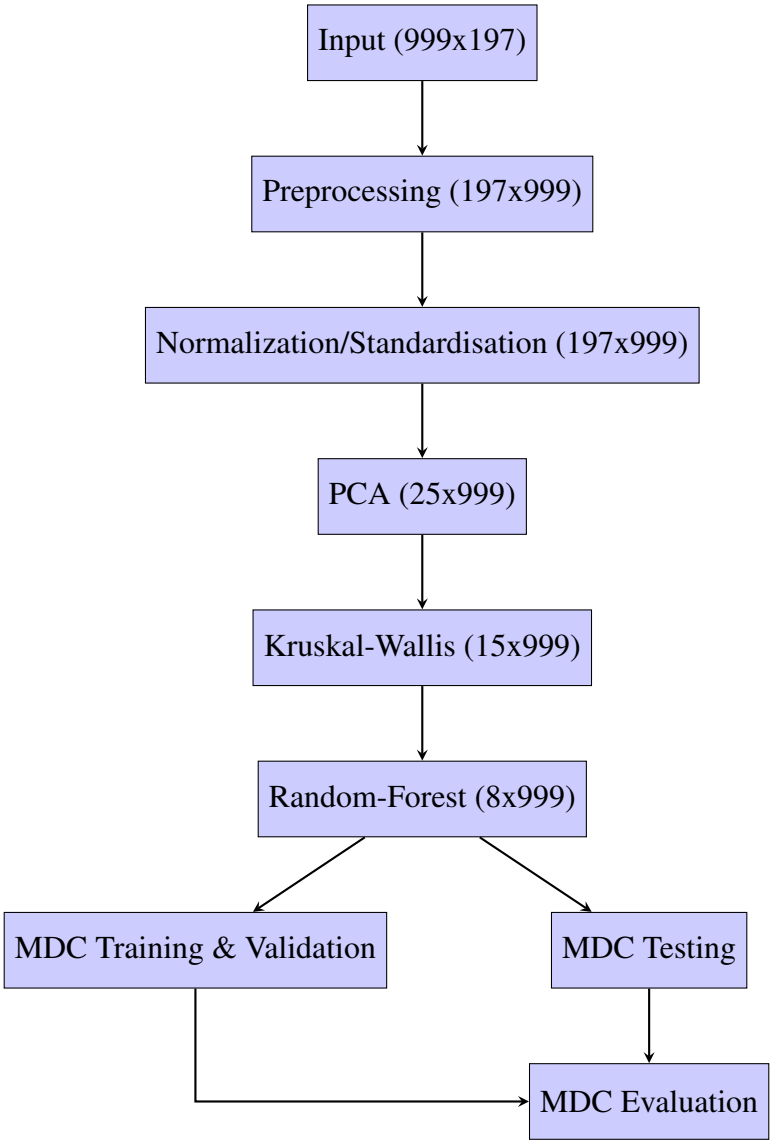
- MFCCs - $MFCC_n = \sum_{i=1}^{N} log_{10}(S(i)) cos[n(i - \frac{1}{2})\frac{\pi}{N}]$
- Chroma vector - $CV_i = \sum_{j=1}^{12} S(i) w_j(i - k)$

Spectral features:

- Mel-frequency spectrogram - $S_m(k, i) = \sum_{j=1}^{N} H(k - j) M(j, i)$
- Constant-Q transform - $CQT(k, i) = \sum_{j=1}^{N} Q(k - j) M(j, i)$
- Chromagram - $CG(i, j) = \sum_{k=1}^{12} S_m(k, i) w_j(k)$

The GTZAN dataset is a valuable resource for constructing a musical genre classifier using pattern recognition techniques. The goal of this classifier is to accurately classify audio clips into their corresponding musical genres. Various pattern-recognition and machine learning algorithms such as Minimum-Distance Classifiers, LDA Fisher, PCA, Support Vector Machines, Random Forests, and Neural Networks can be trained using the GTZAN dataset and evaluated on their performance. By accurately classifying audio clips, the musical genre classifier can be used for various applications such as music recommendation systems, personalized playlists, and automatic music tagging.

4 General Architecture



This flowchart represents the pattern recognition processing steps involved in this project.

The input to the system is a matrix of dimensions 999x197, which represents a set of 999 samples, each with 197 features - the GTZAN dataset. The first step in processing is to perform preprocessing on the input data, which involves reshaping the matrix to dimensions 197x999 and splitting it into training, validation, and testing data.

Next, the data is normalized or standardized to ensure that each feature has a similar scale, which can help with accuracy in later steps.

After normalization, principal component analysis (PCA) is applied to the data to reduce its dimensionality. The output of PCA is a matrix of dimensions 25x999, which represents the most significant 25 principal components.

The next step is to perform a Kruskal-Wallis test, which is a non-parametric statistical test used to determine if there are significant differences between groups of data. In this case, it is used to select the 15 most significant features from the 25 principal components.

After feature selection with the Kruskal-Wallis test, the data is fed into a random forest algorithm to extract the importance of the 15 remaining features. The output of the random forest algorithm is a matrix of dimensions 8x999, which represents the 8 most significant features for genre classification.

Finally, a multi-dimensional classifier (MDC) is trained and validated using the selected features. The MDC is then tested on a separate set of data and evaluated for accuracy in genre classification.

The feature reduction sequence of 197-25-15-8 was used in the current project, but it is important to note that this sequence may not be optimal for this dataset and classification problems. It is possible that important features are being discarded at the beginning of the sequence, leading to suboptimal performance. Because of this fact, this sequence may be altered in the future.

5 Dataset

5.1 Loading Dataset

The dataset is provided in "CSV" format and the function "readtable" was used to load the dataset into the workspace, where we retrieved the features and the according labels. The labels were then transformed from "strings" into a numerical format (1-10) using the "ismember" function so the whole dataset is numerical.

5.2 Dataset Inbalance

The problem of dataset imbalance is a common issue that can affect the performance of machine learning models. When working with binary classification, it is often the case that one class has significantly fewer instances than the other. This can lead to a bias towards the majority class, resulting in poor classification accuracy for the minority class.

To address this issue, oversampling can be used to generate synthetic data for the minority class. This involves creating new instances of the minority class by either duplicating existing instances or generating new ones based on the existing data. By doing this, the dataset becomes more balanced and the model can learn to better distinguish between the two classes.

In the case of a multiclass classification problem, the dataset can be easily balanced by shuffling the instances uniformly across all classes. This ensures that each class has an equal number of instances, allowing the model to learn from all classes equally.

5.3 Data Splitting

The first subset is the development set, which comprises 80% of the original dataset. This set is used to develop and fine-tune your machine-learning model before testing it on the other subsets.

The second subset is the testing set, which comprises 20% of the original dataset. This set is used to evaluate the performance of your machine learning model after it has been trained on the training set.

The third subset is the training set, which comprises 80% of the development set. This set is used to train your machine-learning model on the features and labels of your dataset. The fraction of the original dataset allocated to the training set is calculated as 0.8 multiplied by the fraction allocated to the development set.

The final subset is the validation set, which comprises 20% of the development set. This set is used to validate the performance of your machine learning model during training and to ensure that it is not overfitting the training data. The fraction of the original dataset allocated to the validation set is calculated as 0.2 multiplied by the fraction allocated to the development set.

6 Normalization/Standardisation

Normalizing a dataset is an important preprocessing step in pattern recognition and machine learning, which involves transforming the numerical features of a dataset to a common scale. The goal of normalization is to remove the scale differences between different features and ensure that they all contribute equally to the analysis. This is important because many algorithms, such as PCA, Kruskal-Wallis, K-Nearest Neighbors and Support Vector Machines, are sensitive to the relative magnitudes of the features. Failure to normalize the data may result in one or more features dominating the analysis and negatively impacting the performance of the model. By normalizing the data, we can ensure that the machine learning algorithm treats each feature equally, resulting in more accurate and fair analysis. Additionally, normalization can help to reduce the impact of outliers and make the dataset more robust to noise. Overall, the goal of normalization is to improve the quality of the analysis and enable more accurate and reliable predictions.

6.1 Normalization techniques

For this project, we used three different normalization techniques:

- **Z-score normalization**

  The z-score normalization, also known as standardization, transforms the data to have a mean of zero and a standard deviation of one. It is commonly used when the input data is normally distributed. The formula for z-score normalization is given by:

  $z = \frac{x-\mu}{\sigma}$

  where x is the original data point, μ is the mean of the dataset, σ is the standard deviation of the dataset, and z is the normalized data point.

- **Unit Vector normalization**

  The unit vector normalization, also known as vector normalization, scales each data point to have a unit norm. It is commonly used in machine learning applications that involve calculating distances between data points. The formula for unit vector normalization is given by:

  $\mathbf{u} = \frac{\mathbf{x}}{||\mathbf{x}||}$

  where $\mathbf{x}$ is the original data vector, $||\mathbf{x}||$ is the Euclidean norm of the vector, and $\mathbf{u}$ is the normalized vector (i.e., a unit vector).

- **Min-max normalization**

  The min-max normalization scales the data to a fixed range, typically between 0 and 1. It is commonly used when the range of input data is known and the values are evenly distributed within the range. The formula for min-max normalization is given by:

  $x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$

  where x is the original data point, $x_{\min}$ is the minimum value in the dataset, $x_{\max}$ is the maximum value in the dataset, and x' is the normalized data point (in the range of 0 to 1).

These normalization techniques are useful for preparing data for pattern recognition and machine learning models as they help in improving the accuracy of the models by ensuring that the input data is on a similar scale. Choosing the right normalization technique depends on the specific characteristics of the dataset and the pattern recognition model being used.

7 Feature Reduction And Selection

Feature selection and reduction play a crucial role in pattern recognition and machine learning. One of the main reasons for their importance is the curse of dimensionality, which refers to the fact that as the number of features increases, the complexity of the model also increases. This, in turn, can lead to overfitting, decreased generalization performance, and longer training times.

Feature selection and reduction techniques aim to mitigate the curse of dimensionality by selecting the most informative features or reducing the dimensionality of the feature space. By selecting only the most relevant features, we can reduce the noise and improve the model's performance, while reducing the computational cost.

Furthermore, feature selection and reduction can also help in better understanding the data and identifying which features are most relevant to the task at hand.

7.1 PCA

PCA (Principal Component Analysis) is an algebraic technique used to transform a high-dimensional dataset into a lower-dimensional dataset while retaining most of the original variation. It works by finding the principal components (linear combinations of the original features) that capture the most variation in the data. The number of principal components chosen determines the amount of variation retained in the reduced dataset.

PCA involves finding the eigenvectors and eigenvalues of the covariance matrix of the data. The eigenvectors correspond to the principal components and the eigenvalues represent the amount of variance captured by each principal component. The projection of the data onto the principal components results in a new dataset with a reduced number of dimensions, while still capturing most of the original variation.

Despite its usefulness, there are several problems associated with PCA that can impact its effectiveness and the interpretability of the results. Some of the main problems of PCA are:

- Assumes linear relationships: PCA assumes that the relationships between the input variables are linear. This means that if the relationships are nonlinear, PCA may not be effective at capturing the underlying structure of the data. In such cases, other dimensionality reduction techniques that can capture nonlinear relationships may be more appropriate.

- Requires standardized data: PCA requires the data to be standardized so that each variable has a mean of 0 and a standard deviation of 1. This means that the technique is sensitive to the scale of the input variables. If the input variables are not standardized, the resulting principal components may be influenced more by variables with larger variances than by variables with smaller variances.

- Can be sensitive to outliers: PCA can be sensitive to outliers, as they can have a large impact on the variance of the data. This means that outliers can have a significant influence on the resulting principal components, potentially leading to misleading results.

- Interpreting the results can be difficult: Although PCA can effectively reduce the dimensionality of the data, interpreting the resulting principal components can be difficult. The principal components themselves are linear combinations of the original variables, and their interpretation can be complex, particularly when the number of principal components is large.

- May not preserve all of the information: PCA reduces the dimensionality of the data by identifying the principal components that capture the maximum variance in the data. However, this means that some of the information in the original data may be lost in the transformation process. This can be a problem if it is important to preserve all of the information in the original data.

Despite these problems, PCA can still be a useful technique for dimensionality reduction in many cases, particularly when the input variables have linear relationships and the data is well-behaved. However, it is important to be aware of these problems and to carefully evaluate the results of PCA to ensure that they are meaningful and interpretable.

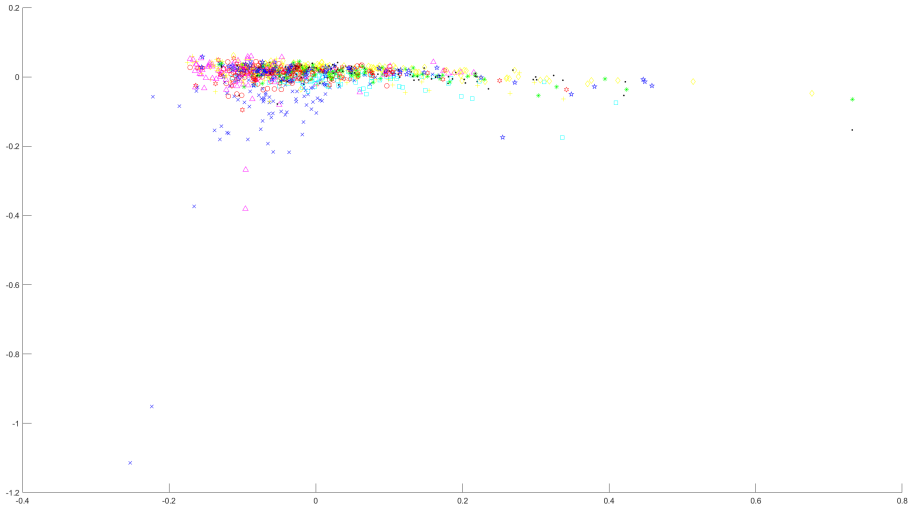


Figure 1: ppatterns after PCA

Even though we can only visualize data in 3D dimensions it is possible to have a notion of how is the data distributed in higher dimensions by analyzing its projection in lower, visualizable dimensions. In this case, we can see that after applying PCA to the dataset, the variance of the data is visibly predominant in one (of possibly many) directions.

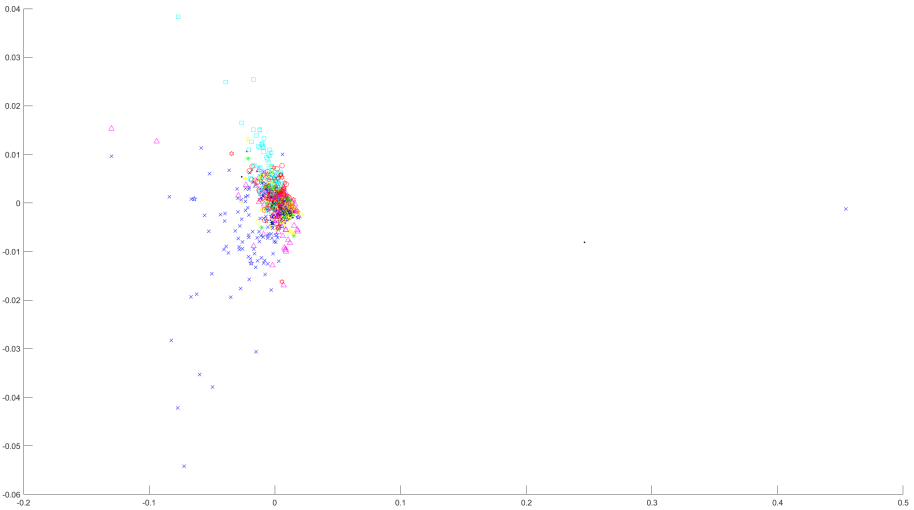


Figure 2: ppatterns after PCA + KW

After applying Kruskal-Wallis to the data after PCA, it is possible to observe some emerging clusters, even though we are in visualizing the feature space projected in a lower dimensional space. This indicates that the dataset is becoming more linearly separable during the feature selection/reduction process.

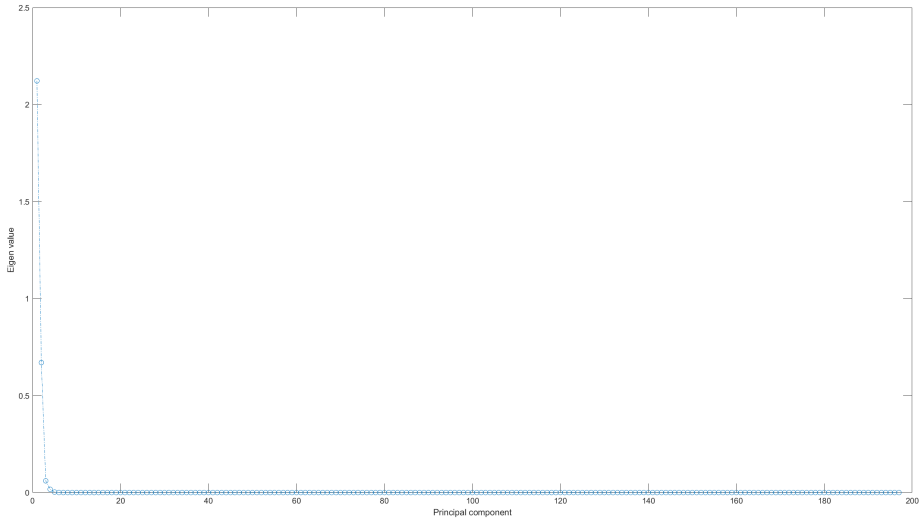


Figure 3: PCA Eigen Values with Z-score normalization

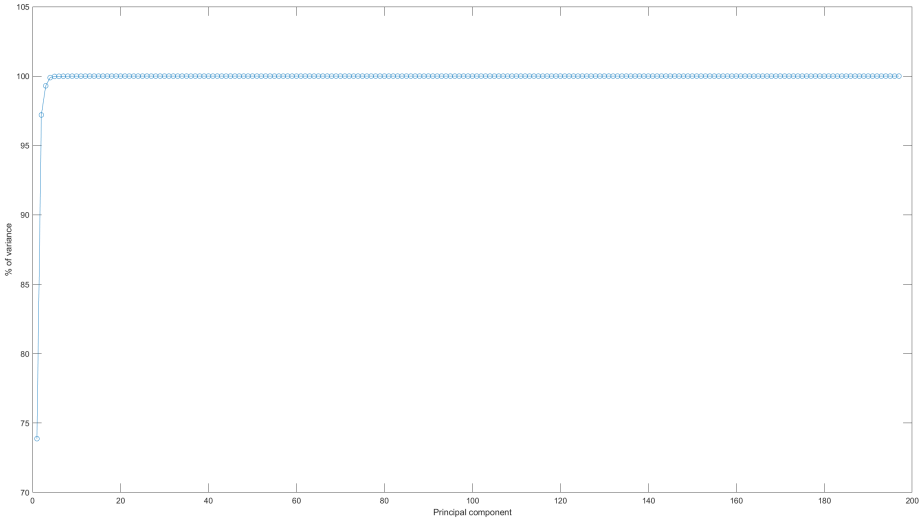


Figure 4: PCA variance with Z-score normalization

After analyzing the graphs above, it is possible to deduce that there are only a handful of features that contribute to the overall variance of the dataset. Despite that fact, a more conservative attitude was taken when applying PCA to the GTZAN dataset, instead of simply taking the mentioned features. This is because, as mentioned before, PCA only captures the variance of each component and variance isn't the only measure of importance for a given feature.

7.2 Kruskal Wallis

Kruskal-Wallis is used to determine which features are most significant in differentiating between different classes in a dataset. The test ranks the features based on their ability to

discriminate between classes, with higher-ranked features indicating a stronger correlation between the feature and class label. Overall, Kruskal-Wallis is a useful tool in feature selection as it provides a statistical measure of the importance of each feature in differentiating between classes, allowing researchers to focus on the most significant features in their analysis.

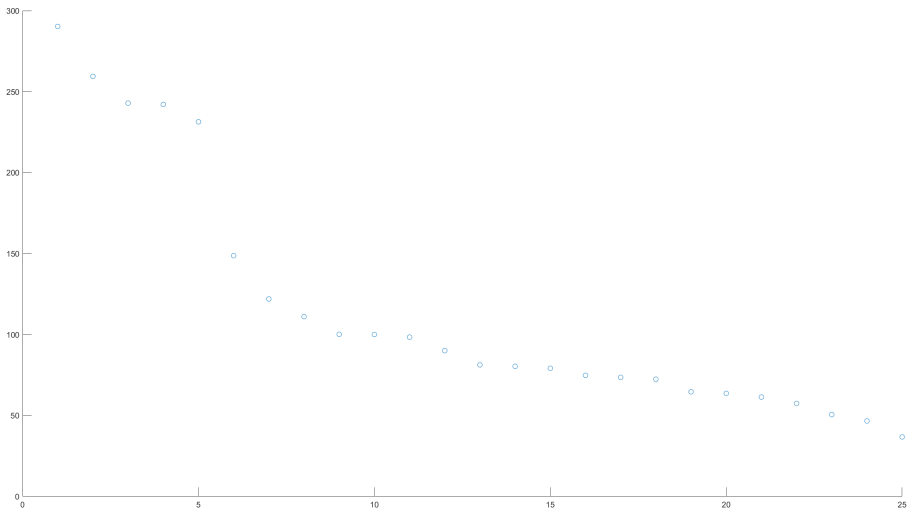


Figure 5: KW scatter ranking

7.3 Correlation

A correlation matrix is a table that displays the correlation coefficients between pairs of variables.

In a correlation matrix, each row and column represents a variable, and the cells in the matrix show the correlation coefficients between each pair of variables. The diagonal cells always have a correlation coefficient of 1.0 because they represent the correlation between a variable and itself.

Correlation matrices are useful for identifying patterns and relationships between variables in a dataset. They can help to identify variables that are strongly related to each other, which can be useful for developing predictive models or understanding complex systems.

In the context of feature selection or reduction, correlation refers to the degree to which two or more variables are linearly related. When there is a high correlation between two variables, they provide redundant information to a machine learning model. In other words, they provide similar information, and using both variables does not significantly improve the performance of the model.

When performing feature selection or reduction, the goal is to select a subset of features that are most relevant for predicting the target variable. Having correlated features in the dataset can lead to several issues in this process:

Redundant features: Correlated features can provide redundant information to the model, which can increase the complexity of the model and slow down training time. This can also lead to overfitting, where the model performs well on the training data but poorly on new, unseen data.

Unstable model: When two or more features are highly correlated, small changes in one feature can lead to significant changes in the other feature. This can make the model unstable and difficult to interpret.

Decreased model accuracy: Correlated features can also decrease the accuracy of the model, as they can introduce bias and noise into the model. This can lead to inaccurate predictions and reduced performance.

Therefore, in feature selection or reduction, the existence of correlation is generally undesired as it can negatively impact the performance and stability of the model. To mitigate this issue, techniques such as correlation analysis or principal component analysis (PCA) can be used to identify and remove correlated features from the dataset.

To gain insight into the interdependence of the variables, we constructed a correlation matrix for every normalization and feature reduction method, and in all combinations, the results showed us that the variables had almost no correlation with each other. Below there is an example of two of the correlation matrices obtained:

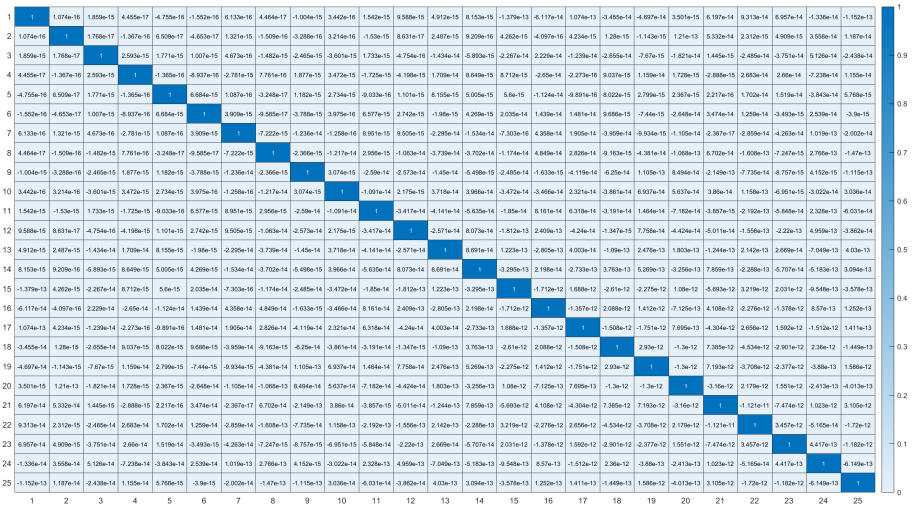


Figure 6: Correlation with "zscore" normalization after PCA

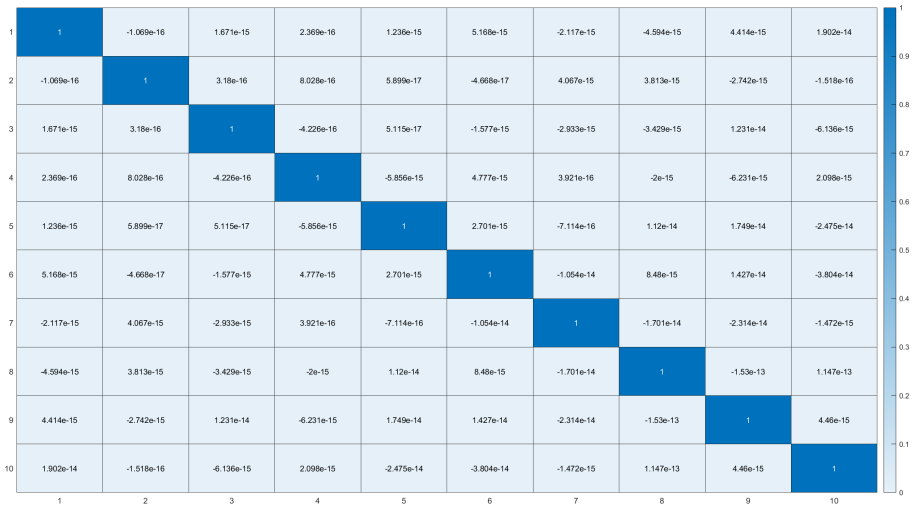


Figure 7: Correlation with "zscore" normalization after PCA and KW

7.4 Random Forest

A random forest is an ensemble learning method that constructs a multitude of decision trees at training time and outputs the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. In the context of our project on music genre recognition, we can use a random forest to identify the most informative features for predicting the genre of a song based on its audio features.

Mathematically: Let X be a set of input audio features and Y be the target variable representing the genre of a song. Random Forests construct T decision trees, where each tree t is built using a random subset of m features from X. The predicted output of the Random Forest is the mode of the predicted outputs of the individual decision trees.

In the context of our project on music genre recognition, we are interested in identifying the most informative audio features for predicting the genre of a song. Random Forests is a powerful method for feature reduction/selection because they can identify and prioritize the most informative features in the data. By constructing multiple decision trees and randomly selecting subsets of features, Random Forests can identify the features that are consistently important across the trees. This is reflected in the feature importance scores that are generated by the algorithm. Features with higher importance scores are more likely to be selected by the algorithm and can be used for feature reduction/selection.

Additionally, Random Forests can handle data with a large number of features and can capture nonlinear relationships between the features and the target variable. In the context of music genre recognition, this is important because there are likely to be complex relationships between the audio features and the genre of a song. By using a Random Forest to identify the most informative features, we can improve the accuracy of our music genre recognition model.

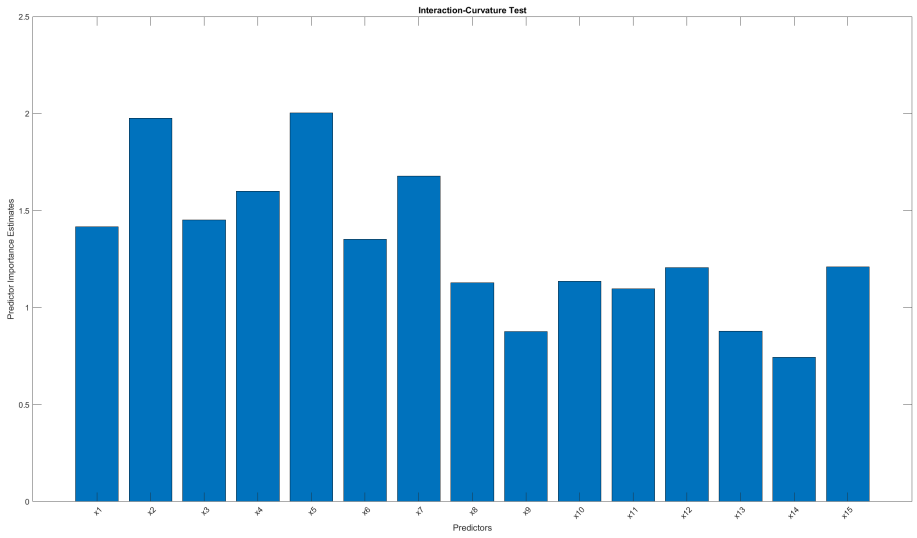


Figure 8: Predictors importance estimated by random forest after PCA

8 Classifiers

In this section we will discuss the Minimum Distance Classifier with several distance formulas, to find out which one is more suitable for this problem.

8.1 Minimum Distance Classifier

A minimum distance classifier is a type of classification algorithm that assigns a test sample to the class with the closest mean or centroid. It works by computing the distance between the test sample and the means of each class, and then assigning the test sample to the class with the minimum distance.

The minimum distance classifier is often used in pattern recognition applications, and is relatively simple and computationally efficient. However, it assumes that the class distributions are Gaussian and have equal covariance matrices, which may not always hold true in practice. Additionally, it can suffer from class overlap, where samples from different classes may have similar mean vectors or centroids, leading to misclassifications. In a minimum distance classifier, many distance metrics can be used to compute the distance between the test sample and the mean of each class. Some commonly used distance metrics include Euclidean distance, Mahalanobis distance, and cosine distance, among others. The used distance metrics are listed below.

8.1.1 Euclidean Distance

d(x,y) = ||x - y||2 = $\sqrt{\sum i = 1^n (x_i - y_i)^2}$

where n is the number of dimensions (features) in the feature space.

8.1.2 Cityblock Distance

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

where n is the number of dimensions (features) in the feature space.

8.1.3 Minkowski Distance

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

where p is a parameter that controls the degree of norm. When $p = 1$, this distance metric is the cityblock distance. When $p = 2$, it is the Euclidean distance.

8.1.4 Chebychev Distance

d(x,y) = $\max_{i=1,\ldots,n} |x_i - y_i|$

This distance metric measures the maximum difference between any feature in the two feature vectors.

8.1.5 Mahalanobis Distance

d(x,y) = $\sqrt{(x-y)^T S^{-1}(x-y)}$ where S is the covariance matrix of the feature space. This distance metric takes into account the correlation between the features in the feature space.

This distance is a useful metric in machine learning and pattern recognition techniques for several reasons:

It takes into account the correlation between features: Unlike Euclidean distance, which treats each feature as independent, Mahalanobis distance considers the covariance matrix of the features. This means that if two features are highly correlated, their contribution to the distance will be reduced, leading to more accurate clustering and classification.

It can handle data with different scales: In Mahalanobis distance, each feature is normalized by its variance. This means that features with larger variances will have a smaller contribution to the distance, which ensures that no feature dominates the distance calculation.

It can identify outliers: Mahalanobis distance can be used to detect outliers in the data because it considers the covariance structure of the data. Outliers tend to have a larger Mahalanobis distance from the center of the data distribution, which makes them easier to detect.

In the context of this project, Mahalanobis distance may be a promising option to use if the data have correlated features and/or features with different scales. It can help to improve the accuracy of clustering and classification models and identify outliers in the data. However, it's important to note that Mahalanobis distance can be computationally expensive for large datasets, so it may not be feasible for some projects.

9 Results

We tested three different normalization techniques and five different distance formulas for the classifiers, as mentioned before, to see which one would perform better in a Binary Classification scenario.

In the next figures we can see the validation results of this experiment, in which we computed five different performance measures:

- Mean-Squared Error (MSE) MSE = $\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$

- Accuracy Accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$
- Specificity Specificity = $\frac{TN}{TN+FP}$
- Sensitivity Sensitivity= $\frac{TP}{TP+FN}$
- F-measure $F_\beta = (1+\beta^2)\frac{precision \times recall}{(\beta^2 \times precision)+recall}$

  Note that recall = sensitivity.

The results which we considered more interesting and showed a reasonable compromise between the five mentioned performance metrics are highlighted. It is important to note that in this context, the f-measure may be the most relevant performance metric since it takes into account the class imbalancing. This is due to the fact that it is computed as the harmonic mean of precision and recall, with a higher score indicating better performance in both precision and recall.

In the context of imbalanced datasets, as one ones used to feed the binary classificators, where one class has significantly fewer samples than the other(s), the F-measure can be a useful metric because it gives equal weight to precision and recall.

However, in some csas, the F-measure does not always compensate for class imbalabcing. There are several alternatives to it that take into account class imbalancing. One such measure is the balanced accuracy, which is the arithmetic mean of the sensitivity and specificity of a classifier. Another measure is the geometric mean, which is the square root of the product of the sensitivity and specificity. Both of these measures give equal weight to each class, regardless of the class distribution.

Another commonly used measure for imbalanced datasets is the area under the receiver operating characteristic curve (AUC-ROC), which measures the trade-off between true positive rate and false positive rate. This measure is often used when the goal is to maximize the true positive rate while keeping the false positive rate low.

The above-mentioned performance metrics may be explored in future work of this project.

	unit vector normalization																								
	chebychev distance					cityblock distance					euclidean distance					mahalanobis distance					minkowski distance				
genre	MSE	acc	spec	sens	f	MSE	acc	spec	sens	f	MSE	acc	spec	sens	f	MSE	acc	spec	sens	f	MSE	acc	spec	sens	f
blues	26,572	0,503	0,563	0,497	0,643	26,572	0,503	0,563	0,497	0,643	27,396	0,497	0,563	0,490	0,636	11,094	0,686	0,750	0,678	0,795	27,396	0,497	0,563	0,490	0,636
classical	0,761	0,818	0,526	0,857	0,892	0,025	0,862	0,368	0,929	0,922	0,157	0,855	0,526	0,900	0,916	0,761	0,881	0,211	0,971	0,935	0,157	0,855	0,526	0,900	0,916
country	21,157	0,597	0,786	0,579	0,724	24,962	0,566	0,786	0,545	0,696	24,962	0,566	0,786	0,545	0,696	39,252	0,478	0,857	0,441	0,607	24,962	0,566	0,786	0,545	0,696
disco	12,736	0,629	0,300	0,651	0,767	13,893	0,616	0,300	0,638	0,757	13,893	0,616	0,300	0,638	0,757	28,233	0,541	0,700	0,530	0,684	13,893	0,616	0,300	0,638	0,757
hiphop	6,440	0,686	0,471	0,711	0,802	6,849	0,692	0,529	0,711	0,805	6,044	0,692	0,471	0,718	0,806	14,491	0,660	0,824	0,641	0,771	6,044	0,692	0,471	0,718	0,806
jazz	13,893	0,654	0,733	0,646	0,772	15,723	0,635	0,733	0,625	0,756	15,723	0,635	0,733	0,625	0,756	4,252	0,748	0,533	0,771	0,847	15,723	0,635	0,733	0,625	0,756
metal	10,572	0,616	0,444	0,638	0,747	6,440	0,673	0,444	0,702	0,792	8,151	0,648	0,444	0,674	0,772	1,233	0,874	0,833	0,879	0,925	8,151	0,648	0,444	0,674	0,772
pop	5,289	0,692	0,474	0,721	0,805	6,440	0,686	0,526	0,707	0,798	6,440	0,686	0,526	0,707	0,798	12,176	0,673	0,789	0,657	0,780	6,440	0,686	0,526	0,707	0,798
reggae	19,025	0,528	0,444	0,539	0,670	36,327	0,472	0,778	0,433	0,592	25,761	0,509	0,611	0,496	0,642	21,893	0,604	0,889	0,567	0,717	25,761	0,509	0,611	0,496	0,642
rock	32,604	0,497	0,692	0,479	0,636	34,440	0,484	0,692	0,466	0,624	32,604	0,497	0,692	0,479	0,636	33,516	0,516	0,846	0,486	0,648	32,604	0,497	0,692	0,479	0,636

Figure 9: MDC performance metrics w/unit vector normalization

	min-max normalization																								
	chebychev distance					cityblock distance					euclidean distance					mahalanobis distance					minkowski distance				
genre	MSE	acc	spec	sens	f	MSE	acc	spec	sens	f	MSE	acc	spec	sens	f	MSE	acc	spec	sens	f	MSE	acc	spec	sens	f
blues	38,264	0,459	0,667	0,442	0,602	39,252	0,453	0,667	0,435	0,595	39,252	0,453	0,667	0,435	0,595	10,572	0,704	0,750	0,701	0,814	39,252	0,453	0,667	0,435	0,595
classical	8,151	0,761	0,875	0,755	0,857	1,063	0,906	0,875	0,907	0,948	4,931	0,811	0,875	0,808	0,891	0,006	0,981	0,750	0,993	0,990	4,931	0,811	0,875	0,808	0,891
country	25,761	0,535	0,643	0,524	0,673	30,818	0,497	0,643	0,483	0,636	27,396	0,509	0,571	0,503	0,652	37,289	0,453	0,643	0,434	0,592	27,396	0,509	0,571	0,503	0,652
disco	9,566	0,629	0,375	0,657	0,761	8,610	0,642	0,375	0,671	0,771	9,082	0,635	0,375	0,664	0,766	17,667	0,616	0,750	0,601	0,738	9,082	0,635	0,375	0,664	0,766
hiphop	4,931	0,736	0,500	0,759	0,840	5,289	0,742	0,571	0,759	0,843	5,289	0,742	0,571	0,759	0,843	4,931	0,736	0,500	0,759	0,840	5,289	0,742	0,571	0,759	0,843
jazz	21,893	0,591	0,842	0,557	0,706	21,893	0,604	0,895	0,564	0,715	22,642	0,597	0,895	0,557	0,709	15,723	0,673	0,947	0,636	0,774	22,642	0,597	0,895	0,557	0,709
metal	3,931	0,679	0,381	0,725	0,797	2,516	0,698	0,333	0,754	0,813	3,623	0,686	0,381	0,732	0,802	0,906	0,862	0,762	0,877	0,917	3,623	0,686	0,381	0,732	0,802
pop	3,623	0,748	0,529	0,775	0,846	4,252	0,736	0,529	0,761	0,837	3,623	0,748	0,529	0,775	0,846	2,038	0,811	0,647	0,831	0,887	3,623	0,748	0,529	0,775	0,846
reggae	33,516	0,453	0,650	0,424	0,576	42,289	0,459	0,900	0,396	0,561	31,704	0,465	0,650	0,439	0,589	25,761	0,585	0,950	0,532	0,692	31,704	0,465	0,650	0,439	0,589
rock	38,264	0,459	0,778	0,418	0,578	32,604	0,497	0,778	0,461	0,619	35,377	0,478	0,778	0,440	0,599	27,396	0,497	0,611	0,482	0,630	35,377	0,478	0,778	0,440	0,599

Figure 10: MDC performance metrics w/min-max normalization

	zscore normalization																								
	chebychev distance					cityblock distance					euclidean distance					mahalanobis					minkowski distance				
genre	MSE	acc	spec	sens	f	MSE	acc	spec	sens	f	MSE	acc	spec	sens	f	MSE	acc	spec	sens	f	MSE	acc	spec	sens	f
blues	26,572	0,541	0,692	0,527	0,678	30,818	0,509	0,692	0,493	0,649	29,082	0,522	0,692	0,507	0,661	4,252	0,748	0,462	0,774	0,850	29,082	0,522	0,692	0,507	0,661
classical	1,610	0,836	0,688	0,853	0,904	0,006	0,931	0,688	0,958	0,961	0,157	0,906	0,688	0,930	0,947	0,057	0,918	0,500	0,965	0,955	0,157	0,906	0,688	0,930	0,947
country	21,157	0,560	0,538	0,562	0,701	28,233	0,503	0,538	0,500	0,649	28,233	0,503	0,538	0,500	0,649	21,157	0,572	0,615	0,568	0,709	28,233	0,503	0,538	0,500	0,649
disco	10,572	0,642	0,579	0,650	0,762	10,572	0,642	0,579	0,650	0,762	10,572	0,642	0,579	0,650	0,762	12,176	0,648	0,684	0,643	0,763	10,572	0,642	0,579	0,650	0,762
hiphop	7,704	0,667	0,400	0,694	0,791	8,151	0,673	0,467	0,694	0,794	7,704	0,667	0,400	0,694	0,791	12,176	0,711	0,933	0,688	0,811	7,704	0,667	0,400	0,694	0,791
jazz	10,063	0,673	0,714	0,667	0,780	14,491	0,648	0,810	0,623	0,754	12,736	0,654	0,762	0,638	0,762	1,818	0,792	0,619	0,819	0,873	12,736	0,654	0,762	0,638	0,762
metal	12,176	0,635	0,588	0,641	0,758	10,572	0,679	0,706	0,676	0,790	11,094	0,660	0,647	0,662	0,777	0,761	0,918	0,941	0,915	0,952	11,094	0,660	0,647	0,662	0,777
pop	8,151	0,711	0,667	0,715	0,817	9,082	0,698	0,667	0,701	0,808	8,610	0,704	0,667	0,708	0,813	8,151	0,736	0,800	0,729	0,833	8,610	0,704	0,667	0,708	0,813
reggae	31,704	0,516	0,727	0,500	0,658	31,704	0,503	0,636	0,493	0,649	26,572	0,553	0,727	0,541	0,693	23,403	0,591	0,818	0,574	0,723	26,572	0,553	0,727	0,541	0,693
rock	22,642	0,547	0,684	0,529	0,673	24,962	0,528	0,684	0,507	0,654	23,403	0,541	0,684	0,521	0,667	20,434	0,541	0,579	0,536	0,673	23,403	0,541	0,684	0,521	0,667

Figure 11: MDC performance metrics w/zscore normalization

- MSE - Mean Squared Error
- acc - accuracy
- spec - specificity
- sens - sensitivity
- f - f-measure

We can deduce that the "Classical" genre is the easiest class to classify, as it has the best results overall, and the Mahalanobis distance is the metric for the Minimum Distance Classifier that overall presents better performance.

Here we presented the performance of the MDC classifier on the training and validation sets, but did not provide results on the testing set. We made this decision because we want to properly build and store the classifiers for later testing with a separate dataset, which would allow for a more rigorous evaluation of the performance.

We acknowledged that the performance on the training and validation sets can provide valuable insights into the generalization capability of the highlighted classifiers. By evaluating the performance on the training and validation sets, we can determine how well the classifiers are able to learn and generalize to new examples from the same dataset.

However, we also noted that the performance on the training and validation sets may not necessarily reflect the performance on a separate testing set or real-world data. Therefore, it is important for us to test the classifiers on an independent dataset to evaluate their true performance.

Overall, the presented results provide a valuable first step in evaluating the performance of the classifiers on the GTZAN dataset, and further testing and evaluation will be necessary to fully assess their performance.

In the next figures, we can see the Confusion Matrices for the eight most promising combinations of Class + Normalization + Distance Metric, which correspond to a binary minimum-distance classifier

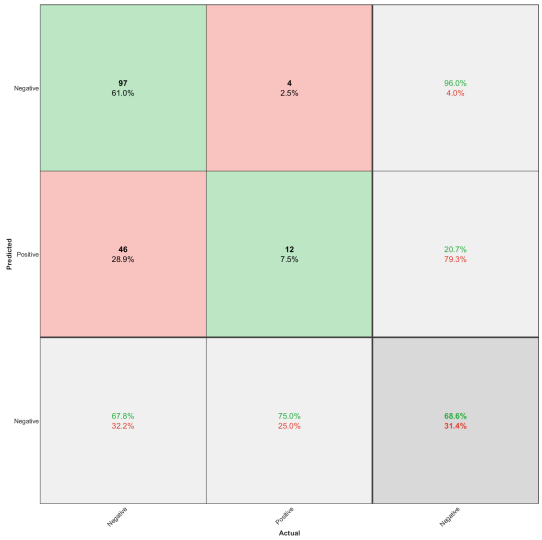


Figure 12: Blues + Unit Vector + Mahalanobis

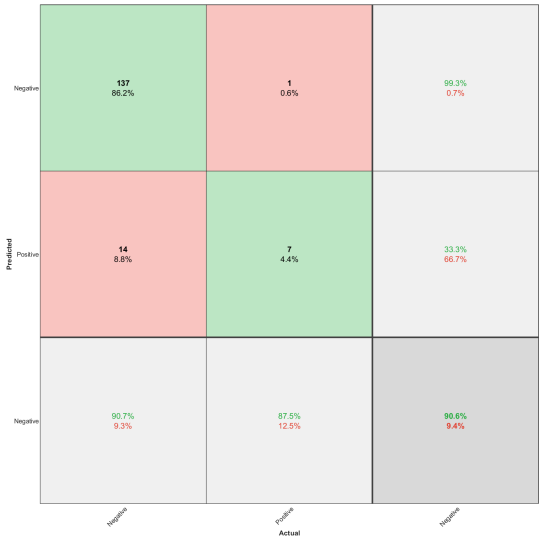


Figure 13: Classical + Range + CityBlock

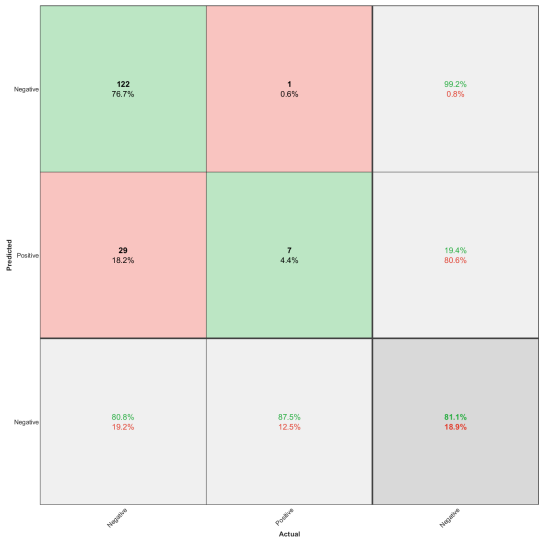


Figure 14: Classical + Range + Euclidean

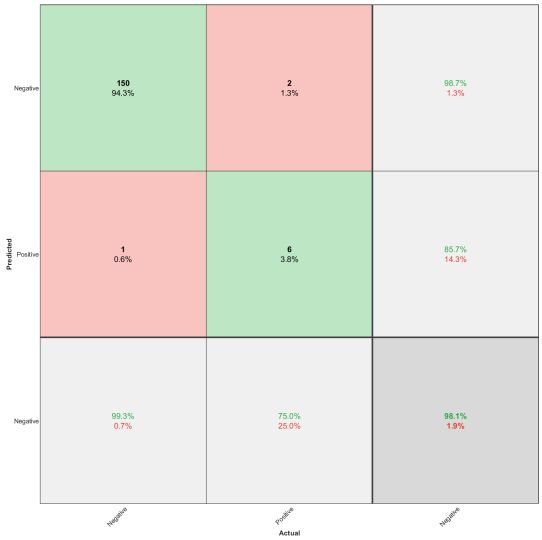


Figure 15: Classical + Range + Mahalanobis

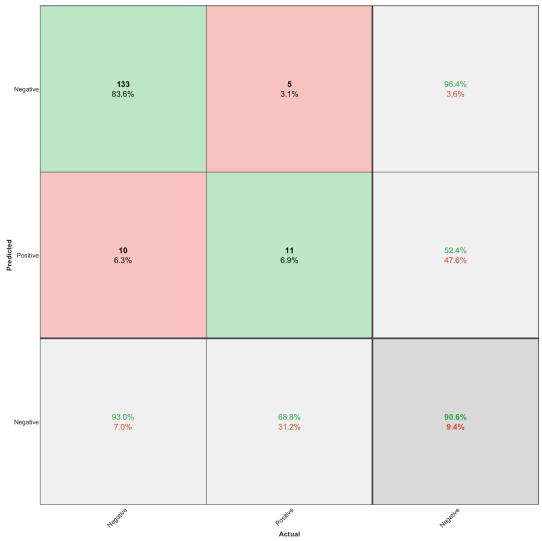


Figure 16: Classical + zscore + Minkowski

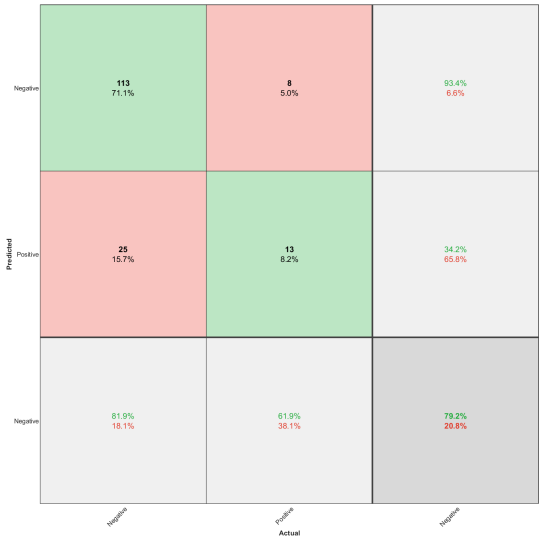


Figure 17: Jazz + zscore + Mahalanobis

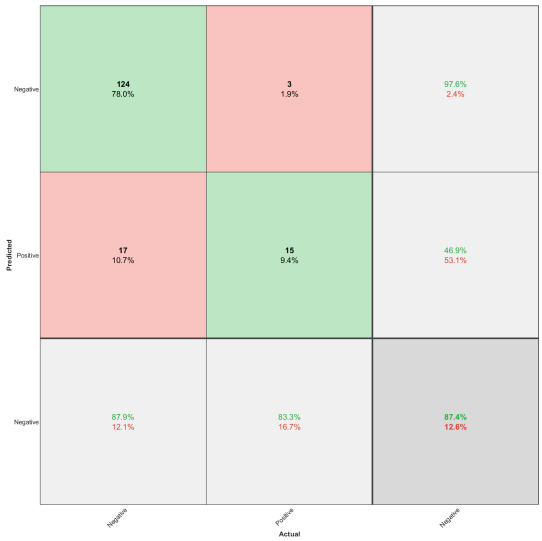


Figure 18: Metal + Unit Vector + Mahalanobis

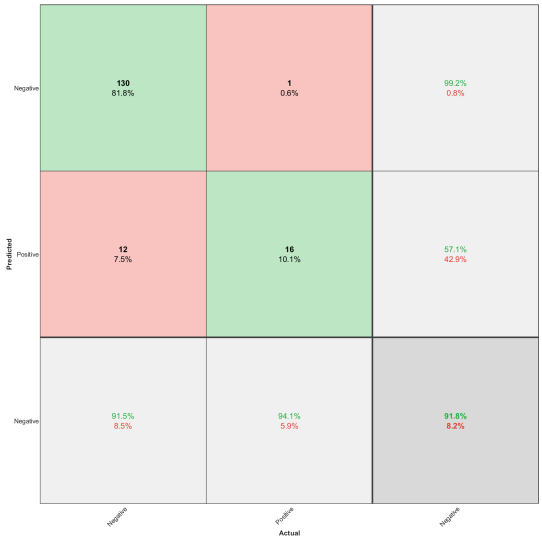


Figure 19: Metal + zscore + Mahalanobis

After analyzing the confusion matrices, it is evident that there is a significant class imbalance in the binary classification dataset. This imbalance is particularly noticeable when examining the percentages related to the true class (1) in the confusion matrix.

In binary classification, the confusion matrix is a table that summarizes the performance of a classification algorithm. It contains four possible outcomes: true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). The true positives and true negatives represent the correct classifications, while the false positives and false negatives represent the incorrect classifications.

When there is a class imbalance, it means that the number of samples in one class is significantly larger or smaller than in the other class. This can lead to poor performance of the classification algorithm because it may become biased towards the majority class.

In the context of the confusion matrix, the class imbalance is visible in the percentages related to the true class (1). For instance, if the dataset contains 90% negative class (0) samples and only 10% positive class (1) samples, then even if the classifier is performing poorly on the positive class, the percentage of true negatives will be high, which may mask the poor performance of the classifier on the positive class.

Therefore, when evaluating the performance of a binary classification algorithm, it is important to take into account the class imbalance in the dataset and use appropriate performance metrics that are not affected by the imbalance, such as precision, recall, F1-score, or the area under the ROC curve (AUC). Additionally, techniques such as resampling or using class weights can help to address the class imbalance and improve the performance of the classification algorithm on the minority class.

10 Conclusion

The main goal of this project was to develop classifiers for genre discrimination. Two scenarios were considered: binary classification (one-vs-all classification) and multiclass classification (classifying all genres together), where in this meta only the binary classification was explored.

In the first step, the data was split into two datasets, one for classifier development and the other for testing. The development dataset contained 80% of the data, and the remaining 20% was used for testing. Before proceeding with the classification phase, the data was normalized/standardized, and feature selection and dimensionality reduction techniques were applied to remove redundant and highly correlated features.

After applying feature selection and dimensionality reduction techniques, the pattern recognition algorithms were applied to the data, and the results were evaluated using appropriate performance metrics. For each genre, a small set of promising binary minimum-distance classifications were chosen and simulated over the test data.

Regarding the results of the experiment, it is evident that MDCs using Mahalanobis distance outperform other classifiers in terms of performance. This can be attributed to the use of a covariance matrix, which allows for auto-adaptation to the shape of each class's respective clusters. The Mahalanobis distance metric considers the correlations between variables, thereby providing more accurate distance measures between samples. This feature, combined with the ability to adapt to the shape of the data distribution, results in a highly discriminative and effective classifier. Overall, these results suggest that MDCs using Mahalanobis distance may be a superior choice for classification tasks, especially when the data distribution is complex and non-linear.

In conclusion, this project successfully developed classifiers for genre discrimination, and the results were promising. Overall, this project provides valuable insights into the use of pattern recognition techniques for music genre classification.

For future work, several opportunities exist to build upon the results of this study. Firstly, implementing Fisher Linear Discriminant Analysis (LDA) as an alternative classifier could improve the performance and accuracy of the model. Additionally, extending the current binary classification approach to a multiclass classification scheme could enhance the model's capacity to classify new samples accurately. Another area for improvement is the utilization of class balancing techniques to enhance the performance of binary classification. Lastly, the development of a graphical user interface for more comprehensive testing of the classifiers could help to evaluate and compare the performance of various classifiers in a more user-friendly manner.

References

[Barbancho et al.(2014)Barbancho, Tardón, and Barbancho] Ana M. Barbancho, Lorenzo J. Tardón, and Isabel Barbancho. Principal component analysis for music genre classification. *Journal of Vibration and Control*, 20(6):954–962, 2014. doi: 10.1177/1077546313476632.

[Dieleman and Schrauwen(2014)] Sander Dieleman and Benjamin Schrauwen. A deep learning approach to music genre classification. *ISMIR*, pages 649–654, 2014.

[Lee and Kim(2012)] Sung-Hyun Lee and Jang-Hoon Kim. A study on music genre classification using the mahalanobis distance. *Expert Systems with Applications*, 39(3):2987–2994, 2012. doi: 10.1016/j.eswa.2011.09.040.

[Nam and Lee(2013)] Juhan Nam and Jaejin Lee. Music genre classification using audio and lyrics features. *IEEE Transactions on Multimedia*, 15(3):537–548, 2013. doi: 10.1109/TMM.2013.2242967.

[Nanni and Lumini(2012)] Loris Nanni and Alessandra Lumini. A comparative study of music genre classification using various classifiers. *Expert Systems with Applications*, 39(1):30–37, 2012. doi: 10.1016/j.eswa.2011.06.029.

[Pons and Serra(2016)] Joan Pons and Xavier Serra. Experimenting with musically motivated convolutional neural networks. *ISMIR*, pages 357–363, 2016.

[Shiu and Wu(2011)] Shau-Yin Shiu and Chun-Min Wu. Ensemble methods for music genre classification. *Expert Systems with Applications*, 38(12):15210–15218, 2011. doi: 10.1016/j.eswa.2011.05.108.

[Tzanetakis and Cook(2002)] George Tzanetakis and Perry Cook. A comparison of musical genre classification methods. *IEEE Transactions on Audio, Speech, and Language Processing*, 10 (5):293–302, 2002. doi: 10.1109/TSA.2002.800560.

[Xie et al.(2021)Xie, Zheng, and Lv] Qianqian Xie, Yefeng Zheng, and Xueqiang Lv. Meta-learning for music genre recognition. *Expert Systems with Applications*, 173:114630, 2021. doi: 10.1016/j.eswa.2021.114630.

[Zhao and Zhang(2018)] Hong Zhao and Kaiqi Zhang. Music genre classification based on transfer learning from music similarity. *IEEE Transactions on Multimedia*, 20(8):2151–2161, 2018. doi: 10.1109/TMM.2018.2801978.

[Zhu et al.(2014)Zhu, Zhang, and Li] Xiaoyu Zhu, Xiaohua Zhang, and Xiaomin Li. Music genre classification using linear discriminant analysis. *Expert Systems with Applications*, 41(4): 1588–1595, 2014. doi: 10.1016/j.eswa.2013.09.052.