Music Genre Multiclass Classification

Machine Learning - Final Assignment Project

Marcelo José Ferrer
Master Program in Artificial Intelligence
University of Santiago de Compostela
Santiago de Compostela, A Coruña
marcelojose.ferrer@rai.usc.es

Maximiliano Hormazábal Lagos
Master Program in Artificial Intelligence
University of Santiago de Compostela
Santiago de Compostela, A Coruña
maximiliano.hormazabal@rai.usc.es

Mohamed Aymen Merchaoui
Master Program in Artificial Intelligence
University of Santiago de Compostela
Santiago de Compostela, A Coruña
mohamedaymen.merchaoui@rai.usc.es

Mutaz Sameer Abueisheh
Master Program in Artificial Intelligence
University of Santiago de Compostela
Santiago de Compostela, A Coruña
mutaz.abueisheh@rai.usc.es

ABSTRACT

This project aims to develop a Machine Learning system to solve a music genre classification of 11 different styles based on metadata, using different Artificial Intelligence classifiers and ensemble methods to increase performance. Pre-processing techniques such as: missing values handle, identifying outliers, eliminating duplicate rows and dimensionality reduction are applied in order to get the dataset for training models.

Other data processing techniques are carried out in relation to the model to be implemented like one hot encoding and normalization. Holdouts, k-folds (for cross validation) and hyperparameters experimentation have been applied.

The feature combination has a proven accuracy around 40% for the basic models and a general accuracy of 42% for the ensemble models.

Harmonic mean of the system's precision and recall values, or F-Score, has a value of 30% for the basic models and 34% for the ensembles.

CCS CONCEPTS

• Computing methodologies • Machine learning • Machine learning approaches • Neural networks

KEYWORDS

Music Classification, Supervised Learning Models, Preprocessing Techniques, Ensemble Methods.

1 Introduction

Music is one of the most ongoing entertainment activities in human life as its streaming industry is increasing dramatically with millions of songs each year. Big organizations like Apple, Amazon and Spotify are continuously spending resources in developing algorithms to get a larger customer base. With that kind of investing, many applications were created to adapt and improve the quality of the service. Some of these programs are music recommendation systems, which are useful to music streaming applications for customizing song selections per user preferences. These systems rely on many factors, being one of the most important identifying the music genre of a song.

The aim of this project is to analyze and evaluate the performance of different approaches for classification music genres by using 10 machine learning techniques (i.e. Artificial Neural Network, Support Vector Machine, Decision Tree, K-Nearest Neighbor, Multilayer Perceptron, Logistic Regression, Nearest centroid, Radius Neighbors, Gaussian Naive Bayes and Ridge Regression) and 7 ensembles techniques, (i.e., Majority Voting, Weighted Majority, Stacking, Bagging, Boosting ADA, Gradient Boost and Random Forest). A public Music Genre Classification dataset from Kaggle[1], an online community of data scientists and machine learning practitioners and subsidiary of Google LLC, was used to perform the analysis and development of the project. It contains around 18,000 instances and 16 different features, described in detail in the section description of the dataset section .

Most of the previous work done in this area regards using machine and deep learning to classify music genres based on processing audio data and extract some features. Recently, a few papers suggest and prove that using metadata of songs can obtain very good results. Moreover, using metadata is much cheaper than processing audio data in terms of training or pre-processing needed time.

A published paper [1] in iSemantics seminar by a group of researchers worked on a similar case with a larger input data size

[1] The dataset is available at the Kaggle repository URL: https://www.kaggle.com/datasets/purumalgi/music-genre-classification

(48,000 samples) from Spotify with 8 music genres. They selected 13 features out of 40 using the chi-square method and processing the information with models of Support Vector Machines, K-Nearest Neighbor and Naive Bayes. The paper concluded that the accuracy of classifying music genres based on metadata, compared to other expensive methods that depend on processing audio data, can be possible with precisions of 80 percent using Support Vector Machines. In many cases, the accuracy only reaches that level when the metadata was collected on high quality music and in the right way with the right techniques.

In another paper in the same seminar [2] the authors examined the effect of feature selection on music genres classification problem, where they used 4 different combinations of features based on highest score in the chi-square method. The difference in these combinations using the same criteria differ only in the number of selected features, in values of 12, 10, 8 and 5 out of 17 total possible attributes.

For this last case, a dataset of 5 genres with 6,000 instances each was used, resulting in 30,000 samples. The Support Vector Machine model was selected for classification and the accuracy of the model reached 80% for combinations which contain 12 and 10 features, and 76% with 8 features, while it's 67% for the 5 features combination. It's worth mentioning that the optimized parameters for SVM were C=100 and gamma=0.1 in the case of 12 features and gamma=1 in the case of 10 and 8 features.

The better metrics obtained in this research field were conducted by 3 researchers in Spain [3], where they found a new method for genre classification based on using high-level melodic features extracted directly from the audio signal of polyphonic music. They also experimented using low-level melodic and combining them. The used algorithms in this case were Support Vector Machines, Random Forest, K-Nearest Neighbours and Bayesian Network. The results show that the average accuracy for all algorithms for low-level melodic features was around 85%, for high-level features 90% and reaches 95% when combining both with a Random Forest classifier, suggesting that combining those features might have promising results.

Regarding the same dataset we are working on in this project, no formal papers have been found, but informal works have been discussed in Kaggle with accuracy values that not exceed 50%.[8]

The rest of the report has been organized in the following way:
Firstly, the development of the project is shown in section 2, starting with the description of the dataset, followed with the methodology subsection, where the selection of procedures, tools and techniques is illustrated along with a summary of the work done. In the next subsections the practical work starts with preprocessing, feature selection and experimentation.
The next section of results shows the outcomes from the development with tables and figures.
The final section of conclusions wrap up what has been discussed in the previous sections, stating also the limitations of the work and recommended future lines of investigation .

2 Development

2.1 Description of the database

In our dataset we have identified three main kinds of features: Mood (like danceability, energy, valence and tempo), Properties (such as instrumentalness, speechiness and loudness) and Context (like liveness and acousticness), in addition to other generic values such as duration. Table 1 shows the description as indicated in [4], column type and number of unique values (in the case of numerical columns, the values ranges will be used):

Column Name	Description	Column type
Artist Name	Names of the artists	Categorical, nominal
Track Name	Name of the songs	Categorical, nominal
Popularity	Song's popularity indicator	Numerical, continuous.
Danceability	Defines how suitable a piece is for dancing based on a combination of certain musical elements.	Numerical, continuous.
Energy	Variable that measure the energy that the song emanates.Typically, energetic pieces feel fast, loud, and boisterous. (Energy grows with the growth of the indicator)	Numerical, continuous.
Key	It is the identifier of the base musical note of the song	Categorical, nominal
Loudness	How much sound pressure a particular source is emitting at a given time. The measure is in decibels(dB) and the values averaged overall loudness in the track	Numerical, continuous.
Mode	Type of musical scale.	Categorical, nominal
Speechiness	Presence of spoken words in the theme (not sung)	Numerical, continuous.
acousticness	How acoustic the song is	Numerical, continuous.
Instrumentalness	Presence of vocal elements in the song. 1 means 100% instrumental	Numerical, continuous.

Liveness	Detects the presence of an audience in the recording.	Numerical, continuous.
Valence	Describe the musical positiveness conveyed by a track. Tracks with high valence sound more positive e.g., happy, cheerful, euphoric, while tracks with low valence sound more negative e.g., sad, depressed, angry	Numerical, continuous.
Tempo	The overall estimated tempo of a track is determined in beats per minute	Numerical, continuous.
Duration in min	Total time of the song in minutes	Numerical, continuous.
Time Signature	Convention used to specify how many beats are contained in each time measure.	Categorical, nominal
Genre Class	Type of Music, this is the output target column	Categorical, nominal

Table 1: Description of the dataset.

As for the Genre classification, in table 2 we can appreciate that we have 11 different types of genres. The number of occurrences of each one in the dataset is also shown, noting that there is a large difference in the counts between genres, where 28% of the data is focused only in one class.

Style	Class	Count of Class	Percentage
Rock	10	4949	28%
Indie	6	2587	14%
Pop	9	2524	14%
Metal	8	1854	10%
Hip Hop	5	1447	8%
Alternative	1	1373	8%
Blues	2	1272	7%
Acoustic/	0	625	3%
Folk			
Instrumental	7	576	3%
Bollywood	3	402	2%
Country	4	387	2%
	Total	**17996**	**100%**

Table 2: Proportion of categories for the target "Class".

In order to optimize model testing, the preprocessing section has been separated from the experimentation and main section. The "*preprocessing.ipynb*" notebook contains everything related to the previous treatment of the dataset.

2.2 Methodology

In order to predict the music genre, a number of steps, techniques and development methodologies should be performed to reach the final models. In this section, a summary of this process is done.

Firstly, some validations and tests should be performed before preparing the data for the training phase. In this preprocessing some changes have to be made based on data types, missing values, outliers, duplication and grouping. Upon all these steps, we should have a cleaned and dataset ready for next phases.

The next part is to reduce the dimensionality of the dataset and select the final features to be used in training. The output of this phase yields new datasets that will be tested later.

Once we obtain the cleaned dataset, data will be separated into input and output sets, then splitted to train and test using the hold out method and then the input sets will be normalized. It's important to mention that Random seed is set so we can obtain the same results every time. Finally the training data is indexed using cross validation technique.

Once the dataset is set, experimentation of the models begins to test for the best model and ensemble. Ten base models will be built (Artificial Neural Network, Decision Tree, K-Nearest Neighbors ,Support Vector Machine, Multi-layer Perceptron, Gaussian Naive Bayes, Logistic algorithms, Nearest Centroid, Radius Neighbors and Ridge Neighbors). Then the best parameters for each one will be determined and with these models seven ensembles will be created (Majority voting, Weighted Majority, Stacking, Bagging, Adapting Boosting, Gradient Boost and Random forest).

In the results section the models outcomes are presented along their final metrics.

For the metrics, as the problem in our project is a multiclass classification, Accuracy (as the data is imbalanced) and F score will be used. Accuracy is the percentage of times that the model has classified the target correctly, F-score is the harmonic mean of the system's precision and recall or sensitivity values. Sensitivity is the proportion of true positives that are correctly predicted by the model, while specificity is the proportion of true negatives that are correctly predicted by the model.

As for the team organization, a Trello project has been created to divide the workload in different manageable subtasks. Also a github solution has been set to allow the sharing and modification in parallel of the code by the members of the team.

Regarding the code, it was developed in Julia with Visual Studio, Jupyter and Docker, and it was structured to be as clean as possible. All necessary functions were declared on separated classes organized by purpose and stored in a utils folder to be imported on the corresponding notebooks.
Separated notebooks have been created for data preprocessing and testing, keeping the main notebook only with the final dataset and best models.
Data preprocessing notebook generates the dataset in a csv format and testing notebook saves the models in a serialized Julia ".jdl" structure. Both notebooks save in the dataset folder their output to be processed later in the main execution of the program.

2.3 Data preprocessing

This section focus on the steps to clean the principal dataset in order to provide to the models the best quality of data:

2.2.1 Set Data Types. First of all, it is crucial to transform the variables of our main dataset into numeric variables (Int, Float, etc) or categorical variables (string), to prevent us from interpreting the content of each column differently in the following analyses. Therefore columns like *"key", "mode", "Class", "time_signature"* have been cast as string despite the fact that inside their values could correspond to numerical values. After this the dataset's description became:

Feature	Type	Mean	Unique	N_Missing
Artist Name	String		9149	0
Track Name	String		15129	0
Popularity	Float64	445121		428
danceability	Float64	0.543433		0
energy	Float64	0.662777		0
key	String		11	2014
loudness	Float64	-791066		0
mode	String		2	0
speechiness	Float64	0.079707		0
acousticness	Float64	0.247082		0
instrumentalness	Float64	0.177562		4377
liveness	Float64	0.19617		0
valence	Float64	0.486208		0
tempo	Float64	122623		0
duration_in min/ms	Float64	2,0074E+10		0
time_signature	String		4	0
Class	String		11	0

Table 3: Description of the dataset according to its type of variable.

2.2.2 Missing Values. There are several ways to determine that the dataset has missing values. In the first instance, there are already columns that contain empty cells. Table 3 shows a description of the variables with information of interest such as: data type, average (for numerical variables), number of unique values (for categorical variables) and the number of missing values. It is important to take into account that forcing a cast in the columns helps to detect wrong types of variables and therefore allows us to identify missing values. That being said, the columns that initially contain NA values are: Popularity, Key and Instrumentalness. Specifically, the "key" variable contains more than 2000 lost data, which are equivalent in quantity to a new feature, as shown in the figure 1.

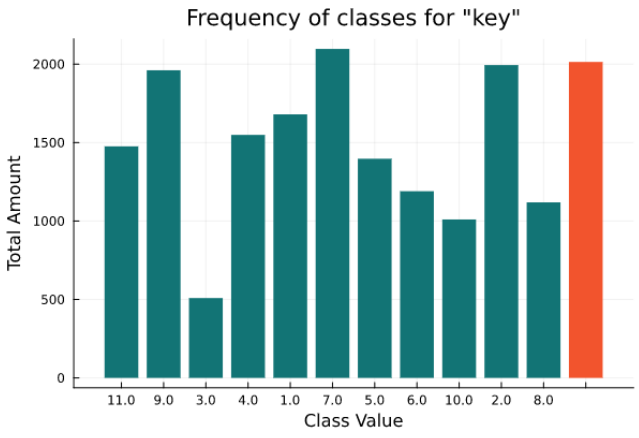

Figure 1: Amount of values for each category for feature key.

.2.2.3 Outlier Analysis. Numerical attributes are susceptible to having out-of-range data (outliers), which can interfere with the correct implementation of the models. In this analysis, the quartile method will be used to identify them. The choice of this method is due to the fact that it is one of the most used and it adjusts to the values of each of the columns without distribution assumptions.

The quartile method considers the existence of two types of out-of-range data:

$$IQR = Q_3 - Q_1 \quad (1)$$

$$Outlier = \pm 1.5 \times IQR \quad (2)$$

$$Extreme\ Outlier = \pm 3 \times IQR \quad (3)$$

Where (1) is the Interquartile Range, the distance of the values between the third and first quartile, are the range of values of the 50% of the column. With that indicator we calculate the limit of *Outliers* and *Extreme Outliers*.

Figures 2 and 3 show the box plots for some of the numerical attributes present in the dataset. Fittingly, the box plots are based on the quartile method to generate the visualization (in the same way that we will identify the outliers).

Features like "Popularity", "Danceability" and "Tempo" have centered ranges and clearly feature upper and/or lower outliers.

On the other hand attributes like "Loudness", "Instrumentalness", "Duration" present a large amount of data out of range.

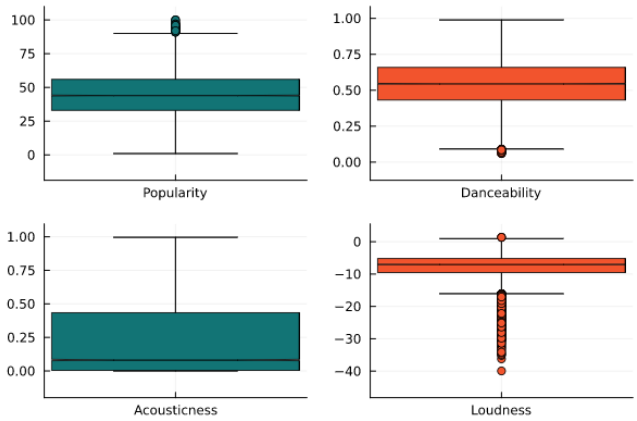

Figure 2: Boxplots for outliers representation part 1.

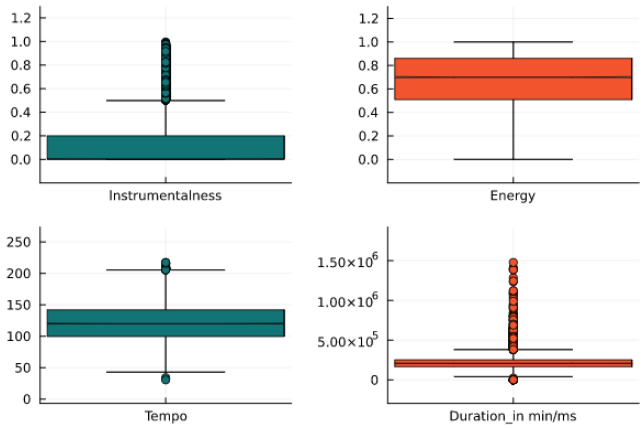

Figure 3: Boxplots for outliers representation part 2.

In summary, the detail of the amount of outlier or extreme outlier data is found in table 4, for which it can be seen that there are only three attributes with no out-of-range data of any kind.

Feature	Outliers	Extreme
Popularity	41	0
danceability	31	0
energy	0	0
loudness	637	181
speechiness	803	1231
acousticness	0	0
instrumentalness	1020	1388
liveness	642	314
valence	0	0
tempo	36	0
duration_in min/ms	3071	174

Table 4: Result of analysis of outliers and extreme outliers.

To continue with the analysis, it was decided not to intervene in the outliers, in order to preserve the variability of the columns as they were 1.5*IQR away on both sides. However, the specific cell will be eliminated for those data that are considered as extreme outliers and will be left as new lost data to be included later in the data imputation process.

The above is valid for most of the attributes, with the exception of the "Loudness" attribute, which in its scale presents both negative

and positive data and which will be later modified to contain only positive data. In this attribute, the outliers will also be left as missing values and the maximum of the column is going to be added to each value to move the range to the positive scale.

After the out of range analysis, the description of the dataset with the amount of missing values is the following:

Feature	Type	N_Missing
Artist Name	String	0
Track Name	String	0
Popularity	Float64	428
danceability	Float64	0
energy	Float64	0
key	String	2014
loudness	Float64	818
mode	String	0
speechiness	Float64	1231
acousticness	Float64	0
instrumentalness	Float64	5765
liveness	Float64	314
valence	Float64	0
tempo	Float64	0
duration_in min/ms	Float64	174
time_signature	String	0
Class	String	0

Table 5: Amount of missing values after outlier analysis.

.2.2.4 Missing Values Imputation. This section will be devoted to explaining each of the data imputation processes. Which ultimately corresponds to replacing the missing data by choosing a specific method.

.2.2.4.1 Imputation for numerical data

It is necessary to point out that the way in which missing values are replaced in numerical attributes differs from what would be done in a categorical column. In the case of numerical features, a correlation analysis was implemented. Notice that the linear relationship between two columns can help us to build the missing data of a column taking into account the one it has a linear relationship with.

To analyze this, the linear correlation has been measured in order to form a correlation matrix. Table 6 shows only those that exceed 0.5 (absolute value).

Feature	Loudness
energy	0.725534
acousticness	-0.52682

Table 6: Correlation greater than 0.5.

Figure 4 visually shows the linear relationship between these attributes. Therefore, the use of a linear regression can be considered to impute the "Loudness" data using the information contained in the "Energy" and/or "Acousticness" attributes. Having the following possibilities:

- Simple linear model with "Energy" or "Acousticness".
- Multiple models considering both variables.
- Variations of linear models with a polynomial function in one of the two attributes.
- Others

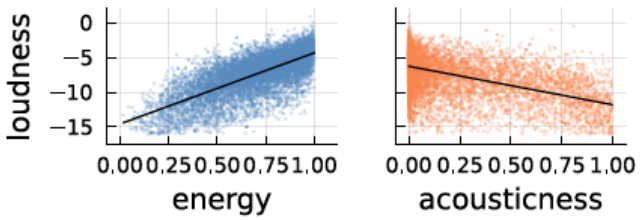

Figure 4: Correlation greater than 0.5.

On the other hand, figure 5 shows the difference between fitting the relation of “Energy” and “Loudness” with a cubic function instead of linear. That is why we are going to compare simple models and multiple models to get the better for “Loudness” imputation. The results are shown on table 7.

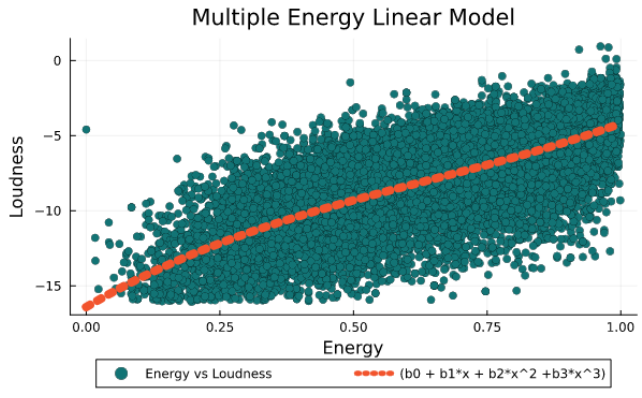

Figure 5: Multiple models using energy with cubic variation.

Model	R^2	R^2 adj
Loudness ~ Energy	0.5264	0.5264
Loudness ~ Energy + Acousticness	0.5271	0,5270
Loudness ~ Energy + Energy2 + Energy3	0.5279	0,5278

Table 7: Performance of linear models for Loudness Imputation.

It is important to notice that we have to check the results comparing R-Squared in simple models and R-Squared Adjusted in multiple models. As we see the multiple cubic model using Energy fits better than the others, we will use this model to predict loudness and fill the empty missing values spaces with this estimation. A residual analysis has been previously considered for the construction of these models to check if it fits the data correctly, however these results will not be investigated in depth since they are not part of the objective of this study.

The rest of the numerical columns were imputed using a KNN imputation model. They are:

- "Popularity"
- "speechiness"
- "instrumentalness"
- "liveness"
- "duration_in min/ms"

We select k=5 in a KNN Regressor (because this is a regression problem) and each column with missing values was imputed with the others numerical features.

After this process the last column that needs to be imputed is “key”.

2.2.4.1 Imputation. for categorical data

As we did in the previous step, we are going to use a model to predict the values of the “key” column, in this case a decision tree for classification instead of regression. Figure 6 shows new frequency values for categories on the “key” column next to the previous frequencies to compare where the ex-missing values are now. It is related to figure 1.

One of the pros of using models for imputation is that they can be more accurate than the mean or other methods. Applying this technique starts with an assumption about the ability to predict loss data from other information.

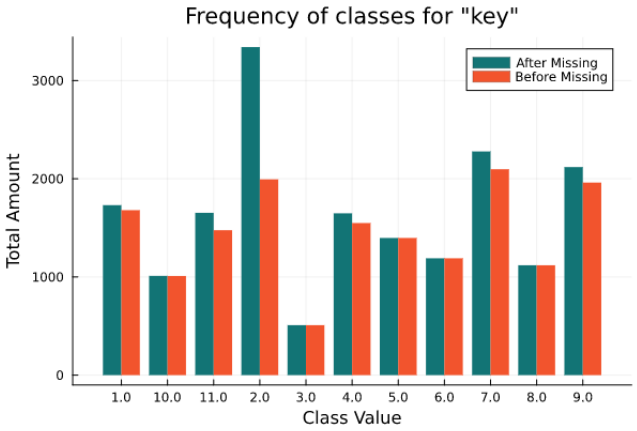

Figure 6: Comparison of values per category on feature “key”.

Finally, all missing values are already imputed.

2.2.3 Other Transformations. The feature “duration_in min/ms” has different types of values inside. It mixes minutes and milliseconds values. In order to avoid small and different values we transformed all columns in seconds renaming it as "duration_seconds".

As it can be seen in the previous description of data, the first and second columns are string columns with the name of song and artist respectively. As shown in table 3, the amount of unique values is too large to be classified and they are going to be eliminated from the data set.

2.2.4 Duplicate data. The goal of this preprocessing step is to check if we have duplicate rows. Starting with 17.996 rows, after checking duplicate data the new rows number is 17.923, that means 73 rows less.

2.2.5 Grouping classes. After analyzing the dimension of our problem, we check the frequency of levels on our target. Figure 7 shows the target’s amount of values.

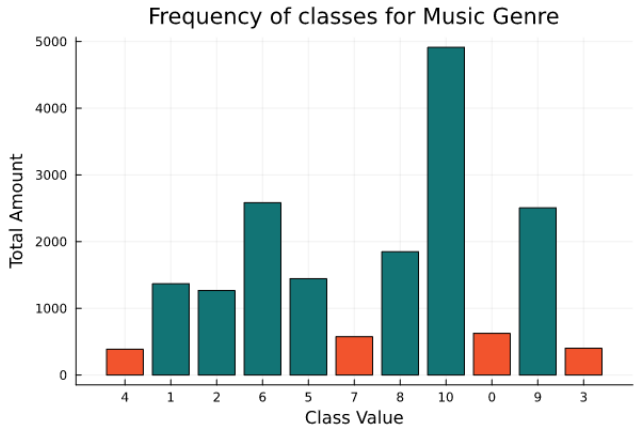

Figure 7: Amount of values per class target.

It is clearly seen that some classes have less presence than the others. To solve this, we decided to group these small classes: 0, 3, 4 and 7 (Acoustic/Folk, Bollywood, Country and Instrumental). If we consider the meaning of these classes (the music genre) all of them correspond to alternative genres far away from the mainstream market.

That is why we created a new class named "Other" with those classes. Figure 8 shows the new distribution of classes after grouping small music genres.

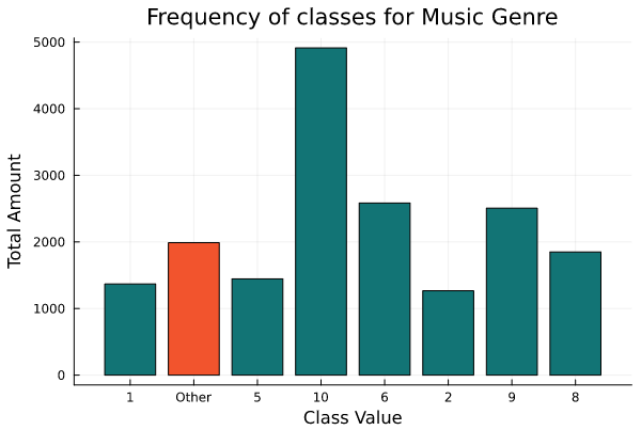

Figure 8: Amount of values per class target after grouping.

2.4 Data Preparation

After having the cleaned version of the dataset, the first step here is to separate the dataset to X (inputs) and Y (output) where the output is only the target column (Music Genre) and the training is the rest of columns. The target column data type has been converted to string.

Using the Holdout function, the data has been splitted into training and testing for each X and Y, where 30% of the data will be used as testing and 70% as training and as mentioned before, in case the code will be run again, we will have the same split due to using random seed! Moreover, Cross validation technique will be used to split between the training and validation dataset while training with 20 folds.

It's important to mention that only in the case of ANN, one hot encoding was implemented for the output column, as this algorithm only receives binary values.

2.5 Feature Selection

Finally it is time to reduce the dimension of the problem in order to use just crucial variables to predict the music genre. For this in the following step we explain the techniques tested and used to select how to simplify the dimensionality.

2.3.1 PCA. Principal component analysis was selected as a model to reduce the dimension maximizing the variability. After transforming to numeric the categorical values, normalizing, splitting and training the PCA model, we reduce the dimension as following:

Train Patterns: (14338, 27) → (14338, 18)

Test Patterns: (3585, 27) → (3585, 18)

2.3.2 Select K-Best. Another chosen way to reduce the amount of variables is using a scoring method to know which variables are more important. Sklearn provides the library "Select k-Best" and gives us the possibility to select the metric.

One of them is the Chi-Square test, this method contrasts the data by using contingency tables. The chi-square test assumes that the two variables are independent, therefore when *p-value>0.05* we can reject the null hypothesis and reach a conclusion of dependence [5].

Specifically, this Sklearn library allows generating a K-Top with the attributes that have a lower p-value with respect to our target.

In addition to this statistical test, the metrics f_classif and mutual_info_classif were used. The three processes were executed in parallel in order to choose the 10 most important columns according to the perspective of each method. Finally, those 30 columns were joined, keeping only the ones within that set.

The result was a set of 12 columns, where the three parallel processes agree on most of their column selection. For this reason, the columns to be used in subsequent processes are:

- Popularity
- Danceability
- Energy
- Loudness
- Acousticness
- Valence
- Tempo
- Duration_seconds
- Key1.0
- Time_signature3
- Speechiness
- Instrumentalness

2.6 Experimentation

As stated before, separate notebooks for testing have been created. These notebooks use functions that run a set of personalized tests for each one of the models and write in a disk file the one with the best metrics to be used later for the main notebook. Constants have been declared at the start of the execution to configure the sets of trials.

The basic sets of Artificial Neural Network have 14 different tests, Support Vector Machine has 18, Decision Tree uses 13 and K-Nearest Neighbor contains 14, Multilayer Perceptron has 15, Logistic Regression has 3 and as for Nearest centroid, Radius Neighbors, Gaussian Naive Bayes and Ridge Regression only one test was done. Through the constant configuration of the program, experimentation can be made on different combinations of Principal Component Analysis, Undersample, Oversample, Normalization, Hyperparameters, split of the Holdout and number of Folds.
Overall, more than 500 different assessment scenarios have been runned for the basic models.

As it was introduced in the first section of the document, a number of Pre-processing techniques have been taken into account and for each technique the set of tests have been run. As one can imagine, the amount of scenarios is too large to individually state each one in this section, but an example with the accuracy of the K-Nearest Neighbor can be made, where the initial value was 33% and after applying Feature Selection and Dimensionality Reduction reaches 39%. After the tuning of its holdouts, folds and hyperparameters the value reaches 40,15%.
On the other hand, other techniques such as Principal Component Analysis (39.1%), Under (33.36%) and Over (28.13%) samples reduced the model precision.

Regarding the ensembles, the sets of testing includes Majority Voting with 5 different basic models combinations, Weighted Majority with 3 different weighted basic models combinations, Stacking with 3 different basic models combinations, Bagging with 6 different basic models as parameters, Boosting ADA with 4 different basic models as parameters, Gradient Boost with 5 different decision trees parameters combinations and Random Forest with 10 different decision trees parameters combinations.
Overall, more than 35 different assessment scenarios have been runned for the ensemble models.

As for execution time, the test set only for the Artificial Neural Network can take more than 4 hours to complete, so testa were done separately first for the 4 more basic models (Artificial Neural Network, Support Vector Machine, Decision Tree and K-Nearest Neighbor) to get the best ones, replacing later the custom Artificial Neural Network for a Multilayer Perceptron of the Sklearn library to start experimenting with ensembles.

Overall the complete execution of the test set of 9 basic models (Artificial Neural Network deactivated) and 7 ensembles with the latest processed dataset take a total amount of one hour.

3 Results

After the preprocessing analysis and experimentation execution, the models for the current scenario and their metrics are shown in table 8. Although the final metrics have a relatively low score, the process stated in the document shows how the data was treated to get the best possible results. The metrics have increased by more than 10% compared to the initial dataset that was tested, but the information used for the research has only a limit to be improved. As stated in the bibliography, the best models existing for this particular dataset don't overcome the 50% of accuracy [8].

Model	Accuracy	Fscore
Artificial Neural Network	0.2379	0.2154
Support Vector Machine	0.418	0.3417
Decision Tree	0.3946	0.3004
K-Nearest Neighbor	0.4015	0.3194
Multi-layer Perceptron	0.4084	0.3352
Gaussian Naive Bayes	0.3647	0.2954
Logistic Regression	0.4067	0.325
Nearest centroid	0.3332	0.2878
Radius Neighbors	0.3165	0.1264
Ridge Regression	0.3918	0.2759
Majority voting	0.4197	0.343
Weighted Majority	0.4232	0.3431
Stacking	0.4208	0.3369
Bagging	0.4188	0.3453
Boosting ADA	0.3596	0.3189
Gradient Boost	0.4128	0.328
Random forest	0.4284	0.3469

Table 8: Metrics results of the best models.

Regarding the configuration of each of the models, table 9 shows the hyperparameters and model ensembles that have the best performance.

Best Model Parameters
Artificial Neural Network: maxEpochs = 1000, learningRate = 0.01, topology = (32, 24, 16, 8), validationRatio = 0.0, transfer function = sigmoid, repetition_training=5, max_epoch=20
Support Vector Machine: tol = 0.001, kernelGamma = 2, C = 4, kernel = rbf, shrinking = true, probability = true, coef0 = 0.0, kernelDegree = 3
Decision Tree: max_depth = 6, random_state = 1, splitter = best, criterion = gini, min_samples_split = 2
K-Nearest Neighbor: n_neighbors = 70, metric = minkowski, weights = uniform
Multi-layer Perceptron: maxEpochs = 1000, learningRate = 0.01, topology = (32, 24, 16, 8), validationRatio = 0.0, activation = identity
Gaussian Naive Bayes: Default parameters
Logistic Regression: max_iter = 1000, multi_class = multinomial
Nearest centroid: Default parameters
Radius Neighbors: Default parameters
Ridge Regression: Default parameters
Majority voting: Best SVM, Best KNN, Best MLP
Weighted Majority: Best SVM, Best DT, Best KNN, Best MLP, Best LR, Best RR, weight=(6, 2, 3, 5, 2, 1)
Stacking: Best SVM, Best DT, Best KNN, Best MLP, final_estimator=SVM
Bagging: Best SVM, n_estimators=10, max_samples=0.5
Boosting ADA: Best DT, algorithm = SAMME, random_state = 0, learning_rate = 1.0, n_estimators = 10
Gradient Boost: max_depth = 5, random_state = 0, learning_rate = 0.1, n_estimators = 10
Random forest: max_depth=11, n_estimators=250, max_features = sqrt

Table 9: Hyperparameters of the best models.

Regarding the models, a side note should be made about the Decision Tree. Although other basic models have better metrics, some ensembles like Random Forest only use this basic model and have metrics that overcome the ones of the other ensembles.

Majority Voting, Weighted Majority Voting, Stacking, Bagging and Random Forest were the 5 best performing ensembles, combining the four most used models along the practices (Support Vector Machine, K-Nearest Neighbor, Multilayer Perceptron and Decision Tree) with the inclusion of Logistic Regression and Ridge Regression only for the Weighted Majority ensemble.

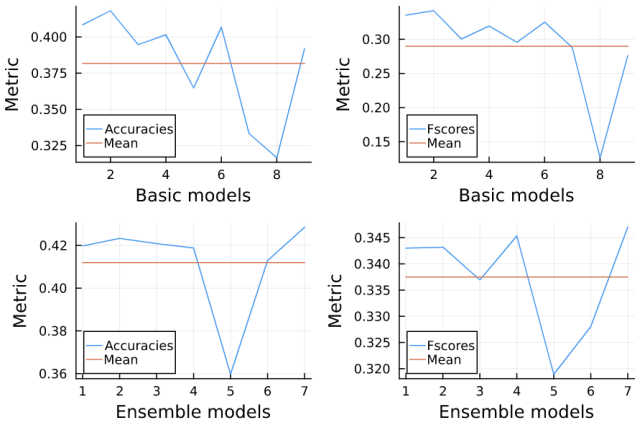

Figure 9: Results comparison between basic and ensemble models.

A final analysis of the metrics and differences between basic and ensemble models results can be seen in the plot of figure 9, which shows the Accuracy and Fscores for each best model and the mean statistic of each category. The difference achieved between the mean of basic and ensemble models has a contrast of 4% in favor of the ensemble models, proving the combination of different types is a better solution than using just one basic model.

4 Conclusions

Classifying the music genre of songs based on metadata instead of signal sounds is a challenging problem. Nevertheless, taking into consideration that there are companies producing this metadata information constantly for almost all songs they have in their catalog, solving this classification problem with metadata will have a huge effect in cost reduction of future works and studies.

To summarize the results of this project, after applying different types of models and ensembles with their best possible parameters the top metrics were reached in ensemble models such as Majority voting, Weighted Majority, Stacking, Bagging and Random Forest with values of over 42% of accuracy and 34% of F-score. Keeping in mind that the best accuracy from previous works for the same dataset was below 50%, the 42% is considered acceptable.

One of the main limitations we had in this project was the lack of target column values in the Kaggle test dataset, which made it impossible to calculate the metrics and check the models performance against it. To solve this situation, the train data was splitted for training and testing. Another issue with the information is that there is a sizable variation between the number of instances for each class, making the data imbalanced and hugely affecting the obtained results. Finally, the dataset size in terms of the amount of total records for all classes is much smaller than the ones used in other researchers' investigations.

For future work related to this topic it is recommended:

- Include data extracted from the melody and/or musical composition of the songs that are analyzed. What we have seen in the musical genre classification using melody features [3] shows that this can lead to much better results in terms of the performance of classification models.
- Collect a larger amount of data proportionally for each of the music styles selected as a target. Since in the publications mentioned in the bibliographical analysis the final number of records is much higher.
- Consult with experts and use other clustering methodologies (such as the use of clustering techniques) to effectively regroup objective categories for better-performing, targeted results

REFERENCES

[1] Rahardwika, D.S., Rachmawanto, E.H., Sari, C.A., Irawan, C., Kusumaningrum, D.P. and Trusthi, S.L., 2020, September. Comparison of SVM, KNN, and NB Classifier for Genre Music Classification based on Metadata. In 2020 international seminar on application for technology of information and communication (iSemantic) (pp. 12-16). IEEE.

[2] Rahardwika, D.S., Rachmawanto, E.H., Sari, C.A., Susanto, A., Mulyono, I.U.W., Astuti, E.Z. and Fahmi, A., 2020, September. Effect of feature selection on the accuracy of music genre classification using SVM classifier. In 2020 International Seminar on Application for Technology of Information and Communication (iSemantic) (pp. 7-11). IEEE.

[3] Salamon, J., Rocha, B. and Gómez, E., 2012, March. Musical genre classification using melody features extracted from polyphonic music signals. In 2012 ieee international conference on acoustics, speech and signal processing (icassp) (pp. 81-84). IEEE.

[4] Khan, F., Tarimer, I., Alwageed, H.S., Karadağ, B.C., Fayaz, M., Abdusalomov, A.B. and Cho, Y.I., 2022. Effect of Feature Selection on the Accuracy of Music Popularity Classification Using Machine Learning Algorithms. Electronics, 11(21), p.3518.

[5] McHugh, M. L. (2013). The chi-square test of independence. Biochemia medica, 23(2), 143-149.

[6] Mage. (2022). Music genre classification. DEV Community. Recuperado 12 de diciembre de 2022, de https://dev.to/mage_ai/music-genre-classification-2jmc

[7] Music Genre Classification. (2021, 7 agosto). Kaggle. https://www.kaggle.com/datasets/purumalgi/music-genre-classification

[8] Muhammedjaabir, J. (2021, 8 agosto). music genre clf. Kaggle. https://www.kaggle.com/code/muhammedjaabir/music-genre-clf