Universitat Politècnica de Catalunya

Machine Learning

Spotify Genre Classification

Eliya Tiram
eliya.tiram@estudiantat.upc.edu

Míriam Méndez
miriam.mendez.serrano@estudiantat.upc.edu

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Curs 2022/2023 Q2

Contents

1 Introduction

In this project we are going to classify different Spotify songs in their respective genre. It is worth to mention that one track doesn't have a single genre, but multiple. The full list of genres included in the dataset are rap, Techno, Techhouse, Trance, Psytrance, Dark Trap, DnB (drums and bass), Hardstyle, Underground Rap, Trap Metal, Emo, Rap, RnB, Pop and Hiphop.

To perform this genre classification we will use this data set provided by Kaggle. In order to build a multi-class classification model to accurately predict the kind of gender, will we be able to?

For this we will have to take into account certain musical audio measures that use Spotify API to classify their songs such as: loudness, tempo, instrumentalness, etc. That is, before starting, we will need a basic knowledge of this information, in order to understand it and make informed decisions for the modeling process.

For this reason we will first start with an introduction to Spotify API, then we will take a look to different related works of music genre classification. With all this information we will start the pre-processing, then we will perform some visualization to look if it is suitable for training our models. The models that we will train will be two linear and two non-linear.

2 Spotify API

The Spotify API contains a large amount of information about artists, audiobooks, playlists,... In this project we will focus on The Track's Audio Features, which are the same features of our Kaggle dataset.

There are 18 audio features:

- **dancability (float):** describes how suitable a track is for dancing. A value of 0.0 is least danceable and 1.0 is most danceable.
- **energy (float):** measure the perceptual intensity and activity in a track. A value of 0.0 is least energetic and 1.0 is most energetic.
- **key (integer):** maps a set of 11 **pitches** using the standard Pitch Class notation. If no key was detected, the value is -1. [1]
- **loudness (float):** averages the decibels **(dB)** of a track. Values typically range between -60 and 0 db.
- **mode (int):** reflects 1 if the track is in a major key and 0 if it is in minor key. [2]
- **speechiness (float):** measures the presence of spoken words in a track. Values above 2/3 are likely composed entirely of spoken words. Values between 1/3 and 2/3 indicate a combination of music and speech. Values below 0.33 typically represent music and other non-speech-like tracks.
- **acousticness (float):** measures the acoustic (sounds produced without electronic amplification or effects). A value of 0.0 is least acoustic and 1.0 is most acoustic.
- **instrumentalness (float):** measures the presence of vocal (rap or a spoken words track). Sound like "Ooh" are treated as instrumental. A value of 0.0 is least instrumental and 1.0 is most instrumental.

[1] This feature was treated as categorical. But we left it as an integer, since in the end we would need to one-hot encode it.

[2] This feature was treated as categorical. But we left it as an integer, since in the end we would need to one-hot encode it.

- **liveness (float):** measures the probability that a track was performed live. A value of 0.0 is non-live performance and 1.0 is a live recording.
- **valence (float):** measures the level of musical positiveness. A value of 0.0 is least valence (e.g. sad, depressed, angry) and 1.0 is most valence (e.g. happy, cheerful, euphoric).
- **tempo (float):** averages the **BPM** (beats per minute) of a track. A lowest value means a peace track and a highest value a speed track.
- **type (string):** the type of the object. All the values in our dataset should be *audio_features*.
- **id (string):** the Spotify ID for the track.
- **uri (string):** the Spotify URI for the track.
- **track_href (string):** a link to the Web API endpoint providing full details of the track.
- **analysis_url (string):** a URL to access the full audio analysis of this track. An access token is required to access this data.
- **duration_ms (integer):** measures the duration of the track in milliseconds.
- **time_signature (integer):** also known as meter, specifies how many beats are in each bar. Maos the time signature ranges from 3 to 7 indicating time signatures of "3/4", to "7/4". [3]

3 Related works

Spotify is not only a software that offers more than 100 million songs, it contains also a powerful AI to curate and design playlist for listeners, recommend music. As well as to classify the genres. There are many studies in genre classification using Spotify data.

Glenn McDonald proposed a website platform that aims to visualize and explore the vast landscape of music genres and subgenres. Currently, This genre study maps over 4,300 genres that Spotify recognizes.

This website was called Every Noise at Once utilizes a combination of music metadata, audio analysis, and clustering algorithms to organize and present genres in a hierarchical map. The project aggregates data from various sources, including Spotify, to create a comprehensive database of genres and subgenres.

Moreover, with the recent success of deep neural networks, a number of studies apply these techniques to speech and other forms of audio data. Representing audio as a spectrogram of a signal which captures both time and frequency information for input to neural networks . However in this project we will be focused on predict the genre from the Spotify API features, therefore we are not gonna go in deep with deep learning, instead with machine learning techniques.

Munkhbat and Ryu [1] proposed a synergistic approach that integrates machine learning algorithms such as KNN, Naïve Bayes, multi-layer perceptron and random forest for music-based stress relief. The obtained accuracy is 78.3%, and the precision and recall values were 80.5% and 54.8%, respectively.

4 Data exploration

The dataset contains a total of 768 instances with 9 variables, 42,305 songs/tracks, each containing 22 features of which 18 were already explained in Section 2. The four new features that weren't in the

[3] This feature was treated as categorical. But we left it as an integer, since in the end we would need to one-hot encode it.

API were: *genre*, the *title*, the *song_name* and the *Unnamed* of the track.

Genre will be target variable in this project, which consists of 15 categorical levels. The features *title* and *song_name* refers to name the track, besides whenever *song_name* is empty *title* has a value. The column called *Unnamed* it behaves as the index of the title feature.

Once we understand all our features, we eliminate the columns that are known to be irrelevant for our classification task: type (all have the audio_features value), uri, track_href, analysis_url, song_name, title and Unnamed. It is worth noting that the track ID column was not eliminated at this stage. Instead, it is kept to track any duplicated values during the pre-processing phase, as it is expected to be unique for each track. This allows for easier identification and handling of duplicates. The result of this stage was a dataset of 15 features (10 numerical and 5 categorical).

Once we finished this basic data cleaning, we take into account the following observations, before starting the pre-processing.

- **Unbalanced Datset:** There are two main genres: Dark Trap and Underground Rap and only one genre with relatively few instances classified in Pop. The majority of genres more less are balanced, except the ones mentioned in the beginning.

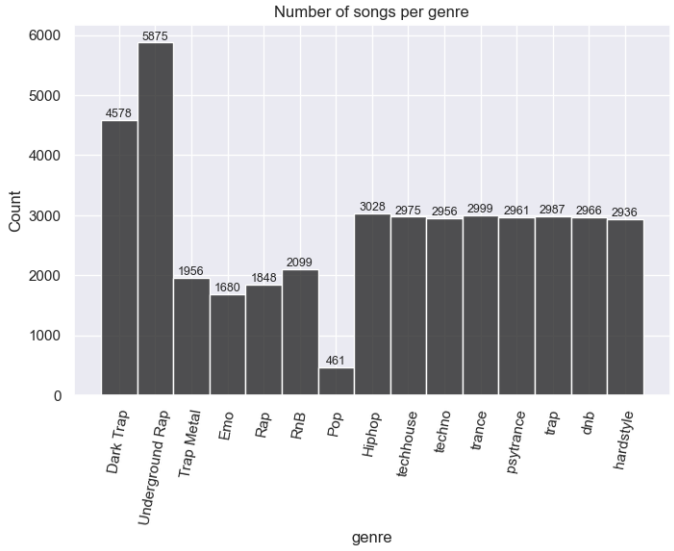
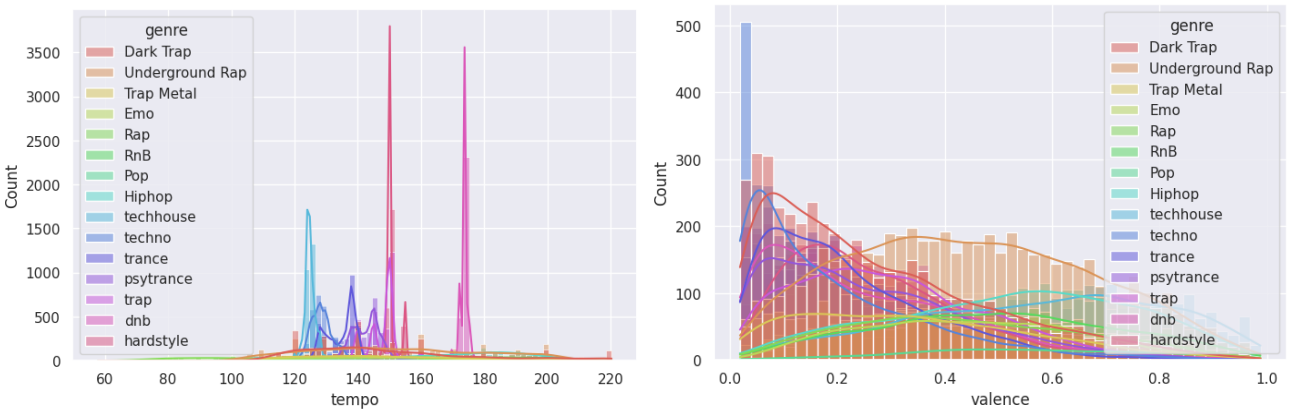


Figure 1: Target distribution

We know that when a class imbalance exists, the default behaviour of the ML model is to over-classify the majority class. It might lead to high accuracy in the training resulting a model is biased with a higher probability of misclassification in the minority classes. Therefore this, we will have to take into account in the pre-processing.

- **Relationship to target:** By analyzing the relationship of each variable with the target, we can begin to extract ideas about which variables are most related to music genres and in what way. When comparing the distributions with the target, we observe that there are notable differences in the distribution. In particular of *tempo* and *valence*.
  If we look at the tempo graph, we can see that the dnb, hardstyle and technohouse genres have very marked tempos, while the rest have more uniform tempos. Therefore, it gives us to understand that there could exist a direct relation of these genres, as for example it happens with the ballads that usually have a long tempo around 60pm. In the case of valence, we observe a considerable decrease in most genres, but in the underground rap, hiphop, RnB and technohause genres, they have a Gaussian shape, that is, we could use the following hypothesis, the higher the valence (positive track), the more likely it is to belong to these 4 genres mentioned above.



(a) histogram of tempo

(b) histogram of valence

Figure 2: Relationship with the genre target variable

4.1 Pre-processing

Duplicated values: Analyzing the duplicate values of our dataset, we realized that we were not facing a multinomial classification problem, but a multi-label classification problem.

This was found by computing the overall number of duplicates in the datset and the number of duplicate ids. We found 3218 duplicate rows and 6428 duplicate ids.

The duplicate rows in the dataset were removed, and then we analyzed the duplicated track IDs. Each track id had two or more different genders associated with them.

In Figure 3, the duplicated track IDs are visually represented, with a distinguishing feature applied.

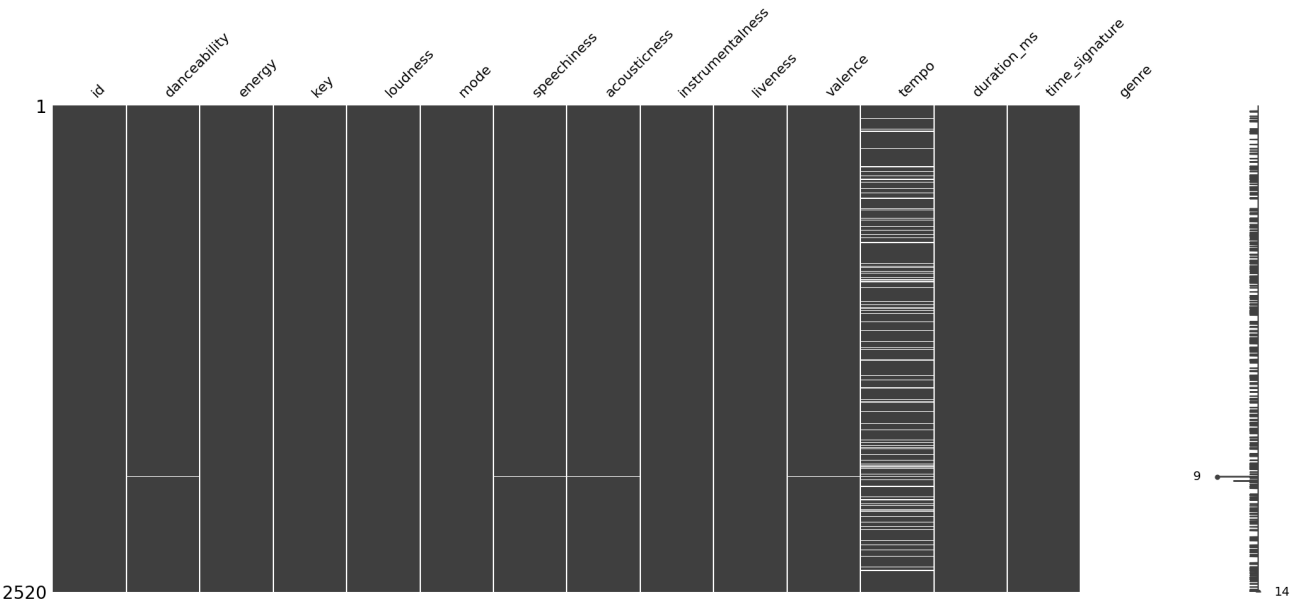


Figure 3: Differences found in the duplicated tracks id set. Tracks with unique values within the same ID are highlighted in grey, while tracks with non-unique values are shown in white.

Moreover, observed significant differences among certain features in the dataset. Notably, the tempo feature exhibited the most variation, with 319 discrepancies. The other features (execpt for genre) had less than two differences.

Furthermore, when examining individual tracks, we found that the track with ID *65ES1qwOB577ZnTkizMXJJ*

showed variations in danceability, speed, acousticness, valence, tempo, and genre features. The second track with notable differences was *697MjF1454XKvZmTuqkWmD*, which exhibited discrepancies in speechiness, valence, and tempo.

In order to solve the track duplication, we decided to do a group by id. In the case of the metrics, as we observed that the differences were minimal and we suspect that probably due to overflow during the computation of them (the error between them was less than 0.001), the first occurrence was established. In the case of the genere, as we decided to solve a muticlass-classification (due to the time limitation), if a track had multiple genres, we decided to stay with the minority class. For example, if a track has associated Pop (class with few instances) and Trap (class with many instances), we would be left with the one of the minority class, in this case Pop, and the association of Trap would be removed. In this way, we also reduce the unbalancing of our target.

Missing values: When running the function to remove NaN's, no change were seen. Additionally, we analyzed the summary statistics to identify any impossible values that could potentially represent missing values.

Surprisingly, we found that there were many zero values (Figure 4). It would not be wired that they were missing values hidden under the value 0 so we started to analyze them.

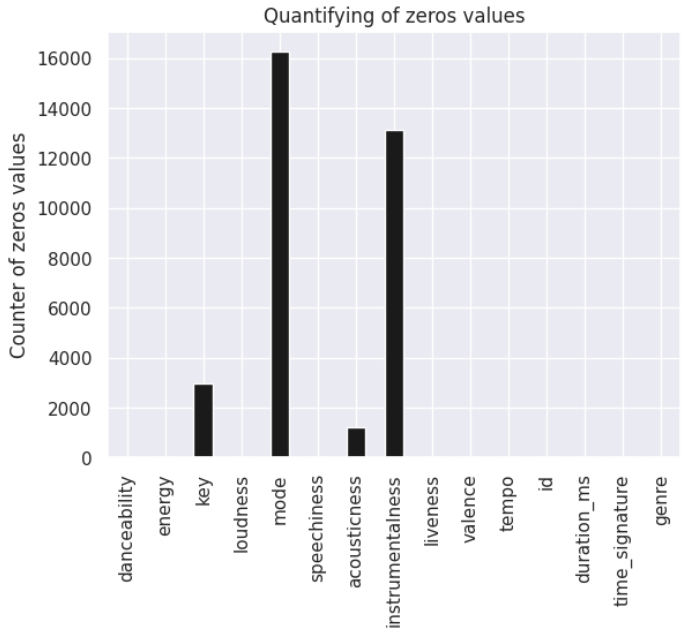


Figure 4: Counter plot of the zeros values

We argue the zeros values of *key* and *mode* features make sThis same influence can be visualized in the scatter plot, where the first 2 components have been visualized. ense, because both are categorical variables and 0 is one of their possible categories. The *acousticness* values too, since it could be tracks without electronic effects and amplifications. Regarding instrumentalness, we can observe a very large number, 36.61% have zeros values over the data. Nevertheless, we analyse deeper and most of these 0s were in underground rap genre,so we decided to leave them. Although we are not underground rap experts, we know that rap is characterized by rhyming words and usually accompanied by a base, so it would make sense for them to have 0 instrumentalness. In conclusion, after a comprehensive review, we found no missing values in the provided source.

Incoherent values: While looking the missing values, the summary statistics, we note that in *loudness* the maximum value of decibel was over the normal values suggested in the description of the features, indicating that the sound is louder than usual. Therefore, we decided to remove the samples over 0dB rather than do an imputation since there were only 0.46%. It is worth to mention that we do this

pre-process step, because if it is not an usual value this will be noise, and we don't want the model look up on unusual elements, since can perjudice the correct categorization of the data. Regarding the rest of the features we didn't encounter any incoherent values.

Outliers: in the presented boxplots (Figure 5), we can definitely see that there are some variables that have values that we *could* consider as outliers. The most prominent ones are loudness, speechiness, acousticness, liveness and duration_ms. We can see that the later three features mentioned are skewed to the right, while the fist one to the left (can be checked on Figure 6), where we saw that any feature dosen't follow the gaussian distribution. Although all of them have outliers, with the exception of valence, in incoherent values we saw that no one is out of place. Hence, we decide to leave all values without treating any.

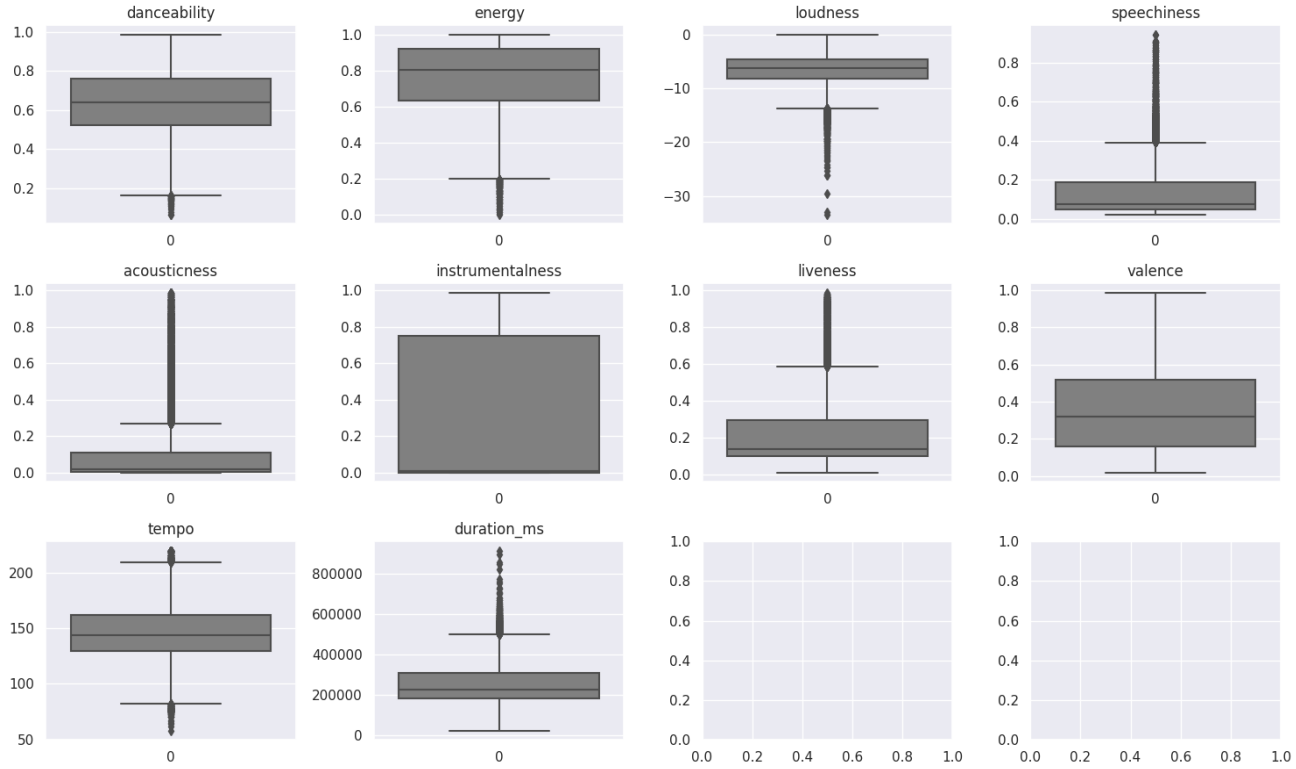


Figure 5: Bpxplots

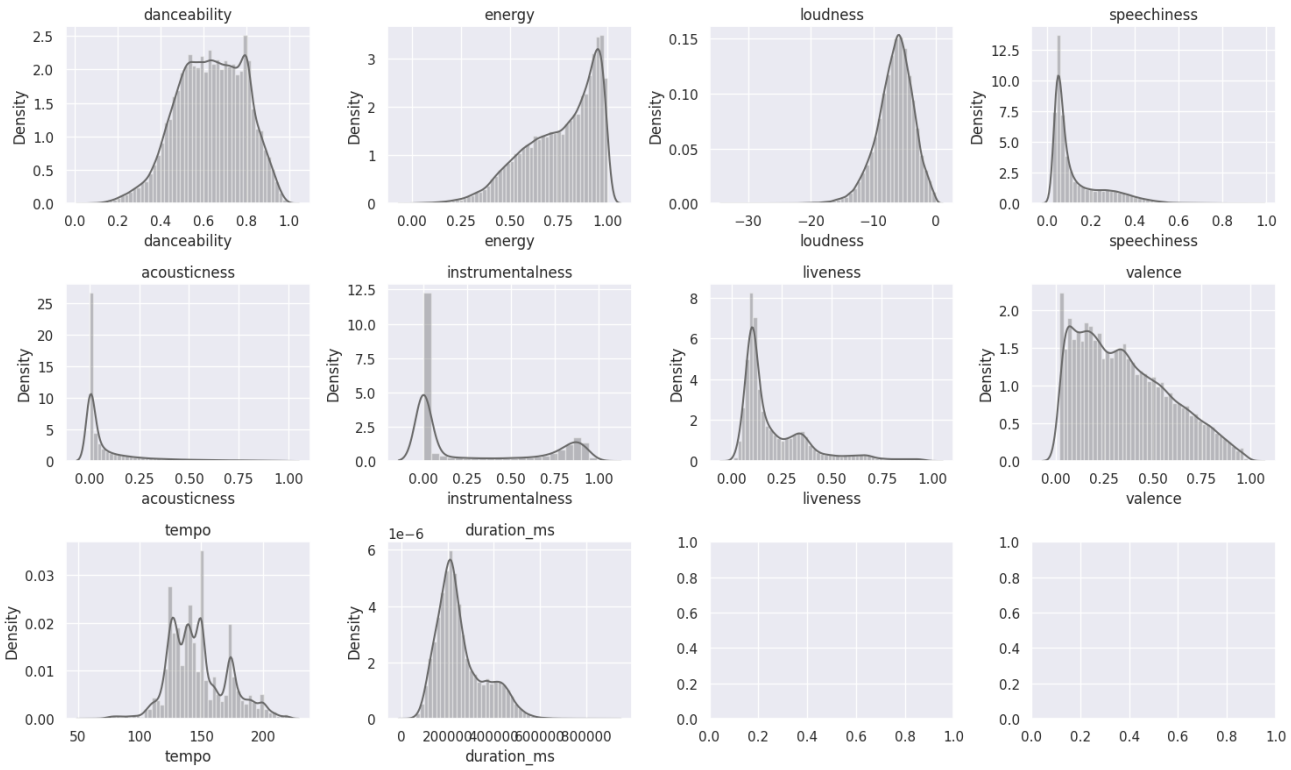


Figure 6: Distplots

Normalization : is a common requirement for many machine learning estimators, such as the RBF kernel of Support Vector Machines or the l1 and l2 regularizers of linear models. Therefore, according to the needs of each of the machine learning algorithms the data will be standarized.

Codification of variables: as mentioned earlier, we have a few variables that are binary/categorical. But we left these variables as they are (expressed in numeric values), as in the end we would need to encode them which would end with the same result.

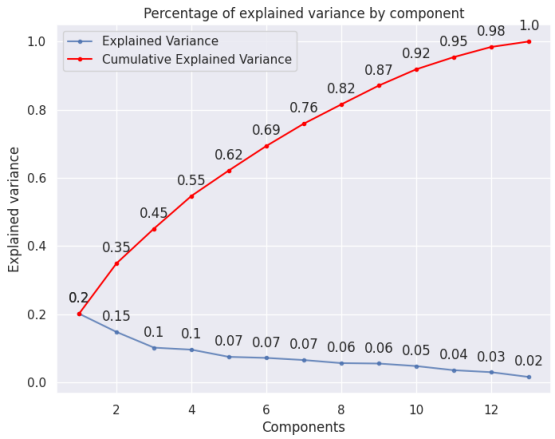
Feature elimination:finished the pre-processing process, the feature id (the id of the track) was deleted.

4.2 Visualization

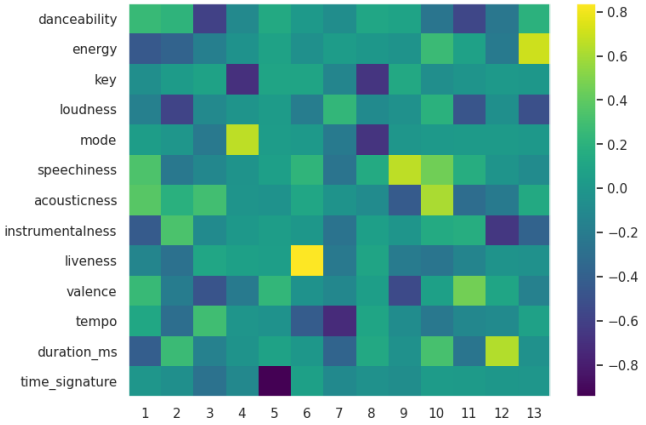
To make a better visualization of our data we have used the PCA. The PCA as it is based on the covariance matrix, we needed to standardize our data beforehand. This will cause the values of the variables to change to a mean of 0 and a standard deviation of 1.

In this visualization (Figure 7) you can see that with 10 components we could explain 92% of the variance of our data. That is, we could reduce our 14 dimensions to 10. This method is useful when we want to reduce the dimensionality.

Furthermore, once the principal components have been calculated, we can analyze the influence of the variables in each component with the heatmap plot. For example, the fifth component collects mostly time_signature information.



(a) Percentage of variance explained cumulatively



(b) Heatmap components

Figure 7: PCA

This same influence can be visualized in the scatter plot, where the first 2 components have been visualized.

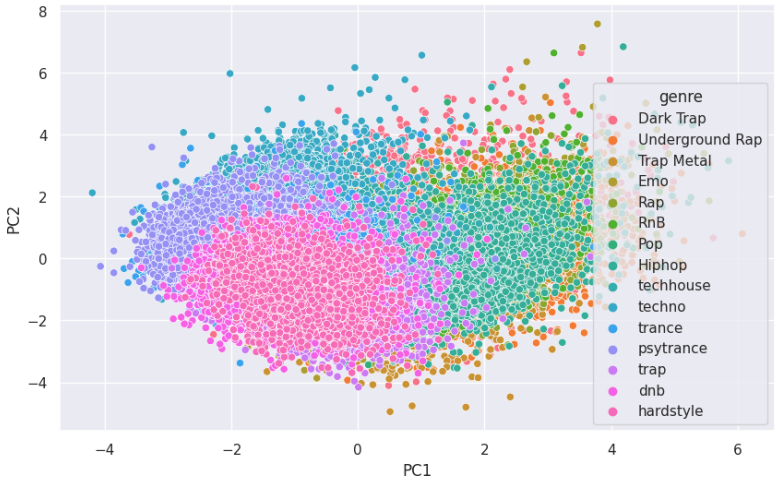


Figure 8: Scatter plot of PC1 y PC2

5 Modeling

5.1 Resampling protocol

The dataset partitioning performed was 1/3 (10714 instances) for the test and 2/3 (24997 instances) for the training. The latter set has been subjected to *5-fold cross validation* to select the optimal hyperparameters.

The procedure to adjust the hyperparameters has been performed with *RandomizedSearch*, except in KNN where *GridSearch* has been used, since the hyperparameter space was not too large. The difference between these two is that *GridSearchCV* exhaustively considers all parameter combinations, while *RandomizedSearchCV* works by testing a given number of given parameters which can be lists or distributions. These distributions can be very useful to us, for example when calculating the regularization parameter, which is used to counteract any bias in the sample and help the model to generalize correctly, avoiding overfitting the model to the training data.

In this search we have tried to fit the maximum number of hyperparameters and leave the minimum possible by default for experimental reasons. However, in cases where other hyperparameters accumulate many possible combinations, we have chosen to reduce the granularity.

5.2 Models

Having done all the pre-processing, our target was still unbalanced, even though we could still see the drastic difference in the target levels. To solve this problem we used the class weights in all the machine learning algorithms. This weights will influence the classification of the classes during the training phase. The whole purpose is to penalize the miss-classification made by the minority class by setting a higher class weight and at the same time reducing weight for the majority class.

There are several algorithms that we can use to train our model. In this project we implement two linear and two nonlinear classification algorithms.

For the linear models we have chosen the following two:

- **Logisitc regression** It was the first model we train since it is a simple and interpretable model that works well for linearly separable data. The optimal hyperparameters include a penalty ratio of 0.56 using elastic-net regularization with the saga solver, and a regularization strength of 0.6. These hyperparameters help control the trade-off between fitting the training data and preventing overfitting. The elastic-net regularization combines both L1 (Lasso) and L2 (Ridge) penalties, allowing for both feature selection and parameter shrinkage. The optimal hyperparameters suggest that the model is trying to balance both penalties to achieve a good performance. It's worth noting that logistic regression assumes linear separability, but the elastic-net regularization can help handle some degree of non-linearity.
- **Linear SVM:** It it also suitable for classification problems and we decided to implement it to compare the results of a linear kernel wuth the RBF kernel. The optimal hyperparameters include a regularization strength of 1.7 with an L2-type penalty and a squared hinge loss function. These hyperparameters indicate that the model is using a regularization term to control the complexity of the SVM's decision boundary. The squared hinge loss function is commonly used for linear SVMs and aims to maximize the margin between classes. The selected hyperparameters suggest a relatively higher regularization strength, indicating a focus on preventing overfitting and achieving better generalization.

For the nonlinear models we have chosen these two:

- **SVM with RBF kernel:** SVM (it is a suitable model for binary classification problems) with an RBF kernel (a very common nonlinear kernel for SVM). The optimal hyperparameters include a regularization strength of 1.97 and a gamma parameter of 1/number of attributes. The gamma parameter controls the shape of the decision boundary, with a smaller value indicating a more diffuse boundary. The provided gamma parameter, $\frac{1}{\#attributes}$), suggests a value that is dependent on the dataset's characteristics. The regularization strength helps control the complexity of the model, and the selected value indicates a preference for preventing overfitting.
- **Random Forest:** It is a nonparametric model suitable for flexible and easy-to-use classification. In addition, it takes into account that individual decision trees tend to overfit the training data, so it uses averaging to improve predictive accuracy and control overfitting. The optimal hyperparameters found have been: 645 trees, maximum depth 200, minimum 5 samples to split the node with minimum 2 samples for leaves, number of predictors to square root per split, control the size of sub-samples with max_samples (bootstrap = True).

6 Results

To evaluate the different models, we have based ourselves on the results of these three: cross validation, training and test.

Before giving any model as valid, we initially check the accuracy of these three models, i.e., that there is no anomaly, for example that the obtained values are not train accuracy $\geq$ CV accuracy $\geq$ test accuracy or train accuraccy is much higher than test accuracy (possible overfitting).

Modelos	Train accuracy	5-Fold CV accuracy	Test accuracy
Logistic regression	0.559	0.556	0.563
Linear SVM	0.54	0.53	0.544
SVM with RBF kernel	0.691	0.633	0.646
Random Forest	0.979	0.68	0.695

Table 1: CV, Train y Test accuracy of the models

Overall, we can see in Table 1 that the models have a relatively similar performance across the training accuracy, 5-fold cross-validation (CV), and test accuracy, with values around 0.5 - 0.6. Except for Random Forest, which the training test give us a very high accuracy on the training set, indicating a potential overfitting. However, it also demonstrates relatively good performance on the cross-validation and testing datasets, suggesting better generalization compared to the other models.

Moreover, it's worth noting that accuracy alone may not provide a complete picture of model performance, since it is only inform us the proportion of cases that were predicted correctly.

Therefore in Figure 9 there are displayed the precision, recall, and F1-score in order to have more insights, this is very recommended specially when treating with an imbalanced target.

The precision would inform us what fraction of predicted genres that are actually predicted correctly.How many songs are really in his genre?

The recall would inform us the number of correctly predicted genres out of its number of actual songs. How many songs the model identify correctly for each genre?

The F1-scores provides a balanced measure of a model's performance, it is the harmonic mean of precision and recall. Represents the overall performance of the model in terms of correctly identifying songs of that genre, balancing both precision and recall.

The support indicates the number of songs for each genre in the dataset.

	precision	recall	f1-score	support
Dark Trap	0.571	0.337	0.424	1168
Emo	0.381	0.528	0.442	472
Hiphop	0.452	0.388	0.418	793
Pop	0.131	0.438	0.202	146
Rap	0.354	0.469	0.404	537
RnB	0.402	0.314	0.353	576
Trap Metal	0.382	0.561	0.454	542
Underground Rap	0.501	0.322	0.392	1188
dnb	0.765	0.798	0.781	722
hardstyle	0.576	0.588	0.582	701
psytrance	0.786	0.861	0.822	796
techhouse	0.717	0.775	0.745	681
techno	0.793	0.808	0.801	825
trance	0.656	0.647	0.652	862
trap	0.665	0.672	0.669	705
accuracy			0.563	10714
macro avg	0.542	0.567	0.543	10714
weighted avg	0.579	0.563	0.562	10714

(a) Logistic Regression scores

	precision	recall	f1-score	support
Dark Trap	0.568	0.320	0.409	1168
Emo	0.332	0.536	0.410	472
Hiphop	0.452	0.393	0.421	793
Pop	0.092	0.110	0.100	146
Rap	0.322	0.462	0.380	537
RnB	0.372	0.316	0.342	576
Trap Metal	0.379	0.432	0.404	542
Underground Rap	0.515	0.323	0.397	1188
dnb	0.669	0.831	0.741	722
hardstyle	0.602	0.496	0.544	701
psytrance	0.740	0.834	0.784	796
techhouse	0.623	0.771	0.689	681
techno	0.735	0.807	0.769	825
trance	0.623	0.636	0.630	862
trap	0.594	0.679	0.634	705
accuracy			0.544	10714
macro avg	0.508	0.530	0.510	10714
weighted avg	0.549	0.544	0.536	10714

(b) Linear SVM scores

	precision	recall	f1-score	support
Dark Trap	0.610	0.544	0.575	1168
Emo	0.679	0.750	0.713	472
Hiphop	0.481	0.469	0.475	793
Pop	0.279	0.116	0.164	146
Rap	0.584	0.473	0.523	537
RnB	0.438	0.479	0.458	576
Trap Metal	0.485	0.491	0.488	542
Underground Rap	0.467	0.508	0.487	1188
dnb	0.951	0.992	0.971	722
hardstyle	0.839	0.886	0.862	701
psytrance	0.892	0.935	0.913	796
techhouse	0.862	0.880	0.871	681
techno	0.851	0.864	0.857	825
trance	0.820	0.838	0.829	862
trap	0.818	0.777	0.797	705
accuracy			0.695	10714
macro avg	0.670	0.667	0.665	10714
weighted avg	0.690	0.695	0.691	10714

(c) Random Forest scores

	precision	recall	f1-score	support
Dark Trap	0.616	0.431	0.507	1168
Emo	0.573	0.633	0.602	472
Hiphop	0.464	0.426	0.444	793
Pop	0.169	0.438	0.244	146
Rap	0.384	0.566	0.457	537
RnB	0.408	0.384	0.395	576
Trap Metal	0.424	0.565	0.484	542
Underground Rap	0.510	0.360	0.422	1188
dnb	0.911	0.932	0.921	722
hardstyle	0.777	0.833	0.804	701
psytrance	0.874	0.915	0.894	796
techhouse	0.807	0.828	0.817	681
techno	0.849	0.847	0.848	825
trance	0.784	0.799	0.791	862
trap	0.796	0.732	0.763	705
accuracy			0.646	10714
macro avg	0.623	0.646	0.626	10714
weighted avg	0.657	0.646	0.646	10714

(d) SVM with RBF kernel scores

Figure 9: Precision, Recall, F1-score and support in test set of all the models performed

Taking into account the figure above, these are the main characteristics that we found in each model:

Logistic Regression: The model shows varying performance across different genres. It achieves relatively higher precision, recall, and F1-score for genres like "dnb," "techno," and "trance," while it performs poorly for genres like "Pop" and "Emo." The weighted average F1-score is 0.562, indicating moderate overall performance.

Linear SVM: The Linear SVM model performs similarly to logistic regression but with slightly lower accuracy. It shows varied performance across different genres, with higher precision, recall, and F1-scores for genres like "dnb," "psytrance," and "techno." The weighted average F1-score is 0.536, indicating moderate overall performance.

SVM with RBF kernel: The SVM with RBF kernel model performs slightly better than logistic regression and linear SVM. It achieves higher precision, recall, and F1-scores for most genres compared to the other models. The model performs well for genres like "dnb," "hardstyle," "psytrance," and "techno." The weighted average F1-score is 0.646, indicating a relatively good overall performance.

Random Forest: The model shows varying performance across different genres. It achieves relatively higher precision, recall, and F1-score for genres like "dnb," "techno," and "trance," while it performs poorly for genres like "Pop" and "Emo." The weighted average F1-score is 0.562, indicating moderate overall performance.

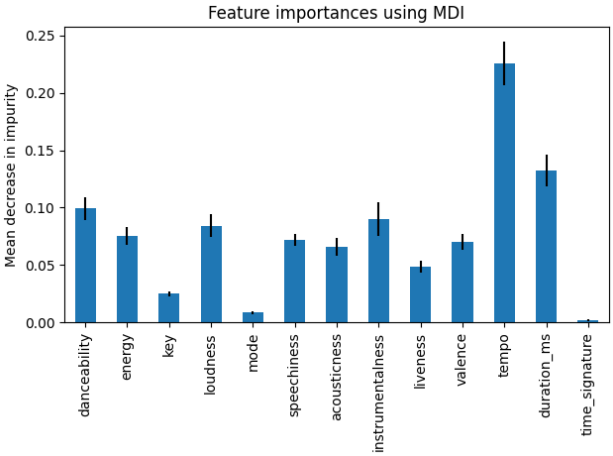
Overall, the nonlinear models, SVM with RBF kernel and Random Forest, outperform the linear models in terms of accuracy, precision, recall, and F1-score. Random Forest was the one, which achieves the highest precision, recall, and F1-scores. Although, Raandom Froest show some signs of overfitting based on the difference in accuracy between the training and validation, test sets, it still achieves the highest test accuracy among the models evaluated.

Additionally, Random Forest models are known for their ability to handle complex relationships and capture nonlinear patterns in data. They are robust against overfitting compared to individual decision trees due to ensemble averaging. The high number of trees in the Random Forest model (645 trees) and its maximum depth (200) indicate that the model has enough capacity to capture the complexity of the data while controlling overfitting.

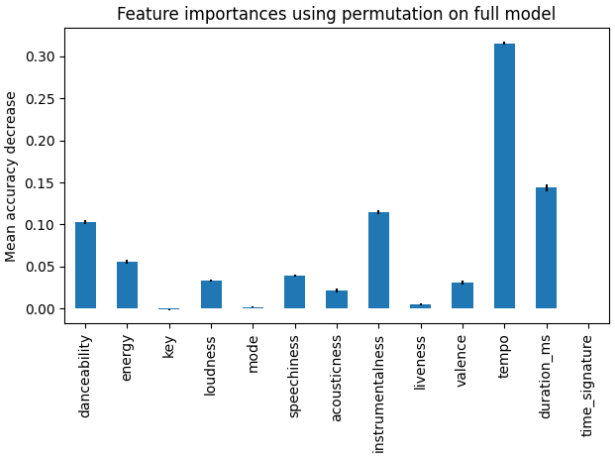
Therefore, we decided to chose Random Forest as a final model, in order to predict given a song its genre.

7 Final model

One of the main advantages of Random Forest is that it computes which are the most important features. In the case of this model, two different algorithms were executed: feature importance based on mean decrease in impurity and feature importance based on feature permutation.



(a) Feature importance based on mean decrease in impurity



(b) Feature importance based on feature permutation

Both algorithms more or less coincide in which are the most important features for this model: *duration_ms*, *instrumentalness*, *danceability*, and lastly the most significant variable to predict genres was *tempo*.

8 Conclusions

n this project, we aimed to classify Spotify music into different genres using machine learning models. We explored four different models: Logistic Regression, Linear SVM, SVM with RBF kernel, and Random Forest. After evaluating the models using various metrics, considering their performance on the training, cross-validation, and test sets, as well as their ability to generalize, we have reached the Random Forest was the best model in order to predict the genres of Spotify songs.

Although there were indications of overfitting in the Random Forest model due to the relatively higher

accuracy on the training set compared to the CV and test accuracy, the overall performance and generalization ability of the model were still favorable.

Moreover, we explore our final model. Features such as duration_ms, instrumentalness, danceability, and tempo were identified as influential in the classification process.

Overall, this project demonstrates the application of machine learning techniques in classifying music genres and highlights the effectiveness of the Random Forest model in this context.

9 Possible extensions and known limitation

The biggest limitation that we have found was the time, since some models, especially the nonlinear ones, which had a duration of 13 hours of training, were left overnight and were not evaluated until the next day.

Another limitation that we have found in relation to the execution time has been the Coolab framework, since all the resources were exhausted in other subjects of the master, however without using GPU, Coolab disconnected us in the middle of the execution.

Regarding the dataset, we have considered that it is a very complete dataset, it lets you explore several things during the pre-processing and the explanation of the variables is very complete, which is something to add because in most of the datasets that we had seen so far, it was scarce or poorly written.

If we had more time, we would have liked to play with the different existing techinques to balance the data in machine learning like undersampling, oversampling, and data synthesis and compare them with each other, along with our results using class weights. In particular, we would have liked to get to try SMOTE (Synthetic Minority Oversampling Thecnique), since it is a technique that we have never heard of before and is very popular in machine learning to balance data.

Regarding the multi-label classification, due to the lack of time, we needed to address the basics and if we had time to implement it, but with the setback of the delivery date, and the final exams, it has not been possible for us. However, it was started, and the data was already preprocessed to start training said algorithm. This is at the end of the code, we wanted to implement a ClassifierChain but time was running out on us and also if we reported it in the report, we wanted to show it well, not in any way. Although we have not implemented it, we have learned a lot about these types of algorithms (meta), since before trying to implement ClassifierChain we have searched for information on the subject.

In conclusion, we have learned a lot in carrying out the project and it has also helped us understand many concepts and it is gratifying to be able to apply many of the concepts and exercises carried out in theory and laboratory classes, since transferring academic knowledge to a real problem is not always possible.

References

[1] K.H. Munkhbat K. & Ryu. “Classifying songs to relieve stress using machine learning algorithms”. In: *Smart Innov. Syst. Technol* 157 (2020), pp. 411–417.