Università di Pisa

Text Analytics Project

Music Genres Identification from Song Lyrics

Text Analytics
AY 2022/2023

Simone Gallo
Lorenzo Pieri 578814
Cristian Stortoni 639184
Rocco Tiesi 636678

Introduction

Due to the ease and speed with which new content can be published, the growth of the music market (such as *Spotify*, *Youtube*, and *Amazon Music*) and the research field of music information retrieval (*Mir*) led to the construction of enormous music digital libraries, whose content continues to expand daily.
Users frequently rely on a choice that is influenced by music genre when exploring new content or cataloging well-known content. Because of the requirement for classification and the volume of data that must be processed daily, it is now crucial to effectively and automatically catalog both new and old information.

The objective of this project is to categorize a song's genre given its lyrics by selecting one of a variety of music genres (e.g., Pop, Rock, Hip-Pop and so on).
We utilize a dataset including hundreds of thousands of songs, each with its own lyrics and classified as belonging to one of the twelve genres in the dataset, to create the classification model and to carry out various operations on the analysis of song lyrics. In particular, in addition to categorizing each song's genre, a study of the emotions expressed in its lyrics was done in an effort to find a correlation or connection between one or more of the emotions and the musical genre.
Eventually, we labelled each song *"explicit"*, *"not explicit"* in relation to the fact that it contains bad words or not.

1 Data Understanding and Data Preparation

The initial step in our process are the understanding of the dataset and the searching for errors and issues inside it.
A total of 362,237 records compose the dataset, which has 5 attributes: *song* (the song's title), *year* (the year the song was released), *artist*, *genre*, and *lyrics*. The dataset was generated from the *MetroLyrics* website database. However, at the time this report was published, it was no longer accessible on *Kaggle*, most likely due to copyright issues.

A few details on the attributes: our analysis revealed that the attribute *year* is largely useless for conducting analysis on this data because some values are inconsistent with the song to which they refer. Our conclusion is that the year mentioned refers to the publication of the track's most recent version, which denotes a remastered version or something equivalent. This forces us to omit some analysis of this attribute during the course of our study. Additionally, there are various issues with the artist attribute, such as the same artist being written differently, typically because of typos, or having the first and last names inverted. Since our analysis primarily considers the musical genre and it is not necessary to have accurate values for the artist's name, this problem is resolved in the subsequent duplicate lyrics check.

Using the Python library *pandas*, we imported the *lyrics.csv* file into a dataframe to begin the dataset analysis. A fresh dataset with a total of 266,556 records is created after more than 90,000 records with *null* values in the lyrics attribute are removed using the library function. The language for each song lyric is extracted using another library called *langdetect*. The intention is to produce a dataset containing solely English-language songs so that the complete analysis can be performed on it. The outcome is a dataset of 238,230 songs.

Records with the music genres *"Other"* and *"Not Available"* were excluded from our research since, in our opinion, the songs in these two groups show a diversity of genres, artists, and lyrics that would complicate classification tasks. These are songs that, for reasons unknown to us, were kept unclassified despite the fact that they could have been previously categorized with other genres present in the dataset. Thus, we removed these records from our dataset obtaining a total of 216,671 songs.

1.1 Duplicated songs

Removing duplicates in the dataset was the next phase. Firstly, we were able to eliminate 13,962 songs with lyrics that were exactly the same, character by character, using the *drop duplicates* function of the *pandas* library.
The songs were then grouped according to title, resulting in a total of 20,968 titles that were potentially duplicates because they appeared more than once in the dataset. We needed to develop a technique to distinguish those songs that were equivalent lyrically even if the difference was only one character, such as

a punctuation mark, an abbreviation, a typo, or a different verse. Tokenizing the lyrics and running a comparison on them was our choice. The IDs of all the records with the same title were then gathered into a list for each record. After tokenizing the song lyrics, the first 10 tokens of each song that had potential duplicates were compared to the first 10 tokens of each song that might have been its duplicate (so, with its same title). The two songs were considered equivalent if at least 7 out of 10 tokens matched.
For a number of reasons, we chose to label these songs as duplicates using the mentioned method. Firstly, it is possible that the same song's lyrics appeared in the dataset with different characters or short abbreviations; secondly, as was discussed in the previous section, some songs appeared in the dataset with errors in the artist field. Last but not least, cover songs are frequently performed in genres other than those of the original song. Therefore, keeping track of these songs would make genre classification difficult. It should be noted that, in each of the three examples, the lyrics taken from *MetroLyrics* may not be an exact match but may instead give a shorter version of the lyrics or lyrics with different words or verses. After comparing the *year* value of each duplicate song to the *minimum year* value of all songs having the same tokens, only the song with the oldest year in the dataset was kept. The goal was to preserve either the original song (in the case of covers) or just one record (in the event of tracks by the same artist that were duplicated).

We present two instances to better illustrate the method's outcome. First and foremost, there are seven records with the title *"Honesty"* in the data. Only two records actually contained the same song—the track *"Honesty"* by American musician Billy Joel and its cover by Beyonce from a different year—despite the fact that seven titles featured the same title. Although, the song lyrics differed only slightly, our algorithm properly determined that these two were duplicates. The second example is the song *"Slow Love"* which appears on three records but is identical only on two of them. They feature the same year but a different artist.
In both situations, the technique effectively kept one record of the duplicates, in order to reduce confusion in the training of classification models.

The final outcome is a dataset containing 189,393 English songs without duplicates that was used for all our further analysis.

1.2 Class distribution

After removing the two categories *"Other"* and *"Not Available"*, we have a total of ten genres in our target variable "genre." As shown in Figure 1, there is a significant imbalance in the dataset, particularly in the class *"Rock"*, which makes up about 50% of the dataset's records. This imbalance issue was conclusively taken into account in subsequent analyses by using balancing techniques like *Random Undersampling*, *Random Oversampling* and *SMOTE*, comparing the outcomes with the analyses done on the original distribution.

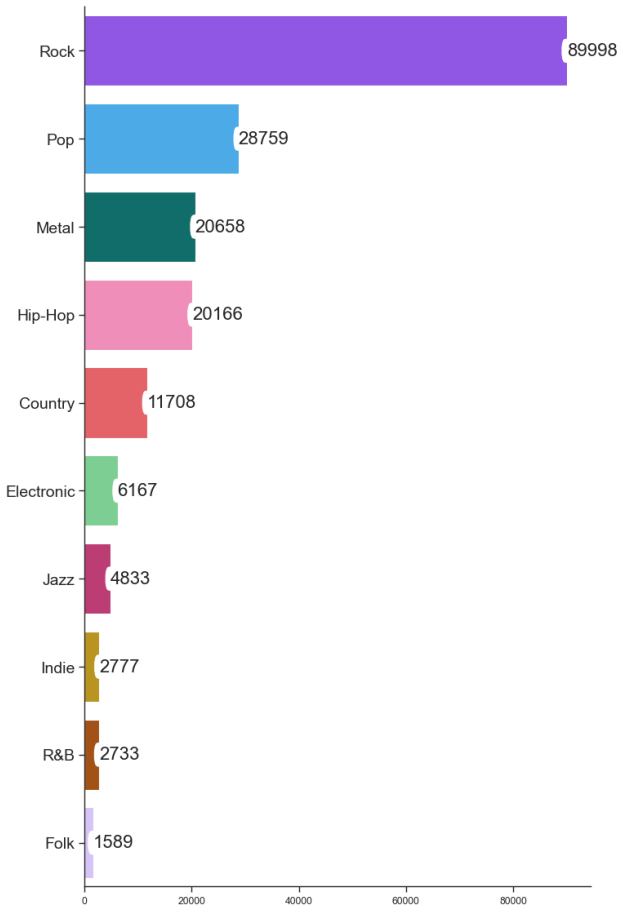

Figure 1: Distribution of *Genres*

1.3 Lexical Diversity

The examination of the lexical diversity of song lyrics constitutes one of the first analyses on the final dataset. We specifically looked at how long the lyrics are and how many different words are present in each text (i.e. the vocabulary). In most texts, using a wider vocabulary is considered a sign of superior articulation. Nevertheless, in music, this may not always be the case. Music repetition serves rhythmic and other cognitive functions, reducing the effectiveness of comparing the lexical diversity as it is. However, we may compare the vocabularies of songs from other genres, attempting to tie the number of unique words in a lyric to the complexity of its meaning and genre.

Figure 2 demonstrates how there is a correlation between the length of the lyrics and the average vocabulary used in each genre. We expected that *Hip-Hop* would have a larger vocabulary given the longer average length of its lyrics. But we don't believe this is the only factor contributing to the genre's lexical diversity given the presence of numerous artists, mostly from the rap style, who prominently use a lot of different words in their songs. It can be also observed that our predictions about *Pop* and *Electronic* music are largely accurate. Indeed, we can anticipate more phrases and words repetition in both genres, typically in choruses.

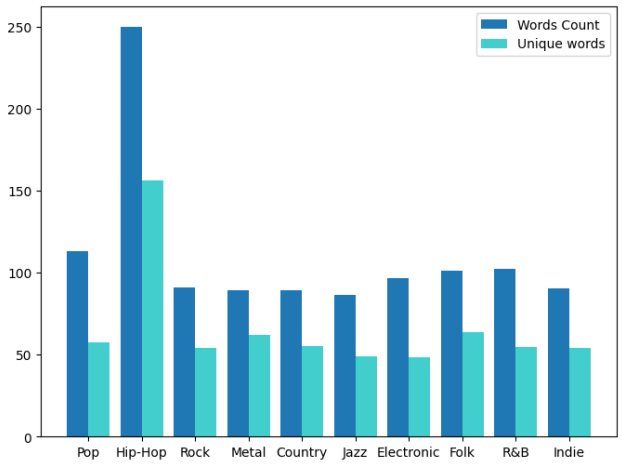

Figure 2: Words and Vocabulary for each genre

1.4 Text Preprocessing

Preprocessing techniques on the song lyrics are required before we can move further with the dataset analysis. According to a preliminary examination of the strings in the *"lyrics"* column, the lyrics contain a variety of special characters as well as certain *"labels"* that indicate information and song segments.
Therefore, we created the *clearLyrics* function, which introduces a new column in the dataframe with the cleaned lyrics of each song, ready to be utilized in the following analysis.
As a result, this function first eliminates unnecessary spaces, special tabulation characters, such as the new-line character (\n), punctuation (aside from apostrophes), and special tabulation characters. Following that, all musical data including the identification of a verse or chorus (identified in the dataset with the word inside square brackets, e.g. [Verse 1:]) and the number of repetitions of a verse (whether in the format x2, x3, x4 and so on or in the format 2x, 3x, etc.) were eliminated.

Due to the frequent use of several words that were not really helpful to the cause, we also considered providing an additional set of stopwords from scratch. For instance, it was chosen to eliminate *musical sounds*, such as the words "re", "oh", "la", "eh", and words or abbreviations in various forms that were not taken into account in the default collection of stopwords in the NLTK library. In addition, the choice to use tokenization on whitespaces is dictated by the desire to keep words such as "gonna" or "wanna" intact, since a lot of slang words are present in the lyrics.

After cleaning the text, analyses were performed on the frequency of unigrams, bigrams, trigrams, and collocations. The results don't reveal anything particularly fascinating, which is presumably because the texts in this dataset present some unique issues that are problematic. The analysis of *ngrams* was made challenging by the existence of odd characters and some annotations that are difficult to distinguish from the rest of the lyrics.

1.5 Clustering

We used clustering to look for easily distinguishable groups of songs and examine their distribution. To do this, the first step was to vectorize all the lyrics using

the *TfidVectorizer* function. It transformed each lyric into a *TF-IDF feature* matrix by taking into account each word in the lyrics and setting the *max feature* parameter to 3,000 and the range of ngrams to 1, i.e., considering individual words in the lyrics.
After obtaining this matrix, we used a dimensionality reduction technique to produce a two-dimensional representation of the values we had obtained. TruncatedSVD was the method used since it performs well on matrices of this kind and it gave us a new array of values used for performing clustering.

Next, we chose the K-Means clustering algorithm for representing the clusters. To determine the number of clusters, we used the Elbow approach and a calculation of the Silhouette Score, which led us to select 3 clusters. Figure 3 illustrates the clustering outcome, with a single point cloud corresponding to the full song dataset. This illustration shows us how the song lyrics of the ten genres are somewhat similar and how it is difficult to make a clear separation.

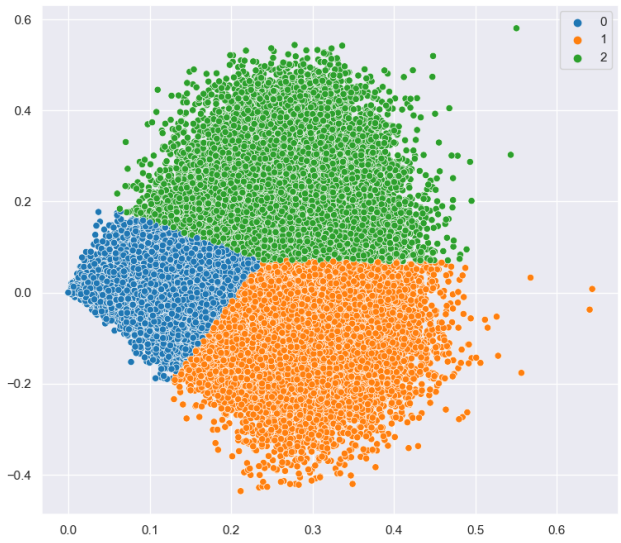

Figure 3: K-Means: Clusters

A more intriguing conclusion can be drawn by examining the distribution of genres within the three clusters (Figure 4). It is possible to see that the *Hip-Hop* genre is strongly represented in cluster 2 (the cluster in green in the figure above). This indicates that, while taking into account its lyrics, *Hip-Hop* is the only genre that somehow stands out from the rest.

Additional analysis include vectorizing the texts

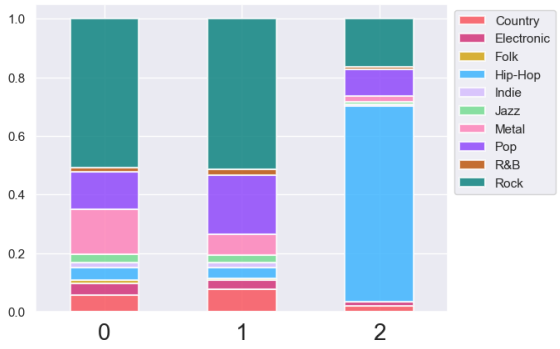

Figure 4: K-Means: Class Distribution

while taking bigrams into consideration, as well as a number of clusters equal to 4. The outcomes in these two instances are very similar to those just presented.

As a dimensionality reduction method, *UMAP* (Uniform Manifold Approximation and Projection) was tested in place of *TruncatedSVD*. However, because the results in this case were so unsatisfactory, it was decided to finish the clustering study using the initial method.

2 Classification

Following all of the preceding processes, we arrive at the heart of our analysis: classification. We immediately concentrated ourselves on the first issue with our dataset: imbalance. We were certain that the classification would be difficult because we had ten classes that were quite uneven in terms of the frequency of texts between them. The model that tends to classify all classes more evenly is preferred in a multiclass analysis with the same accuracy. This is why we opted to use sampling techniques like *Random Undersampling*, which also assisted us in reducing computational weight. In addition, oversampling techniques like *SMOTE* and *Random Oversampling* were used to evaluate how the models performed.

Therefore, we started to try a very wide range of combinations of vectorization, sampling techniques and basic classifiers to see which direction to choose.
As vectorization we tried: *Count Vectorial* considering bigrams/trigrams; *TfidVectorizer*; *Word2vec*.
Overall, during all the various tests we tried these Classifiers: *Decision Tree*; *Logistic Regression*; *SVM* (Linear and Non-Linear); *NaiveBayes*; *Random Forest*.
As Advanced Classifiers we used: *Artificial Neural Network*; *Recurrent Neural Network*; *biLSTM*.

We also attempted nearly all of the analyses with Part-Of-Speech (POS). On the latter, we received results that were somewhat inferior but more or less identical, therefore we did not take them into further consideration during the analysis.
Between all the results we obtain (Figure 5), we need to find a way to exclude various combinations and classifiers. Therefore, we decide to take into account the model's performance both in terms of computational time and also in terms of the various analysis metrics (F1-Score, Accuracy, etc). As a result, we will only consider the most intriguing ones in the report.

The best classifier for the original datset is the ANN with an accuracy of 0.62, as indicated in Table 1, but this data is misleading. Because, as we predicted, the effect of the imbalance results in classes that classify relatively well, such as *"Rock"* and *"Hip-Hop"* with 0.79 accuracy. Whereas others that do not, such as *"Indie"* and *"Folk"*.
However, in the undersampled dataset, which the best classifier is the Non Linear SVM, we obtain a high score for *Hip-Hop* while losing accuracy on *Rock*, our most frequent class. This means that *Hip-Hop* is quite recognisable from the model, regardless of the amount of data; not surprisingly, it is the genre with the longest lyrics, as seen in Figure 2.
Although there are better combinations than undersampling in Figure 5, we favour undersampling over sampling at similar accuracy since the latter forces the nature of our dataset.

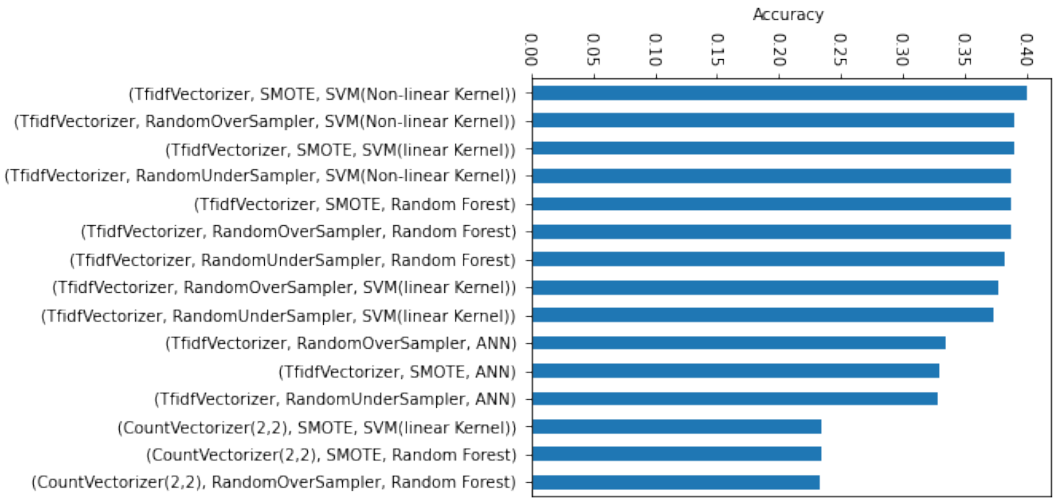

Figure 5: Best 15 classifiers

ANN		**NonLinear SVM**	
Dataset	*Original*	**Dataset**	*Undersampled*
Vectorization	*TfidfVect*	**Vectorization**	*TfidfVect*
Accuracy	0,62	**Accuracy**	0,39
WeightedAvg	0,56	**WeightedAvg**	0,39
Genres	**F1 Score**	**Genres**	**F1 Score**
Country	0,33	*Country*	0,40
Electronic	0,02	*Electronic*	0,28
Folk	0,00	*Folk*	0,36
Hip-Hop	0,79	*Hip-Hop*	0,75
Indie	0,00	*Indie*	0,31
Jazz	0,13	*Jazz*	0,33
Metal	0,33	*Metal*	0,62
Pop	0,58	*Pop*	0,23
R&B	0,33	*R&B*	0,28
Rock	0,72	*Rock*	0,19

Table 1: Best 2 classifiers using Tfidfvectorizer

RandomForest		**RandomForest**	
Dataset	*RandomOver*	**Dataset**	*SMOTE*
Vectorization	*Word2Vec*	**Vectorization**	*Word2Vec*
Accuracy	0,97	**Accuracy**	0,87
WeightedAvg	0,98	**WeightedAvg**	0,87
Genres	**F1 Score**	**Genres**	**F1 Score**
Country	1,00	*Country*	0,87
Electronic	1,00	*Electronic*	0,95
Folk	1,00	*Folk*	1,00
Hip-Hop	0,98	*Hip-Hop*	0,79
Indie	1,00	*Indie*	0,99
Jazz	1,00	*Jazz*	0,97
Metal	0,98	*Metal*	0,79
Pop	0,92	*Pop*	0,71
R&B	1,00	*R&B*	0,99
Rock	0,89	*Rock*	0,68

Table 2: Best 2 classifiers using Word2Vec

To boost the performance of our models, we try to apply the Word2Vec embedding technique, which yields no significant results in both the original and undersampled datasets. Except when using Random Oversampling and SMOTE on an oversampled dataset. As seen in Table 2, employing the Random Forest as a classifier yields an accuracy of 0.97 with the former and 0.87 with the latter. However, as previously stated, these results were acquired by employing oversampling approaches that could force the clarity of the text in the textual context, yielding inconsequential results.

2.1 BERT

We attempted to implement a Neural Network model for classification using BERT (*Bidirectional Encoder Representations from Transformers*). Few tests were run due to the scale of our dataset and the significant resources and time needed for model creation and evaluation using BERT, even using a GPU to train the model.

We used an 80/10/10 split to divide the dataset into training, validation, and test sets for our analysis. After cleaning the texts, we performed *Random Undersampling* on the data. In order to lower the computational load of the model and balancing the training set, so that all ten genres had the same amount of entries.
Using a pre-trained tokenizer, we transform all the texts using the *BertTokenizer* class from *Hugging Face* into the format required by BERT. Due to the characteristics of BERT (that works with a maximum of 512 tokens) and our data, we set a *max lenght* parameter for the tokenizer equal to 400.

We fixed a *dropout* value equal to 0.3 and a number of *epochs* of 5 for the first Neural Network model. A linear layer with a *ReLU* activation function receives the embedding vector of tokens and then we have a vector of size 10 at the end of the linear layer, one for each of the 10 genres. The model produced unsatisfactory results, returning an accuracy on the test set of 0.327. However, we saw that the loss and the accuracy on the validation set were becoming better with each passing epoch, so we chose to implement a new model.

For this new model we decided to change the *dropout* to 0.4 and to double the number of epochs. Even though accuracy and loss during training improved epoch after epoch, the loss over validation reached a peak after a few epochs to then decrease after that. As a result, the accuracy on the test set is about the same as the prior one (0.326). Therefore, we concluded that adding more epochs will probably not help enhance the outcome.

We reach the conclusion that additional tests should be conducted with more data but, due to our hardware limitations, we were unable to develop new models.

3 Emotion Analysis

When expressing emotions, music has great power. While the various rhythms, sounds, and effects used in music can temporarily alter our feelings, the lyrics are also an essential component.
To determine whether there is any relationship between musical genre and emotions, we examined the lyrics of each song to discover the emotions associated with each word. To accomplish this, the *NRCLex* library and its lexicon were employed, which assigned to each individual word from zero to multiple emotions from the set of emotions provided by this library: *joy, trust, anger, fear, surprise, sadness, disgust, anticipation.*
Following the cleaning process described in the first part of this report, we also lemmatized all of the lyrics using NLTK's *WordNetLemmatizer* in order to make this activity more precise.
The library also includes assigning the *positive* or *negative* sentiment to each word. We made the decision to omit the two sentiments from the data because the objective of our analysis is to concentrate solely on the previously mentioned emotions.

After assigning emotions to individual words, the frequency of each emotion was then calculated for each lyrics.

Calculating the average values for each genre, as shown in Table 3, allows for a preliminary study of these data. We can observe that the range of values is quite small, indicating a fairly narrow distribution of values and, consequently, that lyrics across genres have a similar emotional impact. However, specific values can be found in certain emotions, such as a "peak" of *joy* in the *Jazz* and *R&B* genres, or *fear* and *anger* in the *Metal* and *Hip-Hop* genres. Figure 6 helps us visually understand the values of joy and anger (the emotions with the highest standard deviation values) as well as how, in contrast to other musical genres, those genres are the extremes.

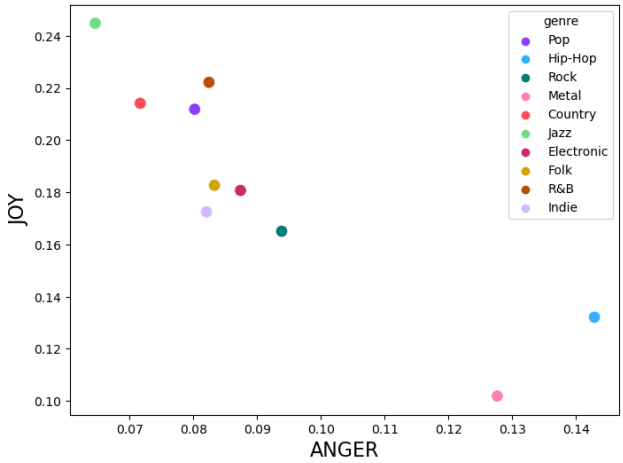

Figure 6: Average values per genre - Anger and Joy

Following that, we extracted the outlier songs based on the frequency of emotions. Songs that fall into the last percentile of the genres' value distribution were identified as outliers, and we looked at some information on those records.
What we found is that, even when a genre's frequency is relatively low in comparison to the others, like Metal in joy outliers, every genre has at least a few songs that belong to the group of outliers for each emotion. However, this is further evidence that the majority of the songs span a limited diversity, and it is once more challenging to clearly distinguish the genres.

genre	**fear**	**anger**	**trust**	**surprise**	**sadness**	**disgust**	**joy**	**anticip**
Pop	0,116705	0,080202	0,152812	0,080295	0,130248	0,059841	0,211848	0,163459
Hip-Hop	0,143987	0,142900	0,140988	0,075709	0,124253	0,104151	0,132137	0,129725
Rock	0,130645	0,093847	0,151161	0,077848	0,139944	0,072124	0,165079	0,163652
Metal	0,179155	0,127668	0,126071	0,062339	0,165057	0,095515	0,101887	0,131028
Country	0,100231	0,071661	0,162568	0,082309	0,133518	0,058769	0,214136	0,174844
Jazz	0,090753	0,064599	0,165511	0,082611	0,121639	0,053562	0,244810	0,170515
Electronic	0,125669	0,087389	0,147431	0,075242	0,133828	0,060365	0,180726	0,159351
Folk	0,120277	0,083304	0,163982	0,075577	0,131178	0,061287	0,182709	0,167211
R&B	0,116057	0,082445	0,154984	0,073573	0,128037	0,063653	0,222171	0,155055
Indie	0,121568	0,082055	0,157431	0,081734	0,144969	0,062635	0,172501	0,171705

Table 3: Average frequencies per genre

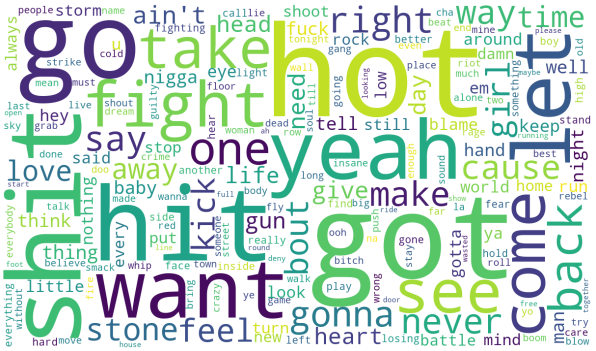

Figure 7: *Anger* Word Cloud

Figure 8: *Joy* Word Cloud

We continue analyzing the *joy* and *anger* emotions. Analysis of the distribution of outliers reveals that the band AC-DC is the artist most prevalent in this group for the emotion of *anger*, whereas B.B. King and the Beatles are the ones for the emotion of *joy*. In Figure 7 and 8, we can see the most frequent words regarding the outliers of these two emotions.

The distribution of songs in the reduced dimensional space is visualized in Figure 9 using the t-SNE (*t-distributed stochastic neighbor embedding*) approach. Once more, the lyrics are grouped together into a single cloud of points. Nevertheless, in contrast to the other genres (e.g. *Rock*, which is dispersed over the entirety of the cluster), we can see how the *Hip-Hop* (in orange) is grouped in the bottom right of the plot. *Jazz*, on the other hand, is mostly present towards the top of the plot.
We also attempted to use UMAP (*Uniform Manifold Approximation and Projection*) to display this distribution of values for all the songs, but we were unable to get any results that were really interesting.

This ends the analysis of the lyrics' emotions. The findings demonstrate that the frequency of emotions found in the lyrics included in this dataset is insufficient to make a clear distinction between the ten genres. There is no discernible relationship between a particular feeling and one or more genres, not even after considering the songs with the highest emotion values. The only exception is *Hip-Hop*, which stands out in its lyrics and whose songs take on similar values and that are dissimilar to those of the other genres.

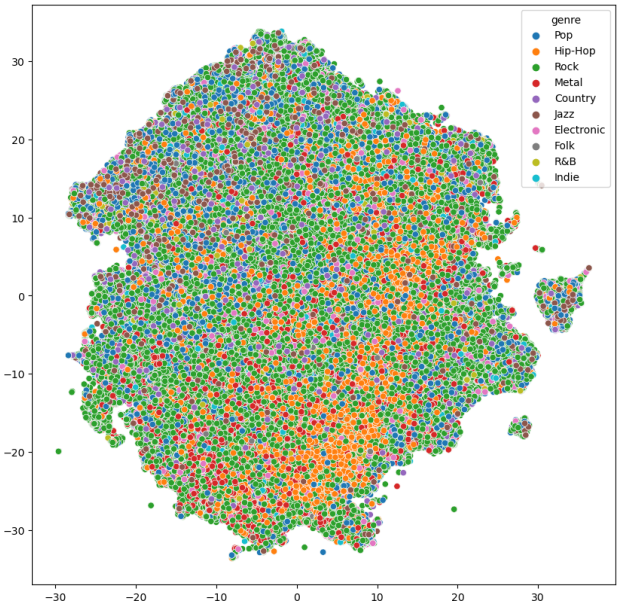

Figure 9: Plot of songs' emotions (t-SNE)

4 Explicit Lyrics Analysis

Nowadays, every online music platform has parental restrictions that prevent youngsters from seeing explicit content when they use their service.
As a result, it is critical to appropriately recognise explicit content. When a song is released, the explicit logo is used whether the lyrics or content of the song contain strong language, references to violence, discriminatory words, etc.
As far as we are concerned, we would like to carry out a separate work than the main one. As a result, we decided to examine the lyrics of the songs to determine which were explicit and which were not, and classify them accordingly.

4.1 Lyrics labelling

The first step in our study was to compile a list of swear words. We used a collection of words developed by *musixmatch* researchers who analyse the use of profanity in lyrics to create the *badwords* list.
In order to assign the correct labels *explicit*, *not explicit*, we developed a simple function `is_explicit`, which cleans the lyrics before tokenizing them.
Secondly, each token is checked against the *badwords* list. If a lyric contains at least 5 swear words, the song is marked as *explicit*. Differently, it will be assigned as *not explicit*.
Finally, we added this new set of labels to a new column called `is_explicit` in our dataset.
Because certain lyrics might contain forbidden keywords (for example, 'bi**h' and 'ni**a') that were captured by regular expressions, we decided to classify them as bad words as well. We determined the percentage distribution of the songs split into the two groups after we constructed the dataset with the songs accurately labelled. We discovered that over 75% of the songs are not explicit, while the remaining 25% contain swear words. As a result, the dataset is highly unbalanced, as it is with genres.

To have a better understanding, we computed the distribution of `is_explicit` per genre (Figure 10). As expected, *Hip-Hop* contains approximately 80% explicit songs, whereas the other genres contain approximately 20% explicit songs. This is because the *Hip-Hop* genre is comprised of rap tracks that are well-known for their abundance of swear words.

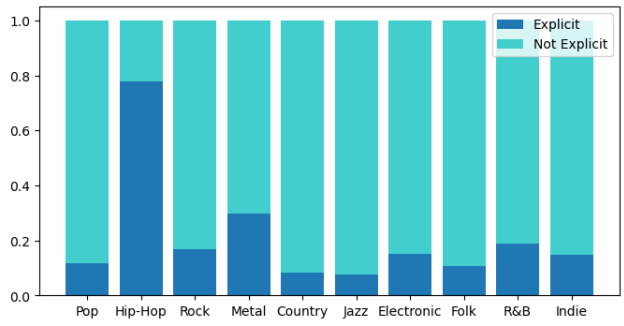

Figure 10: Explicit lyrics distribution per genres

4.2 Classification

As a last task for our analysis of explicit and non-explicit lyrics, we do a classification to see if the classifiers perform better with this work rather than the main one (classification of songs in genres).
We perform the vectorization using *Tfidfvectorizer* with unigrams and then we run some classifiers such as *Gaussian Naive Bayes*, *Random Forest*, *ANN*, *RNN*, and *Bi-LSTM*.

As can be seen in Table 4, we choose the same three classifiers that we have already described in section 2 to compare the two classifications.
The Random Forest yields the finest outcomes.
If we compare the accuracy of the Random forest in this classification and the accuracy of the same classifier in the main classification, the one associated to this classification has a higher value. We believe that the classifier will find this task easier because here it has to work only with two labels rather than ten different genres.

Random Forest	**Accuracy**	**F1-Score**
	0,96	[0,97 0,91]
	Recall	**Precision**
	[0,99 0,85]	[0,96 0,98]
ANN	**Accuracy**	**F1-Score**
	0,96	[0,97 0,90]
	Recall	**Precision**
	[0,99 0,84]	[0,95 0,97]
Naive Bayes	**Accuracy**	**F1-Score**
	0,87	[0,91 0,74]
	Recall	**Precision**
	[0,87 0,84]	[0,95 0,67]

Table 4: Classification Report on the Original dataset

However, as seen in subsection 4.1, there is a significant imbalance in favour of non-explicit lyrics. As a result, we used random undersampling to balance the dataset and then running the classification. Table 5 shows the results.
As with the original dataset, the random forest is the best classifier for our task. Nonetheless, the accuracy is decreased with the balancing dataset.

Random Forest	**Accuracy**	**F1-Score**
	0,92	[0,93 0,92]
	Recall	**Precision**
	[0,97 0,88]	[0,89 0,96]
ANN	**Accuracy**	**F1-Score**
	0,91	[0,92 0,91]
	Recall	**Precision**
	[0,95 0,88]	[0,89 0,95]
Naive Bayes	**Accuracy**	**F1-Score**
	0,86	[0,86 0,86]
	Recall	**Precision**
	[0,86 0,87]	[0,87 0,86]

Table 5: Classification Report on the Undersample dataset

Eventually, as it was shown, all of the three classifiers provided, both for the original and the balanced datasets, perform better in this task.

Conclusions

This project proposes the exploration, analysis and classification of music genres starting from song lyrics. The various tasks were carried out on a dataset that, after being suitably prepared, presents 238,230 records.

First, analyses were made on the text and on the vocabulary to measure the lexical diversity of the different genres, in which *Hip-Hop* appear to have the most extensive vocabulary in relation to the length of the lyrics. Then, we used the TF-IDF feature to perform K-Means clustering to examine the distribution of the different genres: among the three clusters identified, Hip-Hop lyrics stand out from the other genres, strongly represented in cluster 2, while other genres are more or less equally distributed among clusters.

Follows lyrics' emotion analysis to determine any relationship between musical genre and emotions. Using NRCLex, we assigned to each lyric different emotional values and then computed the emotions for each genre. Results show that all genres share very similar emotional values and, also in this scenario, *Hip-Hop* differs from other genres in presenting lyrics that share similar values to each other but are different from the other genres.

After this initial investigation, we addressed the genre classification task by exploring a wide range of models with different configurations, starting from Decision Tree to get to the BERT transformers. Given the class imbalance of the dataset, different sampling techniques have been exploited. The best performance was reached with random oversampled data, then using Word2Vec as word embedding over a Random Forest classifier.

Finally, we added a new feature to the dataset by automatically labelling the explicit lyrics. Then, trying three distinct models, we were able to classify explicit lyrics with a Random Forest classifier with an overall accuracy of 0,96.

We can infer that, given this dataset, obtaining a highly accurate classification is difficult due to both the strong similarities among genre lyrics and the great imbalance of the classes. As a result, superior outcomes can be seen in the most common and distinct genres, such as *Hip-Hop*, *Rock*, *Pop* and *Metal*.