Politecnico di Torino **Université Paris-Saclay**

A MACHINE LEARNING APPROACH TO MUSICAL SUCCESS PREDICTION AND GENRE EMERGENCE

Supervisor:
Dr. Giulio Prevedello

Candidate:
Claudio Ascione

Academic Supervisor:
Ch.mo Prof. Alessandro Pelizzola

Academic Year 2021–2022

Abstract

In the past years, Spotify and YouTube have been dominating the music production scene, thus storing valuable information about how songs achieve popularity and become a successful hit. Past researches tried to determine the hit-potentiality focusing on the audio characteristics and showed the limitation of this approach. In this work, we studied popularity dynamics of songs including, together with audio characteristics, features from artist and release's context, and processed these using machine learning techniques. As metrics for popularity, we employed the cumulative views on YouTube achieved by a song's video on this streaming platform. First, we investigated the relationship between songs with similar audio feature and their release date to identify sub-genres with concomitant popularity. Then, we processed our data by fitting the songs' popularity, ranked by their followers' number on Spotify, using a Zipf's model to provide a reference for a song's success accounting for the fame previously achieved by its artist. Thus, comparing the actual performance against the one predicted by the Zipf's law, we defined a success score whose forecast, using different machine learning models, could determine which features had most predictive potential. From our results, release dates showed little to no relation with music sub-genres, as the distribution of release dates by sub-genre peaked in almost the same times. Namely, songs are released irrespective of their genre. Moreover, while the correlation between YouTube views and followers' number on Spotify was moderate, stronger dependence resulted from songs with negative Success score. We hypothesised that these songs only achieve "business as usual" performance, as opposed to those with positive score that outperform instead. Using supervised learning techniques, we showed that audio and contextual features from Spotify and YouTube (e.g., the number of subscribers of the YouTube channel featuring the song) were sufficient to train a model capable of determining with good accuracy the success score of different songs: in particular, a great influence on the prediction was determined by the YouTube context in which the song was released.

Contents

Chapter 1

Introduction

As of now, the music industry is adjusting to a digital revolution front-run by several streaming platforms (Spotify YouTube, Apple Music, etc.) that led in the past decade to a change of the paradigm on how music is enjoyed. One of the main questions arising among music practitioners is how to forecast success of specific tracks. There are many different effects that might affect the dynamics of a hit song [1]. The experience felt by the consumer, for example, is regarded to have a particular influence since it is able to capture the consumer's reaction that triggers interest [2]. This process could lead to a recurrent pattern in the acoustic evolution of songs and the statistical physics approach turned out to be useful to determine these emerging patterns [3]. In terms of pure success, however, it is far more interesting to look at the problem from a more machine-learning oriented point of view. It was demonstrated that there is indeed a dependence between audio features and success and that it could be possible to predict on-chart's results by songs up to a certain margin of error [4, 5, 6, 7, 8, 9, 10, 11, 12]. On a more methodological level, different approaches have been tried, such as Middlebrook's work [13] who has been able to predict optimally the success (considered as finding a song in the Billboard Hot-100) using supervised learning techniques (such as neural networks, random forests, and Support Vector Machine), or Yang's work [14] in which it was attempted to apply convolutional neural networks to address the problem. Moreover, it is interesting to see how contextual information, such as the genre and the artist popularity could influence the prediction. Asai [15] demonstrated how the artist's fame above all contributed to a prolonged presence of songs in top-charts, while Pham [16], instead, tried to improve the accuracy of the prediction by adding artist's and track's contextual features (such as the artist's name, release date, composer,etc.) to the training set, as well as Araujo [17] who tried to introduce contextual information about the presence of songs in past Spotify's rankings. On the other side, another interesting decision that could affect the performance of a song is the time period in which it is released. Discussions with music industry experts here in Sony CSL revealed that most marketing strategies, put in place by production companies, thoroughly assess the right

timing at which a song must be released in order to maximize exposition and revenues. This might yield that similar songs could be released in analogous times of the year, therefore leaving a trace of a genre spiking in certain times of the year.

In this study, we tackled the problem of detecting release-strategies patterns associated to specific sub-genres and success prediction. In Chapters 2, we will introduce the methods and theoretical foundation on which our research was anchored. In Chapter 3, we will report the results obtained by our analysis. In Chapter 4, finally, we will discuss said results and future developments for the line of research on music success prediction.

Chapter 2

Methods

In this chapter, we will describe data collection and composition; then, we will formally introduce the methods and algorithms used to conduct the study on the audio features and on the prediction of the success score.

2.1 Data collection

The data was obtained through Spotify's and YouTube's APIs, in two different stages. First, from Spotify, a fetcher program scraped every day all the information associated to the songs that have ever appeared in selected playlists. In particular, once a song was detected inside a playlist, the fetcher kept track of its evolution even after its removal from said playlist. Then, the song's title and artist were queried from YouTube to save the three most viewed videos associated to the track. To determine the song's main video, the one that the fetcher had seen for the longest time was selected; if this was not discriminant enough, then the one having the most views was chosen.

The fetched data was grouped in five main groups:

- Audio features, which comprises of acousticness, danceability, duration (in ms), energy, instrumentalness, liveness, loudness, speachiness, tempo and valence;
- Playlist features, made up of two sub-datasets: playlists' followers and track position;
- Context YouTube, consisting in the songs' channel's videocount and followers;
- Context Spotify, comprising of the songs' artists' number of followers and popularity;
- Reaction features, made up of comments, likes and dislikes under songs' Youtube's videos;

- Popularity measure, consisting of songs' YouTube views and Spotify's popularity.

Aside from these, we had other data regarding static description of the tracks, like the explicitness or the type of album the song was released in; most of this information, however, proved to be unimportant for our study and the sake of our results.

2.2 Audio features' analysis

The audio features are a part of the dataset that describes the audio characteristics of the tracks. To each song, we assigned a 10 dimensional vector, each component describing a specific musical aspect of the track. In order to represent these quantities it was necessary to apply a dimensionality-reduction algorithm. The one which appeared to be the most stable for our data was the UMAP reduction [18, 19]. This method, in brief, consists of two phases: a graph construction in high-dimensional space and an optimization of said mapping into a low-dimensional graph. In the first phase, we construct a weighted graph where each node is an observation, and its nearest neighbors are the k closest other observation with respect to a given distance (usually Euclidean). The weight on each edge is the distance between two nearest neighbors, rescaled by a factor depending on k, in order to preserve the high-dimensional proximities. In the second phase, instead, we try to optimize the low-dimensional graph layout with a process that can be viewed as a stochastic gradient descent on individual observations or as a force-directed graph layout algorithm [20]. This causes, analogously to what happens for t-SNE [21], points which are far from each other to be drawn further away, and those that are close, closer. However, in contrast with t-SNE, this algorithm can preserve the global structure of our observations.

In order to evaluate the emergence of genres, we tried the clustering algorithms K-Means, Density-based Clustering and Gaussian Mixtures, which define music genre, respectively, by: the proximity between similar songs, the local density, and Gaussian-like concentration, within the space of audio features.

2.2.1 Clustering methods

K-means. K-means [21] is a pretty straight-forward and simple algorithm. Let us consider a set of N unlabeled observations, $\{\mathbf{x}_i\}$ $i = 1, \dots, N$ where $\mathbf{x}_i \in \mathbb{R}^p$ and p is the number of features, and $\{\boldsymbol{\mu}_k\}$ $k = 1, \dots, K$ set of K cluster centers. The idea behind is to look for centers by minimizing this following cost function,

$$\mathcal{C}(\{\mathbf{x}, \boldsymbol{\mu}\}) = \sum_{k=1}^{K} \sum_{n=1}^{N} r_{nk} (\mathbf{x}_n - \boldsymbol{\mu}_k)^2 \tag{2.1}$$

where r_{nk} is a binary variable called the assignment, which will assume values 1 if $\mathbf{x}_n$ belongs to cluster k, 0 otherwise. The algorithm will therefore consist of

two steps:

- Given assignments r_{nk}, we minimize $\mathcal{C}$ to obtain the centers. This yields,

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} r_{nk} \mathbf{x}_n \tag{2.2}$$

- At fixed $\{\boldsymbol{\mu}_k\}$, we again minimize $\mathcal{C}$ with respect to the assignments.
- We repeat these steps iteratively until some convergence is met.

It is clear how this method privileges points spatially close to each other, exploiting a euclidean metric to evaluate the distances; also, let us notice how this algorithm is guaranteed to converge at some point. However, a major drawback is that it considers the variances of different clusters as a fixed equal quantity, so in cases where this assumption fails the algorithm can lead to a faulty result.

Density-based clustering. We mentioned beforehand that one of the main characteristic we wanted to highlight was that densely-packed agglomerate should yield for a musical genre. This can be reproduced through Density-based (DB) Clustering [21], which supports the idea that clusters are defined by regions of space with higher density of data points; those considered outliers or noise are instead expected to be part of low-density areas. In addition, DB clustering assumes local-density estimation of data as a statistically reliable quantity, therefore it is possible to order points by their densities, which in turn implies that, through this method, it is possible to determine clusters of different shape and sizes, highlighting the presence of outliers in the space. Let us have $\mathbf{X} = \{\mathbf{x}_n \in \mathbb{R}^p,\ n = 1, \dots, N\}$ a set of datapoints with p number of features. Defining the ϵ-neighborhood of point $\mathbf{x}_n$ as

$$N_\epsilon(\mathbf{x}_n) = \{\mathbf{x} \in \mathbf{X} \mid d(\mathbf{x}, \mathbf{x}_n) < \epsilon\} \tag{2.3}$$

where $d(\cdot, \cdot)$ is the Euclidean distance, we can consider N_ϵ as a crude estimate of the local density. Moreover, we can define $\mathbf{x}_n$ as a core-point if at least N_{min} points are in N_ϵ, where N_{min} is a free parameter of the algorithm that sets the scale of the size of the smallest cluster one should expect. Finally, a point $\mathbf{x}_i$ is said to be density-reachable if it is in the N_ϵ of a core-point. With this in mind, the DBSCAN algorithm (the most prominent Density-based clustering methods) will unfold in the following way:

- Until all points in $\mathbf{X}$ have been visited; do
  - Pick a point $\mathbf{x}_i$ that has not been visited
  - Mark $\mathbf{x}_i$ as a visited point
  - If $\mathbf{x}_i$ is a core point; then
    - Find the set $\mathcal{C}$ of all points that are density reachable from $\mathbf{x}_i$.

    - $\mathcal{C}$ now forms a cluster. Mark all points within that cluster as being visited.

- Return the cluster assignments $\mathcal{C}_1, \ldots, \mathcal{C}_k$, with k the number of clusters. Points that have not been assigned to a cluster are considered noise or outliers.

Notice the only two parameters DBSCAN asked for are ϵ and N_{min}, with a computational cost of $O(N \log N)$. From this algorithm, a more sophisticated approach is represented by the HDBSCAN algorithm [22] [23]. This is an algorithm that attempts to apply an hierarchical approach to the concept of Density-based Clustering and, therefore, it tries to span over a set of possible ϵ (defined through a Single-Linkage hierarchical process) in order to find the optimal value. It only asks for one parameter m_{pts}, a smoothing factor in density estimated of clusters, (i.e. how smooth my cluster must be in order to stop the search) but its computational time tends to be of superior order than the one for DBSCAN, $O(N^2)$.

Gaussian Mixture models. Lastly, following the hypothesis of Gaussian distribution for our clusters, let's discuss Gaussian mixture models [21]. In these, points are drawn from one of K Gaussians with mean $\boldsymbol{\mu}_k$ and covariance matrix $\boldsymbol{\sigma}_k$, such that

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\sigma}) \sim \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})\boldsymbol{\sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})^T\right). \tag{2.4}$$

Denoting π_k as the probability of drawing a point from the k-th Gaussian, the total probability of generating point $\mathbf{x}$ from the model is

$$p(\mathbf{x}|\{\boldsymbol{\mu}_k, \boldsymbol{\sigma}_k, \pi_k\}) = \sum_{k=1}^{K} \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\sigma}_k)\pi_k. \tag{2.5}$$

We can extend this equation taking into account a dataset $\mathbf{X} = \{\mathbf{x}_i,\ i = 1, \ldots, N\}$ such that

$$p(\mathbf{X}|\{\boldsymbol{\mu}_k, \boldsymbol{\sigma}_k, \pi_k\}) = \prod_{i=1}^{N} p(\mathbf{x}_i|\{\boldsymbol{\mu}_k, \boldsymbol{\sigma}_k, \pi_k\}) \tag{2.6}$$

It is useful now to introduce a latent variable $\mathbf{z} = (z_1, z_2, \ldots, z_K) \in \{0, 1\}^K$ that quantifies to which cluster a datapoint $\mathbf{x}$ belongs to, assuming each datapoint can belong only to a single cluster (we will denote the total set of latent variables $\mathbf{z}$ as $\mathbf{Z}$). This implies that the probability of observing $\mathbf{x}$ given $\mathbf{z}$ will be

$$p(\mathbf{x}|\mathbf{z}; \{\boldsymbol{\mu}_k, \boldsymbol{\sigma}_k\}) = \prod_{k=1}^{K} \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\sigma}_k)^{z_k} \tag{2.7}$$

and the probability of observing a given value of latent variable

$$p(\mathbf{z}|\{\boldsymbol{\pi}_k\}) = \prod_{k=1}^{K} \pi_k. \tag{2.8}$$

We can now use Bayes' rule to obtain the joint probability $p(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta})$, where $\boldsymbol{\theta} = \{\boldsymbol{\mu}_k, \boldsymbol{\sigma}_k, \pi_k\}$,

$$p(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta}) = p(\mathbf{x}|\mathbf{z}; \{\boldsymbol{\mu}_k, \boldsymbol{\sigma}_k\})p(\mathbf{z}|\{\boldsymbol{\pi}_k\}) \tag{2.9}$$

which, reapplying Bayes, implies that the conditional probability of the data point $\mathbf{x}$ being in the k-th cluster, $\gamma(z_k)$ given the model parameters $\boldsymbol{\theta}$ will be

$$\gamma(z_k) \equiv p(z_k = 1|\mathbf{x}; \boldsymbol{\theta}) = \frac{\pi_k \mathcal{N}(\mathbf{x}|\mu_k, \sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\mu_j, \sigma_j)}. \tag{2.10}$$

Ideally, we could now apply MLE and obtain $\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} p(\mathbf{X}|\theta)$. However, due to complexity it is extremely hard to obtain the maximum in this way, hence a new strategy is needed. Another possible way to obtain the parameters is through Expectation Maximization (EM) [24]. Given an initial guess for the parameters, EM computes them iteratively by setting

$$\theta^{(t+1)} = \arg\max_{\theta} \mathbb{E}_{p(\mathbf{Z}|\mathbf{X};\theta^{(t)})}[\log p(\mathbf{X}, \mathbf{Z}; \theta^{(t)})], \tag{2.11}$$

yielding,

$$\mu_k^{(t+1)} = \frac{\sum_{i=1}^{N} \gamma_{ik}^{(t)} \mathbf{x}_i}{\sum_{i=1}^{N} \gamma_{ik}^{(t)}} \tag{2.12}$$

$$\sigma_k^{(t+1)} = \frac{\sum_{i=1}^{N} \gamma_{ik}^{(t)} \left(\mathbf{x_i} - \boldsymbol{\mu}_k^{(t+1)}\right) \left(\mathbf{x_i} - \boldsymbol{\mu}_k^{(t+1)}\right)^T}{\sum_{i=1}^{N} \gamma_{ik}^{(t)}} \tag{2.13}$$

$$\pi_k = \frac{1}{N} \sum_{i=1}^{N} \gamma_{ik}^{(t)} \tag{2.14}$$

where $\gamma_{ik}^{(t)} = p(z_{ik}|\mathbf{X}; \theta^{(t)})$.

2.3 Success score

To analyze a song's success, its audio features do not suffice: they do not convey any information about the artist's fame at the moment of release, or any other information regarding the fanbase/followers status, that can indicate the potential public size listening to the song. By leveraging power-law's structure of YouTube views, we defined a success score that could take artist's past popularity into account. In [25] it was reported that an ordered sequence of data ranked by frequency (i.e. the most frequent in position 1, the second-most in position 2 and so on) behaves as a power-law with respect to the rank, such that

$$F(s) \sim r_V(s)^{-\alpha}, \tag{2.15}$$

where $F(s)$ is the frequency associated to the song s, $r_V(s)$ a function that associates each song to its YouTube views rank and α is the power-law coefficient.

This model is called Zipf's law and it is able to effectively predict the behavior of the bulk of the ranking curve of our data, but it fails when it tries to estimate the less frequent events. Therefore, a straightforward linear fitting cannot be a good strategy to retrieve this pattern, since it would give too much weight to the fat tail. As shown in [26], the solution to this problem can be of different types: however, the one we found most useful for our data was establishing an exponential binning, such that the effect of the fat tail can be mediated by an exponential sample of elements.

Using this model, it is possible to measure if and how much a given song, ranked based upon a different criterion, outperforms the prediction based on the YouTube data, and how the different features affect this outcome. To do so, it was necessary to introduce a new metric, which we generally called 'success score'. Defining a new criterion for ranking our songs, we can consider the views of the r_F-th ranked song w.r.t. this criterion as

$$V(s) \equiv V(r_F(s)) \tag{2.16}$$

where $r_F(s)$ is function associating each song to a ranked position (i.e. $r_F(s)$ is the rank of song s with respect to the criterion F). To define the success score we consider the difference between the views of s and its prediction from the Zipf's law considering this ordering, s.t.

$$\text{score}(s) = \log_{10}(V(s)) - \log_{10}(c r_F(s)^{-\alpha}), \tag{2.17}$$

where $r_F(s)^{-\alpha}$ is the Zipf's law fitted on the YouTube-ranked songs and c is a constant value needed to rescale the model to fit the data correctly (notice that c can be retrieved from the previously introduced linear fit as $c = 10^b$ where b is the intercept of the linear fit). As a variant, we also considered its sign, i.e.,

$$\text{sgn}(\text{score(s)}) = \begin{cases} +1 & \text{if score(s)} > 0, \\ 0 & \text{else.} \end{cases} \tag{2.18}$$

Up to this point, our goal was to predict the score (or at least its sign), training our model with different subsets of features. To this end, different supervised learning tools were used.

2.3.1 Supervised learning

We tried to apply three main family of models: Linear models, Random forests' models and LGBM models.

Linear models. To have a first picture of the behavior of our data, we decided to apply linear models, in particular an OLS regression for score and a Logistic regression for sgn(score). As shown in [21], given $\boldsymbol{X} \in \mathbb{R}^{n \times p}$ set of data where n is the number of songs and p the number of features such that $\boldsymbol{X} = \{\mathbf{x}_i \in \mathbb{R}^p, i = 1, \ldots, n\}$, $\boldsymbol{\omega} \in \mathbb{R}^p$ set of parameters and $\boldsymbol{y} \in \mathbb{R}^n$ set of

labels, this model retrieves the right value of the parameters by minimizing the L^2 norm of the difference

$$\min_{\boldsymbol{\omega}\in\mathbb{R}^p}[\boldsymbol{X}\boldsymbol{\omega}-\boldsymbol{y}] \implies \boldsymbol{\omega}=\arg\min[\boldsymbol{X}\boldsymbol{\omega}-\boldsymbol{y}]. \tag{2.19}$$

On the discrete side, we introduce the logistic function as

$$\sigma(s)=\frac{1}{1+e^{-s}}. \tag{2.20}$$

This model tries to minimize the following cost function in order to retrieve the parameters $\boldsymbol{\omega}$, i.e.

$$\boldsymbol{C}(\boldsymbol{\omega})=\sum_{i=0}^{n} y_i\log\sigma(\boldsymbol{x}_i^{\boldsymbol{T}}\boldsymbol{\omega})-(1-y_i)\log(1-\sigma(\boldsymbol{x}_i^{\boldsymbol{T}}\boldsymbol{\omega})) \tag{2.21}$$

$$\implies \boldsymbol{\omega}=\arg\min[\boldsymbol{C}(\boldsymbol{\omega})]. \tag{2.22}$$

These models give us a first, very simple, idea of how the prediction could come out, however to improve accuracy we need more sophisticated models.

Random forest models. As shown in [21, 27, 28], a random forest is a type of model consisting of a collection of tree-structured predictors $\{h(\boldsymbol{x},\Theta_k),\ k=1,\dots\}$ where Θ_k are independent identically distributed random vectors. Each tree casts a unit vote for the most popular class at input $\boldsymbol{x}$. So, before heading into the discussion of these structures, let us introduce the concept of decision tree. Consider a graph $G=(V,E)$, $E\subset V\times V$ with V subdivided into three subsets, $V=D\cup C\cup T$ of D decision, C chance and T terminal nodes. In a decision node, the decision maker selects an action, i.e. one of the edges stemming from this node (one of the edges having the node in question as the parent). In a chance node, one of the edges stemming from it (a reaction) is selected randomly. Terminal nodes represent the end of a sequence of actions/reactions in the decision problem. For each edge $e\in E$, we let $e_1\in V$ denote its first element (parent node) and let $e_2\in V$ denote its second element (child node). Let us also introduce 2 function: one denoting payoff s.t. $y:E\longrightarrow\mathbb{R}$ and the other denoting probabilities s.t. $p:\{e\in E\mid e_1\in C\}\longrightarrow[0,1]$. We can now define a decision tree as a tuple (G,y,p) satisfying the following conditions:

1. there exists $r\in V$ (root) such that $\forall v\in V/r$ there exists a unique path from r to v, written as $r\to v$;
2. all and only terminal nodes have no children, i.e. $\forall v\in V: v\in T \iff \nexists e\in E: e_1=v$;
3. $p(\cdot)$ is correctly defined, i.e. $\forall v\in C:\sum_{e\in E,\,e_1=v}\ p(e)=1$.

The actual action of a tree is easy to understand: at each level, it asks questions in order to partition the data into smaller subsets, with leaves of this tree consisting in the end point of this sequential partition. There are different methods

to aggregate trees together; however, the most common is the BAGGing (Bootstrap AGGregation). Imagine to have a very large dataset, $\mathcal{L}$, subdivided in M subsets, $\{\mathcal{L}_1, \ldots, \mathcal{L}_M\}$. If each partition is large enough to learn a predictor, we can create an aggregate of predictors, trained on each subset. For continuous predictors, this is generally an average, s.t.

$$\hat{g}_A^{\mathcal{L}}(\boldsymbol{x}) = \frac{1}{M} \sum_{i=1}^{M} g_{\mathcal{L}_i}(\boldsymbol{x}) \tag{2.23}$$

where $g_{\mathcal{L}_i}(\boldsymbol{x})$ is the value of the predictor for the subset of the dataset $\mathcal{L}_i$ applied to vector $\boldsymbol{x}$. Analogously, for classifiers it becomes a majority vote, s.t.

$$\hat{g}_A^{\mathcal{L}}(\boldsymbol{x}) = \arg\max_j \sum_{i=1}^{M} I[g_{\mathcal{L}_i}(\boldsymbol{x}) = j], \tag{2.24}$$

where $I[g_{\mathcal{L}_i}(\boldsymbol{x}) = j]$ is an indicator that assumes value 1 if $g_{\mathcal{L}_i}(\boldsymbol{x}) = j$, 0 otherwise. The usefulness of BAGGing stands on the fact that this way of aggregating results can significantly reduce the variance of the estimator without affecting the bias.

LGBM models. For Random Forests' models, the decisional structure was solid, however the estimator considered were very simplistic. An improvement could be represented by the Gradient Boosted models and, in particular, the LGBM, or Light Gradient Boosted Method. Firstly, let's give a general definition for the Gradient Boosted method [21]. It combines the concept of boosting and gradient descent to produce an ensemble of decision trees. So, let's first denote a decision tree j with T leaves as $g_j(\mathbf{x})$ and parametrize it by two quantities: a function $q : \mathbf{x} \in \mathbb{R}^p \to \{1, \ldots, T\}$ that maps each data point to a specific leaf and $\omega(\mathbf{x}) \in \mathbb{R}^T$ that assigns weights to the leaves, such that $g_j(\mathbf{x}_i) = \omega_{q(\mathbf{x}_i)}$. We now need a cost function; specifying the prediction of our ensemble for a datapoint $(y_i, \mathbf{x}_i)$ as

$$\hat{y}_i = g_A(\mathbf{x}_i) = \sum_{j=1}^{M} g_j(\mathbf{x}_i), \qquad g_j \in \mathcal{F} \tag{2.25}$$

where M is the number of members of the ensemble and $\mathcal{F}$ the space of trees, we can define the cost function as

$$\mathcal{C}(\mathbf{X}, g_A) = \sum_{i=1}^{N} l(\hat{y}_i, y_i) + \sum_{j=1}^{M} \Omega(g_j), \tag{2.26}$$

where $l(\hat{y}_i, y_i)$ is a term that measures the goodness of predicition (it's usually considered differentiable and convex) and $\Omega(g_j)$ is a regularizer, needed to avoid overfitting on our datapoints. We will consider this regularizer rather general, but it depends on the type of Gradient Boosted tree we are trying to use. To

form the ensemble,we will use an iterative method that will consider a family of predictors of the form

$$\hat{y}_i^{(t)} = \sum_{j=1}^{t} g_j(\mathbf{x}_i) = y^{(t-1)} + g_t(\mathbf{x}_i) \tag{2.27}$$

(Notice, $\hat{y}_i^{(M)} = g_A(\mathbf{x}_i)$ by definition). For large t, each tree is a small perturbation to the predictor, s.t. we can expand the cost function in Taylor series to second order:

$$\mathcal{C}_t = \mathcal{C}(\mathbf{X}, g_t) = \sum_{i=1}^{N} l(y^{(t-1)} + g_t(\mathbf{x}_i), y_i) + \sum_{j=1}^{M} \Omega(g_j) \tag{2.28}$$

$$\simeq \mathcal{C}_{t-1} + \Delta\mathcal{C}_t, \tag{2.29}$$

where

$$\Delta\mathcal{C}_t = a_i g_t(\mathbf{x}_i) + \frac{b_i g_t(\mathbf{x}_i)^2}{2} + \Omega(g_t) \tag{2.30}$$

and

$$a_i = \partial_{y_i^{(t-1)}} l(\hat{y}_i^{t-1}), \tag{2.31}$$

$$b_i = \partial^2_{y_i^{(t-1)}} l(\hat{y}_i^{t-1}). \tag{2.32}$$

Finally, we choose the t-th decision tree to minimize $\Delta\mathcal{C}_t$. Differently, the LightGBM takes into account and implements two novel techniques, the GOSS and the EFB: the first excludes large amounts of data with small gradients and computes only on the remaining, while the second bundles mutually exclusive features to reduce their number. It is shown this method reduces the computational time needed to perform training by up to 20 times, keeping the same accuracy. For a more detailed description of LGBM models, we refer to [29].

Chapter 3

Results and Analysis

In this chapter, we will discuss the results obtained from our study regarding the analysis of audio features and the prediction of the success score, providing graphical representation and quantitative evidence in support of our findings.

3.1 Absence of temporal pattern for genre publication

One of the main goal of this study was to determine, if present, the insurgence of genres clusterization in audio features and check if similar songs are released in analogous intervals of time throughout the year. So the first step we took was to train a UMAP model and visualize our data. The result is shown in Fig. 3.1. Data distribution is observed to be homogeneous anywhere, with one giant dense block of points occupying the majority of the space, and an isolated smaller island in the bottom right part of the picture, that might indicate a specific genre.

It is interesting to see how the distribution evolves over time, separating all the tracks in subsets by groups of songs released in the same 30-days interval, as it is showed in Fig. 3.2. The tracks' subsets are evenly distributed, that is, even though they do not show the same density at each month, the distribution of points is even across the map, so the points dislocate for each interval of time like they do in the general case.

At this point, we investigated different clusters, their emergence over time and how the emergence affect the distribution per interval of time of our tracks. First, we trained a KMeans model on the data, to see if some first simple considerations can be made on the data. In order to look for the optimal number of clusters, we analyzed the behavior of the distortion score with the number of cluster K in Fig. 3.3. Applying the elbow method, we can see that for $K = 7$ a slight stabilization of the score is detected, hence we'll consider our number of clusters as 7. At this point, we trained our KMeans model, obtaining the results shown in Fig. 3.4. The retrieved clustering is not clear, as different clusters are identified (due to the simplistic nature of KMeans) but are not well separated

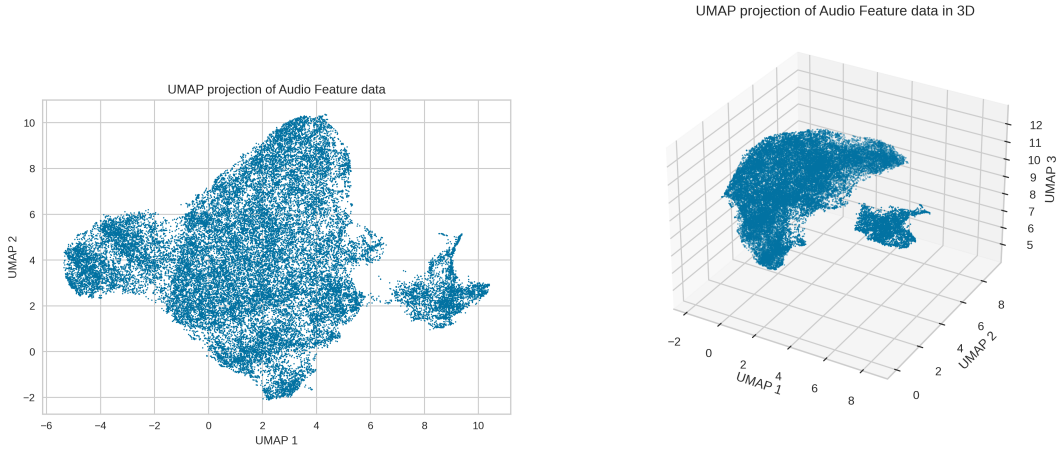


Figure 3.1: UMAP projection of Audio Feature data in 2D (left) and 3D (right). We can notice in both graphs an isolated cluster on the bottom right part, which might indicate a solitary genre.

or dense: most of them converge on the same block of data but are not really isolated. This hypothesis can be corroborated by the silhouette score analysis shown in Fig. 3.5. The low average score, in particular, supports the lack of real separation between clusters. We then tried to apply the HDBSCAN model as it can take the local density of points into account for determining clusters, rather than relying only on the spatial distance.

Using default values for m_{pts}, the results obtained is shown in Fig. 3.6. Again, no clear separation in the main block of data; however with this method the single "island" of points is identified as part of a single cluster. So far, this is the only instance we can make about emerging genre in our data but it only relates to a selected group of songs, in the sense that we can't argue about any other emerging genre outside of this specific sample. Moreover, this method gives us no information about how the centers behave and where they are located (differently to what is possible exerting from KMeans); therefore, we need to look out for another model, leveraging on the assumption that our data is distributed in a Gaussian-like pattern.

To that end, we applied the Gaussian mixture model. Again, to set the number of clusters K, we performed the elbow method analysis on the AIC score for different number of centers Fig. 3.7 [30], returning $K = 7$. For this number, the clustering process we are looking for is shown in Fig. 3.8. Two main isolated clusters can be identified in the graph (leftward side of the graph and rightward island), however the two clusters are not well separated (especially the rightward one, in which part of that island is associated with a different cluster that is somewhat of a background-noise cluster present across the entire mapping).

Moreover, having localized the different centers, we could analyze how many points are associated to each cluster for monthly subsets of songs, trying to exert

Figure 3.2: Distribution of songs in UMAP projection, grouped by month of release. Some months are more dense than the rest, like the 4-th. However, there is no dynamics shown in this representation that yields for a clear separation of genre by release date's time intervals.

a pattern describing the behavior of genre's publications with their release dates. The result is shown in Fig. 3.9. We see two main peaks in the songs' distribution of the number of publications over time, associated to February/March and September/October in which most songs seems to be released; unfortunately, the distributions are homogeneous and independent of the center/cluster (even though we could stress the fact that there are some centers that present a higher percentage of released songs, which, however, is true for any interval considered and not only for specific ones).

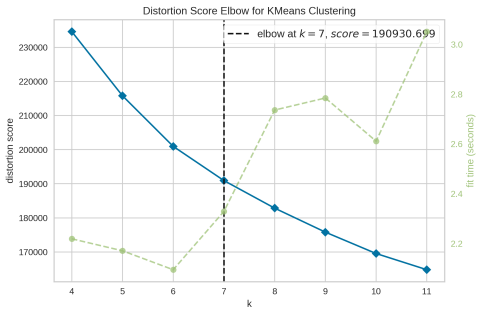


Figure 3.3: Distortion score analysis with elbow method. Notice the elbow, though very light, is detected at $K = 7$.

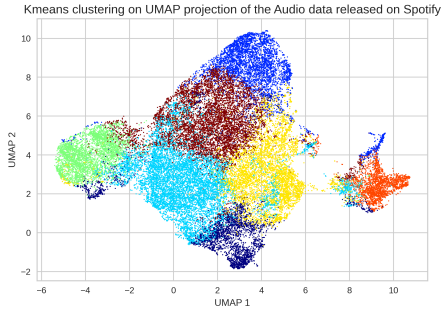


Figure 3.4: KMeans method applied to the Audio Feature dataset. No clear separation between clusters emerges.

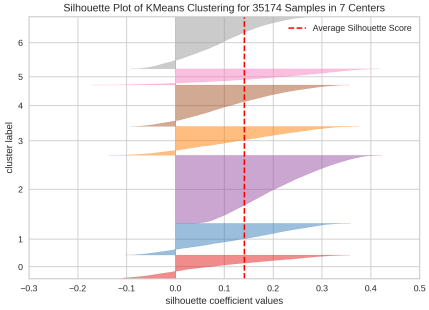


Figure 3.5: Silhouette plot for KMeans model with $K = 7$ clusters. Notice the average score is not very high, hence the clusters are not well separated.

3.2 Success prediction

The success of a song is shaped in three stages: first the song is released, then it gathers attention and finally it reaches the peak of popularity within 15 days from release (Fig. 3.10). As such, to analyze a song's success dynamics, time series must be pre-processed to match these phases. That is, in our case, syn-

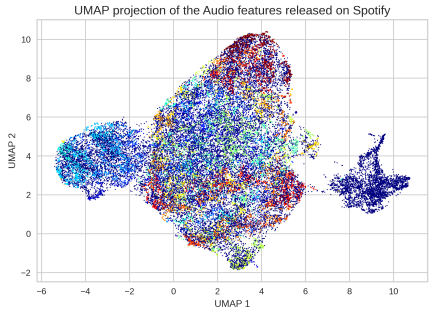


Figure 3.6: HDBSCAN model trained on the dataset. Notice how, in this case, the single island is considered as an individual, well separated cluster, while the rest of the data doesn't show clear separation.

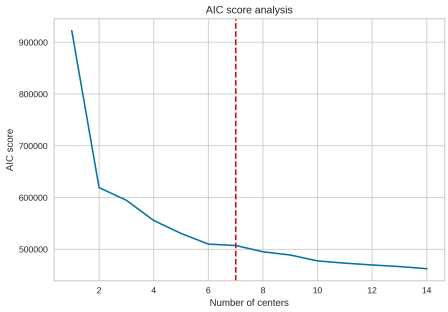


Figure 3.7: AIC score analysis for the Gaussian Mixture model. In this case, the optimal value is shown to be $K = 7$ (vertical dotted line).

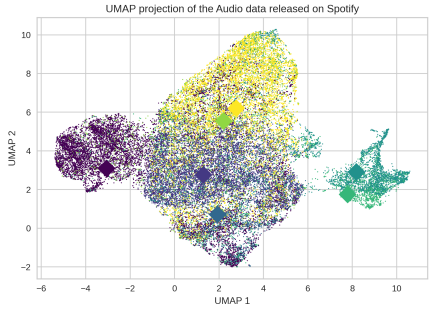


Figure 3.8: Gaussian Mixture clustering on UMAP-projected Audio Feature data. Clear single clusters can be seen in the extreme right and left part of the graph, even though the rightward island is separated into two main distributions.

chronizing all songs to the first day in which they appeared in data and their YouTube views were recorded.

This date was, then, stored and used to synchronize all the other features coherently. Such a step was necessary in order to analyze a track's evolution independently of its release period, and to be able to compare the evolution

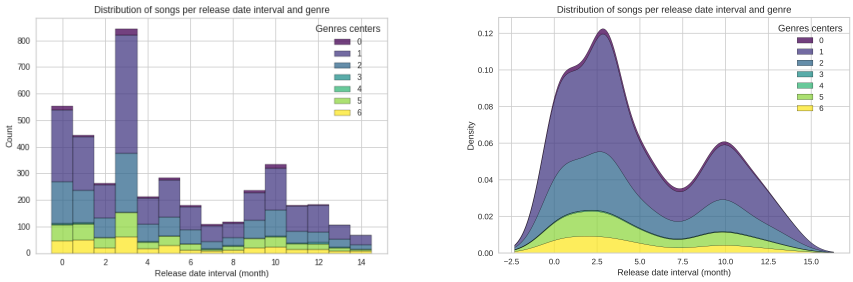


Figure 3.9: Histogram and KdE plots of the distribution of songs by 30-days intervals from late November 2020. It is shown how there is no privileged release month related to different genres.

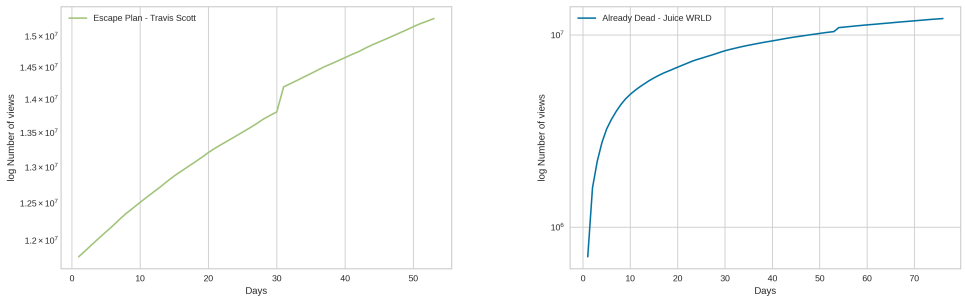


Figure 3.10: Cumulative views recorded day by day for 'Escape Plan' by Travis Scott and 'Already Dead' by Juice WRLD in log scale. Notice that the first one has already reached its "stationary" value when we started recording the data since the value of cumulative views changes minimally, while the second one reaches it after 15 days.

together in a standard way. To better understand this concept, we refer to Fig. 3.11 as an example. These two randomly chosen songs show analogous behaviors in terms of time-dependent YouTube views' increase per day with a first big spike on the release date and then a decrease interrupted by occasional smaller spikes.

Moving forward, to study success prediction, at first we decided to check if there could be a time pattern that described the evolution of any song. Defining our datapoints as $\mathbf{x} \in \mathbb{R}^p$ where $p = 8$ is the number of features we associated to any song, we trained a UMAP model in order to reduce dimensionality and visualize the data. The results obtained are shown in Fig. 3.12

No clear pattern emerges from the time series, either coloring by time instances or by songs. Afterwards, we decided to approach the problem with a different method. When considering the evolution of a song popularity behavior, we assume it will reach a "saturation point" after which variations in terms of success/views are minimal: from our data, we identified this point to be around 15 days after the release date (Fig. 3.10). Hence, taking into consideration the YouTube views of songs 15 days after their release date and ordering them up, we obtain the curve showed in Fig. 3.13. The behavior for low rank is consis-

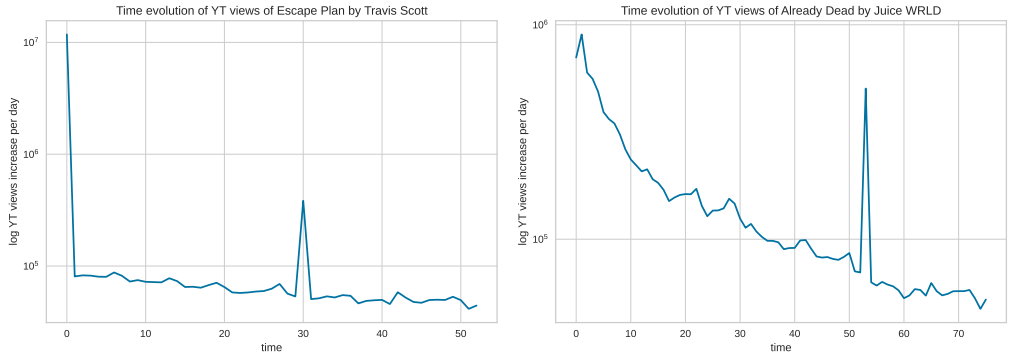


Figure 3.11: Time evolution of new YouTube views per day for 'Escape Plan' by Travis Scott and 'Already Dead' by Juice WRLD in log scale. Notice how the behaviors of these two songs is analogous: first a very peaked spike, then a decrease, with occasional lower spikes.

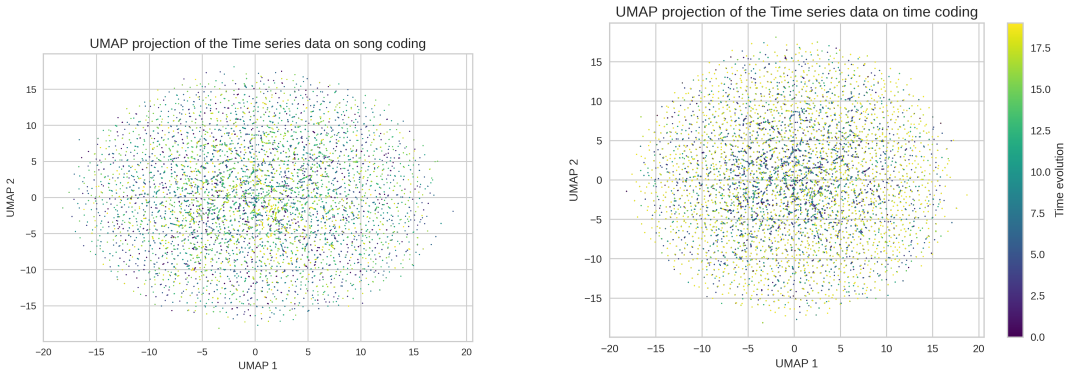


Figure 3.12: UMAP projection of the time series of all the tracks, in which each datapoint is a p-dimensional vector with coordinates representing a specific success feature at a specific instant of time. No clear pattern emerges in this analysis.

tent with what we would expect from a power-law behavior. However, we can already see how the tail behaves rather differently. In order to fit a power-law, we have to bin our data exponentially, thus reducing the effect of the tail. To choose the optimal binning, we have studied the behavior of the mean standard deviation over all bins, as shown in Fig. 3.14. For $N = 13$, the curve shows the last significant drop in the standard deviation and, following the "elbow method" approach, we chose this number as our optimal estimated number of bins, N_{bin}.

Fitting the Zipf's law, we obtain the curve shown in Fig. 3.15. Notice however, that the tail is still badly approximated by the power-law: as shown in [25], there are other function we could try to fit our model that should produce more accurate results.

Subsequently, we wanted to show how much a different rank should correspond in terms of success. To this end, we tried to reorder the data by the number of artist followers each song has on Spotify. The result is shown in

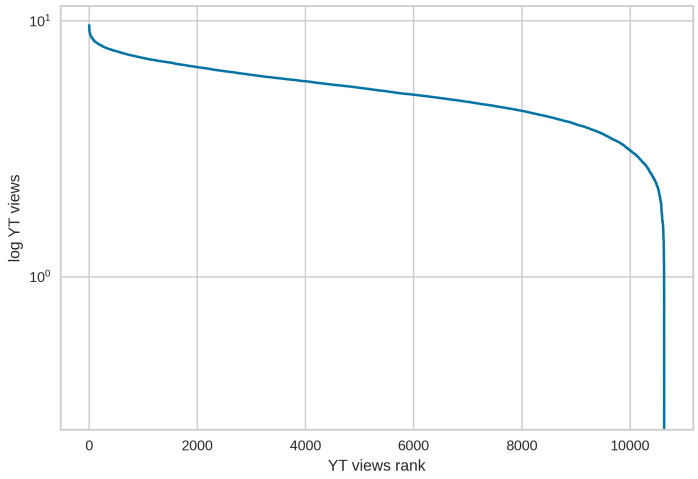


Figure 3.13: Logarithm of YouTube views ordered from top to bottom. For the first 8000 points, the curve behaves like a power-law (this will be more clear later).

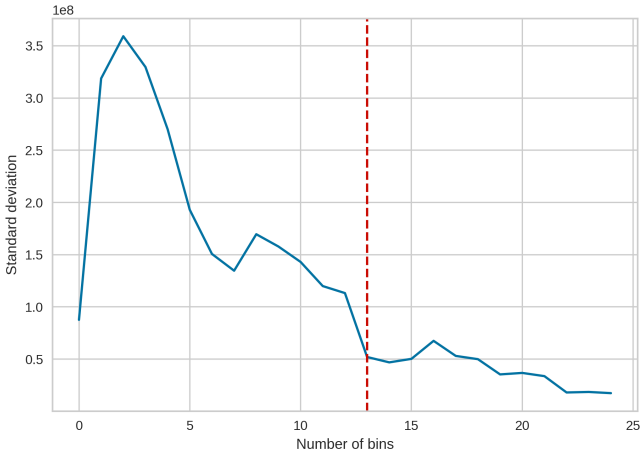


Figure 3.14: Mean standard deviation vs number of bins. We set the optimal number of bins at the last significant drop in the standard deviation.

Fig. 3.16. It is clear how the different ordering produces a very noisy dynamics, but after applying a rolling mean to the data it is possible to see how the curve behaves similarly as the one in Fig. 3.13.

An analogous analysis can be done ordering the data by the song popularity on Spotify, obtaining similar results (Fig. 3.17). The dynamic in this case seems to be more noisy, even though the general outlook is similar.

Effectively, this noise is generated by the discrepancy of the different rankings. The Zipf's law predictions on the rankings of the artists' popularity (used from now on) provides a measure of how many YouTube views a song should achieve, based on the artist accrued fame. This concept is well represented

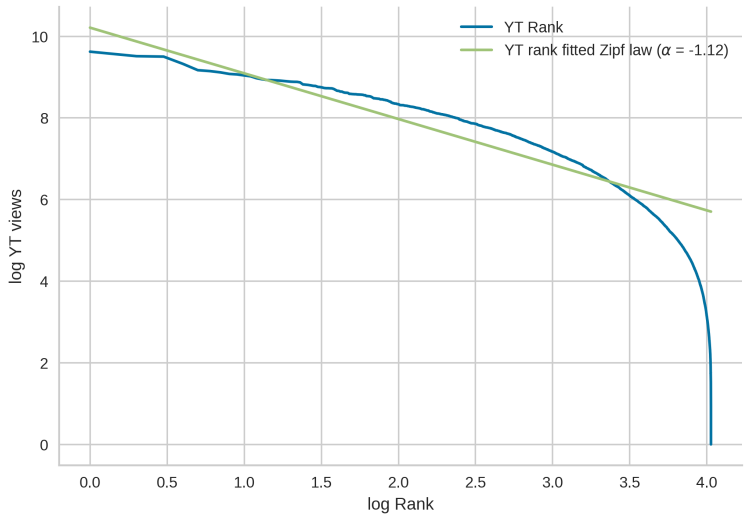


Figure 3.15: YouTube rank curve vs Fitted Zipf's law in log-log scale. The fit is trustworthy up until $r \simeq 3.5$, where the tail behavior heavily affects our data, showing in this interval the simplicity of our approximation.

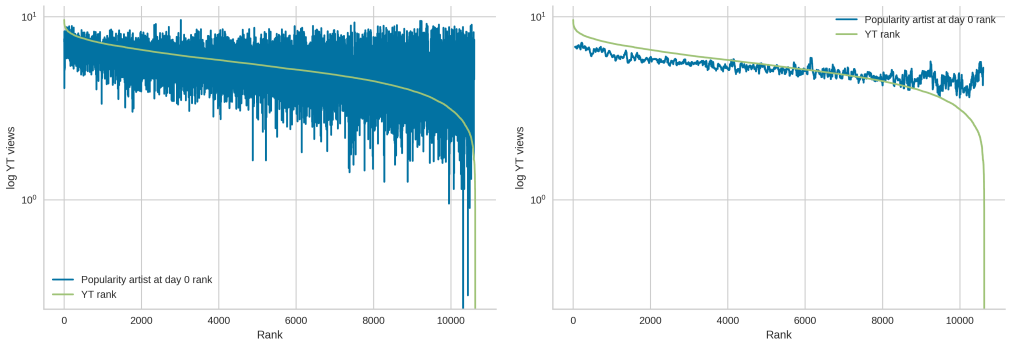


Figure 3.16: Log Views YouTube ordered by the artist popularity at day 0 vs Rank. The general behavior is consistent with what produced when ordered by the number of Views on YouTube, however it is incredibly noisy. In order to give a better look at the graph, we applied a rolling mean to the data.

by Fig. 3.18, in which the artists' popularity-ranked YouTube views are shown in comparison with the Zipf's law prediction. The magnitude of this discrepancy is the success score, defined in Chapter 2. Before heading into the success prediction part, it could be interesting analyzing the correlation between the YouTube ranked views and the artists' popularity ranked YouTube views to give more insight about the nature of the score.

We would expect these two quantities to be extremely correlated. However, as we can see in Fig. 3.19, the correlation between the two rank is moderate (Spearman's correlation = 0.415). Separating the songs in the two subgroups (positive and negative scored), we obtain a type of distribution shown in Fig. 3.20. Notice how the correlation on positive-scored songs is very low (as we could expect) and how the correlation on all the other songs significantly

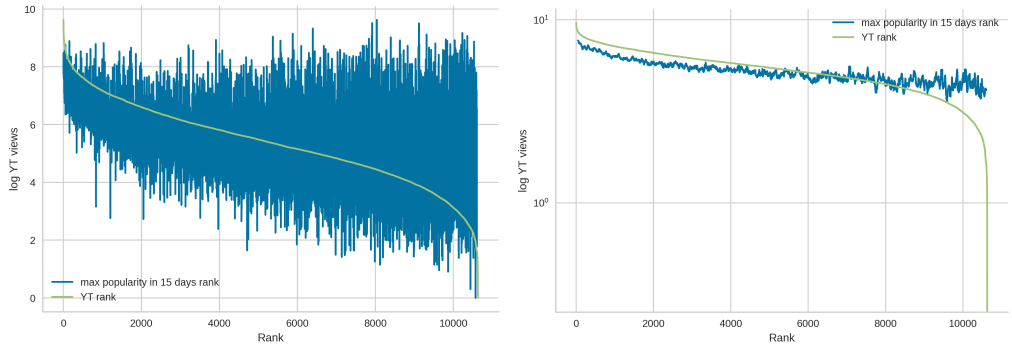


Figure 3.17: Log Views YouTube ordered by the max of song popularity in the first 15 days after the release date vs Rank. The general behavior is very similar to what happens in Fig. 3.16, however the noisy seems to be more relevant here.

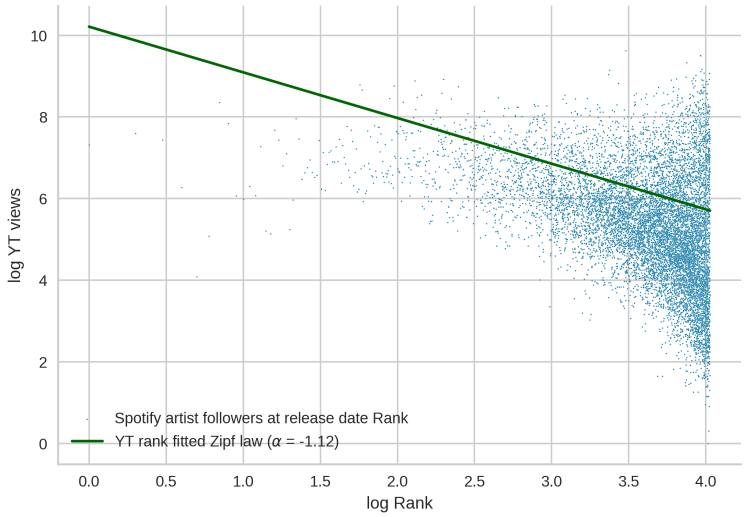


Figure 3.18: Fitted Zipf Law vs Spotify artist follower's rank in log-log scale. The perfomance of the track is estimated by comparison with the Zipf's law fit. Notice, in particular, how the Artist follower's ranked songs uniformly distribute around the Zipf's law prediction.

increases when taking these out of the picture. This finding suggests that under-performing songs display a "business-as-usual" dynamic, where song's success is correlated with past fame achieved by its artist, which is not the case for over-performing songs, whose success behaviour, in fact, is different and do not depend much on the artist followers.

For completeness, we also show the correlation between YouTube ranked distribution and Spotify song's popularity ranked distribution in Fig. 3.21. Again, the correlation is low but we could draw a consideration analogous to the previous one made. This case, however, is a less meaningful one, since the information provided by this quantity is an analogous quantification to the YouTube views, in the sense that both quantities provide insight on the song's performance on their respective platforms.

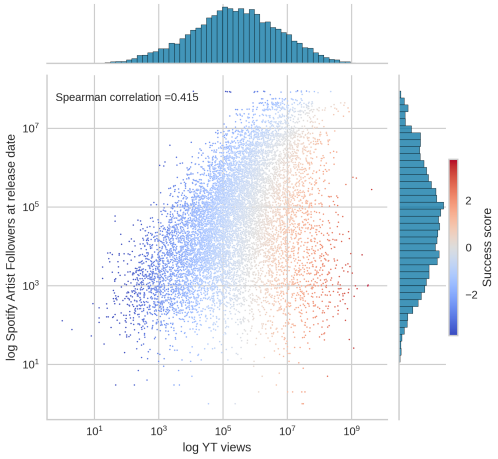


Figure 3.19: Grid map representing the distribution of points with respect to the YouTube views rank and the Spotify artist popularity rank in log-log scale. Notice that positive-scored songs are concentrated on the right-most side of the graph, while negative-scored ones are more present in the central part of the map, showing somewhat of a linear behavior.

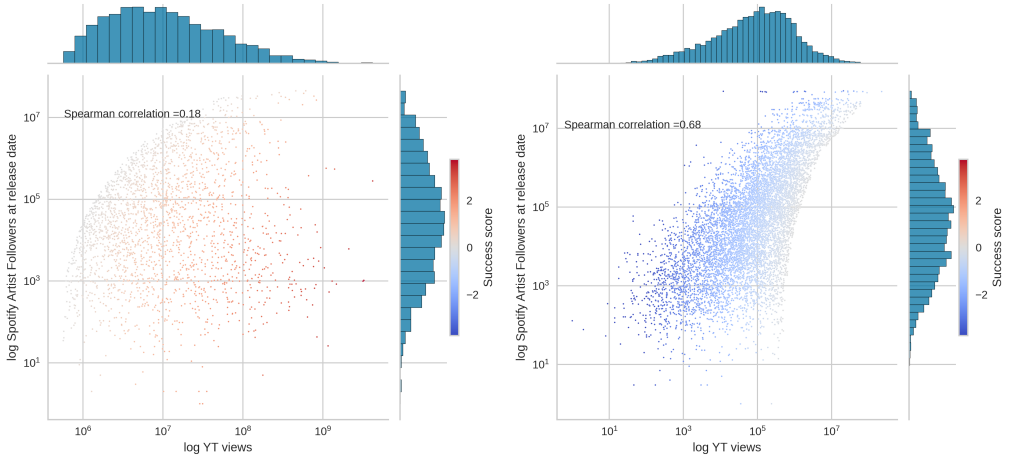


Figure 3.20: Grid maps representing the distribution of points with respect to the YouTube views rank and the Spotify artist popularity rank in log-log scale, separating data by positive and negative score. Notice that the positive-scored songs show very little correlation between the two rankings, while the negative-scored ones show a stronger correlation than the general case.

The crucial part of our study is to draw a prediction for this score, which requires the training of a model to forecast the performance of any song. To do so, we applied different models presented in the Methods chapter (the reader should bear in mind that for all the discrete classifiers we considered the discrete representation of the score). To benchmark the best performances of the models, we considered a training set composed by all these features and varied the model (Fig. 3.22). For the regression problem, LGBM model scored best in terms of R^2 score and mean square error. A similar result is obtained for the classification problem (Table 3.1).

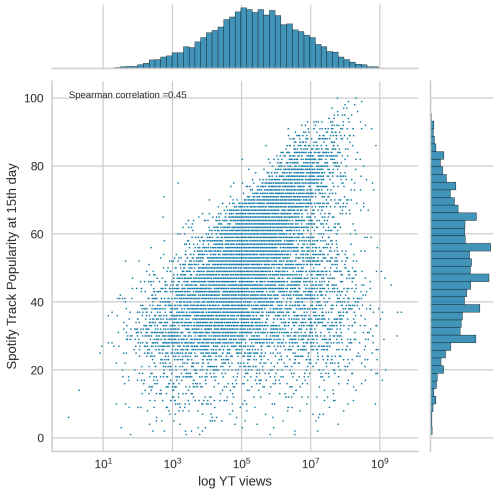


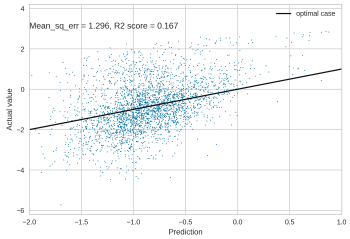
Figure 3.21: Grid map representing the distribution of points with respect to the YouTube views rank and the Spotify song popularity rank where the x-axis is log-scaled. The behavior and the correlation is analogous to the case of the Spotify artist popularity rank.

Ideally, a prediction of success should be best drawn before the release date of a song, but song's inherent characteristics (e.g. audio features) do not provide good accuracy. In our study, we assessed the contribution, in terms of predictive potential for success, of different set of features as they become available during the "life cycle" of a song. To this end, fixing the model as the LGBM, we varied the features included in the training set by adding each time new ones to those previously available. As a baseline, we will consider the audio features, to which we recursively add the features for the playlists, the Spotify context, the YouTube context, and the reactions.

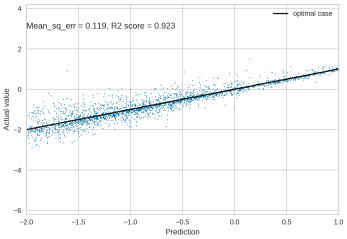
	Logistic regression	Random Forest	LGBM
F_1 score	0.941	0.960	0.968
AUC metric	0.954	0.987	0.990

Table 3.1: Accuracy and error on the prediction of the score for YouTube ranked data with different models. Notice how, in this case, all the models predict similar results.

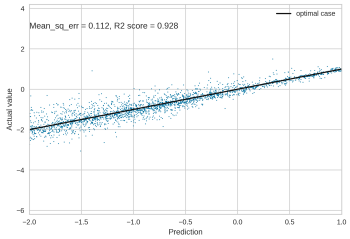
The results are shown in Table 3.2. When given the context on YouTube and the Reaction features (in this case comments under the YouTube video), a big improvement in the accuracy appears in our prediction. This makes sense since the score is computed from data ordered by YouTube views and therefore the YouTube context could be an important driver of the success of a song, as well as any descriptor of the YouTube situation. Moreover, the improvement yielded by the comments is also to be considered as a "post-op" data (data available after the release date), so statistically dependent to the number of views: in a sort of way, we could see it as an estimator for views, and since the score is calculated on success data it makes sense that this quantity greatly improves accuracy. This concept is better shown by Fig. 3.23, that shows the scatter plot of the



(a) Scatter plot of prediction vs actual value with Linear regression.



(b) Scatter plot of prediction vs actual value with Random Forest regression.



(c) Scatter plot of prediction vs actual value with LGBM regression.

Figure 3.22: Different scatter plots for the three models applied: Linear regression, Random forest regression and LGBM regression. Notice how the R2 score and mean square error are worse for 3.22a compared to the other two methods. This explains the very sloppy displacement of points around the optimal value, yielding a much more complex relation between the score and its features. Instead, 3.22b and 3.22c are much more accurate, with this last one showing slightly better values for the two parameters.

YouTube views with the number of comments on release date, highlighting how these two quantities are heavily correlated.

	Audio Feature	+ Playlist feature	+ Context Spotify	+ Context YouTube	+ Reaction features (Comments only)
R^2 score	0.001	0.03	0.05	0.54	0.82
Mean square metric	1.62	1.58	1.55	0.75	0.29
F_1 score	0.76	0.77	0.77	0.84	0.92
AUC metric	0.51	0.54	0.54	0.70	0.87

Table 3.2: Accuracy and error on the prediction of the score (continous and discrete). As we can see, big improvements appear when the model is trained considering YouTube context.

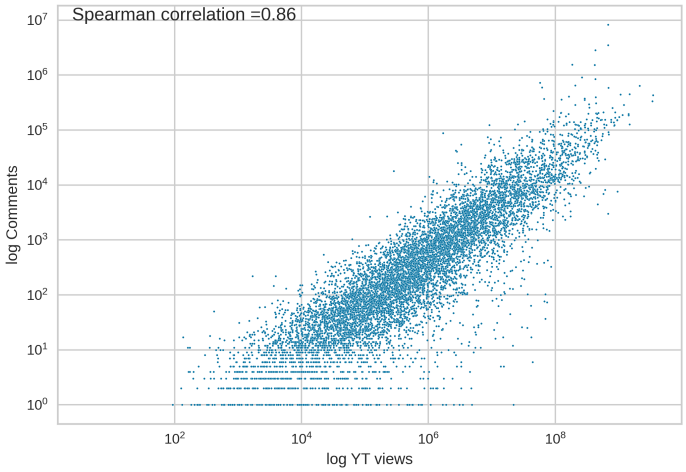


Figure 3.23: Scatter plot of the YT views of each song vs its number of comments in log-log scale. Notice how the two quantities show a great statistical correlation.

Chapter 4

Discussion and conclusion

The music industry is undergoing a great change in the conception of music production, led by streaming (Spotify, YouTube, etc.) and music video platforms (YouTube). In this new scenario, a lot of new data about music consumption has surfaced, creating the premises for a more data-driven approach to this field. In this study, we focused on the analysis of a genre-emerging pattern for temporal publication data and on the prediction of success. For the first objective, we obtained no correlation between song's release date and their genre, even though we found that a higher number of songs was released in certain times of the year (i.e. February/March and September/October). For the second objective, we were able to produce a model that could predict the success score of a song with good accuracy and noted that, for YouTube ordered views, the YouTube-contextual information presented was relevant to the refinement of the prediction, thus showing the influence of video counts and channel subscribers to the view number for a song's video.

Our findings entails two results: first, we defined a metric of success that discounts the contribution of an artist's past performance; second, we assessed the predictive potential of track features grouped by availability over time, thus providing a reference of how predictable success is during the life cycle of a song.

Future research could establish how the prediction of success could be improved before the release date of a song. For instance, more audio features (presently obtained from Spotify API) could be engineered to provide a better description of the sound qualities, including, for example, the presence of specific melodic patterns (e.g. the millennial whoop), instrumental devices (e.g. auto-tune), or even lyrics characteristics.

Moreover, information about music advertisement and promotion, especially on social media, would likely provide insights about the success score. Social networks have gained a big influence on society in the last years and have enabled artists to establish a more direct relationship with their the public. TikTok, in particular, is regarded as the platform where many new talents emerge and songs become popular together with their viral video. Therefore, quantifying how often a track is embedded in videos, or how many citations a song (or its

artist) received in social media, could be relevant to improve success prediction before the publication date.

List of Figures

List of Tables

Bibliography

[1] Seungkyu Shin and Juyong Park. On-chart success dynamics of popular songs. *Advances in Complex Systems*, 21(03n04):1850008, 2018.

[2] Khaled Boughanmi and Asim Ansari. Dynamics of musical success: A machine learning approach for multimedia data fusion. *Journal of Marketing Research*, 58(6):1034–1057, 2021.

[3] Paulo Ferreira, Derick Quintino, Bruna Wundervald, Andreia Dionísio, Faheem Aslam, and Ana Cantarinha. Is brazilian music getting more predictable? a statistical physics approach for different music genres. *Physica A: Statistical Mechanics and Its Applications*, 583:126327, 2021.

[4] Myra Interiano, Kamyar Kazemi, Lijia Wang, Jienian Yang, Zhaoxia Yu, and Natalia L Komarova. Musical trends and predictability of success in contemporary songs in and out of the top charts. *Royal Society open science*, 5(5):171274, 2018.

[5] Yizhao Ni, Raul Santos-Rodriguez, Matt Mcvicar, and Tijl De Bie. Hit song science once again a science. In *4th International Workshop on Machine Learning and Music*. Citeseer, 2011.

[6] Eva Zangerle, Michael Vötter, Ramona Huber, and Yi-Hsuan Yang. Hit song prediction: Leveraging low-and high-level audio features. In *ISMIR*, pages 319–326, 2019.

[7] Rutger Nijkamp. Prediction of product success: explaining song popularity by audio features from spotify data. 2018.

[8] Jianyu Fan and Michael Casey. Study of chinese and uk hit songs prediction. In *Proceedings of the International Symposium on Computer Music Multidisciplinary Research*, pages 640–652, 2013.

[9] Agha Haider Raza and Krishnadas Nanath. Predicting a hit song with machine learning: Is there an apriori secret formula? In *2020 International Conference on Data Science, Artificial Intelligence, and Business Analytics (DATABIA)*, pages 111–116. IEEE, 2020.

[10] R Rajyashree, Anmol Anand, Yash Soni, and Harshitaa Mahajan. Predicting hit music using midi features and machine learning. In *2018 3rd International Conference on Communication and Electronics Systems (ICCES)*, pages 94–98. IEEE, 2018.

[11] Adewale Adeagbo. Predicting afrobeats hit songs using spotify data. *arXiv preprint arXiv:2007.03137*, 2020.

[12] Dorien Herremans, David Martens, and Kenneth Sörensen. Dance hit song prediction. *Journal of New Music Research*, 43(3):291–302, 2014.

[13] Kai Middlebrook and Kian Sheik. Song hit prediction: Predicting billboard hits using spotify data. *ArXiv*, abs/1908.08609, 2019.

[14] Li-Chia Yang, Szu-Yu Chou, Jen-Yu Liu, Yi-Hsuan Yang, and Yi-An Chen. Revisiting the problem of audio-based hit song prediction using convolutional neural networks. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 621–625. IEEE, 2017.

[15] Sumiko Asai. Factors affecting hits in japanese popular music. *Journal of Media Economics*, 21(2):97–113, 2008.

[16] Bang-Dang Pham, Minh-Triet Tran, and Hoang-Long Pham. Hit song prediction based on gradient boosting decision tree. In *2020 7th NAFOSTED Conference on Information and Computer Science (NICS)*, pages 356–361. IEEE, 2020.

[17] Carlos Vicente Soares Araujo, Marco Antônio Pinheiro de Cristo, and Rafael Giusti. Predicting music popularity using music charts. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 859–864, 2019. doi: 10.1109/ICMLA.2019.00149.

[18] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[19] Yingfan Wang, Haiyang Huang, Cynthia Rudin, and Yaron Shaposhnik. Understanding how dimension reduction tools work: An empirical approach to deciphering t-sne, umap, trimap, and pacmap for data visualization. *J. Mach. Learn. Res.*, 22(201):1–73, 2021.

[20] Helen Gibson, Joe Faith, and Paul Vickers. A survey of two-dimensional graph layout techniques for information visualisation. *Information visualization*, 12(3-4):324–357, 2013.

[21] Pankaj Mehta, Marin Bukov, Ching-Hao Wang, Alexandre GR Day, Clint Richardson, Charles K Fisher, and David J Schwab. A high-bias, low-variance introduction to machine learning for physicists. *Physics reports*, 810:1–124, 2019.

[22] Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 160–172. Springer, 2013.

[23] Ricardo JGB Campello, Davoud Moulavi, Arthur Zimek, and Jörg Sander. Hierarchical density estimates for data clustering, visualization, and outlier detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 10(1):1–51, 2015.

[24] Miin-Shen Yang, Chien-Yo Lai, and Chih-Ying Lin. A robust em clustering algorithm for gaussian mixture models. *Pattern Recognition*, 45(11):3950–3961, 2012.

[25] Xu Cheng, Cameron Dale, and Jiangchuan Liu. Statistics and social network of youtube videos. In *2008 16th Interntional Workshop on Quality of Service*, pages 229–238. IEEE, 2008.

[26] Lada A Adamic. Zipf, power-laws, and pareto-a ranking tutorial. *Xerox Palo Alto Research Center, Palo Alto, CA, http://ginger. hpl. hp. com/shl/papers/ranking/ranking. html*, 2000.

[27] Bogumił Kamiński, Michał Jakubczyk, and Przemysław Szufel. A framework for sensitivity analysis of decision trees. *Central European journal of operations research*, 26(1):135–159, 2018.

[28] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[29] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

[30] P. Stoica and Y. Selen. Model-order selection: a review of information criterion rules. *IEEE Signal Processing Magazine*, 21(4):36–47, 2004. doi: 10.1109/MSP.2004.1311138.