Will your next song will be a hit?. *

Álvaro Novillo Correas *Universidad Carlos III*

This article aims to uncover the formula behind a song's success by applying various machine learning techniques to classify whether a song will be a hit or not based on Spotify API parameters (*Spotify Developer Documentation*, N.d.) extracted for the top 1000 songs from 2023 in Spotify. Additionally, we scraped some chord progressions from Ultimate Guitar Guitar using a custom Python scraper to unveil the musical secret behind the top chart songs.

Keywords: pandoc, r markdown, knitr

Introduction.

In the previous section of the project, we explored how to extract the mood of a song using the Spotify's Spotify API parameters. To complete this analysis, we will now analyse the top 1000 songs on Spotify in 2023. In addition to the API parameters from the previous section, new impact variables such as streaming and the number of playlists where the songs are present have been added. We have added some musical features that were not present in the previous section, such as the key and mode of the song, and some chord progressions from the chorus. These progressions were extracted from Ultimate Guitar using a custom scraper created in Python[1].

The scraper searches for the song on Ultimate Guitar, retrieves the chords from the chorus, transposes them to C for easy progression extraction, and extracts the progression from the chords (as chord degrees).

To have a better understanding of the analysis performed in the article, we need to have first some basic chord theory understanding:

We all are familiar with the C major scale, that is C – D – E – F – G – A – B – C (or do-re-mi-fa-sol-la-si-do). If we take a look at the distance each note inside the scale have from each other in a keyboard, we will have the following equation:

$$W \check{} W \check{} H \check{} W \check{} W \check{} W \check{} H$$

Where **H** denotes a *half-step*, which is is the distance between two adjacent keys on a piano keyboard, and it is the smallest musical interval in music. And **W** is the distance equal to two half steps, known as a *whole step*.

If we take a closer look, There are seven different notes in each scale. Scale degree refers to the position of each scale note. Each scale note has a scale degree name. For example, C is the first scale degree of C major scale. Stacking two *generic third* notes above each scale note, we obtain the chords inside a scale, also known as **Diatonic Chords**

*Replication files are available on the author's Github account (https://github.com/AlvaroNovillo/music_mood). **Current version**: marzo 08, 2024; **Corresponding author**: novillocorreasalvaro@gmail.com.

[1]The scrapper can be found inside `chordParser.py`. You can download on my GitHub page., inside the folder `UltimateGuitarChordParser`

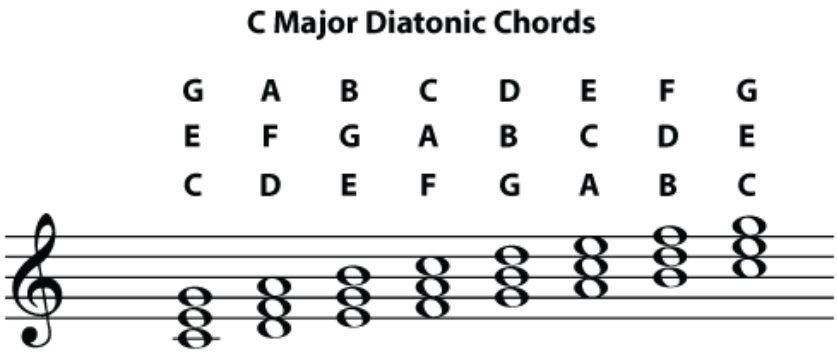

Figure 1: C Major diatonic chords.

Each diatonic chord is labelled with a roman numeral number:

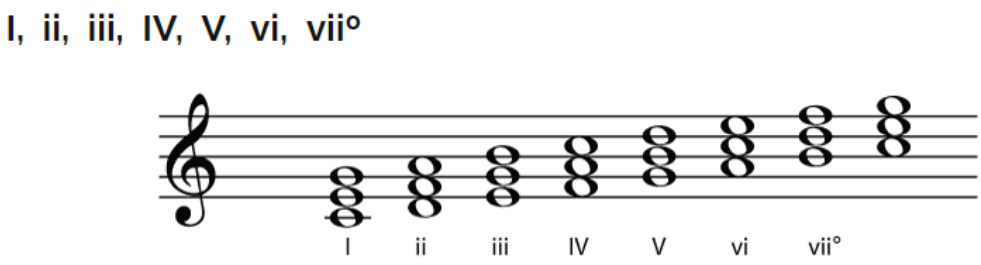

Figure 2: Chord degree in a Major scale.

Where upper roman numeral number refers to a **Mayor** (happy) chord, and lower numeral number refers to a **Minor** (sad) chord Music, among other art forms, has an special ability to evoke emotions, shaping our experiences and perceptions. In this article, we will delve into the intricate relationship between music and emotions, using machine learning techniques to analyze and categorize songs based on their emotional content. The seventh chord in major scale, labeled as vii^o is always be a **diminished** chord

For any minor key, the chord degrees are as follows:

$$\text{i} \ - \text{ii}^o - \text{III} - \ \text{iv - v - VI} \ - VII$$

Where now the second (ii^o) is a **diminished** chord

Table 1: Top 1000 songs in Spotify. Sorted from the most streamed to the least of them

	track_name	year	month	day	playlists	charts	streams	tempo	key	mode	danceability	valence	energy	acousticness	instrumentalness	liveness	speechiness
56	Blinding Lights	2019	11	29	43899	69	1.0000000	171	C#	Major	50	38	80	0	0	9	7
180	Shape of You	2017	1	6	32181	10	0.9618372	96	C#	Minor	83	93	65	58	0	9	8
87	Someone You Loved	2018	11	8	17836	53	0.7795150	110	C#	Major	50	45	41	75	0	11	3
621	Dance Monkey	2019	5	10	24529	0	0.7734538	98	F#	Minor	82	54	59	69	0	18	10
42	Sunflower - Spider-Man: Into the Spider-Verse	2018	10	9	24094	78	0.7581469	90	D	Major	76	91	50	54	0	7	5
163	One Dance	2016	4	4	43257	24	0.7327212	104	C#	Major	77	36	63	1	0	36	5

Exploratory Data Analysis

As we did in the last part, we will take first take a look at all the variables obtained from the Spotify API, Fig. 3

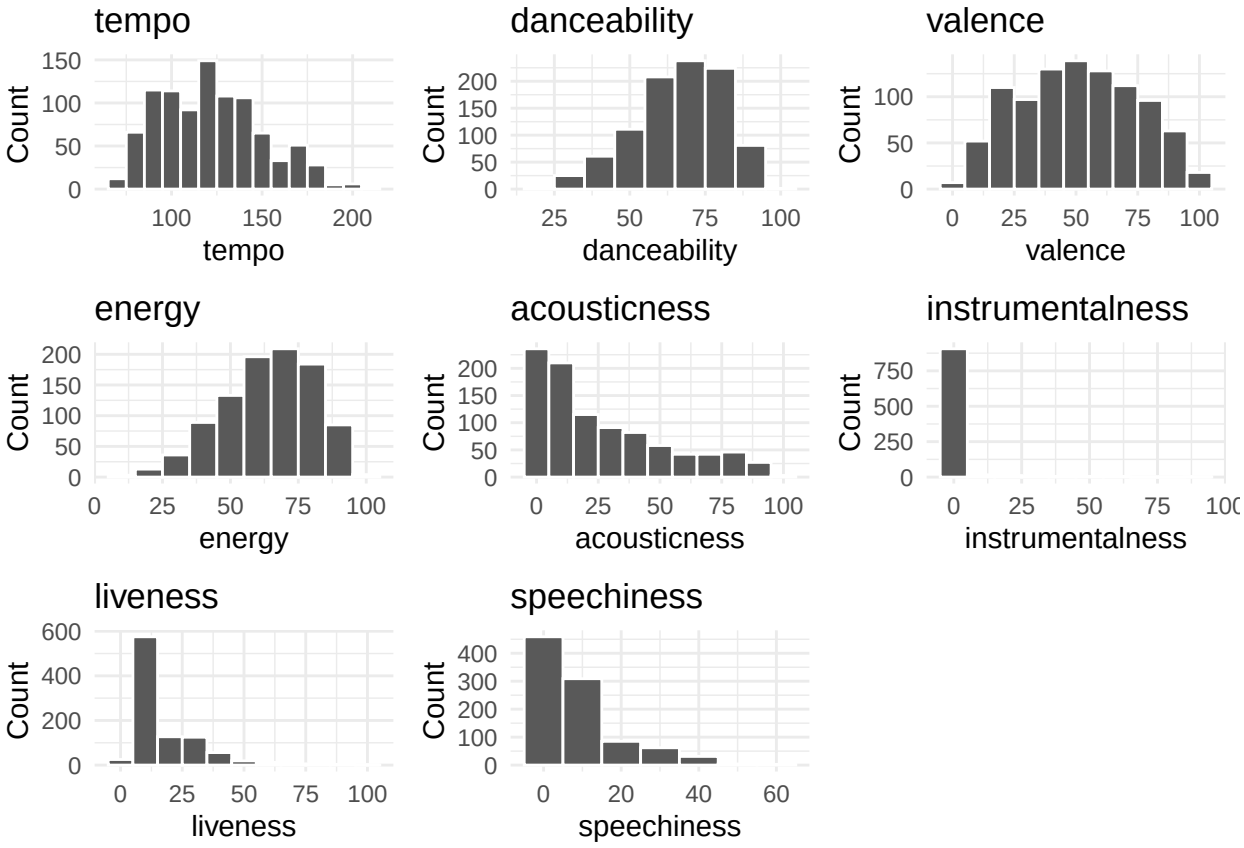

Figure 3: Histogram of the Spotify API features

Most of the features seems to follow either a normal distribution or a log-normal distribution, as we observed in the dataset used in part 1. We can see how the different values of this variables are related with the number of streams, Fig. 4

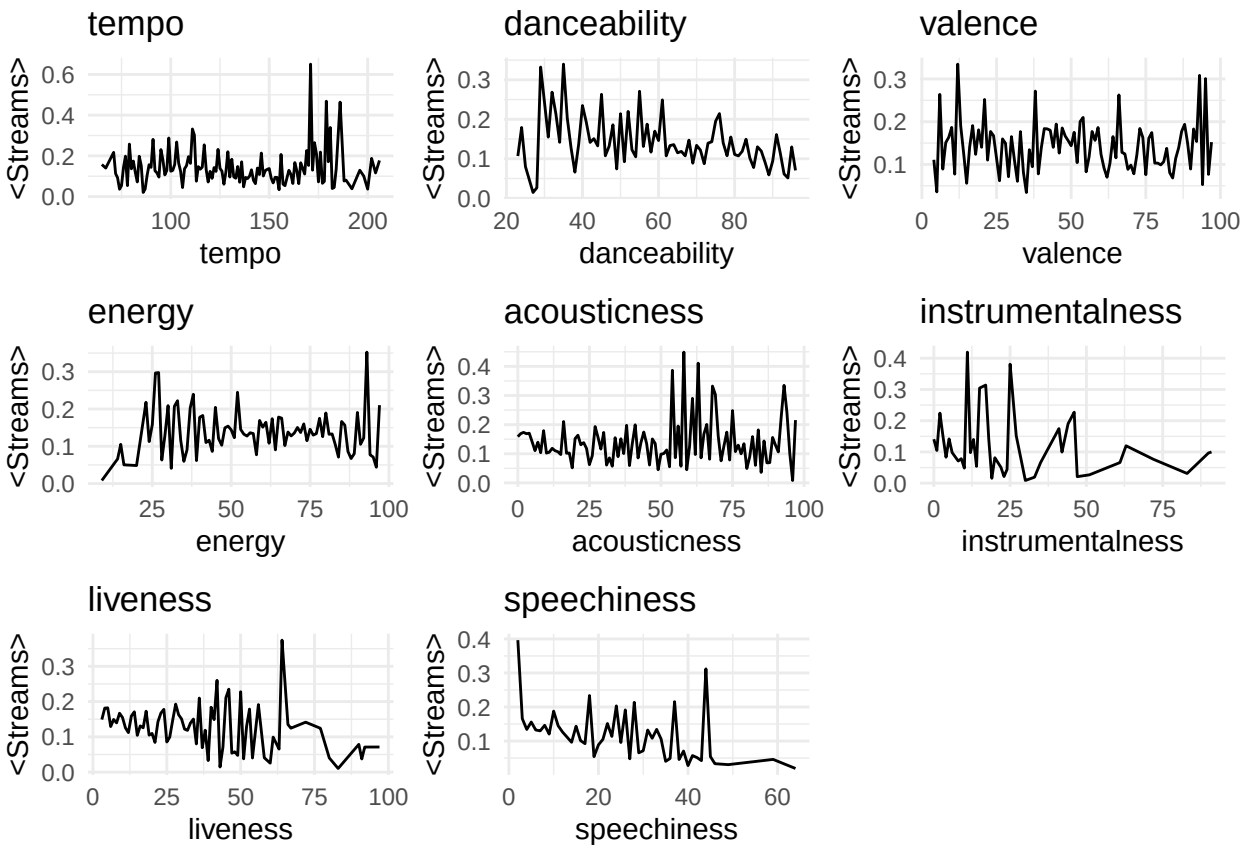

Figure 4: Average value of streams for each value of the different features extracted from the Spotify API.

Inspecting Fig. 4, we can see that `tempo` has a great impact in the amount of streams each song has. The most popular songs seems to be energetic, with high BPMs and adequate danceability. We can see this patterns in the top streamed songs showed in Table 1.

Now, lets see how different keys and modes relates with the amount of streams a song has.

Table 2: Top 3 keys with the highest average streams

key	mean_streams
C#	0.16
E	0.16
A#	0.15
D#	0.15

As we can see in the table 2, C# is the key with the higher average percentage of streams, followed by E and D#. These results are quite impressive, as it has always been said that both G and C are the most popular keys to use when composing a song, but that doesn't mean that the most popular songs are written in these keys.

About the modality of such keys, it seems that major songs seems to accumulate more streams

Table 3: average streams for each mode

mode	mean_streams
Major	0.14
Minor	0.13

than minor songs, Table 3. Note also that there are more Major songs (549) in the dataset than minor (403).

To start the EDA, we need first to merge the chords we were able to scrape from *Ultimate Guitar* to their corresponding songs. The scraper was able to characterize correctly the chord progressions for N songs from the dataset.

After an initial inspection of the dataset, we will apply dimensionality reduction tecniques in order to capture different groups to classify our songs into.

Multidimensional scaling (MDS)

Multidimensional scaling (MDS) is a dimensionality reduction technique that visualises the pairwise dissimilarity or similarity between data points. It is particularly useful for datasets containing both qualitative and quantitative data, as MDS can handle different types of input distances, including those based on categorical variables.

Unlike Principal Component Analysis (PCA), which works well with quantitative variables, MDS is versatile and applicable to mixed data sets. MDS results provide a low-dimensional representation that preserves the original dissimilarity or similarity structure, making it valuable for revealing patterns, clustering and interpreting relationships in diverse datasets with a mix of variable types.

The `dist` function in R uses Euclidean distance by default for numerical variables. However, when categorical variables are present, dist converts them to binary indicators (dummy variables) and calculates dissimilarity based on these indicators. The Jaccard distance, which measures the dissimilarity between two sets, is commonly used for binary data.

Alternatively, Gower's distance is designed to handle mixed data types and offers more flexibility in handling both numeric and categorical variables. The `daisy` function in the `cluster` package is often used for this purpose.

We choose to use Gower's distance rather than Euclidean distance in the context of mixed data because Gower's distance is specifically designed to handle datasets that contain a combination of numeric, categorical and ordinal variables. When working with different types of variables, such as continuous measures, categorical labels or ordinal rankings, traditional distance measures such as Euclidean may not be appropriate due to their assumptions about data types..

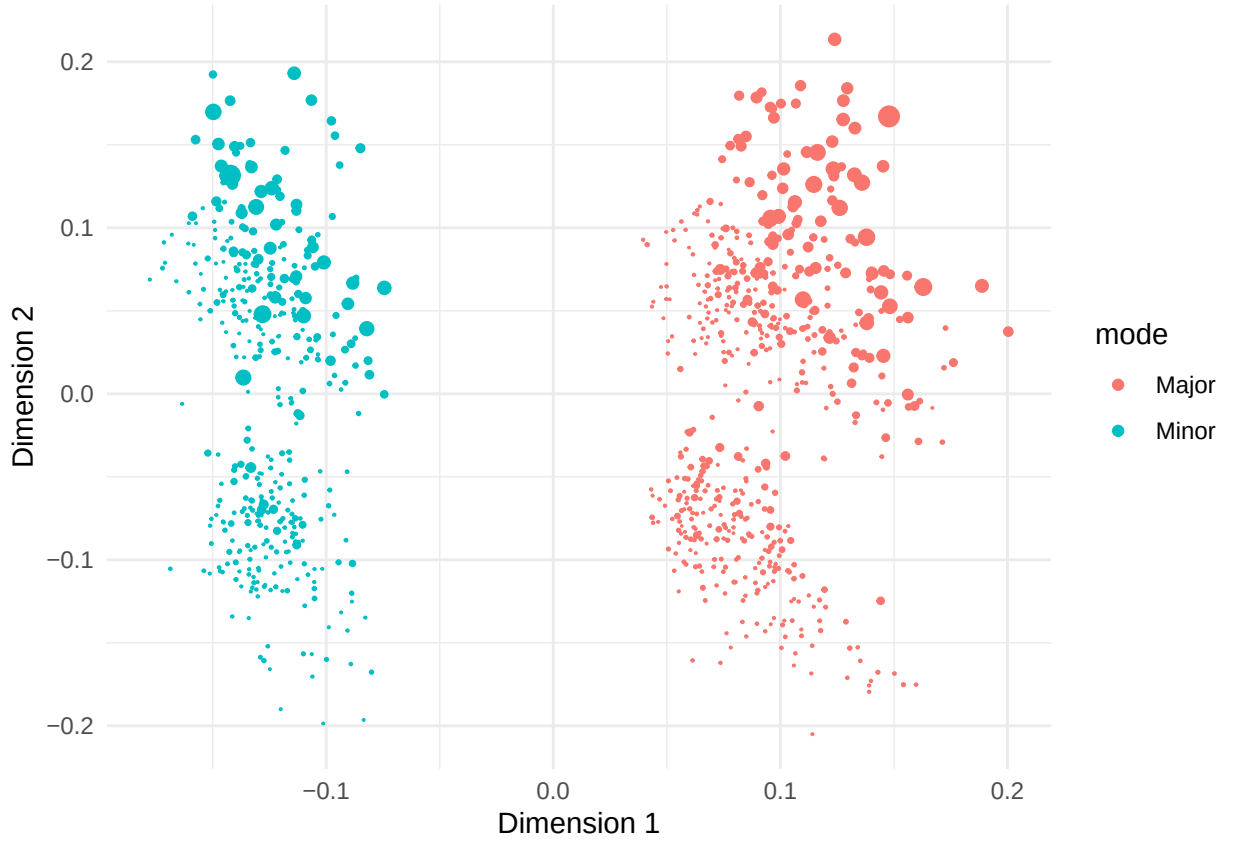

Figure 5: MDS using Gower's distance Scatterplot. The size of the points is proportional to the number of streams that each song has

Gower distance provides a balanced treatment of categorical variables, ensuring that dissimilarity is calculated on the basis of common categories. These attributes make Gower distance a robust and flexible option for measuring dissimilarity, especially in real-world scenarios with heterogeneous data types.

The Gower distance $d_G(x, y)$ between two data points x and y is calculated as

$$d_G(x, y) = \frac{\sum_{i=1}^{n} w_i \cdot s_i(x, y)}{\sum_{i=1}^{n} w_i}$$

where $s_i(x, y)$ is the dissimilarity measure for each variable, w_i is the weight assigned to each variable, and n is the total number of variables. This formula accommodates different variable types and scales, providing a comprehensive dissimilarity metric for mixed data sets.

From Fig. 5 we can see that our dataset is divided into major and minor songs, with those in the upper part having more streams and those in the lower part having fewer.

Now, applying Cluster Analysis, we will extract k different groups to characterize our dataset.

Cluster Analysis

To define the categories in our dataset, K-means clustering will be applied. K-means is an unsupervised machine learning algorithm that aims to partition n data points into k clusters. The

algorithm works by minimizing the sum of squared distances between the data points and their respective cluster centroids.

The process involves the following steps:

- 1. Randomly select k data points as the initial cluster centroids.
- 2. Assign each data point x_i to the nearest cluster centroid μ_j based on the Euclidean distance.

$$d\left(x_i, \mu_j\right) = \sqrt{\sum_{n=1}^{N} \left(x_{i,n} - \mu_{j,n}\right)^2}$$

  where N is the number of dimensions/features 1 .

- 3. The objective of K-means is to minimize the sum of squared distances within each cluster, which can be expressed as:

$$WSS = \sum_{j=1}^{k} \sum_{x_i \in S_j} \left\| x_i - \mu_j \right\|^2$$

  where S_j is the set of data points in cluster j, μ_j is the centroid of cluster j, and k is the total number of clusters 1 .

- 4. The cluster centroids are updated by taking the mean of all data points assigned to that cluster:

$$\mu_j = \frac{1}{\left|S_j\right|} \sum_{x_i \in S_j} x_i$$

  where $\left|S_j\right|$ is the number of data points in cluster j.

Looking at Fig. 6, we can see how the dataset can be characterized into 4 different categories:

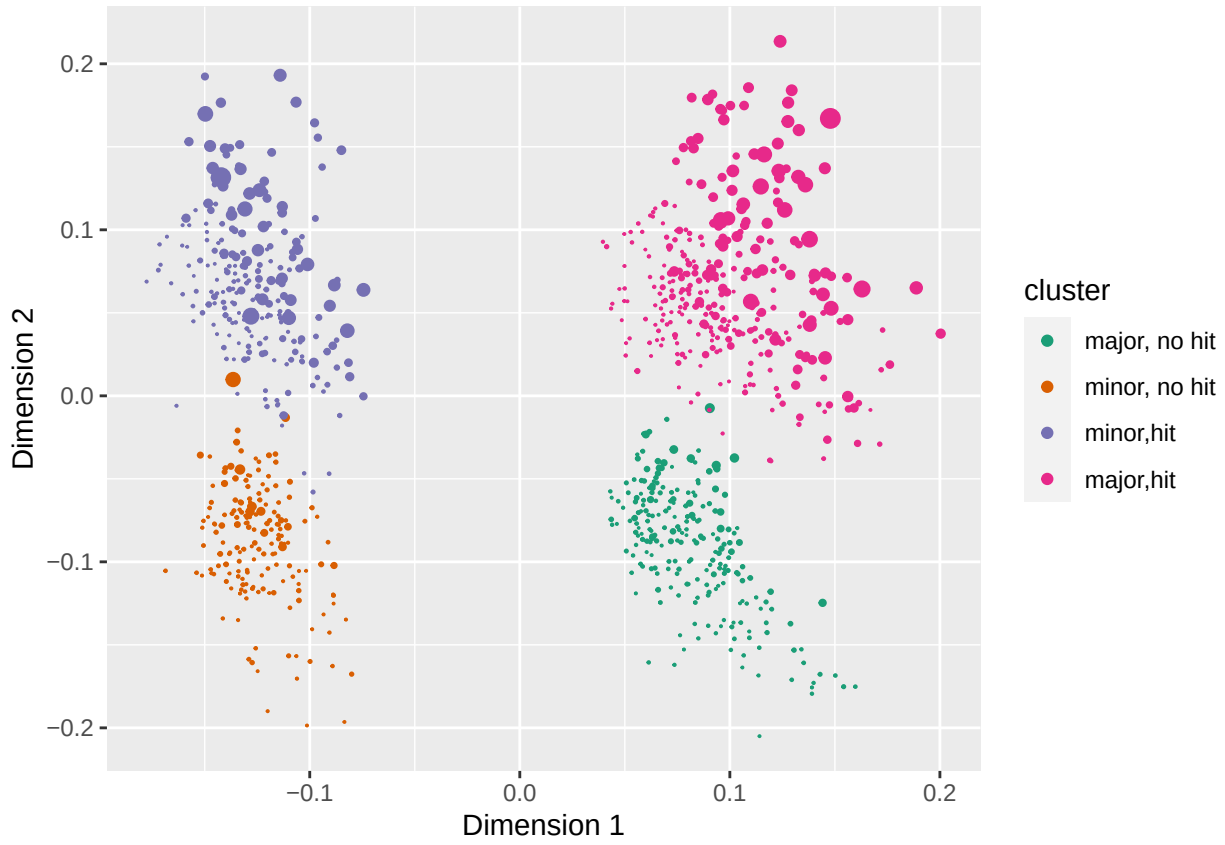

Figure 6: Visualization of the clusters with respect to the Multidimensional Scaling components. We can see how the dataset can be divided in four diferent categories

Table 4: Average streams in each cluster categories

cluster	mean_streams
major,hit	0.19
minor,hit	0.17
major, no hit	0.08
minor, no hit	0.08

The first two categories (located in the upper part of the chart) represent the major and minor songs that have been a hit. We can see that the average streams of these categories are high, being higher for songs with major modes, Table 4. The two lower categories represent the songs that do not have such a big impact.

Now that the dataset has been classified, we can try to extract the most common progressions of each category. With our Python Scraper, we have been able to characterize the chord progressions of the chorus of 188 songs from the dataset, Table 5

Lets inspect the most common chord progressions in each category:

The results presented in Table. 6 provide deep insights into the current music industry. In

Table 5: Chord progressions dataset.

Name	Progression	EndDifferent	NumSectionChords	nonDiatonicChords	extendedChords
vampire	I-IV-V-IV	V-vi-IV	4	0	0
Cruel Summer	I-V-vi-IV		4	0	0
Sprinter	ii-III-vi-ii-vi-I-ii-I-ii-vi-ii		4	1	1
Ella Baila Sola	VI#-ii#-I#		3	3	1
un x100to	IV-iii-vi-vii-I-iii-ii-V		7	1	4
Super Shy	vi		1	0	1

Table 6: Predominant chord progressions in the categories defined

cluster	Progression
major.no.hit	I-IV
minor.no.hit	vi-IV-V
minor.hit	vi
major.hit	I-IV

general, producers aim to compose versatile songs that can work for both active and passive listening. Specifically, degrees I, IV, and VI work particularly well when creating melodies based on the pentatonic scale. This scale allows for the repetition of simple elements that can resonate with the listener and fit into every chord without modifying any notes from the scale. It is important to avoid adding new content to the text, so no new aspects have been introduced. The only dominant that appears (V) does not resolve on the I degree, as is common in classical harmonic progressions, but on the IV degree. This is known as a broken cadence and creates a sense of continuity and growth, which is often found in pop songs that use simple progressions to maintain the listener's interest throughout the song.

After analysing the dataset, we will use Spotify's API features and machine learning algorithms to predict two scenarios: the song's modality and its potential for trending based on the four categories mentioned earlier.

Once we label each observation, we can train our machine learning algorithm to predict these categories using only the Spotify API features.

Methodolody

Prediction of the song modality

To train our models, the dataset was divided into a training set (80%) and a test set (20%) to train our models. Shuffling was necessary as the dataset was ordered from the most streamed song to the least. Our approach to finding the best model involved fitting several models using all numerical features available in our dataset, including 'danceability', 'acousticness', 'energy', 'instrumentalness', 'liveness', 'valence', 'loudness', 'speechiness', and 'tempo'. We will use the `caret` package (Kuhn and Max, 2008), apply 5-fold cross-validation, and center and scale the predictors.

KNN

The KNN classifier predicts the class of a given test observation by identifying the observations that are closest to it. Therefore, the scale of the variables is important. Variables on a large scale will have a greater impact on the distance between observations and, consequently, on the KNN classifier than variables on a small scale.

```
## k-Nearest Neighbors
##
## 763 samples
##   8 predictor
##   2 classes: 'Major', 'Minor'
##
## Pre-processing: scaled (8), centered (8)
## Resampling: Cross-Validated (5 fold, repeated 1 times)
## Summary of sample sizes: 610, 610, 610, 611, 611
## Resampling results across tuning parameters:
##
##   k   ROC        Sens       Spec
##    5  0.5492026  0.6454545  0.4174038
##    7  0.5593990  0.6613636  0.4084135
##    9  0.5600044  0.6318182  0.4296635
##   11  0.5672296  0.6363636  0.4330769
##   13  0.5729280  0.6772727  0.4051923
##   15  0.5683717  0.6750000  0.4177404
##   17  0.5640895  0.6818182  0.3804808
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was k = 13.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Major Minor
##      Major    77    57
##      Minor    32    23
##
##                Accuracy : 0.5291
##                  95% CI : (0.4553, 0.602)
##     No Information Rate : 0.5767
##     P-Value [Acc > NIR] : 0.91859
##
##                   Kappa : -0.0063
##
##  Mcnemar's Test P-Value : 0.01096
##
##             Sensitivity : 0.7064
##             Specificity : 0.2875
##          Pos Pred Value : 0.5746
```

```
##          Neg Pred Value : 0.4182
##              Prevalence : 0.5767
##          Detection Rate : 0.4074
##    Detection Prevalence : 0.7090
##       Balanced Accuracy : 0.4970
##
##        'Positive' Class : Major
##
```

The model does not seem to work at all. Lets try to choose the best threshold, which in this case is 0.43

Table 7: Confusion matrix using the best threshold

	Major	Minor
Major	53	42
Minor	56	38

With an accuracy of 0.48, the model is not being able to classify properly the dataset

#Random Forest

Let's try to apply Random Forest to see if we're able to build a model capable detecting the mode of a song:

```
## note: only 7 unique complexity parameters in default grid. Truncating the grid to 7 .

## Random Forest
##
## 763 samples
##   8 predictor
##   2 classes: 'Major', 'Minor'
##
## Pre-processing: centered (8), scaled (8)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 611, 610, 611, 610, 610
## Resampling results across tuning parameters:
##
##   mtry  ROC        Sens       Spec
##   2     0.5780007  0.7136364  0.3716346
##   3     0.5742078  0.7068182  0.3870673
##   4     0.5657304  0.7250000  0.3962019
##   5     0.5617209  0.7090909  0.4178846
##   6     0.5640805  0.7000000  0.4118269
##   7     0.5638093  0.7181818  0.3993269
##   8     0.5663683  0.7022727  0.4056250
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Major Minor
##      Major    81    53
##      Minor    28    27
##
##                Accuracy : 0.5714
##                  95% CI : (0.4976, 0.643)
##     No Information Rate : 0.5767
##     P-Value [Acc > NIR] : 0.588749
##
##                   Kappa : 0.0841
##
##  Mcnemar's Test P-Value : 0.007661
##
##             Sensitivity : 0.7431
##             Specificity : 0.3375
##          Pos Pred Value : 0.6045
##          Neg Pred Value : 0.4909
##              Prevalence : 0.5767
##          Detection Rate : 0.4286
##    Detection Prevalence : 0.7090
##       Balanced Accuracy : 0.5403
##
##        'Positive' Class : Major
##
```

Just using the Spotify's API features seems to not be enought to get the mode of a song. Lets use SVM with a radial kernel to try again

SVM

```
## line search fails -0.01170497 0.3016774 1.220502e-05 -1.620522e-06 -3.743035e-09 8.092508e-09

## Support Vector Machines with Radial Basis Function Kernel
##
## 763 samples
##   8 predictor
##   2 classes: 'Major', 'Minor'
##
## Pre-processing: centered (8), scaled (8)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 611, 610, 611, 610, 610
## Resampling results across tuning parameters:
##
##   C          ROC        Sens       Spec
##        0.25  0.5703497  0.9113636  0.127115385
```

```
##        0.50  0.5710959  0.9204545  0.111634615
##        1.00  0.5731294  0.9159091  0.105000000
##        2.00  0.5809408  0.9750000  0.012307692
##        4.00  0.5449902  0.9886364  0.015384615
##        8.00  0.5742674  0.9954545  0.015480769
##       16.00  0.5775923  0.9522727  0.061538462
##       32.00  0.5853764  0.9772727  0.018509615
##       64.00  0.5820067  0.9954545  0.018653846
##      128.00  0.5380278  0.9863636  0.024615385
##      256.00  0.5654321  0.9954545  0.000000000
##      512.00  0.5550945  0.9909091  0.015480769
##     1024.00  0.5149596  1.0000000  0.000000000
##     2048.00  0.5388751  0.9886364  0.009230769
##     4096.00  0.5480840  0.9863636  0.018701923
##     8192.00  0.5517739  0.9909091  0.012451923
##    16384.00  0.5417772  0.9909091  0.009375000
##    32768.00  0.5557214  0.9744318  0.054026442
##    65536.00  0.5085741  0.9454545  0.089615385
##   131072.00  0.5396820  0.9818182  0.009326923
##
## Tuning parameter 'sigma' was held constant at a value of 0.1316823
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.1316823 and C = 32.

## Confusion Matrix and Statistics
##
##           Reference
## Prediction Major Minor
##      Major    96    68
##      Minor    13    12
##
##                Accuracy : 0.5714
##                  95% CI : (0.4976, 0.643)
##     No Information Rate : 0.5767
##     P-Value [Acc > NIR] : 0.5887
##
##                   Kappa : 0.0338
##
##  Mcnemar's Test P-Value : 1.973e-09
##
##             Sensitivity : 0.8807
##             Specificity : 0.1500
##          Pos Pred Value : 0.5854
##          Neg Pred Value : 0.4800
##              Prevalence : 0.5767
##          Detection Rate : 0.5079
##    Detection Prevalence : 0.8677
##       Balanced Accuracy : 0.5154
```

```
##
##        'Positive' Class : Major
##
```

As we can see, just using the Spotify API data, we can't predict the mode of a song. We've tried also to predict the classes defined using MDS

Can we predict if the song is going to be a hit just using Spotify's API data?

KNN

Let's try first the KNN algorithm

```
## k-Nearest Neighbors
##
## 763 samples
##   8 predictor
##   4 classes: 'major.no.hit', 'minor.no.hit', 'minor.hit', 'major.hit'
##
## Pre-processing: centered (8), scaled (8)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 610, 611, 609, 612, 610
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    5  0.2908849  0.03011681
##    7  0.2869637  0.02206214
##    9  0.3117148  0.05288423
##   11  0.3275571  0.07039551
##   13  0.3209692  0.05853346
##   15  0.3184147  0.05299761
##   17  0.3314867  0.07179485
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 17.
```

```
## Confusion Matrix and Statistics
##
##               Reference
## Prediction     major.no.hit minor.no.hit minor.hit major.hit
##   major.no.hit           10            8        10        16
##   minor.no.hit            8            3         5         2
##   minor.hit               8            6        11        15
##   major.hit              20           17        20        30
##
## Overall Statistics
##
##                Accuracy : 0.2857
##                  95% CI : (0.2225, 0.3558)
```

```
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : 0.93017
##
##                   Kappa : 0.0097
##
##  Mcnemar's Test P-Value : 0.03831
##
## Statistics by Class:
##
##                      Class: major.no.hit Class: minor.no.hit Class: minor.hit
## Sensitivity                      0.21739             0.08824           0.2391
## Specificity                      0.76224             0.90323           0.7972
## Pos Pred Value                   0.22727             0.16667           0.2750
## Neg Pred Value                   0.75172             0.81871           0.7651
## Prevalence                       0.24339             0.17989           0.2434
## Detection Rate                   0.05291             0.01587           0.0582
## Detection Prevalence             0.23280             0.09524           0.2116
## Balanced Accuracy                0.48981             0.49573           0.5182
##                      Class: major.hit
## Sensitivity                    0.4762
## Specificity                    0.5476
## Pos Pred Value                 0.3448
## Neg Pred Value                 0.6765
## Prevalence                     0.3333
## Detection Rate                 0.1587
## Detection Prevalence           0.4603
## Balanced Accuracy              0.5119
```

The initial results are not promising, as expected. We will now attempt to fit a variety of models using AutoML to determine if any can accurately classify the defined classes.

AUTOML approach

```
library(h2o)
h2o.init()
h2o.no_progress()
h2o_data <- as.h2o(numeric_data)
splits <- h2o.splitFrame(data = h2o_data, ratios = 0.8, seed = 42)

train_mni <- splits[[1]]
test_mni <- splits[[2]]
aml_mni <- h2o.automl(x = c("danceability", "acousticness", "energy", "instrumentalness", "liven
```

Using AutoML, the best model found is a StackEnsemble of GBM,GLM, and DRF, with an average accuracy of 0.32. Thus, there is not any suitable model to classify the songs just using Spotify's API data.

Conclusion

We've proved in the article that Spotify's API data alone cannot predict a song's modality or potential popularity. However, patterns have been identified in the most streamed songs. Songs written in a major key appear to be better received by users, with those written in C# receiving the highest average number of streams. The most common chord progression found among the most streamed major songs is

$$I - IV$$

Which is commonly used in popular pop songs. Also, top streamed songs tend to be energetic, with moderate tempo (the average tempo of the top 10 most streamed songs is 117, with a 59% of energy).

Although our analysis emphasizes the importance of certain musical elements, such as modality and chord progressions, in determining a song's popularity on Spotify, it is important to acknowledge that these factors only account for a portion of a song's success. Harmony is a fundamental aspect, comprising approximately 15-20% of the overall composition. However, a song's appeal and potential for success are ultimately determined by the intricate interplay of melody, rhythm, instrumentation, and other factors. Therefore, while it is noteworthy that major key songs and specific chord progressions are prevalent among the most streamed tracks, success in the music industry remains multifaceted and unpredictable. There are no guarantees based solely on these elements.

References

Kuhn and Max. 2008. "Building Predictive Models in R Using the caret Package." *Journal of Statistical Software* 28(5):1–26.
URL: *https://www.jstatsoft.org/index.php/jss/article/view/v028i05*

Spotify Developer Documentation. N.d. https://developer.spotify.com/documentation/web-api. Accessed: February 19, 2024.