HOW TO PREDICT THE HIT POTENTIAL OF A SONG

Marine JACQUEMIN-LORRIAUX

MSc Data Analytics & Artificial Intelligence

Promotion 2018-2019

EDHEC Business School

Master Thesis Director: Christophe Croux

Abstract

Every week, thousands of songs are released on the market, but only a small fraction of those encounters great and popular success. Foster by the streaming platforms which revolutionized music consumption in less than a decade, the competition is harsh.

This master thesis aims at predicting the hit potential of a song. Based on data related to audio and meta contents exported from Spotify web API, I first focus on understanding how music has evolved over time and the analysis of the predictor variables. I then work on a classification problem, test and compare a set of algorithms depending on defined performance criteria to find the best model.
In this master thesis, I developed my own coding notebooks in both R and Python to perform ETL (extract, transform, load), text analysis and predictive modeling.

From the ever-changing business model, to the unprecedented influence of rap and hip-hop movements, this paper demonstrates that it is possible to successfully respond to listeners changing tastes, trends and innovation, using Data Science.
Being able to perform accurate predictions is a must at a time when music consumption relies on individual behaviors and recommendation systems.

Keywords: Predictive modeling, Data Science, Machine Learning, Data Analytics, Classification, R, Python, Music Industry, Streaming

Acknowledgments

First of all, my gratitude goes to EDHEC Business School for giving me the opportunity to find my way and offering me support and countless resources to achieve my professional goals.

I am grateful to Florian Pelgrin for having created a challenging MSc in Data Analytics & Artificial Intelligence, which sharpened my desire to continue in this field and helped me to develop the tools I need to be an effective Data Scientist.
I would also like to thank Christophe Croux, my thesis supervisor, for the time he took directing this thesis.

I would like to thank all the people involved in the realization of this thesis, people who showed interest to the subject, gave me insights and encouraged me.
Finally, my deepest gratitude goes to my mother, who has always provided me with continuous and unconditional support.

I – Introduction

"The club isn't the best place to find a lover, so the bar is where I go." From the first notes, anyone can recognize the great worldwide music success of Ed Sheeran. First song to hit the 2 billion streams, "Shape of you" broke all the Spotify records for steams. It took only a few months for this song to become the most-streamed song of all time on the streaming platform. Originally conceived for Rihanna, what made this song an instant success? Is it the pop, dancehall song written in the key of C♯ minor with a tempo of 96 beats per minute? The lyrics about a budding romance? Or the meeting of Ed Sheeran's creative hand and sensibility? When it comes to music, there is a bunch of parameters to take into account.

A massive volume of track is released daily, and yet only a tiny fraction of them will meet a popular success. Today, the hit of a track is measured by the number of streams on the streaming platforms, which completely revolutionized the way we consume music in less than a decade. Over the years, the notion of ownership has progressively disappeared, replaced by the unlimited stream of music in exchange for a monthly fee. Such a business model fosters a harsh completion to catch the listener's attention, at a time when we are used and encouraged to swipe, scroll and skip at a fingertip.

In a fast and ever-changing musical ecosystem, how to predict the hit potential of a song? Throughout this study, I try to pierce the secrets of a hit song receipt. From Data collection to exploration and modeling, I use a Data Science methodology to work on the topic. The overall objective is to understand the relevant features impacting the business and find an accurate predictive model that could bring real added-value in its business applications.

Prior to working on predictive modeling, I spent time on understanding the evolution of the music industry and its business model, impacted by cultural changes and technological evolutions throughout the years. Then, I focused on the current hegemony of the streaming platforms, especially Spotify, and set up a data collection strategy to build an appropriate dataset. Once the

data ready, I dedicated an entire section to the features understanding. The objective of this section is to study the music evolution through six decades, find trends and insights that could help giving a proper definition of a hit song and define a time scope for the data training. The last sections follow a data science methodology, with the definition of my classification problem, the theoretical explanation of the models I used and trained, the results and comparison of the algorithms' outputs, and finally discussion about the potential business applications, improvements, and limits of such predictive models.

II – State of play of the music industry

II.1 – History of the music industry: from phonograph to streaming

II.1.1 - A serie of innovations

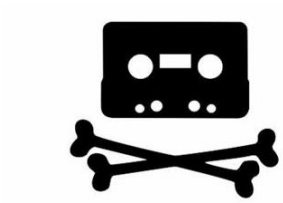
Back to 1877, Thomas Edison invented a machine that could both record and play music thanks to a thin metal cylinder. The phonograph was born.
Over the next decades the phonograph took different forms, from the cylinder shape to flat disk, until the vinyl appeared in 1948, and later one the taps. Music became mobile and by the late 60's, most cars had track players.

The Japanese brand Sony rocked the game in the 80's with the launch of the Walkman giving a whenever wherever power to listeners. This period was also marked by the launch of MTV, music labels and the marketing of artists. More than audio content, music became a whole show with pioneers like Madonna or Mickael Jackson. During the 90's the sales of CD's were booming and music industry was at its best.

Figure 1: Evolution of music industry over 150 years

II.1.2 - The growing challenge of online music piracy

Before the rise of Internet, the piracy issues were quite easily tackled by laws, as the materiality of the copied products made it easily detectable. But in the early 2000's the definition of piracy with the globalization of internet connection at home. The 2000's saw the rise of peer-to-peer (P2P) file sharing platforms and illegal downloading of mp3 files. Software like Napster, with which the area of piracy started, succeeded to convince millions of people.



This trend drastically affected the sales of CDs. Although long legal battles and sites shutdowns, the damage was done. Sharing music over the internet became a standard and, foster by the rise of the digital areas, was here to stay.

In 2001, Apple iPods and mp3 players introduced the legal digital downloads but the sales of audio content only couldn't be a sustainable model for the music industry. More tours and marketing around the artist are now the main revenue stream.

What if the response to piracy and lost in revenue was streaming? The Swedish platform Spotify has been the first to offer users a whole collection of music, instead of buying and owning the songs, in exchange for a monthly charge or advertisement. A bunch of competitors arose over the last few years and a change in behavior due to streaming low cost and ease of use nearly put an end to music online piracy.

II.1.3 - Streaming dominates the music industry

Over the years, the way we consume music quickly evolved. People do not want to own music anymore but stream it.

A recent study led by RIIA (Recording Industry Association of America) found out that three quarters of the music industry revenues were generated by streaming platforms. With a growth of 30% year-to-year, streaming is the new standard for music industry to be sustainable.

Streaming is now the fastest growing segment of the music industry. According to Goldman Sachs, streaming services are expected to be worth $37bn by 2030.

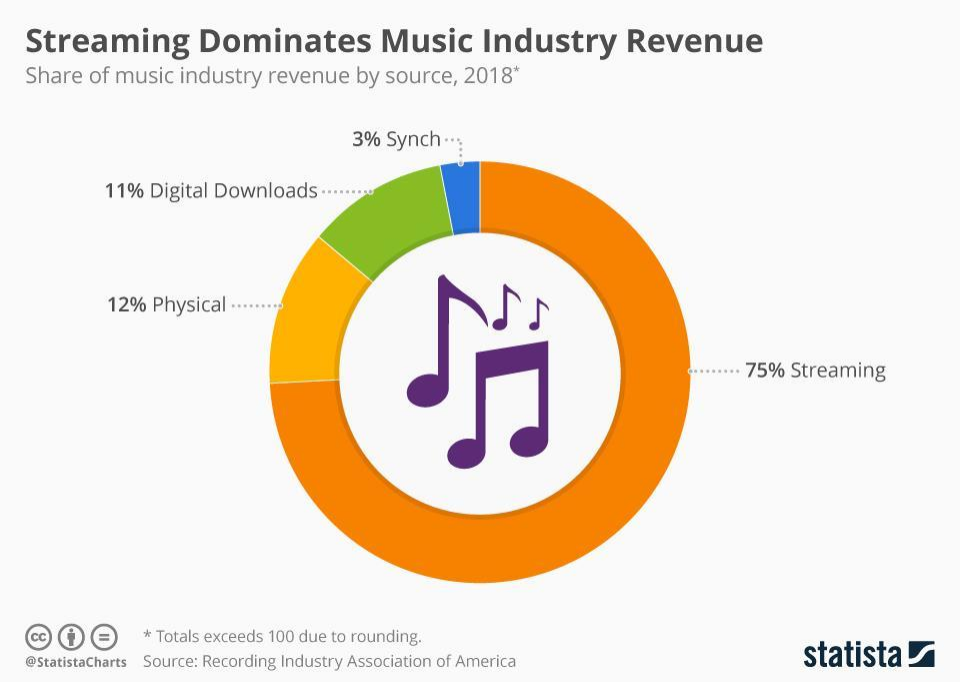


Figure 2: Music Industry Revenues Streams

II.2- Freemium models: Spotify and the new actors

II.2.1 - Freemium model: a new way of consuming music

Freemium is a pretty new economic model, based on both free and premium services. The goal is to make money based on a free service. This business model relies on a mix between a free and a premium access, for which you have to pay in exchange for a high quality service. The strategy is focused one the conversion rate: convert « free » users to « premium » users.

This model is particularly adapted to web-based companies, as you need a huge number of users to convert to be profitable. Skype is one of the pioneer when it comes to freemium, and well-known among expats. The platform delivers a free communication service between computers, but there is a charge as soon as the user makes calls from a computer to a mobile phone. This split offer gives a conversion rate lower than 10%, but this is how Skype makes money.

LinkedIn is another example of a freemium-based economic model. You can create a profile and add contact for free, and the professional social network charges you to access more advanced functionalities.

What makes freemium a successful model? Despite the large number of users, the premium service proposed by the company has to bring significant value. In the music streaming platforms industry, the premium service generally includes a limitless access to high quality music with no advertisement. The strategy to convert free to premium is optimized by « annoying » free users with a lot of ads between songs and offer a restrictive service.

Let's then calculate the conversion rate of Spotify: Over 207 million monthly active users, 96 million of these are Premium subscribers, which makes huge a conversion rate of 46%, far away from the usual rates of freemium models.

II.2.2 - Spotify: the world's biggest music streaming platform

II.2.2.1 - History of an instant success

Founded in 2006 in Stockholm by Daniel Ek and Martin Lorentzon, Spotify is the biggest streaming platform is term of subscribers, and offers an impressive collection of 40 million songs, podcasts and any audio content. In return for a percentage, the music labels agree to share their contents on the streaming platform.

Spotify was an instant success and gradually gained both ad-supported and premium users. The platform successfully marketed new features like family payment plan or Discovery Playlist. The brand even went public in April 2018, with a market cap of $26.5 billion after the first day of trading.

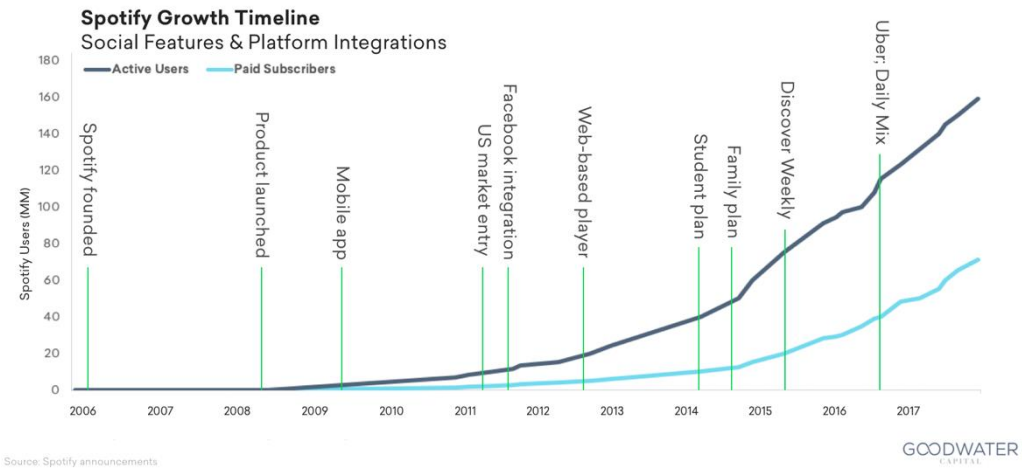
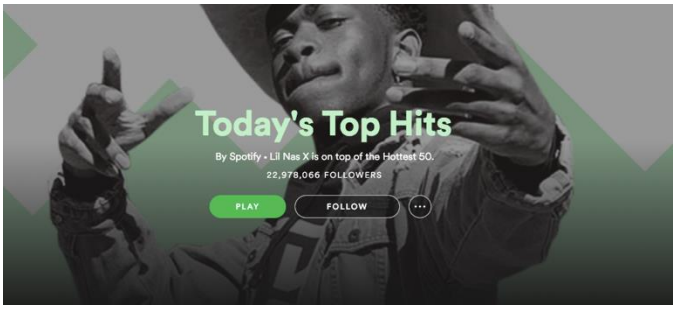


Figure 3: Spotify Growth Timeline since 2006

II.2.2.2 The power to make or break an artist's career

With 207 million users and 44% of them listening to Spotify on a daily basis, the platform has a huge power on success or failure of a content. Spotify strongly influences its users' habits by suggesting music, allowing him to create his own playlists or listening to branded playlists.



Behind the objective of driving users' satisfaction, Spotify is indeed a career booster and appearing in one the Spotify's playlist is a must for any artist. A study led by researchers at the European Commission's Joint Research Center and the University of Minnesota has shown that a song listed in the playlist "Today's Top Hits", which counts 23M followers (as of April 2019), generates between on average 20 million more streams and $150,000 for the artist.

Although Spotify claims to democratize the access to audio content worldwide, approximately 87% of its contents come from the 4 biggest music labels. This questions the monopoly, objectivity and equity of streaming platforms, artists and labels being completely dependent. Many also agreed that artists are not paid enough.

II.2.3 - A competitive business: Who run the market?

If we look at the global picture, Spotify is the worldwide streaming leader, away from its main competitors Jay-Z's Tidal, Pandora, Amazon Prime Music and obviously Apple Music.

However, since last year Apple Music has overtaken Spotify in terms of subscribers in the US market. Since its launch in July 2015, Apple Music deploys a taught and aggressive strategy to gain market shares. With its hundreds of millions of iOS customers, its unbelievable marketing power, the leadership of the brand is more a matter of when rather than a matter of if.

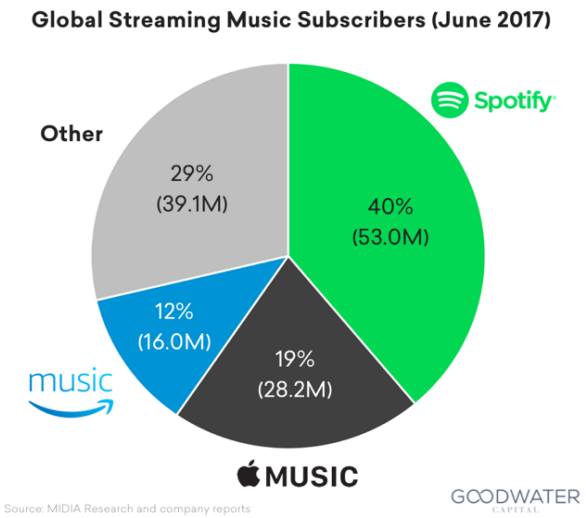


Figure 4: Who run the music streaming market? (source: Midia Research)

II.2.4 - Which future for music industry?

From music to video and gaming, streaming is quite recent but everywhere in our daily lives. What is the future of music streaming services? From a business point of view, the question is to know whether the streaming platforms will be able to adapt their business to the top-notch tech such as virtual reality (VR) or voice recognition, and find a balance to be economically viable. Today all the streaming platforms seem to stabilize around a monthly charge of $10 for the premium subscribers but music industry is changing fast. The viability of the model in the long term is an open question and will have to equally benefit all stakeholders.

II.2.4.1 - A great demographic potential

Streaming is inseparable from a granted access to internet connection. And although we may think that everyone is connected at any time, there is still room from improvement. Let's take the example of Airplane companies, currently equipping their aircraft with Wifi. This is now more than one billion travels who have access to an internet connection while travelling.

In purely demographic terms, 660 million African people are expected to have an internet connection by the end of 2020. A substantial advantage for streaming platforms which presence in Africa is for now limited to 5 countries. High cost of data, smartphone penetration, copyright and local laws are the last but not least barriers to access a booming market.

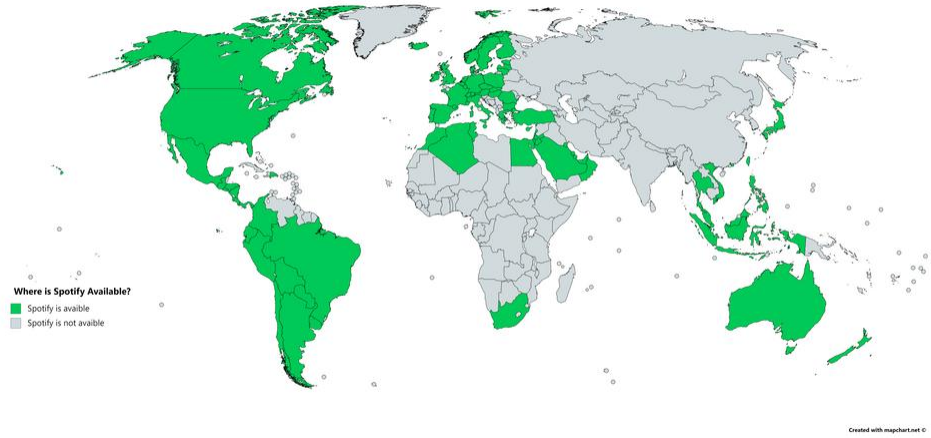


Figure 5: Where is Spotify available?

II.2.4.2 Limitless technologies: Smart speakers, connected devices and music on the go

From a technical view, the main challenge will be to improve the user experience and integrate new features. For example, the rise of virtual reality could offer the possibility for millions of people to attend directly from their living room. The next challenge is for sure to integrate voice recognition in-car, to allow people to use the platforms hand-free while driving.

Most recently, home connected devices like Amazon Alexa or Google home and used as a personal assistant, allow people to enjoy music directly from their favorite streaming platform using voice recognition. Technology moves fast and possibilities seem to be limitless. It is very likely that it won't be long before a new technology shakes the music industry up again.

III – Data collection

The objective of this study is to find a way to predict whether a song will become a hit or not. To do so, we have two possibilities:

- Predict a continuous variable (for example the number of streams, the popularity score …)
- Predict a categorical variable (this is a classification problem, the goal is to label a song as being hit / not hit)

The choice of one of those alternatives will be demonstrated later on.

III.1 - Data sources: which data are available?

III.1.1 - Different sources

When it comes to music datasets, the choices are infinite. From to the well-known Million Song Dataset, there is a wide variety of freely-available collection of audio and meta features.

The Million Song Dataset has been used for many data competitions and helped a lot on researches. It uses a technology developed by Echo Nest to extract audio features of a song. Echo Nest is the "Music Intelligence" company and was the pioneer in the music prediction field. In 2014, Spotify disbursed $100M to acquire Echo Nest, which instantly gave the streaming platform a competitive advantage when it comes to music recommendation driven by algorithms. Echo Nest used to be open-source and its data available via an API used by developers. Unfortunately, Spotify closed it in 2016 to pull the rug out from under its competitors and replaced it by the Spotify API with limited accessible features.

III.1.2 - Choice of Spotify Data

The choice of the dataset depends on what we want to study. Here, I will dedicate part of my work to explain how music has evolved over the years, but the main objective is to predict if a song will become a hit.

As seen in the previous part, Spotify and its competitors are running the game and have the power to make or break an artist career. To make it short, they are the new trend-makers.

Knowing all this, I decided to focus on recent and reliable data, directly coming from Spotify. I realized an analysis using Twitter API over the most recent Tweets mentioning Spotify (see annex 1 to check the R code). The words analysis shows that Spotify is very often associated and compared to its first competitor Apple Music. The picture displays the most frequent words associated to Spotify (Figure 6).

Figure 6: Words analysis over the 1500 most recent tweets, as of 04/2019

Now I have my tweets database, let's do a sentiment analysis. Using the R package "sentiment", the objective is to catch the global sentiment of a tweet. Is the user satisfied about Spotify? Are they Angry?

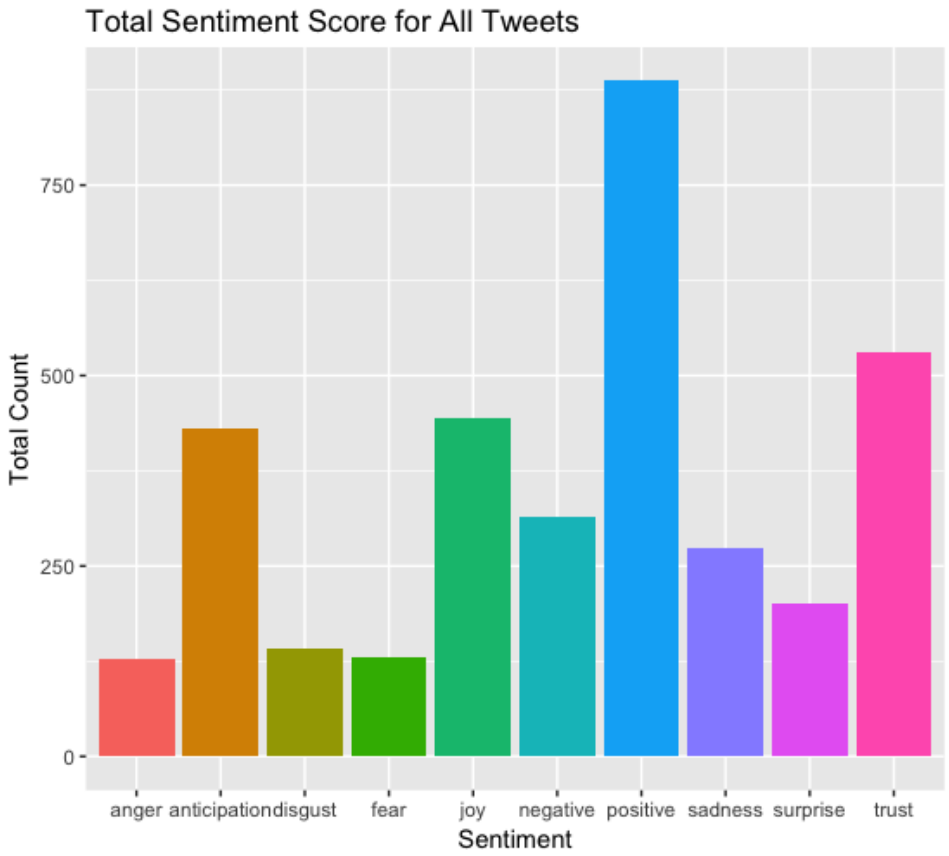


Figure 7: Sentiment Analysis over the 1500 most recent tweets, as of 04/2019

People globally tweet about Spotify in a positive way, followed by sentiments of trust and joy. Thanks to all that had been demonstrated before, we can conclude that Spotify streams are relevant data to study what is happening within the music industry, understand this new way of consuming music and realize a predictive analysis.

III.2 - Features to extract

This part is dedicated to Spotify data and features. Which data are available? What do the features mean? How to extract the data I need?

III.2.1 - Definition of Spotify features: audio vs. meta

Spotify closed in 2016 Echo Nest API and replaced it with its own Spotify API for developers. Here is a non-exhaustive list of the available features with their definitions[1].

Audio: Features purely concerning audio content of a track

Duration: Length of the track in seconds.

Tempo: The average tempo expressed in beats per minute (bpm).

Time signature: A symbolic representation of how many beats there are in each bar.

Mode: Describes if a song's modality is major (1) or minor (0).

Key: The estimated key of the track, represented as an integer.

Loudness: The loudness of a track in decibels (dB), which correlates to the psychological perception of strength (amplitude). Values typical range between -60 and 0 db.

Danceability: Danceability describes how suitable a track is for dancing based on a combination

[1] As defined by Spotify on their website

of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable. Calculated based on beat, strength, tempo stability, overall tempo, and more.

Energy: Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. For example, death metal has high energy, while a Bach prelude scores low on the scale. Perceptual features contributing to this attribute include dynamic range, perceived loudness, timbre, onset rate, and general entropy. Calculated based on loudness and segment durations.

Acousticness: A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.

Speechiness: Speechiness detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value. Values above 0.66 describe tracks that are probably made entirely of spoken words. Values between 0.33 and 0.66 describe tracks that may contain both music and speech, either in sections or layered, including such cases as rap music. Values below 0.33 most likely represent music and other non-speech-like tracks.

Instrumentalness: Predicts whether a track contains no vocals. “Ooh” and “aah” sounds are treated as instrumental in this context. Rap or spoken word tracks are clearly “vocal”. The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content. Values above 0.5 are intended to represent instrumental tracks, but confidence is higher as the value approaches 1.0.

Liveness: Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. A value above 0.8 provides strong likelihood that the track is live.

Valence: A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry).

Meta: Information about non-audio data

Artist, Album, Track name: Name of the artist, album, track

Popularity: The popularity of the track. The value will be between 0 and 100, with 100 being the most popular. The track's popularity is calculated from the number of most recent streams.

ID: Spotify ID of the artist, album, track

Images: Images of the artist/album in various sizes, widest first.

Explicit: Value takes 1 if the song contains explicit lyrics, 0 otherwise.

Other meta features are available but I won't use them later. The audio and meta features defined above are the one extracted and used in the following sections.

The Echo Nest technology combines listening techniques to simulate how people perceive music. "It incorporates principles of psychoacoustics, music perception, and adaptive learning to model both the physical and cognitive processes of human listening. The output of analyze contains a complete description of all musical events, structures, and global attributes such as

key, loudness, time signature, tempo, beats, sections, harmony." [Tristan Jehan & David DesRoches, Analyzer Documentation, 2014]

The program generates a JSON text file that describes the track's structure and musical content. Here is the output for the song "Around the world" by Daft Punk:

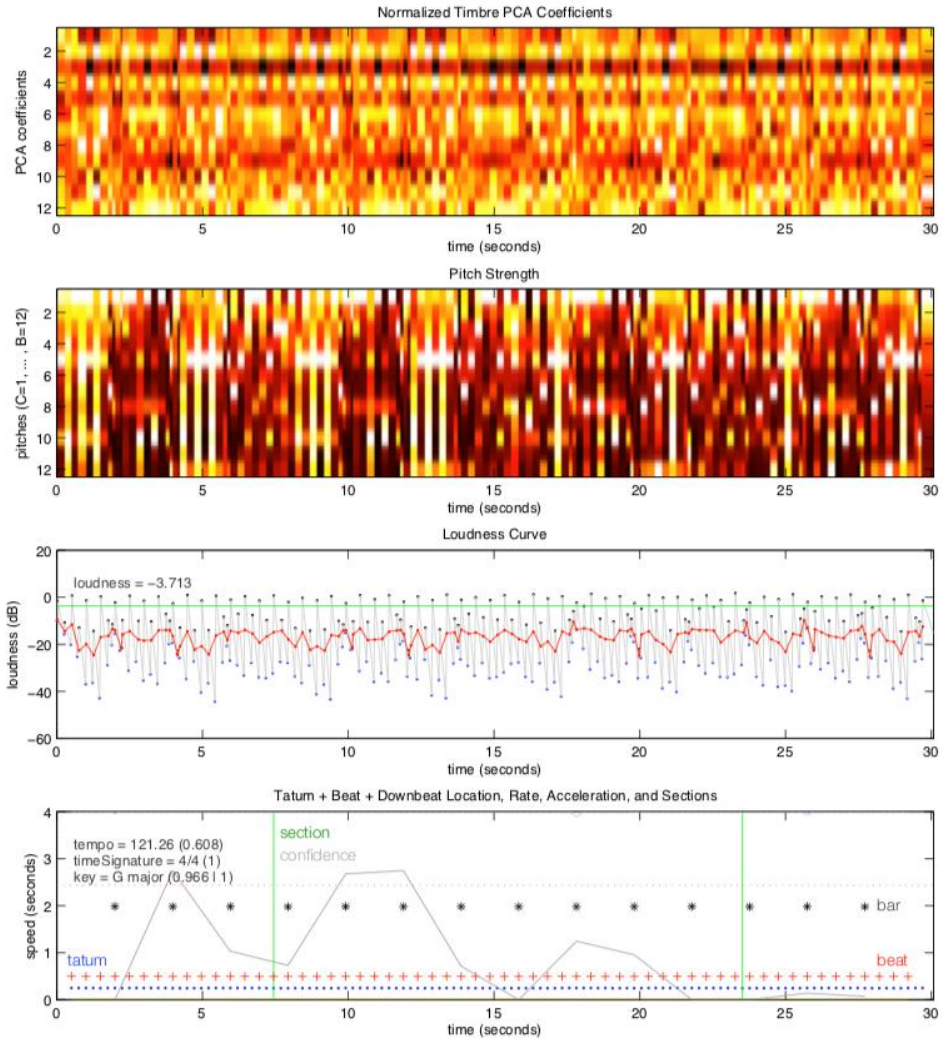


Figure 8: Plot of the JSON data for a 30-second excerpt of "around the world" by Daft Punk

III.2.2 - Spotify for developers and API

Many datasets are available but ideally I can choose the data I want to extract in order to realize my own work on the data. For this reason, I learnt to use the library developed by Spotify, called Spotipy. This library allows to get the data needed using Spotify web API. Do to so, I have created a Python code to extract all the tracks listed in a given playlist. The code returns a data frame (see Tables 1) with the features defined in the previous part for each track and can be found in annex 2.

	name	album	artist	release_date	duration_ms	explicit	danceability	acousticness	danceability	energy	instrumentalness	liveness	loudness
0	HUMBLE.	DAMN.	Kendrick Lamar	2017-04-14	177000	True	0.908	0.000282	0.908	0.621	0.000054	0.0958	-6.638
1	XO TOUR Llif3	Luv Is Rage 2	Lil Uzi Vert	2017-08-25	182706	True	0.732	0.002640	0.732	0.750	0.000000	0.1090	-6.366
2	Shape of You	÷ (Deluxe)	Ed Sheeran	2017-03-03	233712	False	0.825	0.581000	0.825	0.652	0.000000	0.0931	-3.183
3	Congratulations	Stoney (Deluxe)	Post Malone	2016-12-09	220293	True	0.630	0.215000	0.630	0.804	0.000000	0.2530	-4.183
4	Despacito - Remix	Despacito Feat. Justin Bieber (Remix)	Luis Fonsi	2017-04-17	228826	False	0.694	0.229000	0.694	0.815	0.000000	0.0924	-4.328

Table 1: Features extraction for top 2017 songs

To conclude

Because of Spotify leadership on the streaming market with millions of users worldwide, a growing market, challenging technologies and a fast-changing environment, I decided to choose the streaming platform to extract my data and work on it. In the following part, I first will look at the songs' evolution over the past 60 years in order to understand the predictor variables, then I will focus on the most played songs on Spotify and try to find common patterns to finally give a proper definition of what we call a hit song.

IV – Features Understanding: Is there a hit song receipt?

The objective of this section is first to analyze the music evolution over the last 60 years through the understanding of the explanatory variables. I will study both audio and meta features to find insights and common patterns among the world's hit songs.

IV.1 – Features Analysis over the last 60 years

IV.1.1 – Objectives & Data collection

To analyze music evolution over the last 60 years I use the dataset created by Derek Zhao and available on GitHub[2]. This dataset has been created by scrapping the billboard website in order to get the top 100 songs for each year since 1960 until 2017, with their lyrics. It is then easy to add the audio features of those songs using the Spotify web API mentioned in section III.2.2. The final dataset is composed of 5566 songs.
This analysis and all the graphs displayed in this section have been created with Tableau Software.

IV.1.2 – Artists Analysis

Let's start to analyze the artists present in the dataset. Who are the hit-makers? Who have the most long-lasting career?

[2] Derek Zhao, 2017, dataset: https://github.com/zhao1701/spotify-song-lyric-analysis/blob/master/data/billboard-lyrics.csv

IV.1.2.1 - Who dominates the Top Charts?

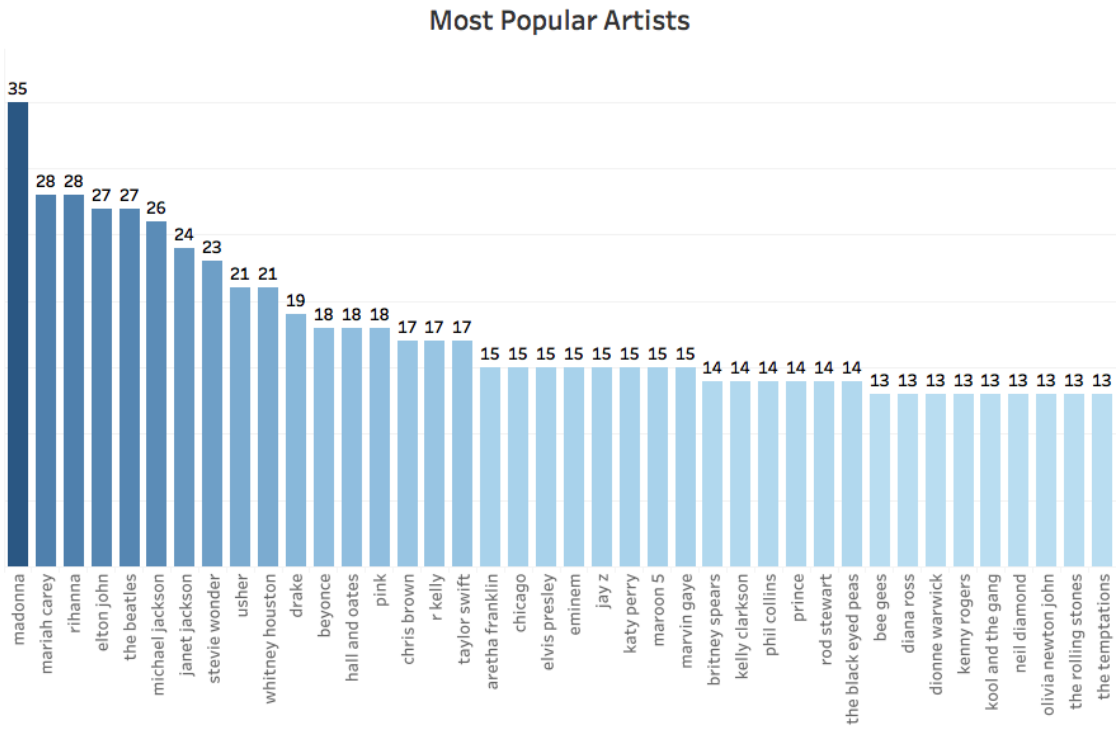


Figure 9: Who are the top performers? Number of hit songs registered in the charts from 1960 to 2017, per artist

This graph shows all the artists having at least 13 songs in the Top 100 charts since 1960. Madonna runs the game with 35 hit singles and may increase her chart score soon with the release of a new album in June 2019. It is worth noticing that females are very well represented in the ranking and parity is respected if we look at the Tops 10 and 20 singers.

IV.1.2.2 - From legendary to single-hit artists

The dataset is composed of 5566 songs for 2283 artists. Among those artists, 1329 produced only one charted song. Finally, 58% of the artists only made a single hit and didn't succeeded to make their career last.

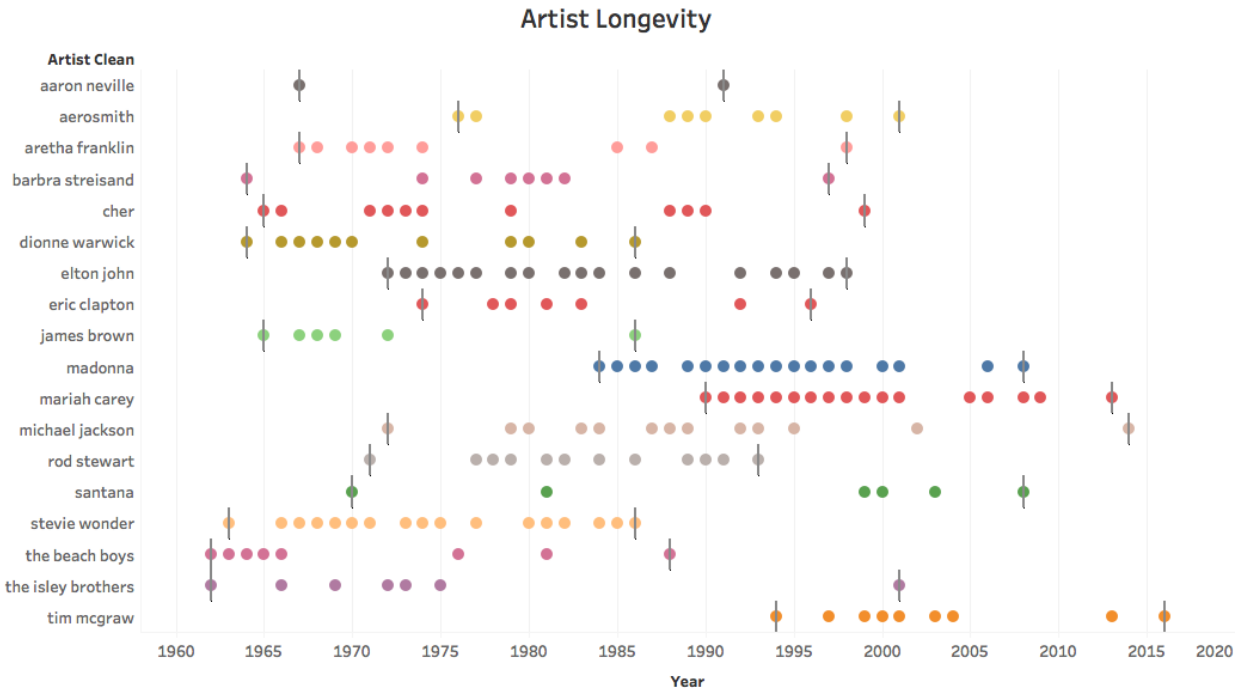


Figure 10: Artists having a career longevity greater than 20 years. One dot corresponds to the release year of a hit song registered in the charts

Figure 10 is a plot of artists having a career larger than 20 years. The time line represents an artist's career, the points represent the hit songs. Only 18 of them have released hit songs in two different decades. This is a consequence of the harsh competition in the music industry. Producing a hit song is definitely not enough to win a place in the business.

IV.1.2.3 - Featuring: a new world of collaboration

We talk about featuring when an artist invites another artist to collaborate on a specific song.

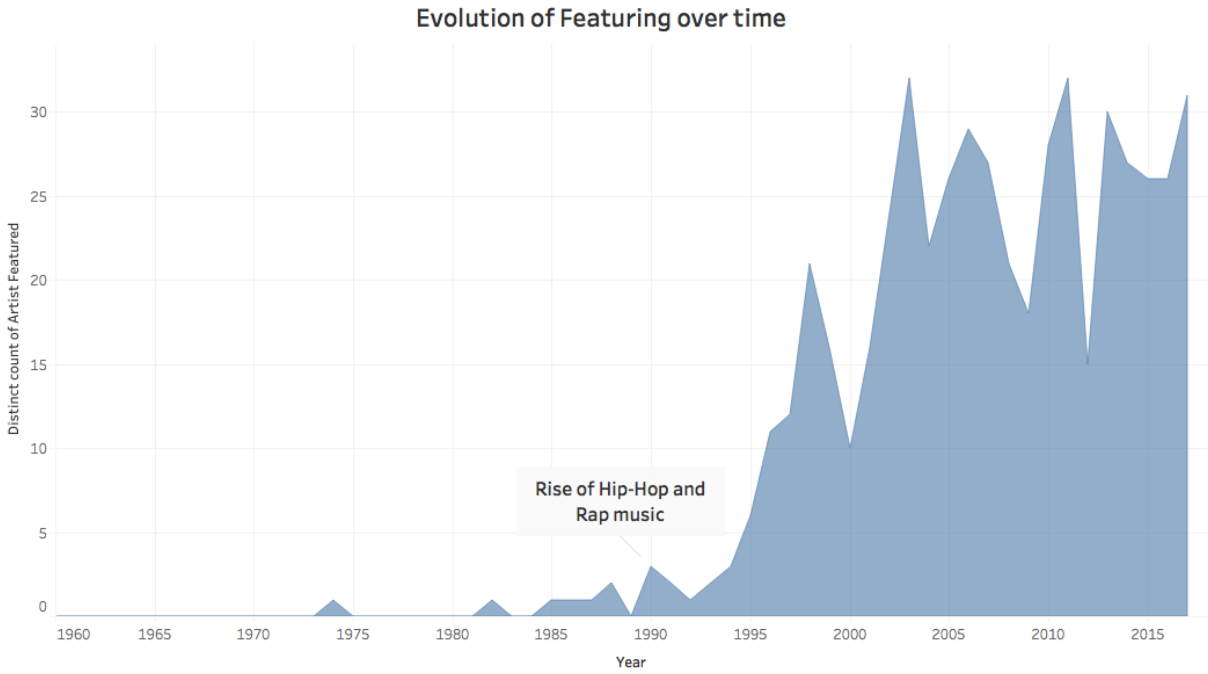


Figure 11: Evolution of the number of top featured songs over the last 60 years

Figure 11 shows a marked increase in the number of collaborations since 1990. A reason for that is the rise of Hip-Hop and rap music during the 90's, which are prime examples of collaborative genres. During the most recent years, more than ¼ of the top songs comes from featuring.

IV.1.2.4 - A drop in diversity

It seems that over the years, a shorter number of artists dominates the charts. This is again a consequence of the competition and globalization of the music industry.

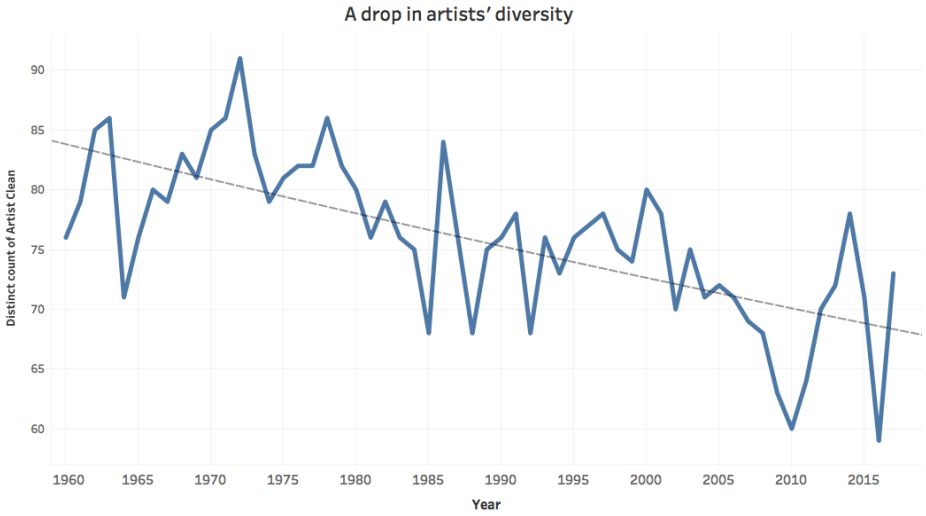


Figure 12: Evolution of the number of unique artists in the charts over the last 60 years

From the graph (Figure 12), we can clearly see the drop in artists' diversity during the 2000's. This is due to online piracy and the very bad period the music industry was living. The labels surely were risk-averse and bet on their most popular and lucrative artists.

The decline in diversity is still an uncertainty of streaming and recommendation algorithms which obviously influence the user's behavior. Towards a globalization of behaviors or limitless opportunities for the users? It is too early for now to draw a conclusion.

IV.1.3 - Audio Features Analysis

Let's start with the evolution of the purely audio content over the years. Here I study the features described in section III.2.1.

Below is the distribution of the audio features. Those features are related to objective audio content of the songs. Lyrics-related features will be analyzed in section IV.1.4

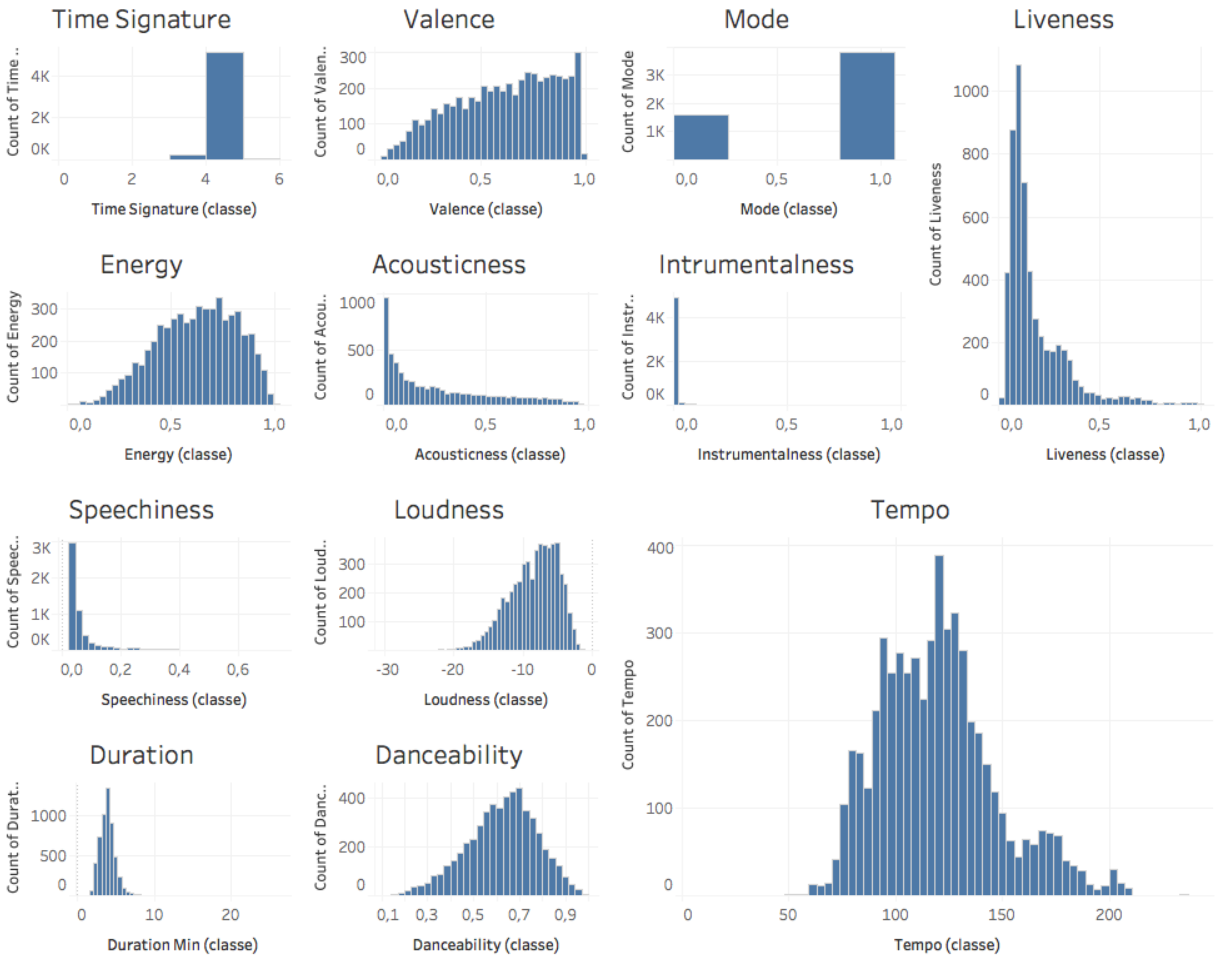


Figure 13: Distribution of audio content metrics

IV.1.3.1 - A trend to go pop

IV.1.3.1.1 - More energy and danceability

The graph below (Figure 14) shows the evolution of danceability and energy metrics and are good indicators of the global trend. High danceability and energy score demonstrate the will for artists to go pop to reach a wider public and grow in popularity.

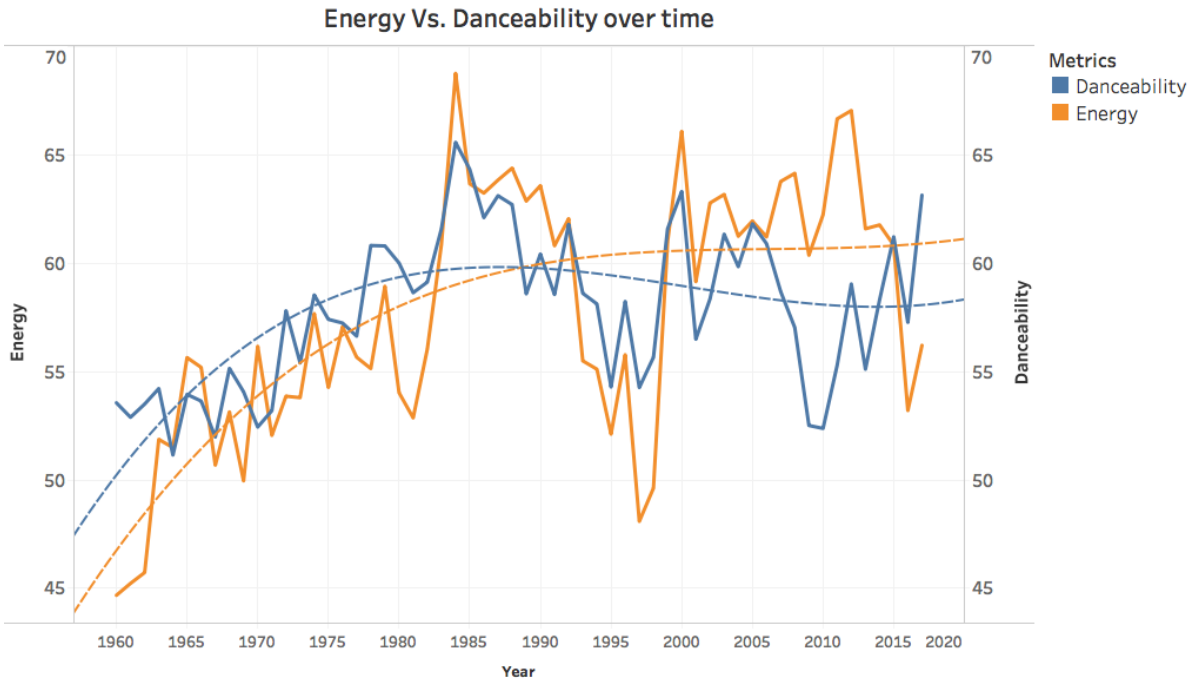


Figure 14: Evolution of Danceability vs. Energy metrics

IV.1.3.1.2 - Louder and sadder

Over the years, the loudness has notably increased. This trend can be explained by the use of compressors to squash the music (to make the quiet parts sound louder). The music industry talks about a "Loudness War"[3] with the objective of producing a song that jumps out of your iPod and sounds louder than the previous one you've been listening to.

[3] All Things Considered, 2009, The Loudness Wars: Why Music Sounds Worse, https://www.npr.org/2009/12/31/122114058/the-loudness-wars-why-music-sounds-worse?t=1556289478468

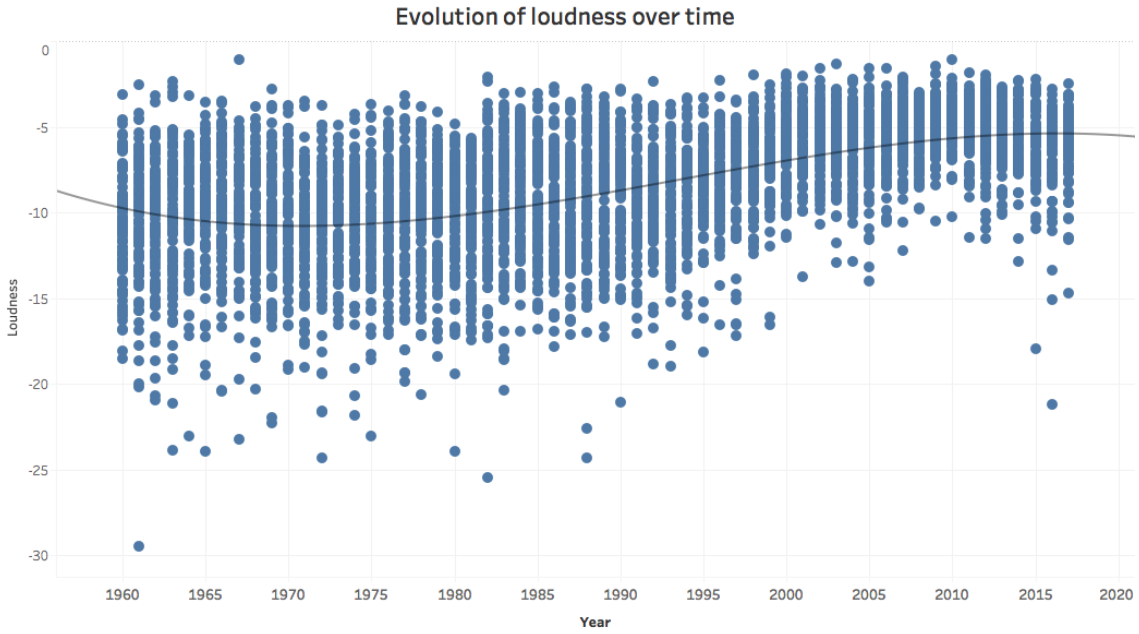


Figure 15: Loudness evolution over time

Although increasing energy and danceability scores, it seems that hits songs became sadder. Valence describes the positiveness of a song. A song with a high valence score is then joyful, happy. The graph below shows the evolution of this score and, surprisingly, valence is smaller and smaller, resulting in sad and less positive sounds.

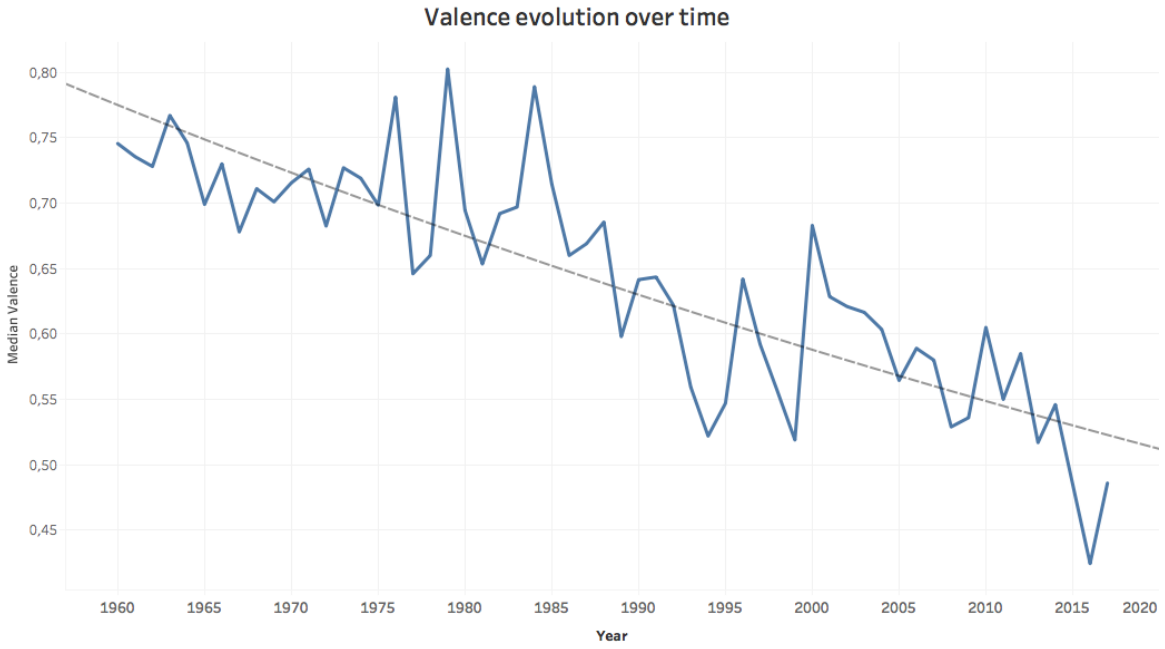


Figure 16: Evolution of valence over time

IV.1.3.1.3 - Less acoustic and more speech

The six last decades saw a drastic decline in acoustic (see Figure 17). First because of the rise of the electric guitar, then the invention of synthesizers and later the rise of EDM (electronic dance music) movements in the 2000's.

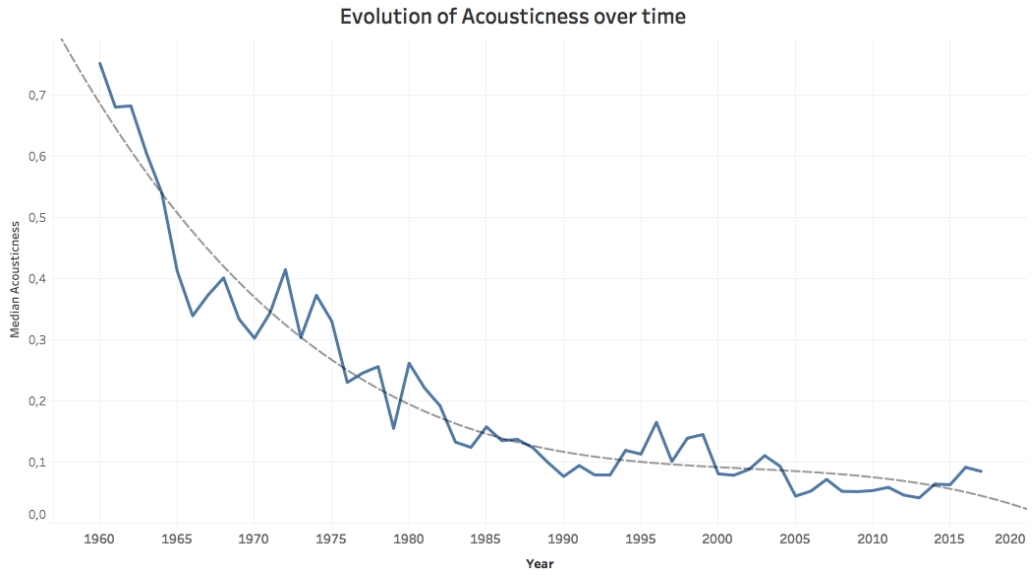


Figure 17: Acoustic evolution over the years

Speechiness measures the presence of spoken words in a recording. Records with a high score are most likely rap songs. If we look at the speechiness evolution, it increases over the years, with an upsurge in the 2000's, which matches with the rise of rap movements. We will learn more about this phenomenon when analyzing audio features and lyrics (see parts II.1.3 and II.1.4).

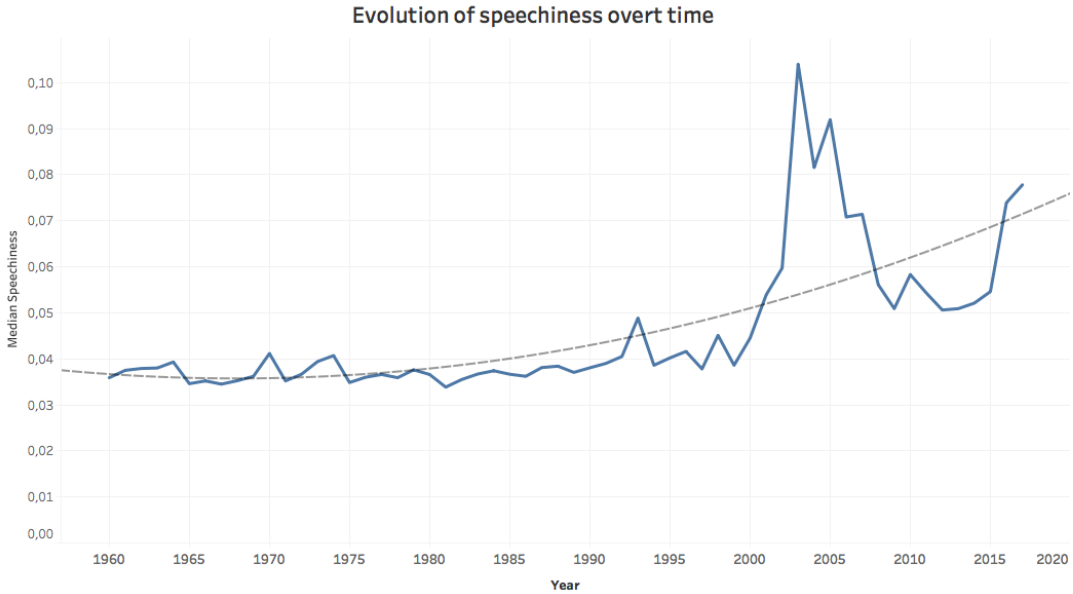


Figure 18: Speechiness over the years

IV.1.3.2 - Shorter songs: A symptom of attention economy

The fast technology evolution over the last decades drastically shifted the way we are consuming and listening music. First with devices allowing storage and personal consumption, most recently with streaming platforms, the user is completely in charge of what he wants to listen, and free to make his own choices. A study led by Hubert Léveillé Gauvin[4], a doctoral student in music theory at The Ohio State University, has shown the effects of what he calls "attention economy", this harsh competition to capture the listener's attention [Hubert Léveillé Gauvin, 2017] . Let's keep on analyzing my dataset and get some insights about this phenomenon.

IV.1.3.2.1 - Drop in duration

From 1960 to the 80's, the average duration of a song constantly increased, thanks to the invention of storage devices like tape, vinyl and CD.

Since the 80's, the average duration of a song keeps decreasing (to reach 3.6min), while the number of words per second increases (see Figure 19). Again, this can be due to rap music which fosters short songs and a high speech rate. But not only. Although it might be obvious, this is the first observation when it comes to attention economy: to grab a limited lifespan attention, make it short.

[4] Hubert Léveillé Gauvin, 2017, *Musicae Scientiae*, https://journals.sagepub.com/doi/full/10.1177/1029864917698010 for more details

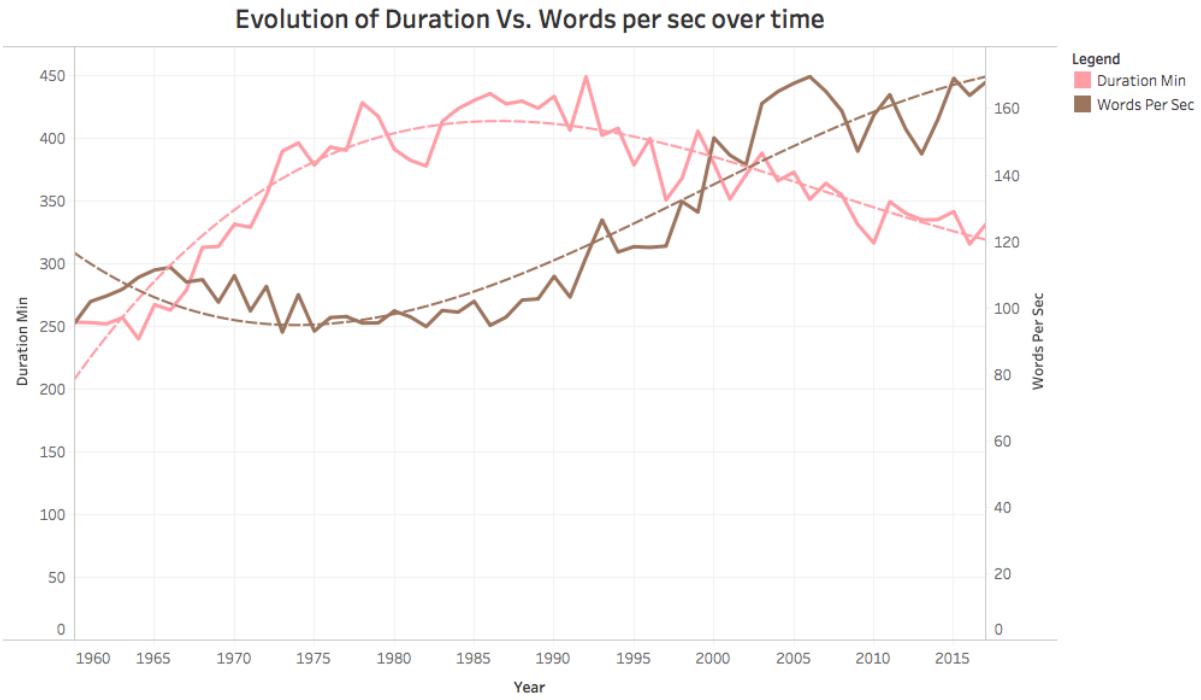


Figure 19: Evolution of songs duration Vs. Number of words per second

IV.1.3.2.2 - Cutting-off the instrumental

Intro has completely disappeared from the most recent songs. The objective is to quickly grab the listener's attention before being skipped. To make the songs' duration decrease, the intro was the easier audio element to cut-off.

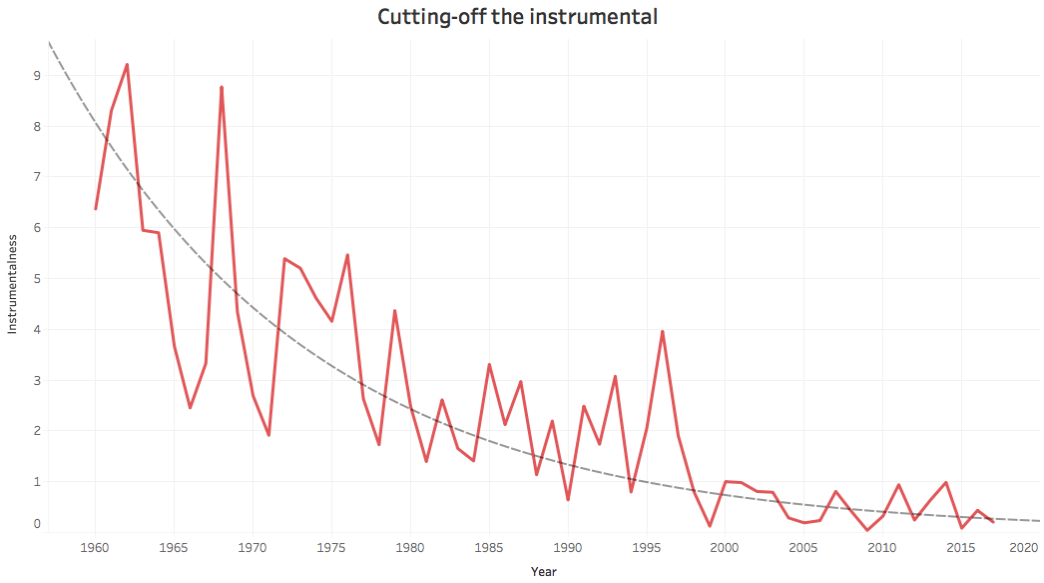


Figure 20: A significant decrease of intrumentalness over time

Hubert Léveillé Gauvin found out that intros were averaging more than 20 seconds in the 80's and are now lasting 5 seconds.

IV.1.3.2.3 -Shorter song names

The time of long, poetic and meaningful song titles is now far away. Still because of a clear objective of efficiently catching attention, song names became shorter and shorter, until they reach a single word.

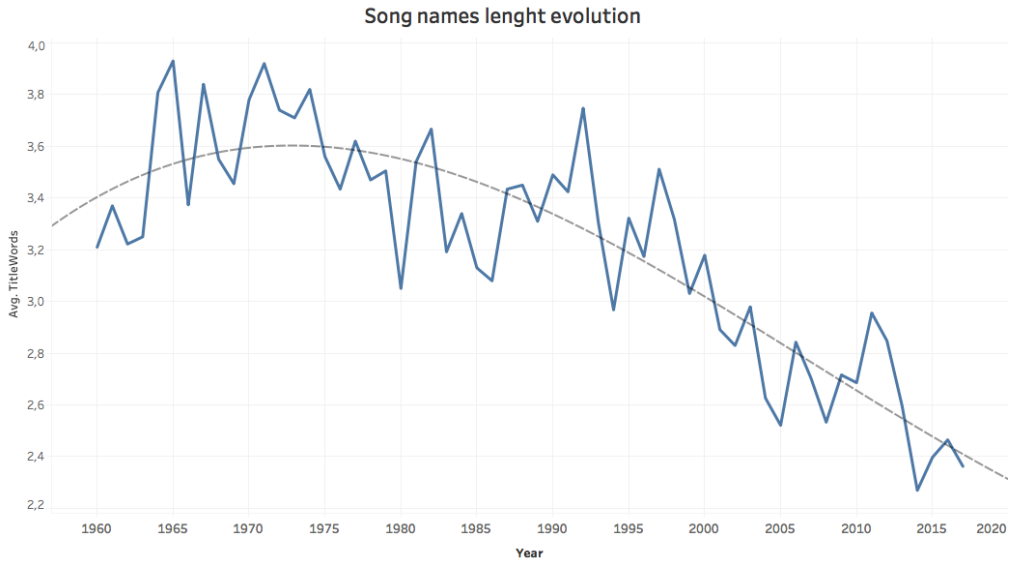


Figure 21: Evolution of the songs' titles length

Figure 21 shows that the number of words composing a song title was close to 4 until the 80's and nearly halved until now.

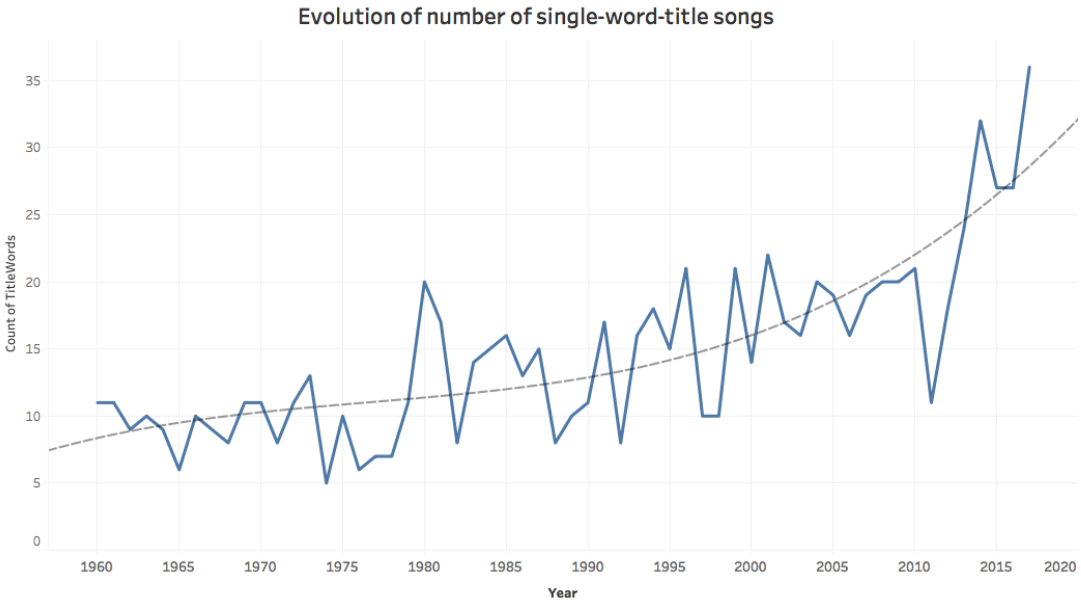


Figure 22: Evolution of single-word-title songs

While less than 10% of the hit songs' titles were a single word during the 70's, more than 25% of the recent hit names are as short.

Shorter songs with no more intro, shorter song titles, are the consequences of the necessity for the music industry to fight to grab an ever-solicited user's attention. Audio contents also had to adapt to some storage changes and compression of sounds on a digital format. If we sum-up the findings, today's hit song is a modern-day pop, single-word title song, energetic, and loud. But what about the lyrics?

IV.1.4 - Lyrics Analysis

In this part, I perform a text analysis on the lyrics available in my dataset. Let's do a first global state-of-play, before analyzing the textual content of the lyrics and answer the following question: How did the lyrics' content evolve over time?

IV.1.4.1 - More textual content, growing vocabulary and explicit songs

In the previous part, we saw with Figure 19 (Evolution of song duration Vs. Words per second) that although the average song duration decreased, the number of words per second stepped up, resulting in an increasing speechiness score.
Figure 23 is a graph that displays the evolution of the number of words in a song vs. the number of unique words. In both cases, there is a clear upward trend.

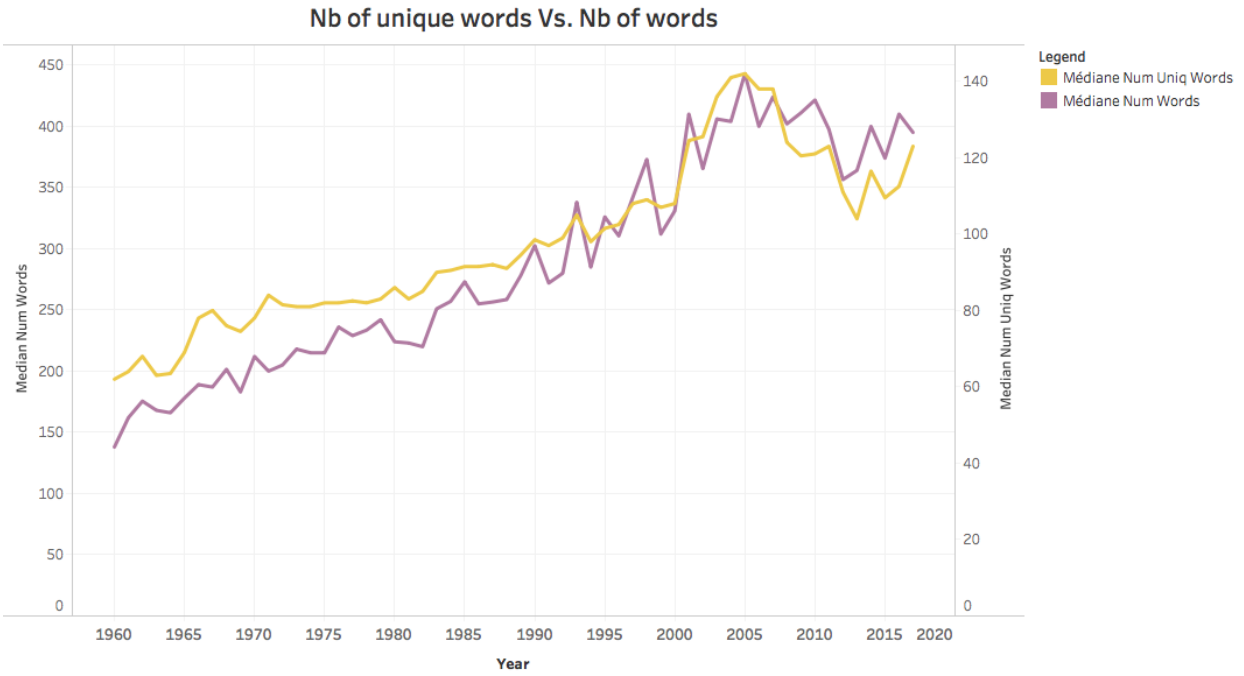


Figure 23: Evolution of #Words Vs. #Unique words over time

We can give credits to rap and hip-hop genres which combine song featuring and a fast speech rate. More words but to say what? Let's have a look at the evolution of the number of songs whose lyrics has been labeled as being explicit. Once again, the upward trend perfectly follows the rise of rap music.

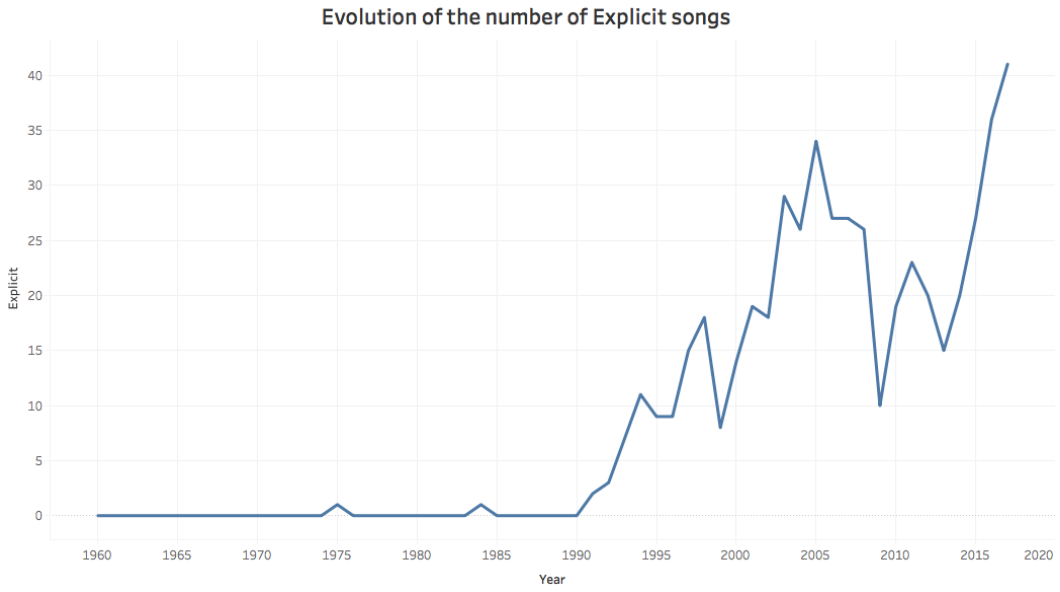


Figure 24: Evolution of the number of songs containing explicit lyrics

The term explicit is defined as follow[5]: "The explicit logo is applied when the lyrics or content of a song or a music video contain one or more of the following criteria which could be considered offensive or unsuitable for children:

- Strong language
- Images of or references to violence, physical, mental abuse;
- References to or images of sexualized behavior;
- Racist, homophobic, misogynistic or other language or behavior that could be considered discriminatory; or
- Dangerous or criminal behavior which could glamorize such behavior or may encourage others to imitate it."

We can therefore expect to see a clear change while exploring the lyrics content more deeply. The objective of the following sections is to study the lyrics through a text analysis. I use Python and meaningful libraries to perform the study. The notebook is in annex 3 and all the charts are created with the python library matplotlib and result from the text analysis.

[5] Source: http://parental-advisory.co.uk/

IV.1.4.2 - Most frequent words

I used the package WordCloud to perform a text analysis over the whole bunch of words appearing in the lyrics since 1960. After removing the common and uninformative words, “love”, ”like” and “know” are the most frequent words (see Figure 25).

Figure 25: Word cloud of the most frequent words contained in the lyrics

Let’s now have a look at the evolution throughout the years. I plot the most frequent words per decade (Figure 26). We can clearly see the increasing explicit songs trend and the evolution of thematic. Love and romance progressively disappeared, leaving room for foul language.

Figure 26: Most frequent words per decade

IV.1.4.3 - Word usage over time: From romance to indecency

Let's focus here on the most common words to see how they are used over time. The code I ran counts the number of time a chosen word occurs over time. The word "love" is less and less used and replaced by "like", showing a drop in romance since the 90's.

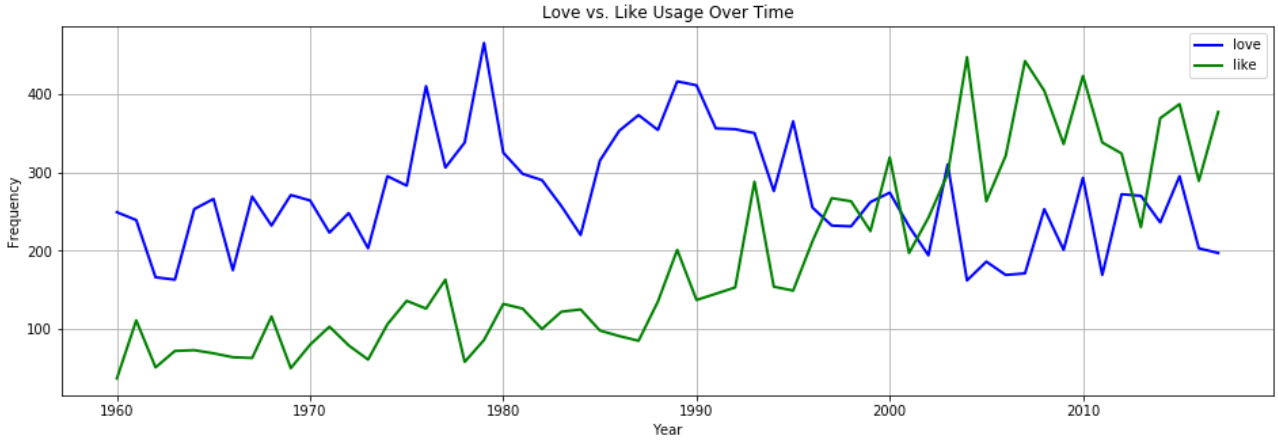


Figure 27: Evolution of "love" and "like" over time

It is also clear that rap music brought its bunch of indecent words which start appearing for the first time in the lyrics during the 90's.

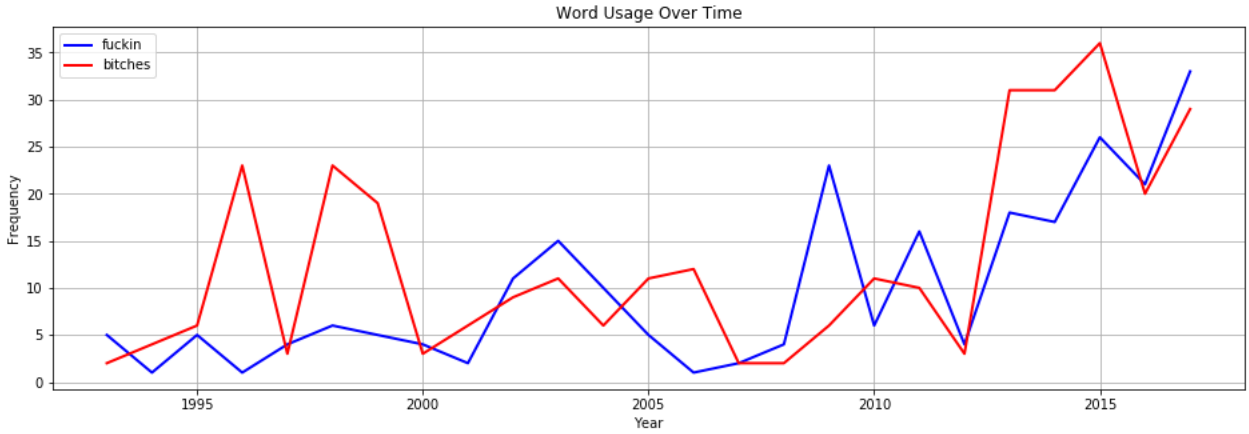


Figure 28: Foul language evolution in lyrics

To keep on with sexism, let's have a look at the words used to define gender. "Girl" is highly preferred to "woman" while "man" is favorited over "boy".

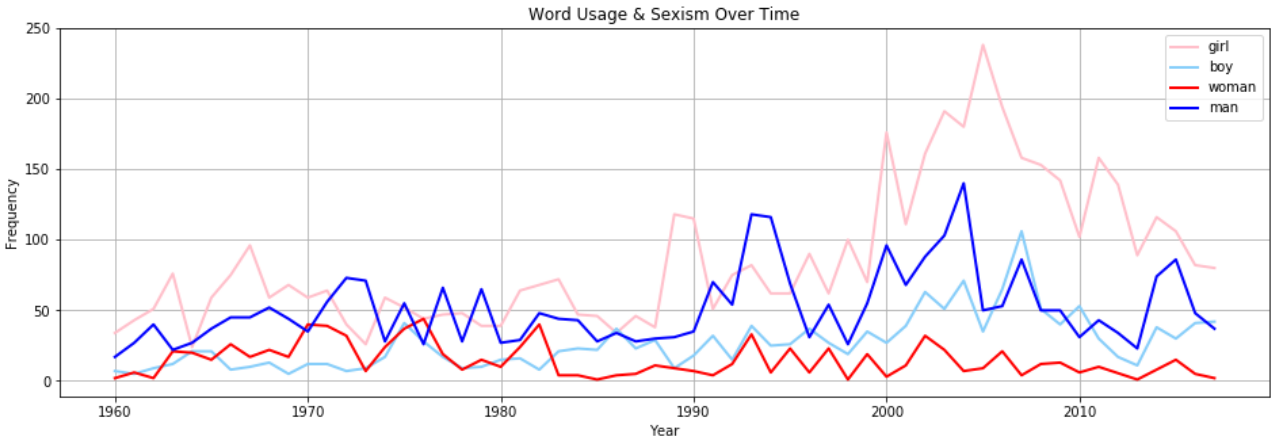


Figure 29: Gender names evolution over time

Hip-Hop and Rap music have definitely highly influenced every aspect of the music industry. From purely audio features to lyrics, those genres took over the business.
Thanks to lyrics scrapped from the web and Spotify audio data, we now have a clear overview of what happened since 1960.

Songs evolved a lot over the last 8 decades. Because of harsh competition, a shrinking market, the rise of new genres, technology improvements, music industry is constantly moving, shaped with new trends. Hit songs of today won't necessarily be hits tomorrow, and hits of the past decades would hardly be listened if they were release today. Therefore, to make accurate prediction of what will be a hit song, I have to train my models on the current definition of a hit song. To do so, the dataset must contain recent data. To confirm my findings, the following section is a study of the most streamed songs on Spotify. The objective is to validate the hypothesis that Spotify data are accurate and follow the trend and insights. From now, I focus only on Spotify data, and consequently on current listeners' behaviors.

IV.2 - Focus on the most played songs on Spotify

IV.2.1 - Objective & Data collection

Now that I chose to work with Spotify data, I dedicate this part to the study of the most streamed songs on the platform. The objective is to understand users' behavior and give a definition a what is a hit song today.

Using Spotify Web API (see code in annex 2), I extract the most played songs (all the songs that have been streamed more than 200 million times) and analyze the data with Tableau software. The dataset is composed of 590 songs, with their audio features.

IV.2.2 - Exploratory Analysis

IV.2.2.1 - History

Spotify was launched in 2006 and, as expected, the most streamed songs on the platform are in majority the one release after its creation. However, there are 39 songs released before Spotify has been launched and still listened more than 200 million time. Those songs are considered as "iconic" and, more than just popular hit songs for one year, survived time.

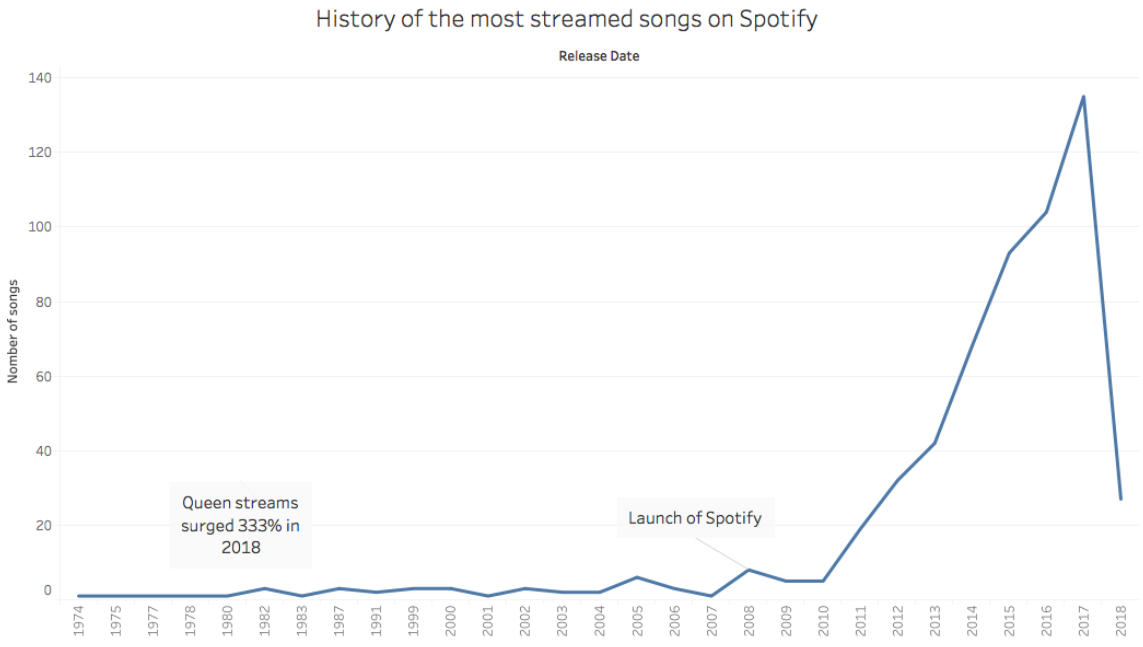


Figure 30: Count of the number of songs having been played more than 200M times on Spotify, per release date

Most of those “old” songs (everything from 1975 to 1983) are from Queen and have benefit from the film Bohemian Rhapsody in 2018, which made streams surged 333%.

IV.2.2.2 - Dashboard

I tried to validate my hypothesis to give a proper definition of a hit song. From the insights we got in part II.1, I produced a dashboard with the charts needed to check the hypothesis. The conclusions are the same as previously: a hit song is a pop, short song, with no intro.
We get exactly the same conclusions concerning audio features. However, we can notice that only 5 out of the 20 first artists performers are women.

Among the 261 artists, 114 made more than one top song, and 46 made more than 3. Indeed, 6% of the artists appear only once in the dataset.

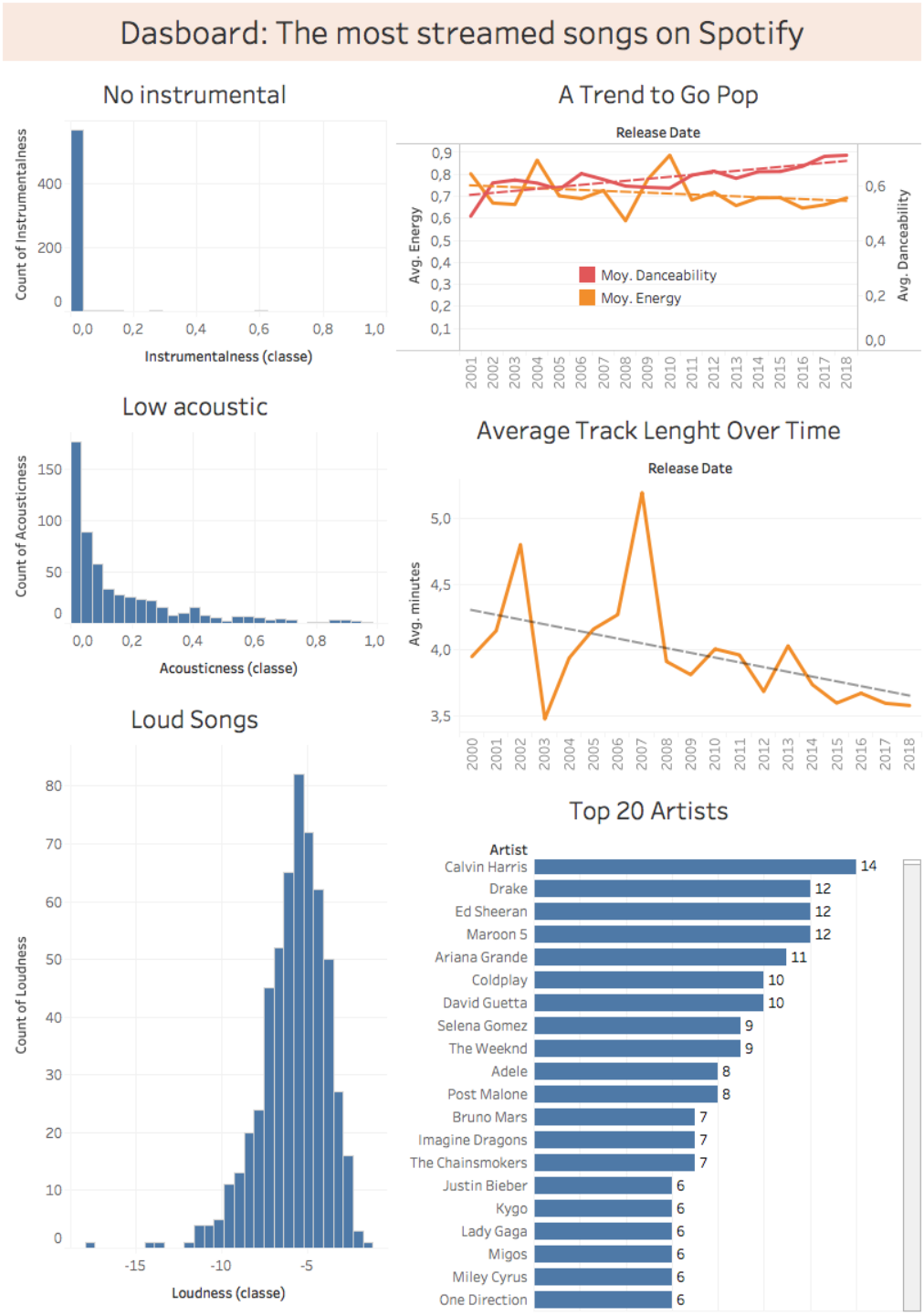


Figure 31: Dashboard of the most streamed songs on Spotify (500 songs that have been played more than 200 million times on the streaming platform)

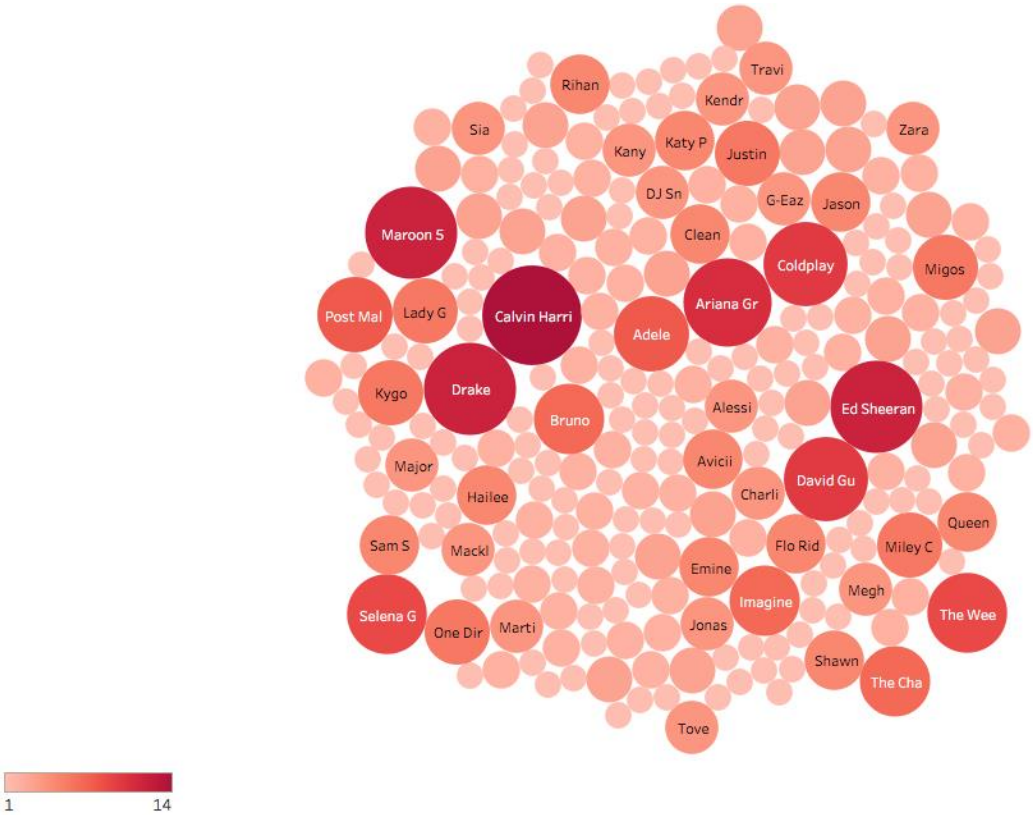


Figure 32: Top Artists according to the number of hits

Spotify data are accurate for the predictive analysis. The question is now: What do we want to predict?

IV.2.2.3 - Popularity: A bias feature

Spotify API offers the possibility to access the Popularity feature defined as a score between 0 and 100, calculated from the number of most recent streams. When we look into the details, it appears that this score is highly time sensitive and gives a snapshot of what people are listening at a given time. As of April 2019, the song “Animals” from Maroon 5 has a popularity score of 4 (despite their 32.7 million monthly listeners), while Aya Nakamura (3.9 million monthly listeners) and “Djadja” song score 94.

I then won't use this feature which could have been useful to predict a continuous variable but is indeed biased.

IV.2.2.4 - Which features for the prediction?

To predict if a song will become a hit, I will work on a classification problem and label the data hit/non hit. I won't use the lyrics of each song as it would require to first perform a text analysis to get the global sentiment of the lyrics and then work with a lot of dummy variables. I then stay focus on the audio content and keep the feature "explicit" as a dummy.

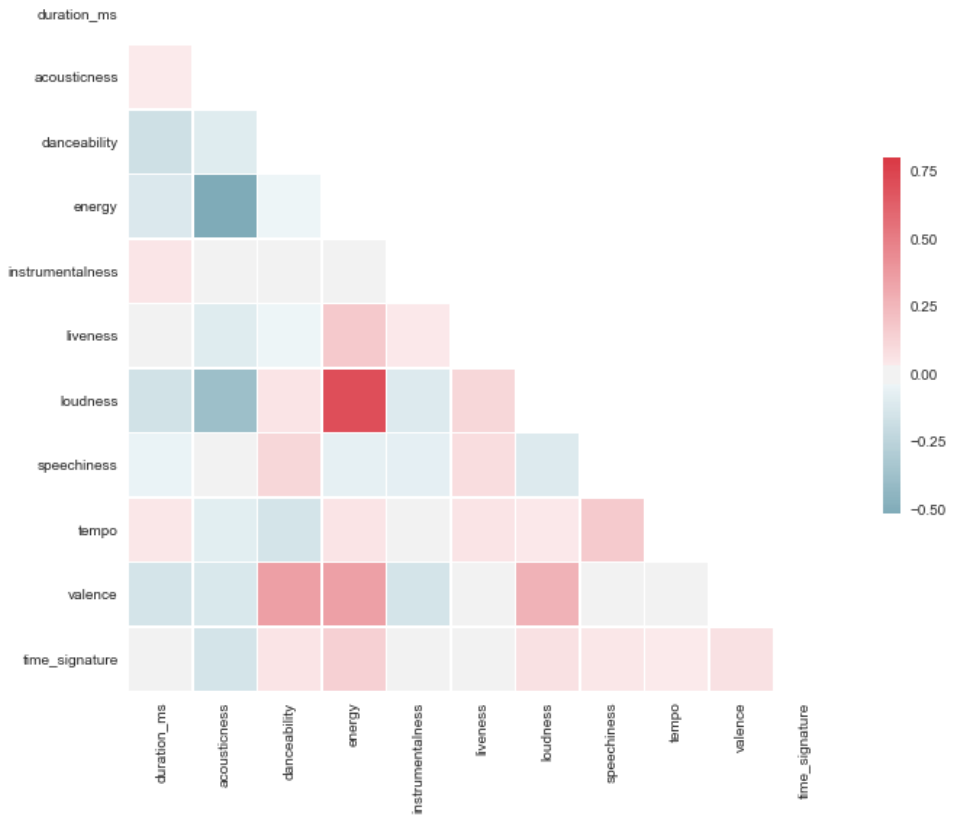


Figure 33: Correlation matrix between audio features

I have 11 audio features on which I will perform a Principal Component Analysis (PCA), a dimension reduction model that can be used to reduce the number of dimensions in a large dataset and still contains most of the information. From the correlation matrix (Figure 33), we clearly see that energy and loudness are correlated, contrary to energy and acousticness which are negatively correlated. Other correlation scores are negligible.

PCA is a mathematical procedure that transforms a number of (possibly) correlated variables into a (smaller) number of uncorrelated variables called principal components. The first principal component accounts for as much of the variability in the data as possible, and each succeeding component accounts for as much of the remaining variability as possible. The goal is dimension reduction and there is no guarantee that the dimensions are interpretable.
The code can be found in a Python notebook in annex 4 (PCA on the most played songs dataset). The first component explains 22% of the variability. The first three components explain 46% of the variability, which is not much. I won't drop any features for the predictive part, otherwise I will lose too much information. (more details in annex 4)

Spotify API is a great source to create personalized datasets. However, some features are not available and could bring a real added value to the analysis. Recording location, lyrics content, the number of streams a track has obtained are for sure interesting information but still kept private by Spotify.

IV.3 - Results and discussions: Is there a hit-song recipe? What is a hit song?

We saw, throughout the analysis of top charts over six decades and booming technologies, that audio contents have to adjust to an ever-changing environment run by a requirement for profitability.

Thanks to available Spotify data, more than studying the top charts, we can tell real data stories. In the past, we could only know which album people were buying. Now, thanks to streaming we can study daily behaviors, know what a user likes, is expecting or dislikes, with quantifiable metrics. Everything is now measurable, which is good news for data science!

As demonstrated earlier, popularity score is a biased metric and the number of streams is a non-accessible feature. To do predictions, I will then work on a classification problem and label my data as hit vs non hit. Let's trust Spotify and consider that songs appearing in the yearly top tracks playlists are hits. We also conclude in part II that, to be accurate, the algorithms must be trained on recent data. Consequently, I chose to work on the past two years' data using Spotify API and the code created in section III.
The following section of this master thesis will be entirely dedicated to the predictive analysis.

V – Methods

V.1 - A classification problem

As seen in the previous sections, given the availability of the data I can't predict a continuous variable, and thus I decided to work on a classification problem. To do so, I extracted the data for the most played songs on Spotify in 2017 and 2018 and labeled those data as hits. Then I extracted the same features for songs from the same artists, same time scope, but very few streams in comparison with the hit songs. Those data are labeled non-hit. The objective is to explain connections between inputs and outputs, and to predict the output y for a new input x.
For all the following sections, I define my observations $\mathcal{d}_n = (x_1, y_1), \ldots, (x_n, y_n)$ where $x_i \in \chi$ are the inputs and $y_i \in \mathcal{Y}$ the outputs.

Here, the output is categorical (Hit/Non-Hit), it is then a supervised classification problem.
The problem is to explain the output type variable by the other 12 variables. I denote by y the variable to explain:

$$y = \begin{cases} 1 & \text{If the song is a hit} \\ 0 & \text{otherwise} \end{cases}$$

The input variables X1,…,X12 are the audio features defined in the previous sections.
Here the problem can be viewed as a classification rule $g: \mathbb{R}^{11} \to \{0;1\}$ such that

$$g(x) = \begin{cases} 0 & P(Y = 1 | X = x) \\ 1 & \text{otherwise} \end{cases}$$

for a threshold s∈[0,1]. Thus, the problem is to estimate P(Y=1|X=x) the probability that a song would be a hit given the 12 input variables.

V.2 - Mathematical models

I used R to test the models defined in the following section. I have observed n = 306 songs. Of the 306 songs, 40% are non-hit songs. The data are not heavily skewed, but slightly imbalanced. We need to remember this while building the models.

To make a comparison between the different algorithms, we split the data into a training set of size 75% and a test set of size 25%. For each model, I first explain it mathematically, and then interpret the output given by the algorithm trained on my data. The complete notebook can be found in annex 5.

V.2.1 - Logistic Regression

Logistic Regression is the appropriate regression analysis to conduct when the dependent variable is binary. I fit the model on the train dataset. The logistic model is defined by:

$$\log \frac{p(x)}{1-p(x)} = \beta_1 x_1 + \cdots + \beta_{12} x_{12}$$

Where $p(x)=P(Y=1|X=x)$. Unknown parameters $\beta 1,\ldots,\beta 12$ are estimated by maximum likelihood.

```
## Call:
## glm(formula = Hit ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9957  -1.0211   0.6025   0.8816   1.7763
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)       4.673156   4.401753   1.062 0.288391
## explicitTrue     -0.172027   0.381535  -0.451 0.652076
## acousticness     -0.726717   0.889824  -0.817 0.414101
## danceability      2.633478   1.424918   1.848 0.064579 .
## energy           -4.148553   1.685324  -2.462 0.013833 *
## instrumentalness -9.975751   5.818584  -1.714 0.086444 .
## liveness         -2.456308   1.294982  -1.897 0.057856 .
## loudness          0.474448   0.125012   3.795 0.000148 ***
## speechiness       1.520418   1.777756   0.855 0.392415
## tempo            -0.003589   0.005744  -0.625 0.532108
## valence           0.325024   0.883777   0.368 0.713047
## time_signature   -0.185900   0.919559  -0.202 0.839790
## duration_min      0.348900   0.233180   1.496 0.134584
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 306.01  on 228  degrees of freedom
## Residual deviance: 255.72  on 216  degrees of freedom
## AIC: 281.72
##
## Number of Fisher Scoring iterations: 7
```

The p-value for each term tests the null hypothesis that the coefficient is equal to zero (no effect). Loudness and energy have a small p-value (<0.05), we reject the null hypothesis, those variables are statistically significant. All the other variables are not statistically significant to explain Hit.

V.2.2 - Stepwise Logistic Regression

Since there are a lot of variables, I want to check which ones are the most significant ones. I propose to make a variable selection procedure with a backward selection approach using BIC (Bayesian Information Criterion).

Stepwise selection procedures define recursive models by adding or deleting one variable at each step.

Here is the process of the backward selection:

1. Let Md the full model (d variables)

2. Fork=d,...,1:

    (a) Define the k models by deleting one variable in Mk

    (b) Choose, among those k models, the one which maximizes R2. Denote Mk−1 this model.

3. Select, among M0, . . ., Md, the best model according to a given criterion.

```
## Call:
## glm(formula = Hit ~ danceability + energy + instrumentalness +
##     loudness, family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1478  -1.0795   0.6479   0.9028   1.8016
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)        3.3358     1.8027   1.850 0.064251 .
## danceability       3.1531     1.1902   2.649 0.008067 **
## energy            -3.4621     1.3941  -2.483 0.013012 *
## instrumentalness  -8.4005     5.4726  -1.535 0.124784
## loudness           0.4293     0.1170   3.670 0.000242 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 306.01  on 228  degrees of freedom
## Residual deviance: 262.41  on 224  degrees of freedom
## AIC: 272.41
##
## Number of Fisher Scoring iterations: 7
```

Here the model selected automatically the best subset with all statistically significant variables. It seems that loudness, danceability, energy are the most statistically significant variables.

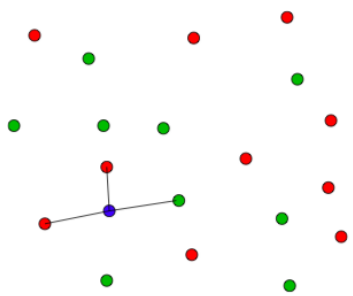
V.2.3 - K-Nearest Neighbors

With KNN I try to find a classification method which will predict whether a song is a hit or not by measuring the similarity with hit/non hit songs.

Let k ≤ n an integer. The k-nearest neighbors estimate is defined by doing a majority vote:

$$\hat{g}_n(x) = MV\big(Y_i : i \in knn(x)\big) = \underset{k \in Y.}{\operatorname{argmax}} \sum_{i \in knn(x)} 1_{Y_i = k}$$

where knn(x) = {i : Xi is among the knn of x among {X1,...,Xn}}



In this example (Figure 34), the objective is to classify a new point with either red or green color. If we take K=3 neighbors, we have two red points and one green. By doing a majority vote, the new point is classified red.

Figure 34: Example of knn classification

I will use ERM (empirical risk minimization) to find the optimal k by doing a grid search with k-fold cross validation. Since the data is imbalanced, accuracy is not the most reliable metric. Hence, I will choose 'k' based on ROC[6].

[6] The receiver operating characteristic (ROC) is defined in section VI.2 - Performance criteria.

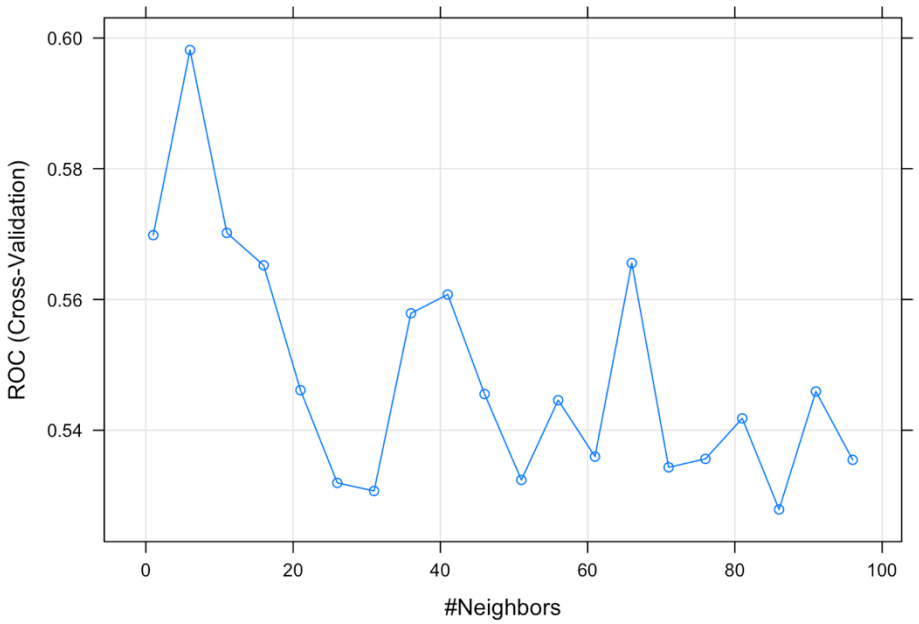


Figure 35: Evolution of accuracy according to the number of neighbors

We can see on Figure 35 that the optimal k is 6. If k increases, then accuracy decreases because of high bias.

V.2.4 - Penalized regression

I will now try to get rid of unnecessary variables by imposing a constrain on the size of the coefficient. Lasso allows to drop some variables while Ridge shrinks the coefficient. Both methods will help to reduce the variance.

Ridge regression shrinks the regression coefficients by constraining the Euclidean norm of the parameters.

Ridge estimates $\hat{\beta}^r$ minimize:

$$\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{d} \left(x_{ij}\beta_j\right)\right)^2 \text{ subject to } \sum_{j=1}^{d} \beta_j^2 \leq t$$

Or equivalently by imposing a penalty on the size of the coefficient:

$$\hat{\beta}^L = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} \left(Y_i - \beta_0 - \sum_{j=1}^{d} X_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{d} |\beta_j| \right\}$$

Lasso regression shrinks the regression coefficients by constraining the L1 norm of the parameters.

Lasso estimates $\hat{\beta}^L$ minimize:

$$\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{d} x_{ij}\beta_j\right)^2 \text{ subject to } \sum_{j=1}^{d} \beta_j \leq t$$

Or equivalently by imposing a penalty on the size of the coefficient:

$$\hat{\beta}^L = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} \left(Y_i - \beta_0 - \sum_{j=1}^{d} X_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{d} |\beta_j| \right\}$$

Variables with small coefficients are excluded from the model.

In both cases, the choice of λ is crucial.

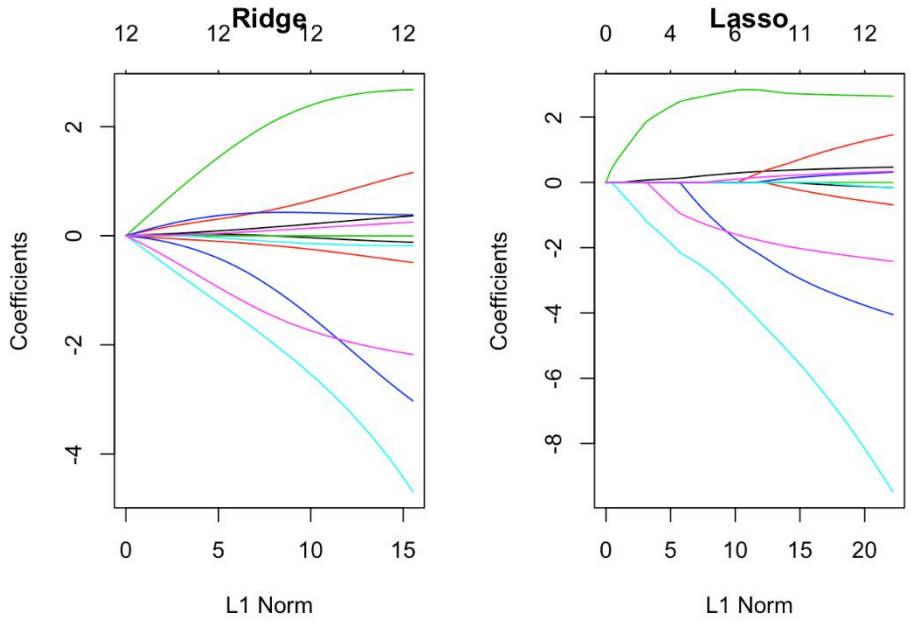


Figure 36: Evolution of the coefficient for Ridge and Lasso models

We see on Figure 36 that the coefficients shrink for Ridge and some of them become 0 for Lasso. Let's get the best shrinkage parameter for both.

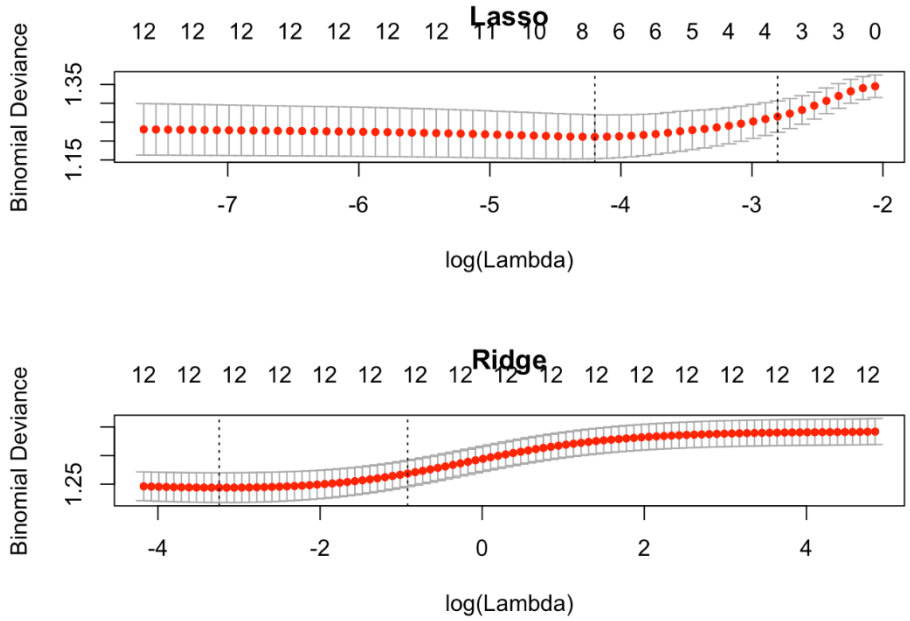


Figure 37: Evolution of λ for Ridge and Lasso models

The optimal λ is selected by doing a cross-validation.

V.2.5 -Trees

Tree algorithms are statistical learning algorithms for both regression and supervised classification.

I focus here on CART algorithm, which restricts attention to recursive binary partitions. At each step, the method splits the data into two regions.

At the end, a majority vote is done in each cell of the partition.

- Each element of the partition is called terminal node.
- The first node is the root node.
- Each split (each question) defines two child nodes, the left and right child nodes.

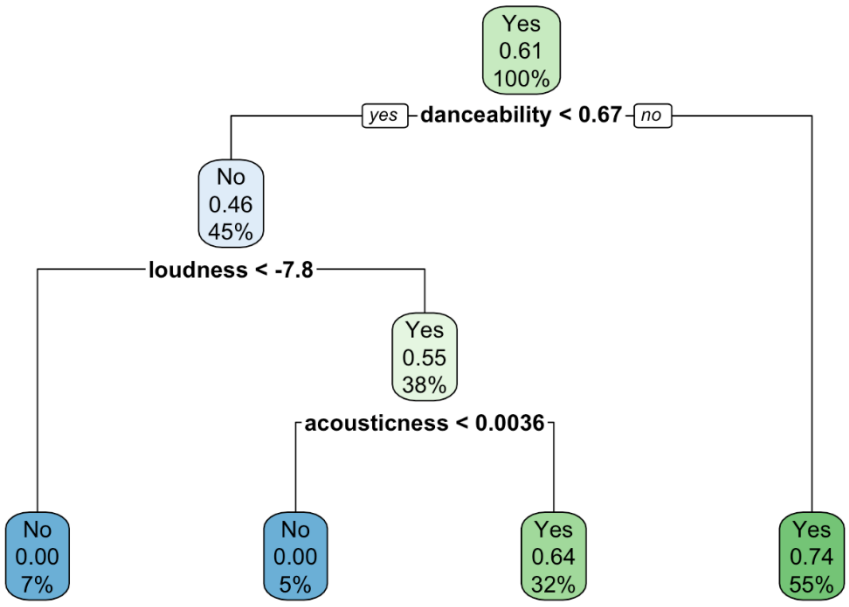


Figure 38: Decision Tree

Trees allow to build many rules based on the split of the variables and identify the important ones. The best split is the one that minimizes impurity, which is a measure of homogeneity of a node.

Figure 38 is the output of the Random Forest algorithm trained on my dataset. If danceability score is greater than 0.67, then the song is classified as being a hit. If danceability is lower than 0.67, then we have to keep on reading the left-child node. The percentages correspond to the proportion of observations in each node.

V.2.6 - Random Forest

A random forest is a collection of trees. The goal is to try to reduce correlations between the trees, to make the trees more different from each other, and get rid of overfitting.

Bagging is a set of algorithms introduced by Léo Breiman [Breiman, 1996], it comes from Bootstrap Aggregating. The idea of **Bagging** is to fit a lot of simple trees and aggregate them, by running the algorithm on different datasets to obtain different trees. We define new datasets by randomly draw dataset with replacement from the training data, this is what we call **Bootstrap.**

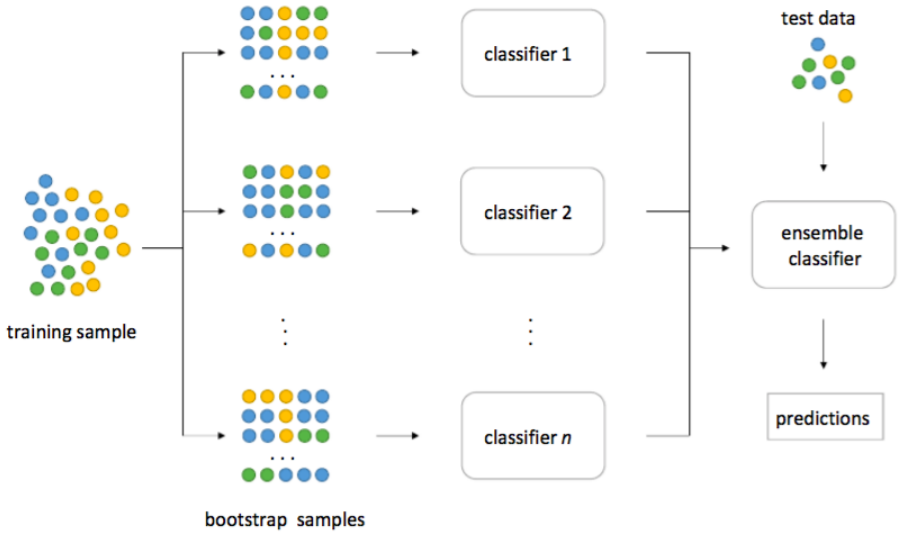


Figure 39: The Bagging approach

To predict a new value, the classification is done on each tree of the forest, before doing a majority vote.

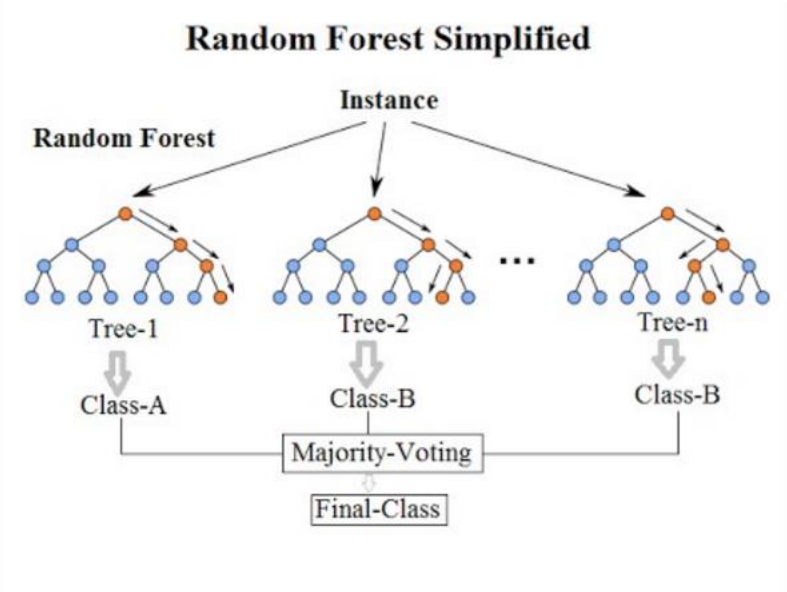


Figure 40: Random Forest operation philosophy

How does random forest perform on my dataset?

```
## Call:
##  randomForest(formula = Hit ~ ., data = train, mtry = as.double(select.mtry$bestTune))
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 31%
## Confusion matrix:
##     No Yes class.error
## No  29  60  0.67415730
## Yes 11 129  0.07857143
```

The type I error (false positive, the non-hits classified as hits) is huge. The classification error for the non-hit class is 67%. This shows the difficulty to distinguish clear patterns that could explain that a song remains subdued.

VI - Results

VI.1 - Performance criteria

To test and compare the performances of the models, we need to choose some performance criteria. For each algorithm, we estimate:

Accuracy: The accuracy of a classifier is the probability of correctly predicting the class of an unlabeled instance, P(g(X)=Y).

Specificity: The classifier specificity (also referred to as precision) is defined as the proportion of true positives on the total number of instances identified as positive [Baldi et al., 2000].

$$\text{specificity} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

With the measures defined by:

- TP or true positives, the number of correctly classified hit.
- TN or true negatives, the number of correctly classified non-hit.
- FP or false positives, the number of non-hit classified as hit.
- FN or false negatives, the number of hit classified as non-hit.

ROC: The receiver operating characteristic (ROC) curve is a graphical representation of the true positive rate (the sensitivity) as a function of the false positive rate (the so called false alarm rate, computed as FP/(FP + TN)). A perfect classifier, with a 100% true positive rate and no false positives would be represented by a (0,1) coordinate.

AUC: An indicator related to the ROC curve is the area under the curve (AUC), that is equal to 1 for a perfect classifier and to 0.5 for a random guess.

VI.2 - Algorithm comparison

We can now make a comparison between the statistical learning algorithms seen previously, applied on the dataset.

To do so, the data are split into:

- a training set of size 229 to fit and calibrate the models;
- a test set of size 77 to estimate misclassification error of each model

Method	AUC	Accuracy	Specificity
Ridge	0.6858345	0.6623377	0.8913043
Lasso	0.6781206	0.6363636	0.8695652
Logistic	0.6753156	0.6233766	0.7826087
Backward	0.6739130	0.6493506	0.8478261
Random Forest	0.6591865	0.6623377	0.9347826
Tree	0.6420056	0.6493506	0.9347826
KNN	0.5389201	0.5844156	0.7173913

Table 2: Performance comparison between selected models

Table 2 displays the results of the different models tested for chosen metrics. We can say that KNN has the worst results and is not adapted to this specific classification problem. This can indicate that there are no clear differences between the audio characteristics of a hit and a non-hit song and that those two classes can be in the same neighborhood.

Ridge, Lasso and Logistic regression have the best results in term of AUC. Random Forest and Trees have excellent specificity.

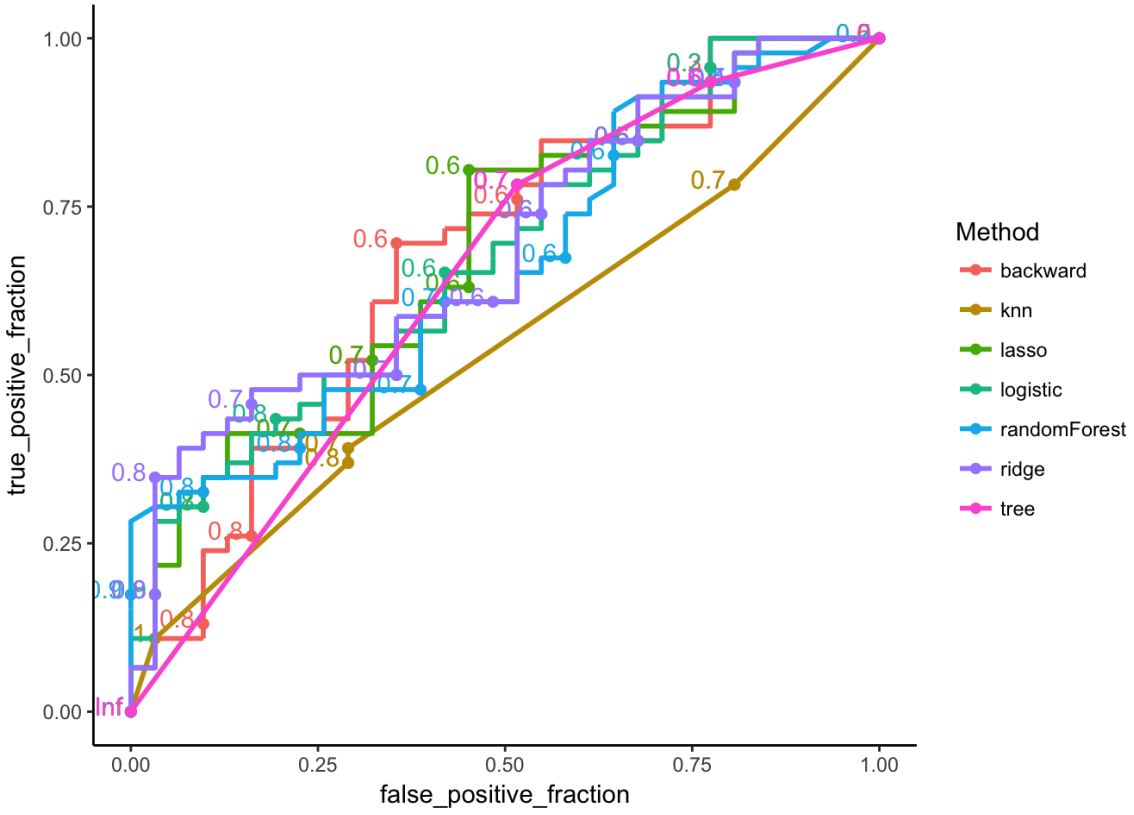


Figure 41: ROC curve for selected models

Globally, the results obtained are not bad but not excellent. This shows the difficulty to predict if a song will become a hit using its audio characteristics only. Here I do not have any information about marketing and labels strategy. This is of course something impactful, as we can imagine that a hit song is first chosen by the music label to be a hit, and then a whole promotion campaign is set up accordingly. Most of today's variety songs are made to be an instant success and are meant to be so even before their release.

Let us not forget that the more data the algorithm can be trained on, the more accurate the prediction is. We can easily imagine that streaming platforms, owning tones of data they keep for themselves, have both the data resources and the technical skills to develop sophisticated tools.

VII – Discussion

VII.1 - Business Applications

How could such predictive models be used? From a business point of view, both for music labels and streaming platforms, there are for sure some ideas to dig through.

VII.1.1 - Taste profiling

Training the algorithms on the most recent hits make it possible to catch the new trends and even predict the forthcoming ones. Those data are particularly valuable for the music labels, which would be then capable of producing a continuous flow of hits, quickly and intelligently respond the global competition, without saturating the market. Being aware of what people are listening to and are expecting is a must at a time when music industry generates incomes from tours and not music releases any longer. Data help music labels to promote the right artist, at the very best time, optimizing profits over the long run.

My models are trained on the most listened songs on Spotify worldwide. This is how I defined a hit song. With the same reasoning and in order to develop accurate recommendation systems, it would be easy to take the most listened songs of a user as the targeted feature, and return songs the user is most likely to love. With the wide range of algorithms available, we could choose to recommend either songs similar to what a user usually listens to, or make him discover new genres.
Streaming platforms are thus the owners of priceless data and are the key actors of the new music industry. Trend makers, taste influencers, sociologists, researchers, they took the lead of the music sector a few years ago and are perfectly placed to deal with the technological challenges.

As Spotify now owns the Echo Nest technology to extract audio features, it is not possible for freelance artists to audit their music.

This is the limit of such predictive models for independent individuals. Spotify owns the technology and provides only a small fraction of the available data. For all the ideas developed above, Spotify is always the key essential actor.

VII.1.2 - Innovative listening and discovery experiences

Technologies allow to transform a music signal into actionable data. Thanks to AI and deep learning, it is now possible to capture the emotional characteristics directly from the audio information. A French start-up called Niland[7] (and bought out by Spotify in 2017) took the lead of this technology by offering customers, from labels to creative agencies, to power audio searches. From a track represented in a 1000-dimensional music space, acoustic similarity and automatic classification can be performed.

Let's say you are a news broadcast and want to find your theme music. You can either describe what you are looking for (ex. Alarming, disturbing music), or come with a track you selected and you would like a similar one. In both cases, the algorithm provides you with track suggestions that might fulfill your specific wishes. The combination of music and AI has yielded a great opportunity to first create a huge music library with incredibly rich and varied data, and revolutionize the way people interact with music.

More than recommendation and predictive systems, capturing emotions is for sure one of the potential future in this field. With the imperative for the music industry to constantly reinvent itself, working on innovative listening is mandatory.

[7] More information about the company on http://niland.io

VII.2 - Toward a globalization of tastes and industry?

We have seen in section II that Spotify and other streaming services had already acquired many markets, but there is still a lot of potential, especially in Africa. Today Spotify is accused of not paying the artists enough and, most importantly, to influence the user behavior. One of the risk for such company using recommendation algorithms, is to stick the listeners into their musical comfort zone, and standardize the music tastes by proposing the same playlists to every listener. Can the streaming platforms, helped by their data and algorithms, control every aspect of the music industry?

VII.2.1 - The seasonal effect

Thanks to billions of data points and science, Spotify is able to predict more or less everything. The streaming service launched a platform call Spotify Insights[8], a website willing to tell data stories based on trends and worldwide events. From what your dog would like to listen, to the end of Game of Thrones, Spotify pretends to know everything.

How Will 'Game of Thrones' End? Stream the Creators' New Playlist to Find Out

APRIL 9, 2019

RUFF DAY

Soothing Sounds for Spot: How Streaming Can Help Your Dog Relax

MUSIC OF MARVEL

With 'Black Panther' and 'Captain Marvel,' Superhero Movie Concept Albums Take Flight

[8] Spotify Insights, https://newsroom.spotify.com/category/trends-data/

Because of a strong mass effect and similarities of behaviors within cultures, it is quite easy to predict and analyze the data linked to seasonal events. Let's take Christmas, the top streamed songs haven't changed for decades and, yes, Maria Carey broke the record for the most streams in a single day last Christmas, 24 years after the song was released. "All I want for Christmas" is indeed streamed approximately 11 million times every Christmas day. But is this song streamed because people like it or thanks to Spotify listing the song in every Christmas playlist such that you hardly can escape from listening to it?

The same phenomenon can be observed for every worldwide event and streaming data say a lot about how people behave and what they expect. For example, movies soundtracks streams are a good indicator about how did people like the film.

There is even an optimal release date for a song. Thus, most of the songs are released on January (see Figure 42), and streaming services historically refresh their catalogues every Friday, making it the industry's go-to release day.

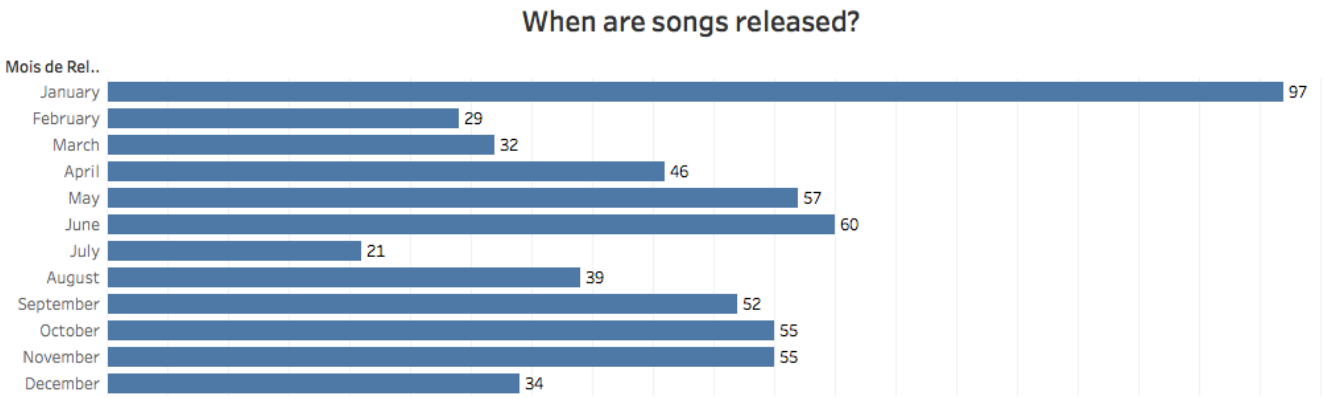


Figure 42: Spotify most played songs released month

VII.2.2 - Unpredictable events

Sadly, most the streaming records are broken after an artist's death. Before Maria Carey, the most streams record was previously held by XXXTentacion's "SAD!", which garnered 10.4 million Spotify

streams on June 19, the day after he was shot dead. Streams surged on average 1000% after an artist dies, so was the case of Avicii, Johnny Halliday and Charles Aznavour.

In every sport event, randomness is also quite important. A study led by Spotify shows the evolution of what people were listening during the Super Bowl[9], the yearly biggest sport event in the US. The study pointed out that there is a strong correlation between the state people are coming from and the music they have been listening to during the match, according to their team's results.

When it comes to sports, streaming data tell a lot. The last Fifa World Cup had been rhythmed by joyous triumphs and bitter disappointments, and data tell a pretty good story of what happened during a match without watching it. Let's take the example of Iceland playing its first-ever World Cup match on June 2019, and take a look at the streams of the national football song "Eg er kominn heim".

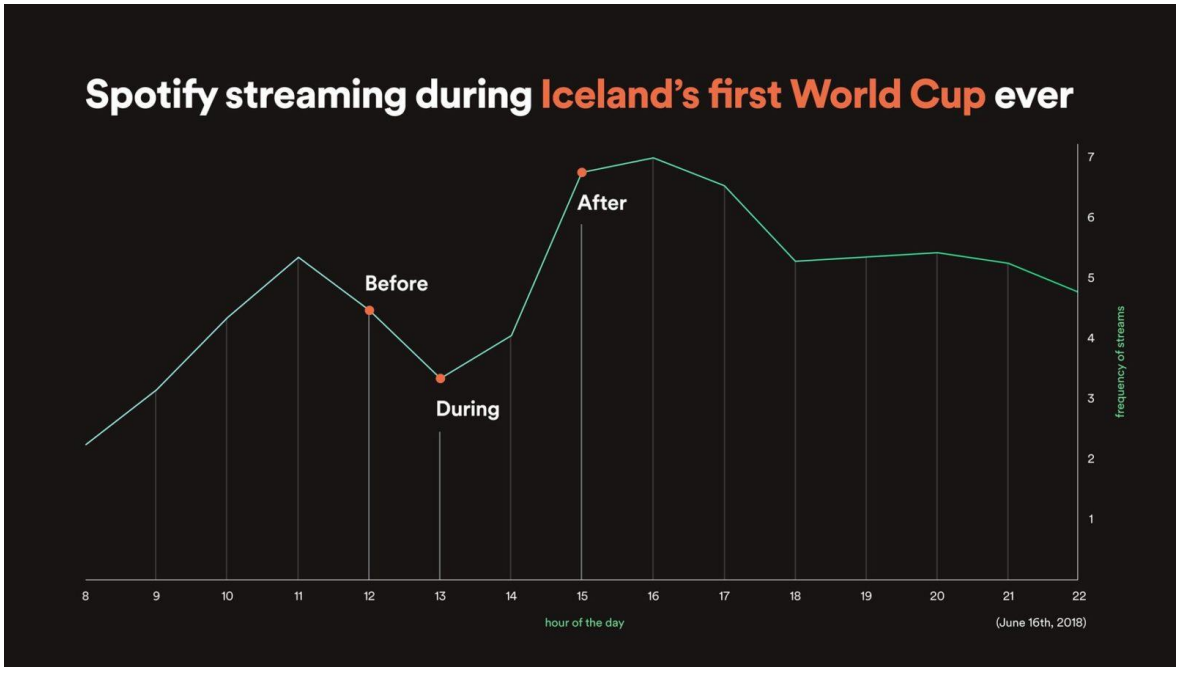
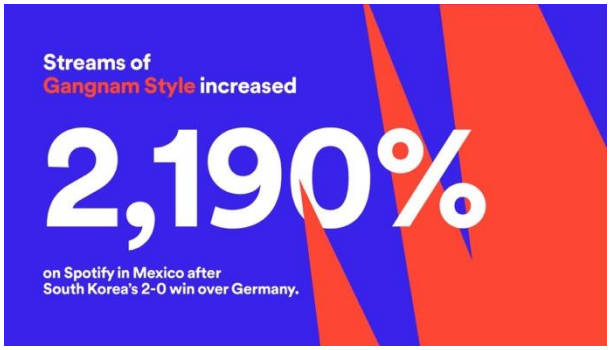


Figure 43: Spotify streaming during Iceland's first appearance (source: Spotify Insights)

[9] See https://insights.spotify.com/us/2016/02/05/data-viz-super-bowl-music/ for more details

We can see on Figure 43 that streaming is correlated to what happened during the match. When Iceland scored its first goal, streams immediately soared.



Source: Spotify

Those events are definitely punctual and, although the figures are impressive, do not influence or bias the predictions done using those data.

Conclusion

Years after years, the music industry experienced cultural trends, changing consumers tastes and booming technologies. This ever-changing business succeeded to respond to innovation and adapt format and content to make the music ecosystem work. From the physical ownership to the digital renting of music, Spotify has changed the music industry and revolutionized music consumption in less than a decade. On the back of its powerful technologies, either self-developed or acquired, Spotify took the lead on predictive analysis and recommendation systems

related to music. Streaming platforms are able to target people individually and efforts are focus on offering individuals the best experience.

Thanks to the digital area and an ever-connected world, the data collected are worth a lot. And the more the data, the more accurate the algorithms are. If predictions look easy to perform, the upcoming challenge is to respond to a collapsing listener's attention and a tendency to scroll and skip expeditiously.

More than predicting which song will be a worldwide hit, machine learning makes it possible to predict what an individual is likely to love and influence his listener's behavior.

For sure there will be something replacing the streaming area one day, impacting again both the music industry business model and the user experience. And undoubtedly AI will be there to meet the challenge. When it comes to technology, possibilities are limitless.

VIII – References

Mansoor Iqbbal, 2019, Spotify Usage and Revenue Statistics, http://www.businessofapps.com/data/spotify-statistics/

Sheldon Pearce, 2018, Considering the Rise of the Super Short Rap Song, https://pitchfork.com/thepitch/considering-the-rise-of-the-super-short-rap-song/

Bruce Houghton, 2019, Queen Spotify Streams Surge 333%, 70% From Under Age 35, https://www.hypebot.com/hypebot/2019/01/queen-spotify-streams-surge-333-70-from-under-age-35.html

Jeroen van den Hoven, 2015, ANALYZING SPOTIFY DATA, https://www.math.vu.nl/~sbhulai/papers/paper-vandenhoven.pdf

Chad Horner, 2019, 9 Surprises from Analyzing Spotify Streams Data, https://readypipe.com/blog/9-surprises-from-analyzing-spotify-streams-data/

Spotify for developers, https://developer.spotify.com/documentation/web-api/reference/tracks/get-several-tracks/

Paul Lamere, https://github.com/plamere/spotipy

Derek Zhao, https://github.com/zhao1701/spotify-song-lyric-analysis

Music Industry Insights, 2018, The Future of Streaming, https://insights.midem.com/music-business/future-streaming-2/

All Things Considered, 2009, https://www.npr.org/2009/12/31/122114058/the-loudness-wars-why-music-sounds-worse?t=1556289478468

Ohio State News, 2017, Has music streaming killed the instrumental intro?, https://news.osu.edu/has-music-streaming-killed-the-instrumental-intro/

Erik Webb, 2018, Analyzing Drake's Catalog Using Spotify's API, https://unboxed-analytics.com/data-technology/analyzing-drakes-catalog-using-spotifys-api/#analyzing-the-data

Kaylin Pavlik, 2016, 50 Years of Pop Music, https://www.kaylinpavlik.com/50-years-of-pop-music/

Eric Skelton, 2018, Why Does So Much New Music Drop at the Same Time Each Friday?, https://www.complex.com/pigeons-and-planes/2018/05/why-does-so-much-music-release-friday-each-week

Herremans, D., Martens, D., & Sörensen, K. (2014)."Dance hit song prediction" . Journal of New Music Research. 43 (3): 291–302

TH Davenport, JG Harris, 2009, IT Sloan Management Review, What people want (and how to predict it), https://analytics.typepad.com/files/2009-what-people-want-and-how-to-predict-it.pdf

Annex 1 : Tweets and Sentiments Analysis

```
library(twitteR) library(sentimentr) library(plyr) library(ggplot2) library(wordcloud) library(RColorBrewer)
library(tm) library(ggplot2) library(ggthemes)

#1. Acquire data directly from Twitter consumer_key <-
"CZ03RZy******" consumer_secret <- "FHD6p******"
access_token <- "111491231******" access_secret <-
"KXgGAjZP2*****"

setup_twitter_oauth(consumer_key, consumer_secret, access_token,

access_secret) # harvest some
tweets

some_tweets = searchTwitter("Spotify", lang="en",n=1500 )

#see what is in the text head(some_tweets)

# only keep the actual text of the tweet some_txt = sapply(some_tweets,

function(x) x$getText()) head(some_tweets) #2 start cleaning

# remove retweet entities

some_txt = gsub("(RT|via)((?:\\b\\W*@\\w+)+)", "", some_txt)

# remove at people

some_txt = gsub("@\\w+", "", some_txt)

# remove punctuation

some_txt = gsub("[[:punct:]]", "", some_txt)

# remove numbers

some_txt = gsub("[[:digit:]]", "", some_txt)

# remove html links

some_txt = gsub("http\\w+", "", some_txt)

# remove unnecessary spaces some_txt = gsub("[ \t]{2,}", "",
some_txt) some_txt = gsub("^\\s+|\\s+$", "", some_txt)

# remove non asci characters

some_txt = gsub("[^\x20-\x7E]", "", some_txt)

head(some_txt,10)

# define "tolower error handling" function
```

```
tryTolower <- function(x){

  # return NA when there is an error

  y = NA   # tryCatch error

  try_error = tryCatch(tolower(x), error = function(e) e)

  # if not an error

  if (!inherits(try_error, 'error'))

    y = tolower(x)   return(y)

}

# lower case using try.error with sapply

some_txt = sapply(some_txt, tryTolower)

# remove NAs in some_txt some_txt =
some_txt[!is.na(some_txt)]

names(some_txt) = NULL #write out clean
data

write.csv(some_txt, file = "spotify.csv") read.csv("spotify.csv")

#use old data now

spotify <- read.csv(file="spotify.csv", header=TRUE, sep=",") some_txt<-

as.character(spotify$x) #3 corpus and bag of words creation

clean.corpus<-function(corpus){   corpus <- tm_map(corpus,
removePunctuation)   corpus <- tm_map(corpus, stripWhitespace)
corpus <- tm_map(corpus, removeNumbers)

  corpus <- tm_map(corpus, content_transformer(removeURL))   corpus <- tm_map(corpus,
content_transformer(tryTolower))   corpus <- tm_map(corpus, removeWords,
custom.stopwords)

  return(corpus)

}

#Create custom stop words and auxiliary functions custom.stopwords <- c(stopwords('english'),
'movie','music') removeURL <- function(x) gsub("http[^[:space:]]*", "", x)

#bigram token maker

#Keep the meta data, apply the functions to make a clean corpus
```

```
head(some_txt)

id<-seq(1,length(some_txt))

#create id column

dd<-data.frame(doc_id=id,text=some_txt)

#Vcorpus want the data in this specific format corpus <-
VCorpus(DataframeSource(dd)) corpus<-clean.corpus(corpus)
#perform the cleaning with one function

#Make a Term Document Matrix depending on analysis

bigram.tokenizer <-function(x) unlist(lapply(ngrams(words(x), 2),

paste, collapse = " "), use.names = FALSE)

trigram.tokenizer <-function(x) unlist(lapply(ngrams(words(x), 3),

paste, collapse = " "), use.names = FALSE)

#use bigrams as terms

tdm<-

TermDocumentMatrix(corpus,control=list(tokenize=bigram.tokenizer))

#if you don't want to use bigrams but unis or tris

tdm.uni<-TermDocumentMatrix(corpus)

tdm.tri<-

TermDocumentMatrix(corpus,control=list(tokenize=trigram.tokenizer))

tdm.m <- as.matrix(tdm)#convert to matrix

tdm.v <- sort(rowSums(tdm.m),decreasing=TRUE)#calc freq and sort the terms

tdm.df <- data.frame(word = names(tdm.v),freq=tdm.v)#combine the terms with their frequencies

tdm.tri.m <- as.matrix(tdm.tri)#convert to matrix

tdm.tri.v <- sort(rowSums(tdm.tri.m),decreasing=TRUE)#calc freq and

sort the terms

tdm.tri.df <- data.frame(word =

names(tdm.tri.v),freq=tdm.tri.v)#combine the terms with their frequencies

#4. Frequency plot of the bigrams top.words<-subset(tdm.df, tdm.df$freq>=20)
```

```
#select a subset of all the bigrams

top.words <- top.words[order(top.words$freq, decreasing=F),]

top.words$word<-factor(top.words$word, levels=unique(as.character(top.words$word)))

#your x variables needs to be a factor in order to have a nice barplot ggplot(top.words, aes(x=word,
y=freq))+geom_bar(stat="identity",

fill='darkred') +coord_flip()+theme_gdocs()+

  geom_text(aes(label=freq), colour="white",hjust=1.25, size=5.0)

#5. Hierarchical dendogram on trigrams tdm2 <-
removeSparseTerms(tdm.tri, sparse=0.99)

#select terms that are less than 99% empty tdm2.df<-
as.data.frame(inspect(tdm2)) hc <- hclust(dist(tdm2.df))
clusMember <- cutree(hc, 4)

labelColors <- c("#CDB380", "#036564", "#EB6841", "#EDC951") plot(hc,yaxt='n')

#6. Word cloud library(wordcloud)

#look at all available color pallettes & choose one

display.brewer.all() pal <- brewer.pal(8,
"Dark2")

pal <- pal[-(1:2)]

wordcloud(tdm.df$word,tdm.df$freq,max.words=500,scale=c(2,1.4),

random.order=FALSE,colors=pal)

#words base on their frequencies is used here

wordcloud(tdm.tri.df$word,tdm.tri.df$freq,max.words=500,scale=c(1,0.4),

 random.order=FALSE,colors=pal)

#one with trigram

#7. Sentiment Analysis: no TDM needed here

library(syuzhet)

mySentiment <- get_nrc_sentiment(some_txt)

#score every word head(some_txt)
head(mySentiment)

tweets <- cbind(some_txt, mySentiment)
```

```
#bind sentiment back with the word

head(tweets)

#combine tweets with a sentiment

sentimentTotals <- data.frame(colSums(tweets[,c(2:11)]))

#sum the different sentiments

#prepare everything for a barplot names(sentimentTotals) <- "count"

sentimentTotals <- cbind("sentiment" = rownames(sentimentTotals),

sentimentTotals)

rownames(sentimentTotals) <- NULL

ggplot(data = sentimentTotals, aes(x = sentiment, y = count)) +   geom_bar(aes(fill = sentiment), stat =
"identity") +

  theme(legend.position = "none") +

  xlab("Sentiment") + ylab("Total Count") + ggtitle("Total Sentiment

Score for All Tweets")
```

Annex 2: How to get data from Spotify ¶

Author: Marine JACQUEMIN-LORRIAUX

Here is a code to extract data from Spotify, using Spotify API.

To do so, here are the requirements:

- Create a Spotify account and get a user name
- Get yourCredentials
- Get your secret key

Got all this ? Let's get started !

```
In [1]: import
        spotipy
        import
        pandas as
        pd
        from spotipy.oauth2 import SpotifyClientCredentials


        # authenticate and connect to the API
                                        client_credentials_manager =
                            SpotifyClientCredentials(client_id='*****'
                                                client_secret='** sp =
         spotipy.Spotify(client_credentials_manager=client_credentials
                                                                _mana
```

Extract data from a given playlist

To extract data from a given public playlist, you will need:

- the name of whoever creates
- the playlist the playlist ID

In this example, I'll get the complete Spotify playlist of the most played songs in 2018.

As the user_playlist function is by default limited to 100 songs, let's create first the **get_playlist_tracks function** that gets the tracks of the whole playlist. Then, we loop over the tracks to return a list of the tracks' ids.

In [7]:
```
# get track ids from playlist

def get_playlist_tracks(username,playlist_id):
    results = sp.user_playlist_tracks(username,playlist_id)
    tracks = results['items']
    while results['next']:
        results = sp.next(results)
        tracks.extend(results['items'])
    return tracks

tracks = get_playlist_tracks('Spotify', '37i9dQZF1DX1HUbZS4LEyL') # ins

ids = []
for item in tracks:
    track = item['track']
    ids.append(track['id'])
```

We can now get the features we need for each track

In [8]:
```
# get song info and audio analysis
from song ids def
getTrackFeatures(id):       meta =
sp.track(id)
      features = sp.audio_features(id)

#Features can be removed/added according to the needs.
      # Meta
      name = meta['name']
album =
meta['album']['name']
      artist =
meta['album']['artists'][0]['name']
release_date =
meta['album']['release_date']
duration_ms = meta['duration_ms']
popularity = meta['popularity']
explicit = meta['explicit']
      available_markets =
meta["available_markets"]
      #image_url = meta['album']['images'][1]['url'] #get the
300x300 f
      # Features
      acousticness =
features[0]['acousticness']
danceability = features[0]['danceability']
energy = features[0]['energy']
      instrumentalness =
features[0]['instrumentalness']       liveness
= features[0]['liveness']       loudness =
features[0]['loudness']       speechiness =
features[0]['speechiness']       tempo =
features[0]['tempo']       valence =
features[0]['valence']
      time_signature =
features[0]['time_signature']

      track = [name, album, artist, release_date,
duration_ms, populari                acousticness,
danceability, energy, instrumentalness, li
speechiness, tempo, valence, time_signature]       return
track
```

In [9]:
```
# loop over track ids to
create dataset tracks = [] for
i in range(0, len(ids)):
```

```
            track =
            getTrackFeatures(ids[i])
            tracks.append(track)

df = pd.DataFrame(tracks, columns = ['name', 'album', 'artist', 'releas
                                     'explicit','available_markets', 'a
                                     'instrumentalness', 'liveness', 'l

            'time_signature'])
```

We get a dataset with chosen features ready to be downloaded.

In [10]:
```
df.head()
```

Out[10]:

	name	album	artist	release_date	duration_ms	popularity	explicit	availab
0	God's Plan	Scorpion	Drake	2018-06-29	198973	90	True	AU, B BO, B
1	SAD!	?	XXXTENTACION	2018-03-16	166605	92	True	[AD, AU, B BO, B
2	rockstar (feat. 21 Savage)	beerbongs & bentleys	Post Malone	2018-04-27	218146	92	True	[AD, AU, B BO, B
3	Psycho (feat. Ty Dolla $ign)	beerbongs & bentleys	Post Malone	2018-04-27	221440	89	True	[AD, AU, B BO, B
4	In My Feelings	Scorpion	Drake	2018-06-29	217925	89	True	[AD, AU, B

In [157]:
```
#Save final dataset
df.to_csv("/Users/*INSERT_PATH*/most_played_2018.csv" , sep = ',')
```

Annex 3: Principal Component Analysis on the most played songs dataset

I do a principal component analysis (PCA) on **audio features** on the most streamed songs on Spotify (songs that have been played more than 200 million times)

In [14]:
```
#import libraries
import pandas as pd
import seaborn as sns
import numpy as np
```

In [4]:
```
# load data
data = pd.read_csv("/Users/marinejacquemin/Desktop/Thesis/data/most
_played_2.csv",index_col=0)
```

In [5]:
```
#drop non float column and popularity
data.dtypes
```

Out[5]:
```
name                 object
album                object
artist
object release_date
object duration_ms
int64 popularity
int64 explicit
bool acousticness
float64 danceability
float64 energy
float64 instrumentalness
float64 liveness
float64 loudness
float64 speechiness
float64 tempo
float64 valence
float64 time_signature
int64 dtype: object
```

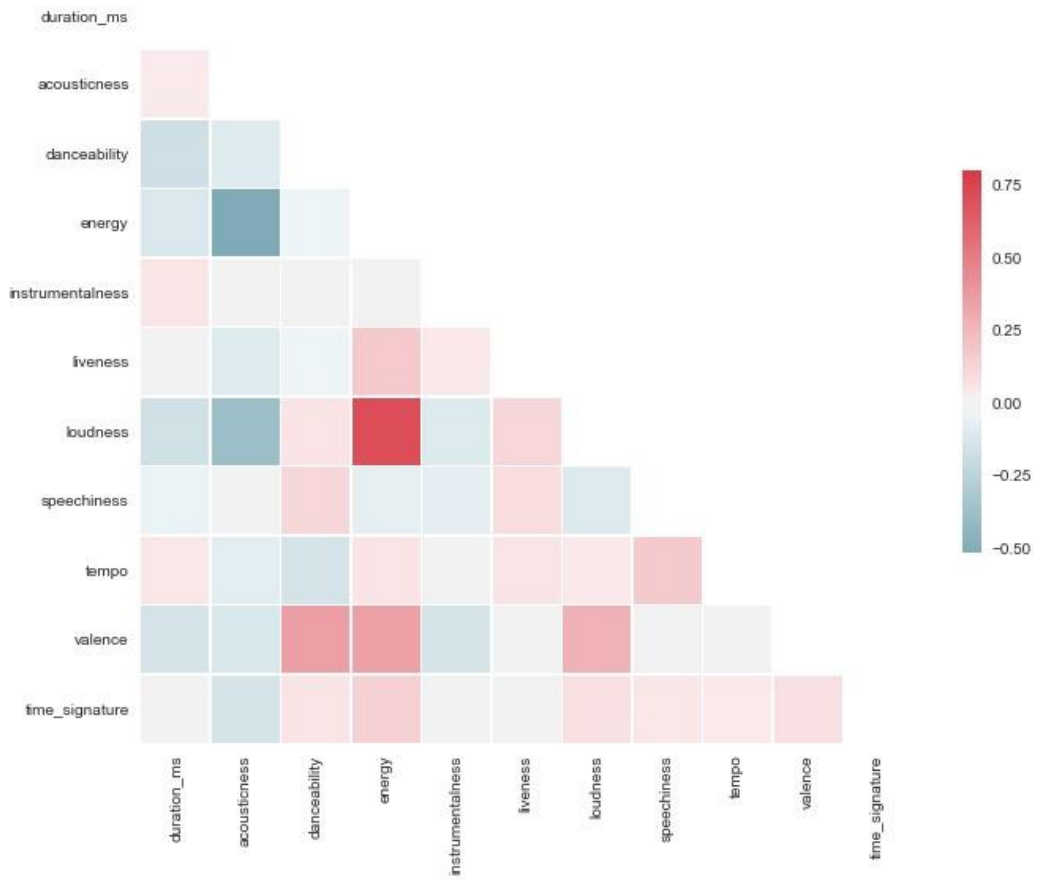
In [6]:
```
df = data.select_dtypes(['number'])
```

In [7]:
```
df = df.drop(['popularity'],axis=1)
```

In [16]:
```
sns.set(style="white")

# Compute the correlation
matrix corr = df.corr()
```

```
# Generate a mask for the upper
triangle mask = np.zeros_like(corr,
dtype=np.bool)
mask[np.triu_indices_from(mask)] =
True

# Set up the matplotlib figure
f, ax =
plt.subplots(figsize=(11, 9))

# Generate a custom diverging colormap
cmap = sns.diverging_palette(220, 10, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.8, center=0,
square=True, linewidths=.5, cbar_kws={"shrink": .5})
```

```
Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x1a23970630>
```



Before going any further, the data need to be standardized not to bias the results

In [8]:
```
X = df.iloc[:,:].values

#on standardise les variables
from sklearn import preprocessing

std_scale = preprocessing.StandardScaler().fit(X)
X_scaled = std_scale.transform(X)
X_scaled.shape
```

Out[8]: (590, 11)

PCA with 3 PC

In [9]:
```
#Let's do a try with 3 PC
from sklearn import decomposition

pca = decomposition.PCA(n_components=3)
pca.fit(X_scaled)
```

```
Out[9]: PCA(copy=True, iterated_power='auto', n_components=3,
                                                    random_state
=None,   svd_solver='auto', tol=0.0,
whiten=False)
```

In [10]:
```
print (pca.explained_variance_ratio_)
print
(pca.explained_variance_ratio_.sum())
```

```
[0.21778184 0.12938341 0.1117616 ]
0.4589268473042103
```

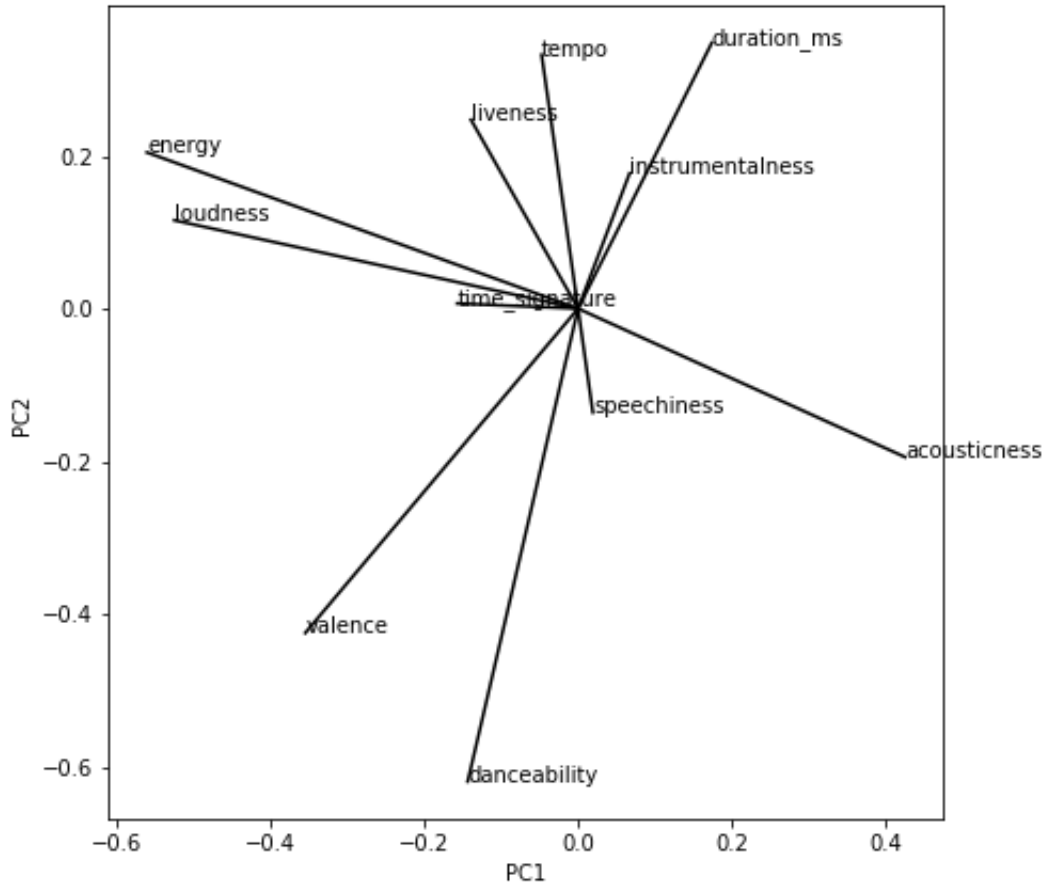
The first component explains 22% of the variability. The first three components explain 46% of the variability.

PC 1 & 2

In [11]:
```
%pylab inline

pcs = pca.components_
fig = plt.figure(figsize=(7,7))
for i, (x, y) in enumerate(zip(pcs[0, :], pcs[1, :])):
     plt.plot([0, x], [0, y], color='k')
     plt.text(x,y,df.columns[i])


plt.xlabel('PC1')
plt.ylabel('PC2')
```

```
Populating the interactive namespace from numpy and matplotlib
```

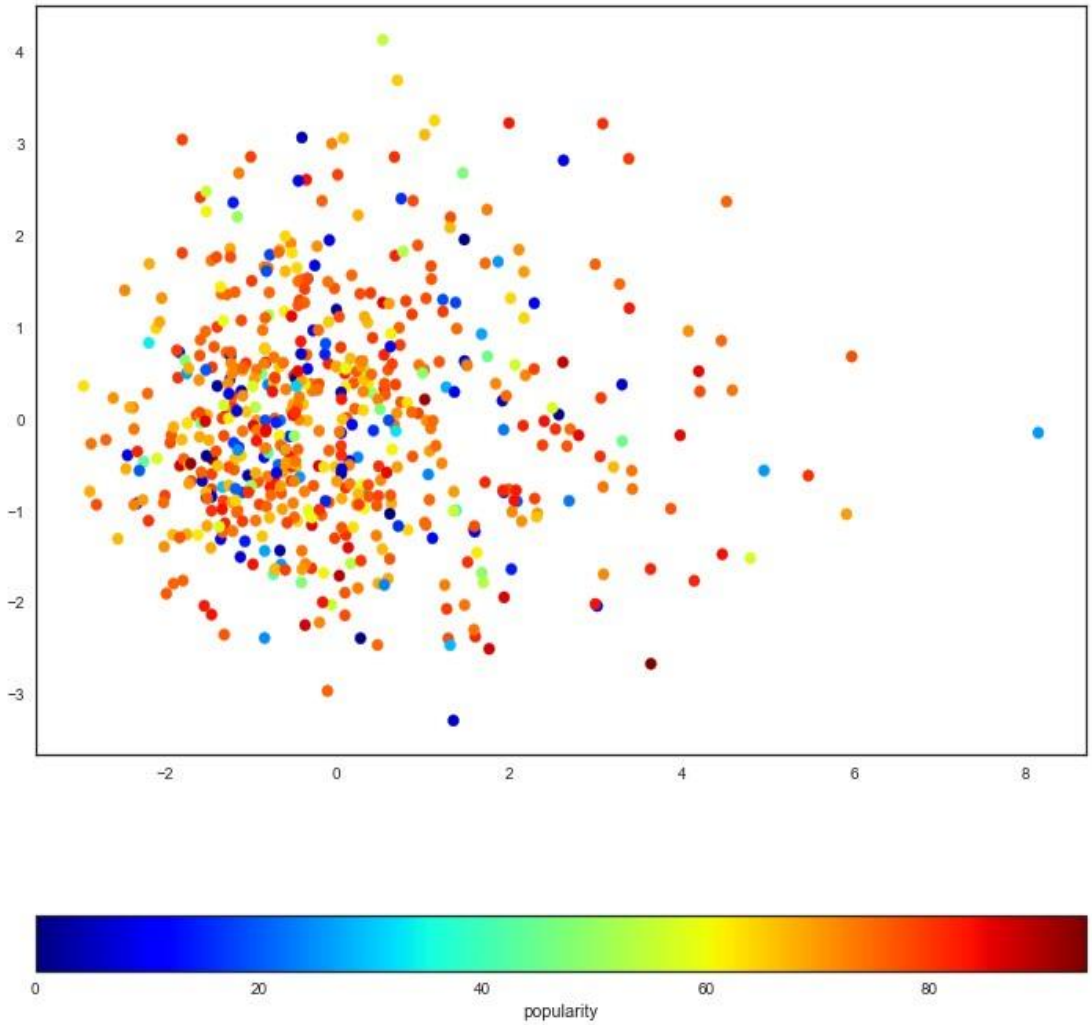
Out[11]: Text(0,0.5,'PC2')



From the correlation circle, we can say that:

- energy and acousticness are negatively
- correlated energy and danceability are
- uncorrelated danceability and duration are
- negatively correlated valence and duration are negatively correlated

In [17]:
```
#Let's see the results graphically

X_projected= pca.transform(X_scaled)
fig = plt.figure(figsize=(12,12))
plt.scatter(X_projected[:,0], X_projected[:,1],
c=data.get('popularity'),cmap = plt.cm.jet)

cbar =plt.colorbar(orientation='horizontal',label='popularity')
```

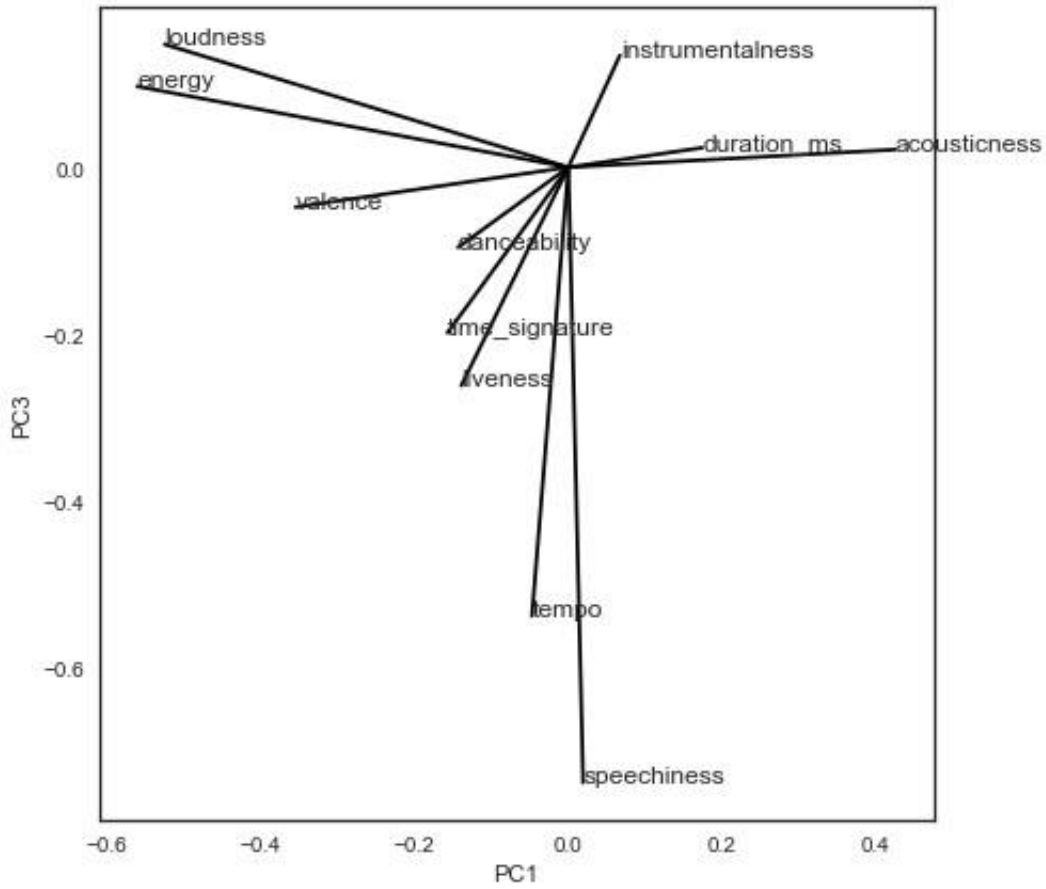


Popularity is not a reliable feature can thus can't be used to define clusters.

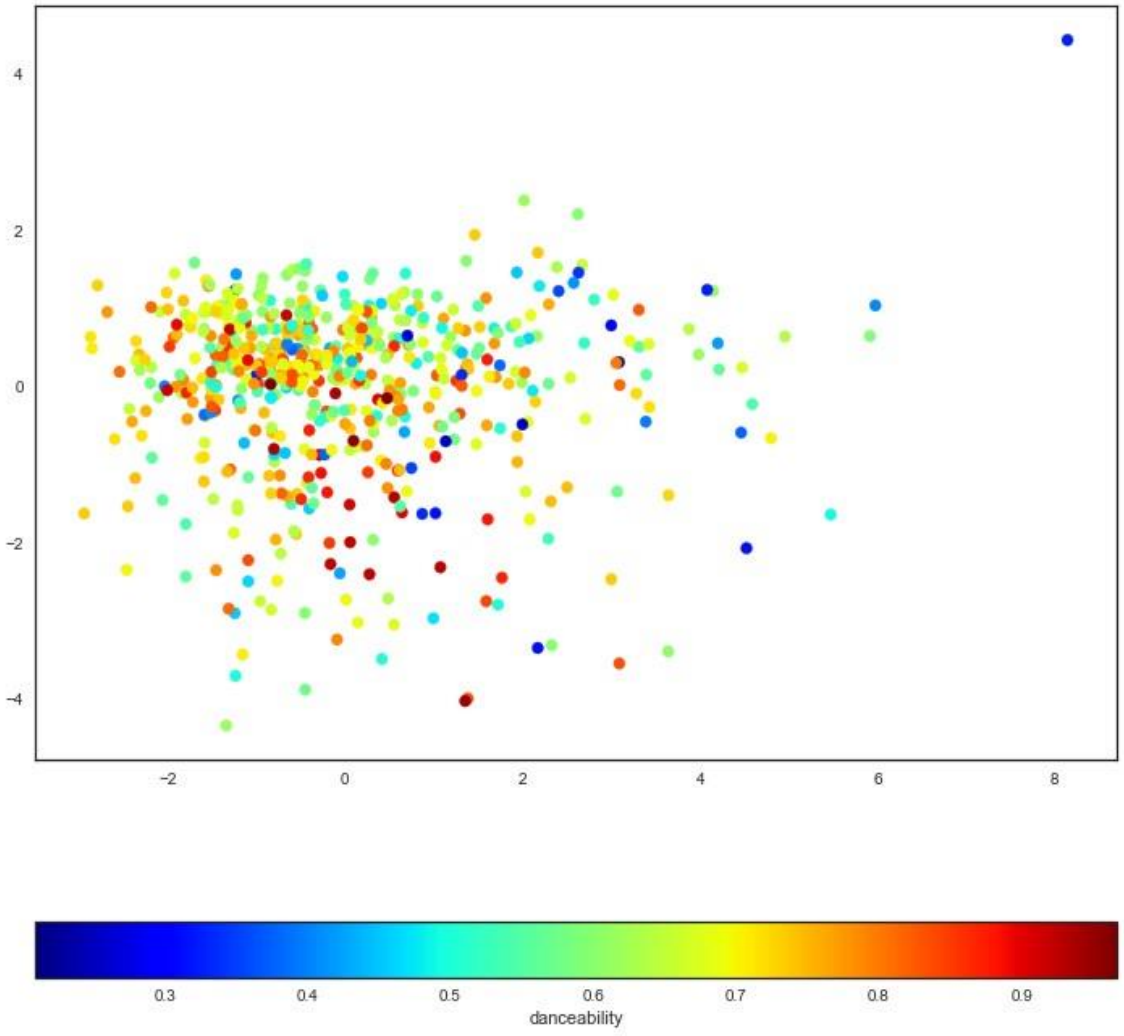
PC 1 & 3

In [18]:
```
pcs = pca.components_
fig = plt.figure(figsize=(7,7))
for i, (x, y) in enumerate(zip(pcs[0, :], pcs[2, :])):
     plt.plot([0, x], [0, y], color='k')
     plt.text(x,y,df.columns[i])


plt.xlabel('PC1')
plt.ylabel('PC3')
```

Out[18]: Text(0,0.5,'PC3')



In [19]:
```
#Let's see the results graphically

X_projected = pca.transform(X_scaled)
fig = plt.figure(figsize=(12,12))
plt.scatter(X_projected[:,0], X_projected[:,2],
c=data.get('danceability'),cmap = plt.cm.jet)

cbar =plt.colorbar(orientation='horizontal',label='danceability')
```



From the correlation circle, we can say that:

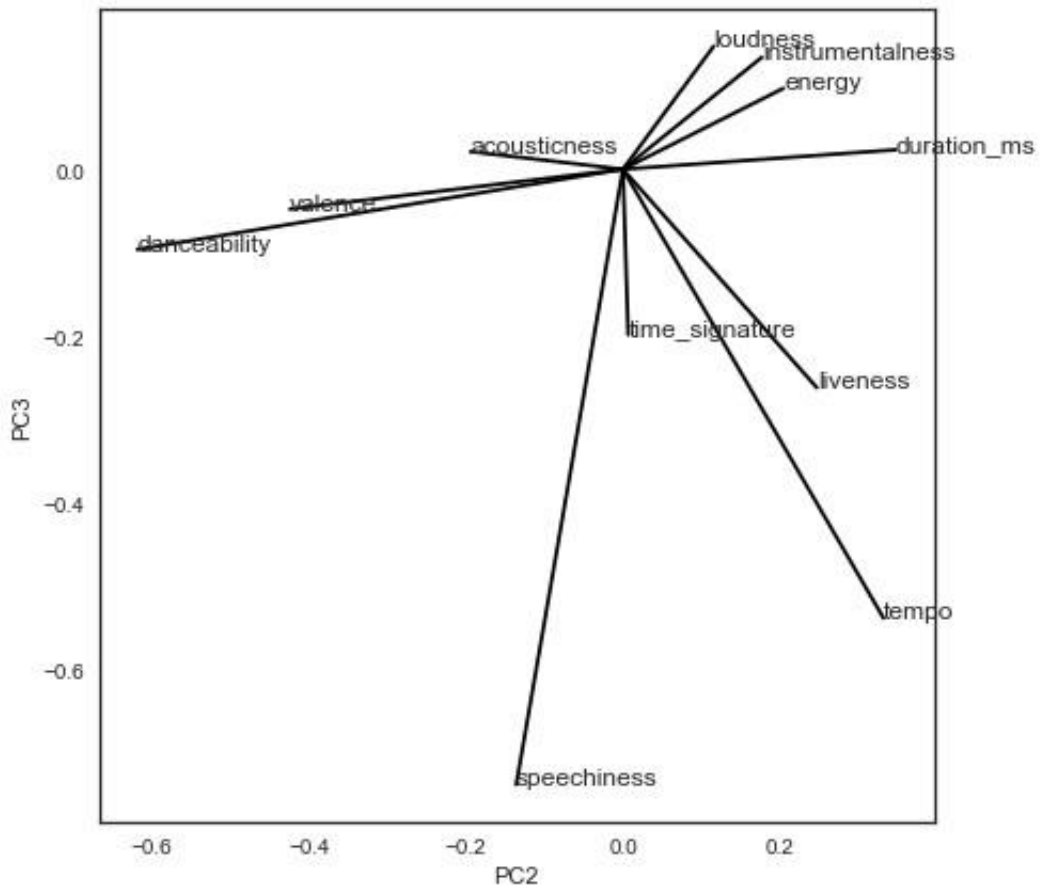
- instrumentalness and speechness are negatively
- correlated energy and loudness are correlated
- loudness and acousticness are negatively
-

correlated speechiness and acousticness are uncorrelated

PC 2 & 3

In [20]:
```
#Contribution of each variable to the PC (2&3)

pcs = pca.components_
fig = plt.figure(figsize=(7,7))
for i, (x, y) in enumerate(zip(pcs[1, :], pcs[2, :])):
     plt.plot([0, x], [0, y], color='k')
     plt.text(x,y,df.columns[i])


plt.xlabel('PC2')
plt.ylabel('PC3')
```

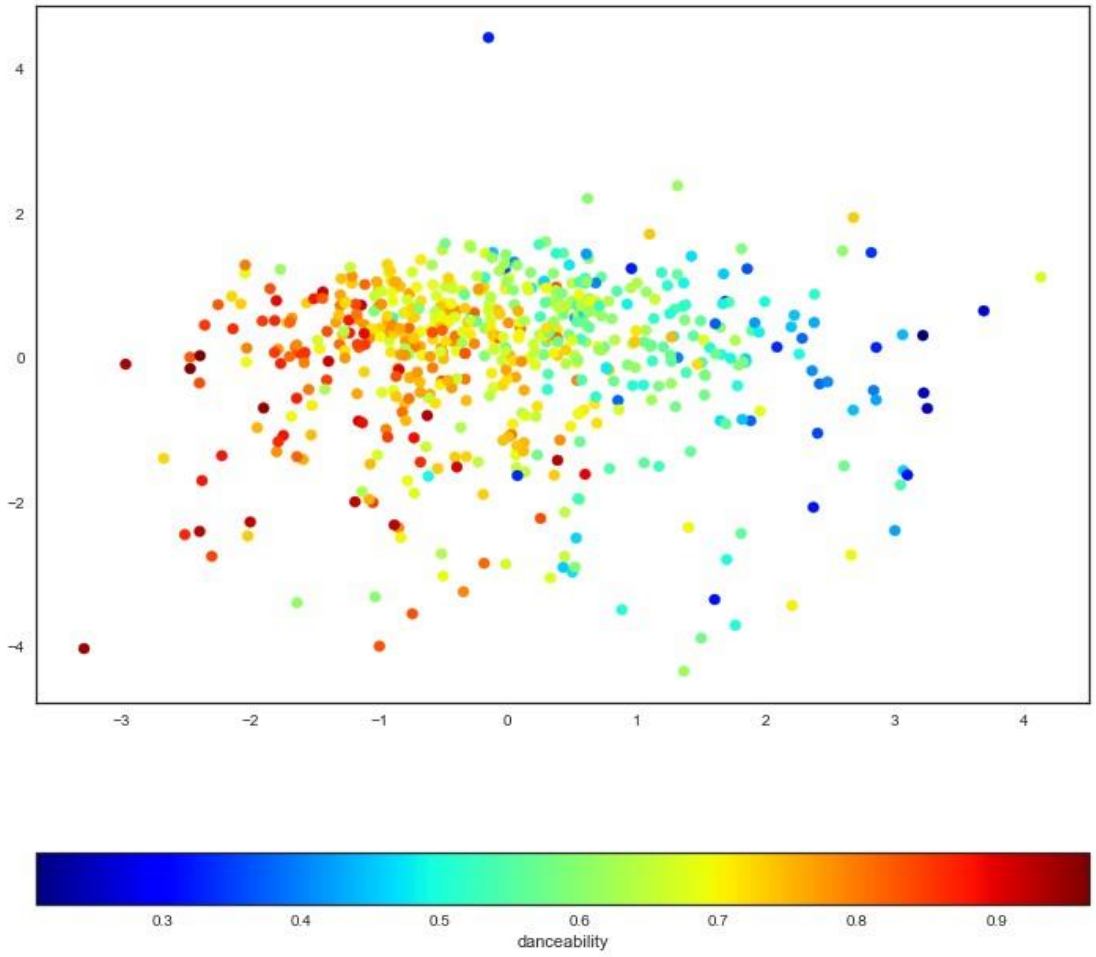
Out[20]: Text(0,0.5,'PC3')



From the correlation circle, we can say that:

- danceability and duration are negatively correlated

In [21]:
```
#Let's see the results graphically

X_projected = pca.transform(X_scaled)
fig = plt.figure(figsize=(12,12))
plt.scatter(X_projected[:,1], X_projected[:,2],
c=data.get('danceability'),cmap = plt.cm.jet)

cbar =plt.colorbar(orientation='horizontal',label='danceability')
```



Annex 4: Lyrics Analysis

The objective of this work is to analyze the lyrics of the Top 100 Billboard charts from 1960 to 2017. Comments can be found in section IV.1.4 of this Master Thesis.

Author : Marine Jacquemin-Lorriaux

In [2]:
```
# import the libraries needed

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import ENGLISH_STOP_WORDS,
Cou ntVectorizer
from wordcloud import WordCloud from
nltk.corpus import stopwords from
sklearn.pipeline import Pipeline from
sklearn.naive_bayes import BernoulliNB
```

In [3]:
```
# load the data df = pd.read_csv('/Users/marinejacquemin/Desktop/Thesis/data/billbo
ards.csv')
```

The most frequent words

In [19]:

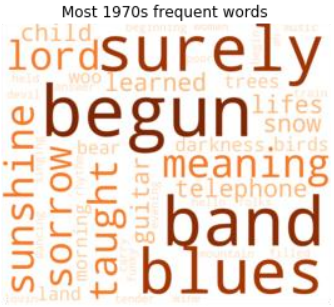
```
# define list of words
lyrics = df['lyrics'].dropna().values.tolist()

# calculate word frequencies
stop_words = list(ENGLISH_STOP_WORDS) + ['im', 'dont']
count = CountVectorizer(stop_words=stop_words)
count.fit(np.array(lyrics))
freqs = np.array(count.transform(np.array(lyrics)).todense()).sum(a
xis=0)

# make word cloud
word_freq_dict = dict(zip(count.get_feature_names(), freqs))
wordcloud = WordCloud(background_color='white', height=600, min_fon
t_size=2,
                      width=1200,
                      max_words=600)
wordcloud = wordcloud.generate_from_frequencies(word_freq_dict)

# display word cloud
plt.figure(figsize=(16,8))
ax = plt.gca()
ax.imshow(wordcloud, interpolation='bilinear')
ax.axis('off');
```

Most frequent words per decade

In [60]:
```
def ngram_wordclouds(n=1, min_df=50):

    # make features and
targets    X = lyrics.copy()
  y = df[df['lyrics'].notnull()]['decade']

  # make and fit Naive Bayes model
stop_words = list(ENGLISH_STOP_WORDS)
stop_words.extend(['shes', 'yes', 'didnt'])

  pipe = Pipeline([('count', CountVectorizer(min_df=min_df, binar y=True,

                    ngram_range=(n,n), stop_words=s top_words)),

            ('bnb', BernoulliNB(binarize=0))])    pipe.fit(X, y)

  # calculate class conditional feature probabilities and
uncondi tioned feature probabilities.
  feature_counts = pipe.named_steps.count.transform(lyrics).toarr ay().sum(axis=0)

  class_feature_counts = pipe.named_steps.bnb.feature_count_    con_feat_probs =
class_feature_counts / class_feature_counts.su m(axis=1).reshape(-1, 1)

  unc_feat_probs = feature_counts / feature_counts.sum()

  # conditional probability / unconditional probability =
lift    lift = con_feat_probs / unc_feat_probs + 1e-10

  # make subplot axes
  fig, axes = plt.subplots(2, 3, figsize=(16,10))    axes = axes.ravel()

  colormaps = ['YlOrBr', 'Oranges', 'Reds', 'Blues', 'Greens', 'P uRd']

vocab = pipe.named_steps.count.get_feature_names()    for i, decade
                in enumerate(range(1960, 2020, 10)):
    # make word-weight dictionary
    color_func = colormap_size_func(colormaps[i], 300)       feature = lift[i,:]

    dictionary = dict(zip(vocab, feature))


    # make word cloud
    wordcloud = WordCloud(background_color='white', height=1200

,

                  min_font_size=5, width=1400, color_fu nc = color_func,
```

```
                    max_words=45, relative_scaling=0.5,
max_font_size=300)
    wordcloud = wordcloud.generate_from_frequencies(dictionary)

    # display word cloud
    axes[i].imshow(wordcloud, interpolation='bilinear')       axes[i].axis('off')
    axes[i].set(title='Most {}s frequent words'.format(decade))
```

In [61]:
```
ngram_wordclouds ()
```



In [51]:
```
# when creating word clouds, maps word fontsize to a color
class colormap_size_func( object ):
    def __init__ ( self , colormap , max_font_size ):
        import matplotlib.pyplot as plt
        self . colormap = plt . cm. get_cmap ( colormap )
        self . max_font_size = max_font_size

    def __call__ ( self , word , font_size , position , orientation ,
                   random_state =None, ** kwargs ):
        if random_state is None:
            random_state = Random ()
        r , g , b , _ = 255 * np . array ( self . colormap ( font_size / self .
max_font_size ))
        return "rgb( {:.0f}, {:.0f}, {:.0f}) " . format ( r , g , b)
```

Words frequency

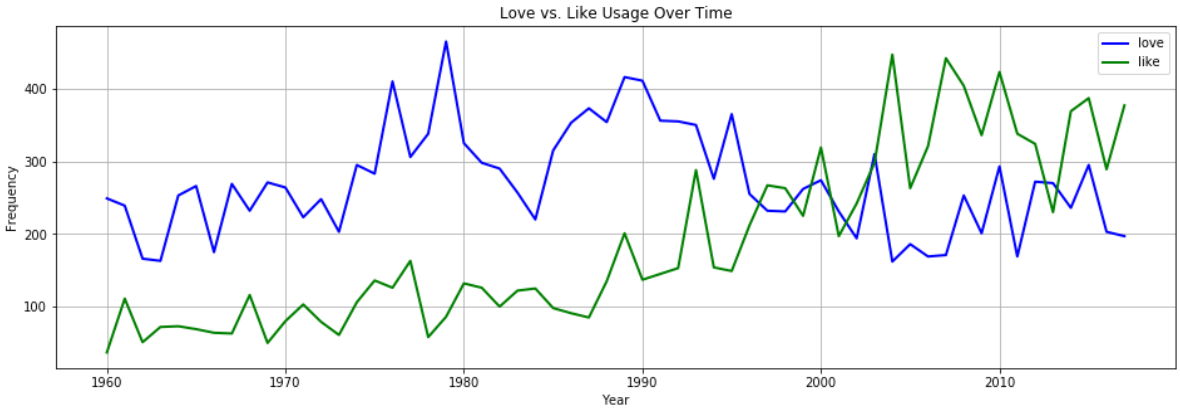
Here I selected specific words and check their evolution over time.

In [34]:
```
# convert each row into a dataframe where each row is a word instea d of song def row_to_df(x):    if not isinstance(x['lyrics'], str): return None    else:
    nonlyrics = x.drop('lyrics')

    lyrics = pd.Series(x.lyrics.split()).value_counts()       lyrics = pd.DataFrame(lyrics).reset_index()       lyrics.columns = ['word', 'count'] lyrics['dummy'] = 1

    nonlyrics = pd.DataFrame(nonlyrics).T       nonlyrics['dummy'] = 1

    return pd.merge(nonlyrics, lyrics, on='dummy')

# concatenates all word-based dataframes into a single word-based d ataframe def tidy_df(df):
  df_list = list()    for index, row in df.iterrows():

    song_df = row_to_df(row) df_list.append(song_df)    return pd.concat(df_list, axis=0)
```

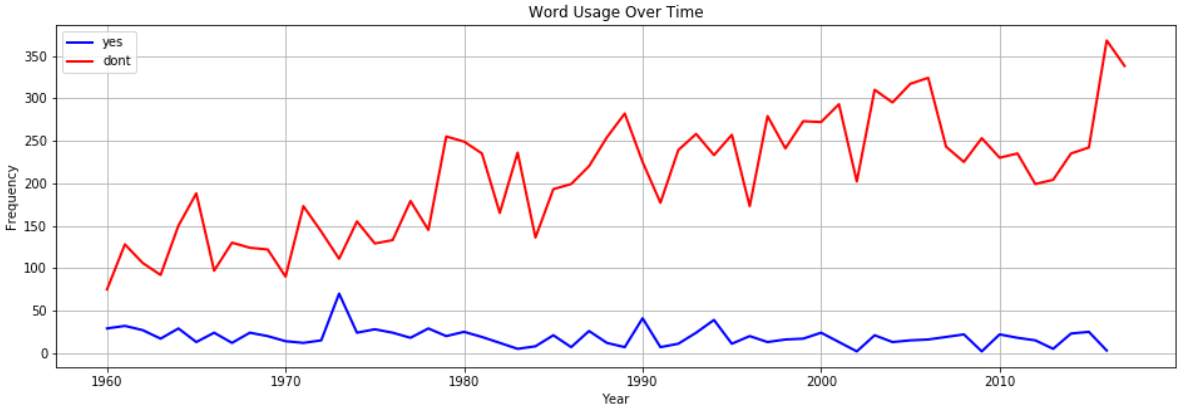
In [35]:
```
df_nonnull = df[df['lyrics'].notnull()]

#lyrics = df_nonnull['lyrics']  df_tidy = tidy_df(df_nonnull)

# make list of words that appear in at least 10 unique songs songs_per_word = df_tidy.groupby('word')['song'].agg(pd.Series.nuni que)

songs_per_word = songs_per_word[songs_per_word >= 10] words_to_keep = list(songs_per_word.index)

# keep only words that appear in at least 10 unique songs and their corresponding years of appearance mask = df_tidy['word'].apply(lambda x: x in words_to_keep) df_tidy = df_tidy.loc[mask,:][['word', 'year', 'count']]

# calculate frequencies of surviving words words_freq = df_tidy.groupby(['year','word'])['count'].sum() wf2 = words_freq.reset_index()
```

In [68]:
```
def word_time_series ( words , colors ):
    plt . figure ( figsize =( 16 , 5))
    ax = plt . axes ()
    words = words
    colors = colors
    for word , color in zip ( words , colors ):
        wf2 [ wf2 [ 'word' ] == word ] . plot . line (
            x = 'year' , y = 'count' , ax = ax , label =word , color =color , li
newidth =2 )
    ax . grid ( True)
    ax . set ( ylabel ='Frequency' , xlabel ='Year' , title ='Word Usage Ove
r Time' )
```

In [44]:
```
word_time_series ( words =[ 'love' , 'like' ] ,
                   colors = [ 'blue' , 'green' ])
```

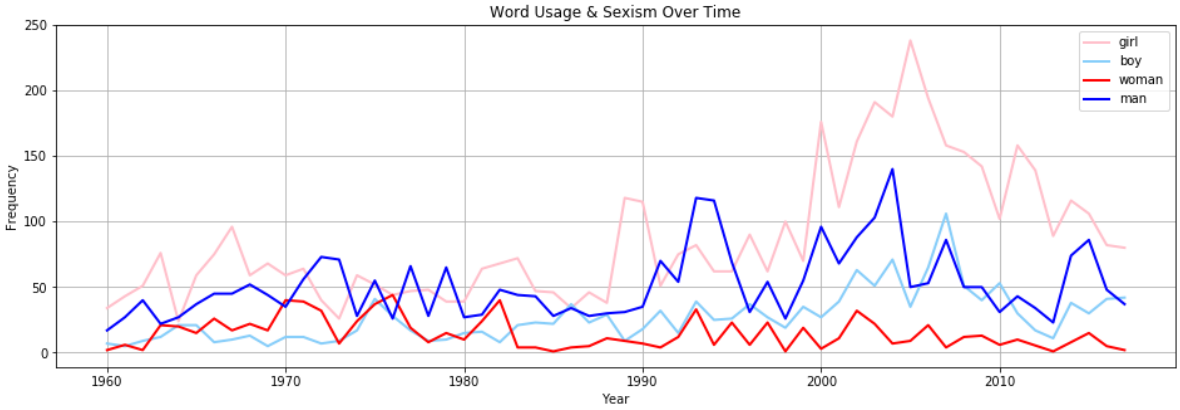


In [60]:
```
word_time_series ( words =[ 'yes' , 'dont' ] ,
                   colors = [ 'blue' , 'red' ])
```



In [67]:

```
word_time_series        ( words  =[ 'girl'       , 'boy'    , 'woman'   , 'man'   ] ,
                                   colors       =  [ 'pink'     , 'lightskyblue'          , 'red'     , 'blue'     ])
```



Annex 5: How to predict if a song will become a hit?

This notebook is the predictive analyis of my master thesis project. For more information about how I built the dataset, please refer to part III.1. To get more details about the mathematical models and the performance criteria, check part III.2.

Author

- *JACQUEMIN-LORRIAUX Marine*

Data Description

- I have built my own dataset using Spotify API
- The dataset is composed of the top streamed trakcs of 2017 and 2018, merged with songs from the same artists, same period, but which are not considered as hit songs.
- Dataset is available on github (Insert link)

Columns Description

- **Danceability** describes how suitable a track is for dancing based on a combination of musical elements. A value of 1.0 indicates high danceability.
- **Duration** indicates the length of the song.
- **Energy** is different from danceability in that it is a perceptual measure of intensity and activity.
- Energetic tracks feel dense, fast, loud, and noisy.
- **Explicit** is a categorical feature, with 1 indicating that a song contains explicit lyrics.
- **Instrumentalness** describes the extent to which the singer is not the primary performer of the song. **Liveness** detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the song was performed and recorded live.
- **Loudness** of a recording is measured in decibels (dB). Loudness values are averaged across the entire song and typically range between -60dB and 0dB.
- **Speechiness** detects the presence of spoken words in a recording. The more exclusively speechlike the recording (ex. talk shows, audio books, etc.), the closer the speechiness is to 1.0.
- **Tempo** is the speed, or the number of beats per minute (BPM).

Time signature is a notational convention to specify how many beats are in each bar of music. **Valence** describes the musical positiveness conveyed by the song. In other words, songs with high valence sound more happy/positive/cheerful while songs with low valance sound more negative/depressed/angry.

Okay then, let's start the analysis

Calling the required libraries…

```
options(warn=-1)
Library(dplyr)
```

```
library(class)
library(rpart)
library(rpart.plot)
library(randomForest)
library(mlbench)
library(glmnet)
library(tidyr)
library(ggplot2)
library(ggcorrplot)
library(caret)
```

```
library(forcats)
library(plotROC)
library(purr )
```

Loading and Checking the data

```
data_input <- read.csv("data_predict.csv",header = TRUE)
#head(data_input)
```

```
## [1] "Number of rows :  306  |  Number of Columns :  18"
```

Let's do some basic checks for consistency and null values

```
#str(data_input)
data_input = select (data_input,-c(X))
summary(data_input)
```

```
##                name                           album
artist   ##  Closer          :  2   ÷ (Deluxe)             :  4
XXXTENTACION :  8
##  I Like Me Better:  2   beerbongs & bentleys   :  4   Drake        :
7
##  Nevermind       :  2   Red Pill Blues (Deluxe):  4   Ed Sheeran   :
7
##  New Rules       :  2   Scorpion               :  4   Post Malone  :
6
##  X               :  2   ?                      :  3   Calvin Harris:
5
##  1-800-273-8255  :  1   17                     :  3   Clean Bandit :
5
##  (Other)         :295   (Other)                :284   (Other)
:268
##      release_date  duration_ms      explicit    acousticness
##  2017-08-25:  7   Min.   : 67579   False:208   Min.   :0.0000778
##  2017-06-30:  6   1st Qu.:187226   True : 98   1st Qu.:0.0328250
##  2018-04-27:  6   Median :208800               Median :0.1075000
##  2018-05-18:  5   Mean   :210376               Mean   :0.1940077
##  2018-06-29:  5   3rd Qu.:228163               3rd Qu.:0.2707500
##  2018-08-17:  5   Max.   :470493               Max.   :0.9810000
##  (Other)   :272
##   danceability        energy       instrumentalness       liveness
##  Min.   :0.2580   Min.   :0.0549   Min.   :0.0000000   Min.
:0.02150
##  1st Qu.:0.6012   1st Qu.:0.5590   1st Qu.:0.0000000   1st
Qu.:0.09773
##  Median :0.6960   Median :0.6815   Median :0.0000000   Median
:0.12400
##  Mean   :0.6821   Mean   :0.6649   Mean   :0.0343279   Mean
:0.16688
##  3rd Qu.:0.7708   3rd Qu.:0.7913   3rd Qu.:0.0000574   3rd
Qu.:0.18650
##  Max.   :0.9640   Max.   :0.9690   Max.   :0.9350000   Max.
:0.84200
##
##     loudness        speechiness         tempo           valence
##  Min.   :-14.948   Min.   :0.0232   Min.   : 64.93   Min.   :0.0368
##  1st Qu.: -7.074   1st Qu.:0.0432   1st Qu.: 96.61   1st Qu.:0.3207
##  Median : -5.662   Median :0.0621   Median :115.32   Median :0.4695
```


-
-
-

```
##  Mean   : -6.004   Mean   :0.1035   Mean   :118.68   Mean   :0.4796
##  3rd Qu.: -4.564   3rd Qu.:0.1217   3rd Qu.:135.89   3rd Qu.:0.6298
##  Max.   : -1.366   Max.   :0.5300   Max.   :198.07   Max.   :0.9690
##
##  time_signature       Hit
##  Min.   :3.000   Min.   :0.0000
##  1st Qu.:4.000   1st Qu.:0.0000
##  Median :4.000   Median :1.0000
##  Mean   :3.993   Mean   :0.6078
##  3rd Qu.:4.000   3rd Qu.:1.0000
##  Max.   :5.000   Max.   :1.0000
##
```

Everything seems to be okay. There is no missing value.
I will convert the *'duration_ms'* from ms to second.
I will convert my target feature *'Hit'* as factor.

```
data_input = mutate(data_input,duration_min = duration_ms/60000)
data_input$Hit = as.factor(ifelse(data_input$Hit==1,"Yes","No"))
```

Okay, so we have done the initial checks and cleaning and we have the data to work with. Let's start by some basic description

```
hittable <- data_input %>% count(Hit) %>% mutate(perc = prop.table(n))
hitperc <- as.double(hittable[hittable$Hit == "1","perc"])
hittable
```

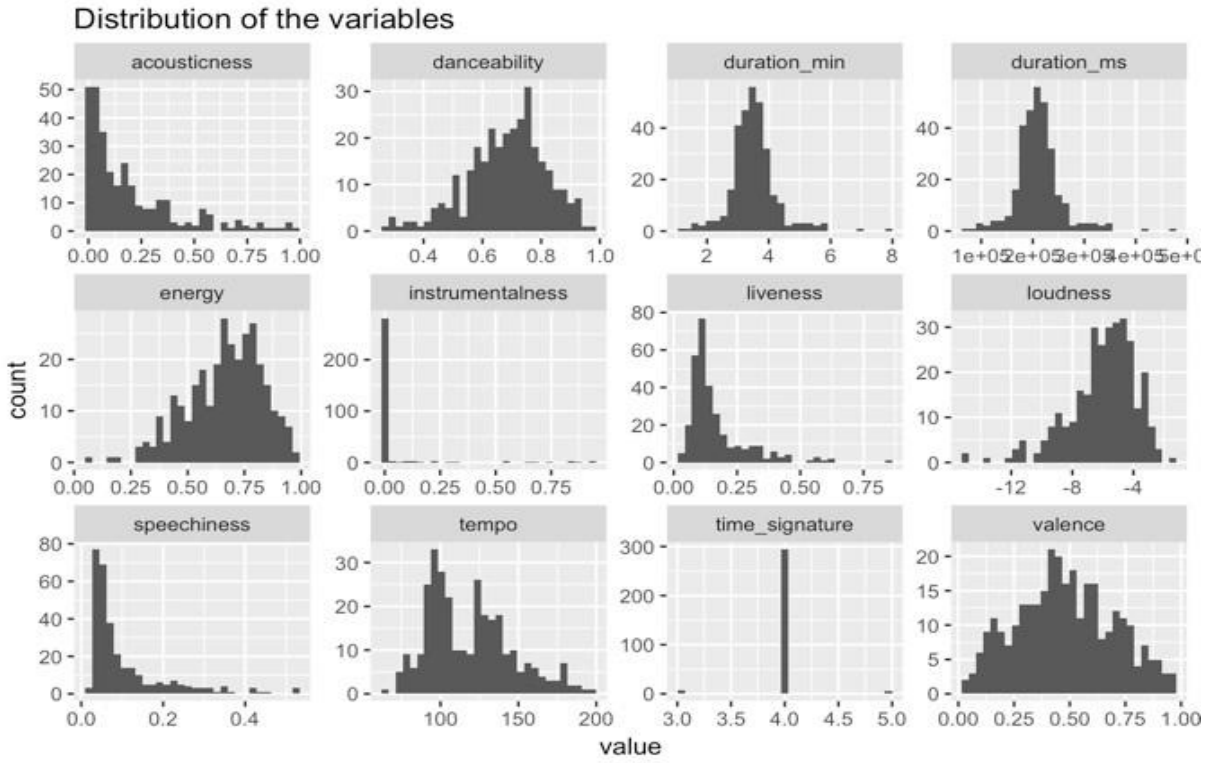
```
## # A tibble: 2 x 3
##      Hit     n      perc
##   <fctr> <int>     <dbl>
## 1     No   120 0.3921569
## 2    Yes   186 0.6078431
```

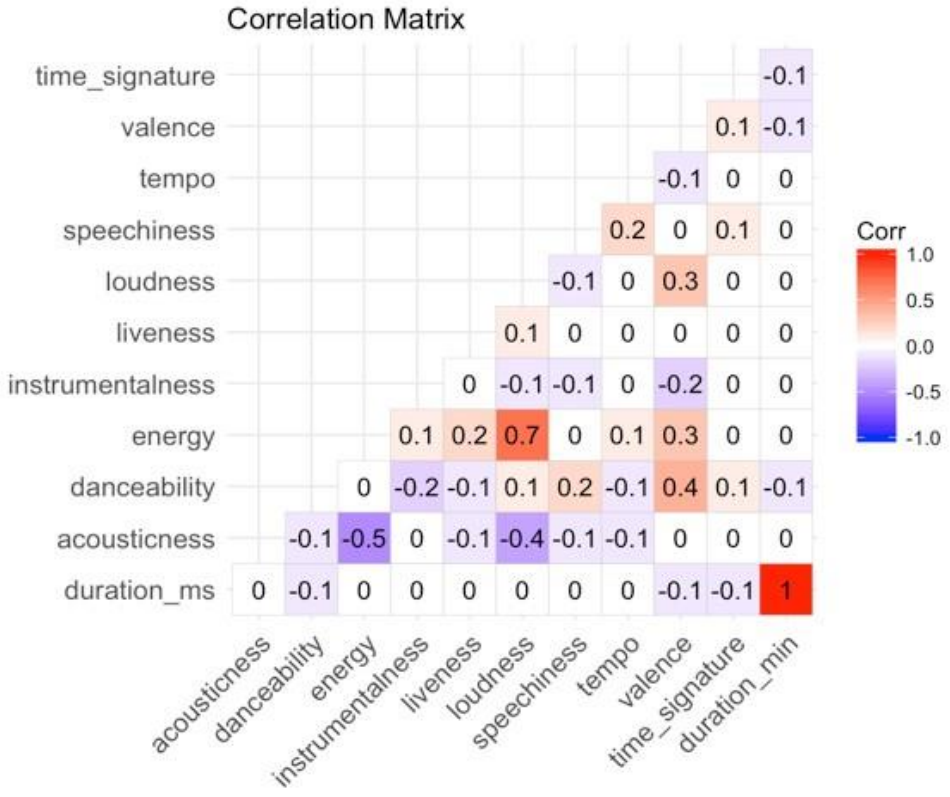
Of the 306 songs, we have **40% non hit songs**. The data is not heavily skewed, but slightly imbalanced. We need to remember this while building the models later.

Let's start exploring using visualization

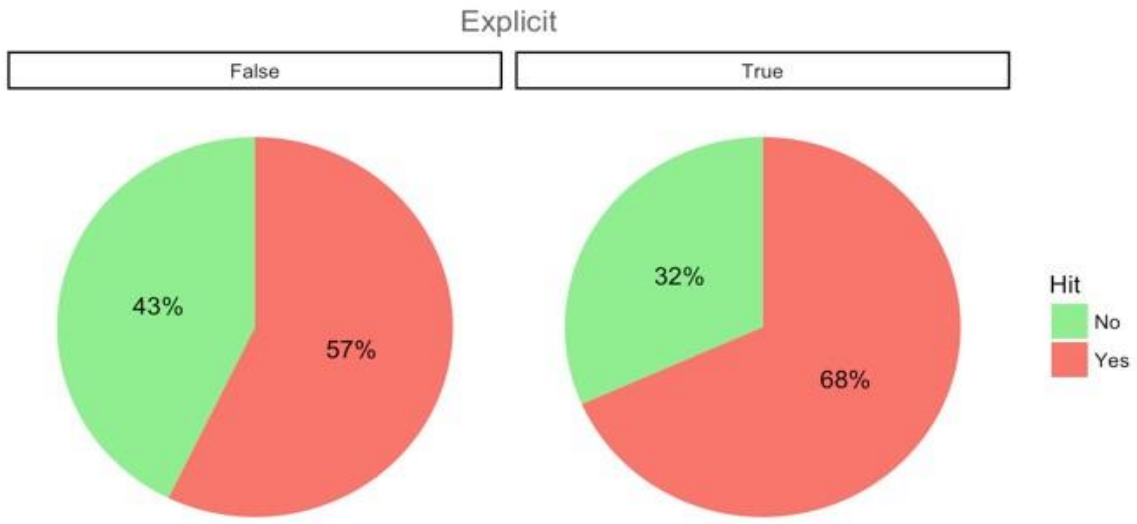
Please note that I am showing only the relevant graphs in this notebook. Also, the code for graphs are hidden for the simplicity of the notebook. Please refer to the .Rmd file for code

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```





Loudness and energy are positively correlated. Energy and acousticness are negatively correlated. (We draw the same conclusion when studying the most ever played songs.)



70% of the songs labeled as explicit are hit songs.

Let's try modeling different algorithms and see which performs best on this data

But first things first

- This is a classification problem
- We split train-test 75%-25%

Data preparation

```
data <- data_input %>% select(-c(name,album,artist,release_date,duration_ms)) #Re
ve the unnecessary columns
set.seed(111)
train.index <- sample(nrow(data), nrow(data)*0.75)
train <- data[train.index,]
test <- data[-train.index,]
train.X <- model.matrix(Hit~.,data=train)
test.X <- model.matrix(Hit~.,data=test)
```

1. **Logistic Regression**

I will fit a logistic regression on the data.train data set. The logistic model is defined by

$$\log\left(\frac{p(x)}{1-p(x)}\right) = \beta_0 + \beta_1 x_1 + \ldots + \beta_{12} x_{12}$$

where $p(x) = P(Y = 1 \mid X = x)$

Unknown parameters

$$\beta_1, \ldots, \beta_{12}$$

are estimated by maximum likelihood. We fit the model

Since I have a binary Hit variable, I start with a logistic regression model. I try to predict the hit probability by fitting all the right explanatory variables on Hit.

```
model.logistic <- glm(Hit~., data=train, family="binomial")
pred.logistic <- predict(model.logistic, newdata=test, type =
"response") summary(model.logistic)
```

```
## glm(formula = Hit ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9957  -1.0211   0.6025   0.8816   1.7763
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)       4.673156   4.401753   1.062 0.288391
## explicitTrue     -0.172027   0.381535  -0.451 0.652076
## acousticness     -0.726717   0.889824  -0.817
0.414101    ## danceability      2.633478   1.424918
1.848 0.064579 .
## energy           -4.148553   1.685324  -2.462
0.013833 *  ## instrumentalness -9.975751   5.818584
-1.714 0.086444 .
## liveness         -2.456308   1.294982  -1.897 0.057856 .
## loudness          0.474448   0.125012   3.795 0.000148 ***
## speechiness       1.520418   1.777756   0.855 0.392415
## tempo            -0.003589   0.005744  -0.625 0.532108
## valence           0.325024   0.883777   0.368 0.713047
## time_signature   -0.185900   0.919559  -0.202
0.839790    ## duration_min      0.348900   0.233180
1.496 0.134584
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.'
0.1 ' ' 1 ##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 306.01  on 228  degrees of freedom
## Residual deviance: 255.72  on 216  degrees of freedom
## AIC: 281.72
##
## Number of Fisher Scoring iterations: 7
```

The p-value for each term tests the null hypothesis that the coefficient is equal to zero (no effect). Loudness and energy have a small p-value (<0.05), we reject the null hypothesis, those variables are statistically significant. All the other variables are not statistically significant to explain Hit.

2. **Stepwise Logistic Regression**

Since there are a lot of variables, I want to check which ones are the most significant ones.

I propose to make a variable selection procedure with a backward selection approach using BIC criterion

```
#select model which optimizes BIC criteria model.backward <-
step(model.logistic,direction="backward",k=log(nrow(train)),trace
=0) pred.backward <- predict(model.backward, newdata=test,
type="response") summary(model.backward)
```

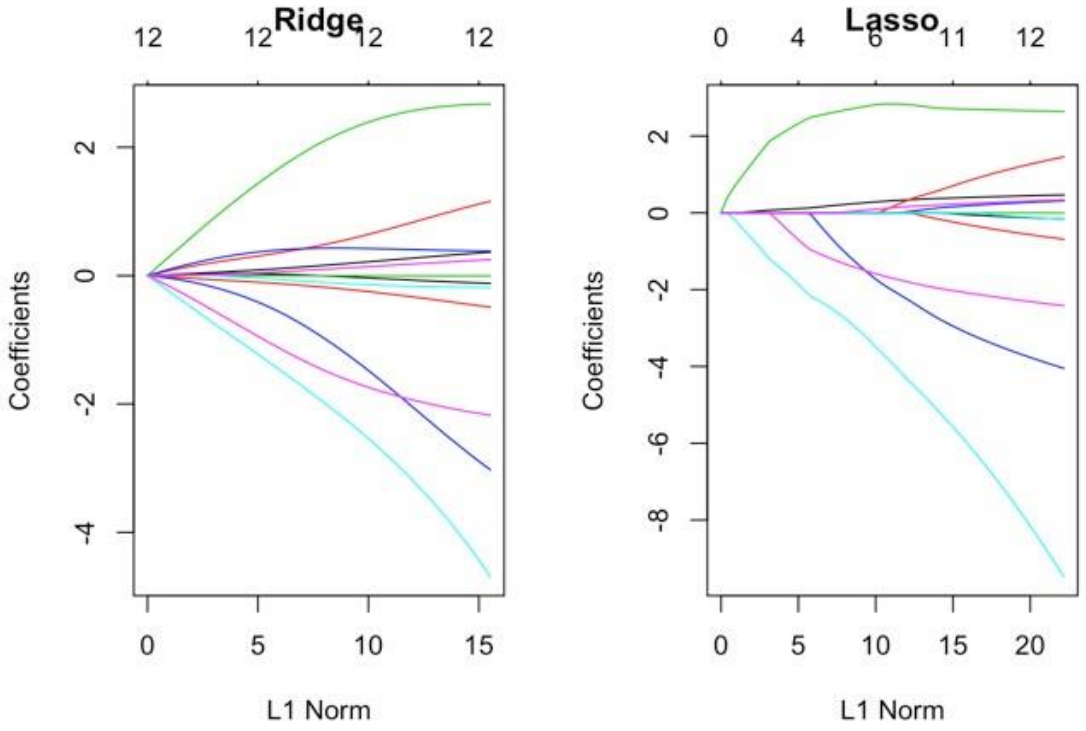
```
##
## Call:
## glm(formula = Hit ~ danceability + energy + instrumentalness +
##     loudness, family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1478  -1.0795   0.6479   0.9028   1.8016
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)        3.3358     1.8027   1.850 0.064251 .
## danceability       3.1531     1.1902   2.649 0.008067 **
## energy            -3.4621     1.3941  -2.483 0.013012 *
## instrumentalness  -8.4005     5.4726  -1.535 0.124784
## loudness           0.4293     0.1170   3.670 0.000242 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 306.01  on 228  degrees of freedom
## Residual deviance: 262.41  on 224  degrees of freedom
## AIC: 272.41
##
## Number of Fisher Scoring iterations: 7
```

Here the model selected automatically the best subset with all statistically significant variables. It seems that loudness, danceability, energy are the most statistically significant variables.

3. **Penalised Regression**

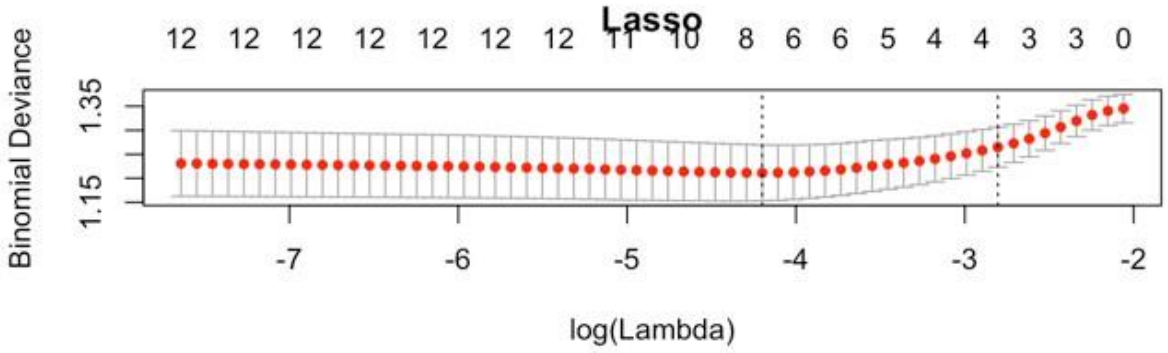
I will now try to get rid of unnecessary variables by imposing a constrain on the size of the coefficient. Lasso allows to drop some variables while Ridge shrinks the coefficient. Both methods will help to reduce the variance.

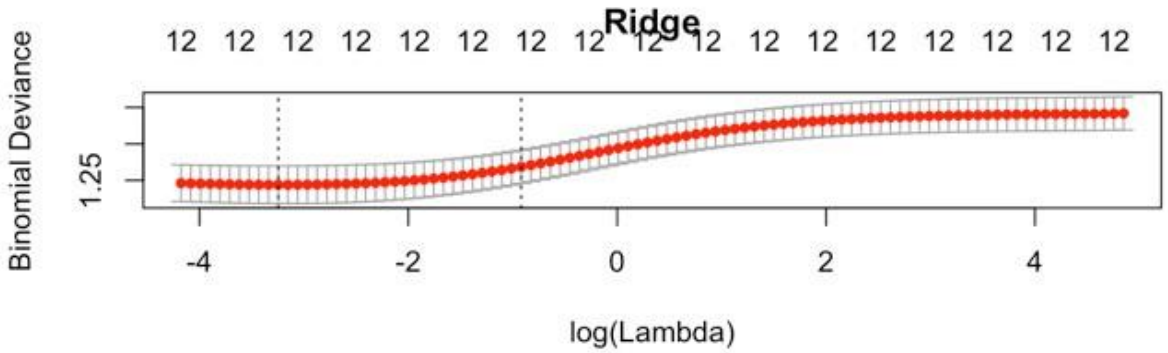
```
#Draw the coefficient paths for ridge and lasso.
model.ridge <-
glmnet(train.X,train$Hit,family="binomial",alpha=0)
model.lasso <-
glmnet(train.X,train$Hit,family="binomial",alpha=1)
par(mfrow=c(1,2)) plot(model.ridge,main="Ridge")
plot(model.lasso,main="Lasso")
```



So we see that the coefficients shrink for RIDGE and some of them become 0 for LASSO. Let's get the best shrinkage parameter for both.

```
#Select the shrinkage parameter
par(mfrow=c(2,1))

model.lassoCV <- cv.glmnet(train.X,train$Hit,family="binomial",alpha=1)
plot(model.lassoCV,main="Lasso")

model.ridgeCV <- cv.glmnet(train.X,train$Hit,family="binomial",alpha=0)
plot(model.ridgeCV,main="Ridge")
```






```
## [1] "Selected Lamba are 0.0150049391 for Lasso and 0.0389381802 for Ridge"
```

```
pred.lasso <- predict(model.lassoCV, newx=test.X, s="lambda.min", type="response")
pred.ridge <- predict(model.ridgeCV, newx=test.X, s="lambda.min", type="response")
```

4. KNN Classification

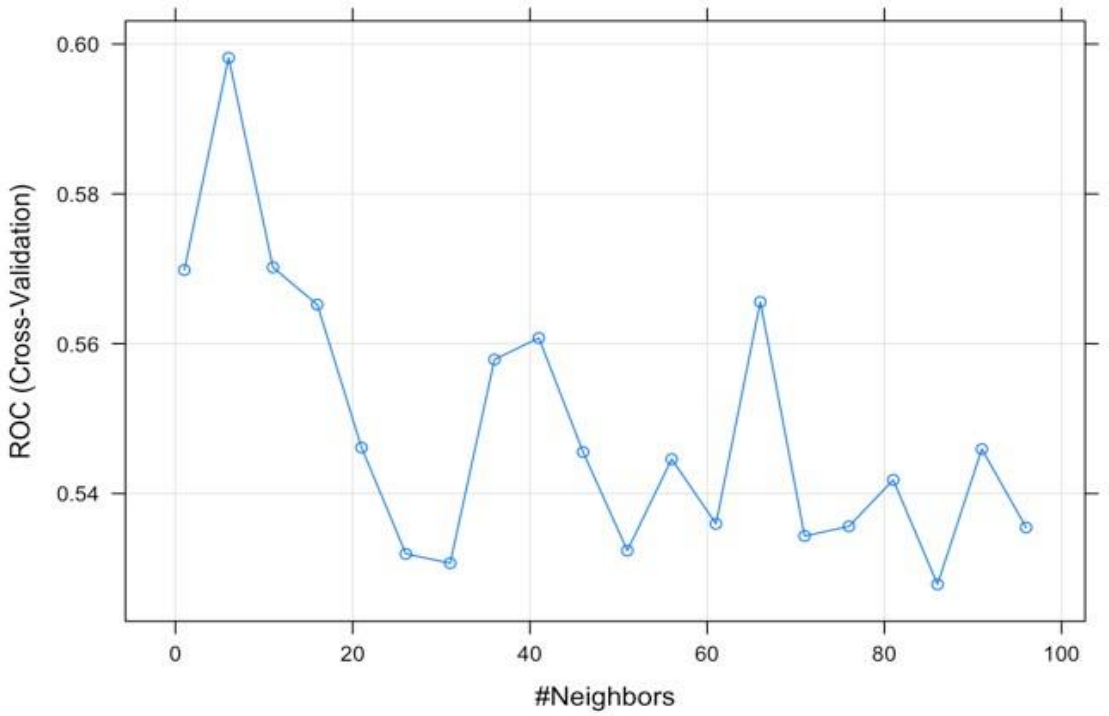
With knn I try to find a classification method which will predict whether a song is a hit or not by measuring the similarity with hit/non hit songs.

I will use ERM (empirical risk minimization) to find the optimal 'k' by doing a grid search with k-fold cross validation. Since the data is imbalanced, accuracy is not the most reliable metric. Hence, I will choose 'k' based on ROC

```
ctrl1 <- trainControl(method="cv",number=10, classProbs = TRUE, summary = twoClassS
ummary)
grid.k <- data.frame(k=seq(1,100,by=5))
select.k <- train(Hit~.,data=train,method="knn",trControl=ctrl1,tuneGrid=grid.k,
                  metric="ROC")
select.k
```

```
## k-Nearest Neighbors
##
## 229 samples
##  12 predictor
##   2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 206, 206, 206, 206, 206, 206, ...
## Resampling results across tuning parameters:
##
##   k   ROC        Sens        Spec
##    1  0.5698413  0.46111111  0.6785714
##    6  0.5981647  0.37083333  0.7714286
##   11  0.5701885  0.22361111  0.8214286
##   16  0.5652282  0.19166667  0.8714286
##   21  0.5461310  0.09166667  0.9142857
##   26  0.5319444  0.06944444  0.9571429
##   31  0.5307044  0.02222222  0.9642857
##   36  0.5578869  0.00000000  0.9857143
##   41  0.5607639  0.02222222  0.9857143
##   46  0.5455357  0.01111111  0.9928571
##   51  0.5323909  0.00000000  1.0000000
##   56  0.5445933  0.00000000  1.0000000
##   61  0.5359623  0.00000000  1.0000000
##   66  0.5655754  0.00000000  1.0000000
##   71  0.5343254  0.00000000  1.0000000
##   76  0.5356151  0.00000000  1.0000000
##   81  0.5418155  0.00000000  1.0000000
##   86  0.5278770  0.00000000  1.0000000
##   91  0.5459325  0.00000000  1.0000000
##   96  0.5354663  0.00000000  1.0000000
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was k = 6.
```

```
plot(select.k)
```




```
## [1] "Optimal k is 6  because it has the highest accuracy. If we increase the k,
accuracy will decrease because of high bias"
```

Now that we have the optimal k, let's use it for knn-classification

```
pred.knn <- knn(train.X, test.X, cl=train$Hit, k = select.k$bestTune,
prob=TRUE)
```

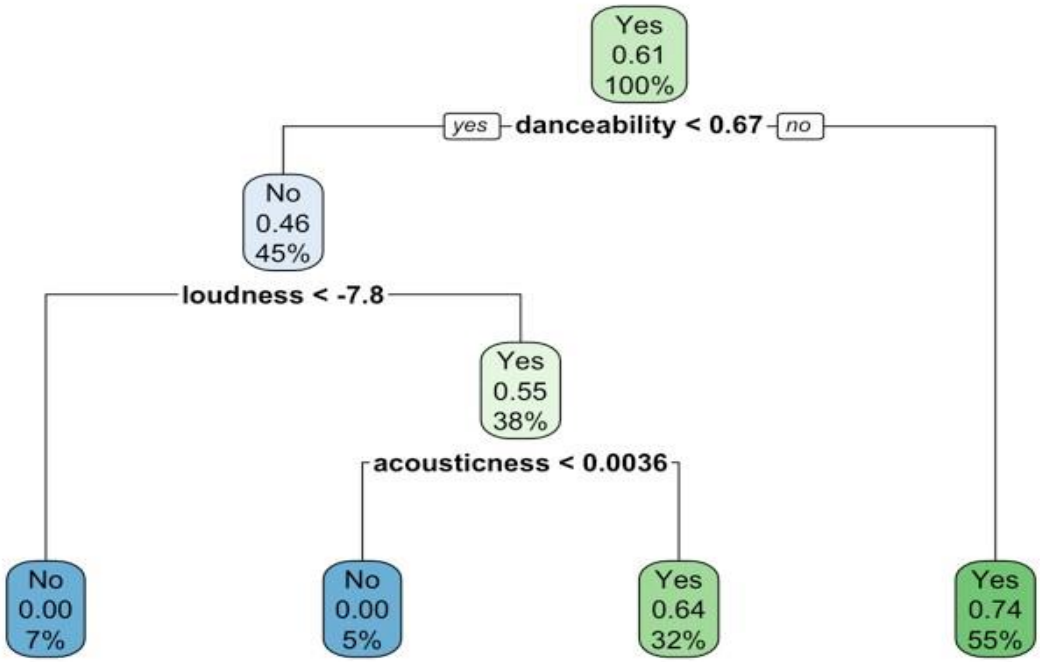
5. **Trees**

Trees allow to build many rules based on the split of the variables. The trees will help me in identifying the important variables to find the best split that minimise impurity

```
model.tree <- rpart(Hit~., data=train, cp=0.00001)
printcp(model.tree)
```

```
##
## Classification tree:
## rpart(formula = Hit ~ ., data = train, cp = 1e-05)
##
## Variables actually used in tree construction:
## [1] acousticness     danceability     duration_min     instrumentalness
## [5] loudness         speechiness      valence
##
## Root node error: 89/229 = 0.38865
##
## n= 229
##
##         CP nsplit rel error xerror     xstd
## 1 0.108614      0   1.00000 1.0000 0.082880
## 2 0.056180      3   0.67416 0.8764 0.080580
## 3 0.033708      4   0.61798 1.0225 0.083206
## 4 0.026217      5   0.58427 1.0449 0.083503
## 5 0.022472      8   0.50562 1.0225 0.083206
## 6 0.011236      9   0.48315 1.0112 0.083046
## 7 0.000010     11   0.46067 1.0225 0.083206
```

```
prune.tree <- prune(model.tree, cp = model.tree$cptable[which.min(model.tree$cptabl
e[,"xerror"]),"CP"])
pred.prune <- predict(prune.tree, newdata=test, type="class")

rpart.plot(prune.tree)
```



We can see that all important variables that we saw in the visualisation part are all used to split and construct the tree.

6. **RandomForest**

With Random Forest, we try to build many trees to predict the label (by taking majority). This method helps in reducing the overfit and is very robust since it uses bagging approach.

```
model.randomf <- randomForest(Hit~., data=train)
model.randomf
```

```
##
## Call:
##  randomForest(formula = Hit ~ ., data = train)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 31.88%
## Confusion matrix:
##     No Yes class.error
## No  41  48   0.5393258
## Yes 25 115   0.1785714
```

Let's try to check the optimal *'mtry'* value.

```
set.seed(1111)
ctrl2 <- trainControl(method="oob")
select.mtry <- train(Hit~.,data=train,method="rf",trControl=ctrl2,
tuneGrid=data.frame(mtry=seq(1,6,by=1)))
model.randomf <- randomForest(Hit~., data=train, mtry=as.double(select.mtry$bestTun
e))
model.randomf
```

```
##
## Call:
##  randomForest(formula = Hit ~ ., data = train, mtry = as.double(select.mtry$best
Tune))
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 31%
## Confusion matrix:
##     No Yes class.error
## No  29  60  0.67415730
## Yes 11 129  0.07857143
```

RandomForest is building 500 trees and uses 1 random variable at each split. The estimate of error rate is
31%

```
pred.randomf <- predict(model.randomf, newdata=test, type="prob")[,"Yes"]
```

Compare different models

```
class.logistic <- pred.logistic %>% round() %>% as.factor() %>%
fct_recode(No="0",Y es="1") class.backward <- pred.backward %>% round()
%>% as.factor() %>% fct_recode(No="0",Y es="1") class.knn <- pred.knn
class.lasso <- pred.lasso %>% round() %>% as.factor() %>%
fct_recode(No="0",Yes="1"
) class.ridge <- pred.ridge %>% round() %>% as.factor() %>%
fct_recode(No="0",Yes="1"
)
class.prune <- pred.prune %>% round() %>% as.factor() %>%
fct_recode(No="0",Yes="1"
```

```
) class.randomF <- pred.randomf %>% round() %>% as.factor() %>%
fct_recode(No="0",Yes ="1")

#contains final class prediction class_data <-
data.frame(logistic=class.logistic,backward=class.backward,knn=class.
knn,lasso=class.lasso, ridge=class.ridge, tree=class.prune,
randomForest=class.rand omF,Y=test$Hit)

#contains probabilities pred_data <-
data.frame(logistic=pred.logistic,backward=pred.backward, knn=attribut
es(pred.knn)$prob,lasso=as.vector(pred.lasso),
ridge=as.vector(pred.ridge), tree=pr ed.prune,
randomForest=pred.randomf,Y=test$Hit)
```

Let's calculate the accuracy

```
class_data1 <- class_data %>% gather(key="Method",value="class",-Y)
class_data2 <- class_data1 %>% group_by(Method) %>% summarize(accuracy=mean(class==
Y)) %>% arrange(desc(accuracy))
class_data2
```

```
## # A tibble: 7 x 2
##         Method  accuracy
##          <chr>     <dbl>
## 1 randomForest 0.6623377
## 2        ridge 0.6623377
## 3     backward 0.6493506
## 4         tree 0.6493506
## 5        lasso 0.6363636
## 6     logistic 0.6233766
## 7          knn 0.5844156
```

Because the data is not completely balanced, the accuracy measure to evaluate my results is not really suited.

Let's compare the models with Specificity

```
spec_data1 <- class_data1 %>% group_by(Method) %>% summarize(specificity=specificit
y(table(class, Y))) %>% arrange(desc(specificity))
spec_data1
```

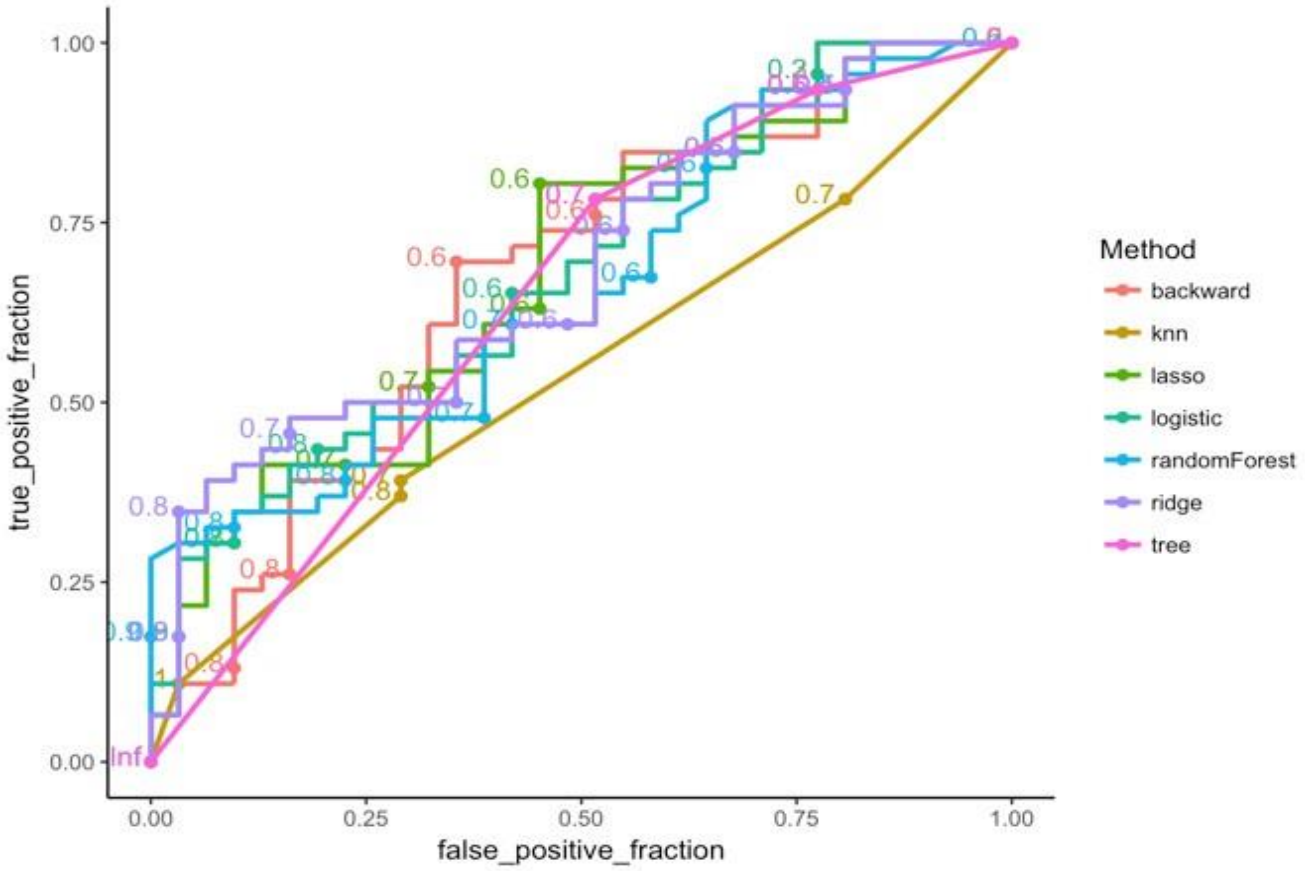
```
## # A tibble: 7 x 2
##         Method specificity
##          <chr>       <dbl>
## 1 randomForest   0.9347826
## 2         tree   0.9347826
## 3        ridge   0.8913043
## 4        lasso   0.8695652
## 5     backward   0.8478261
## 6     logistic   0.7826087
## 7          knn   0.7173913
```

Let's compare the models with ROC and AUC methods

```
pred_data1 <- pred_data %>% gather(key="Method",value="score",-Y)
pred_data2 <- pred_data1 %>% group_by(Method) %>% summarize(AUC=pROC::auc(Y,score))
%>% arrange(desc(AUC))
pred_data2
```

```
## # A tibble: 7 x 2
##         Method       AUC
##          <chr>     <dbl>
## 1        ridge 0.6858345
## 2        lasso 0.6781206
## 3     logistic 0.6753156
## 4     backward 0.6739130
## 5 randomForest 0.6591865
## 6         tree 0.6420056
## 7          knn 0.5389201
```

```
ggplot(pred_data1)+aes(d=Y,m=score,color=Method)+geom_roc()+theme_classic()
```



Final comparison table

```
full_join(pred_data2, class_data2,by="Method") %>% left_join(.,
spec_data1, by="Met hod") %>% arrange(desc(AUC))
```

```
## # A tibble: 7 x 4
##         Method       AUC  accuracy specificity
##          <chr>     <dbl>     <dbl>       <dbl>
## 1        ridge 0.6858345 0.6623377   0.8913043
## 2        lasso 0.6781206 0.6363636   0.8695652
## 3     logistic 0.6753156 0.6233766   0.7826087
## 4     backward 0.6739130 0.6493506   0.8478261
## 5 randomForest 0.6591865 0.6623377   0.9347826
## 6         tree 0.6420056 0.6493506   0.9347826
## 7          knn 0.5389201 0.5844156   0.7173913
```

KNN obtained the worst results. Ridge and Lasso have the best AUC, but trees and random forests have better specificity.