**SUMMARY AND OBJECTIVE**

With this project my aim is to improve my data analysis and data science skills and to create a public repository that I can use to showcase my capabilities.

For this first project I’ve chosen a topic that I’m not familiar with and that I hope it will give me the opportunity to try different data analysis tools that I haven’t been able to try on my date to date job yet. As a little bit of background, I am a manufacturing engineer with extensive experience in the aerospace sector, specifically in manufacturing.

Coming back to the project, I will split this week into different sections.

**DATA COLLECTION**

I have segregated the data collection part into a data\_creation module for tidiness purposes.

I check whether the data has already been collected (which would be stored in parquet file on the current working directory) and if not I collect the data from the following webpage using pandas:

<http://www.umdmusic.com/default.asp?Lang=English&Chart=D&ChDay=1&ChMonth=2&ChYear=1999&ChBand=&ChSong>=

I decided to collect data using parquet files from 1970s to 2020s to be able to see if there is any change in the behaviour of the data across time. I chose to scrape the data 4 times for every month in that year range to create enough granularity of the data. Because the charts are on a weekly basis, I need to collect the data weekly. I also want to create a big enough dataset to apply descriptive and machine learning analysis.

**DATA PROCESSING**

My initial objective is to find out the characteristics that differentiate the number one songs from the rest of the songs. For that I calculate the following features for each song:

* Time in charts
* Quantity of positions climbed from bottom to top
* Quantity of positions climbed from start to top
* Starting position

I work using a summary table to which I can pin the results of the features to each song.

I calculate the next fields to get the previous features:

* The maximum date in the charts
* The bottom position of each of the songs in the charts
* The top position of each song in charts

Once the attributes are calculated, I clean the data through the following steps:

1. I select the songs where the OverallPeakPos (which came with the original data) and the calculated peak position match. To exclude song that peaked outside of the 1970 to 2010 time range.
2. I select the songs where the Entry in Charts or LastDate in charts reflect that the song’s run in the charts didn´t start before or didn´t potentially finish after the time analysis range.
3. I select the songs where the original length of days in charts is at least equal or bigger than the calculated time in charts. This is because if a song has had multiple re-entries across the years in the charts, the calculated time in charts can through silly numbers, where a song originally entered the charts in 1973 and the maximum date of the last time it entered the charts is 2016.

I realized that there is a mismatch between the Overall Total Weeks (the time that the song has been in charts according to the original data) and the calculated time in charts (by subtracting the last parsing date of that song with the entry date of the song).

For example, Mariah Carey’s song “All I want for Christmas is you” entered the charts in 2018-12-01 and exited the charts in 2019-01-01, spanning a total of 31 days. However, the original column specifies a total of 19 weeks in charts, which I think includes for future appearances in following Christmas periods and would not be suitable for the current analysis.

**GENERIC DATA ANALYSIS PER DECADE**

We want to compare the next features for n1 songs vs the rest for each decade:

* Time in charts
* Time to reach top position
* Quantity of positions climbed from bottom to top
* Quantity of positions climbed from start to top
* Starting position

We want to be able to predict if a song has a chance of becoming a hit when entering the charts by its minimum starting position. Once the song has entered the charts, we want to know if it still has a chance of becoming a hit:

* Current position > Maximum bottom position
* Current time in charts < Maximum time to climb to top position

And once a song has climbed to number one we want to know how much more time its got left in the charts:

* Minimum time in charts for n1 songs

We create a dataframe with the following columns to summarize per decade the features outlined above:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Qty songs | Decade | Avg Time in charts | Max time in charts | Min time in charts | Avg Qty of pos climbed total | Avg Qty of pos climbed from start | Avg Starting pos | Song group |

1 graph for positions + 1 graph for time

Bar line graph for positions: show per decade n1 vs rest of songs average of position climbed in total, from start and starting position.

**DISTRIBUTION ANALYSIS PER DECADE**

We want to do an equivalence analysis of the data for each decade, to calculate if the distributions for the characteristics to be used in a machine learning model are equivalent or whether the model should be done for a particular decade.

* Time in charts
* Time to reach top position
* Quantity of positions climbed from bottom to top
* Quantity of positions climbed from start to top
* Starting position

MACHINE LEARNING

What are other attributes that can determine if a song will be a n1?

* Words in lyrics
* Music genre
* ~~Artists: accumulated popularity when the song entered the charts~~
* ~~Technical proficiency of the musicians: awards~~
* ~~Technical proficiency of the musicians: formal training~~
* Topics of music
* Song’s composition: Song’s sheet music
* Song’s recording studio
* Song’s producers

Data sources for a song’s production values:

* Music industry websites: AllMusic, Billboard, Rolling Stone, Pitchfork, NME, The fader, Stereogum, Mixmag
* Music streaming and download services: Spotipy, Apple music API, AWS API, Google Play
* Online databases of music information, such as AllMusic, Discogs, MusicBrainz
* Music industry publications and magazines, such as Rolling Stone and Billboard

Potential predictive analytics to try:

* Random Forest
* XGBoost
* K-means clustering