Master's Programme in Data Science

Spotify hits audio features

Jenniina Laaksonen, Olli Jokinen, Daniel Holmberg

October 20, 2019

University of Helsinki
Faculty of Science

P. O. Box 68 (Pietari Kalmin katu 5)
00014 University of Helsinki

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Utbildningsprogram — Degree programme
Faculty of Science	Master's Programme in Data Science

Tekijä — Författare — Author

Jenniina Laaksonen, Olli Jokinen, Daniel Holmberg

Työn nimi — Arbetets titel — Title

Spotify hits audio features

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidantal — Number of pages
Report	October 20, 2019	8

Tiivistelmä — Referat — Abstract

This report contains the technical details of the mini project How to make a hit? The mini project is part of the course Introduction to data science at the University of Helsinki. In this report we shortly explain the data retrieval, feature selection, forecasting methods and the results.

Avainsanat — Nyckelord — Keywords

Audio, Spotify, Hit, ARIMA, Forecast

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Contents

1. Introduction

Our group's miniproject started with a question how to make a hit song? We defined a hit song: a track which has reached to Spotify Top 200 list. Naturally, we needed a data source that provides the charts and another data source which provides different audio features for each track on the list. Those audio features enables to use different prediction models and visualisations in this projects. As a sources of data, We used spotifycharts.com in order to get weekly top 200 charts and Spotify Web API for audio features.

We wanted to know which audio features are statistically significant in predicting the position of songs and so we implemented a generalized linear model. After the selection process we started forecasting what these features are looking like in up until the beginning of 2021. With those predicted trends in mind song-producers can make decisions on what direction to take with their songs.

2. Data retrieval

2.1 Data sources

We used two different sources of data: spotifycharts.com [1] and the official Spotify API. Spotifycharts.com provided "track position", "track name", "track id" of each track at Weekly Spotify Top 200 list from 2017 to September 2019. Spotify API provided several audio features for each track such as: instrumentalness, tempo, key, duration, streams, danceability etc. which we pulled from Spotify API with "track id". We connected to Spotify API with a python package called "spotipy" [2]. All the audio features and their description can be found from Spotify's developer's site spotify. Those audio features were the key elements for all the predictions and visualisations later in the project.

2.2 Description of processed dataset

Our final data consisted of weekly lists from 2017 to 2019. Essential columns in the DataFrame were: a position (1-200), a track name, a week, streams and all the audio features. Some of values were continuous such as tempo, duration, streams and some were categorical such as key. This was a perfect starting point to start our predictions and visualizations.

Data was clean since all the data from the sources were well structured. After we had joined our two data sources together we were able to smoothly continue the project without handling any missing values.

2.3 Special requirements

The only special requirement for the data retrieval was that Spotify's Web API requires a Spotify Premium account. Otherwise everything all the other action can be redone without any particular requirements.

3. Selection of features

The data from Spotify was relatively clean. There were not any missing values and all the used values were numeric. However we found the dataset to have too many variables and for the purpose of this project we wanted to reduce the amount of variables. The main reason for this was to make our program easier and simpler for the end user. Since our end user (eg. a song writer or a manager in the music business) is likely to have no experience in the field of statistics or data science we wanted to show them only the relevant data. To do this we chose to use a generalized linear model.

In our project we used both R and Python. In the variable selection we used R. We read the weekly charts and handpicked the columns for our analysis. There were two ways to pick the most relevant variables, either using the position or the amount of streams. Both have their pros and cons but for this one we ended up using the positions after comparing the two approaches.

We shuffled and picked the training set and the test set. The test set was used to check the results. Anova and Chisquare tests were used to determine which variables had statistical significance. Based on the results we chose danceability, liveness, loudness and speechiness to be the most important features. Also the key and the duration of the song had significant effect. However those are not continuous variables and in the prediction we would have to treat them differently from these four features.

4. Forecasting

4.1 Method

We used Autoregressive Integrated Moving Average (ARIMA) for the timeseries forecasting. The method rely on specifying three parameters p, d and q well for it to be successful in forecasting. The first parameter, p, is the order of autoregressive (AR) terms which incorporate the effect of past values into our model. The second one, d, is the order of differencing to make our timeseries stationary (it correspond to the "integrated" part of the model). Lastly we have q which is the number of moving-average (MA) terms, this allow us to set the error of our model based on previously observed errors [3].

4.2 Implementation

We utilized a Python package [4] able to automate the process of finding correct parameters for the ARIMA model. It minimizes the value of the Akaike information criterion (AIC). This criterion estimates the relative amount of information lost by a given model, the less the better. Anyway, one parametrization is done in about a tenth of a second, and it tries approximately 5–10 per audio feature being forecasted. The total fit time varied from 0.529 to 1.841 seconds for the different features. A blogpost on our implementation can be found on Towards Data Science[†].

[†] https://towardsdatascience.com/arima-forecasting-in-python-90d36c2246d3

5. Results

5.1 Hit songs' audio features forecast

Our aim is to forecast the important audio features determined in part 3 so that future song producers have a baseline of what kind of music might be successful. The predictions are cut of at the beginning of 2021 since the 95% confidence intervals has grown very large at that point.

We managed to get probable forecasts for our four audio features: loudness, speechiness, liveness and danceability. Loudness and danceability are predicted to drop a bit whereas speechiness and liveness are more likely to increase. Figure 5.1 contain these trends of weekly audio feature means. The confidence interval is indeed large so rough trends are the main takeaway from our predictions. Our work can be found on GitHub[†].

5.2 Lyrics generation

Apart from the audio-feature forecasting, a side-project emerged from playing around with the Spotify top 200 data. The songs' lyrics were downloaded using Genius API [5]. Generating new lyrics have been made easy by Python's textgenrnn [6]. Textgenrnn uses recurrent neural networks to train a model on a given text, 4 epochs were a suitable amount for the model to generate decent lyrics. An example of the page can be seen in figure 5.2. There is a short blogpost[‡] regarding its creation.

[†] https://github.com/hd4niel/Predict-Spotify-Top200

[‡] https://towardsdatascience.com/lyrics-generation-model-to-the-web-f5e03dc5d8b5

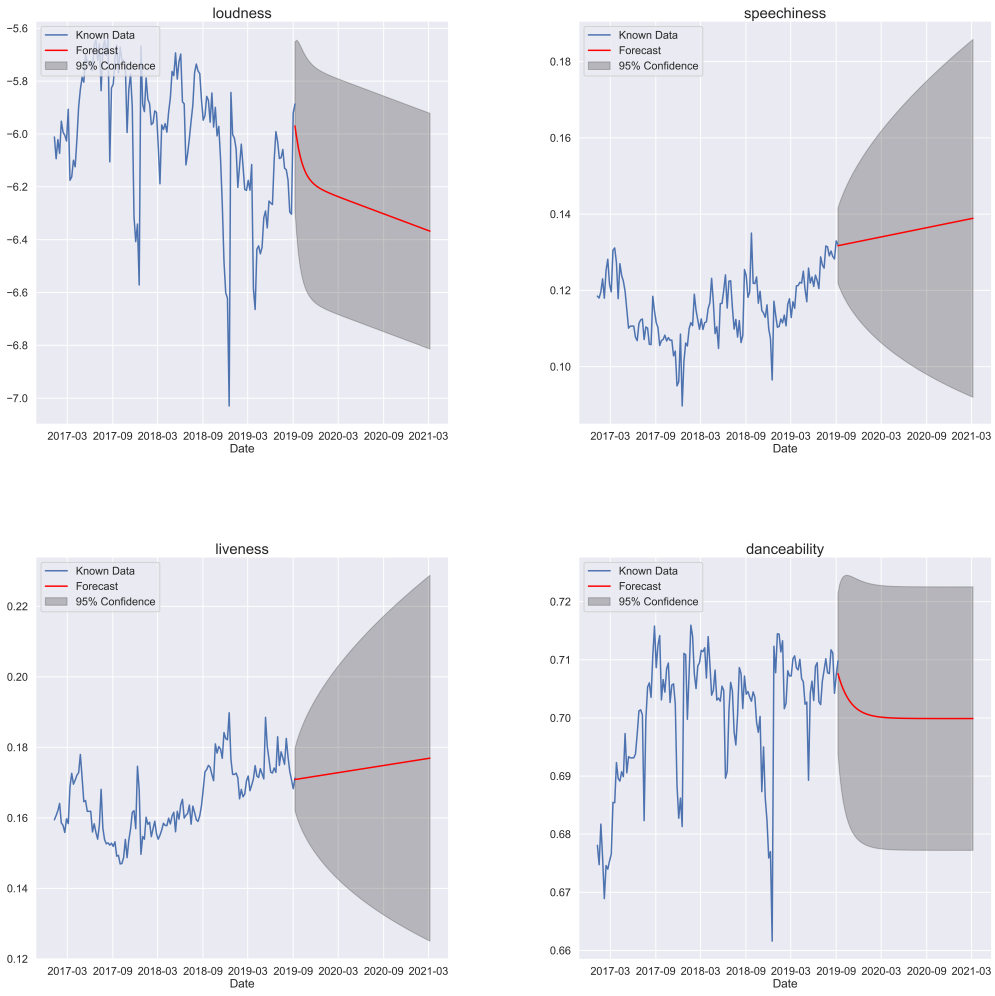

Figure 5.1: ARIMA forecast of audiofeatures based on Spotify top 200 data 2017-2019.

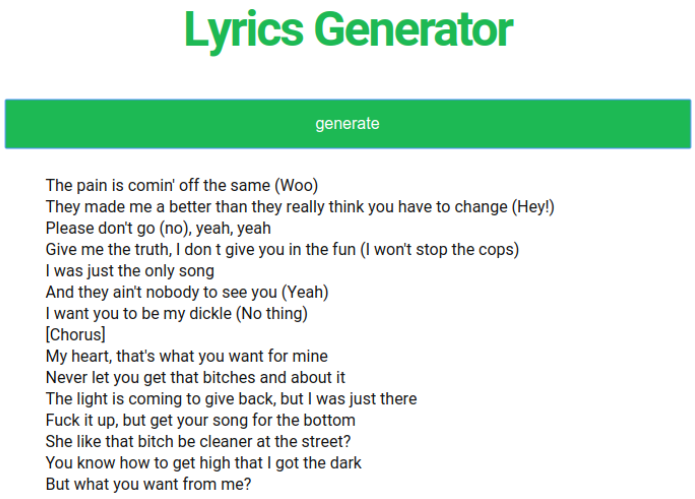

Figure 5.2: Lyrics generation website at damp-harbor-56160.herokuapp.com, it can take a few seconds for the free tier hosting to wake up.

References

[1] "Spotify charts." https://spotifycharts.com.

[2] "Spotipy - a python client for the spotify web api." https://github.com/plamere/spotipy.

[3] M. Mihsra, "Unboxing arima models," June 11 2018. https://towardsdatascience.com/unboxing-arima-models-1dc09d2746f8.

[4] "Automatic ARIMA." http://www.alkaline-ml.com/pmdarima/1.0.0/modules/generated/pmdarima.arima.auto_arima.html.

[5] "Genius developers." https://genius.com/developers.

[6] "Text-generating neural network." https://github.com/minimaxir/textgenrnn.