Competition - Music classification

September 12, 2022, Version 1.2

Andreas Lindholm (was Svensson),
Dept. of Information Technology
Uppsala University
andreas.svensson@it.uu.se

Bo Bernhardsson
Dept. of Automatic Control
Lund University
bob@control.lth.se

Abstract

This document contains the instructions for laboration 1 which is formed as a competition [1]. The problem is to classify a set of 200 songs, and predict whether Andreas Lindholm (former Svensson) would like them or not. To your help you have a training data set with 750 songs, which Andreas has classified as like (1) or dislike (0). You are expected to (i) try a number of classification methods and evaluate their performance on the problem, and (ii) make a decision which one to use and 'put in production' by uploading your predictions to a website, where your prediction will be evaluated and also compared to the performances of the other competitors. You should also document your project according to the instructions in Section 3.

0 Setup

Your handin should follow the structure described in Section 3 below and be handed in via Canvas before the deadline.

1 Problem

The problem is to tell which songs, in a dataset of 200 songs, Andreas Lindholm is going to like (see Figure 1). The data set consists not of the songs themselves, but of high-level features extracted using the web-API from Spotify[2]. These high-level features describe characteristics such as the acousticness, danceability, energy, instrumentalness, valence and tempo of each song.

To your help, you are provided a training data set with 750 songs, each of which Andreas has labeled with LIKE or DISLIKE. You are expected to try out classification methods described in the course, to come up with *one* algorithm that you think is suited for this problem and which you decide to put 'in production'.

1.1 Data sets

The data set to classify is available as `songs_to_classify.csv`, and the training data is available as `training_data.csv`. The format of the data files is described in `readme.txt`. For details on the extracted features, see Table 1. The data sets are available at the web page linked in Canvas.

[1] Thanks to Andreas who originally developed this assignment for the statistical machine learning course at Uppsala University, http://www.it.uu.se/edu/course/homepage/sml/, and has generously allowed us to reuse it. The contribution by the second author has been purely editorial.

[2] https://developer.spotify.com/web-api/get-audio-features/

Figure 1: Andreas Svensson listening to music.

Table 1: Details on the available features (from the Spotify API documentation)

Name	Type	Description
acousticness	float	A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.
danceability	float	Danceability describes how suitable a track is for dancing based on a combination 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.
duration_ms	int	The duration of the track in milliseconds.
energy	float	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.
instrumentalness	float	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.
key	int	The key the track is in. Integers map to pitches using standard Pitch Class notation. E.g. 0 = C, 1 = C major/D minor, 2 = D, and so on.
liveness	float	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.
loudness	float	The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing relative loudness of tracks. Loudness is the quality of a sound that is the primary psychological correlate of physical strength (amplitude). Values typical range between -60 and 0 db.
mode	string	Mode indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived.
speechiness	float	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.
tempo	float	The overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece and derives directly from the average beat duration.
time_signature	int	An estimated overall time signature of a track. The time signature (meter) is a notational convention to specify how many beats are in each bar (or measure).
valence	float	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).

1.2 Background

The problem of predicting user preferences is a hot research topic both in academia and industry: you have probably seen "you would perhaps also like ..." in online services. Within music, a big player has been the Echo Nest, founded in 2005 as a research spin-off from the MIT Media Lab and later acquired by Spotify. Their focus was methods for automated understanding of music, and in 2011 they released a popular benchmark dataset 'the million song dataset' (Bertin-Mahieux et al. 2011) which has become popular in the research community (see, for example, Fu et al. 2011; Oord, Dieleman, and Schrauwen 2013), and has similarities to this project. An overview of the scientific field of music recommendation is found in Kaminskas and Ricci (2012), and some pointers to recent advances can be found in Dieleman (2016) and Jacobson et al. (2016).

2 Technical tasks

2.1 Methods to explore

You should now have some familiarity with the following 'families' of classification methods:

(i) K-nearest neighbor
(ii) Logistic regression
(iii) Discriminant analysis: LDA, QDA
(iv) Tree-based methods: classification trees, random forests, bagging
(v) Boosting

(vi) Support Vector Machines

In the laboration task, you decide yourself which method(s) to explore. You should also decide which features to use and what preprocessing you want to do on the data, such as data normalisation, outlier detection and encoding of categorical variables. Example code for KNN is provided to you via the web page linked in Canvas. This code however only uses a subset of the features, and is not optimized enough, so you shouldn't hand in this code...

2.2 What to do with each method

For the method(s) you decide to explore, you should

(a) Implement the method. We suggest that you use python and sklearn, but you may write your own code or use other packages.

(b) Tune the method to perform well.

(c) Evaluate its performance using, e.g., cross validation. Note that each model needs to be evaluated using *only* the labeled data that is available in `training_data.csv`, i.e. for the purpose of model validation and selection you should *not* use the test data from `songs_to_classify.csv`. (We know it might be possible to cheat here...). Exactly how to carry out the evaluation is up to you to decide.

Once you have completed the aforementioned tasks, you should select which method you consider to be best to use 'in production'. Inspiration for a 'good motivation' can for instance be found in Section 4.5 in James et al. (2013).

When you have decided which method to try 'in production', run it on the test data (for which you do not have the true labels) and submit your results to the server to see how well it performs. You submit results as a string like `010011011`, where `0` means DISLIKE and `1` means LIKE. The web site also contains a leader board, where you can see how well you are doing in predicting Andreas' music taste, compared to the other students. The leaderboard is not available from the start, it opens up (at least) a week before the deadline.

*You only need to submit your final solution to the homepage **once**!* However, for the sake of a fun competition during the project, we allow each student to submit up to one solution per day (only the latest submission each day will be considered). The leader board will be updated every day.

3 Your individual handin

You should hand in a zip-file in Canvas before the deadline. Please name this file with your name , e.g. Bo-Bernhardsson.zip.

The handin should contain a short presentation (say 5-15 slides) and the code needed to reproduce your findings.

(1) A concise description of each of the considered methods.

(2) How the methods were applied to the data (any preprocessing of the inputs, which inputs were used, if the inputs were considered as qualitative or quantitative, how parameters were tuned, etc), including motivations of the choices made.

(3) Your evaluation of how well each method performs on the problem.

(4) Your conclusions.

After the deadline, some presentations will be chosen and you might be asked to present your work in class.

4 Your leaderboard handin

You will need a password, which will be sent to you when the scoreboard opens for submissions. Note the following limitations on the leaderboard. Only one submission per student and day is shown. If

several submissions are submitted in the same day, only your latest one is shown. Submissions made today will be shown from tomorrow. In a tie, the competitor with the least number of submissions wins. If yet a tie, the submission with the earliest high-score wins.

5 Tip for getting started

We have provided a sample code file `knn-example.py`, in which the data set is loaded and a k-NN classifier is trained using scikit-learn.

6 Deadlines & other important dates

See course Canvas page.

Important note

Please keep your solutions non-public, also after the course. The assignment is being reused in later courses.

Good luck!

References

Bertin-Mahieux, Thierry, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere (2011). "The million song dataset". In: *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR)*.

Dieleman, Sander (2016). "Keynote: Deep learning for audio-based music recommendation". In: *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*.

Fu, Zhouyu, Guojun Lu, Kai Ming Ting, and Dengsheng Zhang (2011). "A survey of audio-based music classification and annotation". In: *IEEE Transactions on Multimedia* 13.2.

Jacobson, Kurt, Vidhya Murali, Edward Newett, Brian Whitman, and Romain Yon (2016). "Music Personalization at Spotify". In: *Proceedings of the 10th ACM Conference on Recommender Systems*.

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani (2013). *An introduction to statistical learning. With applications in R*. Springer.

Kaminskas, Marius and Francesco Ricci (2012). "Contextual music information retrieval and recommendation: state of the art and challenges". In: *Computer Science Review* 6.

Oord, Aaron van den, Sander Dieleman, and Benjamin Schrauwen (2013). "Deep content-based music recommendation". In: *Advances in Neural Information Processing Systems 26 (NIPS)*.