Predicting the Popularity of a Song by its Audio Features

Project Report

presented by
Team 8
Marcel-René Wepper (1683678)
Sebastian Tetzlaff (1566485)
Alexander Sailer (1681978)
Lukas Degitz (1686428)
Elina Meyer (1683690)

submitted to the
Data and Web Science Group
Prof. Dr. Heico Paulheim
University of Mannheim

November 2019

Contents

Chapter 1

Application Area and Goals

1.1 Business Case

In modern society music is as present as ever. Economic interest have been increasing continuously over the last years. The global music revenues increased by 9.7% in comparison to 2017, totaling in US$ 19.1 billion in 2018 [IFPI, 2019, p.15]. Furthermore, digital music shares account for 58.9% of global revenue [IFPI, 2019, p. 16]. From a business perspective, popular songs are evidently more desirable than others by generating the most revenue [Sanchesz]. Thus the question arises whether certain hit songs have common characteristics that make them more popular than other songs, especially if this commonality can be measured, analyzed and predicted for new songs. This project deals with the problem if a song's popularity may be prognosed by its audio features. Applied to machine learning, this task can be mapped to the area of classification. In this context the class of interest consists of the most popular songs.

1.2 Goal

The goal of this project is to predict the popularity of a given song by its audio features. The foundation for this prediction is a data set consisting of song audio features, enriched by popularity scores from *Spotify*. Five different machine learning algorithms allow the estimation of song popularity based on this data set. The performances of these algorithms will be evaluated and compared critically to determine the reliability of the five method's popularity predictions and consequently the goal attainment level of this project.

Chapter 2

Data

2.1 Structure and Size of the Data Set

The data set considered in this project consists of multiple parts. The first part of the data was published by a research team from Cornell University and holds songs, their associated meta data and sound features. The sound features were extracted from the original freely available audio files using the python package *Librosa*. In total the data offers 518 numerical attributes for over 100,000 songs. [Benzi et al., 2016, p.1 ff.]

The attributes describing the songs represent different characteristics of spectral features of each pitch class. A pitch class represents all tones that are an octave apart, e.g. all Cs. In order to determine the attributes, the tracks were divided into 2048 equal time windows. For each time window and pitch class the spectral feature is determined. This results in 2048 measured values of a spectral feature per song and pitch class. Afterwards the information of the measured values is aggregated by different heuristic functions. Examples of these heuristic functions are the mean value, minimum value and kurtosis. [McFee et al., 2015]

An example of such a spectral feature is the chroma, representing the intensity of a pitch class. To detect the different chroma characteristics, the intensity of the pitch class C is determined for each time window. The observed values are then processed by different heuristics. The results include the following attributes: the mean of the chroma values and the maximum of the chroma values of pitch class 1 (C). This is done for every pitch class and spectral feature.

To predict the popularity of the songs, an estimation of popularity is required as an additional attribute. Spotify, a commonly used music streaming service, provides the popularity for each song available on their platform. This metadata can be requested by users via the Spotify REST *(representational state transfer)* API

(application programming interface) and contains a measure of popularity on a scale from 1 to 100, 100 being most popular [spo, 2019]. For each track in the first part of the data set the popularity score is requested from the Spotify interface and serves as the target label for classification. Therefore only tracks available on Spotify are further considered.

2.2 Preprocessing

To make sure that the before described data can be used in machine learning methods in a reasonable fashion, the following preprocessing steps are applied.

- **Enriching of song data with the popularity attributes from Spotify:**

  As described, the song data set was first extended by the attribute popularity. Not all songs of the data set are available on Spotify, resulting in missing popularity scores for given songs. Due to this fact, the data set is reduced from 106.574 to 22.890 data records.

- **Removal of null values & duplicate data records:**

  Songs with missing attribute values cannot be compared to songs containing values for every attribute. Therefore songs with missing values are removed from the data set. Duplicate data records may unintentionally influence the prediction and are removed as well. Within this step, 282 records are removed.

- **Label binarization of the popularity attribute:**

  The goal is to predict the popularity of a song based on its remaining attributes. Popularity is hereby the target attribute. The discrete attribute popularity, can take values in the range from 0 to 100. Without adjusting the target label, a classification of the songs into one of the 101 popularity classes would result. In order to apply algorithms and train corresponding models successfully, sufficient training data must first be available for each class. In addition, the data should be distributed relatively equal across the classes. These two conditions cannot be guaranteed in this case. Therefore, the logical conclusion is to limit the popularity to the classes *popular* and *unpopular*.

  The existing discrete labels are converted into these binary labels using a fixed boundary value of 11. Thus, records with a popularity above 11 are considered popular and the others are labeled unpopular.

The used data set only contains freely available songs. In comparison to licensed songs these songs are assumed to not be very popular. Subsequently, the data set contains many records with a popularity value of 0. If these data records are initially neglected, the average popularity is 8. Furthermore, half of the given data records only have a popularity value of 5 or less. The top 25% have a popularity value above 11. The highest popularity value contained in the data set is 70. Consequently, a popularity value above 11 can already be considered as relatively high in relation to the existing data.

- **Standardization of features:** To ensure that no attribute can exert a stronger influence than any of the other attributes while applying the algorithms, the values are scaled to a mean of zero an a standard deviation of one.

- **Splitting of the data set for cross validation and balancing:** The existing data set is split into two subsets. 80% of the data is used as training data. The remaining 20% of the data is used for applying the algorithms and models to validate their performance.

  Additionally, the training data is duplicated four times. Each of these four data sets is split into training and test data so that a cross validation can be carried out. In addition, a copy of each training data set is created. These copies are balanced with regards to their popularity using undersampling of majority class, allowing the algorithms to be trained with balanced and unbalanced data.The described splitting of the data can also be seen in table 2.1.

Total data set size:	22608 (relative amount popular songs: 9.65%)		
Training data set size:	18086 (9.76%)		
Validation set size:	4533 (9.22%)		
	Train	**Balanced**	**Test**
Cross-Validation Split:	13563 (9.75%)	2646 (50.0%)	4523 (9.77%)
	13563 (9.75%)	2648 (50.0%)	4523 (9.77%)
	13563 (9.75%)	2648 (50.0%)	4523 (9.77%)
	13563 (9.75%)	2648 (50.0%)	4523 (9.77%)

Table 2.1: Data, training and cross-validation set sizes

Chapter 3

Data Mining

In this section the five classifier algorithms will be applied to the processed data to allow the popularity estimation of songs.

The k nearest neighbors (knn) and the decision tree method offer simple and efficient application and will serve as appropriate first estimators [Hastie et al., 2009, p. 11].

Random forest and gradient boosting classifiers extend classic decision tree algorithms by introducing the idea of ensemble methods thus providing more powerful prediction models [James et al., 2013, p. 316].

With growing popularity within the computer science community, support vector machines perform well on a variety of classification tasks and will serve as the last considered algorithm [James et al., 2013, p. 337].

3.1 Machine Learning Approaches

The following section explains the implementations of said algorithms.

To start, the different classifiers are evaluated separately using 4-fold cross-validation. In addition to testing balanced and imbalanced data, one parameter with three different values is tested per method. This results in six possible applications for each method from which the best performing is selected by considering the area under curve (AUC) of the receiver operating characteristic (ROC) curves.

All five classifiers are fitted again in the second step using the best performing parameter value, and a balanced and full version of the training data. Finally their performance is evaluated on the validation set and compared.

3.1.1 K Nearest Neighbors

Since the most influential parameter for the knn is the size of the computed neighborhood K, the selection of this parameter is trivial. Neighborhood sizes of three, five and seven are tested against each other.

3.1.2 Decision Tree

Decision tree algorithms allow multiple different parameters to be modified. Even though the number of leaf nodes would have been an interesting parameter to test, the tree size was chosen as the varying parameter. Unlimited trees are evaluated against trees sizes of five and ten.

3.1.3 Random Forest

Random forest is a method which can be used for either classification or regression. Due to the fact that it was not covered during the lecture, a brief introduction will be given in the following.

Random forest classification is composed of different independent decision trees, each of these performing a classification task on the given input data and predicts one class. After proceeding a majority vote by all trees, the actual prediction is chosen [Breiman, 2001, p. 5]. By default, sklearn in python uses a bootstrapping process to build the trees on sub-samples of the training set. Each of the sample sets has the same size as the training set because the sets are sampled with replacement. Therefore, duplicates in the set are allowed. Another factor to ensure differences between the trees is accomplished by a random feature permutation at each split of the trees.

In the application bootstrap is not used, due to the fact that a k-fold cross validation is implemented. The forest size was chosen as the coefficient to be varied and tested which defines the amount of decision trees used in the random forest. In the code 10, 50 and 100 trees were used.

3.1.4 Gradient Boosting

Similar to the random forest method, gradient boosting is based on multiple decision trees and can be used for either classification or regression tasks [Hastie et al., 2009, p. 337]. Instead of receiving independent predictions of each tree, this method uses sequential trees which are trained one after another based on the residuals of the previews one [James et al., 2013, p. 316]. In each iteration - e.g. each new built tree - a random sub sample of the training set is drawn without replacement. This is used “to fit the base learner” [Friedman, 2002, p. 1] and update

the current tree afterwards. Similar to random forest, the validation was performed based on 10, 50 and 100 trees.

3.1.5 Support Vector Classifiers (SVC)

To deal with non-linear decision boundaries the data is transformed to a higher dimensional space using so called *Kernel functions*. The parameter to be evaluated was chosen to be the three different *Kernel functions*: *linear*, *polynomial* and *Radial Basis Function* (rbf).

In linear one uses a hyperplane or a set of hyperplanes - defined by its support vectors - to seperate different classes. In polynomial kernel, the dot product is calculate by increasing the power of the kernel. RBF kernel is a function whose value depends on the distance from the origin.

3.2 Evaluation

The parameters obtained from the cross validation are 5 neighbors for the K Neighbors, 100 as the Decision Tree max depth, 5 as the number of estimators for the Random Forest, 100 estimators for the Gradient Boosting and the RBF kernel for the SVC. The ROC curve obtained from fitting the classifiers with the full train data and evaluating it with the valuation set is shown in figure 3.1 and table 3.1.

During parameter tuning one observation concerning balanced and unbalanced data already became visible and is also reflected in table 3.1. Algorithms fitted on unbalanced data tend to have a f1 score close to zero, even when they may have a better AUC than the ones fitted on balanced data. This is the result of overfitting on the majority class of songs with a popularity score smaller or equal to 11 which is the case for 90.2% of the training set. This can be seen in the case of the random forest and SVC method where precision and recall are close to zero. Therefore, the number of predictions for the popularity above 11 must also be close to zero.

During the evaluation, some of the features seemed to have a significantly higher importance in terms of predicting the popularity class. For the gradient boosting method the standard deviation of the zero-crossing rate had the highest importance. The zero-crossing rate is popular feature in information retrieval of music and is used to classify percussive sounds [Gouyon et al., 2000]. The second most important feature is the spectral-bandwidth which describes the range of tones used in a song. However, as a prediction could hardly be made by the used methods, one might conclude that these features do not seem to have a high impact on the popularity of a song.

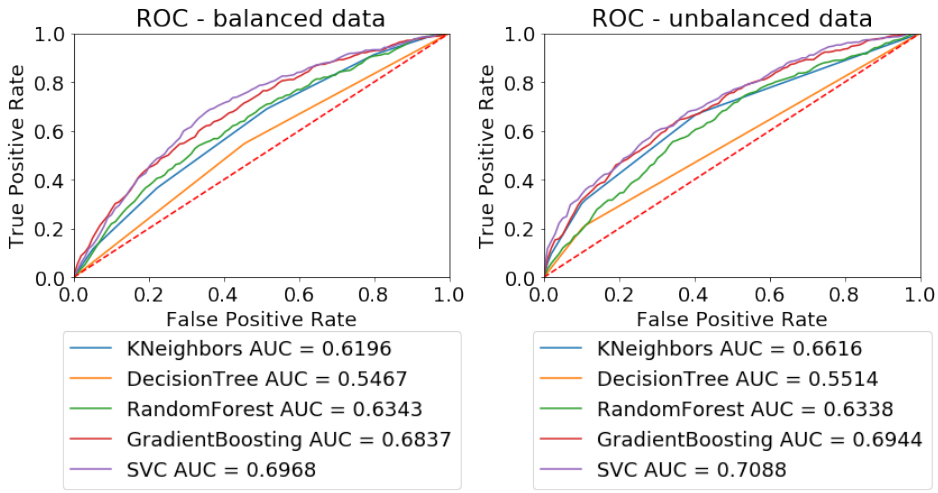

Figure 3.1: ROC of Classifiers with Balanced and Unbalanced Data

Classifier	**Balanced**	**Precision**	**Recall**	**F1 Score**	**AUC**
K-Neighbors	Yes	0.070158	0.690647	0.127377	0.619598
	No	0.009013	0.088729	0.016364	0.661644
Decision Tree	Yes	0.055786	0.549161	0.101283	0.546685
	No	0.021194	0.208633	0.038479	0.55144
Random Forest	Yes	0.066991	0.659472	0.121628	0.634326
	No	0	0	0	0.633835
Gradient Boosting	Yes	0.063094	0.621103	0.114551	0.683732
	No	0.001218	0.01199	0.002211	0.694438
SVC	Yes	0.070646	0.695444	0.128262	0.696829
	No	0	0	0	0.708778

Table 3.1: Performance measure of all classifiers on validation set

Chapter 4

Results

In general, it is to mention that the overall performance of the used classifiers in the selected field of studies showed that predictions on the popularity of a song using a data set of the free music archive cannot be done straightforward. Reasons for this result can be found in various steps of this project.

The popularity obtained from Spotify is calculated using an proprietary algorithm which is not publicly available and therefore, the exact determinants are unknown. During a sample drawn upfront one could see that the popularity highly corresponds with the overall number of plays. Furthermore, Spotify as the only source of popularity does not reflect the overall music market. Other popularity scores could be retrieved from charts which, as a downside, would again depend on different factors such as the country. A worldwide popularity score for all songs can currently not be found.

Another issue is the data set itself. As it is composed of free music, it represents a small sample of the actual music market and might have a bias tending toward unpopular songs. Out of the 22.608 songs that were consistent with the Spotify database, only 8.777 songs had a popularity score of more than 0. In addition to that, the highest popularity score in the data set is 70. This song is called ”White Noise” and does not have a melody or text.

Lastly, the number of features is a challenge. Without having further knowledge about the meaning and impact of the features it is hard to do feature engineering and selection. Generally the algorithms showed difficulties in handling the 518 features.

Bibliography

Search for an item, 2019. URL https://developer.spotify.com/documentation/web-api/reference/search/search.

Kirell Benzi, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. FMA: A dataset for music analysis. *CoRR*, abs/1612.01840, 2016. URL http://arxiv.org/abs/1612.01840.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001. ISSN 1573-0565. doi: 10.1023/A:1010933404324.

Jerome Friedman. Stochastic gradient boosting. *Computational Statistics Data Analysis*, 38:367–378, 02 2002. doi: 10.1016/S0167-9473(01)00065-2.

Fabien Gouyon, François Pachet, and Olivier Delerue. On the use of zero-crossing rate for an application of classification of percussive sounds. 2000.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York, 2009. doi: 10.1007/978-0-387-84858-7.

IFPI. Global music report 2019, 2019. URL https://www.ifpi.org/news/IFPI-GLOBAL-MUSIC-REPORT-2019&lang=en.

Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning*, volume 103 of *Springer Texts in Statistics*. Springer New York, 2013. doi: 10.1007/978-1-4614-7138-7.

Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, volume 8, 2015.

Daniel Sanchesz. What streaming music services pay (updated for 2019). URL https://www.digitalmusicnews.com/2018/12/25/streaming-music-services-pay-2019/.