Song popularity prediction using Spotify data

Nevena Rokvić
Faculty of Technical Sciences
University of Novi Sad
Novi Sad, Serbia
Email: nevrokvic@gmail.com

***Abstract*—Since the Hit Song Science appeared, the field of predicting and understanding hit songs has attracted many researchers who wanted to tackle this problem. Song's popularity depends on many factors: lyrics, season, musical instruments, etc., but we often believe that all mainstream hit songs sound alike. What are the actual components of those songs that make them sound similar is still uncertain. To discover what combination of musical features will lead to a chart-topping hit could be out of great importance to record labels and artists. The music industry is very changeable, and so are the trends. Regarding that, it could be very useful to analyze song before releasing it, and potentially avoid failure. This research attempts to predict whether the song will be popular(hit) or unpopular(non-hit), based on Spotify audio features and song duration. Songs marked as hits reached Billboard Hot 100 chart, and others marked as non-hits didn't. Dataset was split into training and test data. Before building prediction models, audio features and their correlation with the target feature were analyzed. Six predictive models were compared: Logistic Regression, Support Vector Machine, Random Forest, K-Nearest Neighbours, Neural Network and Naive Bayes. All the models were tuned and cross-validated. The best performing algorithm is Random Forest, with 0.83 accuracy on test data.**

***Keywords*- prediction; hit song science; hit songs; popularity; Spotify; Billboard charts.**

I. Introduction

The idea of predicting song's popularity before the song is even released motivated the appearance of Hit Song Science (HSS). This field is a complex combination of factors that may influence certain song's popularity. Some of them include individual emotional history, which makes the prediction even more difficult. Despite the claims that making hit song depends only on a technique, chances are that actual hit-making guidelines are very much unknown. Number of companies offer hit analyzing and predicting services to the record labels, which shows that this field is already perceived as important by the labels [1].

To prove the above mentioned, I have analyzed a group of songs, and further on predicted whether they will reach Billboard Hot 100 chart[2], that is by present day the most referenced music chart. By analyzing audio features retrieved from Spotify[3] the goal was to discover trends and correlations that those feature have with popularity, and also with each other, and afterwards to predict song's popularity based on those features.

It is challenging to solve HSS problem, due to it's complexity. The dataset chosen for the experiment plays a significant role, since the music has changed so much through the years. Despite that, we still have that feeling that all the hit songs sound alike. Apart from audio features, one important part of a song is excluded from this project: lyrics. Catchy lyrics can play significant part in hit-making, but this work is based only on audio features, so they were not considered as important. Lyrics are more transparent to analyze, but on the other hand, their role in hit songs is questionable.

In further text I present dataset and features with more details. Next chapter contains some related works and summary of their experiments. Chapter three explains the methodology that is used in this work, including the dataset and algorithms. Results are presented in chapter four, with graphic and table representation. Summary of the experiment is in chapter four, including some ideas for future enhancements.

II. Related work

There have been numerous investigations that tried different approaches in order to resolve the connection between song's popularity and different musical components. One of them investigated how Spotify audio features relate to the number of streams. The approach they used is not commonly encountered as a solution of HSS problem, and it can be significant to record companies and Spotify itself. Through Spotify API they collected and analyzed 1,000 songs using regression. After data collection and pre-processing, they did a correlation analysis of the audio features using Pearson correlation[4]. The results showed that some of them are negatively correlated (e.g. acousticness and higher streaming count) and some positively (e.g. danceability and higher streaming count). Afterwards, they built a regression model using only relevant variables. As a result of the investigation they concluded that above mentioned attribute-based model couldn't give a satisfactory explanation of a streaming count[5].

Motivation to use Billboard hits instead of Spotify streams as a popularity measure in this work comes from a group of investigators from Stanford University. Their approach was to build a model that predicts whether the song will reach Billboard Hot 100 chart. They randomly collected 10,000 songs from Million Songs Dataset[6], limiting to songs released between 1990 and 2018. Afterwards they combined it with songs that reached Bilboard Hot 100 chart. Final dataset consisted of around 4,000 songs with collected Spotify features through ther API. They used five algorithms to predict songs popularity (Expectation Maximization(EM), Logistic Regression (LR), Gaussian Discriminant Analysis

(GDA), Support Vector Machine (SVM), Decision Trees (DT), and Neural Network (NN)). For prediction they used only audio features(danceability, acousticness, etc.). Instead of that, I included song duration as well. NN and LR provided the best accuracy (0.765 and 0.759)[7].

As an extension to previously mentioned work comes another attempt to predict song's popularity(using Billboard Hot 100), but with notably better prediction results. Since previous works used quite small databases, investigators from the University of San Francisco collected a dataset consisted of approximately 1.8 million songs, which was narrowed down for the investigation purpose to 24,000 songs (half hits, half non-hits). This dataset contains merged songs from Billboard and Spotify. They used LR, NN, SVM and Random Forest (RF) to predict popularity, similar to this work, where apart from that I used two more: Naive Bayes (NB) and K-Nearest Neighbours (K-NN). The results they presented had notably better accuracy score on validation and test data compared to other related works. They reported two best performing algorithms to be RF (0.877) and SVM (0.839)[8].

III. Methodology

Methodology used in this work is explained in this section. I compared six algorithms for prediction: Naive Bayes(NB), Logistic Regression(LR), Neural Network(NN), K-Nearest Neighbours(K-NN), Support Vector Machine(SVM) and Random Forest(RF) algorithm.

A. Dataset and exploratory analysis

The dataset used in this work contains around 18,000 songs released between years 1990 and 2019. It was selected from the original dataset found on Kaggle[9], where the creator collected over 40,000 songs released between 1960 and 2019. Half of them are considered hits (reached Billboard Hot 100 chart), and the other half are non-hits. Hit songs from the '60s, 70s and 80s are notably different than the ones from recent years, and that is why, for the purpose of this investigation, only the last three decades are considered for prediction. Songs from the 80s decade are used only in analysis.

Target label represents song's popularity: 1 if it reached Billboard Hot 100 chart, 0 if it didn't. Popularity was predicted using labels described in the text bellow:

- **Instrumentalness**: song's vocals from 0.0 to 1.0 (the less lyrics it contains, it's more probable to be closer to 1)
- **Danceability**: how apt is the song for dancing from 0.0 to 1.0 (based on some musical parameters such as bass, rhythm, etc.)
- **Loudness**: overall loudness of the song in dB
- **Acousticness**: how acoustic is the song from 0.0 to 1.0 (if the value is closer to 1, there is a high chance that it's acoustic)
- **Valence**: song's positiveness from 0.0 to 1.0 (high valence represents euphoric, cheerful songs, and low valence more depressing, sad songs)
- **Time signature**: how many beats there are in each measure
- **Energy**: how energetic is the song from 0.0 to 1.0 (fast, noisy and loud songs are considered more energetic)
- **Duration**: song's length is in milliseconds
- **Liveness:** it tells whether the song contains audience (if it's performed live), measured from 0.0 to 1.0. Songs performed live have higher liveness
- **Speechiness:** the amount of spoken words in a song from 0.0 to 1.0 (higher speechiness means more spoken words in a song, the opposite from acousticness)
- **Mode:** modality of the song (major-1, minor-0), the type of the scale from which the melodic content is derived
- **Key**: overall key of the song, mapped from Pitch Class Notation to integers (0 = C, etc.)
- **Tempo**: song's tempo in beats per minute (BPM) (in music terms it's a speed of a given peace that comes from the average beat duration)

Danceabiliy, acousticness, energy and valence were analyzed by decades. Unpopular songs released in the 80s have the highest mean value of danceability compared with other decades, as the Table I shows. That is expected, since it suits the 80s music style. Regarding that, songs with lower danceability were marked as unpopular (non-hits), meaning that slower songs with less rhythm were less likely to reach Billboard Hot 100 chart.

When it comes to song's acousticness, in Table II we can see that in the synth-pop and electro music eras (80s and 90s), songs with higher mean values of acousticness were marked as unpopular. Those songs had lower probability to fit into mainstream taste. In other two decades the difference between the two values is not that high.

Another feature is energy, presented in Table III. Values from last two decades have insignificant difference between popular and unpopular songs, while in the 80s and 90s there is a bigger difference, since in those epochs mainstream music style demanded more energetic songs.

Valence is represented in Table IV. The 80s decade is marked with the most positive songs, while the last decade was the one with the lowest values. Songs from the 80s have characteristic upbeat sound, which explains those values. On the other hand, results show that the last decade has a smaller number of "happy" songs that reached the charts, or maybe that the context of "happy song" has changed.

TABLE I
MEAN VALUE OF DANCEABILITY THROUGH DECADES

Decades	Popular songs	Unpopular songs
80s	0.62	0.50
90s	0.65	0.48
00s	0.63	0.46
10s	0.64	0.49

After feature analysis through decades, dataset was merged and feature distributions of hit and non-hit songs were compared. There are 13 features in total, but I chose to present the distribution of danceability, valence, acousticness, speechiness and energy in Fig.1. In the left column are popular songs

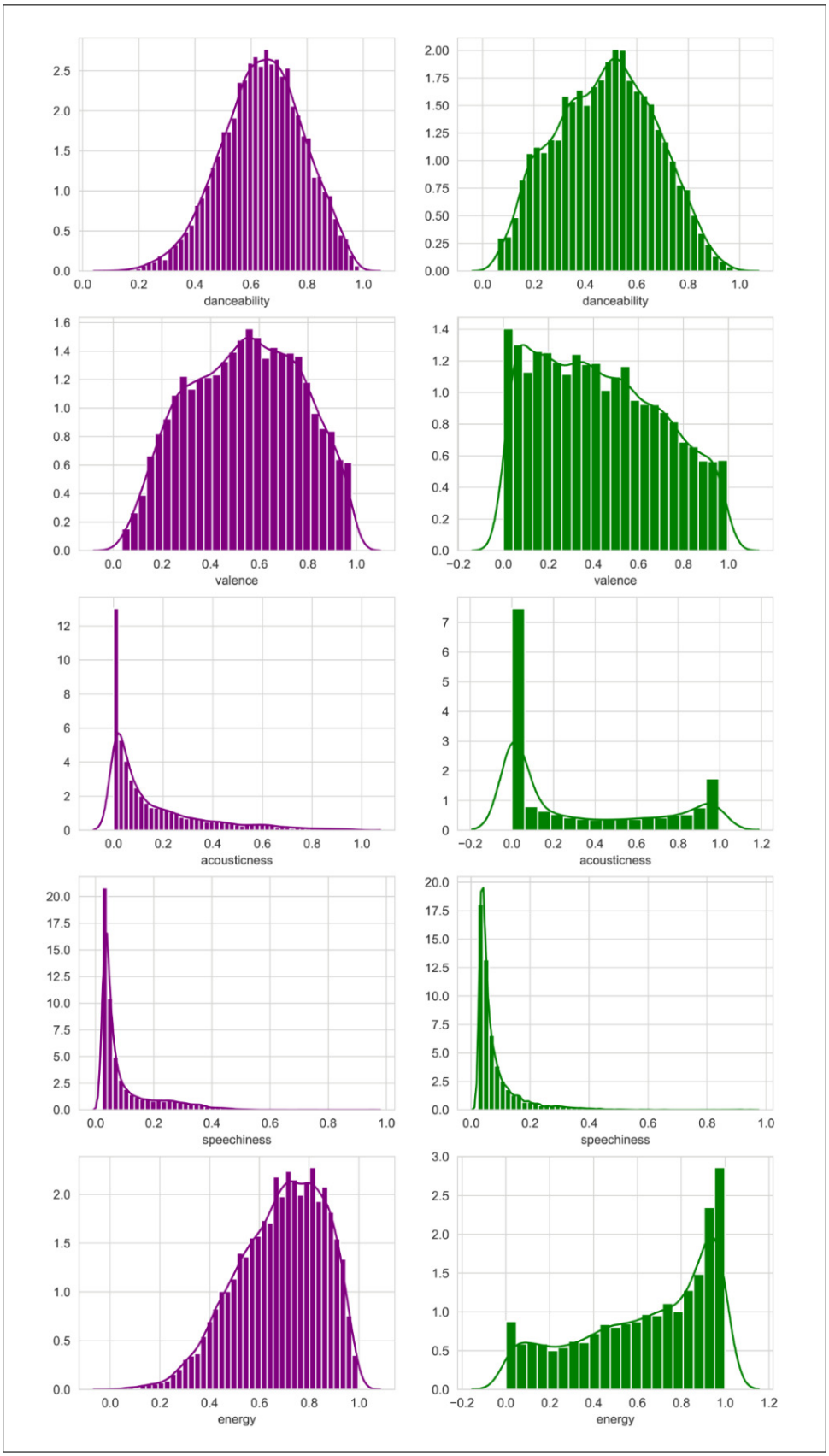

Fig. 1. Distribution of danceability, valence, acousticness, speechiness and energy in popular (left) and unpopular (right) songs

TABLE II
MEAN VALUE OF ACOUSTICNESS THROUGH DECADES

Decades	Popular songs	Unpopular songs
80s	0.20	0.38
90s	0.17	0.41
00s	0.15	0.28
10s	0.16	0.27

TABLE III
MEAN VALUE OF ENERGY THROUGH DECADES

Decades	Popular songs	Unpopular songs
80s	0.65	0.56
90s	0.66	0.55
00s	0.71	0.67
10s	0.68	0.65

TABLE IV
MEAN VALUE OF VALENCE THROUGH DECADES

Decades	Popular songs	Unpopular songs
80s	0.66	0.52
90s	0.58	0.49
00s	0.55	0.41
10s	0.49	0.39

(hits), and in the right unpopular ones (non-hits). Popular songs have higher values of danceability, valence, and energy. According to speechines distribution, the difference between popular and unpopular songs is not big, it tells us that hit songs contain sightly more words. On the other hand, acousticness distribution shows that unpopular songs have higher values of acousticness, which means those songs that are less electronically enhanced have lower chances to reach Billboard Hot 100 chart.

Further I analyzed feature correlation of loudness with energy and danceability with valence using Pearson correlation. There were no missing values, since the data was preprocessed, and all the missing values removed. Categorical values were mapped to numerical ones. As it is shown in Fig. 2, they are positively correlated, which means that louder songs that make more noise are also more energetic. Danceability and valence show that songs with higher valence (happier songs) are more convenient for dancing, and vice-versa.

In the end I applied PCA (Principal Component Analysis)[10] algorithm for dimensionality reduction. The result showed that using eight components 96 percent of the variance is covered, so those eight components will be used for prediction as well.

B. Hyper-parameter tuning and training

Dataset was separated to train/test sets in 80/20 ratio. Cross-validation was done with k-fold validation, to avoid overfitting. I trained and tuned hyper-parameters using scikit-learn Python library[11].

- Logistic Regression

LR is a classification algorithm. It can be used for multi-class problems and binary class problems. LR uses logit

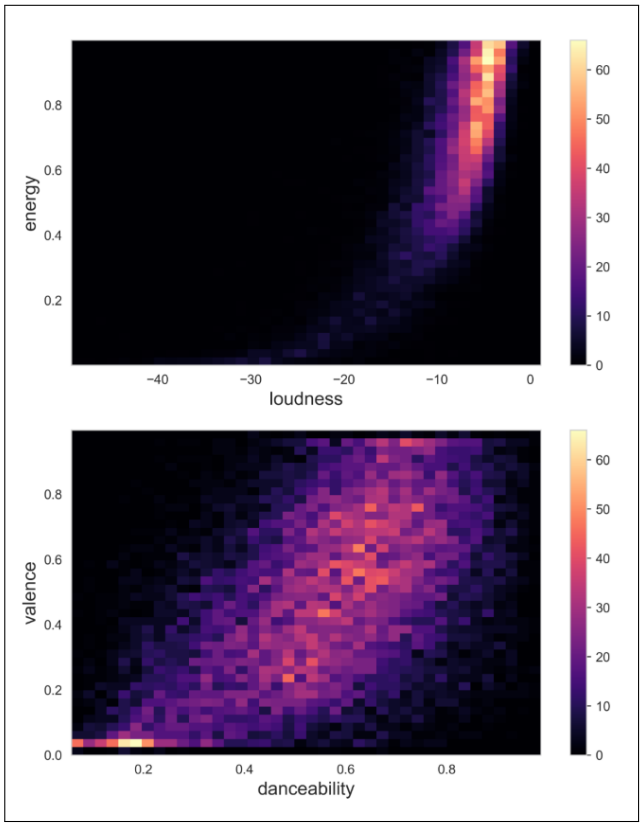

Fig. 2. Correlation of features. Loudness and energy are positively correlated, same as danceability and valence.

(natural log of an odds ratio) and its result is a function that represents the likelihood of a certain outcome. It allows prediction of a discrete value based on a different kind of variables (continuous, discrete, etc.). It normally uses Sigmoid function to map predicted probabilities between 0 and 1[12].

In this work, binary LR model was used. I found it suitable to solve this prediction problem since the outcome is categorical (hit or non-hit). It was tuned using Grid search with cross-validation of five folds, tuned values were following: *C*=1, *class weight* = 1: 0.5, 0: 0.5, *penalty*: "l2", *solver*: "saga".

- Random Forest

Random Forest is an ensemble learning algorithm, used both for classification and regression. It uses bagging sampling approach and the random selection of features to build a collection of decision trees. Every tree from the ensemble acts as a base classifier in order to define the class label. Using majority voting every classifier's vote is used, and the label that gets the most votes is used as a classified label[13].

RF was included in this work because of it's training speed and good performance with high dimensional data. It was tuned using Randomized search. It was trained with 1786 estimators, minimal samples split of two, maximum feature number of four, bootstrap and maximum depth of 40.

- Support Vector Machine

SVM is a classifier defined by separating hyperplane. It separates classes on different sides of a line (hyperplane). It

is used for linear and non-linear classification, which means that the boundary between data doesn't have to be a straight line[14].

SVM is considered a black-box model, however, one of the reasons it was selected for this work is because it's easy to use and memory efficient. It was tuned using Grid search. Model was trained with tuned parameters (C=0 and $gamma$=1).

- K-Nearest Neighbors

K-NN is supervised learning algorithm used for classification. It calculates the distance between data points using Euclidean distance, in order to find nearest neighbours. It calculates the distance between test sample and all the other train samples, and finally decides where to classify depending on the n nearest neighbours. Parameter n represents the number of neighbours from which the classification is done. It is normally an odd number, and the test item will be classified to the class that has more members among those n neighbours[15].

This model was selected since related works also used it to solve a similar problem, and it gave satisfactory results. Also, it is very intuitive, and easy to understand. It was tuned with Grid search, and trained with metric "manhattan" and 19 neighbours.

- Neural Network

NN is a non-linear algorithm that can receive large number of inputs (independent variables) to give one or more outputs (dependent variables). It has many variations, but the most used one is MLP (Multilayer perceptron). It contains one input layer, one or more hidden layers, and one ouput layer. Neurons have a defined number of inputs (from previous layer or outside the NN), and a defined number of outputs. The output is computed from each neuron as a weighted sum of all its inputs, with the help of some activation function[16].

I used NN in this work because one of the related works obtained good results using it for the same problem. This NN has one hidden layer, learn rate of 0.01, rectified linear unit (ReLu) activation function. Learn rate of 0.01 was chosen because it gave the best results among other tested rates, activation function ReLu was chosen because it is known to generalize well. Batch size and epochs were tuned using Grid search. Batch size was set to four, and training lasted 70 epochs.

- Naïve Bayes

Naïve Bayes is a simple probabilistic classifier based on Bayesian Theory. It assumes that target values and attribute values are conditionally independent, and it ignores possible correlations among the inputs. It does not need a big amount of training data to establish the estimated parameters needed for classification[17].

Since it is very easy to implement and doesn't need any tuning, I decided to compare it to other chosen algorithms. Also, I was curious to see how it works predicting hits, since it this problem certain correlations do exist. Naive Bayes (GaussianNB from scikit-learn) has no hyper-parameters.

IV. Results

As a main evaluation metric I used accuracy, since the data is balanced. As additional metrics I present precision and recall as well.

TABLE V
Experiment results

Models	Accuracy		Precision		Recall	
	Val	Test	Val	Test	Val	Test
RF	**0.84**	**0.828**	**0.853**	**0.83**	**0.841**	**0.83**
LR	0.79	0.8	0.832	0.81	0.83	0.8
NN	0.82	0.816	0.831	0.82	0.841	0.82
SVM	**0.822**	**0.821**	**0.843**	**0.83**	**0.838**	**0.82**
K-NN	0.8	0.79	0.822	0.81	0.805	0.79
NB	0.76	0.76	0.78	0.78	0.76	0.76

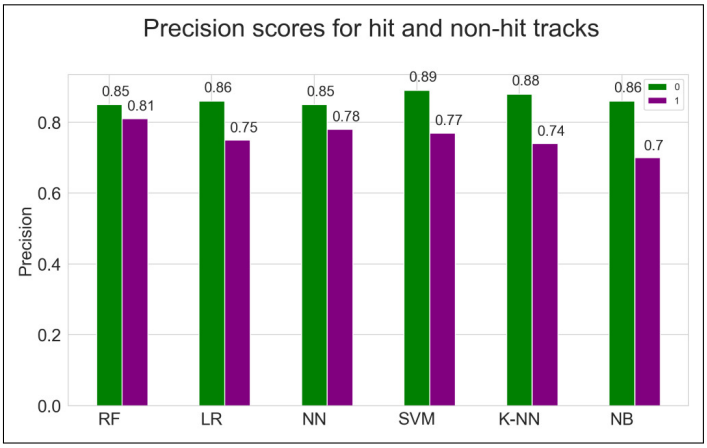

Fig. 3. Precision scores. SVM has the highest precision for non-hit prediction, RF for hit prediction.

Training was conducted with different number of features and selected components from PCA algorithm for comparison. I tried to exclude one of the two correlated features (for example danceability/valence) or to use only features that are highly correlated with target feature (above some threshold, according to Pearson correlation) but in all the cases the accuracy was lower. Also, I tried training some of the algorithms with eight components obtained using PCA algorithm, and it resulted with lowering the accuracy scores (LR to 0.78, RF to 0.8, K-NN to 0.77 on validation data).

The best obtained results were obtained when models were trained with all 13 features. The difference between validation and test results is reasonable, which means that models learned how to generalize, without over-fitting. As it is shown on Table V, the model with the best results is Random Forest, the same as in [8]. My model has slightly lower accuracy of 0.83 (in comparison with 0.88) on test data, but higher then other related work. The worst performing algorithm is Naive Bayes, with 0.76 accuracy. All the models presented in this work outperformed the ones showed in [7], but they are also trained on more data.

Compared with [8], where they used LR, SVM, NN and RF, the performance of the models is quite similar (RF being with the highest difference).

It seems interesting to pay attention on how many predicted hits are actually hits, and the opposite, as it can be costly for the ones who decide to invest. That is why precision is presented in Fig. 3 for both outcomes of the target label. The diagram shows that all the models have higher precision in predicting non-hits. RF has the smallest difference of precision in predicting hits and non-hits, while SVM gives the best result for predicting non-hits (0.89). It means that 89 percent of predicted non-hits are actually non-hits.

V. Conclusion and future work

With the evolution of technology, it is expected from label companies and artists to be interested in knowing the potential destiny of their new song. In this work I tried to contribute to better understanding of what are the actual factors of success are, and what do all hit songs have in common. It is hard to tell, since it depends on many factors, including the current trends.

When it comes to feature analysis, I discovered that songs marked as hits tend to have higher values of danceability, energy and valence. That leads us to the fact that more vibrant and louder songs, that have more rhythm and stronger bass, have greater chances to become chart topping hits than the melancholic, sadder songs with more instrumental sequences.

In terms of prediction, excluding some of the correlated features or using PCA components for model training gave slightly lower accuracy scores. Using models trained with all 13 features I demonstrated that hit songs can be predicted with a relatively high accuracy. RF and SVM outperformed LR, NN, K-NN and NB, but K-NN and NB have surprisingly good precision in predicting non-hits, so they also can be used by record labels to determine if the song's is suitable the market.

In the future work I would like to collect more recent data from the past decade, and include number of streams on Spotify as a new feature. Genre was not included in this investigation, since on Spotify every song belongs to multiple genres. Knowing that, it would be interesting to see which genre contains more popular songs, or how much it affects song's popularity.

The winning combination of features for making chart-topping hits is still unclear, so diverse experiments in this field could contribute, and move the HSS one step closer to more precise conclusions.

References

[1] François Pachet and CSL Sony. Hit song science. *Music data mining*, pages 305–326, 2012.

[2] Billboard hot 100 chart.

[3] Spotify.

[4] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. In *Noise reduction in speech processing*, pages 1–4. Springer, 2009.

[5] Rutger Nijkamp. Prediction of product success: explaining song popularity by audio features from spotify data. B.S. thesis, University of Twente, 2018.

[6] Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere. The million song dataset. 2011.

[7] Elena Georgieva, Marcella Suta, and Nicholas Burton. Hitpredict: Predicting hit songs using spotify data.

[8] Kai Middlebrook and Kian Sheik. Song hit prediction: Predicting billboard hits using spotify data. *arXiv preprint arXiv:1908.08609*, 2019.

[9] Farooq Ansari. The spotify hit predictor dataset (1960-2019).

[10] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[11] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[12] Shahan Syed, Muhammad Mubeen, Adnan Hussain, and Irfan Lal. Prediction of stock performance by using logistic regression model: evidence from pakistan stock exchange (psx). *Asian Journal of Empirical Research*, 8, 07 2018.

[13] Khaled Fawagreh, Mohamed Medhat Gaber, and Eyad Elyan. Random forests: from early developments to recent advancements. *Systems Science & Control Engineering*, 2(1):602–609, 2014.

[14] Savan Fatel. Chapter 2 : Svm (support vector machine) — theory.

[15] k nearest neighbor classifier (knn).

[16] H. Zare Abyaneh. Evaluation of multivariate linear regression and artificial neural networks in prediction of water quality parameters. *Journal of Environmental Health Science and Engineering*, 2014.

[17] Fitriana Harahap, Ahir Harahap, Evri Ekadiansyah, Rita Sari, Robiatul Adawiyah, and Charles Harahap. Implementation of naïve bayes classification method for predicting purchase. pages 1–5, 08 2018.