Utrecht University

Artificial Intelligence

Faculty of Humanities

Automatically Predicting Popularity of Music Tracks Based on Lyrics

Author:
Sander van der Somme
Student number:
6287247

Supervisor
Gizem Sogancioglu
Second Reader:
Denis Paperno

January 28, 2021

Abstract

Hit song science is a machine learning field that focuses on the prediction of the popularity of unreleased tracks. This research aims to examine how different classifiers perform at this task and what features can help predict the popularity of scores. A dataset is built by combining data from the LFM-1b dataset, data retrieved through the Spotify developer API and a lyrics dataset created in a study of Markus Schedl on music information retrieval. After filtering and preprocessing this dataset, features were extracted that comprise of sentiment analysis features, an explicit feature retrieved from Spotify, a repetitiveness feature and tf-idf features. Three models, including a K-Nearest Neighbors classifier, a Naive Bayes classifier and a Support Vector Machine (SVM) were then trained on different combinations of the features. From these classifiers, the SVM performed best when trained on a feature set containing only tf-idf features and obtained an F1-score of 0.60. Unfortunately, each of the models was unable to predict the popularity of highly popular tracks, due to the highly unbalanced feature set.

Acknowledgement

I would like to thank Gizem Sogancioglu, Denis Paperno and Francisca Pessanha for supporting me in conducting this research. Also, I thank Markus Schedl for making the lyrics dataset that he has created in one of his earlier studies available to us to be used in this research.

Contents

Chapter 1

Introduction

In the last decades, the music industry has changed a lot due to various technological advancements. With the arrival of several online music streaming services, music has become more available to the masses. Making music more open to the public led to an annual increase of investments (4.1 billion dollars) by record labels looking for new talents [1]. The collections of music tracks available have become so enormously large [2] new methods needed to be developed to search for and retrieve musical content efficiently. This demand led to the emergence of the machine learning field Music Information Retrieval (MIR). MIR deals with the classification of different topics in music (e.g. emotions, instruments, artists, genre, etc. [3]) and finding similar musical content. Based on the similarity between songs, calculated by comparing different attributes of those songs through machine learning, predictions are made about songs having specific properties. These attributes are also called features and are dividable into different categories [2]: The metadata of a song represents the first group of music features. It describes what artist composed or performed the song, what record label released it, what year it was released, its genre, its song title, and so on. Another category represents the content-based features of a song. This group of audio features encloses various acoustic properties such as timbre, energy, and rhythm. Finally, the last group represents the symbolic nature of songs. It not only represents the semantic value of a melody but also describes associated emotions.

Besides a need for more high-performance music information retrieval systems, the growth in the number of musical content available online also poses a challenge to new artists trying to display their talent to the world. Although putting out content has become more accessible, it has become more difficult for tracks to stand out from the vast array of songs already out there. Even if a song turns out to be successful, it does not mean it will remain that way as the average time a hit survives in the charts has significantly decreased over the decades [4] due to more songs appearing in the charts.

A specific subdomain of MIR is engaged in predicting whether an unreleased song will become a hit. This field of hit prediction is also known as Hit Song Science and could give artists insight into what aspects of musical content affect the popularity of their tracks. This could enable them to release content that will attract more listeners and increase the odds of their songs becoming all-time favourites.

1.1 Related works

Within the field of hit song science and MIR, there already have been conducted several types of research, some of which focus explicitly focus on the lyrical aspect of songs, whereas others focus on acoustic features only. For example, Herremans [3] used different classifiers to predict whether dance songs appeared in the top 10 charts based on acoustic audio features as well as metadata and temporal features. They implemented a range of different types of classifiers of which a Logistic Regression (LR) classifier and a Naive Bayes Classifier (NBC) returned promising results. Research by Interiano et al. [5] used an enormous database of over 500.000 tracks for training a random forests model to predict the popularity of songs. Their features consisted mainly of acoustic features (such as timbre, tonality and danceability) but they also included a 'superstar' attribute which signifies whether the artist of a track has appeared in the charts before. MIR tasks are not reserved to English lyrics only, as is shown by research trying to classify Turkish music tracks [6]. They investigated the performance of an SVM on a large feature set comprising of audio features as well as lyric features (including the presence of different rhyme forms and vocabulary richness). They managed to obtain an astounding accuracy score of over 99%. However, their results might be flawed as they did not use other metrics to measure the performance of their model. If their dataset is highly unbalanced, such a high accuracy score does not convey whether or not the minority class in their dataset is classified correctly often. In that case, adding a metric such as recall to their evaluation would do a better job of assessing the performance of their model, as it could indicate that the minority class is predicted accurately as well. Further, Malheiro et al. looked into Music Emotion Recognition (MER) which focuses on identifying certain types of emotions within songs. This research specifically concentrated on the classification of emotions in lyrics [7]. They used unique features like the presence of slang, the structure of the lyrics, and other semantic features in combination with a Support Vector Machine (SVM) classifier to obtain high accuracy classifications. Schedl [8] analyzed differences in genres between lyrics and the Wikipedia page of the corresponding artist. They found that different genres differ in lyrical features such as repetitiveness and readability. In [9], Choi et al. used several types of classifiers to classify the subject of lyrics and user interpretations of those lyrics. They reported that user interpretations in most cases resulted in more accurate classifications and that SVMs and NBCs ended up returning the best results.

In the majority of the researches already conducted in MIR and the field of hit song science, the focus is mainly on acoustic audio features and metadata and often rather small datasets of less than a couple thousand tracks have been used. Although some of these researches show promising results at performing machine learning tasks within MIR, the influence of lyrics on the popularity of songs is often neglected. Although there are exceptions, the researches that do focus on lyrical features often use standard text-mining features only (such as TF-IDF scores and POS-tagging features). The aim of this research is the prediction of the popularity of pop music by using a larger dataset and focusing on a combination of unique lyrical features and common text-mining features.

1.2 Research Questions & Academic Relevance

To contrast existing works already out there, the aim of this research is the prediction of the popularity of pop music by using a larger dataset and focusing on a combination of unique lyrical features and common text-mining features. Through comparing the outcomes of different machine learning approaches within the field of MIR, insight could be gained into what approaches perform best at various tasks known in hit song science. This could contribute to future research of which the purpose is to optimize hit prediction systems. Therefore, the main research question is as follows:

> *RQ1: Which machine learning approach is best capable of predicting music popularity based on lyrics of songs?*

In addition to identifying what supervised machine learning approach will work best for this specific machine learning task, discovering what factors actually contribute to the popularity or success of new songs might lead us to rethink how we compose music and write lyrics. Also, understanding what exactly makes song lyrics successful could be of use in music generation by AI techniques. Therefore, the following subquestion is posed:

> *RQ2: What features can be extracted from music lyrics that could help predict the popularity of lyrics?*

1.3 Overview

This paper is structured as follows. In the next chapter, a description is given of what dataset was used and how it was processed, as well as what features and classification models were implemented and used. In section 4 the results obtained by the classification models are presented. Finally, in section 6, the conclusions that were drawn are discussed and directions for possible future works are provided.

Chapter 2

Method

This research was done by following a certain order of steps. First, data was gathered from several sources followed by efforts to automatically annotate the data with labels. This step included filtering and discarding redundant or unusable data. The next phase included the preprocessing of the data as well as feature extraction from the data. From those features, models were built and evaluated on their performance. The subsections below discuss why and how each of these steps was executed or implemented.

2.1 Dataset

The data used in this research was gathered from three sources. Different kinds of information such as song and artist information as well as a large number of listening events were obtained from the LFM-1b dataset available online [10]. The LFM-1b dataset is an online dataset of over a billion listening events of music listeners from different music streaming platforms. It has been made publicly accessible especially to be used in MIR tasks. Because the data was spread over multiple files and only a subset of the data was needed, the datasets needed to be merged and adapted to be useful for this project. An overview of the specific steps performed to transform the dataset is provided in figure 2.1.

The work of Schedl & Markus [11] provides a clear description of the contents of each of the files of the LFM-1b dataset. For this project, only the files providing the artist information, the track information and the listening events dataset were required. The other files were left untouched.

To build features from lyrics, the dataset used in the work of Schedl [8] was provided by the author. These lyrics in this dataset were retrieved by using the song information in the LFM-1b dataset as the basis (and thus some fields correspond to the LFM-1b dataset). The contents of both the lyrics dataset and the required files from the LFM-1b datasets are presented in tables 2.1 and 2.2.

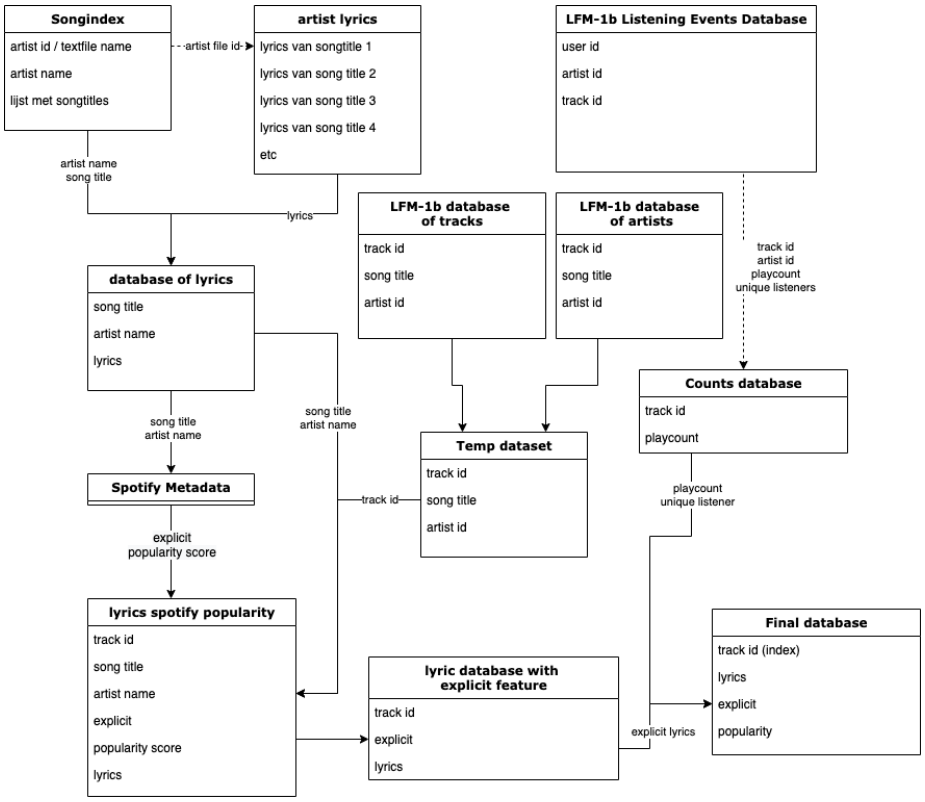

Figure 2.1: Dataset transformation framework

File	Contents
LFM-1b_artists.txt	Artist id, Artist-name
LFM-1b_tracks.txt	Track id, track-name, artist-id
LFM-1b_LEs.txt	User-id, artist-id, album-id, track-id, timestamp

Table 2.1: Contents of the LFM-1b datasets

The lyrics dataset was divided into a song-index file (containing artist ids, names and a list of their songs) and a folder filled with text-files containing lyrics (where each filename was titled as the corresponding artist id and contained the lyrics of each of that artist's songs). To combine the data from the lyrics dataset with the data of the LMF-1b dataset, the right song titles needed to be matched with the right lyrics by looking up those lyrics in the corresponding artist's file. This resulted in a dataset in which each entry contained a song title, artist name and the matched lyrics. Although the artists and tracks files from the LFM-1b dataset contain a lot more entries than the number of tracks matched with their lyrics, only the tracks of which the lyrics were available were used. As most of the features would be created by processing the lyrics, tracks without lyrics would not be not usable. Therefore, these tracks were discarded along with the artists (from the LMF-1b dataset) that were not present in the lyrics database.

During the exploratory phase, it was noted that there was a significant va-

File		Contents
Songindex.txt		Artist-id, artist-name, list of song titles
Lyricsdata		
↪ Artistid.txt	1.txt	list of lyrics per song of artist 1
	2.txt	list of lyrics per song of artist 2
	3.txt	list of lyrics per song of artist 3

Table 2.2: Contents of the lyrics dataset

riety in languages among the lyrics. Taking into account that the larger part of the features would be obtained through the use of text processing techniques, we used the Langdetect python package [12] (which is based on Google's language detection library) to filter out non-English lyrics. It was chosen to do this as the presence of multiple languages could negatively impact the performance of the models and because some text processing techniques (i.e. tf-idf) rest on the principle of composing a vocabulary of the entire lyrics dataset.

Since temporal evolution of hit songs has shown that songs of different decades display differences in characteristics [5] and this research intends to classify modern-day hit songs, the dataset was also filtered to only contain songs that are popular nowadays by discarding older songs. To achieve this, song metadata was retrieved from Spotify's developer web API [13]. Spotify has enabled developers to use music data in their apps by making their track and artist metadata, audio features and much more accessible online through their platforms web API. Spotify has annotated each track with its calculated popularity scores. These scores lay in between 0 and 100 and are based on the number of times the song has been played and how recent those plays are. Although it is unknown how these popularity scores are calculated, Spotify has proven that their sorting mechanism (which uses these scores) performs adequately and therefore, it was decided to use these scores as a filtering tool to only keep songs that are still popular today. Conveniently, Spotify has also annotated their tracks with the 'explicit' attribute that represents the presence of explicit lyrics in a track. To save time and computational resources, the values for the explicit attribute of each track were retrieved simultaneously with the popularity scores, resulting in the first feature. Retrieving the data from Spotify was easily done through the use of the Spotipy python library [14], which provides a function to search the Spotify database using keywords. The Spotify database was queried for the song titles and the artist names from the lyrics dataset. Matching the queries with the right songs and artists did not always return a correct result. For this reason, the tracks of which these attributes were not retrievable from the lyrics dataset were discarded, resulting in a dataset of 7808 tracks.

As the data did not come with precalculated popularity scores, in the following step aimed to annotate the data with self-defined labels. Note that we refrained from using the previously discussed popularity scores of Spotify, as it remains unexplained how these scores have been calculated. Instead, the LFM-1b datasets of listening events, artist information and track information were used to calculate playcount scores which would then be transformed into categoric labels. These scores, that represent how many times a track was listened to were determined for each track by counting how many listening events

referenced the corresponding track id. This was a time-consuming task as the LFM-1b dataset of listening events contained 1.088.161.692 entries. To accomplish the computing of these scores, tracks from the lyrics dataset (that only contained song titles and artist names) needed to be paired with the right track and artist ids first. This was a necessary step, as the listening events database only included artist and track ids and no song titles or artist names. After completing this process, the resulting scores ranged from 1 to over 100.000. It should be recognized that these scores do not fully represent the popularity of songs, as popularity is not defined only by the number of times it has been listened to, but also by the number of unique listeners that have listened to the track. Although only using the playcount of tracks results in a simplistic representation of popularity, the unique listener counts were not calculated due to time constraints. It should also be stressed that this final dataset was biased towards unpopular songs as the majority of the data had a low popularity score whereas only a small fraction had a high popularity score. Also, after computing the popularity scores, it was noted that the most listened songs appeared to be hits released in the early 2010s (for example, "Gotye - Somebody that I used to know" in 2011 and "Fun - we are young" - 2012). It is therefore also worth pointing out that the models will most likely not perform well on predicting songs released nowadays.

Since the goal of classification is to classify data points into different categories, it was required that the datapoints were assigned to classes according to their popularity scores. For that reason, the numeric popularity scores were transformed into textual labels by simply assigning each numeric value falling within a certain range into a certain class. The classes represent groups of tracks that are considered either of low, intermediate or high popularity. Every entry in the database with less than 2000 listening events was labelled as low popularity and every entry with more than 20.000 listening events was labelled as high popularity. Every entry of which its number of listening events falls in between these two boundaries received the label of intermediate popularity.

2.2 Preprocessing

After merging all of the datasets into a final form, it was required to transform the data itself into processable form. The steps for preprocessing the data were performed in the following order:

Encoding categories

Since classification models can only work with numerical values, it was required that the numerical vectors in the dataset were transformed into a numerical datatype. Transforming the boolean explicit feature and the textual label vectors were easily done through the use of the python library Scikit-learn [15]. The explicit feature was turned into a vector with ones and zeros only (representing boolean values True and False) and the label vector was changed into a vector containing only the values 0, 1 and 2 (each representing a class).

Removing digits

During the exploratory phase of the data, the presence of digits in some of the lyrics was discovered. It was opted to remove the digits from the lyrics, as they usually indicate quantities of some sort and stripping them away does not very often change the meaning of a piece of text significantly.

Expanding English contractions

Expanding contractions: It's very common for English lyrics (or all sorts of English texts for that matter) to contain contractions (for example "'aight", "can't", "'cause" and "dunno"). To avoid building lexicons from the lyrics dataset (during the creation of features) in which different entries portray the same meaning, it was chosen to expand the contractions using the python contractions library. The former examples would be translated to "all right", "cannot", "because" and "do not know" respectively. Note that when building a vocabulary designed to retrieve tf-idf scores, expanding these contractions should result in a smaller lexicon and thus reducing the number of dimensions.

Lowercasing characters

As programming languages are case-sensitive, all uppercase characters were translated into lowercase characters. It should be emphasized that due to the limited timeframe, no English abbreviations were expanded prior to the lowercasing of all characters. This consequentially forms a drawback as certain abbreviations will lose their original semantic value. For example, during this process, 'US' will be translated to 'us', and thus its semantic value has changed.

Removing Punctuation

Almost all data contains noise. In-text data, this noise often displays itself in the form of punctuation (although the earlier removed digits can also be considered to be noise). Removing this noise is an important step to do before feeding the data into a model. By using the Python built-in string library, removing punctuation from strings was a simple task as it provides a collection of all punctuation characters (such as comma's, colon's, parentheses, etc). It was chosen to perform this step after expanding all the English contractions, as most contractions contain punctuation.

Tokenization

A very common natural language processing task is tokenizing all the words that are present in the textual data. As earlier mentioned, computers can only work with numeric data. This means that textual data has to be processed in a way to transforms it into numeric data. Tokenization is the act of breaking down a piece of text into its subparts. In this research, each sentence was split into the words that it consists of. For example, the phrase "The cat is chased by the dog" would be separated into a list of the following tokens: 'The', 'cat', 'is', 'chased', 'by', 'the' and 'dog'. Representing the lyrics in this form is required for the tf-idf feature (discussed in section 3.3) which transforms the single word tokens into numerical values that are processable by the computer. Fortunately, the

NLTK python library provides a package developed for tokenizing text which made this task easy to do.

Removing stopwords

In addition to removing noisy data such as digits and punctuation, filtering stopwords can significantly boost the performance of models. Stopwords are words that commonly appear in all sorts of text, such as 'the', 'it', 'because', 'who', 'to', and so on. Not only does removing stopwords reduce the size of a vocabulary built for extracting features, but the presence of stopwords in textual data also does not contribute much to the semantical meaning of a text given that they commonly occur in all sorts of text. Therefore, they can't be used to reliably measure the relevancy of a text, whereas words referring to actual topics can. Again, the NLTK library was used, as it comes with a corpus package which contains a collection of stopwords.

Lemmatization

Lastly, another very popular text processing technique is a form of text normalization where words are reduced to a root form. For instance, the word 'caring' would be shortened to 'care'. Another form of text normalization is stemming. Although stemming also reduces words to a root form, the implementation of lemmatization was preferred as lemmatization. Unlike stemming, lemmatization reduces the word to a root form that still is considered as a word of the language and often preserves the semantical meaning of a word. In the earlier example, stemming would have turned 'caring' into 'car'. Consequentially, the semantical meaning of the token would now refer to a vehicle instead. Performing lemmatization was a straightforward task as most of the hard work already covered by the NLTK stemming package. It includes a WornNetLemmatizer class which did a good job of normalizing the lyrics in the dataset.

2.3 Feature Extraction

To build and train each of the different classifiers, a group of initial features was selected. A subset of those features that ended up being used for training the models is described below. The features that were not implemented will be discussed in the future works section. Note that some of the features were computed between certain preprocessing steps.

Sentiment Analysis

As music lyrics are known to convey certain messages and moods, it was decided to use a feature based on the attitude of song lyrics. The NLTK python library that has been broadly used for natural language processing tasks [16]. By using this library, retrieving scores for different sentiments such as positivity, neutrality and negativity were done effortlessly. The NLTK library provides a sentiment analysis package called Vader which is specifically designed for analyzing sentiments in raw text. Therefore, this feature was extracted before any preprocessing task was applied to the lyrics. The SentimentIntensityAnalyzer

class supplied by the Vader package uses the unprocessed text to calculate the different sentiment scores. For example, the sentence 'The music is nice' would return the scores 0.7, 0.0, 0.6 which represent the positive, negative and neutral sentiment respectively. It is even able to take into account emphasis of emotion expressed in a text through the use of capitalized characters and punctuation (e.g. "I REALLY ENJOYED THE MOVIE!!" displays a very intense positive sentiment, and thus would return a high positive score). It also returns a compound score which is a weighted and normalized metric that combines positivity, neutrality and the negativity scores. By including these features, the dataset contained a total of five ready-to-use features.

Repetitiveness

It is a well-known property of lyrics that parts of the lyrics are repeated. Comparable to the repetitiveness feature used by Schedl [8], a feature was implemented based on the assumption that lyrics tend to repeat itself. The works of Schedl explained that repetitiveness of lyrics could be calculated by how much the lyrics can be compressed. Although Schedl refers to an article on text compression, it was opted to calculate compression scores more simplistically. A compression score was computed for each tokenized lyric by dividing the number of unique tokens in a lyric by the total amount of tokens in that lyric and then taking the complement of that score (see function below). This step was performed immediately after removing all stopwords from the lyrics. Removing stopwords significantly decreased the number of words that occur very often in all texts and thus do not contribute much to the repetitiveness of a text.

$$Compressionscore(lyric) = 1 - \frac{|unique\ tokens|}{|tokens|}$$

Tf-idf features

Term frequency-inverse document frequency (tf-idf): The last features were extracted through calculating tf-idf scores. Tf-idf is a measure for evaluating the relevancy of words to a document within a set of documents. Scores obtained are represented in a sparse matrix. Each column represents a word and its values represent the relevancy of that word to a lyric. Since the vocabulary of all lyrics combined makes up for over 100.000 words, dimensionality reduction was applied (by setting the max-features parameter of the svc model to 1000) to reduce the matrix to a maximum of a thousand columns. These remaining columns describe the thousand most relevant words in all the lyrics. The matrix was then converted into a dataframe and thus contained a thousand features.

Because the tf-idf feature set was significantly bigger than the combined feature set (consisting of the explicit feature, the sentiment analysis features and the repetitiveness feature), it was not clear whether or not the tf-idf features would dominate the other features. Therefore it was chosen to compose different feature sets and train each of the models on each of the sets to analyze the impact of the tf-idf features on the final performance on the models. Table 2.3 describes which feature sets were composed and which features they comprise.

After the preprocessing of the data was finished and all the needed features were extracted from the lyrics, redundant columns (such as track id, song title

Feature Sets	Features
Mixed set	Explicit, sentiment analysis features, repetitiveness feature
Tf-idf Set	Tf-idf features
Concatenated set	Tf-idf Set, Mixed set features

Table 2.3: Contents of feature sets

and the lyrics) in the final dataset were dropped as they are not useful for training the models.

2.4 Classification Models

Using the created set of features for each track in the database, several popular models were built, each of which used for classifying the tracks into the earlier described classes (low, intermediate or high popularity). Using the Scikit-learn library two Support Vector Machines (SVM), a Naive Bayes Classifier (NBC) and a K-Nearest Neighbours Classifier (KNN) were implemented. As each of these models has been proven to perform well in other related works, implementing them in this research seemed to be a good choice.

2.5 Performance evaluation

Once the models were built, the dataset was split into training and test sets, where 80% and 20% of the dataset were assigned to the training set and test set respectively. Ultimately, the test set contained 1562 test data points of which approximately 100, 700 and 760 were of low, intermediate and high popularity respectively (although it varies a little each time the program is run). The training set was then used in cross-validation and parameter tuning of the model. Afterwards, the models were fitted to the training data and each of the models' performances on the test set was evaluated by computing several evaluation metrics: accuracy, precision, recall and the F1-score. Essentially, accuracy measures how many data points from the entire test set have been correctly predicted. Precision measures the ratio of how many of the predicted positive labels are actually positive according to their real label. Recall measures the ratio of how many of all data points in a class actually have been predicted correctly. F1-score combines both the precision and recall scores through weighted averaging. Each of these metrics gives insight to a different aspect of the performance of a model and can be calculated by using the following formulas below.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$
$$precision = \frac{TP}{TP + FP}$$
$$recall = \frac{TP}{TP + FN}$$
$$F1 - score = 2 * \frac{precision * recall}{recall + precision}$$

Here, TP and TN stand for True Positives and True Negatives (the amount of correctly predicted positive and negative labels), and FP and FN stand for False Positives and False Negatives (the amount of incorrectly predicted positive and negative labels). Note that especially recall and f1-scores are important in this study as the used dataset is unbalanced. A low recall score conveys that a certain class is not predicted well, which could indicate that a class that makes up for only a small part of the dataset is classified poorly. Also, since the dataset was biased towards tracks of low popularity it was chosen to include weighted averages in the calculation of the precision and F1-scores. This means that the results of these metrics were found by averaging the computed scores of a metric for each of the classes, with the smaller classes having a bigger impact on the final score than the larger classes. The accuracy scores do not need to be weighted as it is not computed for each class separately. Lastly, unweighted averaging was used to compute the recall scores. Weighted averaging should deal with the unbalanced labels. To assess whether the scores are good or bad, baseline accuracy and recall scores were computed (the other scores are weighted and since it is unknown what weigths the model used during evaluation, we refrain from calculating these scores by hand). These scores are based on the notion that a classifier would assign the majority class label to each data point in the test set. These scores represent a bottom threshold and will be used in comparison with the actual performances of the models. If a model obtains a score lower or equal to this baseline, it means the classifier does not perform well. Any score higher displays that the model does a relative better job at the task at hand. Finally, for further evaluating the models, the results were visualised in confusion matrices. Plotting the confusion matrices was a simple task through the use of the python libraries Matplotlib [17] and Seaborn [18].

Chapter 3

Results

In this section, the obtained results from evaluating the models are discussed. First, for each of the models, the optimal parameters retrieved from running parameter optimization are provided. Then, the best results for each of the models (after training on the feature sets the models performed best on) are visualized and analyzed using confusion matrices. Afterwards, the performances of each of the models on the different feature sets will be compared.

3.1 Optimal parameters

During the training phase of the models, the set of parameters for each of the models was optimized through grid search and cross-validation. These parameters were then used in predicting the test sets and are given in table 3.1. After parameter tuning and training, the SVM model performed best when trained on the tf-idf feature set. It returned the best results when the regularization parameter C was set to 1 and a linear kernel was used, while the rest of the parameters were kept to their default values configured by the sk-learn library. As discussed in the section on the performance of the SVM model, a second SVM model with balanced class weights was created, trained en evaluated as well. It performed best when fitted to the concatenated feature set with its regularization parameter set to 5 and the usage of a linear kernel.

Model	Parameters		
KNN	metric='euclidean'	neighbors=19	weights='distance'
SVM unbalanced	C=1	kernel='linear'	
SVM balanced	C=5	kernel='linear'	

Table 3.1: Optimal parameters of models (trained on feature sets that returned the best results)

The KNN model performed optimally after training on the mixed feature set when the n_neighbors parameter, the weighting parameter and the metric parameter were set to 19, 'distance' and 'euclidean', respectively. Since the gaussian NBC model did not have any parameters to be optimized, this model was not used for parameter tuning (and thus is not present in table 3.1). Note that prior to the analysis of each of the models' performances on the mixed feature set, tf-idf feature set or the concatenated feature set, the parameters were retuned for optimal results since the number of features might influence which set of parameters performs best.

3.2 Performance of KNN

The first model trained and tested was the KNN model. After fitting the model to the mixed feature set and predicting the labels of the test set, evaluation of the model returned an accuracy of 0.55, a precision score of 0.54 and an F1-score of 0.54 (see table 3.2, which also shows the baseline scores in the top row). These scores all indicate that the KNN model classified around half of all data points in the test set correctly. However, the model fails to get a high recall score which could reveal that the model is not good at differentiating the high priority class. Besides, the baseline accuracy score is very high (score of 0.46) which indicates that if the model would assign all test data points the majority class label (intermediate popularity), the model would still correctly classify half of the data points. The confusion matrix in figure 3.1 visualizes how all of the data points are classified by the KNN classifier. The different shades of blue represent how many of the data points are classified into a certain class. Darker shades of blue indicate higher amounts of data points that are classified into a class and the diagonal of squares represents correct classifications.

It can be easily seen that most of the data points belonging to the low popularity or intermediate popularity categories have been given the correct labels. Also, both of these categories seem to be mistaken for each other often. In contrast to majority classes, not a lot of points (from all of the classes) are classified into the high popularity class. In fact, the KNN model almost does not classify any of the highly popular tracks correctly. This is congruous with the low recall score. Although these scores are not good, they are slightly better than the baseline scores. It might be worth looking into the possible causes of the low scores on all metrics so that the performances of the models could be improved.

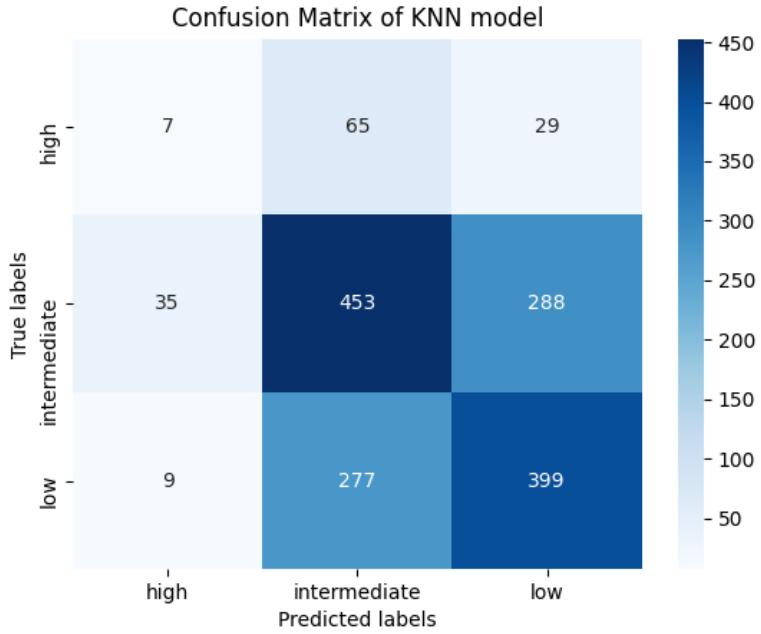

Figure 3.1: Confusion matrix for KNN model trained on the mixed feature set

3.3 Performance of SVM

The second model that was evaluated was the SVM model, which reached the best accuracy of 0.58 on the mixed feature set. The model also obtained a precision score of 0.59 and an F1-sore of 0.60. Although the overall performance of this model accomplishes the best classification results from all of the models (including the baseline performances) trained on all the different feature sets, these scores are still not high, as less than two out of every three data points are classified correctly.

By looking at the visualized results in the confusion matrix in figure 3.2 it can be recognized that, like the classifications that are done by the KNN model, a major part of the data points have been classified as intermediately or highly popular, whereas very little data points have been classified into the high popularity group. The SVM does also seem to make the same intermediate-low popularity mix-up but tends to classify more intermediate tracks correctly without negatively affecting the amount of correctly classified low popular tracks very much. This could account for the higher accuracy, precision and f1-score as the majority of the dataset is made up of tracks labelled as intermediate popular. It does not fully explain the surprising relatively high recall score of 0.60, as again none of the highly popular songs has been grouped into the right class. It would be useful to identify what exactly causes the low recall score and the mix-up between the two majority classes.

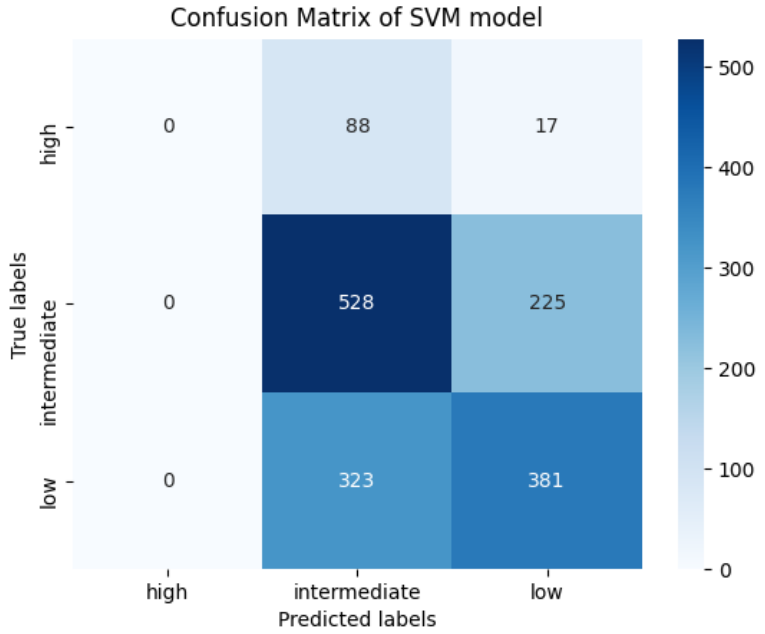

Figure 3.2: Confusion matrix for SVM model with unbalanced class weights, trained on the tf-idf feature set

After initially training the previously discussed SVM model, a parameter which could set the class weights of the labels to be balanced was discovered. According to the sk-learn documentation, this parameter should enable the SVM model to better deal with unbalanced datasets. To improve the results of the SVM, another SVM was build and then trained on all of the different feature sets with this parameter configured to be 'balanced'. Unfortunately, the scores of the original model could not be matched by the new 'balanced' model when trained on the mixed feature set and the tf-idf feature set. Surprisingly, however, the models' performance does exceed those of the other classifiers after training on the concatenated feature set. The results are visualized in figure 3.3.

The confusion matrix shows that changing the configuration of the class weights has led to a slight but promising increase in the amount of correctly classified minority class entries. Further examining what triggered this increase could contribute to the strengthening the model's performance.

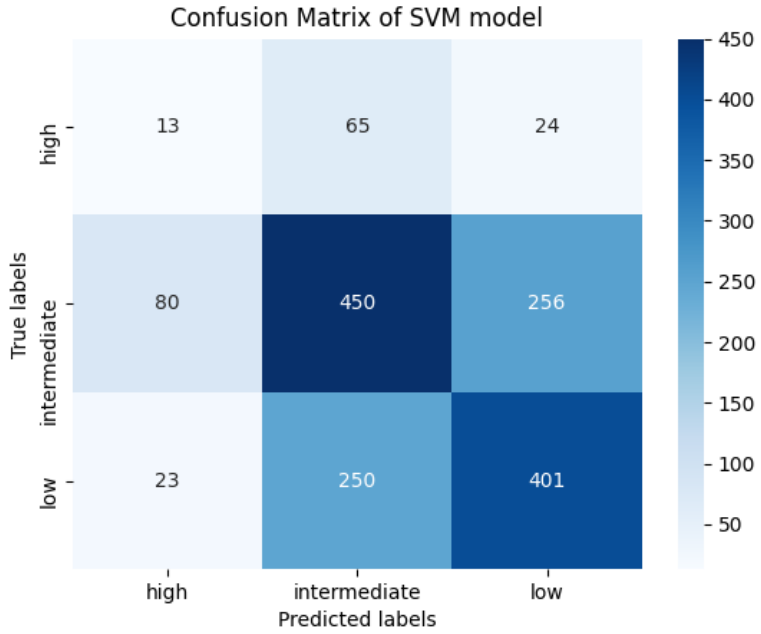

Figure 3.3: Confusion matrix for SVM model with balanced class weights, trained on the concatenated feature set

3.4 Performance of Naive Bayes Classifier

The last model to be evaluated was the Naive Bayes Classifier, which gained the best results after training on the mixed feature set. It got an accuracy score and a precision score of 0.53 and obtained an F1-score of 0.57. It also retrieved a recall score that is fairly high compared to the baseline performance. This indicates that more data points within some classes have been correctly classified without the total number of accurate predictions being significantly increased. Hence, there is still a lot of room for improvement.

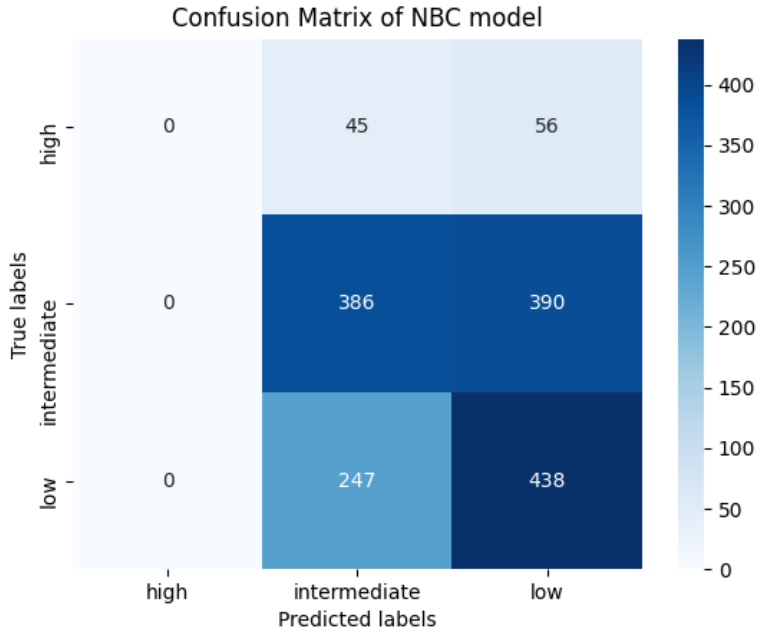

Figure 3.4: Confusion matrix for NBC model trained on the mixed feature set

The confusion matrix in figure 3.4 reveals that similar to the KNN model and the unbalanced SVM model, no data points were classified as highly popular, and most data points were classified as either low popular tracks or intermediate popular tracks. The same mixup can be noticed here as well since data points within these two categories are often mistakenly classified into the other class.

3.5 Performance of models on different feature sets

In addition to the mixed feature set, a large set of tf-idf features was computed as well. After evaluating the models on the mixed feature, set the models were trained and tested again twice, but instead, the models were fitted to the tf-idf feature set and a concatenated set of all features, to compare the impact of tf-idf on the performances of the models. The results are displayed in table 3.2 and for each of the feature sets, the best-obtained scores are highlighted. Also, in appendix B, these results can be found in tables that give a simpler overview for comparing the scores per metric.

By looking at this table it can be easily seen that although the KNN model has a slightly higher accuracy score, the unbalanced SVM generally performs better on the mixed feature set. From all the models, the NBC and the unbalanced SVM both have the highest recall scores. These models both seem to be more capable in distinguishing the minority class representing the highly popular songs. The unbalanced SVM also outperforms the other models across almost all metrics. Although the balanced SVM comes close to matching the

Feature Set	Model	Accuracy	Precision	Recall	F1-score
-	Baseline	0.46	-	0.33	-
Mixed Feature set	KNN	**0.55**	**0.54**	0.41	0.54
	NBC	0.53	0.53	**0.57**	0.54
	SVM unbalanced	0.54	**0.54**	**0.57**	**0.56**
	SVM balanced	0.37	0.46	0.40	0.30
Tf-idf Feature set	KNN	0.56	0.54	0.40	0.52
	NBC	0.31	0.53	0.39	0.37
	SVM unbalanced	**0.58**	**0.59**	**0.62**	**0.60**
	SVM balanced	**0.58**	0.58	0.43	0.58
Concatenated Feature set	KNN	0.54	0.51	0.38	0.52
	NBC	0.32	0.52	0.40	0.37
	SVM unbalanced	**0.58**	0.54	0.41	**0.56**
	SVM balanced	0.55	**0.56**	**0.43**	**0.56**

Table 3.2: Metric scores of each model fitted on different feature sets

best scores, it has a considerably lower recall score, which makes it a worse candidate for differentiating the highly popular tracks from the less popular tracks. Surprisingly, the balanced SVM exceeds the performances of the other models after being trained on the concatenated feature set. However, although it's recall score is better than those of the other models, a score of 0.43 is really low. It should be noted that none of the obtained scores by any of the models, on any of the feature sets, is particularly high. The highest obtained F1-score is 0.60, which is evidence that there still is a lot of room for improvement.

Chapter 4

Conclusion

This research aimed to compare different machine learning approaches and discover and find an answer to the following research question:

> *RQ1: Which machine learning approach is best capable of predicting music popularity based on lyrics of songs?*

To answer this question, three different types of classifiers were built, trained and evaluated: a KNN model, an NBC model and an SVM model. A public dataset of listening events and music information was used and annotated with popularity labels. Each of these classifiers was then trained on different feature sets which were extracted from the lyrics dataset and evaluated by computing accuracy, precision, recall and F1-scores. Finally, the performances of these models were compared to answer the main research question, as well as the following subquestion:

> *RQ2: What features can be extracted from music lyrics that could help predict the popularity of lyrics?*

To train the classifiers, lyrics were used to extract features. These features comprise of sentiment analysis features, a repetitiveness feature, an explicit feature and a large set of tf-idf features. Depending on what model is used, different combinations of different features result in different scores, but no model yielded very successful results. The KNN, NBC and SVM models were not able to sufficiently distinguish the high popular class from the less popular classes. The KNN and NBC model performed best on a feature set comprising only of the sentiment analysis, repetitiveness and explicit feature. Of all the models, the SVM performed best when trained on the tf-idf feature set.

Thus, after comparing the performances of all the different models, this study found that an SVM is best capable of predicting the popularity of a track when only tf-idf features are used but there is a lot of room for improvement, as it was not able to adequately predict whether a song is highly popular.

Chapter 5

Discussion

The results of this study showed that, to a certain degree, machine learning models can be learned to predict the success of a song based on lyrical features only. It showed that most of the models were able to predict the popularity of over half of the tracks correctly. However, the confusion matrices used for visualizing the results showed that none of these models was competent enough to identify the high popular songs. These bad outcomes could have different causes.

First of all, it is worth noting that the dataset used for this research is very skewed. Lesser popular songs are better represented in the dataset compared to songs that would appear in the charts. This could be the result of our best efforts of labelling the data. The labels could simply not correspond sufficiently enough with the actual popularity of the tracks (e.g. some intermediately popular songs maybe should have been labelled as high popular, or vice-versa). Instead of computing the labels by counting how many times a song has been streamed, maybe it would be more effective to include a unique listener count as well: popularity usually is not just defined by how many times a song has been listened to, but also by how many listeners have listened to that song. Besides false labels, the data set might not contain enough data for the models to learn how to identify hit songs.

Secondly, the dataset contains songs of different genres and lyrics of different genres could have different factors that contribute to a track's popularity. Although the genre of a song could be introduced as a feature for predicting the popularity of songs, genres are generally considered as track metadata and should not be included in the scope of this research as the aim is to test whether popularity can be predicted by using lyrics features only. Not filtering the dataset for a specific genre introduces another variable into the dataset that should not have been there. Because this extra variable was not used as a feature, it could have made predicting the popularity of tracks correctly a more difficult task for the models.

Additionally, filtering the LFM-1b dataset for tracks that are released more recently by using the Spotify popularity score introduces a lot of uncertainties. The popularity score is a black box as it is unknown how exactly these scores are calculated. It is known, however, that these scores are based on the number of streams of a track and how recent those streams were. This means that old hit songs that are still streamed often today receive a high popularity score, and

thus would not be filtered out of our dataset. Although discarding older tracks was a necessary step (because older tracks show different characteristics than modern songs and despite that they were hits once, the factors that made them popular do not contribute much to the popularity songs released today), using the Spotify popularity scores is not a reliable tool for the task.

Finally, the poor performance of the models could be due to the used features not contributing to the popularity of the tracks or the models overfitting to the data. Because the models' performances are better than randomly predicting the labels of the tracks, it is not expected that the features do not affect the popularity of the tracks. Therefore, this possible cause can be ruled out. Unfortunately, the results do not show whether the models could have overfitted to the data and this should be investigated in future studies.

Although it could be the case that the classifiers used in this research are not competent enough for this task, other works have shown more promising results with these models and it is, therefore, more likely that the bad outcomes are caused by one the underlying problems previously discussed. It would be valuable to investigate each of these different causes to improve the performances of these models so that these could be used in future works.

Chapter 6

Future Works

Looking back, this research could be improved and adapted to deliver better results in future studies. By devoting more attention on creating a balanced dataset, better and unbiased models could be trained. Such a dataset could be developed by adapting the dataset used for this research by undersampling the majority classes or oversampling the minority classes. Each track could be assigned a more accurate label that better represents its popularity through including the unique listener counts of tracks in the computation of a popularity score. Other ways of representing the popularity tracks are possible as well, but will not be discussed here. If machine learning approaches were to be applied for predicting the success of unreleased hits, it would be valuable as well to use a dataset containing more recent hits as the currently used dataset will most likely not reflect the characteristics of modern songs well.

Given the limited time available during this research, certain features were not implemented and are left to be included in future works: the complexity of the lyrics, presence of rhyme patterns, perspective (point of view used in lyrics). Of course, there is still a lot of ground to be covered when it comes to predicting hit songs based on metadata features, acoustic features, socio-economic features and lyrics features combined.

To contrast this work, unsupervised machine learning methods (for example, K-means) could be used for automatically grouping tracks based on their similarities. Analyzing the results of such a clustering task could then lead to new insights about what contributes to the success of a song.

Lastly, further research could be conducted into the prediction of songs within different countries, cultures and genres, as the popularity of songs is expected to be determined by very different factors within these fields.

Bibliography

[1] *The Role of a Record Company*. URL: https://powering-the-music-ecosystem.ifpi.org/.

[2] M. A. Casey et al. "Content-Based Music Information Retrieval: Current Directions and Future Challenges". In: *Proceedings of the IEEE* 96.4 (2008), pp. 668–696. DOI: 10.1109/JPROC.2008.916370.

[3] Dorien Herremans, David Martens, and Kenneth Sörensen. "Dance Hit Song Prediction". In: *Journal of New Music Research* 43.3 (2014), pp. 291–302. DOI: 10.1080/09298215.2014.881888. eprint: https://doi.org/10.1080/09298215.2014.881888. URL: https://doi.org/10.1080/09298215.2014.881888.

[4] Sumiko Asai. "Factors Affecting Hits in Japanese Popular Music". In: *Journal of Media Economics* 21.2 (June 2008), pp. 97–113. DOI: 10.1080/08997760802069895.

[5] Myra Interiano et al. "Musical trends and predictability of success in contemporary songs in and out of the top charts". In: *Royal Society Open Science* 5.5 (May 2018), p. 171274. DOI: 10.1098/rsos.171274. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5990848/.

[6] Önder Çoban. "Turkish Music Genre Classification using Audio and Lyrics Features". In: *Suleyman Demirel University Journal of Natural and Applied Sciences* (May 2017). DOI: 10.19113/sdufbed.88303.

[7] Ricardo Malheiro et al. "Emotionally-Relevant Features for Classification and Regression of Music Lyrics". In: *IEEE Transactions on Affective Computing* 9 (Jan. 2018), pp. 240–254. DOI: 10.1109/TAFFC.2016.2598569.

[8] Markus Schedl. "Genre Differences of Song Lyrics and Artist Wikis: An Analysis of Popularity, Length, Repetitiveness, and Readability". In: *The World Wide Web Conference*. WWW '19. San Francisco, CA, USA: Association for Computing Machinery, 2019, pp. 3201–3207. ISBN: 9781450366748. DOI: 10.1145/3308558.3313604. URL: https://doi.org/10.1145/3308558.3313604.

[9] Kahyun Choi et al. "Music subject classification based on lyrics and user interpretations". In: *Proceedings of the Association for Information Science and Technology* 53.1 (2016), pp. 1–10. DOI: https://doi.org/10.1002/pra2.2016.14505301041. eprint: https://asistdl.onlinelibrary.wiley.com/doi/pdf/10.1002/pra2.2016.14505301041. URL: https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/pra2.2016.14505301041.

[10] Markus Schedl. "The LFM-1b Dataset for Music Retrieval and Recommendation". In: *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*. ICMR '16. New York, New York, USA: Association for Computing Machinery, 2016, pp. 103–110. ISBN: 9781450343596. DOI: 10.1145/2911996.2912004. URL: https://doi.org/10.1145/2911996.2912004.

[11] Markus Schedl. "Investigating country-specific music preferences and music recommendation algorithms with the LFM-1b dataset". In: *International Journal of Multimedia Information Retrieval* 6 (Mar. 2017). DOI: 10.1007/s13735-017-0118-y.

[12] Michal Danilak. *langdetect*. Oct. 2016. URL: https://pypi.org/project/langdetect/.

[13] Spotify (2020). URL: https://developer.spotify.com/documentation/web-api/reference-beta/#endpoint-get-track.

[14] Paul Lamere. *Welcome to Spotipy! — spotipy 2.0 documentation*. 2014. URL: https://spotipy.readthedocs.io/en/2.16.1/.

[15] Fabian Pedregosa et al. "Scikit-learn: Machine learning in Python". In: *Journal of machine learning research* 12.Oct (2011), pp. 2825–2830.

[16] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[17] John D Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in science & engineering* 9.3 (2007), pp. 90–95.

[18] Michael Waskom et al. *mwaskom/seaborn: v0.8.1 (September 2017)*. Version v0.8.1. Sept. 2017. DOI: 10.5281/zenodo.883859. URL: https://doi.org/10.5281/zenodo.883859.

Appendix A

Figures

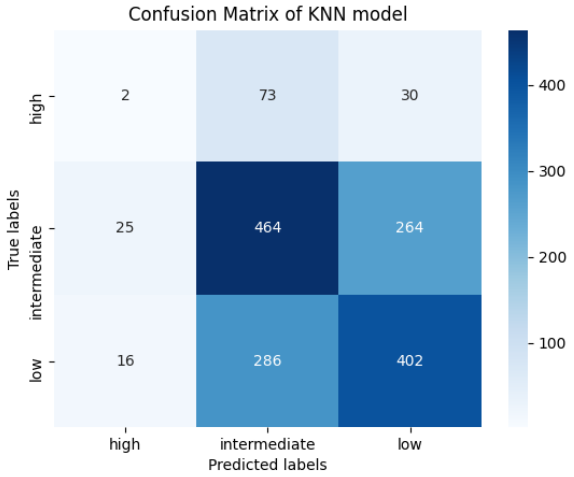

Figure A.1: Confusion matrix for KNN model trained on the tf-idf feature set

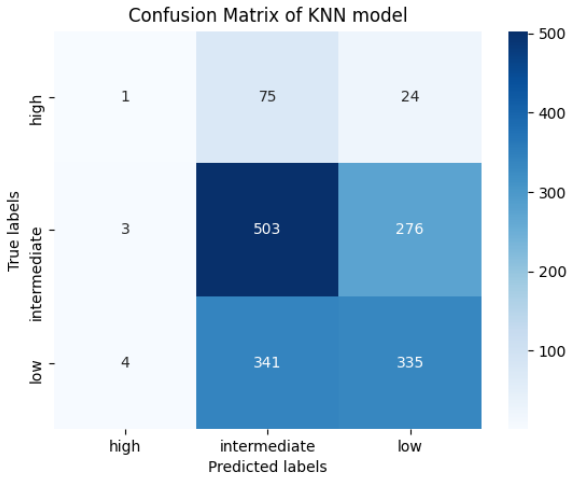

Figure A.2: Confusion matrix for NBC model trained on the concatenated feature set

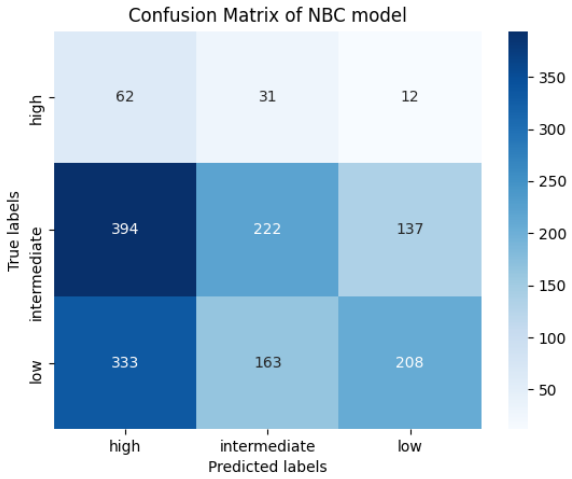

Figure A.3: Confusion matrix for NBC model trained on the tf-idf feature set

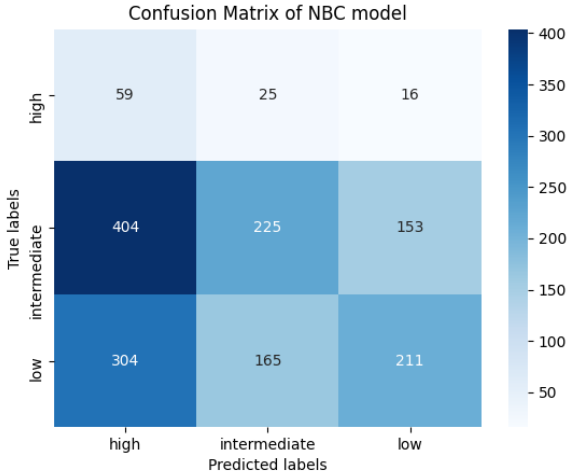

Figure A.4: Confusion matrix for NBC model trained on the concatenated feature set

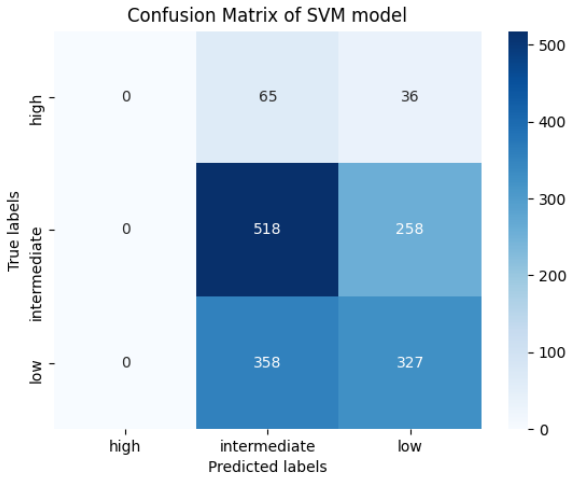

Figure A.5: Confusion matrix for SVM model with unbalanced class weights, trained on the mixed feature set

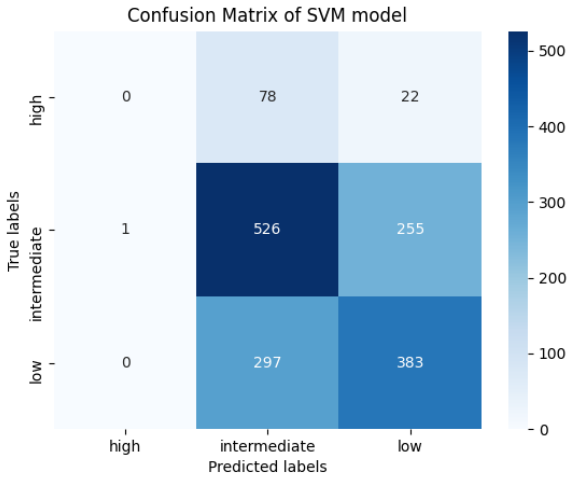

Figure A.6: Confusion matrix for SVM model with unbalanced class weights, trained on the concatenated feature set

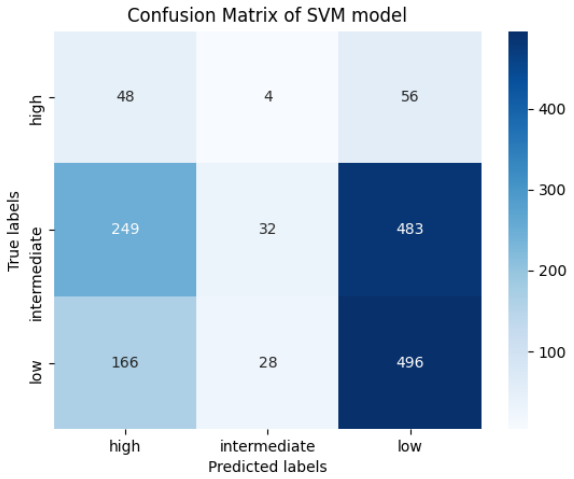

Figure A.7: Confusion matrix for SVM model with unbalanced class weights, trained on the mixed feature set

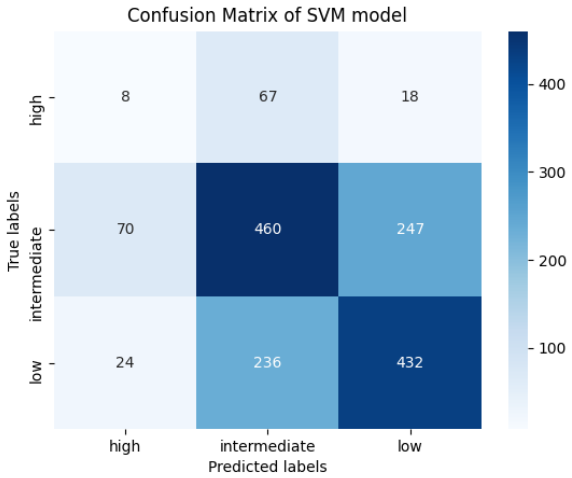

Figure A.8: Confusion matrix for SVM model with balanced class weights, trained on the tf-idf feature set

Appendix B

Tables

Model	Mixed Feature set	Tf-idf feature set	Concatenated feature Set
KNN	0.55	0.56	0.54
NBC	0.53	0.31	0.32
SVM unbalanced weights	0.54	0.58	0.58
SVM balanced weights	0.37	0.58	0.55

Table B.1: Accuracy scores of the models on different feature sets measured in accuracy

Model	Mixed Feature set	Tf-idf feature set	Concatenated feature Set
KNN	0.54	0.54	0.51
NBC	0.53	0.53	0.52
SVM unbalanced weights	0.54	0.59	0.54
SVM balanced weights	0.46	0.58	0.56

Table B.2: Averaged precision scores of the models on different feature sets

Model	Mixed Feature set	Tf-idf feature set	Concatenated feature Set
KNN	0.41	0.40	0.38
NBC	0.57	0.39	0.40
SVM unbalanced weights	0.57	0.62	0.41
SVM balanced weights	0.40	0.43	0.43

Table B.3: Unweighted averaged recall scores of the models on different feature sets

Model	Mixed Feature set	Tf-idf feature set	Concatenated feature Set
KNN	0.54	0.52	0.52
NBC	0.54	0.37	0.37
SVM unbalanced weights	0.56	0.60	0.56
SVM balanced weights	0.30	0.58	0.56

Table B.4: Averaged F1-scores of the models on different feature sets