Springboard Data Science Career Track - Final Project - Dec. 2020 cohort

Chart rank prediction using song lyrics and musical features

A) Introduction

1. Problem Statement

Most would agree that lyrics are an important part of pop songs. However, are they also predictive of the chart ranking of the song? And are they more or less predictive than the music itself? This project aims at answering these questions.

2. Background

Artists or record companies might want to predict sales of their newly produced songs or even produce songs using the proven characteristics of successful songs. In terms of musical features, research has shown what makes a successful song, for instance, top 100 ranked songs were more likely to be rated 'happy' and sung by a female voice. However, success in terms of lyrics has seldom been explored.

3. Goal

The goal consists in building a proof-of-concept model that classifies songs as lower or higher ranked better than chance. The objective is to determine whether conventional text classification algorithms are able to learn from song lyrics in terms of their chart ranking and have any predictive power in terms of classifying a song as higher or lower ranked. Different datasets that have undergone different preprocessing steps are used.

B) Data

1. Data Sources

First, I use lyrics of pop songs contained in the Billboard Year-End Hot 100 between 1965 and 2015 (5100 songs in total) by Kaylin Pavlik (link). Since the 1960s, the Billboard magazine has published rankings of popularity of pop songs in the US. The year-end Top 100 are based on sales, airplay, and more recently, streams. It is the most straight-forward

measure for commercial success of a song. Second, I use musical features for the corresponding songs obtained from the Spotify API. Spotify is a popular music streaming platform and provides 12 audio features for each featured track, including acousticness, liveness, speechiness and instrumentalness, energy, loudness, danceability and valence (positiveness), and descriptors like duration, tempo, key, and mode.

2. Data wrangling and preprocessing

2.1 Preprocessing of artist and song names (link):

As a first step, the artist and song names in the lyrics data of the total of 5,100 songs had to be transformed to a readable format for the Spotify API. Therefore, the following steps were performed:

1. All featuring artists were removed from the artist names
2. Concatenated words in song name strings were split using wordninja, a package that allows to probabilistically split concatenated words using NLP based on English Wikipedia unigram frequencies
3. A clean artist-song name string was created
4. Using requests, for each of the artist-song-name string musical features were retrieved from the Spotify API.
5. The artist-song-name strings of 63 songs that could not be found through the Spotify API were edited manually (for example, “los del r'io mac arena bayside boys mix” to “"los del rio macarena")
6. 16 songs could not be found on Spotify

2.2 Preprocessing of song lyrics (link):

The following steps are executed in order to preprocess the lyrics text data:

1. Drop 222 acoustic/instrumental songs
2. Remove punctuation/non-alphanumeric characters in the text with regular expressions
3. **Split strings** using wordninja
4. **Tokenization** using word_tokenize from nltk: splitting text into small units (tokens) necessary for classification algorithms “understanding” the text. I use unigrams (only one word per token)
5. **Lemmatization** using WordNetLemmatizer from nltk: grouping together the different inflected forms of a word so they can be analysed as a single item word stem).

6. **Removal of stopwords** using stopword from nltk.corpus: stopwords are a list of words that are very common but don't provide useful information for text classification. They are removed using a predefined list from nltk.corpus

2.3 Engineering of custom text features (link):

1. Proxy for word difficulty: Avg. number of words with more than 2 or 3 syllables
2. Avg. no. of **rhyming words** per song using the package Phyme
3. **Share of word classes** by song using part-of-speech (POS)-tagging (nltk.pos_tag): Word classes refer to nouns, verbs, adjectives, adverbs and many more subcategories of words
4. Other text features: No. of unique words, total no. of words

2.4 TF-IDF vectorization

TF-IDF, short for **term frequency–inverse document frequency**, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. The tf–idf value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general. tf–idf is one of the most popular term-weighting schemes today (Source: Wikipedia). The tf-idf values of each word are defined as as the product of the term frequency $tf(t,d)$ and inverse document frequency $idf(t,D)$:

$$tfidf(t,d,\ D) = tf(t,d) * idf(t,D) \text{ with } idf(t,D) = log\frac{N}{\sum D:t\in d1} \text{ and } tf(t,d) = \frac{f_{t,d}}{max\, f_{t',d}},$$

where $f_{t,d}$ is the number of times that term (word) *t* occurs in document *d* relative to the maximum frequency of the term *t'* with highest frequency in *d*. *N* is the total number of documents in the corpus and $\sum D: t \in d\ 1$is the number of documents *d* term *t* occurs in. After tfidf-Vectorization, the 1000 most important tf-idf features were selected using a Random Forest model.

2.5 Word2vec

The word2vec algorithm uses a neural network model to learn word associations from a large corpus of text and produces word embeddings using a shallow two-layer neural net. Word2vec takes as its input a large corpus of text and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space. Word vectors are positioned in the vector space such that

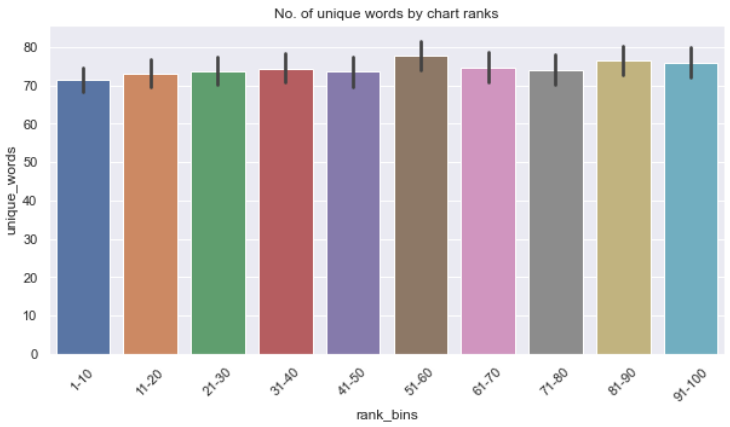
words that share common contexts in the corpus are located close to one another in the space. The vectors are chosen such that a simple mathematical function (the cosine similarity between the vectors) indicates the level of semantic similarity between the words represented by those vectors (source: Wikipedia).

I create word vectors with **300 dimensions** for each word in the songs using word2vec and then generate averages by song. These resulting song vectors can be used as inputs to the classification models. Using cosine similarity, I can also retrieve songs with most similar lyrics to a sample song:

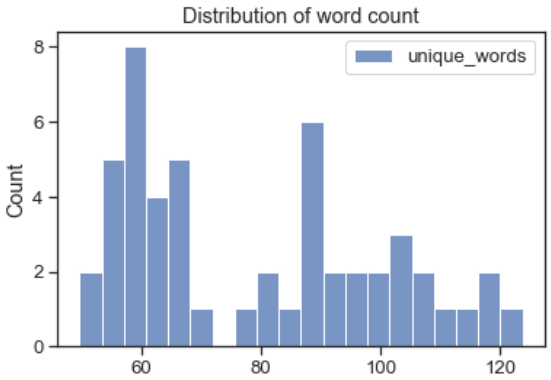
```
the 10 most similar songs in terms of lyrics to  missy elliott work it

[('missy elliott work it', 0.9999999999999996),
 ('mellow man ace men tiros a', 0.9484630016339509),
 ('das efx they want ef x', 0.9437145107964253),
 ('missy elliott gossip folks', 0.9427691397536225),
 ('gerardo rico suave', 0.9414519954109468),
 ('warren g this dj', 0.9400277009206248),
 ('chingy holi dae in', 0.9337144262071362),
 ('bobby shmurda hot boy', 0.9333345314117946),
 ('fergie fer ga licious', 0.9314863330805254),
 ('puff daddy cant nobody hold me down', 0.9307284347860835)]
```

3. Exploratory Data Analysis

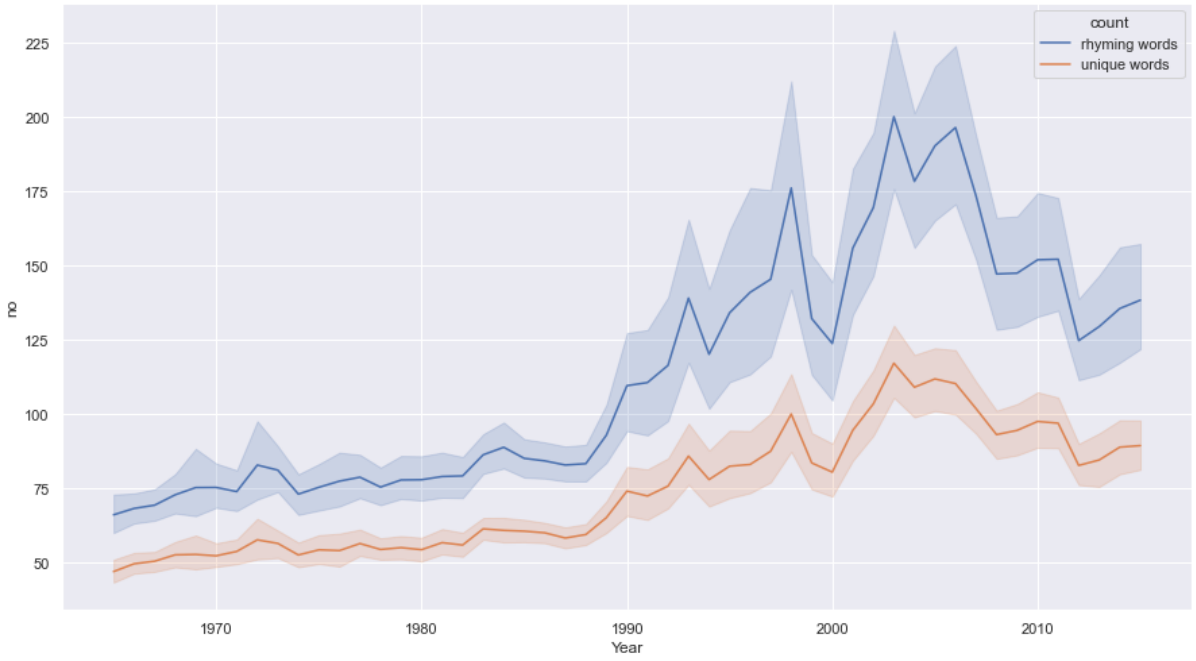
First, I explore the number of unique words per song by chart rank. From visual inspection, there is no clear association.



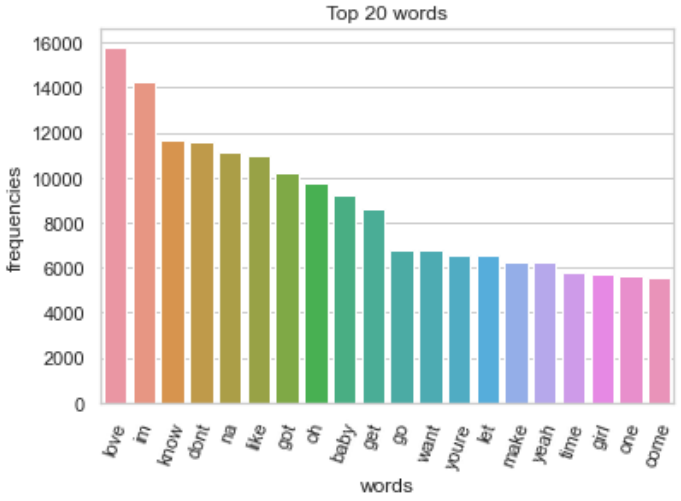
The distribution of words per song seems to be bimodal with peaks at around 60 and 90 unique words per song.

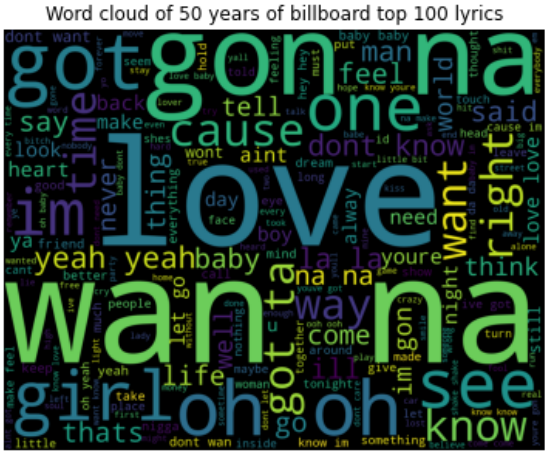


When looking at the no. of unique words by year, it becomes very evident that the number steadily increased from the 1950s around 50 words and peaked around the year 2002 at around 125 words per song. The pattern of the number of rhyming words over time follows that of the number of unique words, peaking in the beginning of the 2000s. Is this related to the growing popularity of rap music?



Finally, these are the most frequent words across all song lyrics. Unsurprisingly, “love” is the number one.





C) Modelling

1. Outcome to predict

Since the objective of this project is to determine whether anything can be learned from lyrics and/or music features in terms of higher or lower placement in the Billboard charts, two labels are generated: one for the top 50, one for the top 10. The intuition behind this is that we do not know at which level of popularity do lyrics become important - potentially they only seperate top 10 songs from the rest, not top 50 songs.

2. Model selection

I use 5 datasets: 1 dataset consisting in music features, 3 datasets including song lyrics features (with song lyrics that have undergone different preprocessing steps lined out above), and 1 dataset containing all features:

- tf-idf scores
- word2vec averages per song (song vectors)
- music features
- custom text features (POS tags, unique words, total words, rhyming words, difficult words, word length)
- all of the above

Then, I use the following models (with 3 fold cross validation/tuned hyperparameters and ROC-AUC as the evaluation metric)

Models and hyperparameter grids

Model	**Hyperparameter grid**
LogisticRegression	*GridSearch Cross-validation:* C = [0.00001,0.0001, 0.001, 0.01, 0.1,1,10,100,1000, 10000] penalty = ['l1','l2', 'elasticnet'] solver='liblinear
RandomForests	*RandomSearch Cross-validation:* n_estimators = [250,300, 350] max_features = ['auto', 'sqrt'] max_depth = [int(x) for x in np.linspace(10, 110, num = 11)]

	max_depth.append(None) min_samples_split = [2, 5, 10] min_samples_leaf = [1, 2, 4] bootstrap = [True, False]
XGBoosted trees (all data)	*RandomSearch Cross-validation:* gamma=[i/10.0 for i in range(0,5)] max_depth = [int(x) for x in np.linspace(3, 20, num = 3)] min_child_weight=range(1,6,2)
Support Vector Machines (SVM) (all data)	*RandomSearch Cross-validation:* C= [0.1, 1, 10, 100, 1000] gamma=[1, 0.1, 0.01, 0.001, 0.0001] kernel=['rbf']

Models and tuned hyperparameters

Models	**Hyperparams.**	**Datasets**				
		word2vec	***tf-idf***	***custom***	***music***	***all***
Logistic regression	C	1e-05	100	0.01	10,000	1,000
	penalty	l2	l2	l1	l1	l2
Random forest	n_estimators	250	250	300	250	300
	min_samples_split	5	10	5	5	2
	min_samples_leaf	4	2	4	4	1
	max_features	auto	auto	auto	auto	auto
	max_depth	30	None	70	10	90
	bootstrap	False	True	True	True	True
XGBoost	min_child_weight	-	-	-	-	5
	max_depth	-	-	-	-	11
	gamma	-	-	-	-	0.1

SVM	C	-	-	-	-	10
	gamma	-	-	-	-	0.1
	kernel	-	-	-	-	rbf

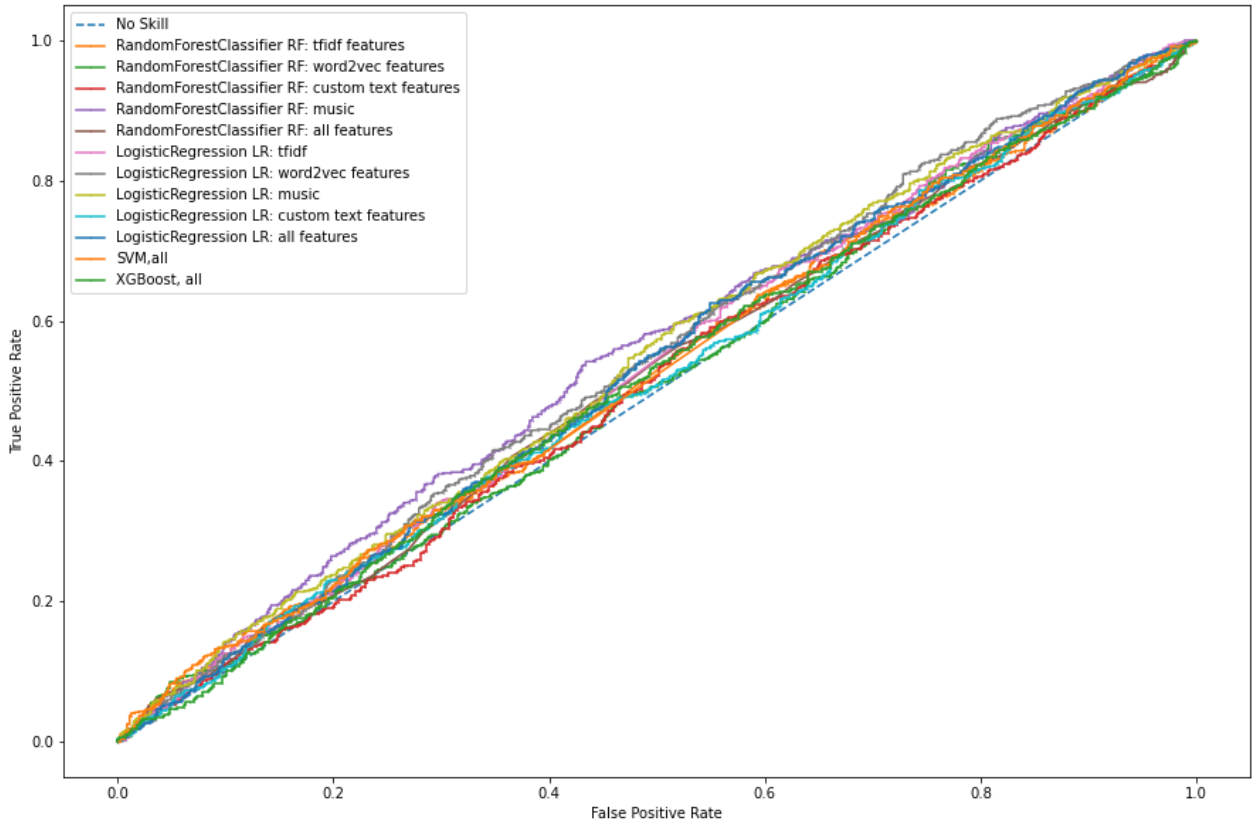
3. Model evaluation

A suitable performance measure for this type of classification problem consists in the Receiver Operator Characteristic-are under curve (ROC-AUC). The ROC is a probability curve that plots the true positive rate of the model (TPR) against the false positive rate (FPR) at various threshold values. The Area Under the Curve (AUC) is a summary of the ROC curve and measures how well the classifier is able to distinguish between classes. An excellent model with near perfect separability between classes has ROC-AUC near 1, and a model with no class separation capacity has a ROC-AUC of 0.5 (it predicts the positive class as well as pure chance). The criterion for success for this project is to find a model that separates higher and lower ranked songs better than chance, that is, with an ROC-AUC score>0.5. This cutoff may be well below what is needed for a successful product, but since the aim is to test a proof-of-concept-model that is better than chance.

Performance evaluation

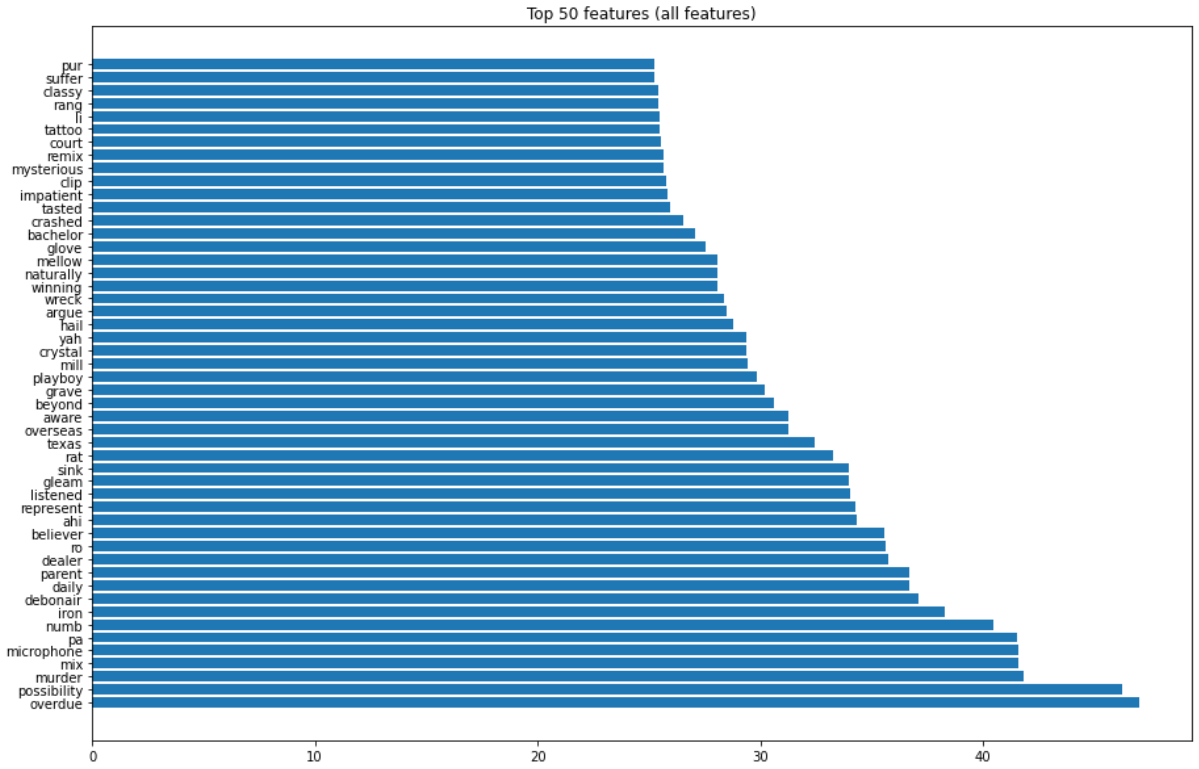
Model	**Dataset**	**ROC-AUC score**
Logistic regression	word2vec	0.542
	tfidf	0.532
	custom text	0.517
	music	0.544
	all	0.532
Random forest	word2vec	0.511
	tfidf	0.522
	custom text	0.511
	music	0.551
	all	0.521
XGBoost	all	0.518
Support vector machines	all	0.525

ROC Curve

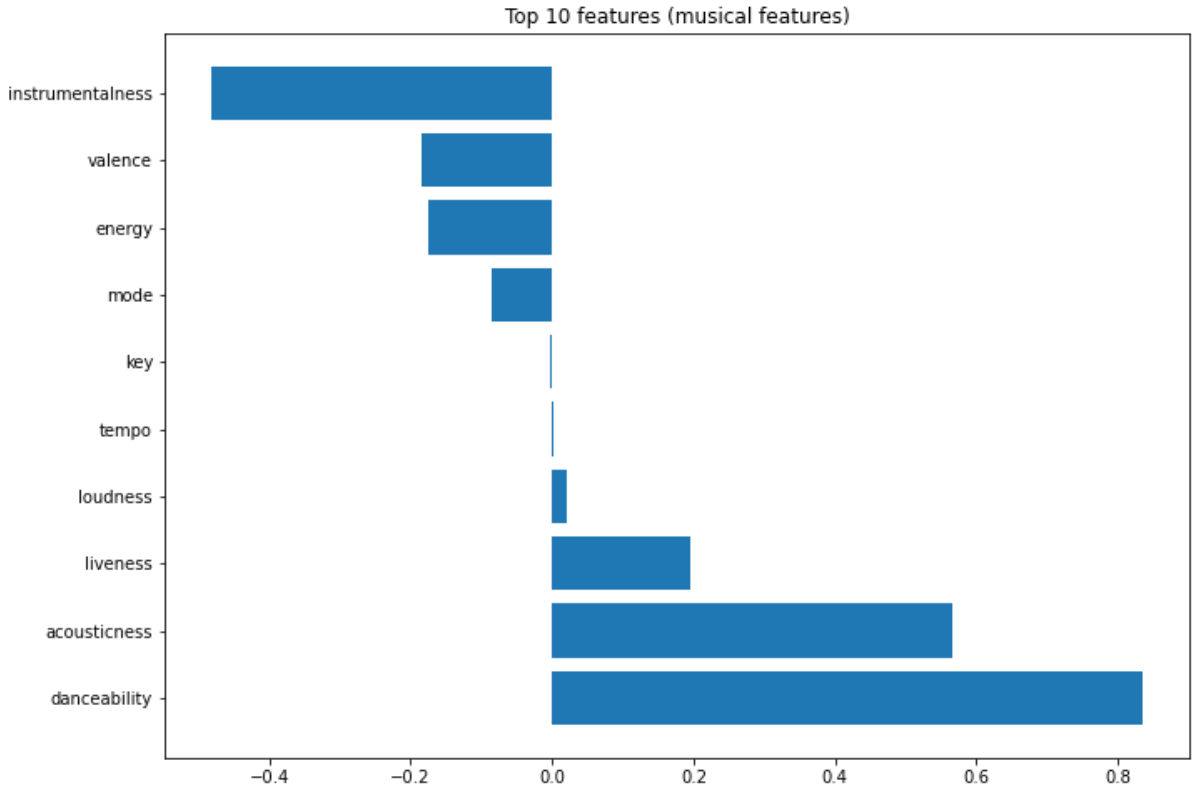


4) Feature Importance

Here I plot the top feature importances for the tuned Logistic Regression models in terms of (1) all features and (2) only musical features. The top 50 in terms of all features consist only of tf-idf scores for particular unigrams.



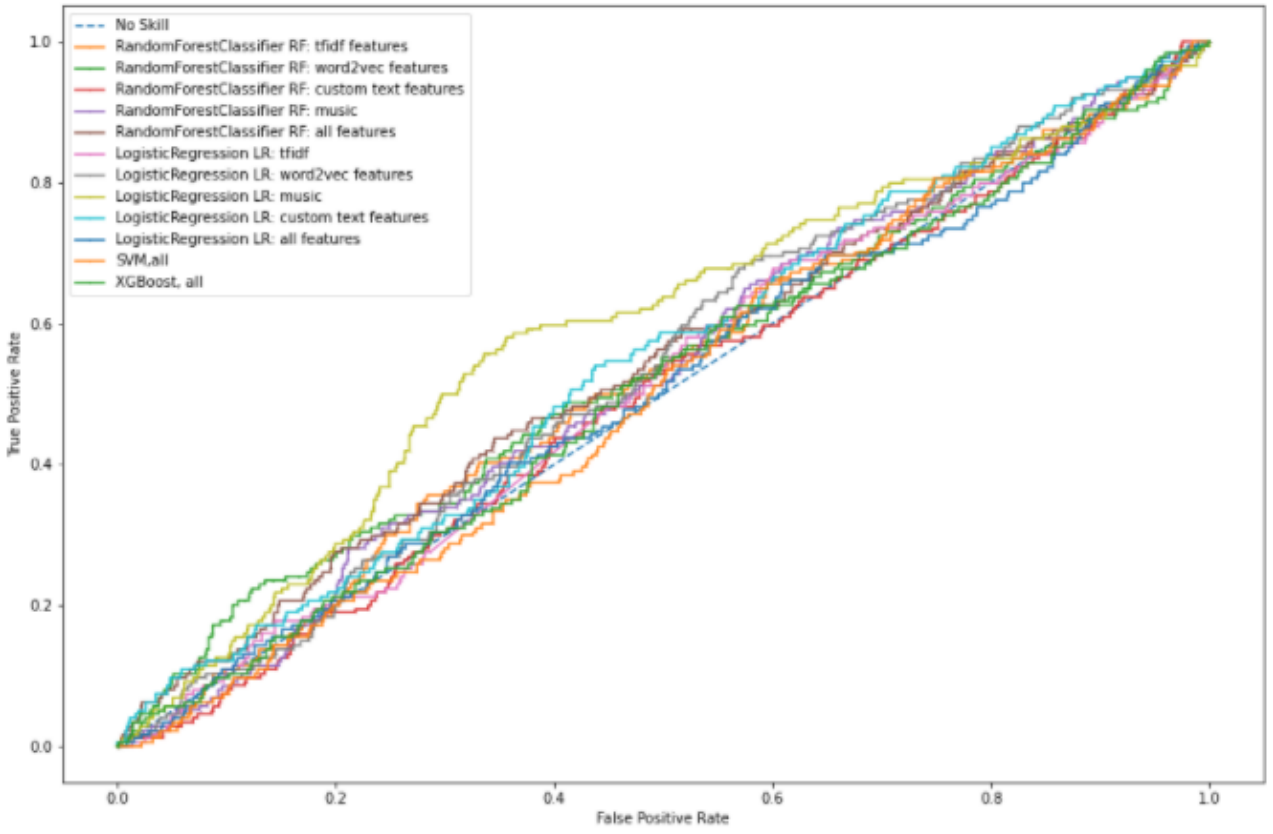
Looking at the top 10 music features shows that songs with higher danceability and acousticness are ranked higher, while songs with high valence and instrumentalness are ranked lower.



5) Alternative Outcome

When an alternative outcome is modelled, namely the top 10 songs, performance metrics do not change substantially, except for the Logistic Regression model with only music features with an ROC-AUC score of 0.587 instead of 0.544.

```
Model=RandomForestClassifier; Data=RF: tfidf features; ROC AUC=0.524
Model=RandomForestClassifier; Data=RF: word2vec features; ROC AUC=0.534
Model=RandomForestClassifier; Data=RF: custom text features; ROC AUC=0.499
Model=RandomForestClassifier; Data=RF: music; ROC AUC=0.527
Model=RandomForestClassifier; Data=RF: all features; ROC AUC=0.539
Model=LogisticRegression; Data=LR: tfidf; ROC AUC=0.513
Model=LogisticRegression; Data=LR: word2vec features; ROC AUC=0.536
Model=LogisticRegression; Data=LR: music; ROC AUC=0.587
Model=LogisticRegression; Data=LR: custom text features; ROC AUC=0.543
Model=LogisticRegression; Data=LR: all features; ROC AUC=0.505
Model=SupportVectorMachines; Data=SVM: all; ROC AUC=  0.503304294369353
Model=XGBoost; Data=XGB:all; ROC AUC=  0.5079553819149538
```



D) Conclusion and future research

The overall classification capacity of the tested models and preprocessing steps is very limited, though with ROC-AUC scores of >0.5 slightly better than pure chance. Tfidf, word2vec and custom text features do not yield substantially different performance in terms of separating higher and lower ranked songs. It is unclear whether lyrics in the Billboard top 100 just don't differ that much from each other, or whether the models or preprocessing steps are inadequate. One potential solution would be to get more lyrics data for songs not ranked in the BIllboard top 100 - maybe unranked lyrics are substantially different from top 100 ranked song lyrics. With more data, the application of neural nets would be another

option. Furthermore, a more advanced language representation model to create word embeddings such as BERT (Bidirectional Encoder Representations from Transformers) could be applied to generate features for classification.