Million Song Project

Group 3, Final report

Project statement

The project aim is to predict the song hotness - or 'song hotttnesss' [sic] - based on its features (tempo, loudness, segment data etc), and metadata (artist information, year of release etc).

Project background and literature review

This project is based on the Million Song Dataset (MSD) provided by labROSA at Columbia University.

The majority of record company profits are made from smash hits, and a great deal of money and effort is spent on seeking out the next success. Thus, predicting song hotness would be a useful way to test new songs and predict their commercial potential. To our knowledge, this aspect has not been extensively explored in the literature– previous analysis using the MSD has concentrated on predicting year, genre and providing recommendations to users.

Previous attempts to predict hit songs have had mixed success, possibly due to the unpredictability of cultural markets[1]. Although there are industry rules of thumb on how to write hit songs, there is comparatively little in the machine learning space to automatically detect hits. Early attempts used acoustic and lyric-based features to build support vector machines have not resulted in reproduceable successful models, which is explained by the fact that these features are not informative enough[2].

More advanced features that have been extracted in the MSD may provide a better basis for song hotness prediction and have been used in a small number of studies to predict song popularity. One of the most relevant is Pham et al[3], which used song characteristics consisting of both acoustic features and metadata from MSD. The authors applied a number of machine learning algorithms (SVMs, neural networks, logistic regression, Gaussian discriminant analysis, and linear regression) to predict whether a song is popular or not. When designed as a classification task, the best models achieved around 80% test set accuracy compared to a 75% baseline (labelling only the top 25% of songs as being popular). The best regression model achieved average error of 0.134 compared to a baseline of 0.159.

---

[1] Salganik et al (2006) Experimental Study of Inequality and Unpredictability in an Artificial Cultural Market https://www.princeton.edu/~mjs3/salganik_dodds_watts06_full.pdf

[2] Herremans et al (2014) Dance Hit Song Prediction http://antor.uantwerpen.be/wordpress/wp-content/papercite-data/pdf/herremans2014dance.pdf

[3] Pham et al (2015) Predicting Song Popularity http://cs229.stanford.edu/proj2015/140_report.pdf

Preliminary EDA

The full dataset of a million songs is around 270 GB. There is also a subset of 10,000 songs (around 2.5 GB) that we will use for the purposes of developing the prediction approach, as indicated in the project guidelines. Both datasets are provided as compressed HDF5 files.

In the main dataset are audio analysis features for each track, including the target variable of song hotness. This has values ranging from 0-1, and indicates the popularity of a song. It is assigned algorithmically based on various metrics, including news/blog mentions, play counts, music reviews, radio airtime and Billboard rankings.

There are also an additional 53 variables that include:

- Several **artist and track IDs** that can be used for linking with other datasets: The Echo Nest[4] ID, Music Brainz, 7digital and playme.com.
- **Echo Nest tags**, which include 'artist terms', 'similar artists'.
- **Music Brainz tags** include the fields 'year', 'artist mbtags' and 'artist mbtags count'. These tags are applied by humans and there are fewer of them compared to the Echo Nest ones, but where they do exist they tend to be cleaner.
- **Artist data,** such as location, 'hotttnesss' and familiarity
- **Song data,** such as tempo, key, duration, title
- **Track analysis data,** containing arrays of features that correspond to different segments or sections of the track. The main acoustic features are pitches, timbre and loudness, as defined by the Echo Nest Analyze API. The API provides these for every "segment", which are generally delimited by note onsets, or other discontinuities in the signal. The API also estimates the tatums, beats, bars (usually groups of 3 or 4 beats) and sections.

In terms of the data quality, we identified the following:

- `Song_hotttnesss` is missing for 43% of the rows in the subset. If these are removed, around 25% of the remaining rows have hotness values of 0
- Several columns are non-informative and can be dropped:
    - `Analysis sample rate:` is the same value for all tracks
    - `Energy` and `Danceability`: contain only zeros
    - `Artist latitude` and `longitude`: have more than 60% missing data
- `Year` has around 25% of rows with zero, which can be considered missing data. These are left as they are for now.

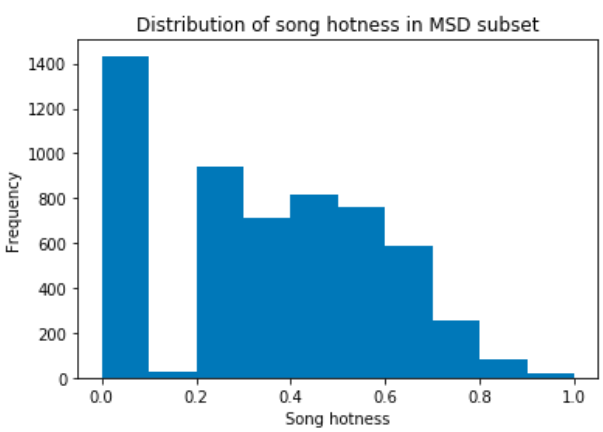
The data is accompanied by additional files that provide the following information:

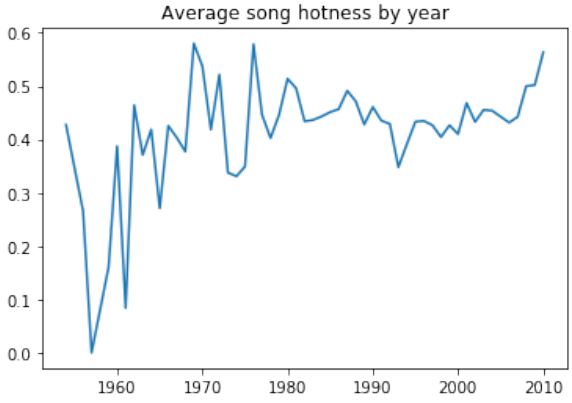
- List of track Echo Nest IDs
- List of all artist IDs
- List of all unique artist terms
- List of all unique musicbrainz tags
- List of the tracks for which there is year information

[4] Now owned by Spotify

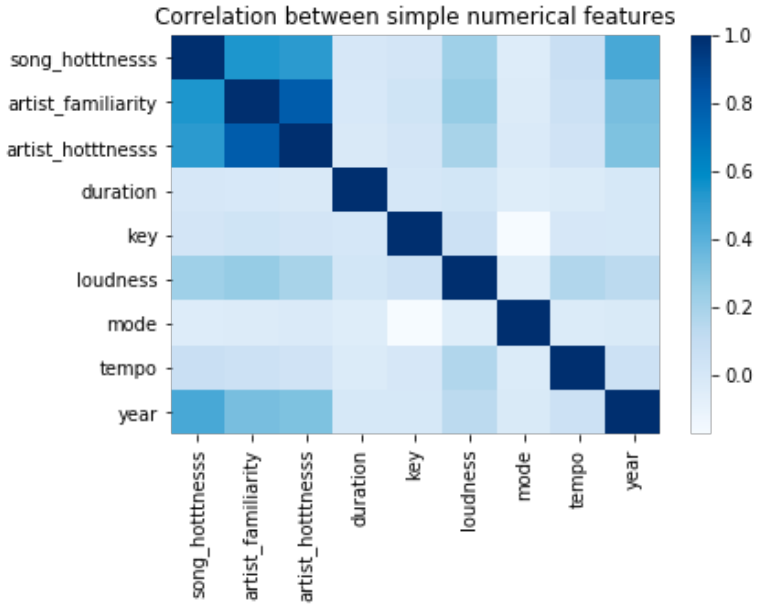
- List of artist latitude and longitude, where available
- Summary file of the whole dataset containing metadata but no audio analysis
- SQLite database containing track metadata
- SQLite database containing links from artist ID to tags
- SQLite database containing similarity among artists

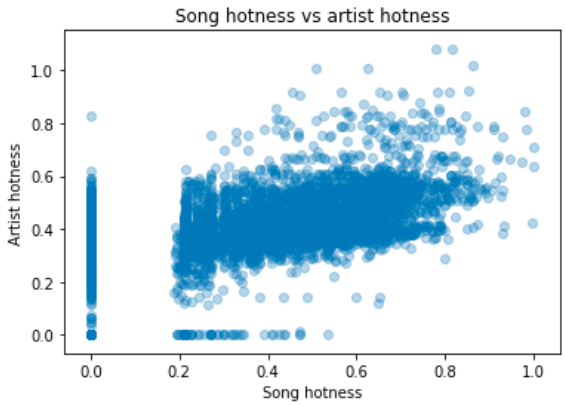
The distribution of song hotness is skewed with around 25% of the tracks having zero values (of the 5648 that do not have NaN values). Average song hotness per year seems to have a slight upward trend. We must also consider that musical tastes change over time.

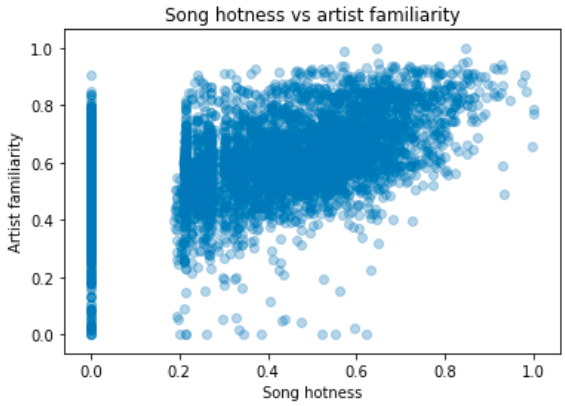




Exploration of correlations with the single-valued numerical columns shows that artist familiarity and artist hotness are strongly correlated with the target variable. As previously identified, there seems to be a positive correlation between song hotness and year, as well as with loudness.







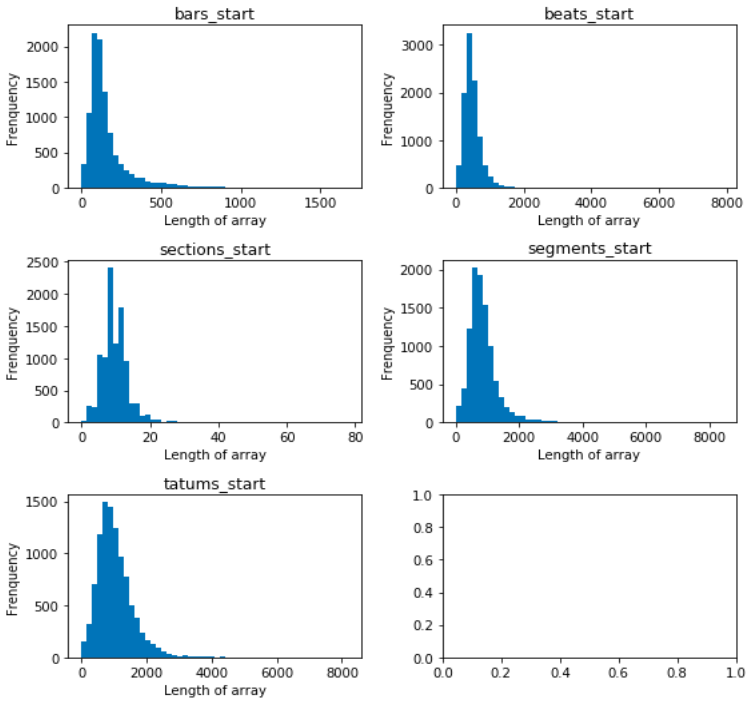
Several of the columns also contain numerical arrays of data for distinct parts of each song:

1. **Bars**: A bar (or measure) is a segment of time defined as a given number of beats.
2. **Beats:** A beat is the basic time unit of a piece of music; for example, each tick of a metronome. Beats are typically multiples of tatums.
3. **Sections:** Sections are defined by large variations in rhythm or timbre, e.g. chorus, verse, bridge, guitar solo, etc.
4. **Segments:** A set of sound entities (typically under a second) each relatively uniform in timbre and harmony.
5. **Tatums:** Tatums represent the lowest regular pulse train that a listener intuitively infers from the timing of perceived musical events (segments).

All of these parts have two columns associated with them:

- Start: the markers explained above, in seconds.
- Confidence: indicates the reliability of its corresponding attribute. Elements carrying a small confidence value should be considered speculative.

It is already clear from the descriptions that the arrays will be of different lengths since the length of the tracks and frequency of the measures will vary. Below are histograms showing the frequency of array lengths for the songs in the subset.

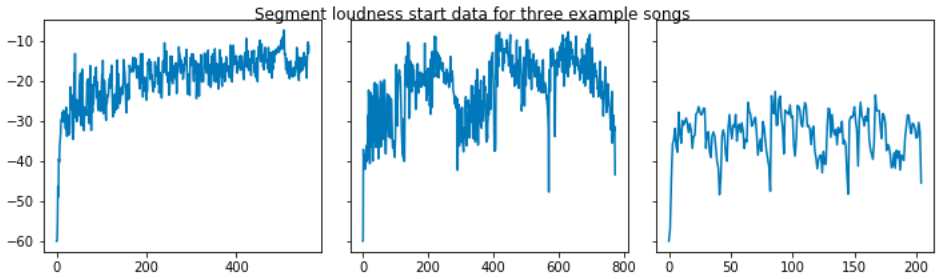


The most detailed information is for segments, which contain several other features:

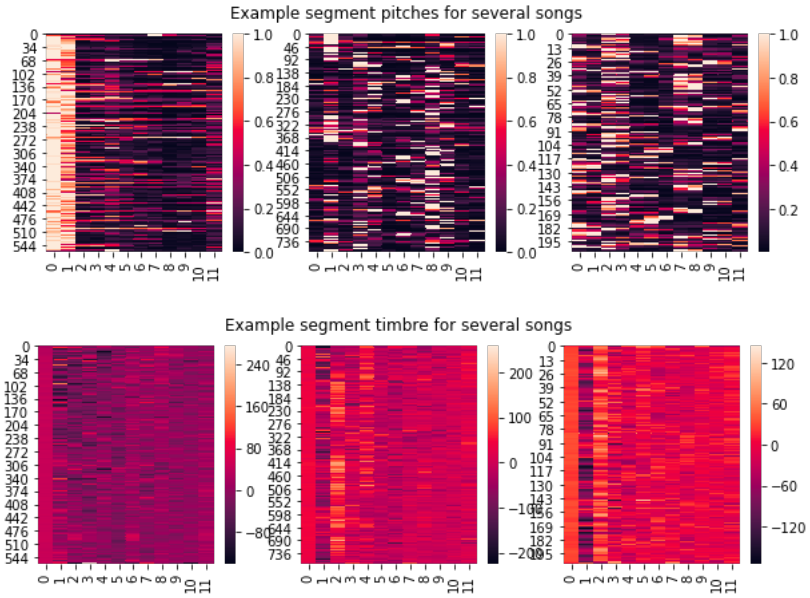
- Loudness_max: peak loudness value within the segment
- Loudness_max_time: offset within the segment of the point of maximum loudness
- Loudness_start: loudness level at the start of the segment
- Pitches: is given by a “chroma” vector, corresponding to the 12 pitch classes C, C#, D to B, with values ranging from 0 to 1 that describe the relative dominance of every pitch in the chromatic scale.
- Timbre: is the quality ('colour') of a musical note. It is what makes a particular musical sound different from another, even when they have the same pitch and loudness. The dataset represents timbre as a 12-dimensional vector of 'MFCC-like' features, each of which are unbounded values roughly centered around 0.

The number of segments per track is variable and each segment can itself be of variable length - typically they seem to be around 0.2 - 0.4 seconds but can be as long as 10 seconds or more.

Information on loudness at the start of each segment is shown below for three example songs. The outputs for loudness_max are similar.



The 12-dimensional arrays contained in the pitch and timbre columns are shown below for the same three example songs.



Genre and style information

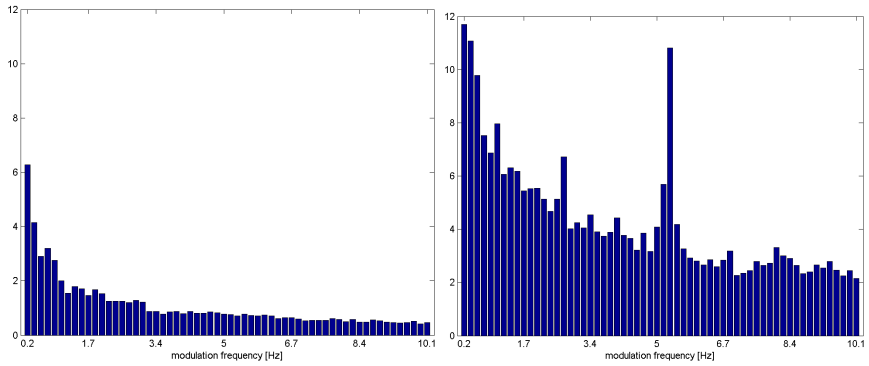
We started to explore whether the MSD can be augmented with information from other sources. We tried to merge the data with genre and style information for Vienna University[5] by joining on the track IDs. Unfortunately, the results were incomplete, with more than half of records having no genre information and around three-quarters having no style information.

Audio information

http://www.ifs.tuwien.ac.at/mir/msd/download.html#groundtruth provides data on audio-extracted information in ARFF format (https://www.cs.waikato.ac.nz/~ml/weka/arff.html). We started to look at rhythm histograms (http://www.ifs.tuwien.ac.at/mir/audiofeatureextraction.html#RH):

[5] Available at http://www.ifs.tuwien.ac.at/mir/msd/download.html

The Rhythm Histogram features are a descriptor for general rhythmics in an audio document. The magnitudes of each modulation frequency bin of all critical bands are summed up, to form a histogram of "rhythmic energy" per modulation frequency. The histogram contains 60 bins which reflect modulation frequency between 0 and 10 Hz. For a given piece of audio, the Rhythm Histogram feature set is calculated by taking the median of the histograms of every 6 second segment processed. E.g. classical vs. rock music:



ARFF format is difficult to read directly (version incompatibility) hence a direct CSV read is done. Match is done against track id. There are 60 measurements per track. The idea is to use these measurements as a feature to identify “hot” songs.

Statistical Spectrum Descriptor

The Sonogram is calculated as in the first part of the Rhythm Patterns calculation. According to the occurrence of beats or other rhythmic variation of energy on a specific critical band, statistical measures can describe the audio content. Our goal is to describe the rhythmic content of a piece of audio by computing the following statistical moments on the Sonogram values of each of the critical bands:

mean, median, variance, skewness, kurtosis, min- and max-value

Data is cleansed and pre-processed (AudioDataPrep.ipynb)

Modelling

We created two sets of models, treating song hotness as either a binary variable for classification or a continuous variable for regression. The classification problems defined a song as being “hot” if they were in the top 25% of the song hotness scores. The reason for choosing this threshold is that it seemed a reasonable trade-off between having enough positive examples and the fact that not many songs are truly hot in practice.

Baseline

We built a baseline model based on the features captured in the million song dataset as provided below. Please note that some of these features as noted earlier in EDA we removed from the data for building a model. Though we tested our model first with the subset, we finally trained our model on the entire data of million song to provide our results.

Features used for baseline model

Features from MSDS:

end_of_fade_in, start_of_fade_out, duration, loudness, tempo, key, key_confidence, mode, mode_confidence, time_signature, time_signature_confidence, artist_familiarity, artist_hotttnesss, year

Features for genre and style based on one hot encoding:

genre_Avant_Garde, genre_Blues, genre_Children, genre_Classical, genre_Comedy_Spoken, genre_Country, genre_Easy_Listening, genre_Electronic, genre_Folk, genre_Holiday, genre_International, genre_Jazz, genre_Latin, genre_New Age, genre_Pop_Rock, genre_Rap, genre_Reggae, genre_Religious, genre_RnB, genre_Stage , genre_Vocal, style_Big_Band, style_Blues_Contemporary, style_Country_Traditional, style_Dance, style_Electronica, style_Experimental,style_Folk_International, style_Gospel, style_Grunge_Emo, style_Hip_Hop_Rap, style_Jazz_Classic, style_Metal_Alternative, style_Metal_Death, style_Metal_Heavy, style_Pop_Contemporary, style_Pop_Indie, style_Pop_Latin, style_Punk, style_Reggae, style_RnB_Soul, style_Rock_Alternative ,style_Rock_College, style_Rock_Contemporary, style_Rock_Hard, style_Rock_Neo_Psychedelia

Features from rhythm:

rythm_0, rythm_1, rythm_2, rythm_3, rythm_4, rythm_5, rythm_6, rythm_7, rythm_8, rythm_9, rythm_10, rythm_11, rythm_12, rythm_13, rythm_14, rythm_15, rythm_16, rythm_17, rythm_18, rythm_19, rythm_20, rythm_21, rythm_22, rythm_23, rythm_24, rythm_25, rythm_26, rythm_27, rythm_28, rythm_29, rythm_30, rythm_31, rythm_32, rythm_33, rythm_34, rythm_35, rythm_36, rythm_37, rythm_38, rythm_39, rythm_40, rythm_41, rythm_42, rythm_43, rythm_44, rythm_45, rythm_46, rythm_47, rythm_48, rythm_49, rythm_50, rythm_51, rythm_52, rythm_53, rythm_54, rythm_55, rythm_56, rythm_57, rythm_58, rythm_59

Please note that we normalized numeric features before using them to train the model.

Removing null song hotness: As we noted earlier, we removed songs for which we didn't have song hotness available in the million song dataset. As a result, from 1 millon songs, after removing songs for which song hotness was not available, we were left with 581,965 songs.

Classifying song hotness: We decided to build a classification model that predicts whether song is popular or not. To do that, we classified song hotness which is a numeric value into binary value. We took top 25% songs as hot songs and classified them as a popular song. As a result, we had 145,406 songs classified as a popular song and 436,559 as not a popular song.

Train and Test split: We split the entire data into training and testing set. 70% of songs were sampled for training set and rest 30% songs were sampled in testing set.

Algorithms for baseline model: We tried following model algorithms to train and test our model that predicts whether a song is popular or not.

Classical algorithms

1- Logistic Regression
2- Naïve Bayes
3- Support Vector Classification
4- Random Forest

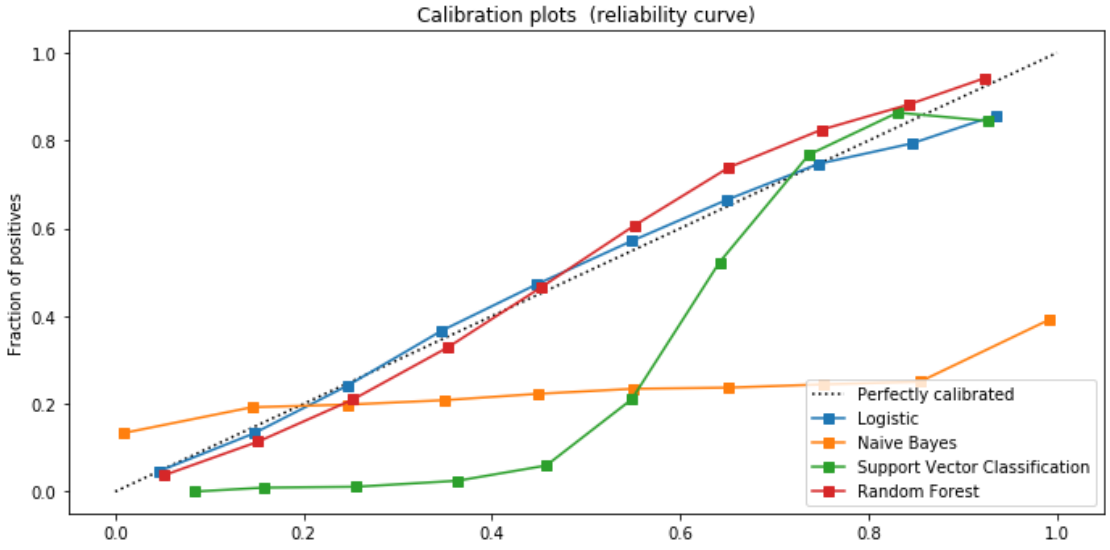
Neural Network

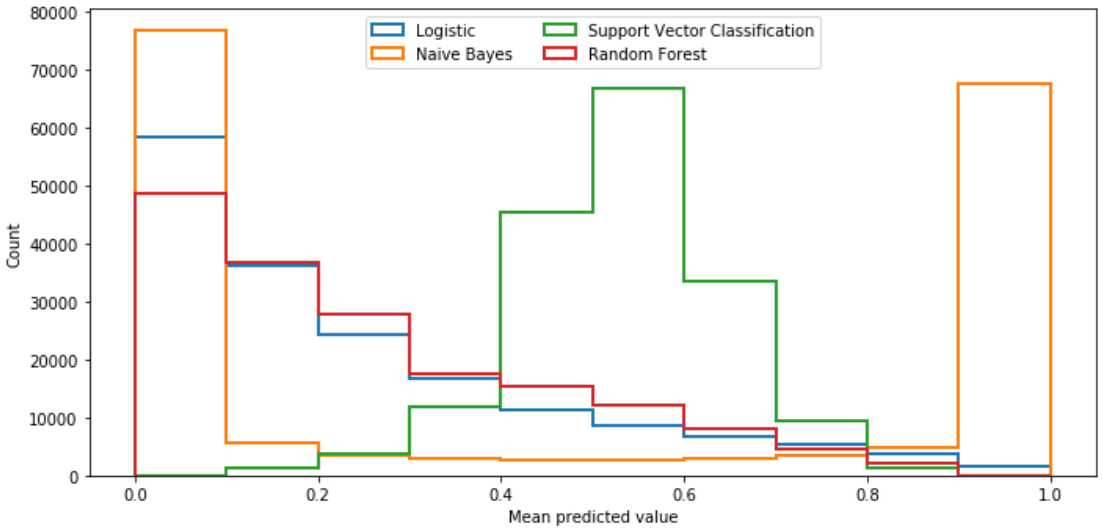
1- Multi layer perceptron

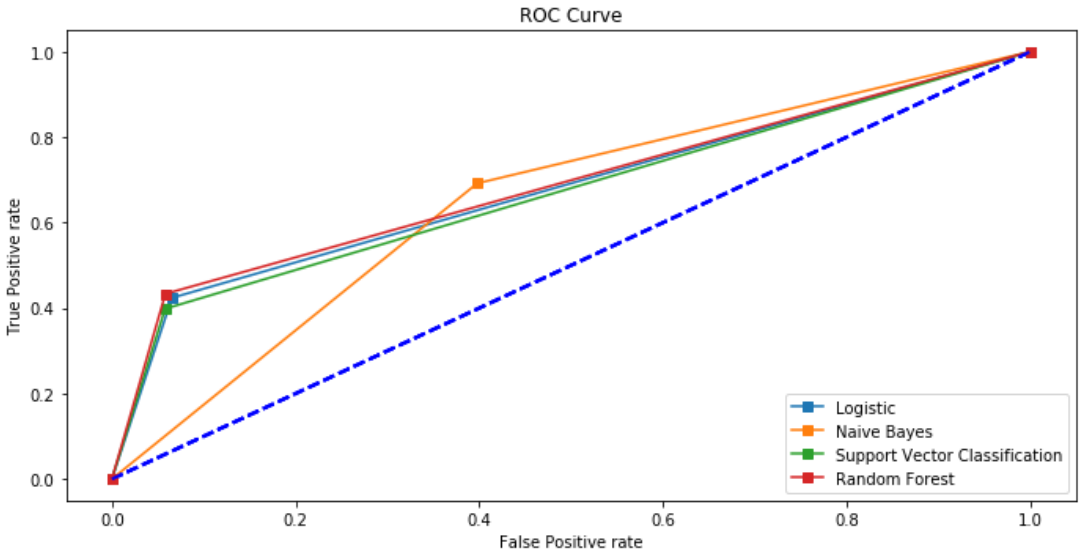
Measures for model assessment: We tested our models on model accuracy and auc (area under the curve) to ensure that our assessment is not biased because of unbalanced dataset.

Results for Classical algorithms:

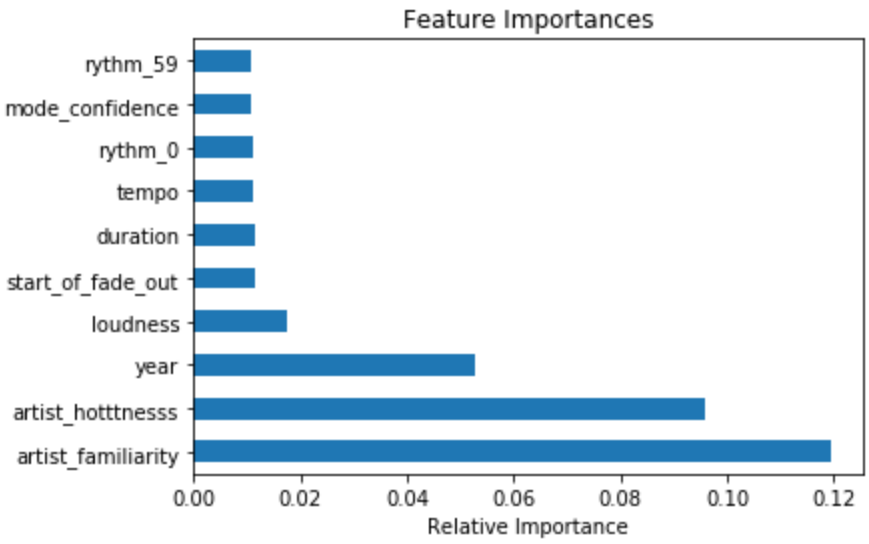
- Accuracy and AUC (test) for Logistic: **80.7%** and **0.76** respectively
- Accuracy and AUC (test) for Naive Bayes: **62.5%** and **0.61** respectively
- Accuracy and AUC (test) for Support Vector Classification: **80.7%** and **0.76** respectively
- Accuracy and AUC (test) for Random Forest: **81.5%** and **0.77** respectively





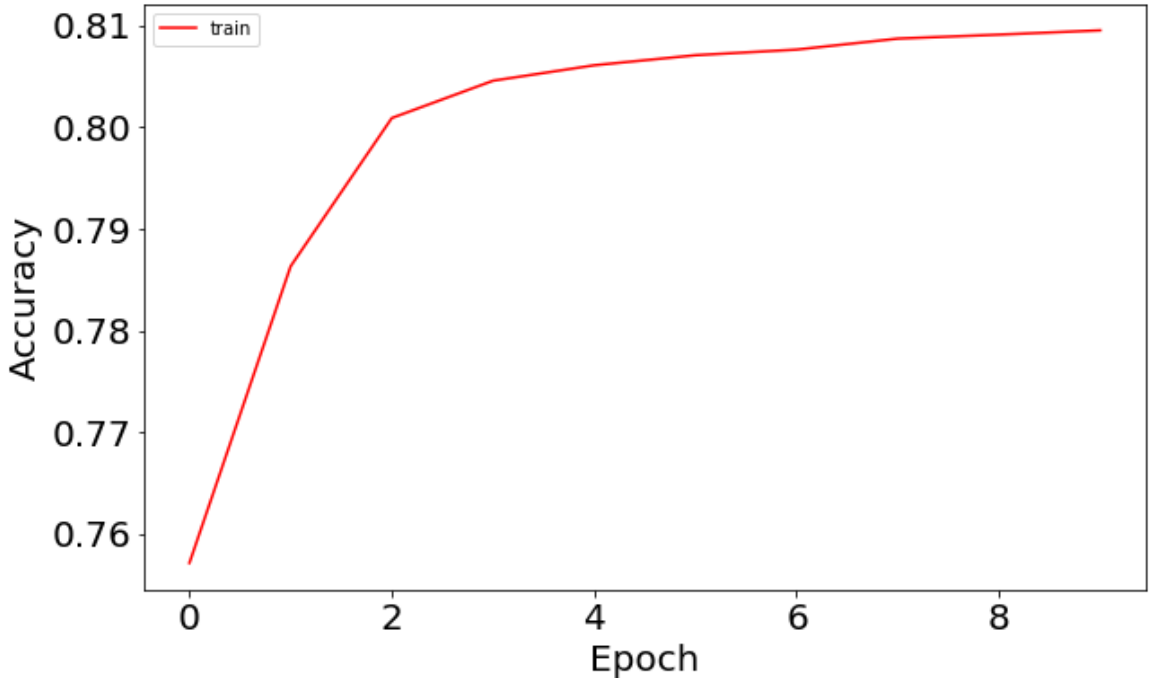


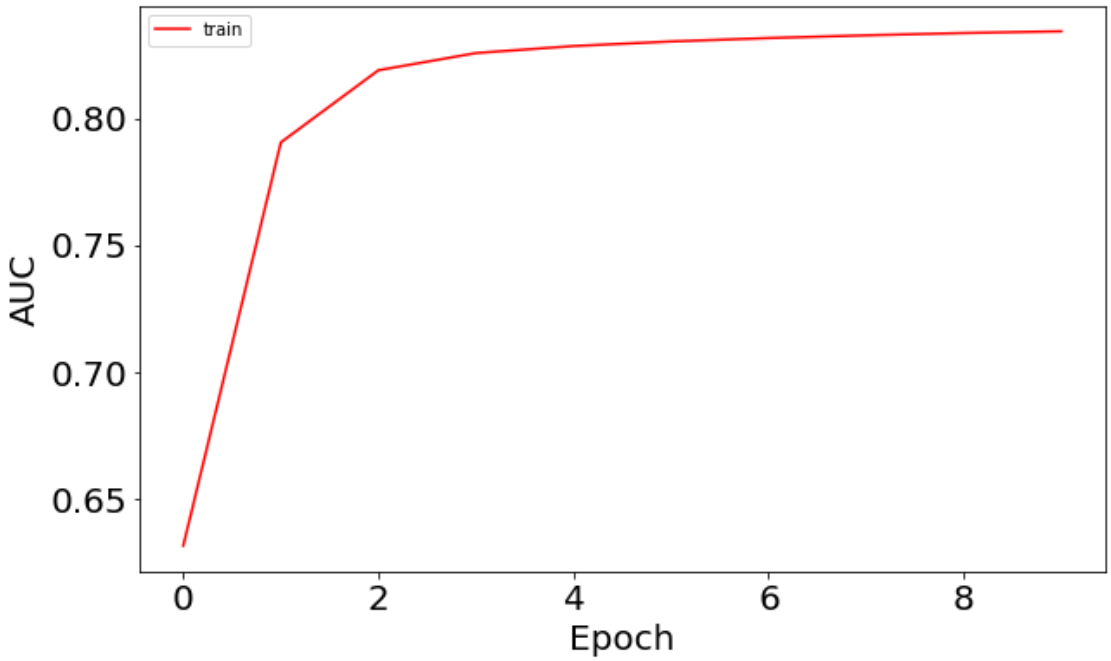
Feature Importance for Random Forest model (Random Forest provided the best accuracy and area under the curve)



Results for Neural Network

We plotted a 2 layer perceptron to train the model to predict song hotness. We received accuracy = **80.67%** and AUC = **0.77** on test data.





MLP provided similar results as RandomForest but RandomForest was still doing slightly better.

Audio features in the dataset arrays (ArrayAnalysis.ipynb)

As found in the EDA, the core dataset contained various audio features in numerical arrays. The most informative of these was the segment data, which included features as well as the time stamps.

The data were processed into a 26-dimensional vector for each segment, consisting of concatenated data in the following order:

- loudness at start of segment
- maximum loudness in each segment
- 12 timbre features
- 12 chroma features

The arrays were cropped by taking the first 350 segments and excluding tracks that don't have at least this number of segments. This should be enough to capture the main parts of each song.

Two models were trained on this dataset:

Model 1: CRNN network

The CRNN first passes the data through a series of convolutions before feeding to a Gated Recurrent Unit (GRU). Given the imbalance in the training data, the classes are balanced in the model fitting.

- It uses **1D convolution layers** that perform convolution operations just across the time dimension. This is to extract features from each time slice (rather than 2D convolution more frequently seen in image processing). Dropout and max pooling are used to prevent overfitting.
- The output of the CNN step is fed into a **GRU, which should find the short- and long-term structure of the audio features**. Recurrent neural networks (RNN) are used for understanding sequential data. They make a hidden state t dependent on the hidden state in the previous time step.

Model 2: Parallel CNN RNN network

The CRNN model above uses RNNs as a temporal summarizer, but the GRU uses the output of the CNNs. In a parallel CNN-RNN model, the architecture has the following structure instead:

- The signals are passed through 2D convolutions followed by max pooling and the final output is flattened. This is similar to image processing where the signals are now processed in 2D.
- Separately, the signals are passed through RNN
- The outputs of both steps are concatenated and passed to a dense network.

This approach allows us to preserve the temporal relationships of the original signals and use CNN and RNN in parallel.

Unfortunately, neither the classification nor regression version of these models yielded any improvement over the baseline.

Additional audio features (AudioModel.ipynb)

http://www.ifs.tuwien.ac.at/mir/audiofeatureextraction.html contains links to several datasets which were created from MSD audio features. Specifically, we looked at:

- Rhythm histograms: general rhythmics in audio document, stored as 60 bins which reflect modulation frequency from 0 to 10 Hz. Feature set is calculated by taking the median of the histograms of every 6 second segment processed (60 values per track)
- Statistical spectrum descriptors: according to the occurrence of beats or other rhythmic variation of energy on a specific critical band, audio content is described by statistical moments of critical bands (mean, median, variance, skewness, kurtosis, min- max-value) (168 vector of values per track)

These audio datasets were combined (following overall “collage” idea from https://www.pyimagesearch.com/2019/01/28/keras-regression-and-cnns/), using the preprocessing dataset (e.g. removal of song-track mismatches http://millionsongdataset.com/blog/12-2-12-fixing-matching-errors/ , restricting the track set to match the given subset of MSD).

CNN was created to attempt to find meaningful relationship between audio features and 'song_hotttnesss'. In this way, audio features represent a ‘picture’ on which CNN is trained.

- Input is reshaped to look like a ‘picture’, i.e. 228 per-track data values are reshaped to 12 x 19 ‘rectangle’
- Several sets of 2D convolution, activation (RELU) and max pooling filters (16, 32, 64)
- Single ‘channel’ is used as if it’s a monochrome ‘image’
- For regression, Linear activation is used in the last layer while for classification, Sigmoid is used

Unfortunately, regression version of this model hasn’t yielded any improvement over the baseline. Classification version of this model has done better than regression (~75%) but still worse than baseline.

Text to Topics to Hits (TextTopicHits.ipynb)

For some songs, there are lyrics available (https://labrosa.ee.columbia.edu/millionsong/musixmatch). To explore relationship of song words and hit topics, the following model was done:

- Combined dataset was created off Musixmatch
- Dataset was parsed to go from words and their counts (which was apparently done for copyright reasons) to “sentences” where each word appears as many times as in the original text
- Corpus and vocabulary mapping created
- Latent Dirichlet Allocation was executed on the full corpus to extract 4 topics, 10 words each
- Latent Dirichlet Allocation was executed on only hit songs (defined as top 75% quantile for ‘song_hotttnesss’ field) corpus to extract 4 topics, 10 words each for hit songs
- Words which appear only in hits and not in all songs were detected (‘hit words’)
- Prediction of a hit song was made based on presence of the ‘hit words’

Results fluctuate a bit from run to run but resulting accuracy is still low (~40%), i.e. this model did not yield any improvement over the baseline.

Lyrics data (ML_Model_with_Lyrics_all_data.ipynb)

We download lyrics data from musixmatch for million song dataset to see if song lyrics can help predict song hotness. Please note lyrics data was only available for 237,662 songs out of million songs in million songs dataset.

This data was downloaded in the form of bag of words containing 5000 unique words for 1 million songs.

Sample for lyrics data:

	track_id	mxm_tid	word	count	is_test
0	TRAAAAV128F421A322	4623710	i	6	0
1	TRAAAAV128F421A322	4623710	the	4	0
2	TRAAAAV128F421A322	4623710	you	2	0
3	TRAAAAV128F421A322	4623710	to	2	0
4	TRAAAAV128F421A322	4623710	and	5	0

We removed stop words (word captured in nltk English stop words and small words length <2 characters) from this dataset. We also downloaded unstemmed version (musixmatch/mxm_reverse_mapping.txt) of the words captured in lyrics data to make better sense of the words and replaced the stemmed words with unstemmed version of those words.

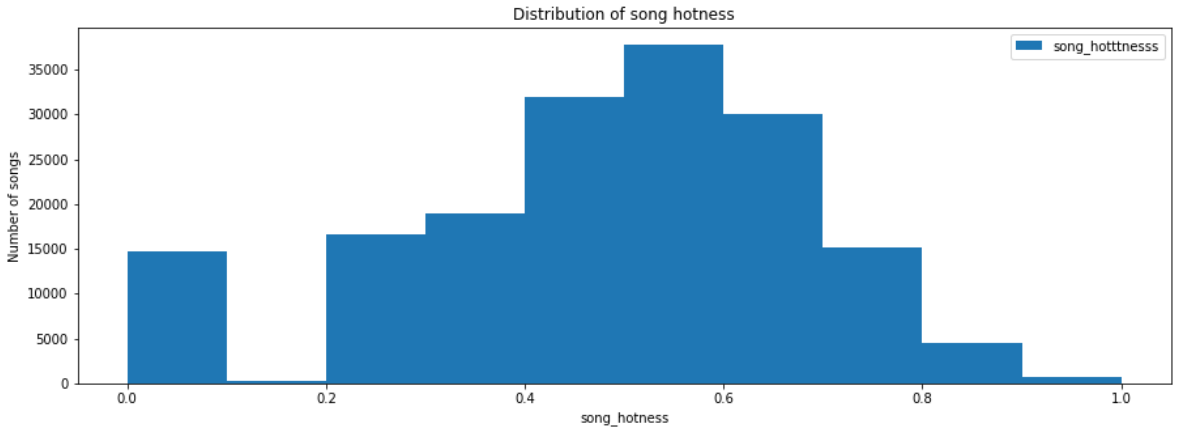
Word cloud of lyrics data:

Example for stemmed vs unstemmed words

	original	unstemmed
0	dÃa	dÃa
1	pido	pido
2	hatr	hatred
3	pide	pide
4	yellow	yellow
5	four	four
6	sleev	sleeve
7	sleep	sleep
8	thirst	thirst
9	upsid	upside

When we merged lyrics data with our main data and removed songs with missing hotness, we were left with 170,573 songs.

Distribution of song hotness for this final dataset was as below.



Approach to use lyrics data for predicting song hotness:

1- Perform LDA to find optimal number of topics

2- Fit LDA model for the optimal number of topics

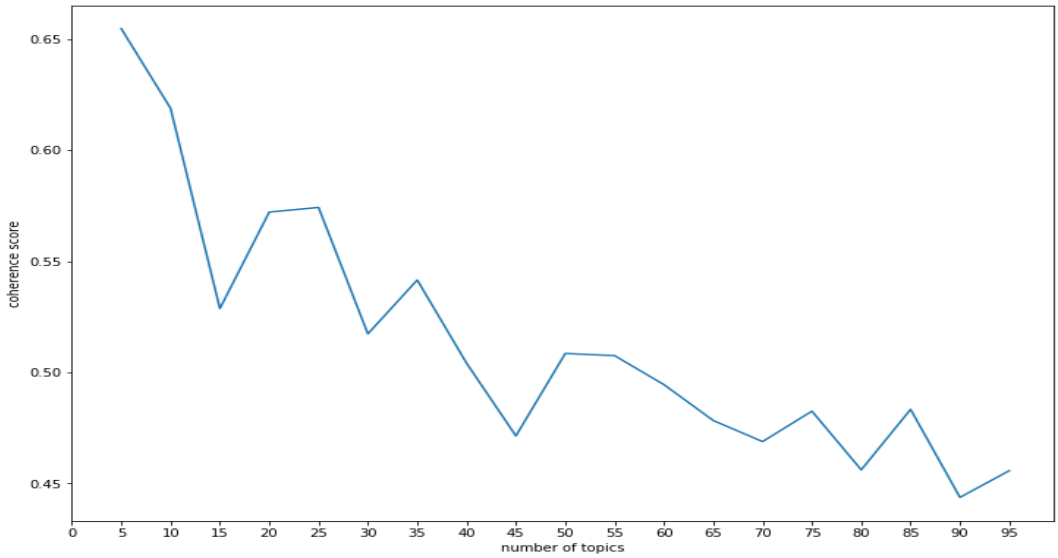
3- Encode top topic for each song as 1 and other topics as 0

4- Use Topics information *only* to predict song hotness

5- Use Topics information *in addition to* the features used in baseline model to predict song hotness

Please note that we took a sample of 25,000 songs to build LDA model because of memory limitations

Results: Chart for Coherence score for different number of topics

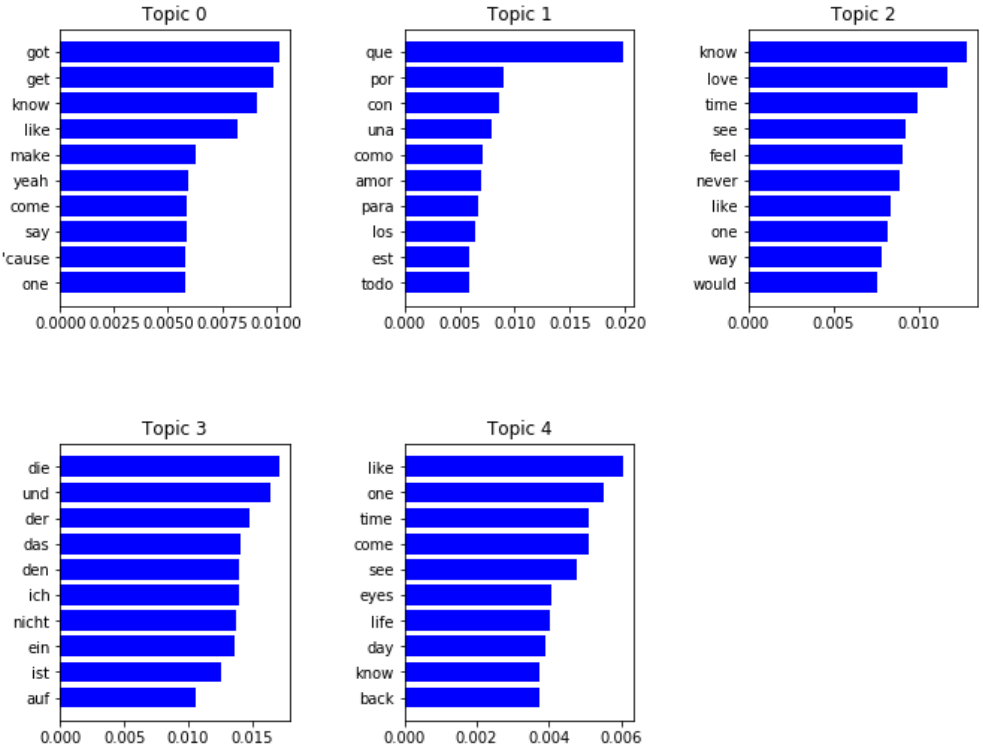


We concluded from the chart above that optimal number of topics was somewhere around 5. Hence, we selected 5 topics to fit our LDA model.

A sample of 15 words for each of the 5 topics is shown below.

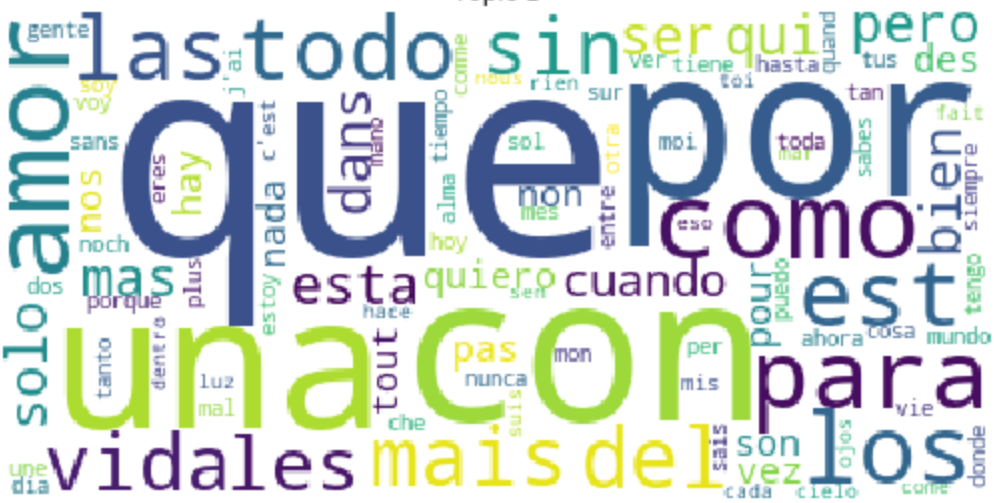
	1	2	3	4	5
1	bullet	bark	alegre	aussi	acaba
2	beans	bleed	awake	evidence	ace
3	alegre	borrow	advice	askin'	advice
4	acaba	blir	apart	escucha	amongst
5	alegria	born	andar	ask	apart
6	afraid	bord	anticipating	lucky	ahora
7	amongst	blaze	acaba	every	bold
8	add	bowl	ace	esperar	ahhh
9	'bout	drugs	best	estar	alegre
10	ace	blink	afford	eres	amazing
11	advice	bla	add	atomic	approach
12	altar	dieser	amongst	euch	anticipating
13	apart	bald	animal	execution	affection
14	approach	blisters	'fore	esperanza	ali
15	angry	blessed	alegria	everyday	because

Plots for top topics with top words for each topic



Word clouds for different topics

Topic 1



Topic 2

Interestingly, there seem to be a topic created with words that are non English in song lyrics. It will be interesting to see if this topic can predict popularity of a song.

We used cosine similarity measure to measure correlation between topics and song lyrics.

Sample dataset with cosine similarity:

track_id	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
TRAAARJ128F9320760	0.040825	0.013608	0.040825	0.000000	0.040825
TRAABHC128F933A3F8	0.024434	0.036651	0.036651	0.024434	0.036651
TRAACER128F4290F96	0.021953	0.021953	0.043906	0.010976	0.032929
TRAADLH12903CA70EE	0.000000	0.020851	0.000000	0.020851	0.000000
TRAAFRM128F9320F58	0.000000	0.000000	0.000000	0.044721	0.000000

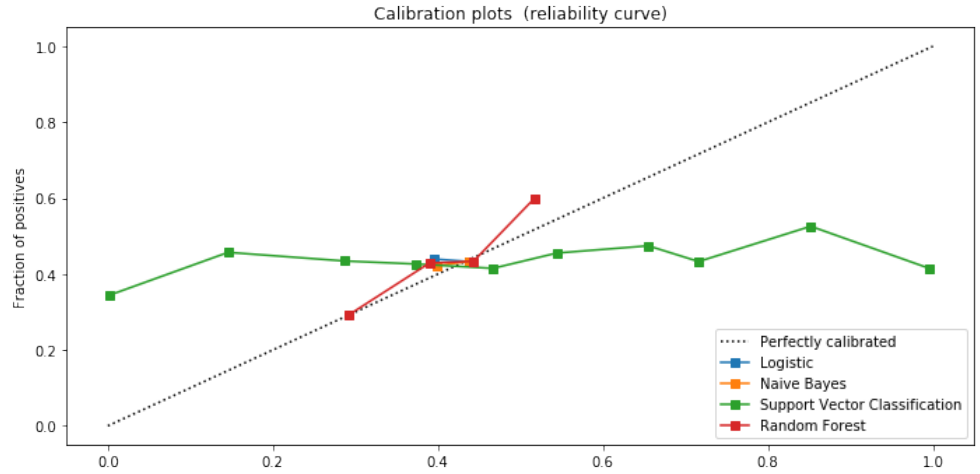
We classified top topic (highest in cosine similarity score) as 1, other topics as 0 for each song. Sample shown below:

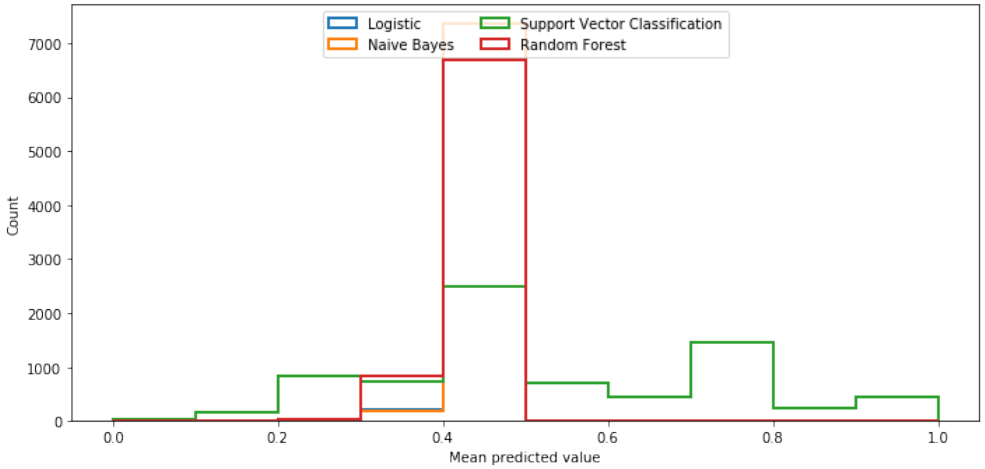
track_id	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
TRAAARJ128F9320760	1	0	1	0	1
TRAABHC128F933A3F8	0	1	1	0	1
TRAACER128F4290F96	0	0	1	0	0
TRAADLH12903CA70EE	0	1	0	1	0
TRAAFRM128F9320F58	0	0	0	1	0

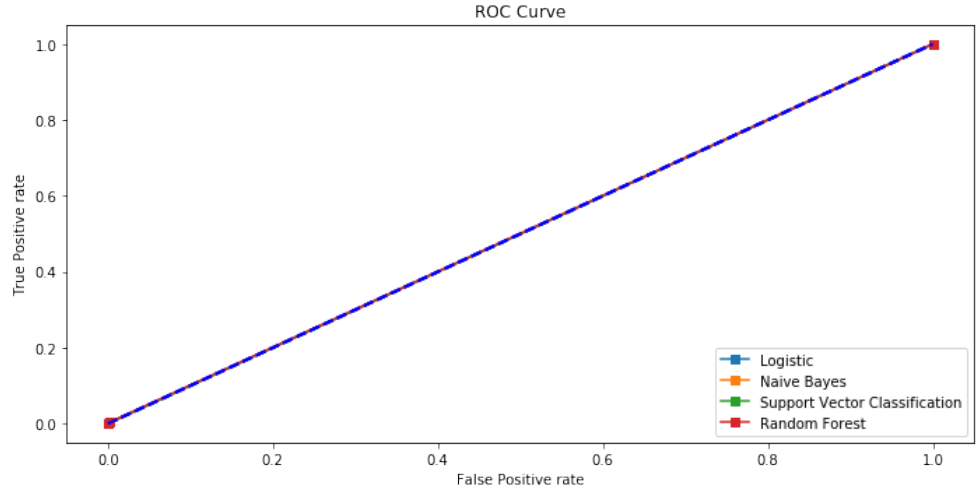
Training and testing models with Topics only: We used topics information *only* to predict song hotness, results are provided below.

```
Accuracy and AUC (test) for Logistic: 56.7% and 0.00 respectively
Accuracy and AUC (test) for Naive Bayes: 56.7% and 0.00 respectively
Accuracy and AUC (test) for Support Vector Classification: 56.7% and 0.00 respectively

Accuracy and AUC (test) for Random Forest: 56.7% and 0.00 respectively
```







As we can see above, topics that are derived from song lyrics are not predictive for song hotness. Next we used topics in addition to the features used in baseline model to predict song hotness.

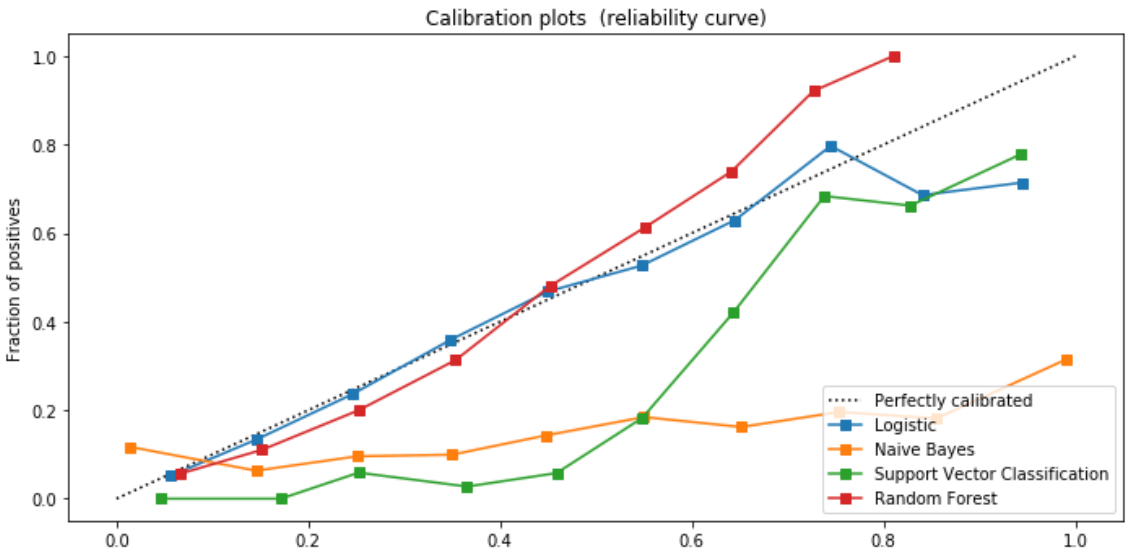
Training and testing models with Topics and features used in baseline model: We added topics derived from LDA into the dataset used for building baseline model to predict song hotness.

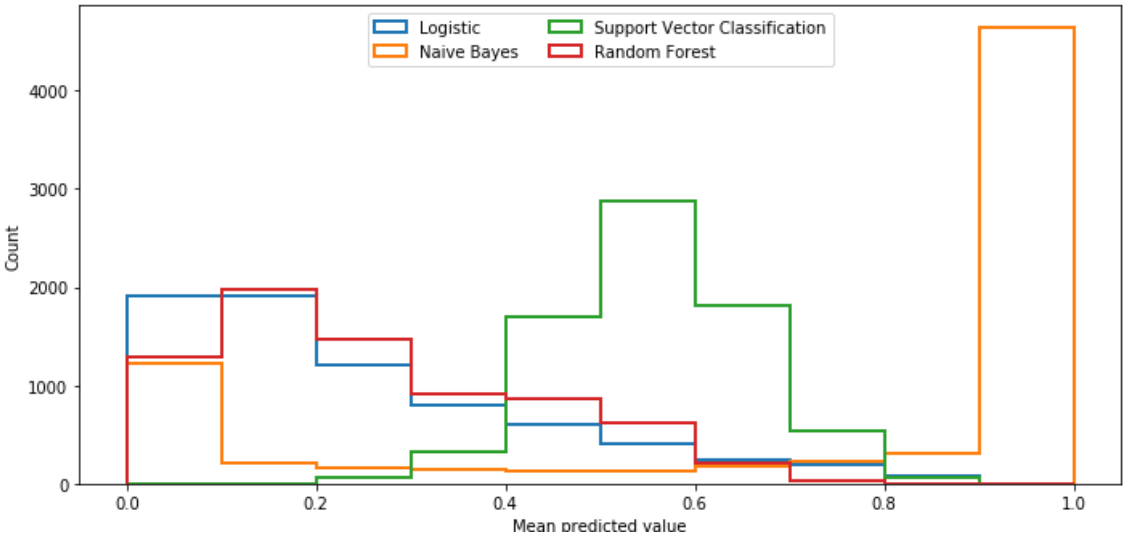
As discussed during building baseline model, we performed the following before building this model.

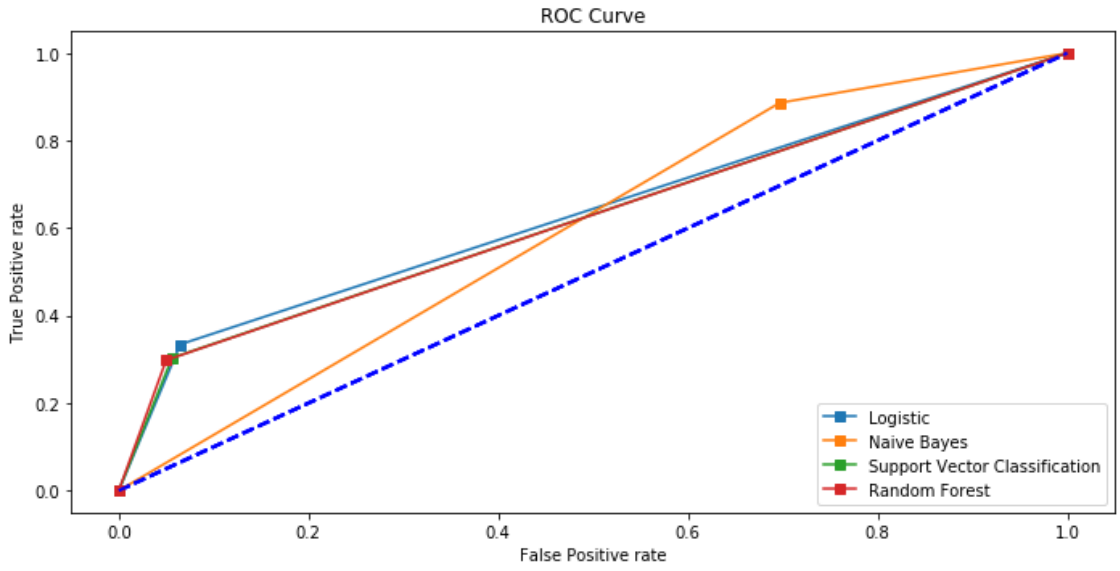
1- Removing null song hotness
2- Classifying song hotness into binary
3- Train and Test split
4- Building model to predict song hotness with Classical algorithms
    a. Logistic Regression
    b. Naïve Bayes
    c. Support Vector Classification
    d. Random Forest
5- Building model to predict song hotness with neural network
    a. Multi layer perceptron

Results for Classical algorithms:

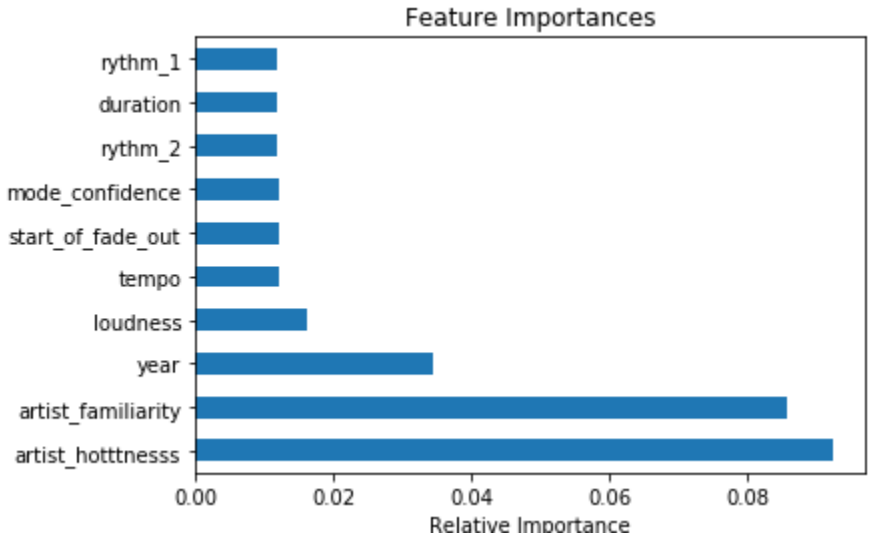
```
Accuracy and AUC (test) for Logistic: 78.7% and 0.72 respectively
Accuracy and AUC (test) for Naive Bayes: 44.7% and 0.59 respectively
Accuracy and AUC (test) for Support Vector Classification: 78.6% and 0.72 res
pectively
Accuracy and AUC (test) for Random Forest: 79.0% and 0.73 respectively
```


Calibration plots (reliability curve)
Fraction of positives
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Perfectly calibrated
Logistic
Naive Bayes
Support Vector Classification
Random Forest


Logistic
Naive Bayes
Support Vector Classification
Random Forest
Count
4000
3000
2000
1000
0
0.0
0.2
0.4
0.6
0.8
1.0
Mean predicted value

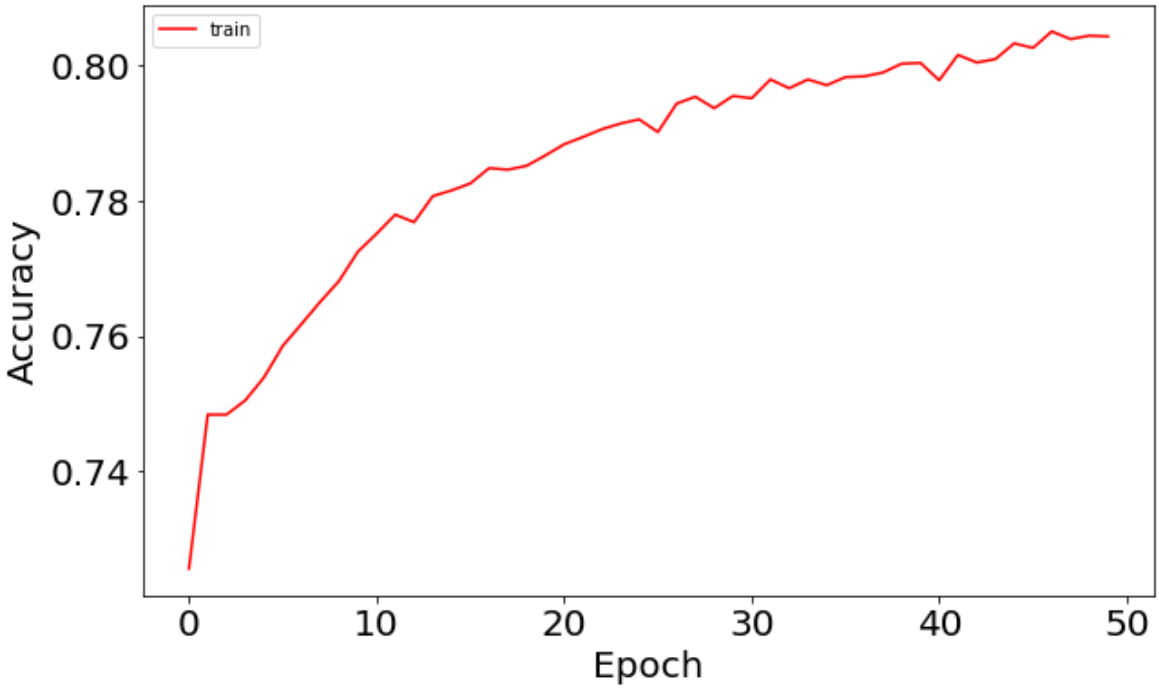


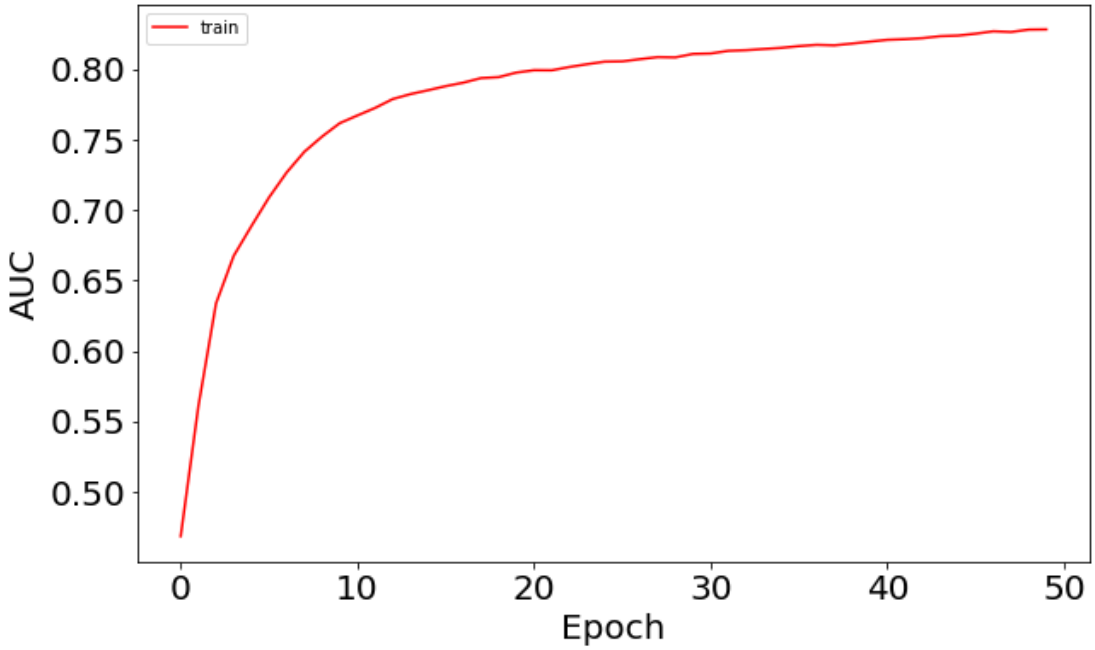
Feature Importance for Random Forest model (Random Forest provided the best accuracy and area under the curve)



Results for Neural Network

We plotted a 2-layer perceptron to train the model to predict song hotness. We received accuracy = **80.44%** and AUC = **0.82**.





As we have seen above, the accuracy of our models for both classical algorithms (i.e. Random forest) and multi-layer perceptron didn't improve over our baseline model even after utilizing the topic information from lyrics data. Some of the reasons for that could be:

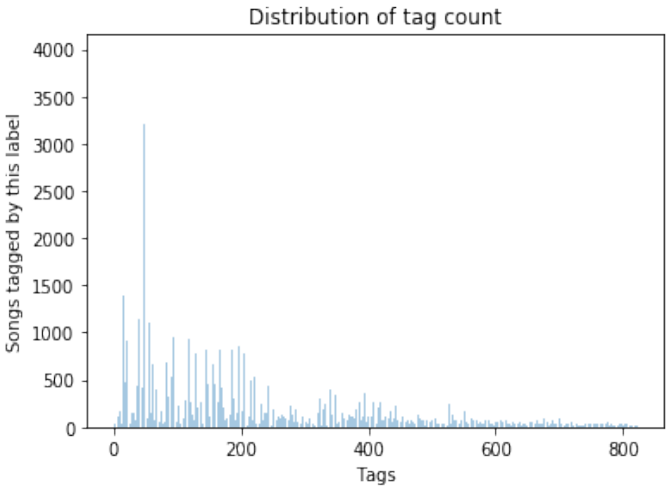
1- Lyrics data was not available for entire million song dataset and hence the data that was available could be biased at source.

2- The approach we used to summaries lyrics using LDA to identify topics is not probably appropriate in predicting song hotness.

3- We probably didn't use the correct architecture for our neural network with optimal parameters that could predict song hotness with higher accuracy.

4- In next steps, we could address above points to improve accuracy of our baseline model.

Artist terms (Tags.ipynb)



The data set contains artist terms for each song. In all, there is 3100 artist terms in use in the subset. These tags may be terms like «new wave», «indie rock», «americana» or similar. Each song is tagged with a number of such terms. The song with most terms linked to it, had 82 terms while others had zero.

We cleaned up the terms and counted the number of terms each song was tagged with as a parameter. Then we filtered out the terms that were used for less than 20 songs. This left us with 824 terms that we one hot encoded for the models, in addition to the parameter stating the count of terms used for the song (before the removal of the sparsely used terms).

To ensure that the artist terms are as common as possible across the different songs, we did also implement stemming. A few examples look like this:

Original	Stemmed
progressive	progress
progressive	progress
alternative	altern
death metal	death met
alternative metal	alternative met

We chose several models on this data that we trained on for the artist terms; both classification and regression models.

As for classification, these models were used:

- K-Nearest Neighbor
- Linear support vector machine

- Non-linear support vector machine
- Gaussian Process Classifier
- Decision Tree Classifier
- Random Forrest Classifier
- Multilayer Perceptron Classifier
- AdaBoost Classifier
- Gaussian Naive Bayes Classifier
- Quadratic Discriminant Analysis
- Neural networks

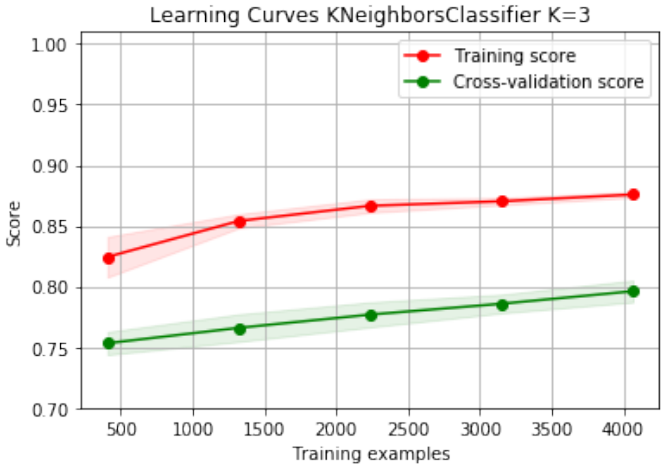
While for the regression models, we chose the following:

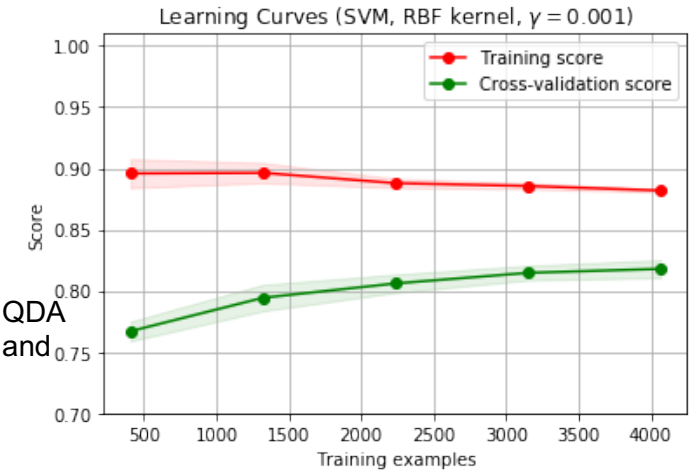
- 2 different Neural nets with K-Fold Cross Validation
- Gradient Boosting

Before we ran the models, we classified songs in the upper quartile of “song_hotttnesss” and the rest as “not”.

This was what we trained the models on, and what we tried to beat.

As for the models, K-Nearest Neighbor and Non-linear support vector machine achieved the best results on our initial run. These were ran again with cross validation to ensure a more valid result.





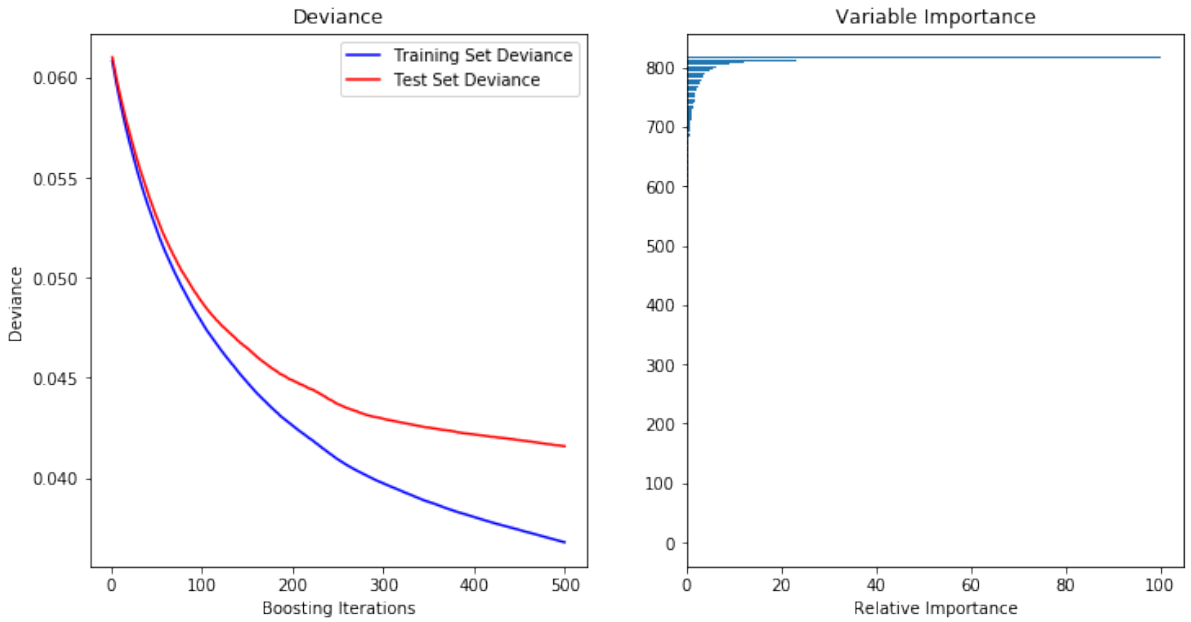
QDA and

Random Forrest did not improve the result compared a baseline of declaring all songs as not hot. Naive Bayes did substantially worse. The rest ended somewhere in the middle.

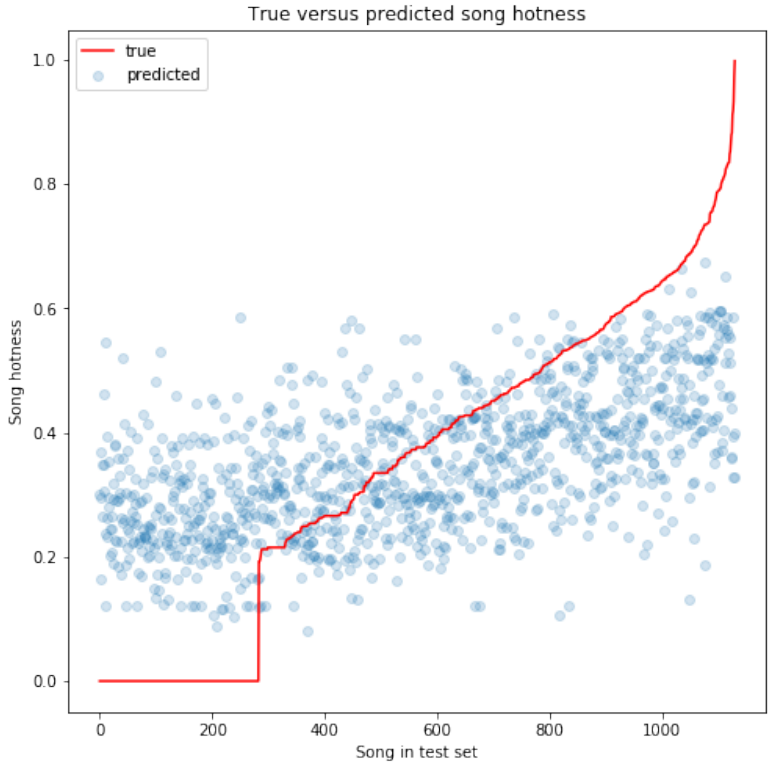
We tried several combinations of neural networks as well and achieved similar results to QDA and the Random Forrest. This is a substantial improvement above the baseline model, but unfortunately not enough to rely on if the goal is to be a multimillionaire songwriter.

We also tried to run regression models on the artist terms, as stated above.

The gradient boosting model gave the following deviances and variable importance:



However, when we compared the true hotness to the predicted hotness from the gradient boosting model we saw that this result was not of much use:



As one may see, the predicted song hotness increases slightly as the true song hotness increase.

However, this effect is very weak.

And similarly important, the model does not predict any songs above approximately 0.65, while we in the classification section only classified songs with a score above 0.75 as hot. Hence, the regression model has not classified any songs as hot according to our earlier definition.

So even though we can see that there is a correlation between the song's hotness and the predicted hotness, this correlation is too weak to be useful.

Conclusions

There is no quick and easy formula to create a super-hit. At least not based on the available information in the million songs dataset. We have discovered a few ways to improve the chances of creating a hot song but did not thru the help of machine-learning find an easy receipt.

We started by exploring how the dataset could be augumented with information from other sources. Unfortunately, the results ended up being incomplete, with more than half of the records having no genre information and around three-quarters having no style information.

Then we looked at rhythm histograms, to see if the “rhythmic energy” could guide us towards our goal. We created two sets of models for classification and regression, and in the prior classified the top 25 % of songs according to the song_hotttness as “hot” and the rest as “not”. Hence, the baseline model for classification should beat 75 %.

Classical models, such as Random Forest and Support Vector Classification achieved comparable result as CRNN network and parallel CNN RNN network, just above 80 %.

We continued to work with the lyrics, to see if there are a relationship between song words and hit topics, that could aid us in predicting a songs hotttness. We combined the dataset, mapped the data, and used Latend Dirichlet Allocation on the full corpus to extract 4 topics of 10 words each.

We discovered words which appear only in hits, that were used to predict the songs hotttness. The models result fluctuated from run to run, but did not give us any improvement over the baseline.

Then we combined lyrics data with the dataset to see if song lyrics could help predict a songs hotness. Lyrics were only available for about a quarter of the songs.

We used LDA to find the optimal number of topics, encoded topics and ran models both to use topics information as the sole predictor of song hotttness as well as using this information in addition to the features used in the baseline model.

The models did not improve the result achieved with the baseline model.

Lastly, we checked if artist terms could be used to predict a songs hotttness. These are terms such as “black metal”, “alternative rock” or “electronica” that are linked to the songs thru the artists.

We tried both several classification models as well as regression models. Our work show that this approach creates an improvement over the baseline model. However, the results did not improve enough to create unemployment in the record label or among music professionals.

All in all, we have found several approaches to achieve better results than the baseline model when it comes to classification, while the regression models are found to be too weak to add any substantial value. This clearly shows that there is a great potential in using machine-learning in the music industry. However, there is need for additional research to what we did to create models that can create a seismic shift in how hits are made.