# Music Sentiment Analysis

Rahul Gaonkar(rpg283)
Manish Nagdevani(man514)
Seo Gregory Pallichirayil (sgp322)

## Introduction

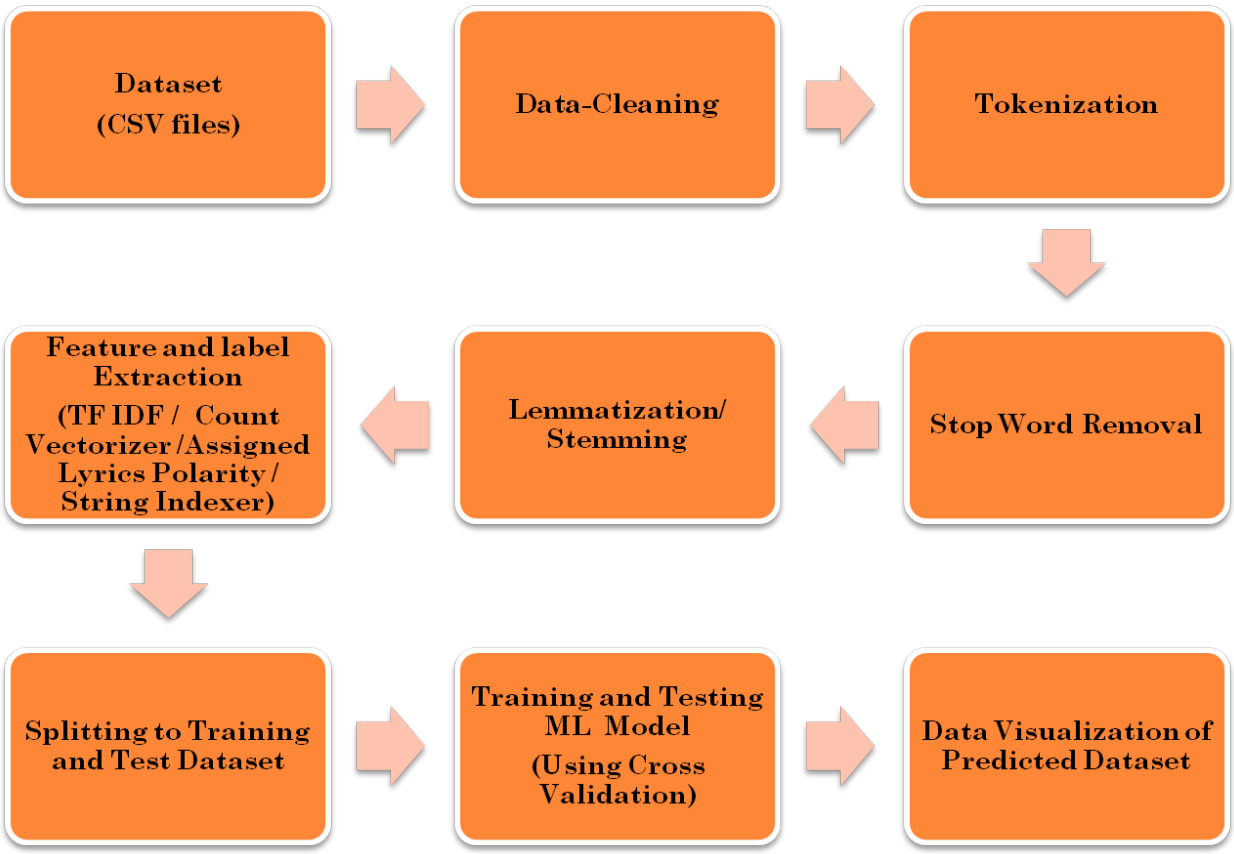
The aim of the project is to determine the mood of the song based on the lyrics. Various components of a song contribute to its sentiment/mood in general. Some of these components include lyrics, tempo, background music etc. The approach used in this project is to simply understand the pattern of words in the song lyrics for different categories for example, Happy Songs and Sad Songs.

## Background

Lots of machine learning tools are available today to perform Natural Language Processing tasks. Task like predicting the sentiment of a text are quite common in Machine Learning domain. The challenge is to perform these tasks at scale.

## Architecture

This project can be broken down to logical stages which are common in any NLP project. The following describes the stages broadly:



1. Data Gathering
2. Data Preprocessing
3. Feature Extraction
4. Feature Selection
5. Model building & Tuning

## Data Gathering

This is a supervised Machine Learning classification problem. Hence, a labeled set of data was created which contained lyrics of about 1000 english songs and the label stating whether the song was Happy or Sad. This data was available in a csv format on the Internet on the github[1] repository of one of the researcher.

## Data Preprocessing

The data was uploaded to the HDFS and loaded in Spark using pyspark. Following steps were performed to preprocess the input data:

1. **Data Cleaning**

- The lyrics data obtained from the internet contained a lot of empty strings, undesired characters, punctuations etc.
- Data cleaning stage was designed to clean the text data and remove all the undesired characters including Punctuation marks, Digits, White Spaces and special characters

2. **Tokenization**

- After cleaning the data,the lyrics is divided into a set of words using the process of tokenization which helps us to work on words directly to get the sentiment of the song using the lyrics

3. **Stop Words Removal**

- Stop words are the words in a language that are used to build a sentence. Words like “the”,”a”,”of” etc are the common known stop words.
- These words constitute the maximum frequency of occurrence when compared against all the other words
- These words don’t add any value in the sentiment of the sentence or a paragraph or a lyrics on whole and therefore should be removed

4. **Stemming**

- In linguistic morphology and information retrieval, stemming is the process of reducing inflected (or sometimes derived) words to their word stem, base or root form[2]
- Words like “cats”, “catlike”, “catty” all map down to the base word “cat”, stemming is the approach to derive the base/root word from a given word
- It is necessary to perform stemming to reduce the number of potential features by limiting different inflated words to just their base word

5. **Lemmatization**

- Lemmatization is an alternative approach from stemming to removing inflected words by determining the parts of speech and utilizing WordNets lexical database of English .
- Words like “leafs” and “leaves” all map down to “leaf”.This gives better results then stemming while determining the sentiment of the text i.e. lyrics

6. **Label Encoding**

- The process of encoding a categorical variable is termed as Label encoding
- It basically maps all the different categories/classes to some integer value
- In our case, mood is a categorical variable and can be either Happy or Sad
- This is done using String Indexer in spark

## Feature Extraction

- Machine learning models can’t work with textual features and require some sort of numerical features
- A common practice in training text models is by taking into account the frequency of the words in the document or corpus
- There are 2 techniques to achieve this: Count Vectorization and TFIDF Vectorization

**1. Count Vectorization**

- This approach simply calculates the frequency of words in the entire corpus and creates a vocabulary of words
- Typically a vocabulary of happy and sad words is created

**2. TFIDF**

- TFIDF is an alternative to calculate word frequency
- It assigns a word frequency score which tells how important the word is with respect to the document and the entire corpus
- For example it assigns a higher value to a word that is frequent in a few documents but not across all the documents in the corpus

**3. Polarity Assignment of Lyrics**

- Another important feature that helped us in determining the sentiment of the song based on lyrics is Polarity Assignment of Lyrics.
- We used the **nltk.sentiment.vader** to assign the polarity value to a lyrics based on the positive and negative words present in the lyrics.
- Negative words would usually have a negative polarity value and positive words would have a positive polarity value.
- The assumption used here was that happy songs would have more number of positive words as compared to the negative words and sad songs will have more number of negative words as compared to the positive words
- The polarity of the lyrics is the sum of the polarity value of all the words inside the lyrics.

# Feature Selection

- Computing the sentiment of a lyrics - poetic text isn't easy as it involves considering the pattern of the words used, sequence of the words etc
- Sequence of words play a vital role in determining the sentiment of the lyrics. For example, "i am sorry" - this 3-gram makes more sense in determining the sentiment, than just simply considering individual words(unigram) "i", "am", "sorry"
- For the sake of this project, we considered unigrams and bigrams both as type of feature(we tried higher order grams but the performance of the model degraded)
- With all the machine learning models, we tried out 7 permutations:
    1. Count Vectorization on Unigram
    2. Count Vectorization on Bigram
    3. Count Vectorization on Unigram & Bigram
    4. TFIDF Vectorization on Unigram
    5. TFIDF Vectorization on Bigram
    6. TFIDF Vectorization on Unigram & Bigram
    7. Unigrams TFIDF Vectorization and Lyrics Polarity Value
- Broadly speaking, the code consistently contained 7 pipelines for SVM and Naive Bayesian model, and each model was trained on features obtained after performing the corresponding step as described above. For example, for any model, Pipeline 1 corresponds to training a model on features obtained from permutation 1, Pipeline 2 corresponds to training a model on features obtained from permutation 2 and so on

# Model Building

**1. SVM**

- A linear classifier that works on the principle of Optimal Separating Hyperplane
- We tried different pipelines on SVM that fetched different accuracies
- Refer the graph in the results section to compare the accuracies we achieved for different permutations

**2. Naive Bayesian**

- A probabilistic model that computes the probability of a class based on Bayesian Inference
- We tried different pipelines on Naive Bayes model that fetched different accuracies

**3. Gradient Boosting**

- A prediction model in the form of an ensemble of weak prediction models, typically decision trees
- We tried only the pipeline with "**Unigrams TFIDF Vectorization and Lyrics Polarity Value"** feature and fetched the accuracy.

# Tuning

1. **Cross-Validation**

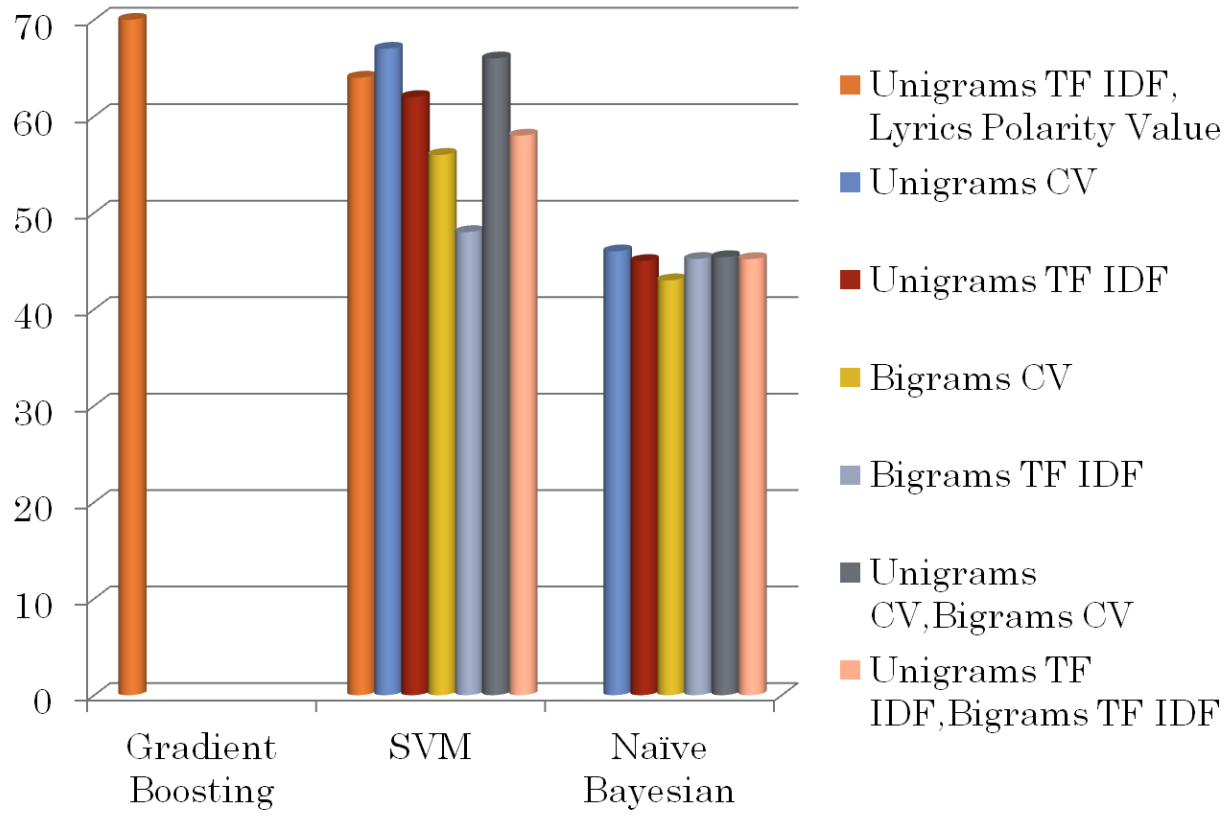
- Cross Validation is the technique to make sure the model is not overfitted
- This is usually performed using K-Fold method
- For this project, we cross-validated all the model pipelines by following a 8-fold validation & 10-fold validation approach

2. **Hyperparameter tuning**

- The process of fine tuning the model by varying its parameters is called Hyperparameter tuning
- It requires expertise on the underlying math of the machine learning model
- For this project, since we got best accuracy on Gradient Boosting after applying cross-validation, we fine tuned Gradient Boosting
- Following parameters of the Gradient Boosting model were tuned:
    1. Max Number of Iteration: Ranged from 10 to 40, we got best accuracy on 10, higher values takes more time for the model to converge

# Results

Below is the bar graph and tabular representation of all the models(X-axis) used and their respective accuracy(Y-axis) for different combinations of features selected:



| Features / Model | Gradient Boosting | SVM | Naïve Bayesian |
|---|---|---|---|
| **Unigrams TF IDF, Lyrics Polarity Value** | 70 | 64 | |
| **Unigrams CV** | | 67 | 46 |
| **Unigrams TF IDF** | | 62 | 45 |
| **Bigrams CV** | | 56 | 43 |
| **Bigrams TF IDF** | | 48 | 45.21 |
| **Unigrams CV,Bigrams CV** | | 66 | 45.4 |
| **Unigrams TF IDF,Bigrams TF IDF** | | 58 | 45.2 |

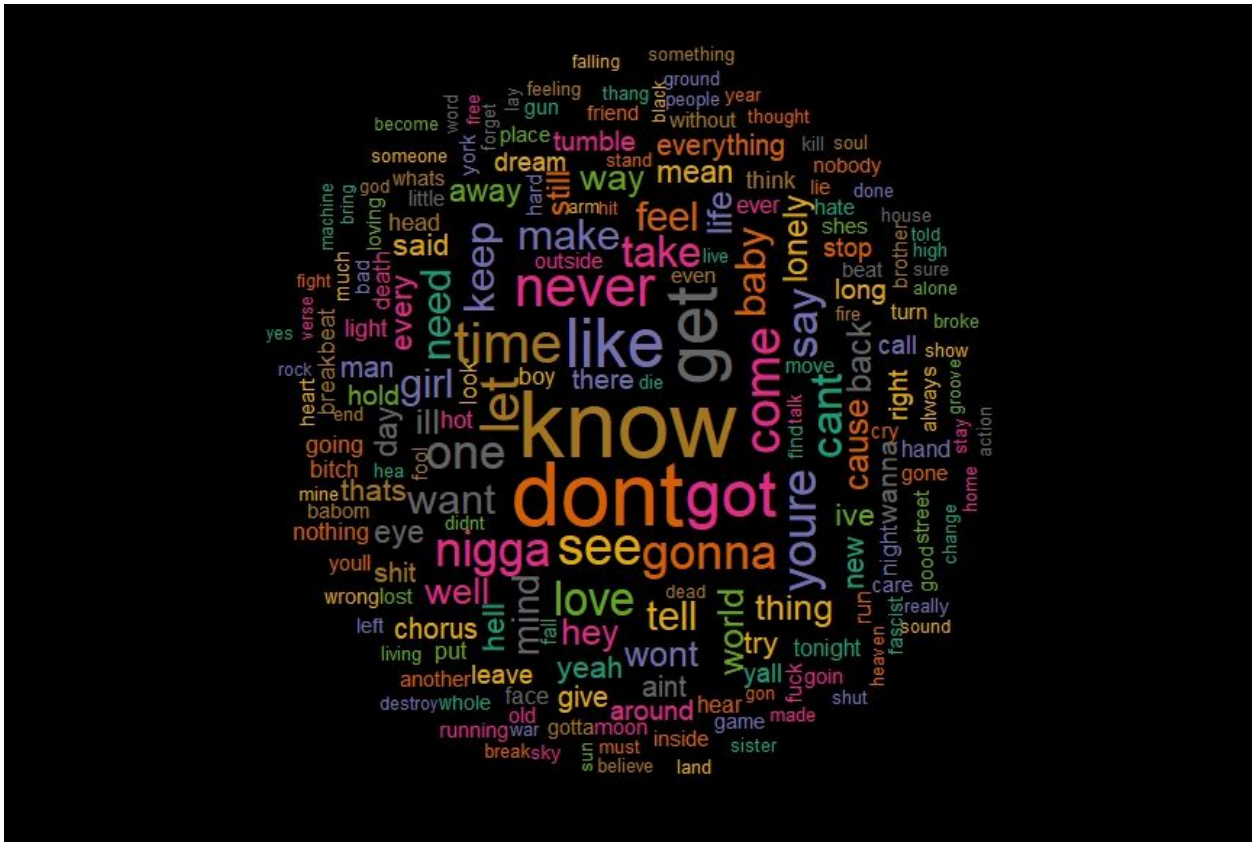
## Visualization

After the classification of the songs into happy and sad, We Collected the most frequently occuring words in lyrics of both happy and sad songs and made a few visualizations from it.

**Word Cloud of Happy Words**

This Visualization represents the frequently occuring happy words, and the words shown in larger font size represents that those are more frequent than the others.

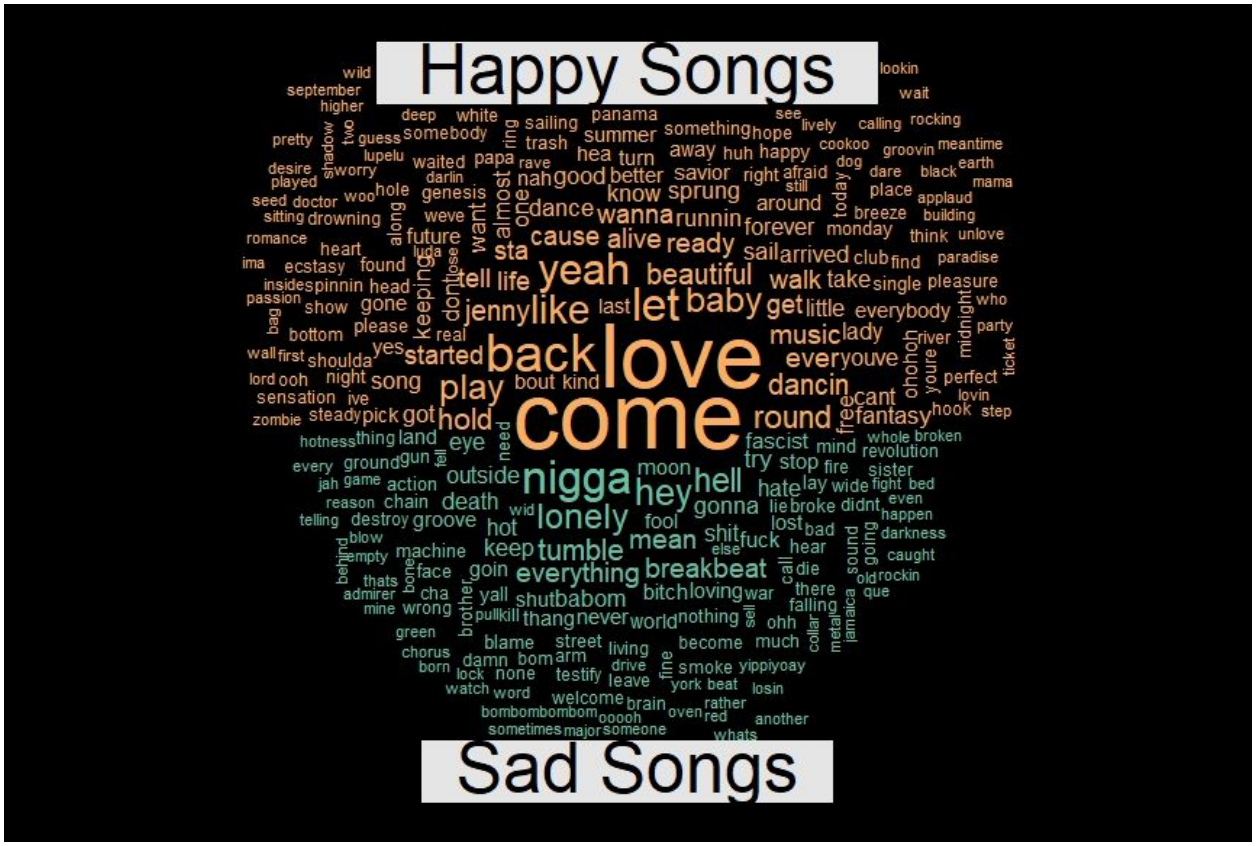
**Word Cloud of Sad Words**




This Visualization represents the frequently occuring sad words, and the words shown in larger font size represents that those are more frequent than the others.

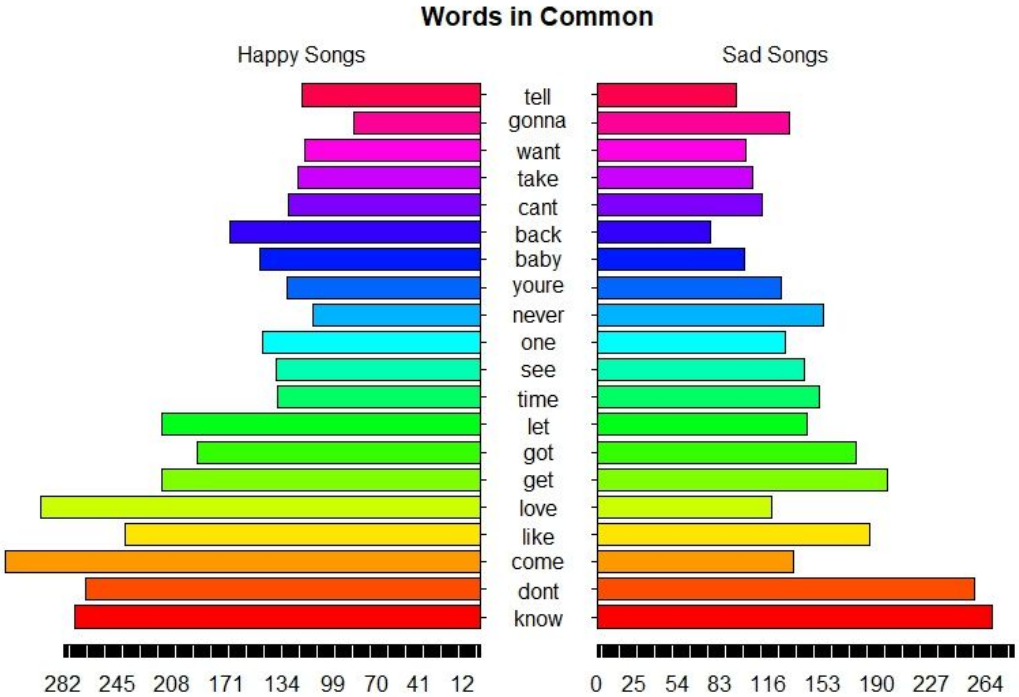
**Comparison Cloud**

In Order to compare the frequency of words between sad songs and happy songs , we need a comparison cloud. Below is a comparison cloud we designed in R language , using the words of the songs we predicted as Happy or Sad Songs.




**Pyramid Plot**

Songs might be categorically divided, but still there are always some common words in both the categories, in order to visualize these words we designed a pyramid plot in R , where we have taken the most number of common words and designed horizontal bar graphs indicating their frequency.



## Future Work

- Merge the ML model with an existing music application to provide better recommendation of songs to users.
- To provide different tabs for recommendation of songs like by mood, genre, artist, etc
- To add more types of sentiments/ labels like romantic, motivational etc.

## Conclusion

The Music Sentiment Analysis Project helps in predicting the mood/sentiment of the song.It can be used as a recommendation system when integrated with music app to recommend songs of similar mood/sentiment to the user.

## References


[1] Sebastian Raschka : MusicMood. https://github.com/rasbt/musicmood

[2] https://en.wikipedia.org/wiki/Stemming

[3] E. Cambria, S. Poria, R. Bajpai, and B. Schuller. SenticNet 4: *A semantic resource for sentiment analysis based on conceptual primitives. In: COLING*, pp. 2666-2677 (2016)