NLP Lyrics Final Presentation

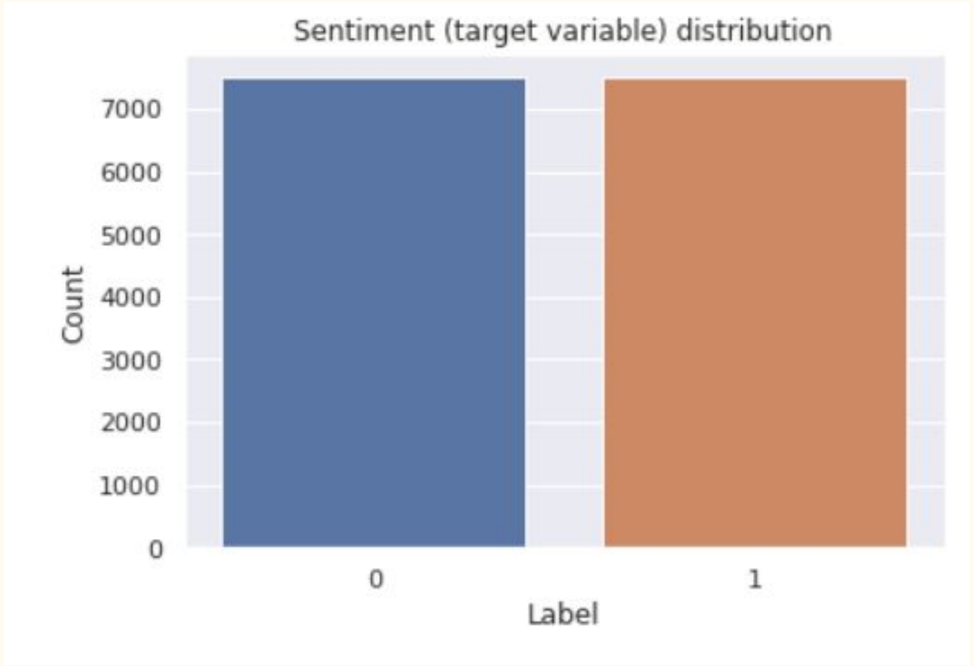
Mrunal Prakash Gavali, Jefferson Perez Diaz

Positive and Negative Valence

Dataset: Song Lyrics dataset of different artists labeled using Spotify valence audio feature :

kaggle_dataset

From this dataset we will take a sample of extreme positive and negative song lyrics.



Lyrics NLP Project

- The goal of the project is to predict the overall positive and negative feelings aroused by a song using Spotify valence audio feature as a label to train our model based on the binary (extreme positive and negative) valence.

- The Spotify API describes the valence label as a measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track.

- Tracks with high valence arouses positive feelings such as happiness, cheerfulness, etc, while tracks with low valence arouses negative feelings such as sadness, depression, etc.

- For the extreme cases, that is the cases where the song demonstrates a high measure of negative (< 0.33) or high positive feelings(> 0.67), the labels have been converted into 0's and 1's and used to perform supervised NLP sentiment analysis.

Why switch to Spotify lyrics dataset ?

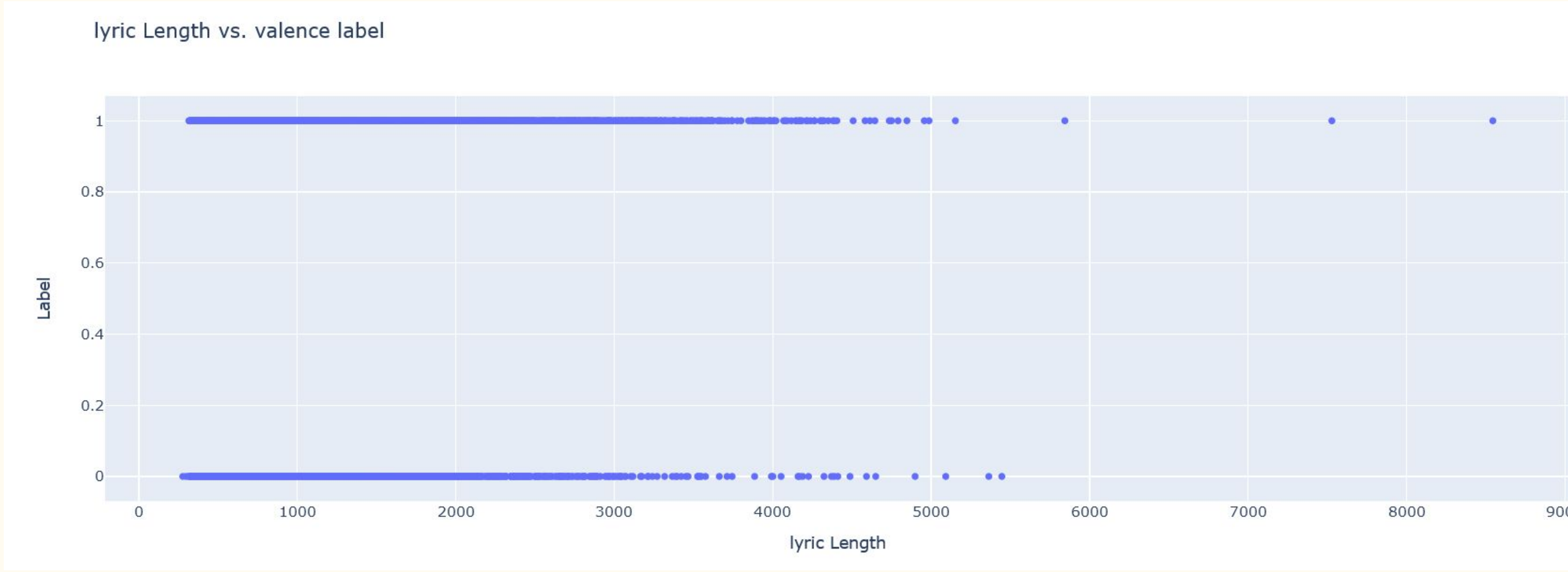
- The spotify lyrics dataset is labeled by professional experts.This dataset took into account rhythm, tempo and audio features to label the songs.

- Because the previous dataset we used for part-1 was not labeled. We were labeling the data by taking bag of words into account. But the labeled result was random. Most of the times, words used in the lyrics are not good indication of songs that would arouse positive or negative feelings.

- Also, the sarcasm and other contextual meaning in lyrics is lost. So, there is a high probability that the labels for songs generated are not good for the problem we are trying to solve. Which is the biggest concern because model learns what is fed into it. So, garbage in garbage out issue.

- Also, the dataset was too small.

Lyric Length vs Valence Label



LSTM

LSTM

<u>Tokenization</u>

The process of taking out the tokens out of the sentences is tokenizing.

Tokens basically mean the individual words in the sentence. example:

sentence: ['i love my cat']

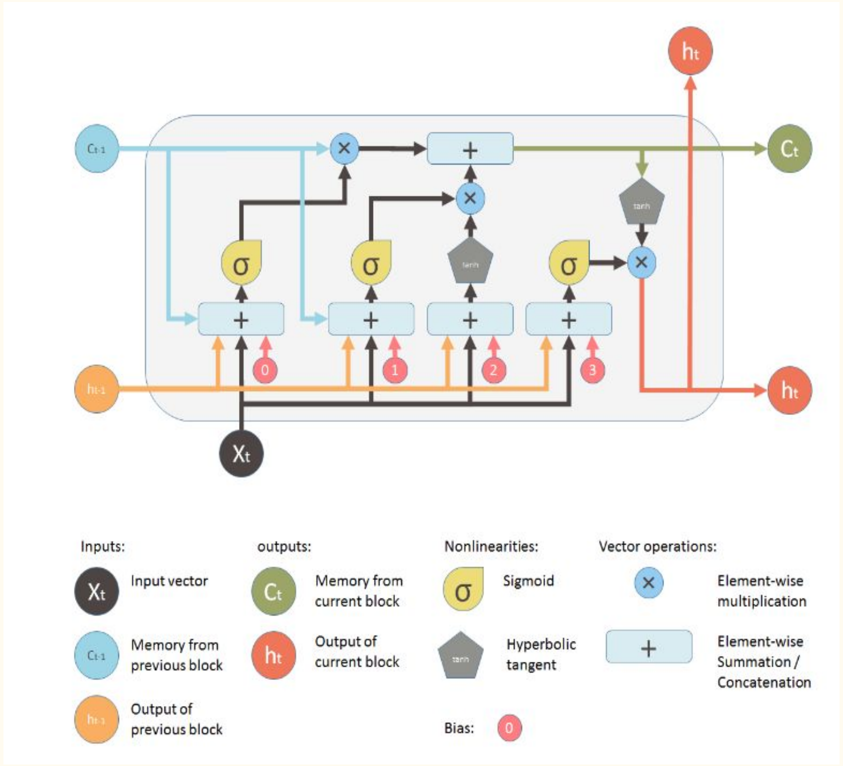
tokens: ['i', 'love', 'my', 'cat']

The tokenizer then converts these tokens into sequence of integers where each word is given its word index.
 example:
tokens: [['i', 'love', 'my', 'cat']['i', 'have', 'a', 'cat']]

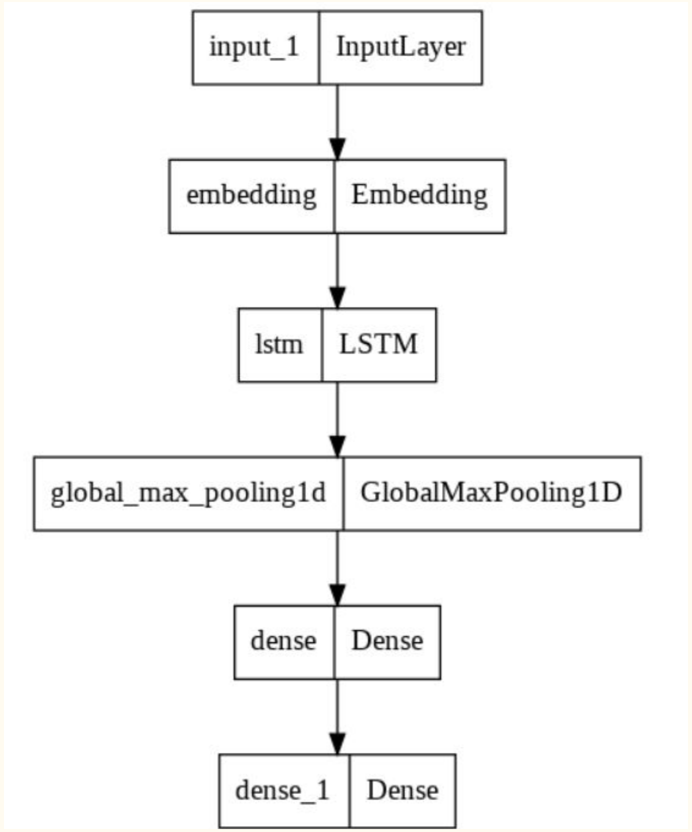
sequence: [[1, 2, 3, 4][1, 5, 6, 4]]

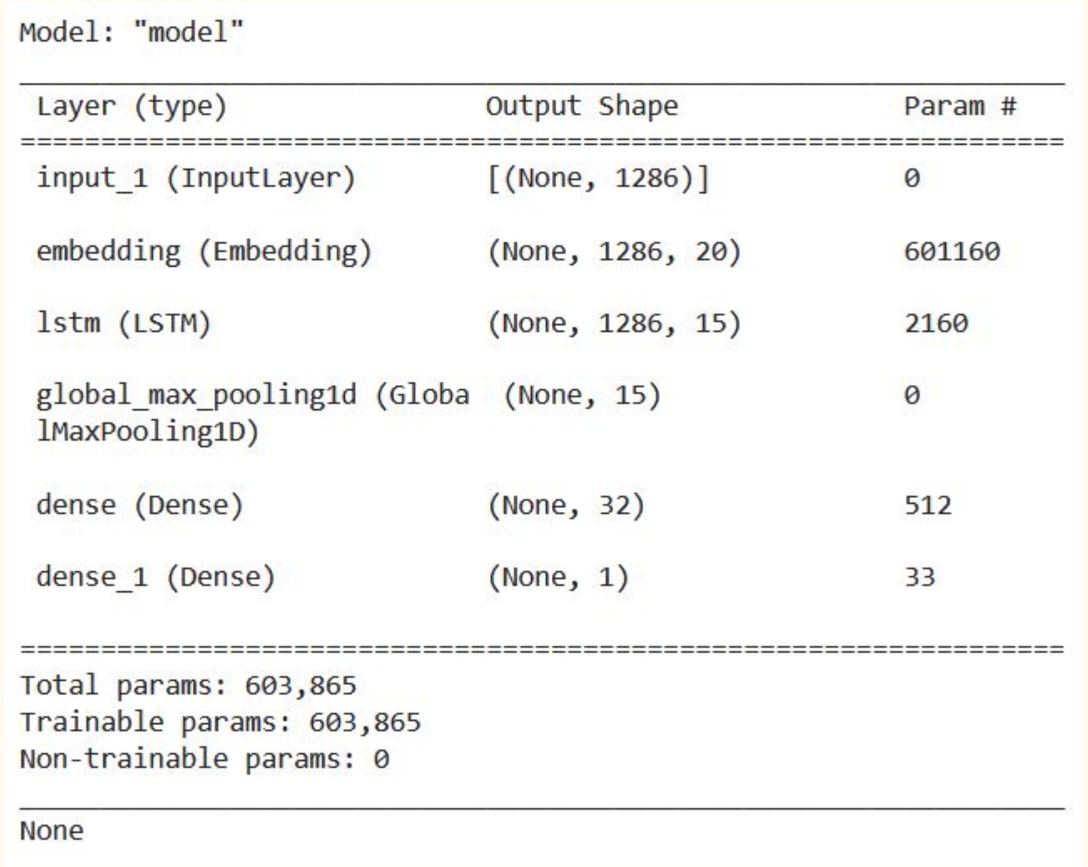
<u>Padding</u> is implemented for each sequence so that the training and testing sequences are of same length.
[['i', 'love', 'my', 'cat']['i', 'have', 'a', 'cat'][‘i’, ‘have’,’a’, ‘cat’,’and’,’dog’]]

[[1, 2, 3, 4,0,0][1, 5, 6, 4,0,0][1,5,6,4,7,8]]



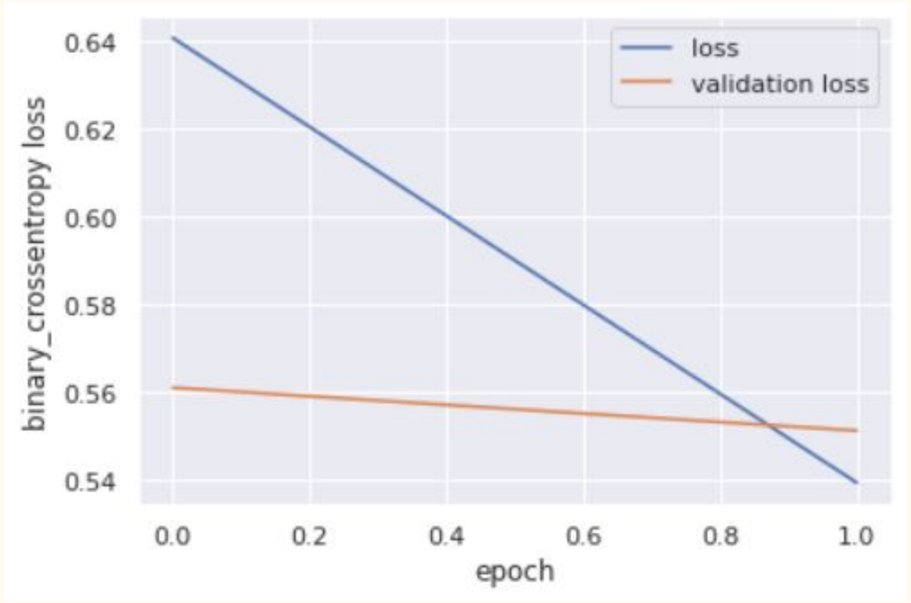
LSTM Model for Spotify Lyrics

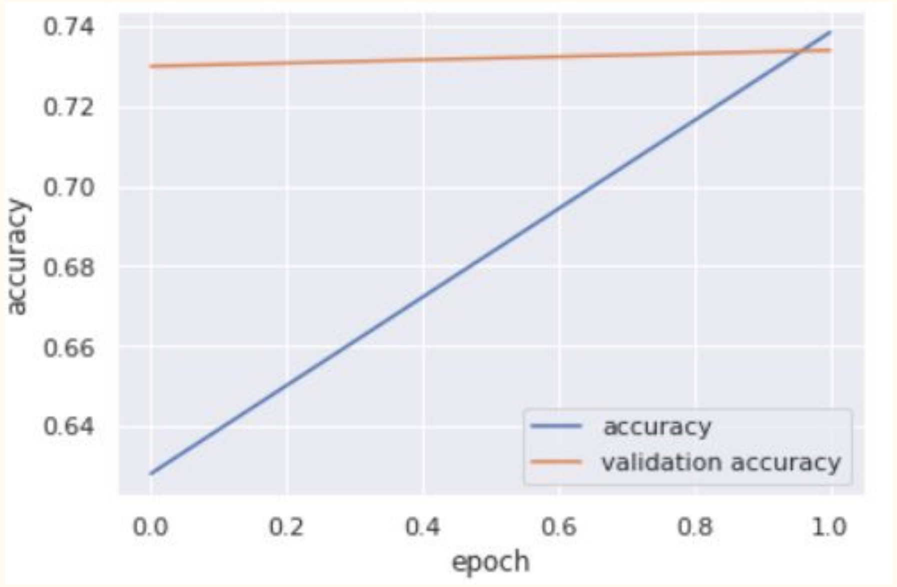




```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 1286)]            0

 embedding (Embedding)       (None, 1286, 20)          601160

 lstm (LSTM)                 (None, 1286, 15)          2160

 global_max_pooling1d (Globa  (None, 15)               0
 lMaxPooling1D)

 dense (Dense)               (None, 32)                512

 dense_1 (Dense)             (None, 1)                 33

=================================================================
Total params: 603,865
Trainable params: 603,865
Non-trainable params: 0
_________________________________________________________________
None
```

Data Mining Technique 1 - LSTM





LSTM

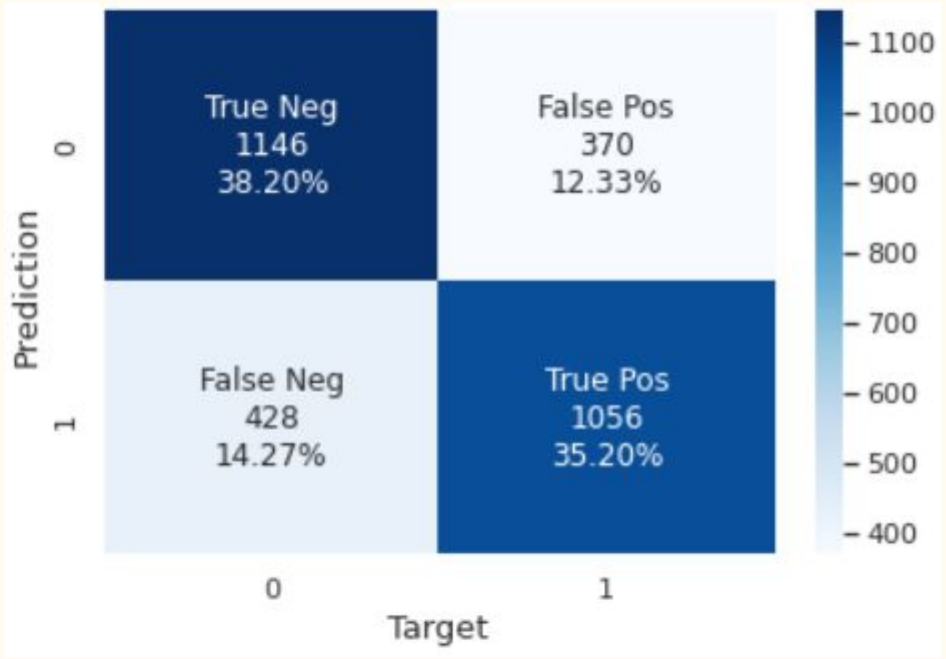
1. Use sigmoid activation instead of softmax. Using Softmax reduces the model accuracy to 49%-50%.
2. Using Optimizer ‘adam’ instead of ‘SGD’. SGD reduces the model accuracy to around 50%.
3. Use 2 epochs instead of 3 epochs. After 2 epochs the training accuracy increases at the cost of the decrease in validation accuracy and increase in validation loss.
4. Using Bidirectional layer decreases the accuracy by 2%. The validation loss also increases along the epochs and the validation accuracy decreases.
5. Global max pooling operation for 1D temporal data. Downsamples the input representation by taking the maximum value over the time dimension.

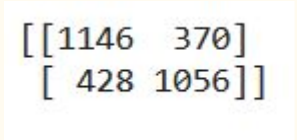
Note:

1. If the model overfitted, you would see that training loss < validation loss.
2. If The validation accuracy is greater than training accuracy. This means that the model has generalized fine. It can be considered in two ways:

- training data had several arduous cases to learn
- validation data containing easier cases to predict

LSTM Confusion Matrix

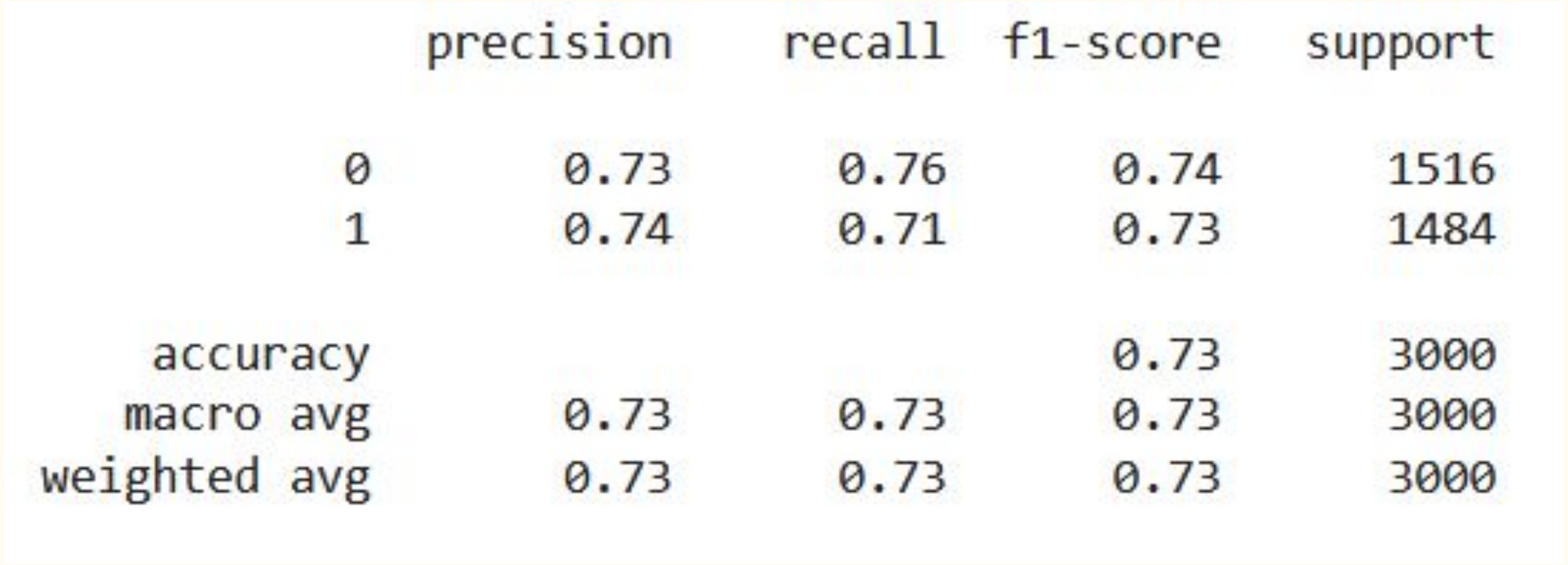




```
[[1146  370]
 [ 428 1056]]
```

- Accuracy = 73%
  - Amount of correctly predicted classes
- Precision = 74%
  - Amount of predicted negative examples that were correctly labeled.
- Specificity = 76%
  - Amount of predicted positive examples that were predicted correctly.
- Sensitivity/Recall = 71%
  - amount of negative examples that were predicted accurately
- F-score = 73%
  - Harmonic mean of recall and precision

LSTM Evaluation report



```
              precision    recall  f1-score   support

           0       0.73      0.76      0.74      1516
           1       0.74      0.71      0.73      1484

    accuracy                           0.73      3000
   macro avg       0.73      0.73      0.73      3000
weighted avg       0.73      0.73      0.73      3000
```

Sensitivity and Specificity

sensitivity = (True_Positive)/(True_Positive + False_Negative)
sensitivity/recall = 71%

specificity = (True_Negative)/(True_Negative + False_Positive)
specificity = 76%

LSTM prediction examples on unknown songs

1. Party in the USA by Miley Cyrus → Happy song predicted as Positive
2. I Hate U, I Love U Song by Gnash → arouses sad feelings and model predicted it negative
3. Ariana Grande 7 rings → arouses positive feelings predicted correctly as positive
4. Ariana Grande Thank you next next → arouses positive feelings predicted correctly as positive
5. Heaven by Beyonce → sad song predicted correctly as negative
6. Lose you to love me lyrics by Selena Gomez --> sad song predicted correctly as negative
7. Love Me Song by Justin Bieber --> happy song predicted correctly as positive
8. Dark Paradise by Lana Del Ray -> sad song predicted correctly as negative
9. Levitating by Dua Lipa -> happy song predicted correctly as positive

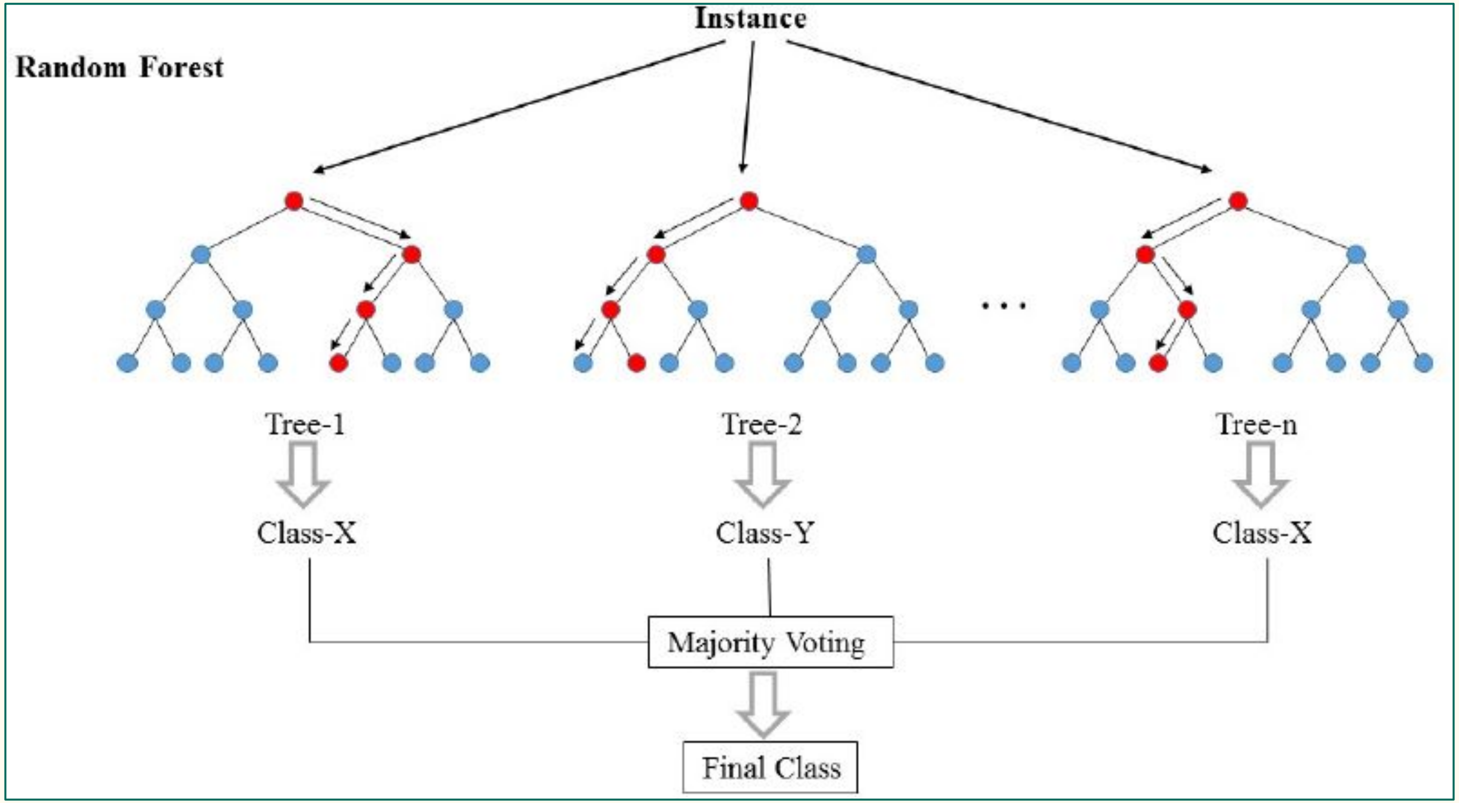
Random Forest Classification

Data Mining Technique 2 - Random Forest Classification

How does it work?

- Random Forest utilizes decision trees as the building block for the model(also known as an ensemble algorithm).
  - You can think of decisions trees as a bunch of conditional statements that yield to either one or another result.
  - This allows the algorithm to have a better predictive performance as it works with multiple learning algorithms to decide the prediction.
  - The idea is to make each tree in the forest give a prediction and based on the most votes, the algorithm predicts the final class for each string.
- Advantages:
  - Less chance of overfitting as different trees will led to different results and final vote will only favored the most class predicted.
  - Low bias

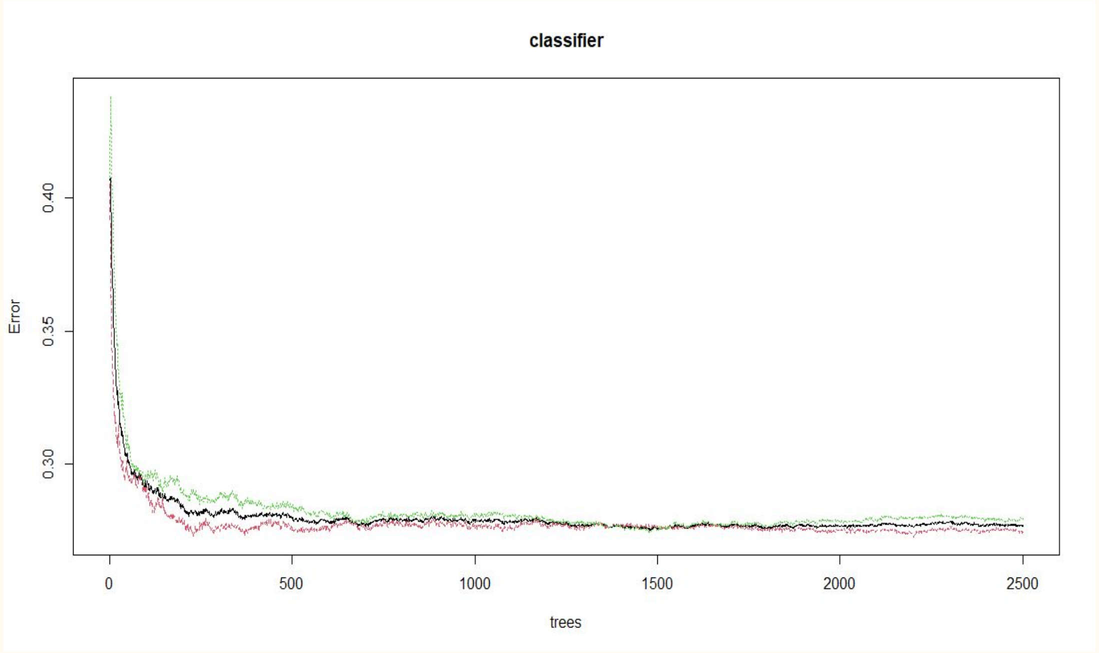
Random forest classification visualization



Evaluating random forest model

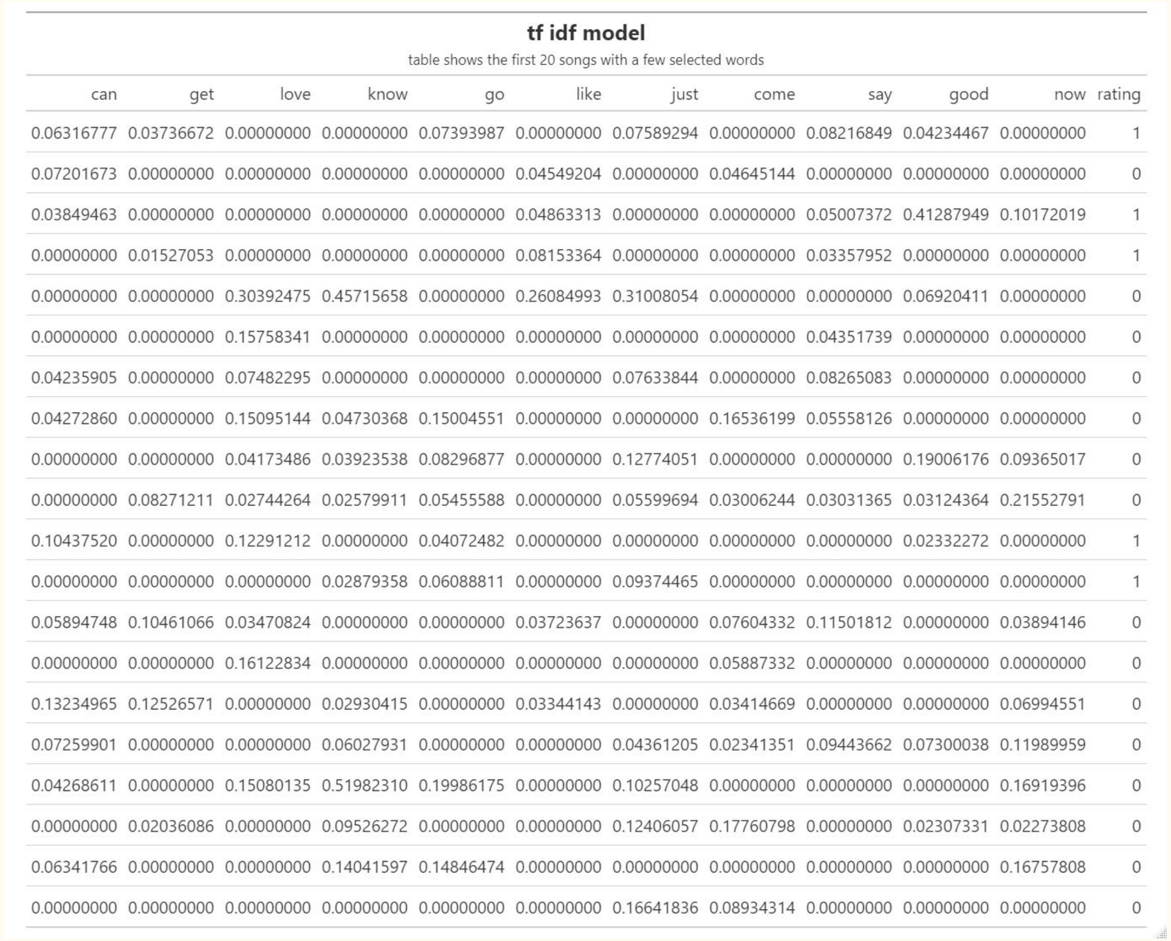
- Random Forest model was run with 2501 trees. Each tree having a random sample 7000 rows to give a prediction.

- The bigger the forest, the lower the error of rate but at the expense of computational time. (Takes about 3 hours to fit the model with 12k rows)



Improve model performance

- Trace the model to check the trees needed to improved the performance of it (trial and error).
- Passed a random sample of the data for each tree (7000) to fit the block with that data.
- Train the classifier with 37 features for each tree this way allowing each feature be checked at least 60 times.
- Switched from bag of words vectorizer to term frequency- inverse document frequency to improve the feature selection when doing dimensionality reduction.



tf idf model

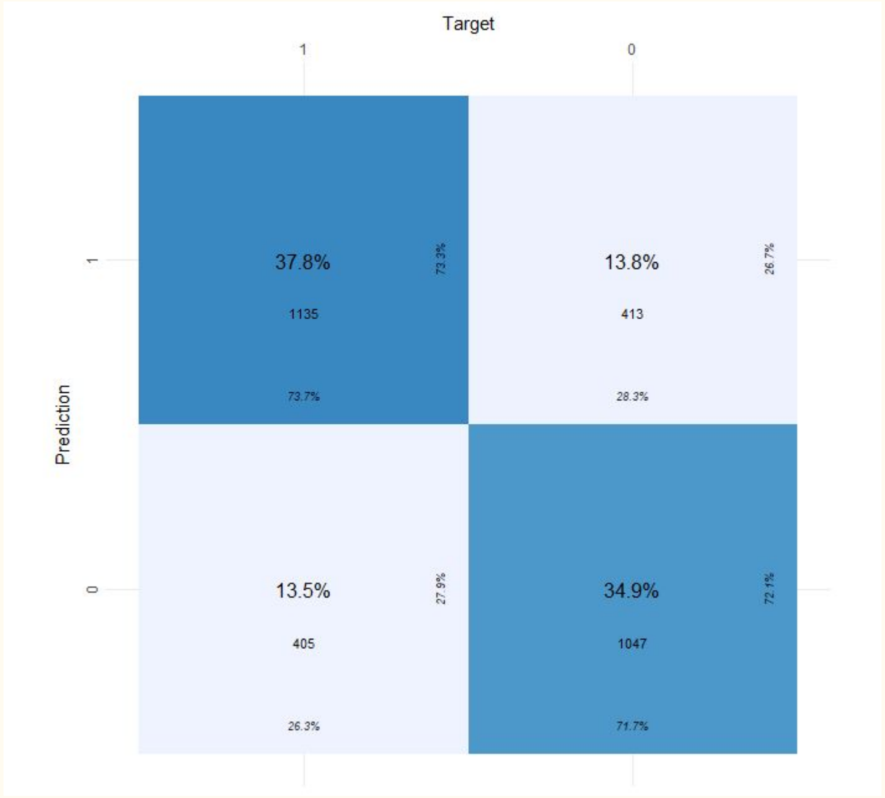
table shows the first 20 songs with a few selected words

can	get	love	know	go	like	just	come	say	good	now	rating
0.06316777	0.03736672	0.00000000	0.00000000	0.07393987	0.00000000	0.07589294	0.00000000	0.08216849	0.04234467	0.00000000	1
0.07201673	0.00000000	0.00000000	0.00000000	0.00000000	0.04549204	0.00000000	0.04645144	0.00000000	0.00000000	0.00000000	0
0.03849463	0.00000000	0.00000000	0.00000000	0.00000000	0.04863313	0.00000000	0.00000000	0.05007372	0.41287949	0.10172019	1
0.00000000	0.01527053	0.00000000	0.00000000	0.00000000	0.08153364	0.00000000	0.00000000	0.03357952	0.00000000	0.00000000	1
0.00000000	0.00000000	0.30392475	0.45715658	0.00000000	0.26084993	0.31008054	0.00000000	0.00000000	0.06920411	0.00000000	0
0.00000000	0.00000000	0.15758341	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.04351739	0.00000000	0.00000000	0
0.04235905	0.00000000	0.07482295	0.00000000	0.00000000	0.00000000	0.07633844	0.00000000	0.08265083	0.00000000	0.00000000	0
0.04272860	0.00000000	0.15095144	0.04730368	0.15004551	0.00000000	0.00000000	0.16536199	0.05558126	0.00000000	0.00000000	0
0.00000000	0.00000000	0.04173486	0.03923538	0.08296877	0.00000000	0.12774051	0.00000000	0.00000000	0.19006176	0.09365017	0
0.00000000	0.08271211	0.02744264	0.02579911	0.05455588	0.00000000	0.05599694	0.03006244	0.03031365	0.03124364	0.21552791	0
0.10437520	0.00000000	0.12291212	0.00000000	0.04072482	0.00000000	0.00000000	0.00000000	0.00000000	0.02332272	0.00000000	1
0.00000000	0.00000000	0.00000000	0.02879358	0.06088811	0.00000000	0.09374465	0.00000000	0.00000000	0.00000000	0.00000000	1
0.05894748	0.10461066	0.03470824	0.00000000	0.00000000	0.03723637	0.00000000	0.07604332	0.11501812	0.00000000	0.03894146	0
0.00000000	0.00000000	0.16122834	0.00000000	0.00000000	0.00000000	0.00000000	0.05887332	0.00000000	0.00000000	0.00000000	0
0.13234965	0.12526571	0.00000000	0.02930415	0.00000000	0.03344143	0.00000000	0.03414669	0.00000000	0.00000000	0.06994551	0
0.07259901	0.00000000	0.00000000	0.06027931	0.00000000	0.00000000	0.04361205	0.02341351	0.09443662	0.07300038	0.11989959	0
0.04268611	0.00000000	0.15080135	0.51982310	0.19986175	0.00000000	0.10257048	0.00000000	0.00000000	0.00000000	0.16919396	0
0.00000000	0.02036086	0.00000000	0.09526272	0.00000000	0.00000000	0.12406057	0.17760798	0.00000000	0.02307331	0.02273808	0
0.06341766	0.00000000	0.00000000	0.14041597	0.14846474	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.16757808	0
0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.16641836	0.08934314	0.00000000	0.00000000	0.00000000	0

Tf_idf vectorizer after we cleaned the strings and removing unnecessary features.

Each column represent the amount of times the word appears in the string while taking into account words that don’t appear frequently in the whole document but in the song.

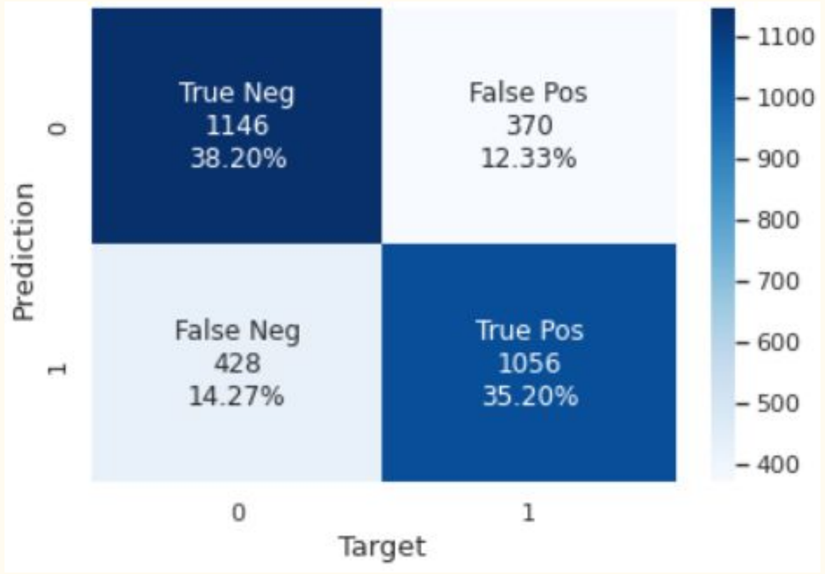
Confusion Matrix and Evaluation



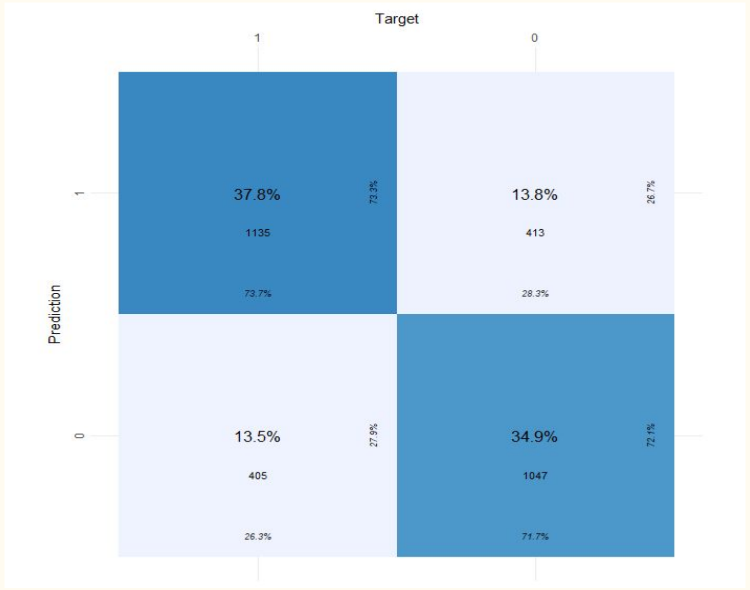
- Accuracy = 73%
  - Amount of correctly predicted classes
- Precision = 72%
  - Amount of predicted negative examples that were correctly labeled.
- Specificity = 74%
  - Amount of predicted positive examples that were predicted correctly.
- Sensitivity/Recall = 72%
  - amount of negative examples that were predicted accurately
- F-score = 72%
  - Harmonic mean of recall and precision

Comparing models

LSTM confusion matrix
Accuracy = 73%



R.F confusion matrix
Accuracy = 73%



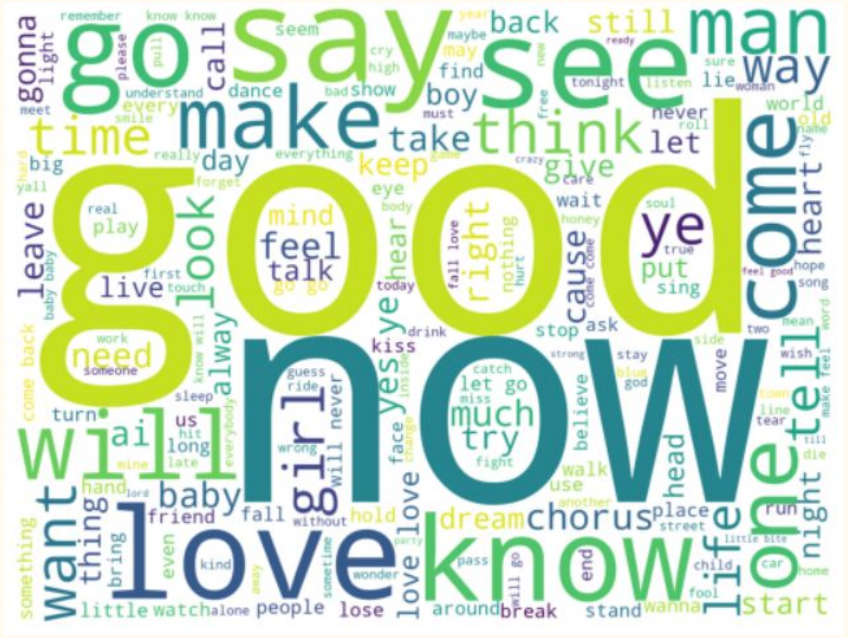
Future Work

- Introducing audio features like tempo, melody,etc. during training.
- Trying bigrams and 3-grams.
- LDA Topic Modeling.
- Trying HuggingFace transformer models.
- Use more data for training using GPU instances.

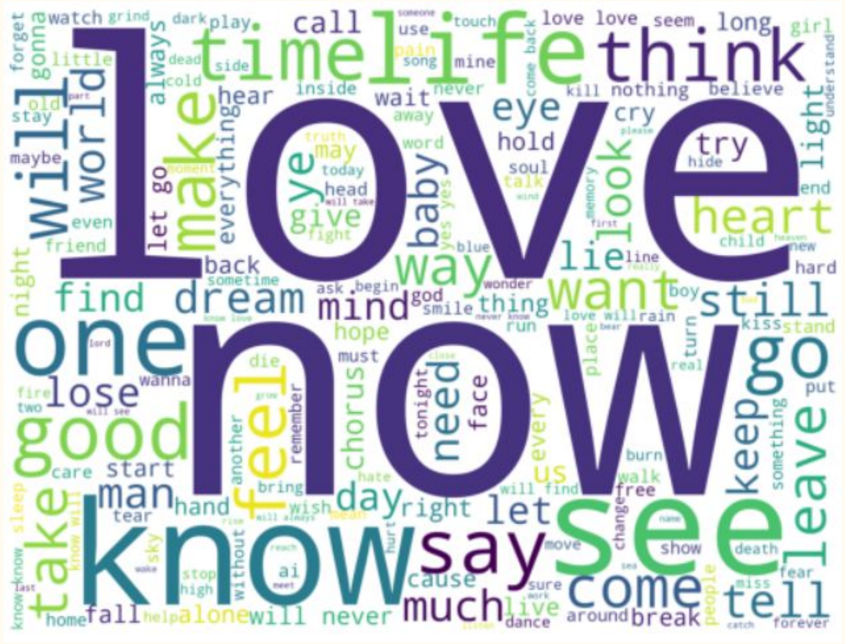
Conclusion

There are many factors that could have prevented the accuracy of our models from passing the 75% threshold. For example labels for tempo, melody, rhythm and all of those audio features could have added bigger weight into the overall net sentiment of the songs like many articles suggest and it could have affected the overall accuracy of both models since we are only taking into account words.

Positive songs wordcloud



Negative songs wordcloud



Thank you