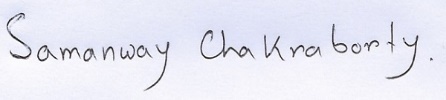
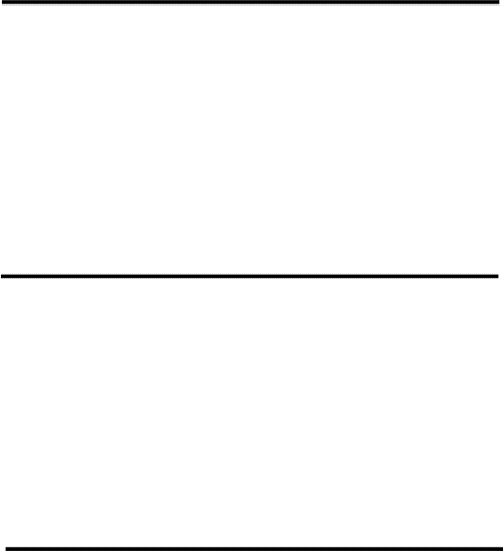


**ACKNOWLEDGEMENT**

I am highly indebted to my project guide Prof. *Sandipan Ganguly* for guiding me and providing constant supervision and necessary information regarding the project & also for support in completing the project. I would also like to thank Mr. Rajib Chakraborty of SNLTR for providing me the annotated corpus and Prof. (Dr.) Dipankar Das of Jadavpur University (Department of Computer Science and Engineering) for guiding me along the way.





*Samanway Chakraborty*

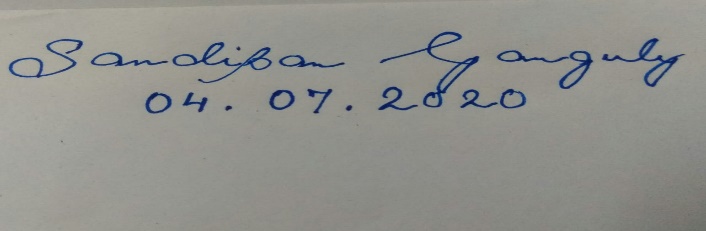
**DEPARTMENT OF COMPUTER APPLICATION**

**HERITAGE INSTITUTE OF TECHNOLOGY, KOLKATA**



**BONAFIDE CERTIFICATE**

Certified that this project report **“Multi-Modal Sentiment Analysis of Rabindra –Sangeet using Deep Learning Techniques”** is the bonafide work of Samanway Chakraborty, student of MCA 6th Semester of Heritage Institute of Technology, Kolkata, who carried out the project work under my supervision, Prof. Sandipan Ganguly in collaboration with SNLTR and Department of Computer Science and Engineering, Jadavpur University.

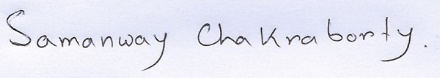




|  |
| --- |
| **Prof. Sandipan Ganguly** |
| Mentor |
| Assistant Professor,Department of Computer Applications |
| Heritage Institute of Technology, Kolkata |
|  |

# **Declaration by Student**

This is to declare that this report has been written by me. No part of the report is plagiarized from other sources. All information included from other sources have been duly acknowledged. I aver that if any part of the report is found to be plagiarized, I’ll take full responsibility for it.





**SIGNATURE**

**Samanway Chakraborty**

Roll No - 30317010038

### **Table of Contents:**

|  |  |  |
| --- | --- | --- |
| Content |  | Page Number |
| **1.Introduction** |  | 6 |
| **2.Data Collection and Processing** |  | 7-8 |
| **3.Features**   * **Lyrics Features** * **Audio Features** |  | 9-12  9  10-12 |
| **4.Feature Extraction** |  | 13 |
| **5.Feature Engineering** |  | 14-15 |
| **6.Classification**   * **Classifiers for Lyrics** * **Classifiers for Audio** |  | 16-28  16-24  24-27 |
| **7.Comparison between audio and lyrics accuracy** |  | 28-31 |
| **8.Data Visualizations** |  | 31-35 |
| **9.Technologies Used** |  | 36 |
| **10.Scope of future work** |  | 37 |

### **ABSTRACT:**

### The voluminous works of Poet Rabindranath Tagore reflects a wide VIBGYOR of his creative persona. Within this great repertoire of creation, the songs he wrote and as well as rendered the tune have a significant place. The entire corpus of these songs which is better known as Rabindra-Sangeet has a number of motifs which are categorized by ‘parjaay’ or class by the poet himself. Each of this class reflects a certain mood of human psyche. We tried to analyse the songs belonging five of the major class or ‘parjaay’ through some of the existing deep learning techniques. We took both text and audio separately and extracted related features to be fed into the machine learning algorithms. We compared the results of these analysis.

### We propose this hitherto unknown means of analysing the works of Tagore will open new avenues to render further deep level analysis of the inherent semantics of his works.

**INTRODUCTION:**

Seldom is it the case that the literary work of an individual embodies the deep rooted Philosophical vision of an entire nation. Poet Rabindranath Tagore and the whole plethora of his creation really signifies such a trait. Among the palette of his creative works the songs have special significance. The poet himself categorized them into a number of class or ‘parjaay’ each belonging to a specific thematic motif. The computational analysis of the lyrics and the tune may establish traits to find out inherent semantic and pragmatic features embedded within. In recent years similar sentimental analysis of natural language text has become an important research topic to analyse the complex semantic features embedded within classical literary texts. We have taken Tagore Songs as they are widely considered as sort of a gold standard of musical verses. This is a primary work that involves analysis of Tagore Songs where lyrics and the musical files (waveforms) taken separately for machine learning analysis. We used five models namely Naive Bayes Classification, Support Vector Machine (SVM), Single Layer Perceptron and Multi-layer Perceptron and Recurrent Neural Network (RNN) to analyze the lyrics. For musical files (.mp3) however we used two models namely RNN and LSTM. Using proper training and test classes the identification of individual parjaay was tested. For dearth of text features from the lyrics data set we also projected certain visualisations depicting (i) the songs with most word count and (ii) most frequent words pertaining to each parjaay.

We have also submitted this work in the 25th international symposium on Frontiers of Research in Speech and Music (FRSM),2020 which is organized by National Institute of Technology,Silchar this year.

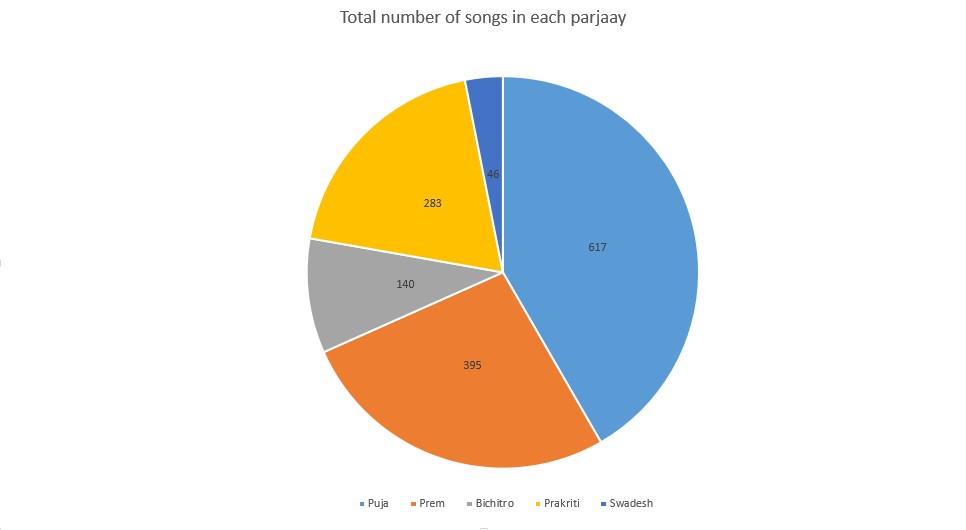
**DATA COLLECTION & PROCESSING**:

It is usual in regards to NLP research to exclude the stop-words that are frequent in textual corpus and do not add any significance in the semantics or sentiment of the text. However, the lyrics of a song are usually shorter in length than normal sentences, and they use a relatively limited vocabulary. They represent the inherent aesthetics and mood of the very song rendered.

For this study, we have acquired the benchmarked dataset from the official website of Society of Natural Language Training and Research or SNLTR (https://rabindra-rachanabali.nltr.org/node/6585).SNLTR is a reputed Goverment Institution which among other myriad of activities focuses on archiving the Corpus of Bengali Literary tradition in Unicode compatible format. We also used the Shruti-Gitobitan archive (http://shurerdhara.ac.bd/shruti-gitobitan/) consisting of the contents of 22 DVDs that encompasses the audio sample of the entire body of songs rendered by Rabindranath Tagore.

We have sampled over the total of 1481 no. of songs. Below is the list of songs belonging to each parjaay (class) respectively.

|  |  |
| --- | --- |
| Parjaay | No. of Songs |
| Puja | 617 |
| Prem | 395 |
| Prakriti | 283 |
| Bichitro | 140 |
| Swadesh | 46 |



Since our study is based on both lyrics and audio, I collect the data and manually label the data. After that, I train this dataset with different classifiers and make a comparative study of the results achieved.

Below are the manually labelled information for each parjaay.

|  |  |
| --- | --- |
| Parjaay Name | Label |
| Prakriti | 1 |
| Puja | 2 |
| Bichitro | 3 |
| Prem | 4 |
| Swadesh | 5 |

## **FEATURES:**

*LYRICS FEATURES*:

##### **TERM FREQUENCY (TF):**

TF us the frequency of the word in each document in the corpus. It is the ratio of number of times the word appears in a document compared to the total number of words in that document. It increases as the number of occurrences of that word within the document increases. Each document has its own TF.

##### **INVERSE DOCUMENT FREQUENCY (IDF):**

Inverse Document Frequency (IDF) is a measure of how common or rare a word is in the entire document set. The closer it is to 0, the more common a word is. This metric can be calculated by taking the total number of documents, dividing it by the number of documents that contain a word, and calculating the logarithm. So, if the word is very common and appears in many documents, this number will approach 0. Otherwise, it will approach 1.

##### **TERM FREQUENCY-INVERSE DOCUMENT FREQUENCY (TF-IDF):**

For the classification of the models, we have used this feature in most cases. TF-IDF is the multiplication of the scores of TF and IDF. The higher the score, the more relevant that word is in that particular document. We need to transform text into numbers, otherwise known as text vectorization, it is a fundamental step in the process of machine learning to for analyzing text and different vectorization algorithms will drastically end results. TF-IDF score can be fed to algorithms like Naïve Bayes and Support Vector machine quite easily, and thus greatly improving the results of more basic methods like word count.

* **N-Gram:**

An n-gram is simply any sequence of *n* tokens (words). Consequently, given the following review text - *“Absolutely wonderful - silky and sexy and comfortable”*, we could break this up into:

* **1-grams**: Absolutely, wonderful, silky, and, sexy, and, comfortable
* **2-grams**: Absolutely wonderful, wonderful silky, silky and, and sexy, sexy and, and comfortable
* **3-grams**: Absolutely wonderful silky, wonderful silky and, silky and sexy, and sexy and, sexy and comfortable etc.

*AUDIO FEATURES:*

* **MFCC:**

This feature is one of the most important method to extract a feature of an audio signal and is used majorly whenever working on audio signals. The mel frequency cepstral coefficients (MFCCs) of a signal are a small set of features (usually about 10–20) which concisely describe the overall shape of a spectral envelope.

Mfccs=librosa.feature.mfcc(x,sr=sr)  
print(mfccs.shape)#Displaying the MFCCs:  
librosa.display.specshow(mfccs, sr=sr, x\_axis='time')

.mfcc is used to calculate mfccs of a signal.

By printing the shape of mfccs you get how many mfccs are calculated on how many frames. The first value represents the number of mfccs calculated and another value represents a number of frames available.

* **Spectrogram:**

A spectrogram is a visual representation of the [spectrum](https://en.wikipedia.org/wiki/Spectral_density) of [frequencies](https://en.wikipedia.org/wiki/Frequencies) of [sound](https://en.wikipedia.org/wiki/Sound) or other signals as they vary with time. It’s a representation of frequencies changing with respect to time for given music signals.

.stft converts data into short term Fourier transform. [STFT](https://www.youtube.com/watch?v=g1_wcbGUcDY) converts signal such that we can know the amplitude of given frequency at a given time. Using STFT we can determine the amplitude of various frequencies playing at a given time of an audio signal. .specshow is used to display spectrogram.

* **Zero Crossing Rate:**

The Zero Crossing Rate is the rate of sign-changes along a signal, i.e., the rate at which the signal changes from positive to negative or back. This feature has been used heavily in both speech recognition and music information retrieval. It usually has higher values for highly percussive sounds like those in metal and rock.

* **Spectral Centroid:**

It indicates where the” centre of mass” for a sound is located and is calculated as the weighted mean of the frequencies present in the sound. If the frequencies in music are same throughout then spectral centroid would be around a centre and if there are high frequencies at the end of sound then the centroid would be towards its end.

* **Tempo and Beats per minute:**

The overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece and derives directly from the average beat duration.

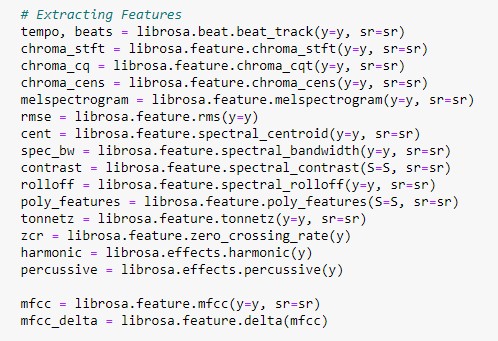
* **Spectral Roll-off:**

The roll-off frequency is defined for each frame as the center frequency for a spectrogram bin such that at least roll percent (0.85 by default) of the energy of the spectrum in this frame is contained in this bin and the bins below. This can be used to, e.g., approximate the maximum (or minimum) frequency by setting roll percent to a value close to 1 (or 0).

* **RMSE:**

We can compute root-mean-square (RMS) value for each frame, either from the audio samples *y* or from a spectrogram *S*.

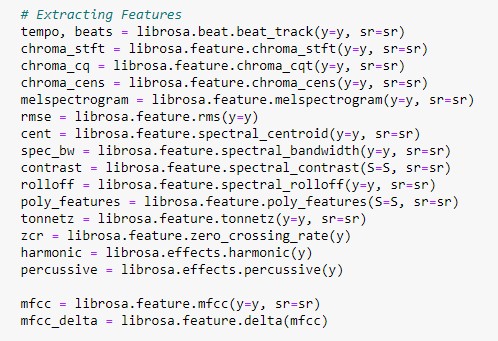
Computing the RMS value from audio samples is faster as it doesn’t require a STFT calculation. However, using a spectrogram will give a more accurate representation of energy over time because its frames can be windowed, thus prefer using *S* if it’s already available.



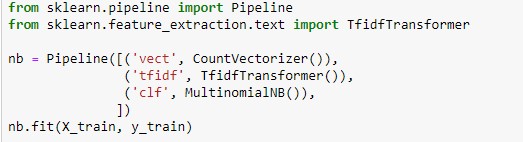
FEATURE EXTRACTION:

**Below are the code snippets of feature extraction for lyrics and audio respectively.**

* **Audio features extraction code:**



* **Lyrics features extraction code:**



**FEATURE ENGINEERING OF AUDIO FEATURES:**

*“As the number of features or dimensions grows, the amount of data we need to generalize accurately grows exponentially.”*

*---- Charles Isbell, Professor, School of Interactive Compution, Georgia Tech*

There are two options to reduce dimensionality:

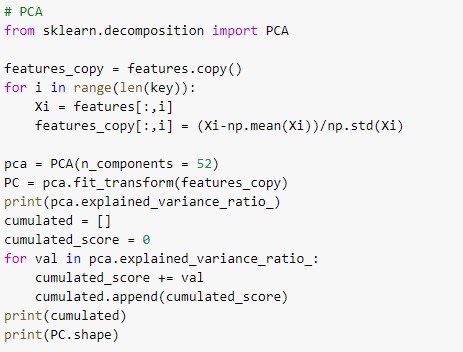
1. Feature elimination: we remove some features directly.
2. Feature extraction: we keep the important fraction of all the features. We apply PCA to achieve this. Note that PCA is not the only method that does the feature extraction.

And here we have chosen the second approach which is using PCA or **P**rincipal **C**omponent **A**nalysis.

PCA is a dimensionality reduction that **identifies important relationships** in our data, **transforms the existing data** based on these relationships, and then **quantifies the importance** of these relationships so we can keep the most important relationships and drop the others. To remember this definition, we can break it down into four steps:

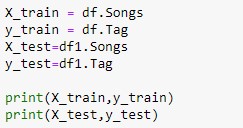
1. We identify the relationship among features through a [**Covariance Matrix**](https://en.wikipedia.org/wiki/Covariance_matrix)**.**
2. Through the linear transformation or [**Eigen decomposition**](https://en.wikipedia.org/wiki/Eigendecomposition_of_a_matrix) of the Covariance Matrix, we get [**eigenvectors** and **eigenvalues**](https://en.wikipedia.org/wiki/Eigenvalues_and_eigenvectors).
3. Then we transform our data using Eigenvectors into principal components.
4. Lastly, we quantify the importance of these relationships using Eigenvalues and keep the important principal components**.**

**The code for principal component analysis is:**



## **CLASSIFICATION:**

After manually labelling and preprocessing of the data we perform the training and testing split on the dataset. After splitting, the next steps include feature engineering. For the lyrics part, I will convert our text documents to a matrix of token counts (CountVectorize), and then transform a count matrix to a normalized TF-IDF representation using TF-IDF transformer and for the audio after creating csv file for each parjaay I will merge them into a single csv file for feature engineering

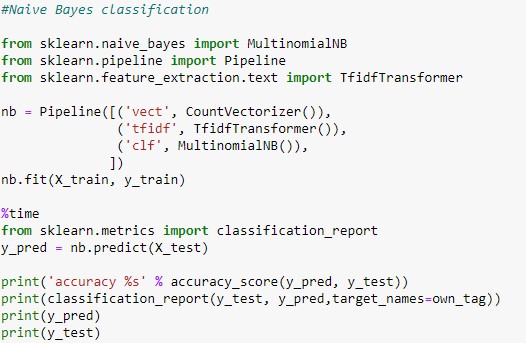


And the splitting ratio I have taken is 70-30, which is 70% is for training and 30% is for testing.

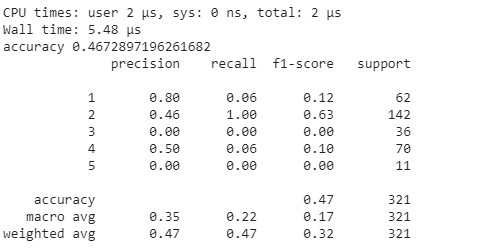
*CLASSIFIERS FOR LYRICS:*

#### **NAÏVE BAYES CLASSIFIER**

After we have our features, we will now train a classifier to try to predict the parjaay of a song based on its lyrics. We will first use the Naïve Bayes classifier, which provides a nice baseline for this task. The library ‘scikit- learn’ includes several variants of this classifier; the one most suited for text is the multinomial variant.

To make the vectorizer => transformer => classifier easier to work with, I will use the ‘pipeline’ class in ‘scikit-learn’ that behaves like a compound classifier. Since we have 5 classes of data, the multi-class classification variant of Naïve Bayes is used.

The output which is obtained is:

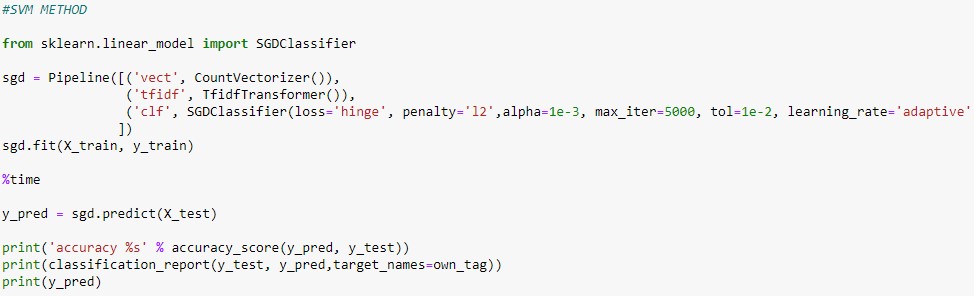


We can clearly see, that the accuracy which has been obtained is 47%. Now we apply another classify and compare the results we get from it.

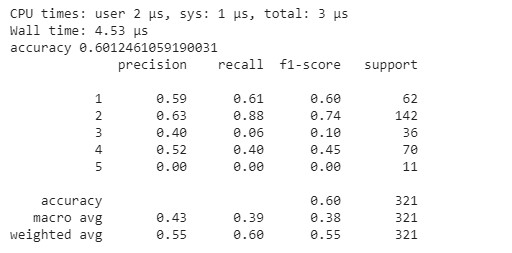
#### **LINEAR SUPPORT VECTOR MACHINE (SVM):**

Linear SVM is widely regarded as one of the best text classification algorithms. It is because most of the text classification problems are usually linearly separable and SVM works incredibly well with such data. The linear kernel is good when there are a lot of features. That is because mapping the data to a higher dimensional space does not really improve the performance. In text classification, both the numbers of instances (document) and features are large. Additionally, training a SVM with a linear kernel is considerably faster than other kernels.

In Python, for creating a linear SVM model, the library ‘SGDClassifier’ is imported from ‘sklearn’. The code for SGD Classifier is given below:



The output which is obtained is:

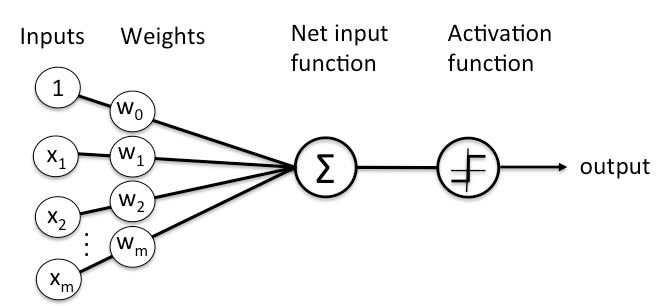


Thus from the output we can see that the Linear SVM model has an accuracy of 60% which is considerably greater than the previous model.

#### **Perceptron and Multi-layer perceptron:**

A perceptron is a neural network unit (an artificial neuron) that does certain computations to detect features or business intelligence in the input data.

A Perceptron is an algorithm for supervised learning of binary classifiers. This algorithm enables neurons to learn and processes elements in the training set one at a time.



There are two types of Perceptron: Single layer and Multilayer.

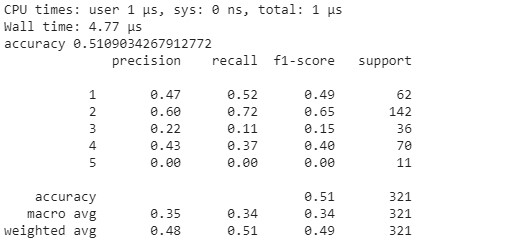
Single layer Perceptron can learn only linearly separable patterns.

Multilayer Perceptron or feedforward neural networks with two or more layers have the greater processing power.

The code for single layer perceptron is:

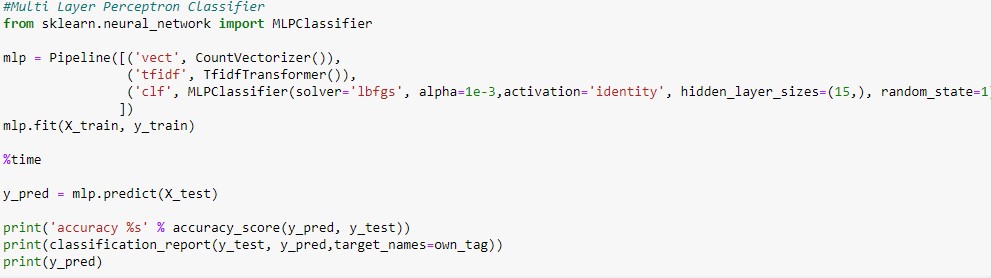


The output which is obtained from single layer perceptron is:

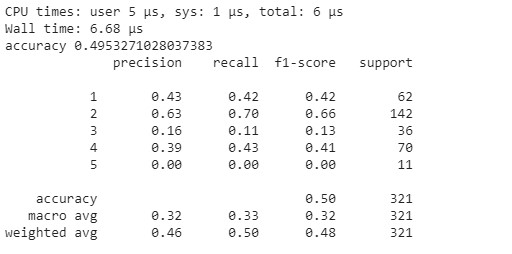


Thus the accuracy is 51%, which is less than the accuracy obtained by linear SVM, but it is still better than the Naïve Bayes Model.

The code for multilayer perceptron is:



The output which is obtained from single layer perceptron is:

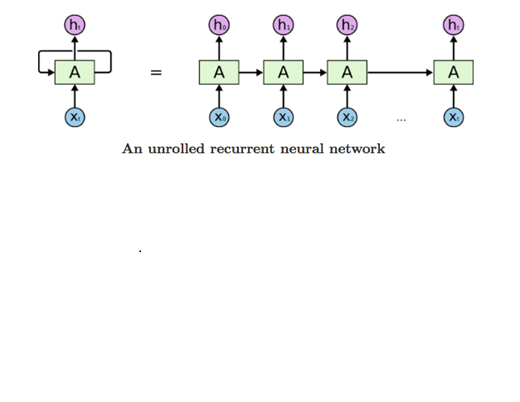


Thus the accuracy is 50%, which is more than accuracy obtained by single layer perceptron.

#### **RNN (Recurrent Neural Network):**

Recurrent Neural Network is a generalization of feedforward neural network that has an internal memory. RNN is recurrent in nature as it performs the same function for every input of data while the output of the current input depends on the past one computation. After producing the output, it is copied and sent back into the recurrent network. For making a decision, it considers the current input and the output that it has learned from the previous input.

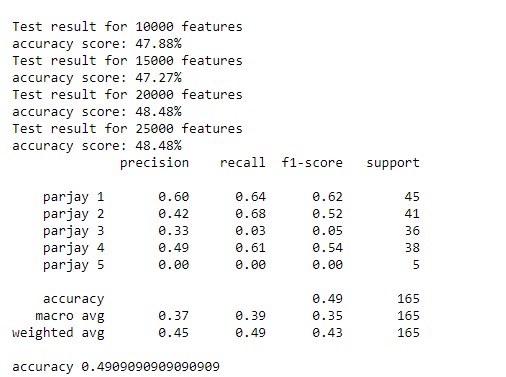
Unlike feedforward neural networks, RNNs can use their internal state (memory) to process sequences of inputs. This makes them applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition. In other neural networks, all the inputs are independent of each other. But in RNN, all the inputs are related to each other.



The code for RNN is:



The output which is obtained from single layer perceptron is:



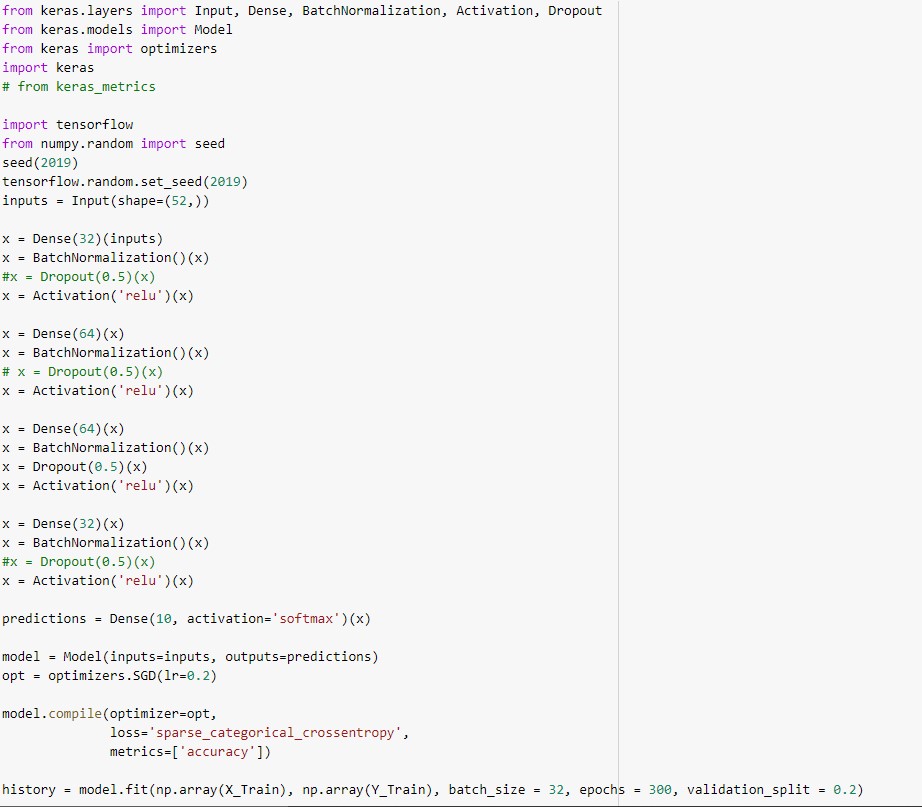
Thus the accuracy is 49%, which is less than the accuracy obtained by linear SVM, but it is still better than the Naïve Bayes Model.

*CLASSIFIERS FOR AUDIO:*

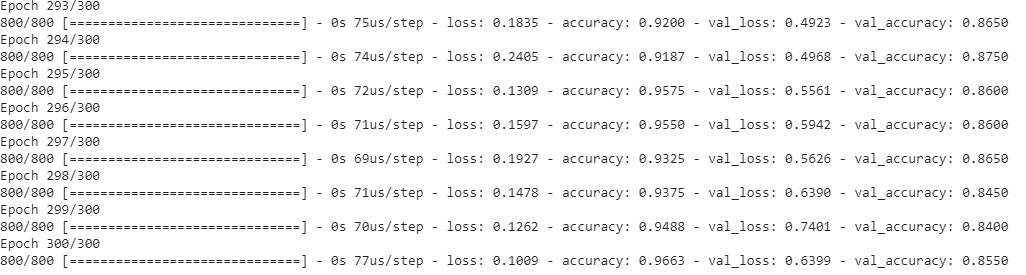
* **RNN(Recurrent Neural Network):**

We’ve used RNN in both lyrics and audio to make a common ground where we can compare them without giving advantage to either lyrics or audio.

The code for RNN for audio is:



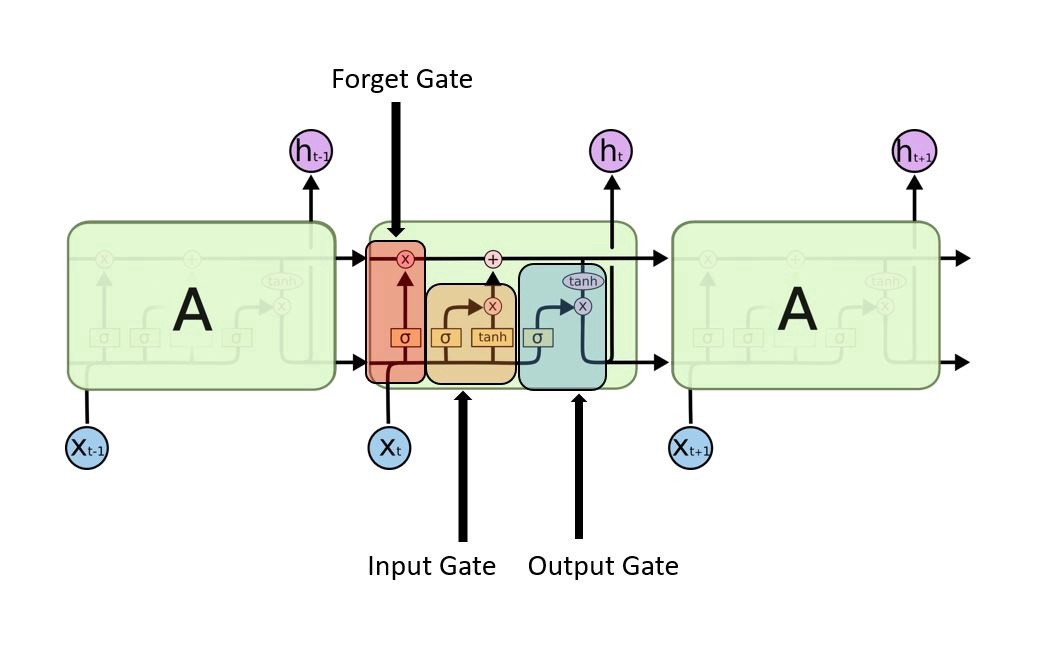
And the output we’ve obtained from this is:



As we can see the accuracy obtained from this is nearly 96%. Which is far better than what we’ve obtained in lyrics.

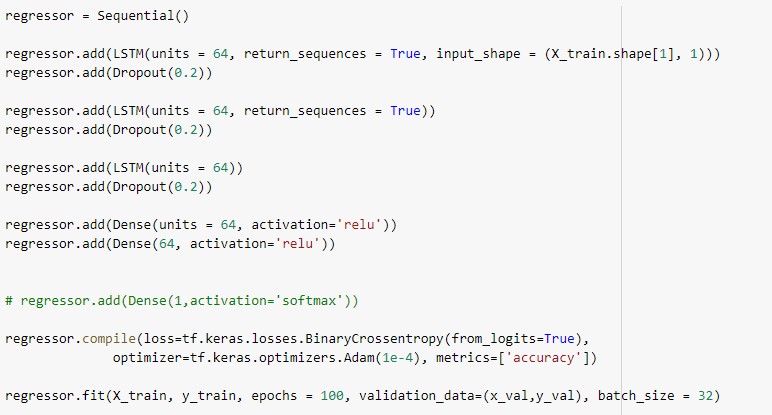
* **LSTM(Long Short Term Memory):**

Long Short-Term Memory (LSTM) networks are a modified version of recurrent neural networks, which makes it easier to remember past data in memory. The vanishing gradient problem of RNN is resolved here. LSTM is well-suited to classify, process and predict time series given time lags of unknown duration. It trains the model by using back-propagation. In an LSTM network, three gates are present:

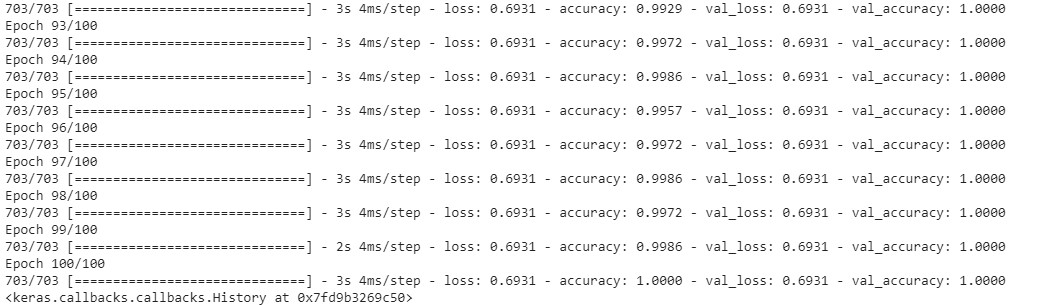


**A Long Short Term Memory Network**

The code for LSTM is:



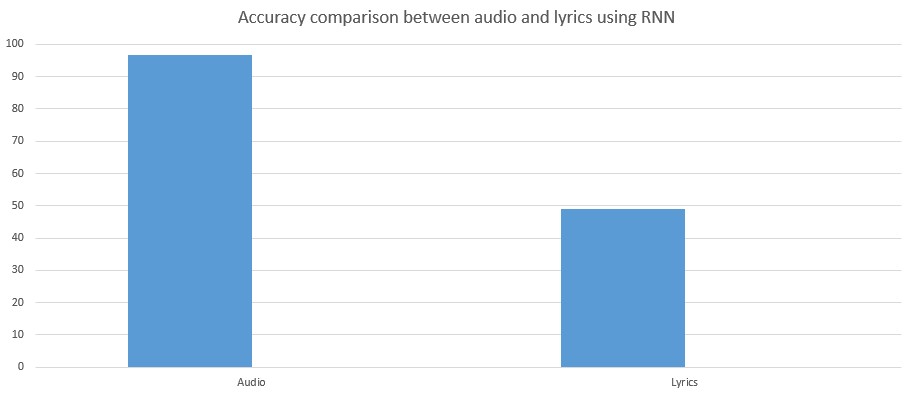
And the output obtained from this is:



As we can see the accuracy obtained is 100%, which is better than RNN.

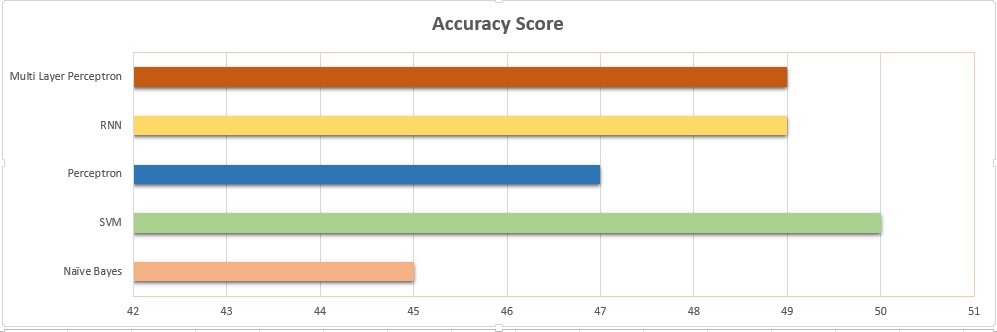
**COMPARISON:**

After classifying lyrics and audio using different classifiers, we compared the outcomes taking one common classifier we used for both lyrics and audio which is RNN.



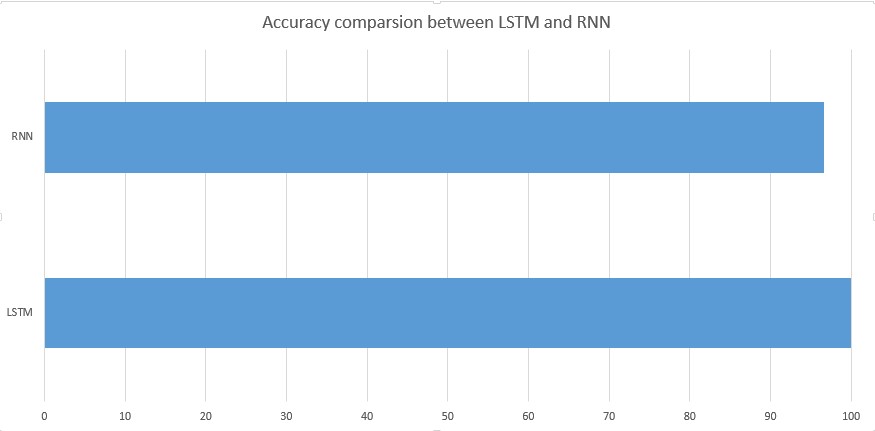
As we can see RNN gives way more accuracy taking audio data as its input.

Other than that we have compared the accuracy of all 5 models which we used to classify lyrics.

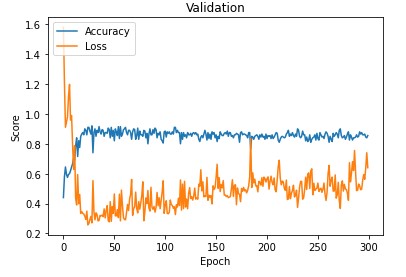
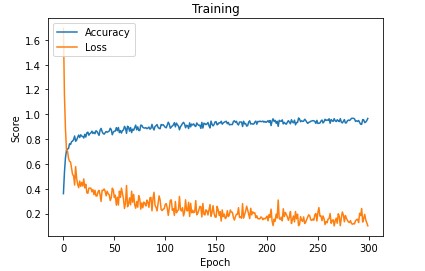


Accuracy score comparison of various classifiers taking lyrics as input

And accuracy comparison of 2 models used for audio is:

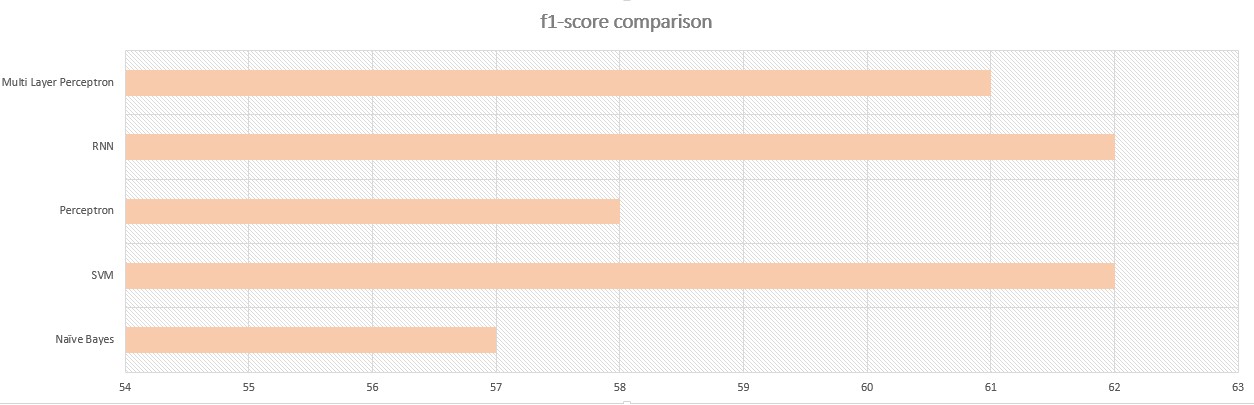


Accuracy score comparison of various classifiers taking audio as input

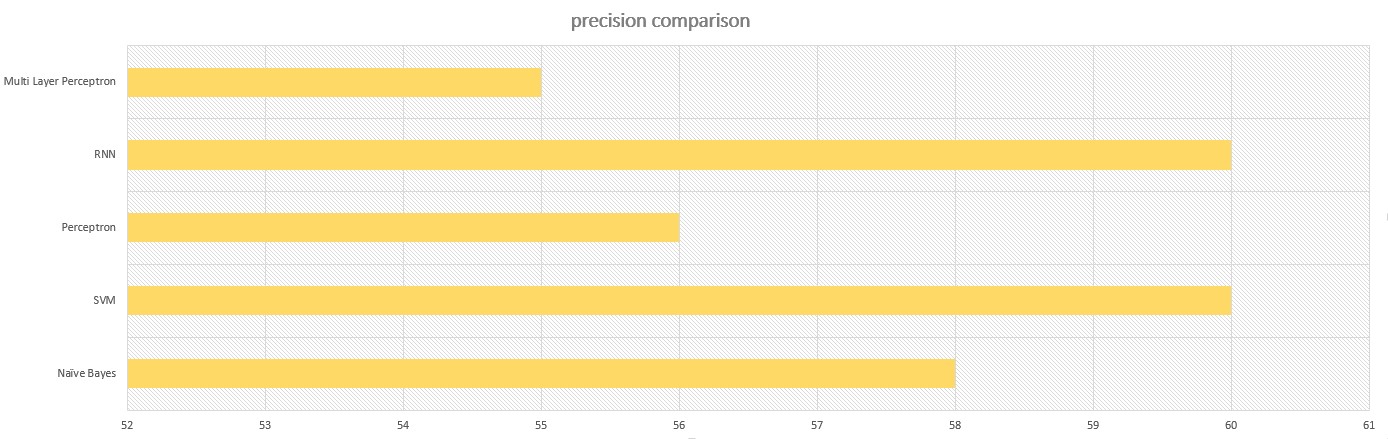


**Value loss and accuracy comparison of training data** **Value loss and accuracy comparison of validation data**

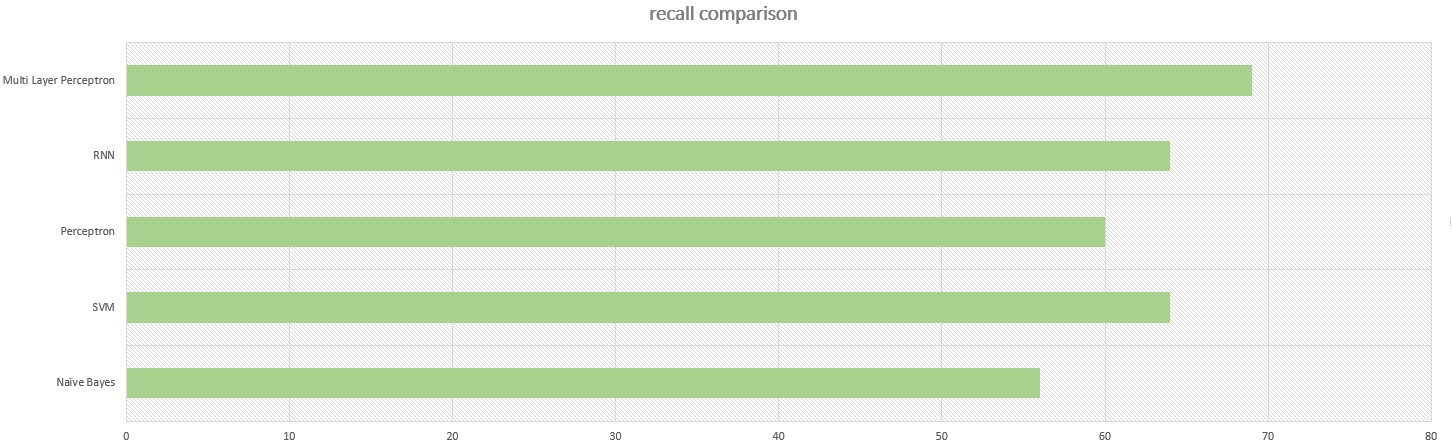
We have also done the comparison of various parameters like f1-score, precision and recall of various classifiers while lyrics was taken as input. The figures are as follows.



f-1 score comparison of different classifiers taking lyrics as input



Precision comparison of different classifiers taking lyrics as input

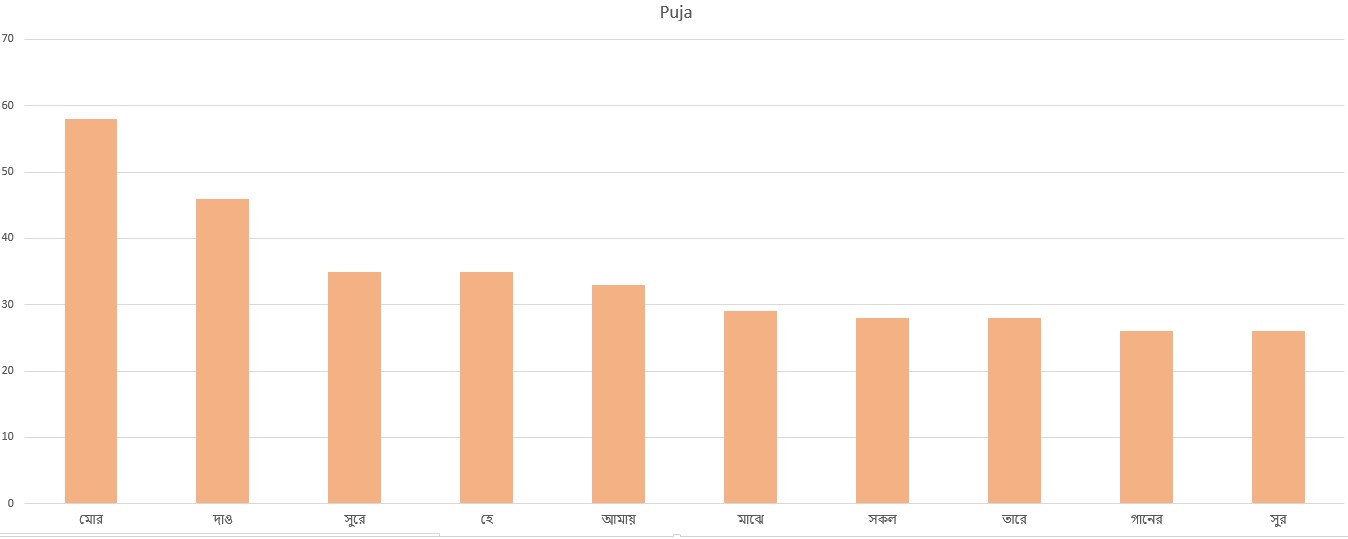


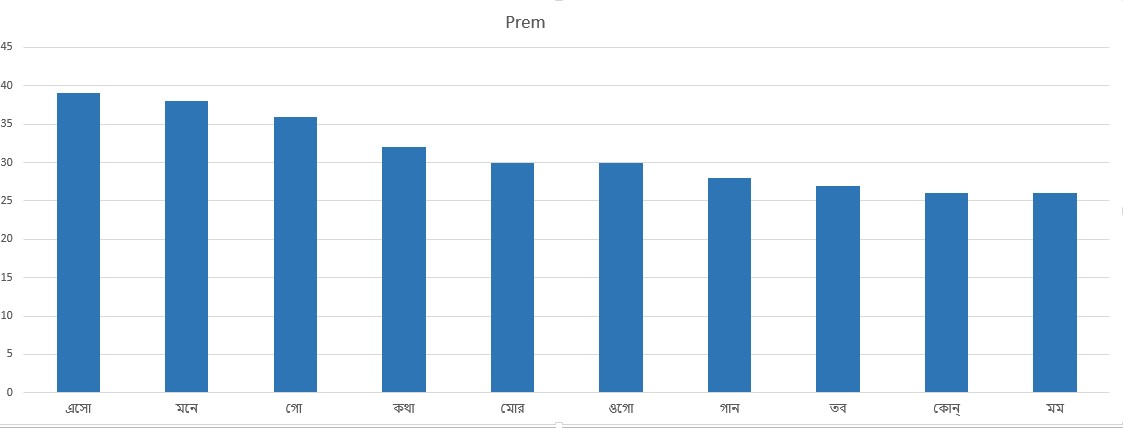
Recall comparison of different classifiers taking lyrics as input

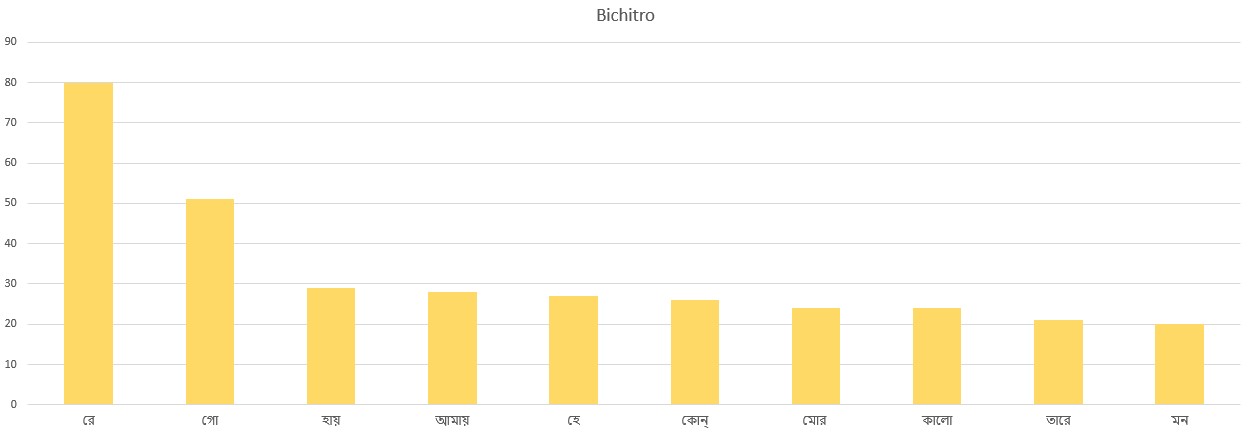
## **DATA VISUALIZATION:**

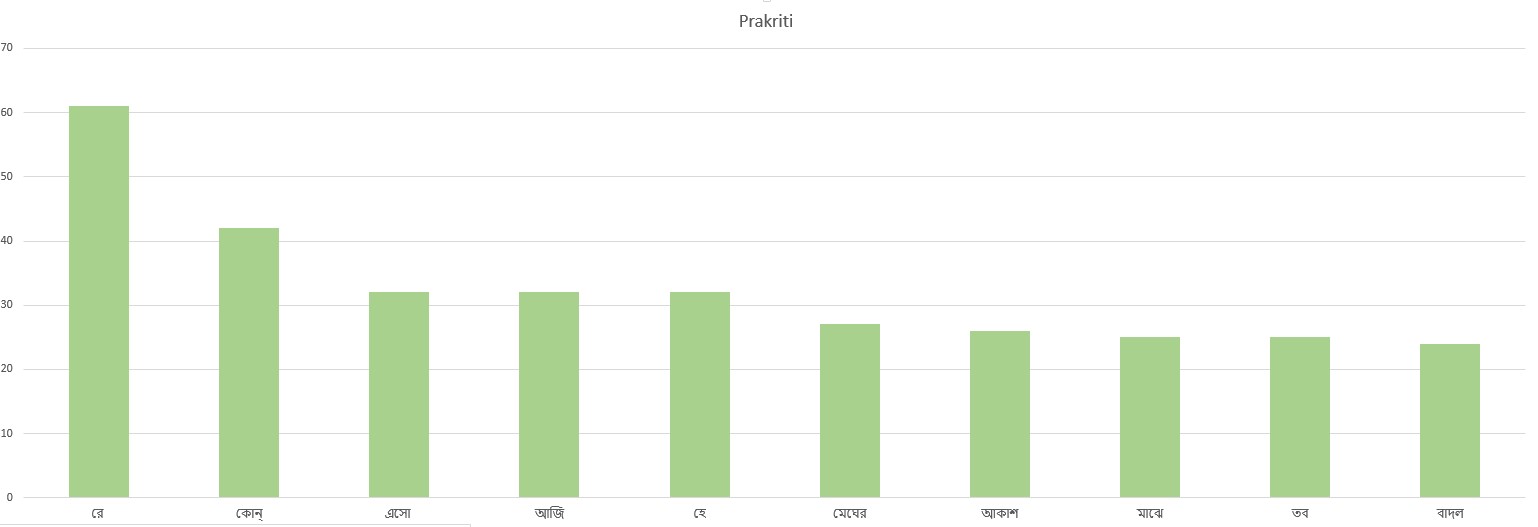
We try to draw an insight from the different kinds of visualization which are possible with the dataset.

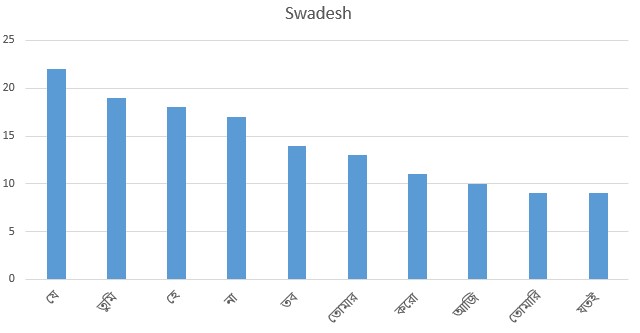
* At first we’ve found out top 10 words from each parjaay from the lyrics dataset. Below are the visualizations.



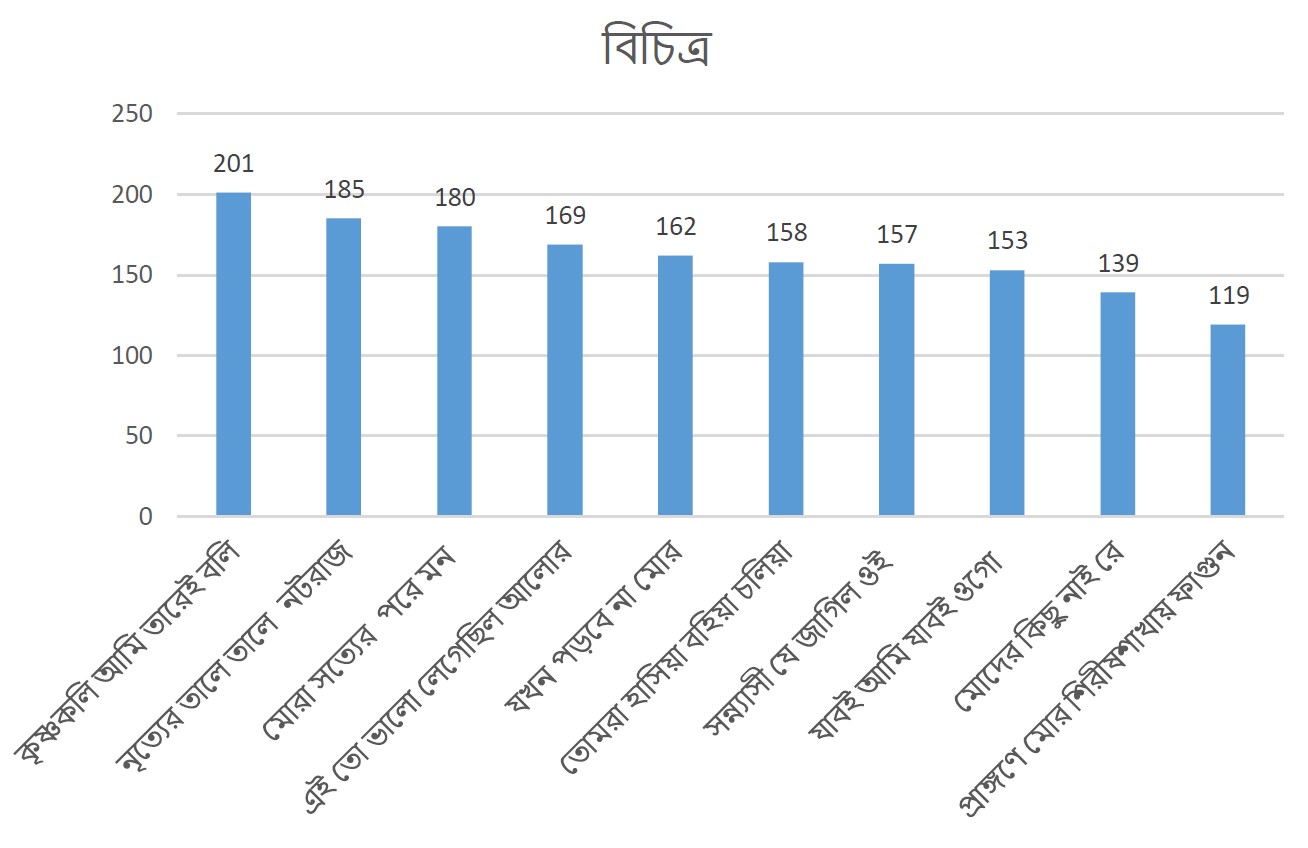


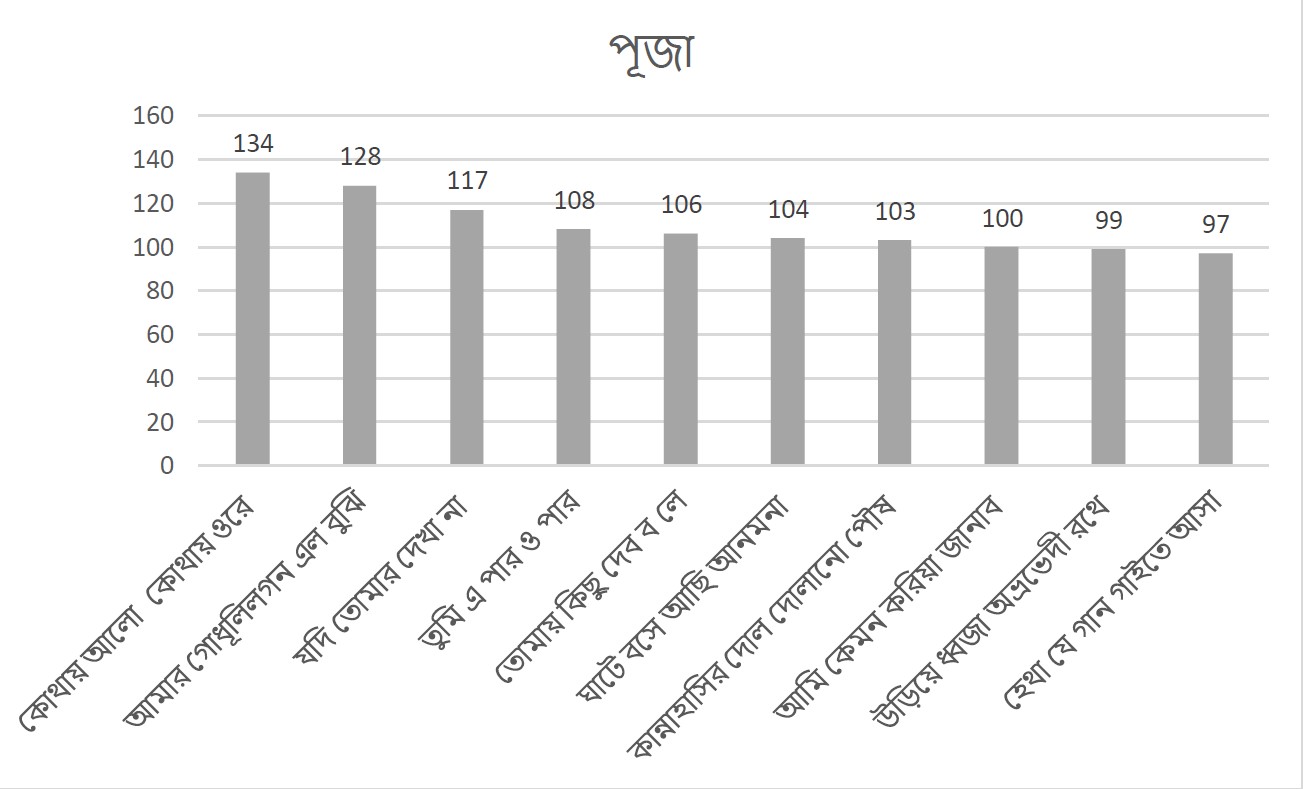
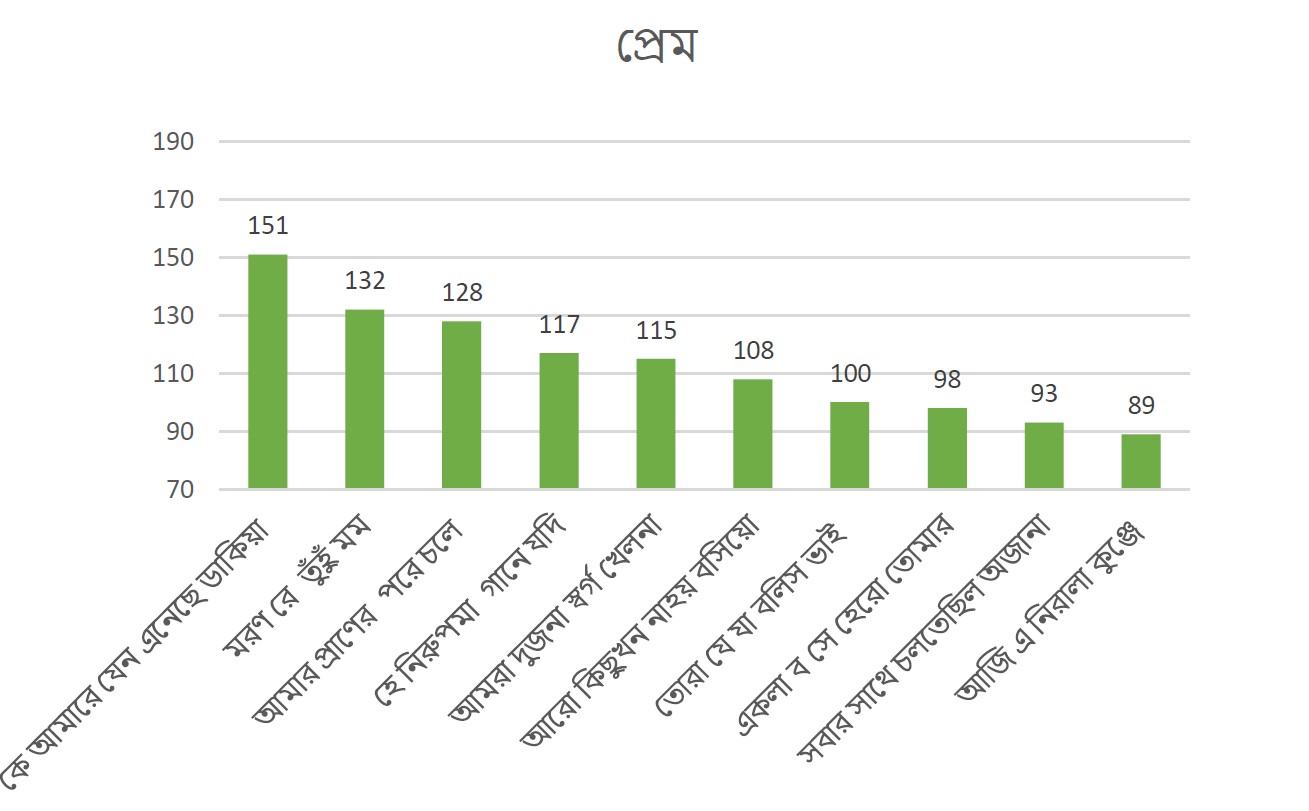


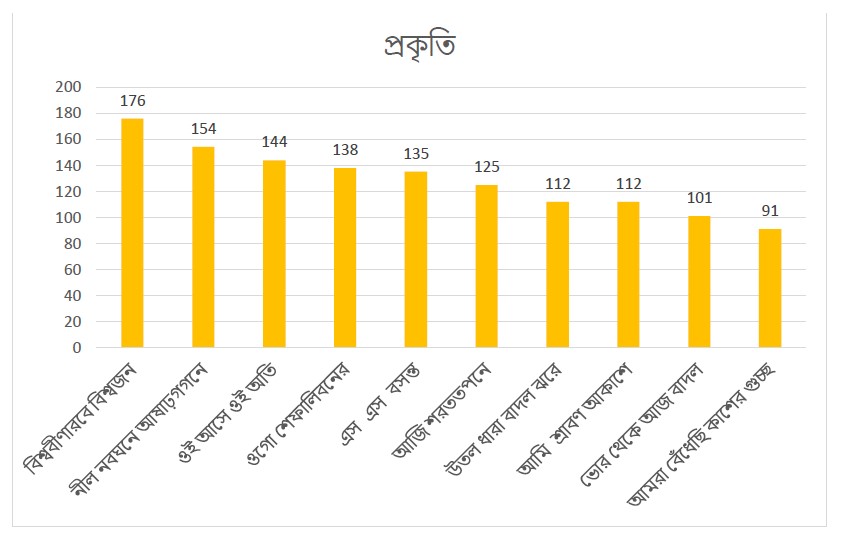


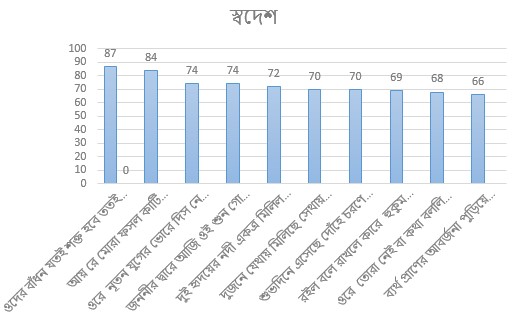


* Another insight we have drawn is top 10 songs from each parjaay in terms of word count. The visualizations are as follows.









**TECHNOLOGIES USED:**

**Editors used to write code:**

* + Jupyter Notebook
  + Spyder
  + Google Colab (For Neural Network)

**Tools used for dataset creation:**

* + Microsoft Excel
  + Google Spreadsheet

**The code is written in Python. So the libraries used are:**

* Pandas
* Numpy
* Gensim
* NLTK
* Sklearn
* Matplotlib
* Tensorflow
* Keras
* Librosa
* PyAudio
* Seaborn

**FUTURE SCOPE:**

1. We have started working on a web page that will be the user interface where the user may provide the lyrics of a song and can select the classifier using which the parjaay prediction will be done. There will also be an option to upload an audio and our model will predict the parjaay of the song.
2. In this project we have taken audio and lyrics of the songs. But we also have started including the notations of the songs that will extend the modal dimension of the work.
3. We are also exploring the possibility of generation of different musical genres using deep learning models like LSTM.
4. We will extend the present work by analysing the works of literary geniuses of Bengal through deep learning based on voluminous samples of their literary corpus which is already been encoded in unicode format.

REFERENCES:

1. Sutcliffe, Richard, Fox, Chris, Root, L., Deane, Hovy, Eduard, &amp; Lewis, Richard, ‘The C@merata Task at MediaEval 2015: Natural Language Queries on Classical Music Scores’, In Proceedings of the MediaEval 2015 Workshop, September 14-15, 2015, Wurzen, Germany.
2. Xie, Stanley, Rastogi, Ruchir &amp; Chang, Max, ‘Deep Poetry: Word-Level and Character-Level Language Models for Shakespearean Sonnet Generation’.
3. Tanasescu, Chris, Kesarwani, Vaibhav &amp; Inkpen, Diana, ‘Metaphor Detection by Deep Learning and the Place of Poetic Metaphor in Digital Humanities’, 2018, Association for the Advancement of Artificial Intelligence.
4. Kotecha, Nikhil, Young, Paul, ‘Generating Music using an LSTM Network’, E4040

Fall2017 TONE report,Columbia University

1. Patra, Gopal, Braja, Das, Dipankar &amp; Bandhopadhyay, Sivaji, ‘Mood Classification of Hindi songs based on lyrics’, In Prcoceeding of the International Conference on Natural Language Processing (ICON- 2015),IIITM-K) Trivandrum, India
2. Patra, Gopal, Braja, Das, Dipankar &amp; Bandhopadhyay, Sivaji, ‘Multimodal Mood Classification - A Case Study of Differences in Hindi and Western Songs’, In Proceeding Of COLING 2016, the International Conference on Computational Linguistics, Osaka, Japan
3. Article on Audio data analysis using deep learning with python,

https://www.kdnuggets.com/2020/02/audio-data-analysis-deep-learning-python-part-1.html

1. Article on Audio feature extraction using Librosa, https://github.com/subho406/Audio-Feature-Extraction-using-librosa/blob/master/Song%20Analysis.ipynb
2. Corpus of the Works of Tagore from the Website of Society of Natural Language

Training and Research, https://rabindra-rachanabali.nltr.org/

1. Shruti Gitabitan Archieve, http://shurerdhara.ac.bd/shruti-gitobitan/