# ACKNOWLEDGEMENT

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

We are also thankful to the Department of Computer Applications for providing us easy access to the project laboratory and other facilities.

*Divyarup Ghosh Dastidar*

*Samanway Chakraborty*

*Alif Ansari*

**DEPARTMENT OF COMPUTER APPLICATION**

# HERITAGE INSTITUTE OF TECHNOLOGY, KOLKATA

## BONAFIDE CERTIFICATE

Certified that this project report **"Parjaay (Genre) Classification and Data Visualization of Rabindra-Sangeet through Machine Learning"** is the bonafide work of *Divyarup Ghosh Dastidar*, *Samanway Chakraborty*, *Alif Ansari*, students of MCA 5th Semester of Heritage Institute of Technology, Kolkata, who carried out the project work under the supervision of Prof. *Sandipan Ganguly*.

**Prof. (Dr.) Souvik Basu**
Head of Department (HOD),
Department of Computer Applications
Heritage Institute of Technology, Kolkata

**Prof. *Sandipan Ganguly***
Mentor
Department of Computer Applications
Heritage Institute of Technology, Kolkata

**EXAMINER**

# DECLARATION BY STUDENTS

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

**SIGNATURE**

***Divyarup Ghosh Dastidar***

Roll No - 30317010016

**SIGNATURE**

***Samanway Chakraborty***

Roll No - 30317010038

**SIGNATURE**

***Alif Ansari***

Roll No - 30317010006

# TABLE OF CONTENTS:

# ABSTRACT:

As people have access to increasingly large music data, music classification becomes critical in music industry. In particular, automatic genre classification is an important feature in music classification and has attracted much attention in recent years. In this project, we perform a detailed study of the textual data of Rabindra-Sangeet Bengali songs.

Firstly, we propose a multi genre (Parjaay) classification of Bengali Rabindra-Sangeet songs using the lyrics of the songs using 4 different kinds of classifiers (Naïve Bayes, Logistic Regression, Linear Support Vector Machine and CNN) and compare the accuracy achieved by each of them. The feature that has been used is TF-IDF (Term Frequency-Inverse Document Frequency). We use simple techniques to extract and pre-process the collected textual data from the dataset. The textual data is in the form of Bengali Unicode text.

Secondly, we also have done data visualizations with the dataset, which includes the most common words of each genre (Parjaay), the top 10 songs in each parjay in terms of word count, as well as the top songs that are related to some specific words.

Lastly, we have created an interactive web page to display the results we have achieved so far in this study.

# INTRODUCTION:

Music has always been an important part in Bengali culture and tradition, specifically Rabindra-Sangeet, which has always been an integral part of it. These songs of Rabindranath Tagore are closely intertwined with the Bengali community even now, after almost 70 years of his demise, and they will continue to be a part of the Bengali culture. In the era of big data, people are faced with a huge amount of music resources and thus the difficulty in organizing and retrieving music data. To solve the problem, music classification and recommendation systems are developed to help people quickly discover music that they would like to listen. Generally, music recommendation systems need to learn users' preferences of music genres for making appropriate recommendations. Since not much work regarding Rabindra-Sangeet songs has been done in terms of data analysis and classification, doing an in-depth study of the huge collection of Rabindra-Sangeet songs for future use has always been the main motivation for doing this project.

In this project, we mainly focus on the genre classification of songs. Extensive work has been done on music genre classification based on acoustic features of a song, e.g., the instrumental accompaniment, the pitch and the rhythm of the song. Nevertheless, little attention has been paid to song classification based on a song's lyrics, especially in our case, Rabindra Sangeet, which only include non-acoustic features. This project explores the potential of classifying a song's genre (Parjaay) based on its lyrics. From here on we will refer to Rabindra-Sangeet genres as parjaay.
The specifics of how the data has been collected and processed is discussed in details in the next section. We have used 4 different types of classifiers and we focus on the accuracy obtained by each of them, and why the model accuracy differs between each of them. Our main idea is to extract the information from a song's lyrics and identify features (In our case, TF-IDF) that help music genre classification.

We also try to present a deeper understanding of statistical viewpoint of the textual data, wherein we show different kinds of visualizations which include:

- The top 10 common words which are associated with the whole dataset of 740 songs as well as the most common words associated with each parjaay.

- The maximum, minimum as well as the average word count associated with the lyrics of each genre, and graphical visualizations of the top 10 and the least 10 songs in terms of word count.

- We also choose a few specific words, and find out the top 10 songs which are associated with those words based on the term frequency of the word in the lyrical corpus.

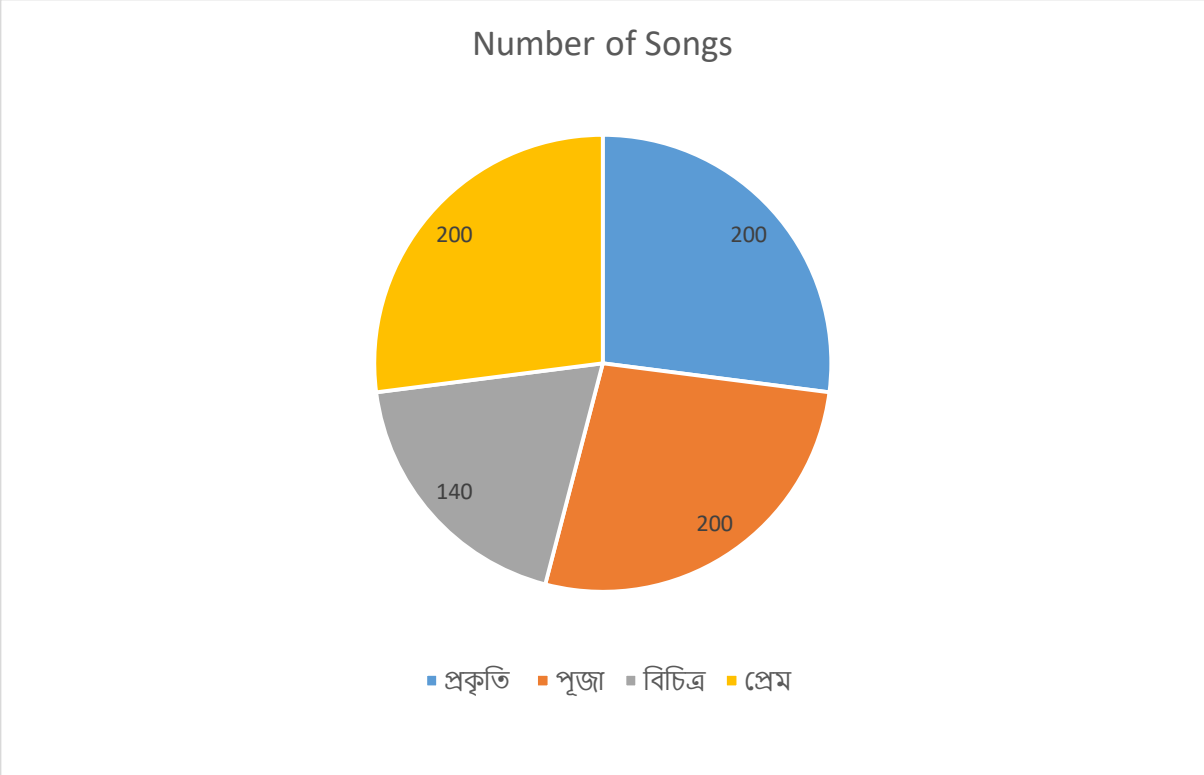
# DATA PROCESSING:

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

# DATA COLLECTION:

Song lyrics are usually shorter in length than normal sentences, and they use a relatively limited vocabulary. Therefore the most important characteristic is the selection of words in a song. Therefore, the most important characteristic is the words in the chosen songs, and not only should they contain the linguistic features which are associated with the language, but the aesthetic feature should also be properly represented. For this study, we have acquired the benchmarked dataset from the official website of Society of Natural Language Training and Research (SNLTR). SNLTR is a reputed Institute whose main focus is the archival of songs, poems, stories and novels of famous Bengali literary personalities like Rabindranath Tagore, Bankim Chandra Chattopadhyay, Sarat Chandra Chattopadhyay and many more, in the soft Unicode compatible format.

From the web page https://rabindra-rachanabali.nltr.org/node/6585, we have taken 4 parjays for our multi-class classification purpose. The number of songs that have been included in each parjaay has been mentioned in the chart below:




The annotated corpus has been manually exported to an excel spreadsheet and converted to UTF-8 readable format from the default Latin1 format. Since Bengali text is not readable in Latin1 format, google spreadsheet has been used for this purpose and the final dataset has been exported as a CSV file. 70% of the dataset has been used for training, while 30% has been used for testing.

| Parjaay | Training | Testing |
|---|---|---|
| Prakriti (প্রকৃতি) | 140 | 60 |
| Puja (পূজা) | 140 | 60 |
| Bichitro (বিচিত্র) | 98 | 42 |
| Prem (প্রেম) | 140 | 60 |

## DATA PRE-PROCESSING:

The raw data set (annotated corpus) had to be processed into a form that can be used to train the models. We had manually classified the dataset by giving associating a song with the corresponding enumeration of a parjaay.

While cleaning the data, we had to make a trade-off between the accuracy of the models which were to be used and the aesthetic and literary property of the data. Meaning,

- If we do not eliminate the punctuations, the stop words, stem and then lemmatize the data, then the aesthetic intricacies of the lyrics stay as it is, but the overall accuracy of the model hence created is considerably lower.
- If we eliminate the punctuations, stop words and lemmatize the data, then the accuracy of the model hence created increases but all the intricacies of the lyrics data is lost.

Here, we focus more on the accuracy part, and hence we extensively pre-process the given data by removing stop words, punctuations and all other special characters in the Bengali text. Extra spaces are also removed from the text. The code for preprocessing is given below:

```
import re

import collections


def clean_text(t1):

    whitespace = re.compile(u"[\s\u0020\u00a0\u1680\u180e\u202f\u205f\u3000\u2000-\u200a]+", re.UNICODE)

    bangla_fullstop = u"\u0964"

    bangla_double=u"\u0965"

    hyphen="–"

    hyphen2="—"

    halfspace="\u200c"

    halfspace1="\u200d"

    randomspace="\ufeff"

    punctSeq   = u"['\"“”‘’]+|[.?!,…]+|[:;]+"
```

```
punc = u"[(),$%^&*+={}\[\]:\"|\'\~`<>/,¦!?½£¶¼© ⅐⅑⅒⅓⅔⅕⅖⅗⅘⅙⅚⅛⅜⅝⅞⅟0⅓¤¿º;-]+"


t1= whitespace.sub(" ",t1).strip()

t1 = re.sub(punctSeq, " ", t1)

t1 = re.sub(bangla_fullstop, " ",t1)

t1 = re.sub(bangla_double, " ",t1)

t1 = re.sub(punc, " ", t1)

t1 = re.sub(hyphen, " ",t1)

t1 = re.sub(hyphen2, " ",t1)

t1 = re.sub(halfspace, " ",t1)

t1 = re.sub(halfspace1, " ",t1)

t1 = re.sub(randomspace, " ",t1)

return t1

df['lyrics']=df['lyrics'].apply(clean_text)
```

This cleaned text is then used as input data for training the model. The enumerations which have been used for the different parjaay's are:

| **Parjaay** | **Enumeration** |
|---|---|
| Prakriti (প্রকৃতি) | 1 |
| Puja (পূজা) | 2 |
| Bichitro (বিচিত্র) | 3 |
| Prem (প্রেম) | 4 |

# 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 analysing 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.

# CLASSIFICATION:

After text cleaning and removing stop words from the data, we only have 54387 words to work with.

```
df['lyrics'].apply(lambda x: len(x.split(' '))).sum()
```

```
54387
```

Now, we perform the training and testing split on the dataset. After splitting, the next steps include feature engineering. We 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. After that we train several classifiers from scikit-learn library.

```
X = df.lyrics
y = df.enum
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state = 30)
```

## 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, we will use the 'pipeline' class in 'scikit-learn' that behaves like a compound classifier. Since we have 4 classes of data, the multi-class classification variant of Naïve Bayes is used.

We fit the train the model using the function 'fit(trainsetX, trainsetY)'. The accuracy report of each of the model has been obtained. The code for implementation of this classifier has been given below:

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfTransformer

nb = Pipeline([('vect', CountVectorizer()),
               ('tfidf', TfidfTransformer()),
               ('clf', MultinomialNB()),
              ])
nb.fit(X_train, y_train)
%time
from sklearn.metrics import classification_report
y_pred = nb.predict(X_test)

print("NAIVE BAYES")
print('accuracy %s' % accuracy_score(y_pred, y_test))
print(classification_report(y_test, y_pred,target_names=my_tags))
```

The output which is obtained is:

```
Wall time: 0 ns
NAIVE BAYES
accuracy 0.5180180180180181
             precision    recall  f1-score   support

        1.0       0.47      0.79      0.59        48
        2.0       0.58      0.68      0.62        66
        3.0       1.00      0.07      0.12        45
        4.0       0.48      0.46      0.47        63

avg / total       0.61      0.52      0.47       222
```

We can clearly see, that the accuracy which has been obtained is 51.8%. 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:

```
from sklearn.linear_model import SGDClassifier

sgd = Pipeline([('vect', CountVectorizer()),
                ('tfidf', TfidfTransformer()),
                ('clf', SGDClassifier(loss='hinge', penalty='l2',alpha=1e-2, random_state=10, max_iter=30, tol=None)),
               ])
sgd.fit(X_train, y_train)

%time
print("LINEAR SUPPORT VECTOR MACHINE")
y_pred = sgd.predict(X_test)

print('accuracy %s' % accuracy_score(y_pred, y_test))
print(classification_report(y_test, y_pred,target_names=my_tags))
```

The output which is obtained is:

```
Wall time: 0 ns
LINEAR SUPPORT VECTOR MACHINE
accuracy 0.5720720720720721
             precision    recall  f1-score   support

        1.0       0.55      0.83      0.66        48
        2.0       0.59      0.71      0.64        66
        3.0       0.67      0.22      0.33        45
        4.0       0.56      0.48      0.51        63

avg / total       0.59      0.57      0.55       222
```

Thus from the output we can see that the Linear SVM model has an accuracy of 57.2% which is considerably greater than the previous model.

## LOGISTIC REGRESSION:

Logistic Regression is an easy and easy to understand classification algorithm which uses a sigmoid function, or also called a logistic function to classify data. This classifier usually works the best with binary classification. It is the special case of linear regression where the target variable, in this case parjaay enumeration, is categorical in nature. For creating this model, 'LogisticRegression' is imported from the 'sklearn.linear_model' library. The code for Logistic Regression is shown below:

```
from sklearn.linear_model import LogisticRegression
from sklearn import metrics

logreg = Pipeline([('vect', CountVectorizer()),
                ('tfidf', TfidfTransformer()),
                ('clf', LogisticRegression(n_jobs=1, C=1e1)),
               ])
logreg.fit(X_train, y_train)

%time

y_pred = logreg.predict(X_test)
print("LOGISTIC REGRESSION")
print('accuracy %s' % accuracy_score(y_pred, y_test))
print(classification_report(y_test, y_pred,target_names=my_tags))
```

The output obtained is:

```
Wall time: 0 ns
LOGISTIC REGRESSION
accuracy 0.5405405405405406
             precision    recall  f1-score   support

        1.0       0.53      0.69      0.60        48
        2.0       0.56      0.68      0.62        66
        3.0       0.62      0.33      0.43        45
        4.0       0.48      0.43      0.45        63

avg / total       0.55      0.54      0.53       222
```

Thus we can see that the accuracy obtained here is 54%, which is less than the accuracy obtained by Linear SVM. But still, this result is much better than the Naïve Bayes Model.

The confusion matrix obtained from this model gives an idea about the false positive and true negatives obtained with the testing data.

```
array([[33,  6,  1,  8],
       [ 5, 45,  7,  9],
       [ 9,  9, 15, 12],
       [15, 20,  1, 27]], dtype=int64)
```

The diagonal is the case where the actual and predicted values are the same. The upper triangle is the number of cases of False Positives, and the lower triangle is the number of cases of True Negatives.

# CONVOLUTIONAL NEURAL NETWORK

CNN is a class of deep, feed-forward artificial neural networks (where connections between nodes do not form a cycle) and use a variation of multi-layer perceptrons designed to require minimal preprocessing. These are inspired by animal visual cortex.

In our case, we run train the model for 3 epochs because it gives the optimal result.

```
Train on 399 samples, validate on 45 samples
Epoch 1/3
399/399 [==============================] - 9s 23ms/step - loss: 1.3325 - acc: 0.4536 - val_loss: 1.0883 - val_acc: 0.8222
Epoch 2/3
399/399 [==============================] - 7s 17ms/step - loss: 0.7748 - acc: 0.8446 - val_loss: 0.9892 - val_acc: 0.7778
Epoch 3/3
399/399 [==============================] - 7s 17ms/step - loss: 0.3251 - acc: 0.9900 - val_loss: 1.2326 - val_acc: 0.5111
```

The output of the accuracy of the model is shown below:

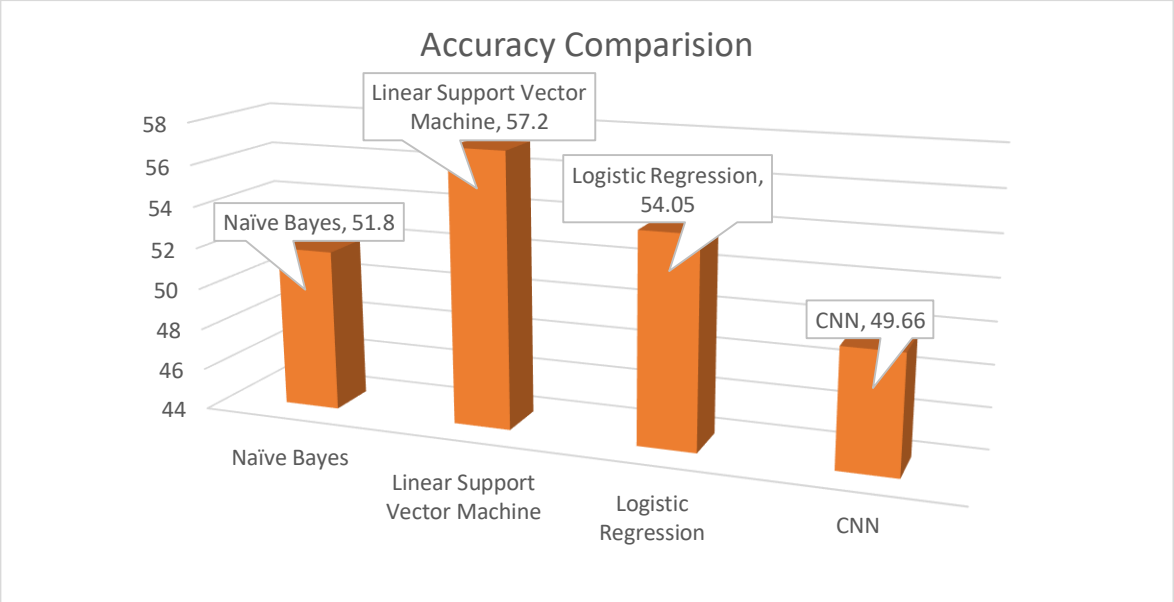
```
[ ]  score = model.evaluate(x_test, y_test,
                            batch_size=batch_size, verbose=1)
     print('Test accuracy:', score[1])

     296/296 [==============================] - 1s 2ms/step
     Test accuracy: 0.4966216216216216
```

Out of all the 4 models, CNN have the least accuracy and that may be because of 2 reasons:

- Firstly because CNN works best with a huge dataset, since our dataset is very small, the model does not converge to an improved result, and thus accuracy decreases.
- Secondly, the very architecture of CNN makes it the best option when working with image data. CNN does not give comparatively better result when the model is trained on textual data.

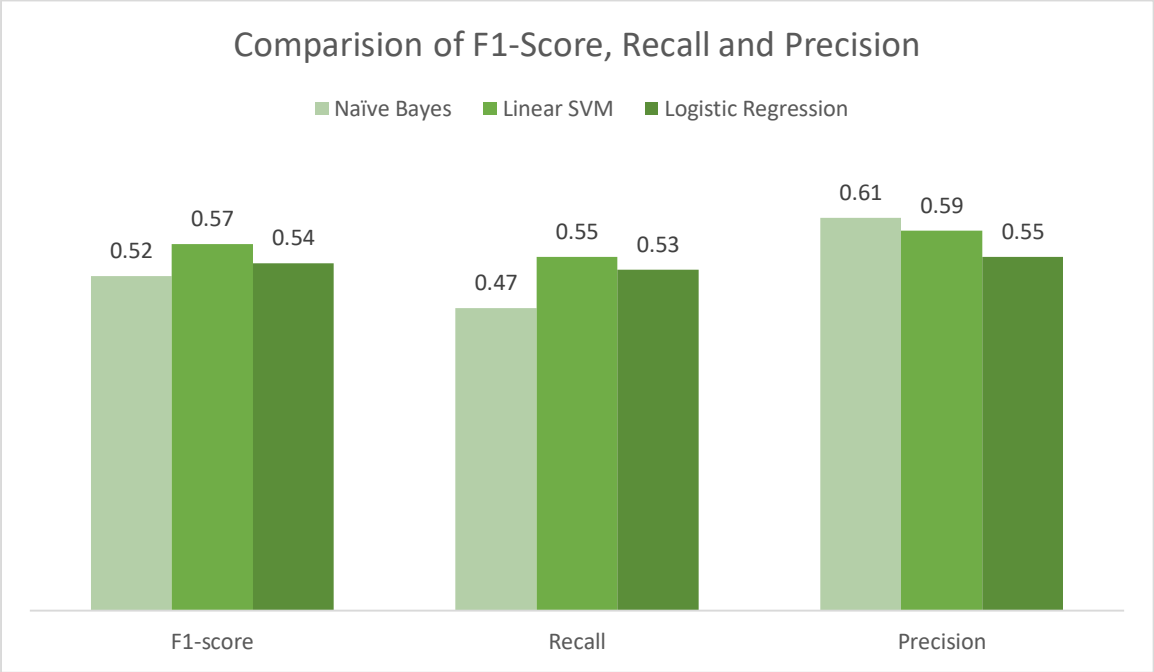
# MODEL COMPARISION

We first compare the accuracy of all the 4 models with the help of a bar graph:




Clearly, Linear SVM has the best accuracy.

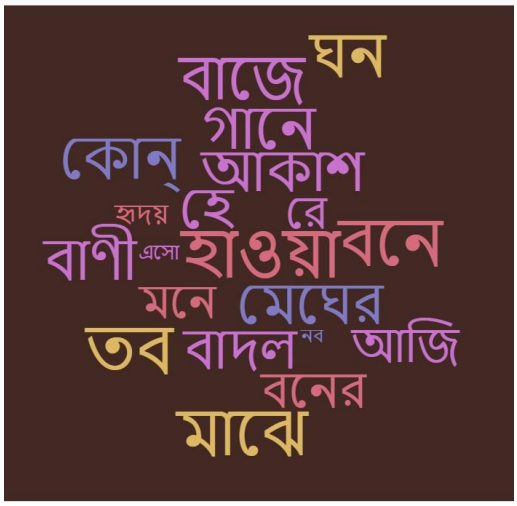
We also compare on the basis of the 3 extra parameters obtained from the first 3 models.



# DATA VISUALIZATION:

We try to draw an insight from the different kinds of visualization which are possible with the dataset. At first, we have found out the 20 most common words which of each parjaay in the dataset. Taking these words, we have created the corresponding word clouds as well.

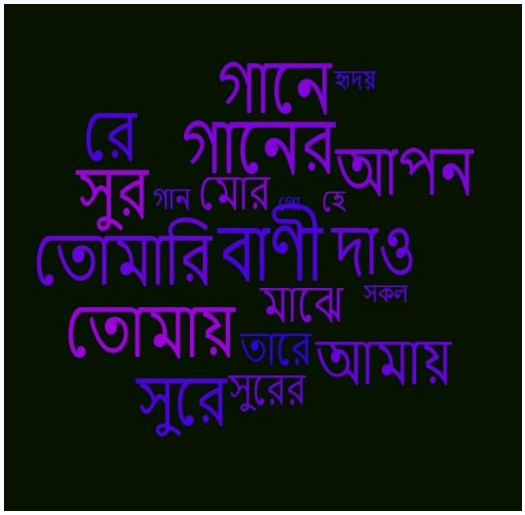
TOP 20 COMMON WORDS OF প্রকৃতি




TOP 20 COMMON WORDS OF প্রেম

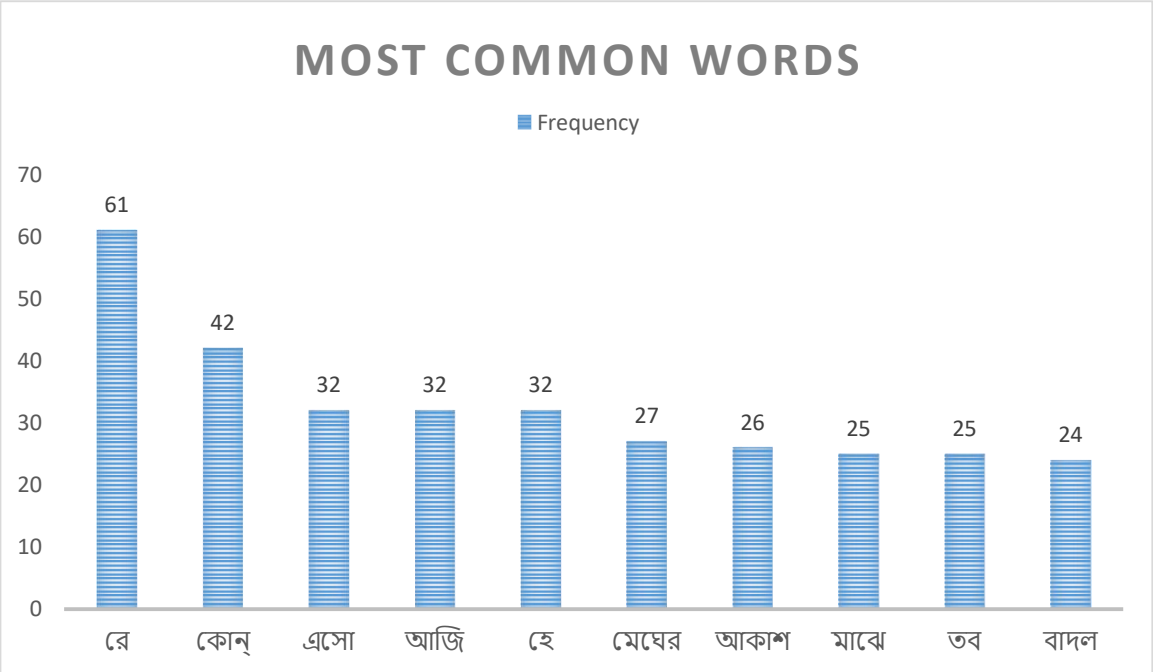

TOP 20 COMMON WORDS OF বিচিত্র


TOP 20 COMMON WORDS OF পূজা



Previously, we had shown the word clouds of specific genres. Now, we can see that some words like "গান" are common in almost all parjaay's, where as some words like "মেঘের" and "পথের" are only common in specific parjaay's and not common to all.

In the following bar graph, we see the most common words of the whole dataset and also the frequency of the word in the whole corpus.




## WORD COUNT:

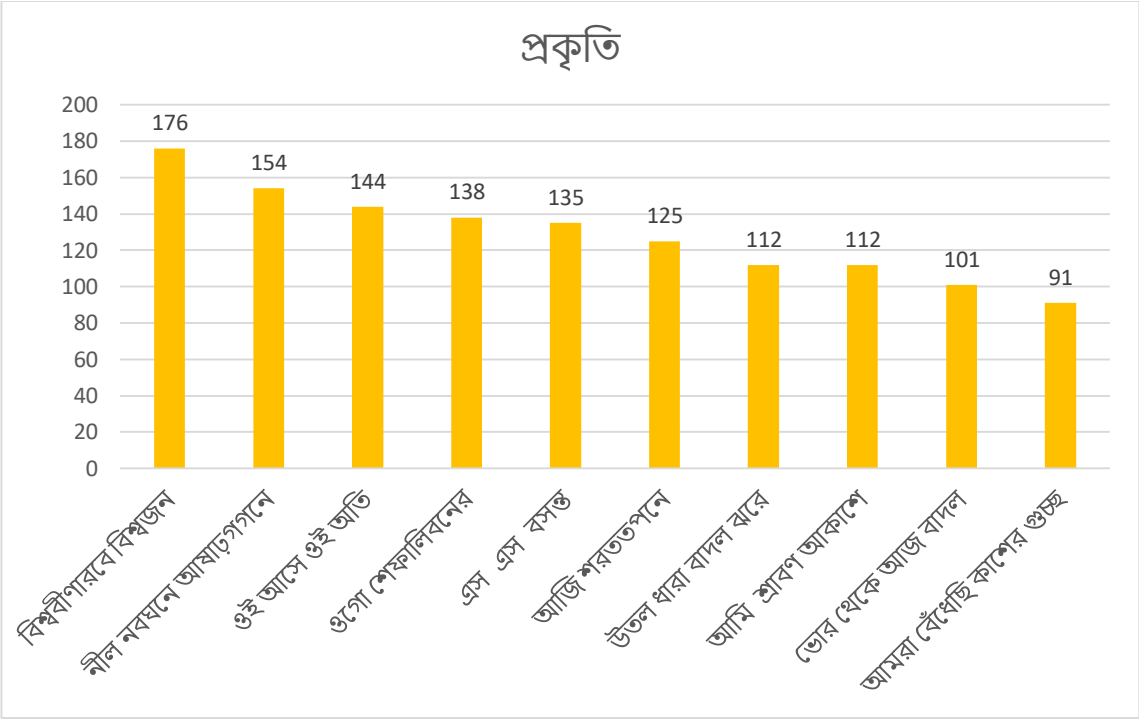
The word count of a song lyric is a very important feature since it gives an idea about the length of the song. Now since we are removing stop words from the song, therefore calculating the word count after that is not accurate. Therefore, the stop words have not been removed from the lyrics while calculating the word count of each song.

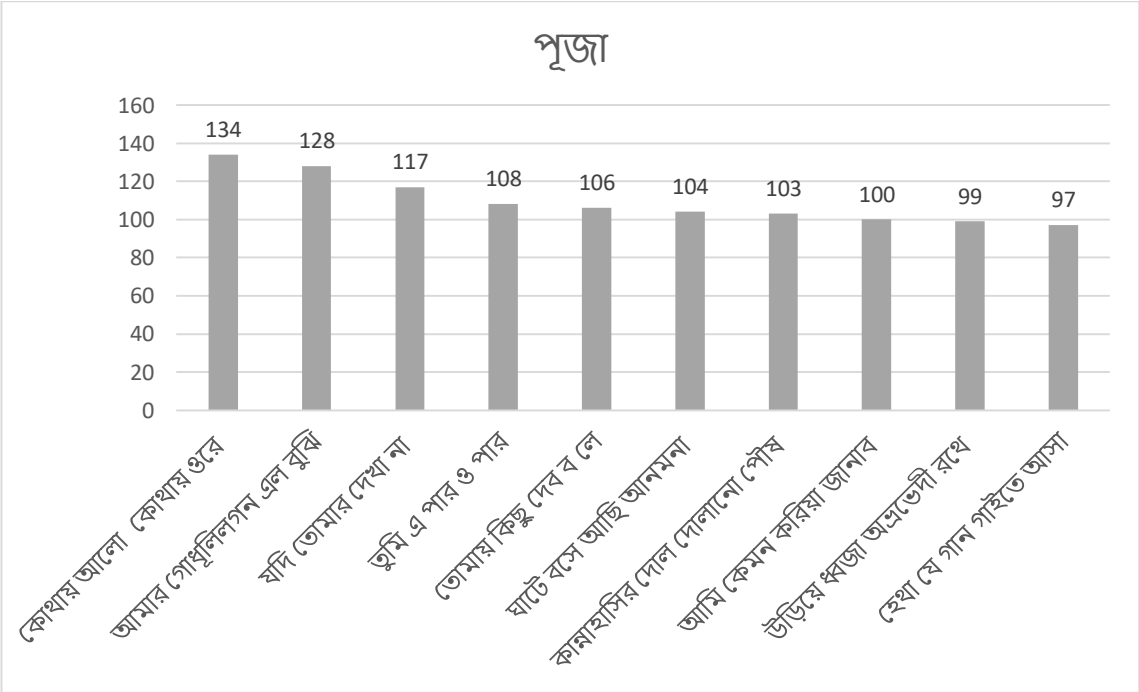
On calculating the word count and sorting them, we have found out:

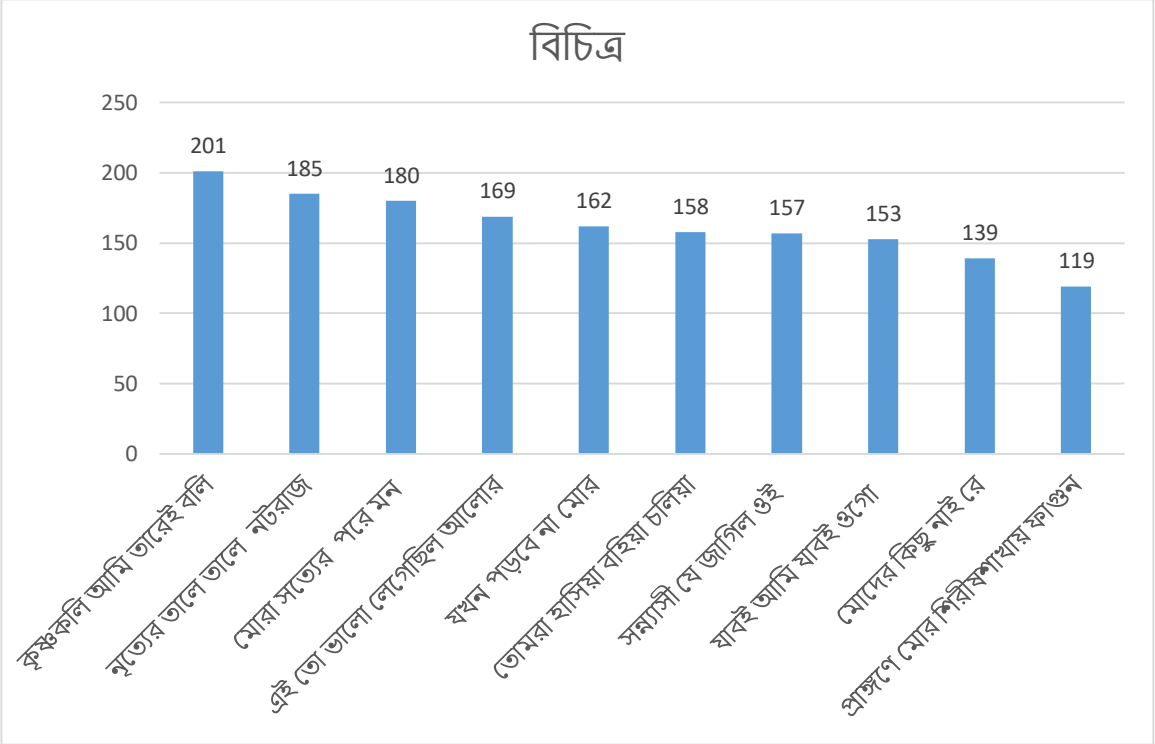
- The top 10 songs of each genre in terms of maximum word count along with the specific word count.
- The word count comparison (Most words as well as least words) between the 4 Parjaay's.

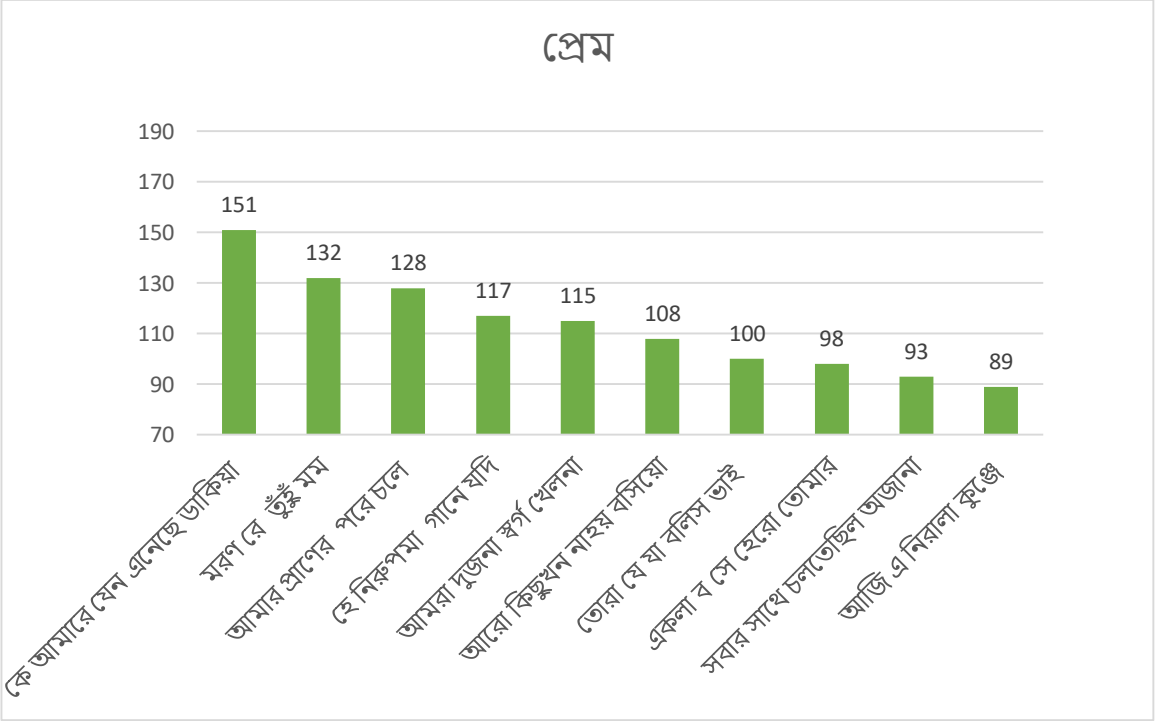
- The word count comparison of the top 10 songs with maximum word count and minimum word count.

Given below are the visualizations:



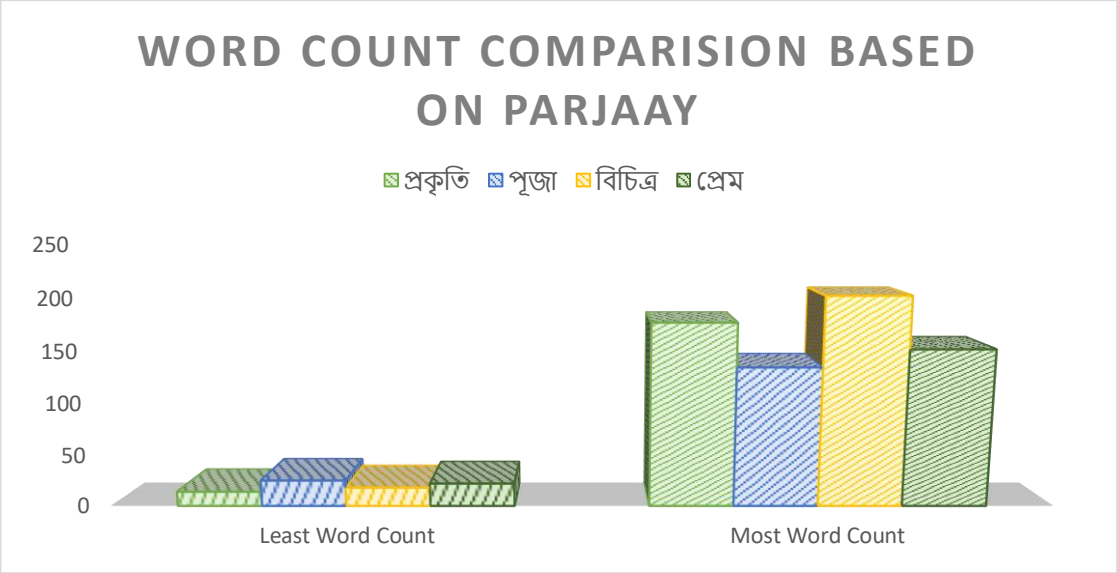


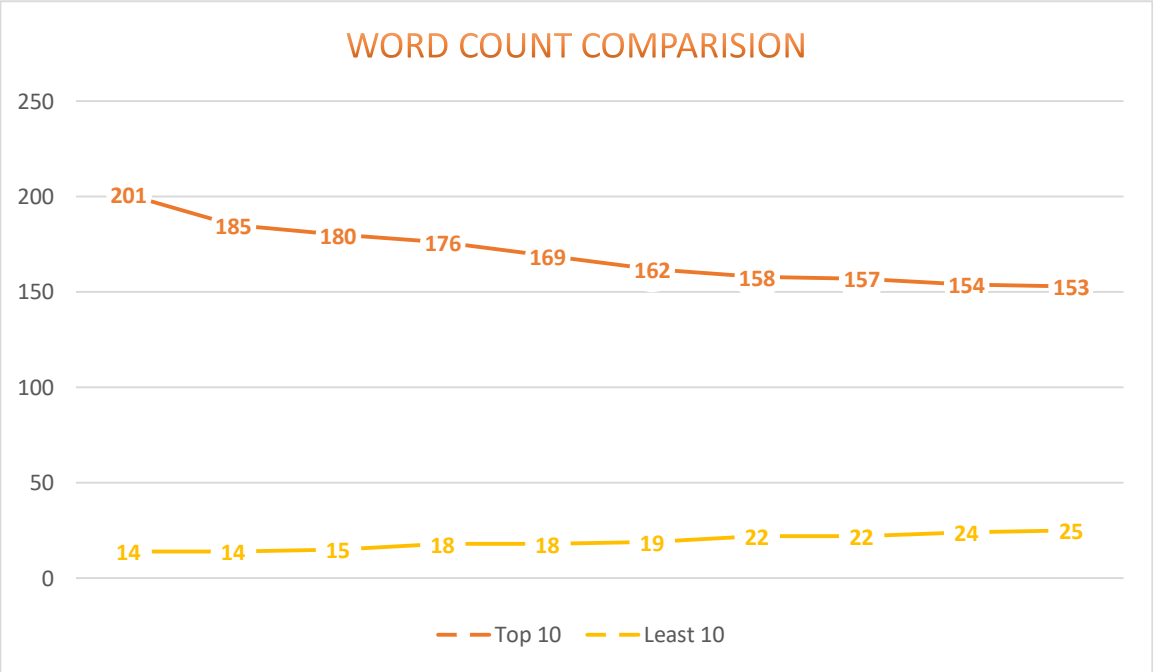

X-Axis: The names of the songs which have the maximum word counts.

Y-Axis: The total word count of the particular song.

Comparing the results hence obtained, we get the following graph:

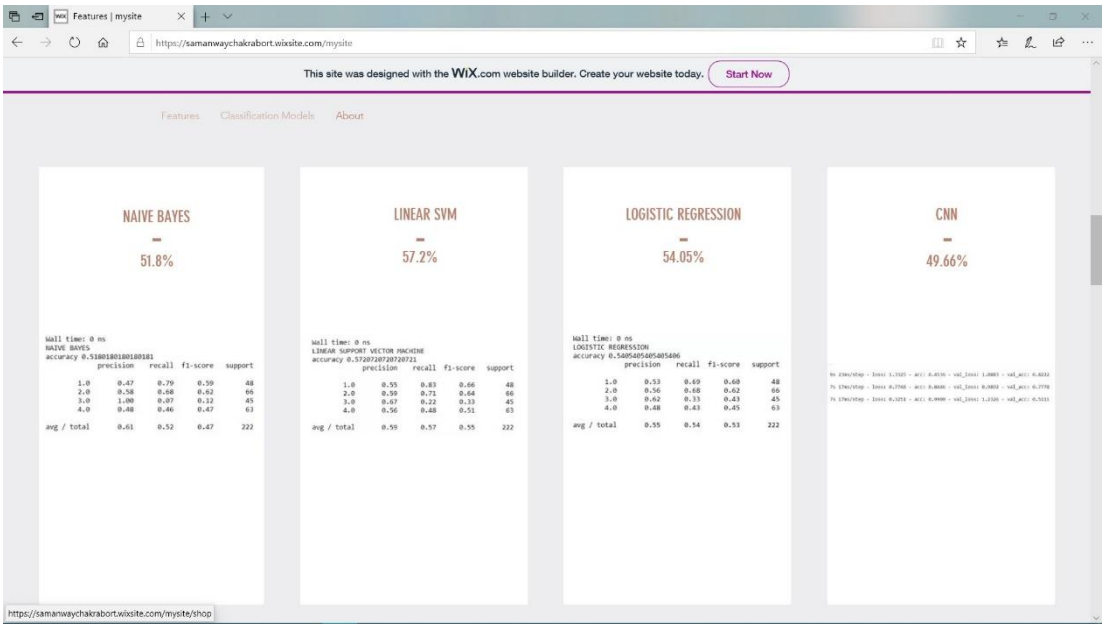



We can clearly see that the song having the most word count is of the parjaay "বিচিত্র" and the song having the least word count is of "প্রকৃতি". If we see the previous 4 graphs, we see a pattern emerging which gives us the insight that songs of the parjaay "বিচিত্র" are usually longer, while the songs of "প্রেম" are shorter in terms of word count. If we consider the final dataset, then the top 10 and bottom 10 in terms of word count can be represented in the form of the following line chart:

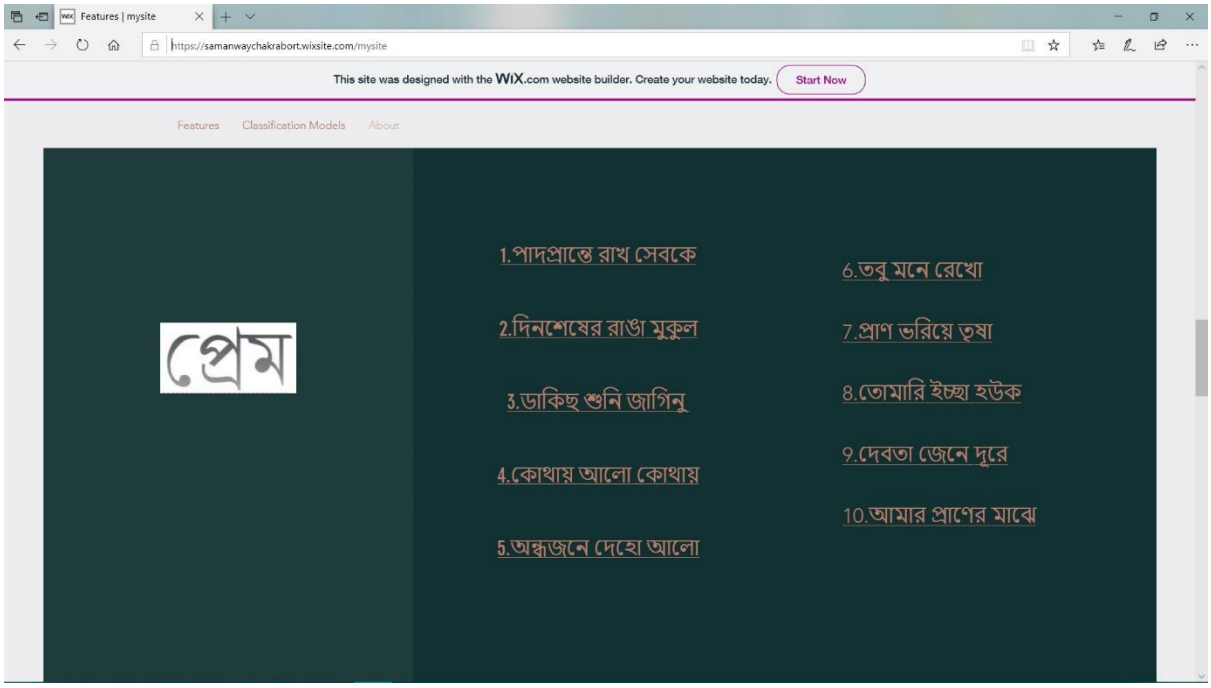


# WEBPAGE

We have created a simple webpage using WIX which contains a homepage, a page which contains all the data visualizations obtained and a page containing the details of the results obtained from the classification models.




We have also made a web page which shows the result of the top songs which are associated with the word "প্রেম". The frequency of the word in each song lyric is calculated for this and the top 10 songs are shown.



# SCOPE OF FUTURE WORK

Although the project has been able to provide some statistical and analytical insight of Rabindra-Sangeet songs with the help of lyrics, some work can still be done.

- More classification models can be used and compared to achieve a better accuracy. The parameters associated can be tweaked as well in order to achieve better results.
- Another attribute "YEAR" could be added which would denote the year the song was written, and more visualizations could be done with the help of it. With the passing years, what kind of songs did Rabindranath Tagore preferred writing could be related as well which would throw a light on the psychological and emotional phases of his life as well.
- We had started working with the audio features using jAudio toolkit as well, taking the input data as audio songs. But unfortunately due to lack of data, proper models could not be created. Hence a multi-modal study comparing the results of the accuracy of the models created from textual data and audio data could be done.
- The webpage is in a very preliminary stage and a few features can be added.
  Firstly, the user will have the option to dynamically write a word and check the top Rabindra-Sangeet songs associated with that word. Secondly, a user interface could be provided so that the user may give the Bengali lyrics of a song, and the user could select the classifier using which the parjaay prediction is to be done. And, the output of the chosen model would print the predicted parjaay of the song on the webpage.

# REFERENCES:

1. Multi-Class Text classification Model Comparision and Selection(Blog Post): Susan Li -https://towardsdatascience.com/multi-class-text-classification-model-comparison-and-selection-5eb066197568

2. What is TF-IDF? (Blog Post) :Bruno Stecanella - https://monkeylearn.com/blog/what-is-tf-idf/

3. Multimodal mood classification (Research paper) – A case study of differences in Hindi and Western songs: Braja Gopal Patra, Dipankar Das and Sivaji Bandhopadhyay.

4. Linear Kernel: Why is it recommended for text classification? (Blog Post): Alexandre Kowalczyk

https://www.svm-tutorial.com/2014/10/svm-linear-kernel-good-text-classification/

5. Understanding Logistic Regression in Python (Blog Post): Avinash Navlani

https://www.datacamp.com/community/tutorials/understanding-logistic-regression-python

6. UNLP at the Media Eval 2015 C@merata Task (Research Paper): Kartik Asooja, Sindhu Kiranmai Ernala and Paul Buitelaar.

7. Mood Classification of Hindi songs based on lyrics (Research Paper): Braja Gopal Patra, Dipankar Das and Sivaji Bandhopadhyay.