SUMMER INTERNSHIP REPORT

Area of Online Internship	AI/Machine Learning/Deep Learning
Intern Name	Reckon Mazumdar
Name of Institution	INDIAN INSTITUTE OF TECHNOLOGY, INDORE
Faculty Mentor Name	Dr. Vimal Bhatia
Duration	2 MONTHS (19/06/2021 to 19/08/2021)
Date of Submission	17/08/2021

Table of Contents

ABSTRACT

In the era of technology, each one of us is expressing our opinion on social media platforms very frequently and these opinions are mostly expressed in regional languages, so the contents mostly generated are in regional languages in nature. Sentimental analysis is the process of extracting the opinions of people and using it to understand people's attitudes, reactions expressed on the web regarding the various issues in the world and is also known as opinion mining. Nowadays with the increasing use of the internet, a lot of information is available on the web which is about different products, movies, songs, books, technologies, etc.

A lot of documents are available which express opinions on different issues. But the main challenge arises in analyzing these documents to produce useful knowledge. Tremendous works in the area of Sentiment Analysis are available for the English language. However, there has been little work done for Indian languages. From the last few years, opinion-rich resources are booming in Assamese and hence there is a need to perform Sentiment Analysis in Assamese. In this report, we have categorized song reviews in Assamese as positive or negative. Two methods- Count Vectorizer and TF-IDF have been used for feature matrix generation. Then we have applied different classifying algorithms to classify the sentiment of the song review as positive or negative. Other approaches have also been implemented and the results obtained have been compared.

INTRODUCTION

Sentiment Analysis is a natural language processing task that helps to identify and categorize opinions expressed in a piece of text as positive, negative, or neutral. It helps to determine the reviewer's point of view on a particular topic. Most of the research in this domain has been focused on the English language. However, increasing user-generated content in Assamese on the internet has motivated us to perform sentiment analysis research on song reviews in Assamese.

Sentiment Analysis in Assamese is very challenging due to the following reasons:

- Assamese is morphologically rich and a free order language as compared to the English language.
- Assamese is a resource-scarce language that causes problems in the collection and generation of datasets.
- Very little research work has been done related to Sentiment analysis in Assamese.
- Unavailability of well-annotated standard corpora.
- Limited resources are available for it like Assamese stop words [1].

RELATED WORK

1. Very little research work has been done related to Sentiment analysis in Assamese. One such work has been done by Ringki Das. She proposed a sentiment polarity classification model by applying machine learning classifiers on low resource Assamese language using lexical features on the news domain. The baseline system works only with a bag of words without any feature-based polarity [2].

2. Another contribution to Assamese text summarization was done by Chandan Kalita et al. They have presented an extractive approach of Text summarization of Assamese, a free word order inflectional Indic language, using WordNet. From their experiment, they got approximately 78% accurate results [3].

PROPOSED WORK

In the section, we have proposed the details of the dataset and models used for Assamese song reviews pre-processing, algorithms used for feature matrix generation, and various classifiers used.

- ## Dataset Used

We have used a total of 1028 Assamese songs reviews for the Sentiment Analysis. We have manually collected all of the reviews from YouTube comments and labeled them as 1 (for positive comment) and 0 (for negative comment). Out of 1028 reviews collected manually, 505 reviews are positive and the rest 523 are negative reviews. For Machine Translation based approach, we also need English reviews, by using Microsoft Azure Cognitive Services (Translator) the native language reviews are translated to relative English reviews.

Loading the data

```
In [5]: df=pd.read_csv('reviews.csv',encoding='utf-8')
        df.sample(10)
```

Out[5]:

	Sentence	Sentiment
495	ইমান অপমান পায়ো লাজ পোৱা নাই	0
251	বহুত ধুনীয়া হৈছে মোজা মোজা	1
922	কিমানবাৰ শুনিলো যদিও হেঁপাহ পেলোৱা নাই	1
188	খুব বেয়া	0
285	মা কছুম মজা ।	1
434	নাজানিছিলো যে অসমী গানবোৰ এইটো ভাল আছিল	1
843	এইবোৰে আমাৰ উঠি অহা প্ৰজন্মক বিপথে পৰিচালিত কৰিছে	0
274	মই তোমাৰ গান শুনাতকৈ কাউৰীৰ মাত টো পছন্দ কৰিম	0
66	মোককেল কেলা ঠাকুৰীয়া	0
1015	এই গীততোৰ সৈতে বহুত স্মৃতি জড়িত হৈ আছে	1

SNIPPET 1: In-language dataset

Loading the data

```
In [4]: df=pd.read_csv('translated.csv',encoding='utf-8')
        df.sample(10)
```

Out[4]:

	Sentence	Sentiment	Translated
101	সুকণ্ঠৰে সভ্য গীত গাব লাগে	0	Sing civilised songs with a voice
458	আও ধুনীয়া ভিডিও	1	Aao Beautiful Videos
837	যদি জান গাব নাজানো তেতিয়া হলে অযথা গান বোৰ গা...	0	If you don't know how to sing, don't sing unne...
206	মই এই গানটো প্ৰতিদিনে বজাওঁ	1	I play this song every day
231	নিজৰ প্ৰতিভাৰ ওপৰত বিশ্বাস ৰাখি আগুৱাই যোৱা ।	1	To keep faith in your talent and move forward.
205	গৰিলা	0	guerrilla
463	চুলি কাটি লো আগত	0	Cut your hair in advance
694	কোনে কোনে দিশ্লিকে কৰিব আহিছে	0	Who is coming to do it in a way?
220	ইমান ধুনীয়া বিহু গানবোৰ শুনিয়েই সৰুৰ পৰা ডাঙ...	1	He grew up from a young age after listening to...
13	কি যে সুন্দৰ গীতটো	1	What a beautiful song

```
In [5]: df.shape
Out[5]: (1028, 3)
```

SNIPPET 2: Machine Translation dataset

- **Approach**

We have used two approaches to classify the sentiment of Assamese reviews as positive or negative.

In-language Classification

This approach is based on training the classifiers in the same language as the text. It relies heavily on the availability of resources in the same language to analyze the sentiment. Thus all training text, testing text are in the Assamese language. The feature representation(Term frequency or TF-IDF) can be varied to see the effect on In-language classification on Assamese song reviews. In this approach, we use a variety of classifiers to train and test the data. We know TF-IDF can be a better way of feature matrix generation as it reduces the effect of very frequent words in documents but does not contribute much to the relevance of the text.

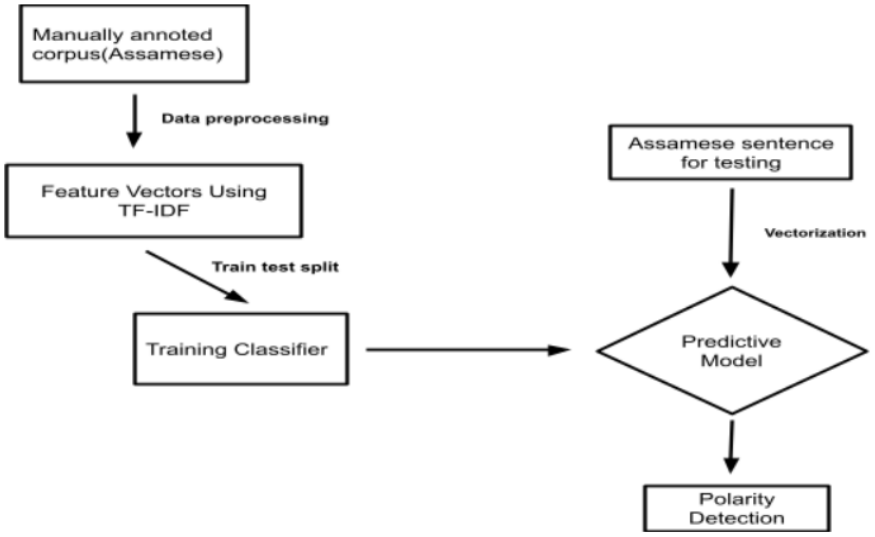


Fig.1: IN-Language classification flowchart

Machine Translation based sentiment analysis

Resources are scarce in Assamese, like the number of stop words and stemming tools for the Assamese language. Thus it enforced us to take into consideration the machine translation-based sentiment analysis approach. In this approach, we have first translated the entire Assamese language dataset into the English language using Microsoft translator API, then we applied the text preprocessing techniques like removing stop words, punctuations, characters that are not alphanumeric, and stemming which were not properly possible in the previous approach. We have used the count vectorizer model for feature matrix generation. Finally, we used a variety of classifiers to train and test the data.

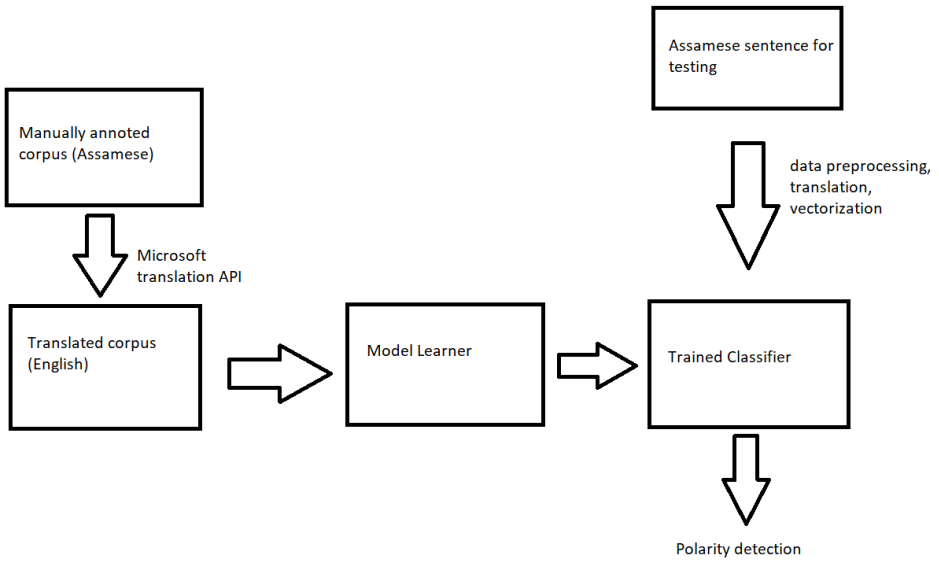


Fig.2: Machine translation based classification flowchart

- **Data Pre-processing**

In In-language technique

In this step, we have first removed duplicate sentences, and then we have tokenized the data. After that, we have removed all those words from each review that do not contribute to the accuracy of classification such as punctuations,special characters, and stop words.
Algorithm:

1. Removing the duplicate sentences
2. Tokenization of the text using NLTK library
3. Removing the stopwords
4. Removing the punctuations and special characters
5. Returning the cleaned text

Removing duplicate values

```
In [9]: #check for duplicate values
        df.duplicated().sum()

Out[9]: 11

In [10]: #removing duplicates
         df.drop_duplicates(subset ="Sentence",
                            keep ='first', inplace = True)
         df.shape

Out[10]: (1016, 2)
```

SNIPPET 3: Removing duplicates in In-language dataset

Data cleaning

In [20]:
```
def transform_text(text):
    #tokenizing
    text=nltk.word_tokenize(text)
    stop=['অতএব', 'অথচ', 'অথবা', 'অধঃ', 'অন্ততঃ', 'অর্থাৎ', 'অর্থে', 'আও', 'আঃ', 'আচ্ছা', 'আপাততঃ', 'আয়ৈ', 'আৰু',
      'আস্', 'আহা', 'আহাহা', 'ইতস্ততঃ', 'ইতি', 'ইত্যাদি', 'ইস্', 'ইহ', 'উঃ', 'উৱা', 'উস্', 'এতেকে', 'এথোন',
      'ঐ', 'ওঁ', 'ওৰফে', 'ঔচ্', 'কি', 'কিম্বা', 'কিন্তু', 'কিয়নো', 'কেলেই', 'চোন', 'ছাৰি', 'ছিকৌ', 'ছেই',
      'ঠাহ্', 'টেট্', 'তত', 'ততক', 'ততেক', 'তেতেক', 'ততেক', 'তত্রাচ', 'তথা', 'তথৈবচ', 'তাতে', 'তেণ্ড',
      'তো', 'তৌৱা', 'দেই', 'দেহি', 'দ্বাৰা', 'ধৰি', 'ধিক্', 'নতুবা', 'নি', 'নো', 'নৌ', 'পৰা', 'পর্যন্ত',
      'বৰঞ্চ', 'বহিঃ', 'বাবে', 'বাৰু', 'বাহ্', 'বাহিৰে', 'বিনে', 'বে', 'মতে', 'যথা', 'যদি', 'যদ্যপি', 'যে',
      'যেনিবা', 'যেনে', 'যোগে', 'লৈ', 'সত্ত্বে', 'সমন্ধি', 'সম্প্রতি', 'সহ', 'সু', 'সেইদেখি', 'সৈতে', 'স্বতঃ', 'হঞে', 'হতুৱা', 'হস্তে',
      'হবলা', 'হয়', 'হা', 'হুঁ', 'হুই', 'হে', 'হেই', 'হেঃ', 'হেতুকে', 'হেনে', 'হেনো', 'হেৰ', 'হেৰি', 'হৈ', 'হোঁ', 'ইঃ', 'ইচ্',
      'চুহ্', 'চুঃ', 'আঁ']
    punc="!\"#$%&'()*+,-./:;<=>?@[\]^_`{|}~।"
    puncword=[]
    #removing stopwords and punctuations
    for i in punc:
         puncword.append(i)
    y=[]
    for i in text:
        if i not in stop and i not in punc:
            y.append(i)
    text=y[:]
    y.clear()
    for i in text:
        i=''.join(j for j in i if not j in puncword)
        y.append(i)
    return " ".join(y)
```

In [21]:
```
transform_text('এই গানটোৰে আপোনাক অকলশৰীয়া হৈ থাকিব বিচাৰে আৰ...')
```

Out[21]: 'এই গানটোৰে আপোনাক অকলশৰীয়া থাকিব বিচাৰে আৰ '

In [22]:
```
df['cleaned_sentence']=df['Sentence'].apply(transform_text)
```

In [23]:
```
df.sample(10)
```

Out[23]:

	Sentence	Sentiment	num_char	num_words	cleaned_sentence
623	আপুনি বুঢ়া অনুগ্ৰহ কৰি সোনকালে মৰিব	0	36	6	আপুনি বুঢ়া অনুগ্ৰহ কৰি সোনকালে মৰিব
48	বৰ ভাল লাগিল আৰু আছে যদি দিব	1	28	7	বৰ ভাল লাগিল আছে দিব
548	এইটো ৰাস্তাৰ কাষত শৌচ কৰা হয়	0	29	6	এইটো ৰাস্তাৰ কাষত শৌচ কৰা হয়
238	মই ইয়াতে মৰিছিলো	0	17	3	মই ইয়াতে মৰিছিলো
657	আহাহা কি অপূর্ব, কি মোহনীয়	1	26	6	অপূর্ব মোহনীয়
221	মোৰ সকলো সময়তে ভাল লগা গান এইটো	1	31	7	মোৰ সকলো সময়তে ভাল লগা গান এইটো
535	তোমাৰ গান শুনিলে সকলো পাহৰি কোনোবা আলোক দিগন্ত...	1	62	12	তোমাৰ গান শুনিলে সকলো পাহৰি কোনোবা আলোক দিগন্ত...
41	কি ফটোৱা	0	8	2	ফটোৱা
998	এই গানটোৰে মোৰ আত্মাক নৃত্য কৰি তোলে	1	36	7	এই গানটোৰে মোৰ আত্মাক নৃত্য কৰি তোলে
297	আপুনি এতিয়াও কিয় মৰি যোৱা নাই?	0	32	7	আপুনি এতিয়াও কিয় মৰি যোৱা নাই

In [24]:
```
df['cleaned_sentence'].shape
```

Out[24]: (1016,)

SNIPPET 4: Data preprocessing in In-language dataset

In Machine Translation technique

Similar to the above approach we have first removed duplicate sentences then converted all the characters to lowercase, performed tokenization, removed the special characters, removed all the stop words and punctuations, at last, we performed stemming on the dataset.
Algorithm:

1. Removing the duplicate sentences
2. All the text is converted to lowercase
3. Tokenization of the text using NLTK library
4. Stopwords are removed
5. Removing the punctuations and special characters
6. Finally stemming is done
7. Returning the cleaned text

Removing duplicate values

```
In [8]: #check for duplicate values
        df.duplicated().sum()

Out[8]: 14

In [9]: #removing duplicates
        df.drop_duplicates(subset ="Translated",
                           keep ='first', inplace = True)
        df.shape

Out[9]: (1005, 3)
```

SNIPPET 5: Removing duplicates in Machine Translation dataset

Data cleaning

In [18]:
```
ps = PorterStemmer()
```

In [19]:
```
def transform_t(text):
    #making the text lowercase
    text = text.lower()
    #tokenizing
    text = nltk.word_tokenize(text)

    y = []
    #keeping the alphanumeric characters only
    for i in text:
        if i.isalnum():
            y.append(i)

    text = y[:]
    y.clear()
    #Removing stopwords and punctuations
    for i in text:
        if i not in stopwords.words('english') and i not in string.punctuation:
            y.append(i)

    text = y[:]
    y.clear()
    #stemming
    for i in text:
        y.append(ps.stem(i))


    return " ".join(y)
```

In [20]:
```
transform_t("It is very difficult to hear such sweet songs ...")
```

Out[20]:
```
'difficult hear sweet song'
```

In [21]:
```
df['Transformed']=df['Translated'].apply(transform_t)
df.sample(10)
```

Out[21]:

	Sentence	Sentiment	Translated	num_char	num_words	Transformed
142	.মই অসমবুজি নাপাওঁ কিন্তু মই এই সুৰ আৰু সংগীত ...	1	I don't understand inequality but I love this ...	60	12	understand inequ love tune music
917	কেনেকৈ ইমান অর্থপূর্ণ শব্দ বোৰ বিচাৰি পায়	1	How to find such meaningful words	33	6	find meaning word
394	আপোনাৰ কণ্ঠস্বৰ অসহনীয়	0	Your voice is unbearable	24	4	voic unbear
314	এইবোৰ হে বিহু শুনিলে শুনি থাকিবৰ মন যায়	1	It's all you feel like listening to when you h...	55	13	feel like listen hear bihu
993	অশ্লীল গীতত	0	in obscene song	15	3	obscen song
952	বহুত ভাল লাগিছে গানটো	1	It's great to have the song	27	7	great song
452	বেয়া লাগিল	0	it felt bad	11	3	felt bad
846	ষাৰ গৰু	0	sher cow	8	2	sher cow
543	গোটেই খন একেবাৰে ছেটেৰিং কৰি পেলাইছে	0	The whole thing is absolutely set.	34	7	whole thing absolut set
776	আপুনি এজন মানসিক ৰোগী	0	You are a mental lying patient	30	6	mental lie patient

SNIPPET 6: Data preprocessing in Machine Translation dataset

- **Feature Matrix Generation**

Once the pre-processing of data is done, we compute the feature matrix using the TF-IDF (in in-language technique) and Countvectorizer (in Machine Translation technique) model.

TF-IDF

For data vectorization in the IN-language technique, we have used the TfidfVectorizer() function available in the scikit-learn library. It is one of the most common feature extraction techniques. It is a statistical measure to find how important the word is in the document. Term Frequency (TF) calculates the occurrence of the word in a single document by the total number of words in the document, whereas inverse term frequency (IDF) tries to find how important the word is in all documents [4].
Statistically, TF and IDF are represented in equations 1 and 2 respectively [4].

$$TF = \frac{Number\ of\ time\ word\ appears\ in\ the\ document}{Total\ words\ in\ the\ document} \qquad (1)$$

$$IDF = \frac{log_{e}(Total\ number\ of\ documents)}{Number\ of\ documents\ which\ contains\ the\ word} \qquad (2)$$

Feature extraction using TF-IDF

```
In [31]: from sklearn.feature_extraction.text import TfidfVectorizer

In [32]: tfidf=TfidfVectorizer(max_features=500)

In [33]: X = tfidf.fit_transform(df['cleaned_sentence']).toarray()

In [34]: X.shape
Out[34]: (1016, 500)

In [35]: Y=df['Sentiment'].values

In [36]: Y.shape
Out[36]: (1016,)
```

SNIPPET 7 : Feature matrix generation using TF-IDF

Count Vectorization

Count Vectorization is a vectorization technique in which a document matrix is maintained. The document matrix contains the words present in each document with the frequency of occurrence of that word in the document. Figure 3 explains the count vectorization with an example.

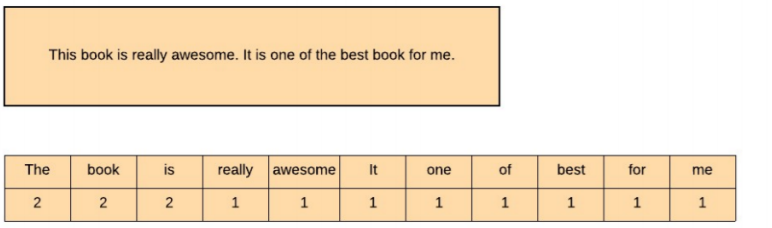


Fig.3: Count Vectorization visualisation [4]

Feature extraction using Count Vectorizer

```
In [26]: from sklearn.feature_extraction.text import CountVectorizer

In [27]: cv=CountVectorizer(max_features=1000)

In [28]: X = cv.fit_transform(df['Transformed']).toarray()

In [29]: X.shape
Out[29]: (1005, 1000)

In [30]: df1 = pd.DataFrame(X, columns=cv.get_feature_names())
         df1.head(10)
```

Out[30]:

	10	17	2020	50	aaji	aao	aap	abai	abl	absolut	...	ya	ye	year	yesterday	yet	yoke	young	youth	zindabad	zubin
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

10 rows × 1000 columns

SNIPPET 8 : Feature matrix generation using Count Vectorization

- **Classification**

Logistic Regression

Logistic regression is a supervised learning classification algorithm and a multi-class logistic model which is used to estimate the probability of response-based predictor variables in which there are one or more independent variables that determine an outcome. The nature of the target or dependent variable is dichotomous, which means there would be only two possible classes.

In simple words, the dependent variable is binary, having data coded as either 1 (stands for success/yes) or 0 (stands for failure/no).

Mathematically, a logistic regression model predicts P(Y=1) as a function of X. It is one of the simplest ML algorithms that can be used for various classification problems such as spam detection, Diabetes prediction, cancer detection, etc**.**

Sigmoid Function

Logistic regression makes use of the sigmoid function which outputs a probability between 0 and 1. The sigmoid function with some weight parameter *θ* and some input x^{(i)}*x*(*i*) is defined as follows:

h(x^(i), *θ) = 1/(1 + e^(-θ^T*x^(i)).*

The sigmoid function gives values between -1 and 1 hence we can classify the predictions depending on a particular cutoff. (say : 0.5)

Note that as (*θ^T)x*(*i*) gets closer and closer to −∞ the denominator of the sigmoid function gets larger and larger and as a result, the sigmoid gets closer to 0. On the other hand, (*θ^T)x*(*i*) gets closer and closer to ∞ the denominator of the sigmoid function gets closer to 1 and as a result the sigmoid also gets closer to 1.

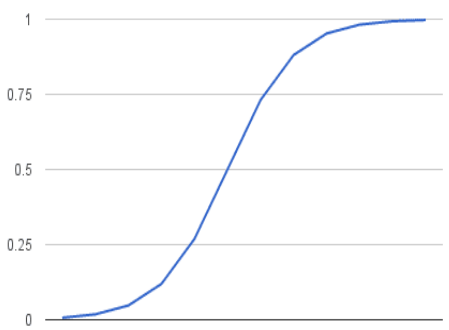


Fig.4: Logistic or sigmoid function

Cost Function and Gradient Descent

The logistic regression cost function is defined as:

$J(\theta)=(-1/m)*\sum i=1 \text{ to } m[y(i)\log(h(x(i),\theta)+(1-y(i))\log(1-h(x(i),\theta))]$

We aim to reduce cost by improving the theta using the following equation:

$\theta j:=\theta j-\alpha*\partial J(\theta)/\theta j$

Here, α is called the learning rate. The above process of making hypothesis (h) using the sigmoid function and changing the weights (θ) using the derivative of cost function and a specific learning rate is called the Gradient Descent Algorithm [5].

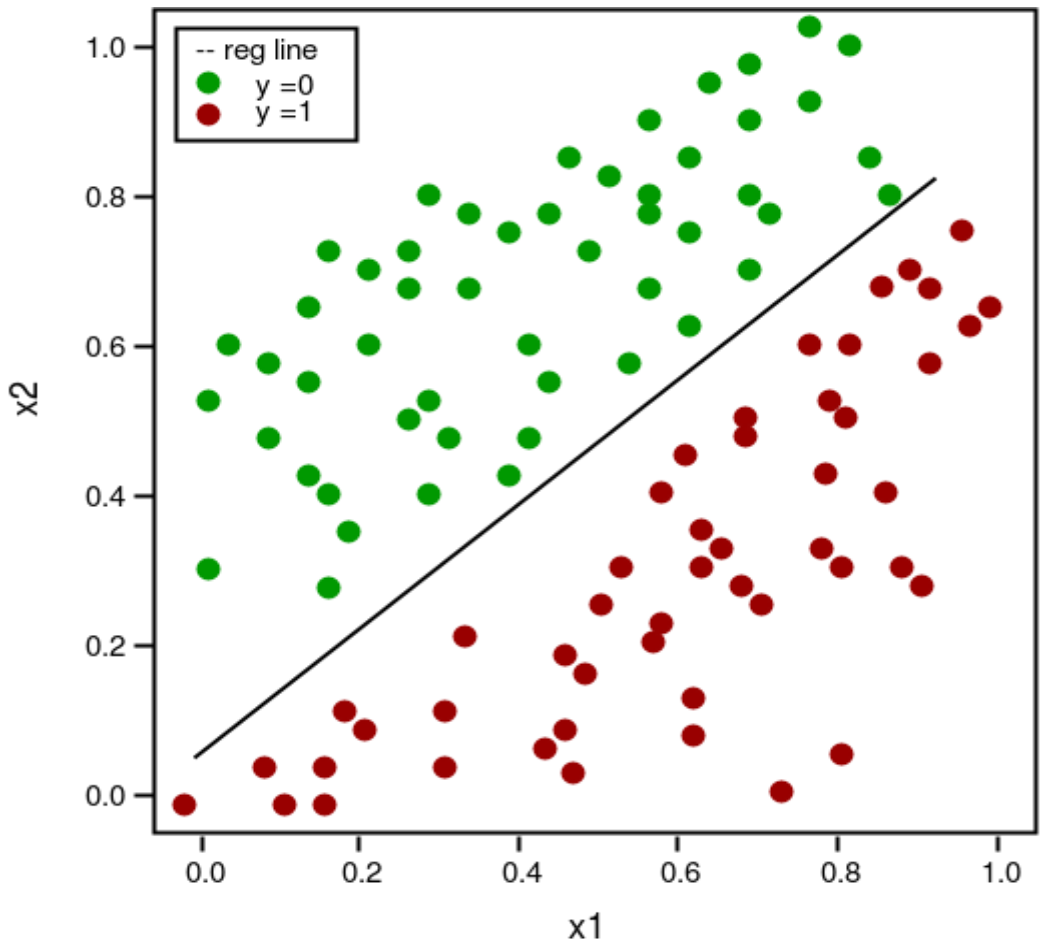


Fig.5: Logistic-regression visualisation [5]

Naïve Bayes

Naïve Bayes is a probabilistic classifier method used when the size of a training set is small. This method is based on the mathematical bayes theorem. There are two class naïve bayes variants for text. Multinomial naïve bayes and benerolli naïve bayes . Multinomial naïve bayes method data follows a multinomial distribution and each feature value is counted . Benerolli naïve bayes data follows a multivariate distribution and each feature is binary [6].

The conditional probability of event X occurs given the evidence Y is determined by Bayes rule by the finding sentiment of review by using a naïve bayes as follows:

P(Sentiment/Sentence) = P(Sentiment)P(Sentence/Sentiment)/P(Sentence)

K-nearest neighbors (KNN)

K-nearest neighbors (KNN) algorithm is a type of supervised ML algorithm that can be used for both classifications as well as regression predictive problems. However, it is mainly used for the classification of predictive problems in the industry. The following two properties would define KNN well:

1. Lazy learning algorithm − KNN is a lazy learning algorithm because it does not have a specialized training phase and uses all the data for training while classification [7].
2. Non-parametric learning algorithm − KNN is also a non-parametric learning algorithm because it doesn’t assume anything about the underlying data [7].

Support Vector Machine (SVM)

Support Vector Machine (SVM) is a linear classifier that works on the concept of hyperplane and decision plane. SVM performs classification by defining the best hyperplane, which differentiates two different classes. Training the model to classify the dataset using the training dataset, which allows defining the best hyperplane. Once we have trained a model and the best hyperplane, we can classify any dataset [8].

Decision Tree

A decision tree is a flowchart-like structure in which each internal node represents a “test” on an attribute and each branch represents the outcome of the test, and each leaf node represents a class label [8].

It uses hierarchical decomposition of the training dataset in which condition on the attribute value is used to split the data. The presence or absence of one or more words indicates predicate or condition. Dataset is divided recursively till leaf nodes which contain a minimum number of records for classification [8].

Random Forest

Random Forest is a powerful and versatile supervised machine learning algorithm that grows and combines multiple decision trees to create a “forest”. It can be used for both classification and regression problems [9].

Decision trees in an ensemble, like the trees within a Random Forest, are usually trained using the “bagging” method. The ensemble method combines predictions from multiple machine learning algorithms together to make more accurate predictions than an individual model [10].

Random Forest grows multiple decision trees which are merged together for a more accurate prediction. The logic behind the Random Forest model is that the individual decision trees perform much better as a group than they do alone. When using Random Forest for classification, each tree gives a classification or a “vote.” The forest chooses the classification with the majority of the “votes.” When using Random Forest for regression, the forest picks the average of the outputs of all trees [9].

Extra Trees

Extra Trees is an ensemble machine learning algorithm that combines the predictions from many decision trees. It is related to the widely used random forest algorithm. It can often achieve as-good or better performance than the random forest algorithm, although it uses a simpler algorithm to construct the decision trees used as members of the ensemble [10].

Each Decision Tree in the Extra Trees Forest is constructed from the original training sample. Then, at each test node, each tree is provided with a random sample of k features from the feature-set from which each decision tree must select the best feature to split the data based on some mathematical criteria (typically the Gini Index). This random sample of features leads to the creation of multiple de-correlated decision trees [10].

Training our model with different classifying algorithms

In [41]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier
```

In [42]:
```
svc=SVC(kernel='rbf', gamma= 1.0, C= 0.8)
knc=KNeighborsClassifier(weights='distance', n_neighbors=3, metric= 'euclidean', algorithm='brute')
mnb=MultinomialNB()
dtc=DecisionTreeClassifier(max_depth= 200, criterion='entropy')
lrc=LogisticRegression(tol= 0.1, solver='liblinear', C=1.0)
rfc=RandomForestClassifier(n_estimators= 1000,max_features='log2', criterion='entropy')
etc=ExtraTreesClassifier(n_estimators= 100, max_features='log2', max_depth= 120, criterion='entropy')
```

In [43]:
```
clfs={
    'SVC':svc,
    'KN':knc,
    'NB':mnb,
    'DT':dtc,
    'LR':lrc,
    'RF':rfc,
    'ETC':etc
}
```

In [44]:
```
def train_classifier(clf,X_train,Y_train,X_test,Y_test):
    clf.fit(X_train,Y_train)
    Y_pred=clf.predict(X_test)
    accuracy=accuracy_score(Y_test,Y_pred)
    precision=precision_score(Y_test,Y_pred)
    f1score=f1_score(Y_test,Y_pred)
    recall=recall_score(Y_test,Y_pred)
    return accuracy,precision,f1score,recall
```

SNIPPET 9 : Classification in In-Language technique

Training our model with different classifying algorithms

In [37]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier
```

In [38]:
```
svc=SVC(kernel= 'linear', gamma= 0.8, C= 0.9)
knc=KNeighborsClassifier(n_neighbors=6, weights= "distance",metric= "minkowski")
mnb=MultinomialNB()
dtc=DecisionTreeClassifier(criterion="gini",splitter= "best",max_depth=150,min_samples_split=6)
lrc=LogisticRegression(tol= 0.001,C= 2,solver= "saga")
rfc=RandomForestClassifier(n_estimators=100,criterion="gini",min_samples_split= 3)
etc=ExtraTreesClassifier(n_estimators= 1000, max_features='log2', criterion='entropy')
```

In [39]:
```
clfs={
    'SVC':svc,
    'KN':knc,
    'NB':mnb,
    'DT':dtc,
    'LR':lrc,
    'RF':rfc,
    'ETC':etc
}
```

In [40]:
```
def train_classifier(clf,X_train,Y_train,X_test,Y_test):
    clf.fit(X_train,Y_train)
    Y_pred=clf.predict(X_test)
    accuracy=accuracy_score(Y_test,Y_pred)
    precision=precision_score(Y_test,Y_pred)
    f1score=f1_score(Y_test,Y_pred)
    recall=recall_score(Y_test,Y_pred)
    return accuracy,precision,f1score,recall
```

SNIPPET 10 : Classification in Machine Translation technique

- **EXPERIMENTAL SETUP AND RESULTS ANALYSIS**

In our project, we have used the dataset of 1028 song reviews in Assamese. They are manually labeled into two classes- positive (1) and negative (0). Then we have generated a feature matrix using TF-IDF and Count Vectorizer models. The obtained feature matrix is fed into different classifiers. We have used the following classifier in our project: SVM, Naive Bayes, Logistic Regression, Decision Tree, KNN, Random Forest, and Extra Tree Classifier. We have also attempted Machine Translation based Sentiment Analysis by first converting the reviews in Assamese to English using Microsoft Translation API. We have used an 80-20 percent split for the training and testing dataset. We just calculated accuracy by comparing the actual class labels of reviews against the predicted ones. We also compare accuracy obtained using various classifiers.

```
In [47]: for name,clf in clfs.items():
             current_accuracy,current_precision,current_f1score,current_recall=train_classifier(clf,X_train,Y_train,X_test,Y_test)
             print("For",name)
             print("Accuracy- ", current_accuracy)
             print("Precision- ",current_precision)
             print("F1 Score- ",current_f1score)
             print("Recall- ",current_recall)
             print("\n")

For SVC
Accuracy-  0.7745098039215687
Precision-  0.7222222222222222
F1 Score-  0.7222772277227723
Recall-  0.8297872340425532


For KN
Accuracy-  0.6666666666666666
Precision-  0.782608695652174
F1 Score-  0.5142857142857143
Recall-  0.3829787234042553


For NB
Accuracy-  0.7450980392156863
Precision-  0.6875
F1 Score-  0.7475728155339806
Recall-  0.8191489361702128


For DT
Accuracy-  0.7009803921568627
Precision-  0.6896551724137931
F1 Score-  0.6629834254143647
Recall-  0.6382978723404256


For LR
Accuracy-  0.7647058823529411
Precision-  0.7169811320754716
F1 Score-  0.76
Recall-  0.8085106382978723


For RF
Accuracy-  0.75
Precision-  0.7216494845360825
F1 Score-  0.7329842931937173
Recall-  0.7446808510638298


For ETC
Accuracy-  0.8137254901960784
Precision-  0.7916666666666666
F1 Score-  0.7999999999999999
Recall-  0.8085106382978723
```

SNIPPET 11 : Result in In-Language technique

```
In [41]: for name,clf in clfs.items():
             current_accuracy,current_precision,current_f1score,current_recall=train_classifier(clf,X_train,Y_train,X_test,Y_test)
             print("For",name)
             print("Accuracy- ", current_accuracy)
             print("Precision- ",current_precision)
             print("F1 Score- ",current_f1score)
             print("Recall- ",current_recall)
             print("\n")

For SVC
Accuracy-  0.835820895522388
Precision-  0.8202247191011236
F1 Score-  0.8156424581005587
Recall-  0.8111111111111111


For KN
Accuracy-  0.7512437810945274
Precision-  0.7272727272727273
F1 Score-  0.7191011235955056
Recall-  0.7111111111111111


For NB
Accuracy-  0.8606965174129353
Precision-  0.81
F1 Score-  0.8526315789473685
Recall-  0.9


For DT
Accuracy-  0.8109452736318408
Precision-  0.7888888888888889
F1 Score-  0.7888888888888889
Recall-  0.7888888888888889


For LR
Accuracy-  0.8756218905472637
Precision-  0.8571428571428571
F1 Score-  0.861878453038674
Recall-  0.8666666666666667


For RF
Accuracy-  0.8159203980099502
Precision-  0.7849462365591398
F1 Score-  0.7978142076502732
Recall-  0.8111111111111111


For ETC
Accuracy-  0.8258706467661692
Precision-  0.7894736842105263
F1 Score-  0.8108108108108109
Recall-  0.8333333333333334
```

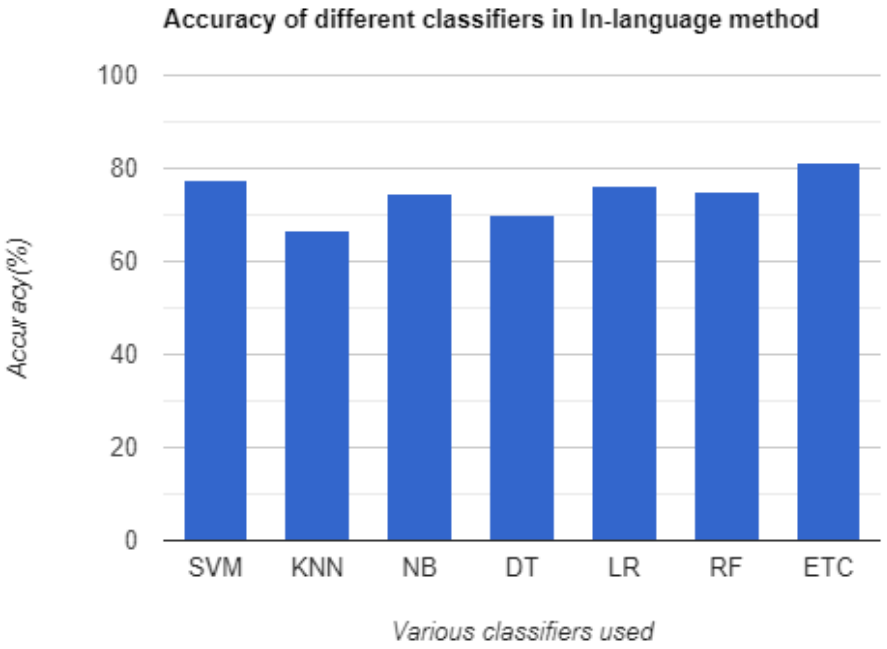
SNIPPET 12 : Result in Machine Translation technique

Classifier Used	In language (%)	Machine Translation (%)
SVM	77.4%	83.5%
KNN	66.6%	75.1%
Naive Bayes	74.5%	86%
Decision Tree	70%	81%
Logistic Regression	76.4%	87.5%
Random forest	75%	81.5%
Extra Trees	81.3%	82.5%

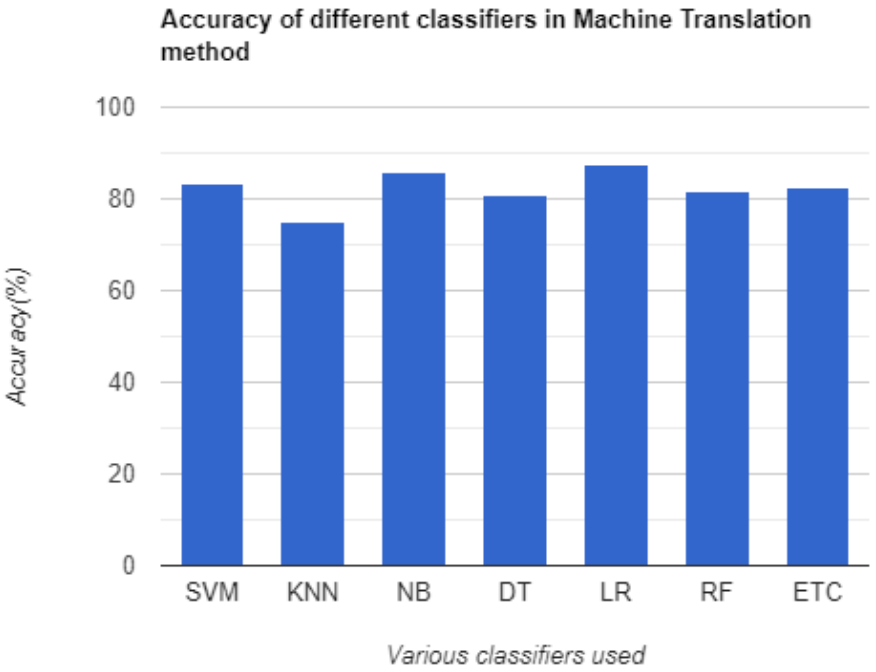
Table.1 : Accuracy obtained with various models using different classifiers

- **Graphical Analysis**

In Language



Machine Translation



- **Deployment of Model**

After training the model using the train test-split method, we get the fine-tuned model. Now to deploy the model we used Streamlit [11], which is an app framework to deploy machine learning apps built using Python. It is an open-source framework that is similar to the Shiny package in R. And Heroku [12] is a platform-as-a-service (PaaS) that enables deployment and managing applications built in several programming languages in the cloud.

Final deployed application can be viewed here:
https://assamese-sentiment-analyzer.herokuapp.com/

- ## Conclusion and Future Work

In our project, we have mainly focused on two approaches.

1. In- In language approach all training text, testing text are in the Assamese language. The feature representation (Term frequency or TF-IDF) can be varied to see the effect on In-language classification on Assamese song reviews. In this approach, we use a variety of classifiers to train and test the data. Among all the classifiers we used Extra tree gave the highest accuracy (81%) and KNN gave the lowest accuracy (66%)

2. In the Machine Translation approach, we have first translated the entire Assamese language dataset into the English language using Microsoft translator API, then we applied the text pre-processing techniques like removing stop words, punctuations, characters that are not alphanumeric, and stemming which were not properly possible in the previous approach. We have used the count vectorizer model for feature matrix generation. Finally, we used a variety of classifiers to train and test the data. Among all the classifiers we used Logistic Regression gave the highest accuracy (87%) and KNN the lowest accuracy (75%)

In the future, we can improvise our dataset to incorporate a deeper level of sentimental sense to train and test our model. With the advancement in research, our hopes are positive, that lemmatization and stemming resources will be in the future available for the native language (Assamese), to make data pre-processing cleaner and to yield better results. Further, we can expand our approach to handle negation rules which are not supported by our present models.

References

[1] Assamese stop words list. [online] available:
https://noixobdo.blogspot.com/2011/03/assamese-stop-word-list.html [Accessed 22 July 2021].

[2] A Step Towards Sentiment Analysis of Assamese News Articles Using Lexical Features.
available:https://www.researchgate.net/publication/350799278_A_Step_Towards_Sentiment_Analysis_of_Assamese_News_Articles_Using_Lexical_Features
[Accessed 22 July 2021].

[3] An Extractive Approach of Text Summarization of Assamese using WordNet.
available: http://www.tezu.ernet.in/~nlp/paper/gwc_12_word.pdf
[Accessed 25 July 2021]

[4] Sentiment Analysis on Indian Indigenous Languages: A Review on Multilingual Opinion Mining.
available: https://www.preprints.org/manuscript/201911.0338/v1
[Accessed 25 July 2021]

[5] A journey of Indian languages over Sentiment analysis: a systematic review.
available:https://www.researchgate.net/publication/329667038_A_journey_of_Indian_languages_over_Sentiment_analysis_a_systematic_review
[Accessed 25 July 2021]

[6] A Sentiment Analysis of Food Review using Logistic Regression.
available: https://www.researchgate.net/publication/334654833
[Accessed 25 July 2021]

[7] KNN Algorithm - Finding Nearest Neighbors.
available:https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_knn_algorithm_finding_nearest_neighbors.htm
[Accessed 25 July 2021]

[8] Recent Advances in Sentiment Analysis of Indian Languages.
available:https://www.researchgate.net/publication/345240744_Recent_Advances_in_Sentiment_Analysis_of_Indian_Languages
[Accessed 25 July 2021]

[9] What Is Random Forest? available:
https://careerfoundry.com/en/blog/data-analytics/what-is-random-forest/ [Accessed 28 July 2021]

[10] How to Develop an Extra Trees Ensemble with Python.
available: https://machinelearningmastery.com/extra-trees-ensemble-with-python/
[Accessed 30 July 2021]

[11] The fastest way to build and share data apps. available: https://streamlit.io/

[12] Cloud application platform | Heroku. available: https://www.heroku.com/