Classifying Toxic Comments From Social Media Posts in Banglish

Fahim Foysal Apurba
Department of Electrical and Computer Engineering
North South University
Dhaka, Bangladesh
fahim.apurba@northsouth.edu
(1921442642)

Rita Rahman
Department of Electrical and Computer Engineering
North South University
Dhaka, Bangladesh
rita.rahman@northsouth.edu
(1911212642)

Alabhia Talukder
Department of Electrical and Computer Engineering
North South University
Dhaka, Bangladesh
alabhia.talukder@northsouth.edu
(1911294642)

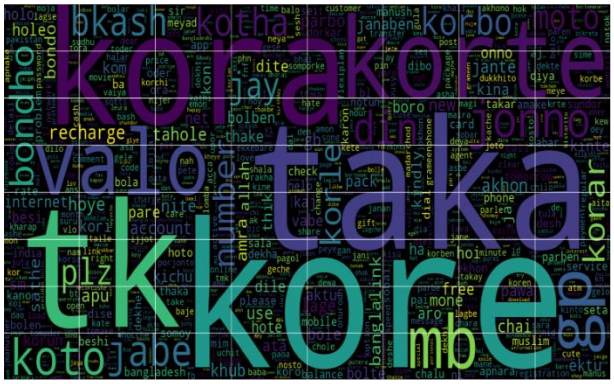

Figure 1: Seeing some Banglish words at a glance.

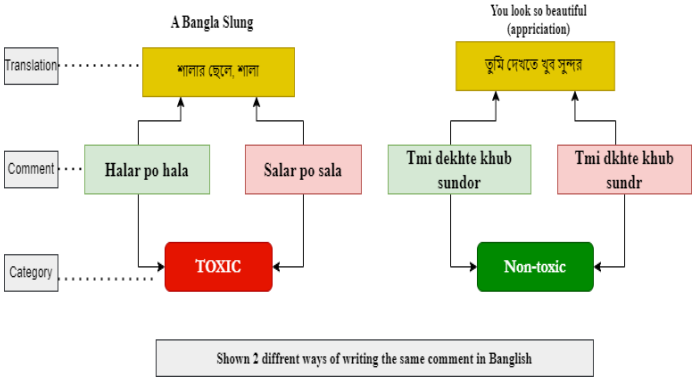

Figure 2: The diversity of Banglish sentences.

***Abstract*— In this era of social media, posting and commenting have become very freely available for people. As virtual environments have become very open clubs to enter and participate in, the tendency to abuse them is increasing. In the context of Bangladesh, both posting and commenting using Banglish have become a trend. However, the problem is that people are spreading toxic thoughts and slang using this unofficial language, which is gradually contaminating the area of the virtual environment. As Banglish is not officially recognized, it becomes difficult to detect toxic things by the social media system. So, our objective is to identify Banglish comments that are toxic. In this work, we have designed an offensive comment detection machine using Natural Language Processing (NLP) for Banglish text. Firstly, we collected an open-source dataset and added more than 20% of data from social media using a web-scraping technique, where we found the reformed dataset was quite unbalanced for two different classes. So, we balanced the dataset usingthe oversampling technique. Secondly, we thoroughly analyzed the behavior of the data and put it into the NLP pipeline. As there is no built-in library for Banglish, we had to design most of the pre-processing and feature engineering stages on our own. Finally, we applied six machine-learning models for various word-embedding. Eventually, we ended up finding that the Support Vector Classifier (SVC) with Term Frequency - Inverse Document Frequency (TF-IDF) combination was showing state-of-the-art performance with 93.08% accuracy and a satisfactory confusion matrix and ROC curve.**

Keywords—Banglish, NLP, Built-in, Oversampling, pre-processing, Feature engineering, Support Vector Classifier (SVC), Term Frequency - Inverse Document Frequency (TF-IDF), confusion matrix, ROC curve.

I. Introduction

Looking at the name, the first thing that comes to mind is what we mean by the word Banglish, as it is not officially recognized. Nowadays, instead of using either Bangla or English, using a mixture of Bangla and English has become a popular trend in Bengalis' informal communication, which is unofficially known as Banglish. Moreover, it also becomes a daily deed to use it in their informal textual communication. Specifically, we have meant that the term Banglish is a representation of Bangla pronunciation in English textual form. Here is an example: Ajke Ami Vat Khabo Na. This texting form has become very regular in social media, but the problem is that people are spreading toxic thoughts using this texting form through posting, commenting, texting, and so on. So, it has become time's demand to train the machine for Banglish text so that it can detect the type of those comments. Generally, the toxic comments are mean, hurtful, or harmful to others [1].

Moreover, it can be rude, offensive, or even include threats. So, that kind of thing creates an unsafe virtual environment. Furthermore, those things are bullying and harassment as well, which affect the flow of thoughts [2]. Besides, we know how dangerous bullying or harassment can be for our mental health. So, this is a problem. Eventually, this problem needs to be solved. For solving this problem, machine learning is one of the best ways. Because nobody can solve a problem as fast as a machine can. For example, a machine can read thousands and millions of comments within a few seconds, which is impossible for a human, but the human brain has far better understanding power and humor than even the best artificially intelligent machine on the earth. As our goal is to save energy and time, we are trying to train machines that can behave in a human-like way. Though it is a very new area for machine learning to work, our goal is to apply a machine-learning approach. NLP is needed to be used to work on this area, but there is a lake of built-in libraries for working on it. That is why, the work becomes tougher. Our approach is to use the NLP libraries for English in a very technical manner for Banglish, as the sentences are written using English symbols. In this specific area, A few established works are there using machine learning, but nobody has worked on making a web app that can detect toxic comments for Banglish text. Here is the limitation, and our work is to solve this limitation. This paper proposes a novel approach for detecting toxic Banglish-comments using machine learning. Firstly, we collected a dataset of Banglish comments through web scrapping. Secondly, we manually labeled it. Then, we preprocessed the data using an NLP pipeline, where there are several techniques like filtering, tokenizing, removing stop words, steaming, and so on. Finally, we trained and evaluated several machine learning models on the preprocessed data and evaluated the performances of the models. The best-performing model was then used to develop a web application that can detect toxic Banglish comments. As there is no established Banglish toxic comment classifier web application, ours' will be added as a contribution in this area. The contributions of this paper are as follows:

1. A novel dataset of Banglish comments which is manually labeled.
2. A way of using the NLP pipeline for preprocessing Banglish text.
3. A web application that can detect toxic Banglish comments.

Eventually, the whole work should have added as a unique contribution to this specific area of thought.

II. Related Work

Comment classification or sentiment analysis using a machine-learning approach is not a new area to work. Bangla text and toxic text classification are also old enough. But classifying toxicity in Banglish comments is a new area. The reason is that Banglish is not something officially recognized. Moreover, working in this field has started very recently. There are very few works have been done so far in this area. Though there are few journals on this topic, there is no built-in library for Banglish in the Natural Language Processing (NLP) domain [3]. However, Banglish is very commonly used by the Bengali-speaking community in informal communication. It is so popular because it is so easy to use. Actually, it is difficult for us to use pure language in our informal communication. The reason is that we feel comfortable using trendy language in our informal communication. In Bangladesh, the communicative language is a mixture of English and Bangla. The reason is that once we were under English colonialism. So, English became our second language. So, this is the estuary where Banglish was born and evolved [4].

If we observe thoroughly, we end up finding that most of the sentiment analysis works are being done based on the English language text. So, working on the Bangla language is now a time's demand for making our language more versatile. Recurrent neural network models are the best performers for doing such stuff. Specifically, Long short-term memory (LSTM) networks perform superbly in such situations [5]. In earlier works, text mining for sentiment analysis from a supervised learning dataset was one of the best techniques. Naive Bayes was the best-performing algorithm for text classification [6]. In the case of Bangla text classification, Bangla texts needed to be converted to their equivalent English form. As follows, the trend was to convert the Banglish text into Bangla first and Bangla to English then. So, it was a very time and energy-consuming process. We resolved this problem in our task by keeping Banglish text as it is for training machine learning models. Transliteration is a process of converting words from one language's alphabet into words from another language's alphabet by substituting similar-sounding letters. Banglish is something like that. Shourov Kabiraj et al.[7] used phrase-based algorithms that divide an input sentence into a collection of words and phrases using neural machine translation. Obviously, this is something done out of the box, which is very effective for the large dataset. Though it is effective for large datasets, it is not optimal for small datasets. Because too much unnecessary work has to be done. We also overcame this problem in our task.

M. F. Mridha et al. [8] detected misspelled textual content in various social media for Bangla and Banglish text. They used many boosting algorithms and ended up finding that

L-boost was working Best. Moreover, they applied neural network models named BERT and LSTM as well. Though his neural network models helped them to find excellent performance, they had to sacrifice too much time and energy consumption. We solved this problem using machine learning models, where we had to sacrifice a small amount of performance.

R. Basri et al. [9] did the same thing as M. F. Mridha did. They also use a neural network model named CNN. CNN is excellent for making good performances but takes too much time and energy consumption.
N. Dey et al. [10] proposed a technique to detect events from Bangla and Banglish comments. Here, they collected the comments from Facebook through web-scrapping. Then, they apply NLP and machine learning models. They showed a pretty good performance. We are not working on event detection, but we made some performance better than them.

S. Hosain Sumit et al. [11] showed a polarity analysis for Bangla text. Here, they showed specialty in the word-embedding part. They tested existing cutting-edge word embedding approaches. Actually, they used the Word2vec Skip-Gram and Continuous Bag of Word to Index model. Here, the accuracy was good but not better than us. In the end, our goal was to analyze all those works and make a better on.

III. Methodology

In this methodology section, we show how we did the whole work. Firstly, we analyzed the data. Then, we realized that the data needed to be balanced. We balanced the data using the oversampling technique. Secondly, we preprocessed the data. We applied filtration, tokenization, stop-word removal, steeming, and lematization techniques. Thirdly, we applied two types of word-embedding techniques. TF-IDF and CountVectorizer. Fourthly, we split the data into test and train datasets. Fifthly, we train six different machine learning models using train data. Finally, we evaluate the models on test data.

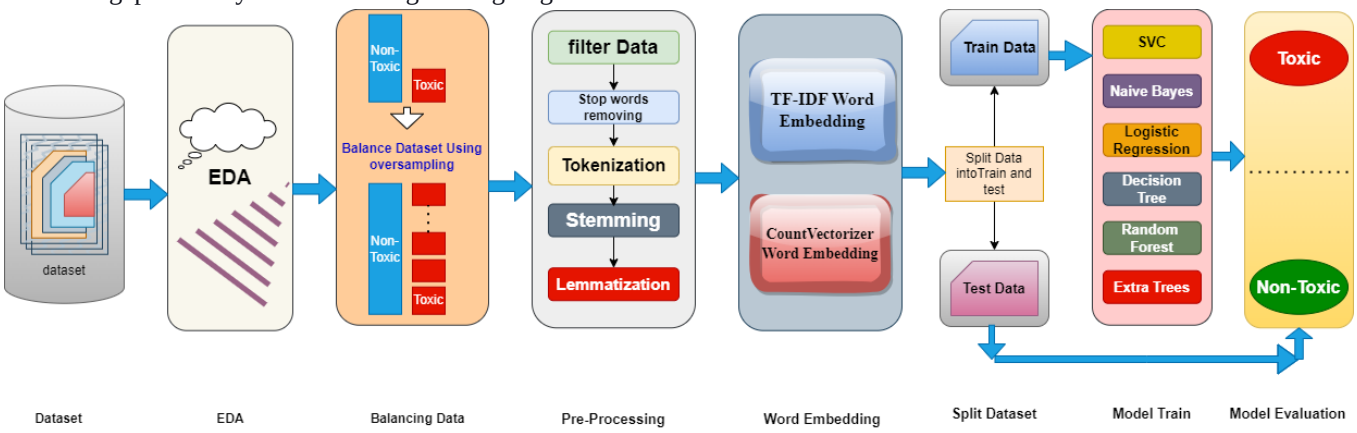

Figure 3: Whole process's flow diagram.

A. Data Collection

We collected a dataset from an open source. This dataset contains 5000 rows of comments. Then, we manually labeled the dataset. After labeling the dataset, there were five features. Here are the followings below: 1. ID, 2. Text-comment, 3. Positive, 4. Toxic, 5. Neutral.
Here, ID is a continuous value. Text-comment is the Banglish comment. Positive is a binary categorical feature. It holds 1 or 0. If the text comment is positive, it takes 1. If the text comment is negative, it takes 0. Toxic and Neutral features are the same as the feature Positive.
We didn't feel that this dataset was big enough to do our work. So, we collected more than 1100 rows of data from Facebook using the web-scraping technique. Then, we manually cleaned the dataset as much as possible and labeled it as well. After that, we merged two datasets and made a new dataset. The new dataset contains 6188 number of rows.

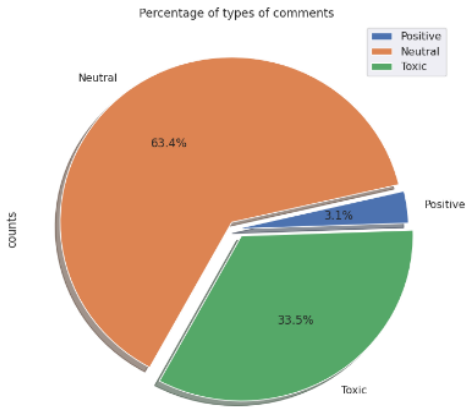

Figure 4: Distribution of texts based on their categories.

In Figure 4, from exploratory data analysis, we got to know that there were 3923 Neutral comments, 2072 toxic comments, and only 193 positive comments. So, we merged the neutral and positive columns and made a new feature named Non-toxic. Then, we got the data is better in shape than before. We show the distribution below in Figure 5.

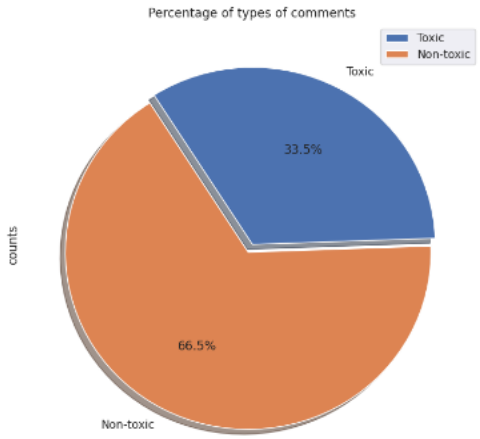

Figure 5: Distribution for only two classes- Toxic and Non-toxic.

Now, there are 4116 Non-toxic comments and 2072 Toxic comments, which is 2:1 in proportion.

B. Feature Engineering

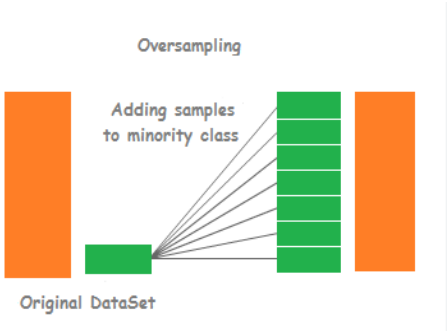

Figure 6: Oversampling visualization.

From EDA, we realized that the data needs to be balanced. So, we applied the oversampling balancing technique to balance the data. In the oversampling technique, the majority class remains the same. On the other hand, the minority class's data is copied several times and added to the dataset until the number of rows of the minority class becomes equal to the majority class.

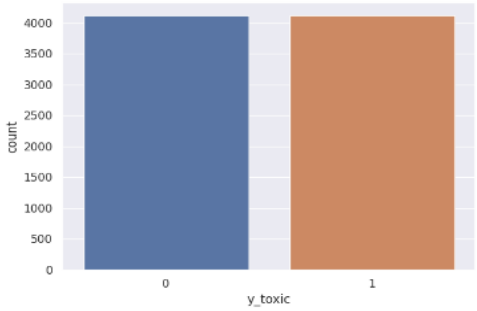

Figure 7: Showing the distribution, after balancing data.

In Figure 7, we can see that both of the classes have the same number of rows. Both contain 4116 number of rows. So, the distribution is 1:1 in proportion. Moreover, after oversampling the number of rows increased and became 8232.

C. Preprocessing

We followed five stages for preprocessing our text data. The five stages are filtration, stop-word removal, tokenization, stemming, and lemmatization.

1. Filtration

Firstly, we convert all the letters of the dataset into lowercase form. Secondly, we remove emails, URLs, punctuations, emojis, non-ASCII and extra spaces. Finally, we cleaned the number and repeating characters. It was how we filtered the dataset and converted it into a very useful form.

2. Stop-word Removing

Firstly, we found out the most frequent 50 words. From the frequent words, we separated the stop-words and removed them. Additionally, from the experience of labeling the dataset, we knew about many stop-words. So, we removed those words. As English words are very common in Banglish sentences, we also removed the stop word of English using the built-in library's functions**.** Here is the stop-words list below in Figure 8.

```
["ki","kii", "hmm", "hu", "tomar", "tumi", "ke", "je", "se", "hae", "he", "are",
 "re", "ne","1e", "tui", "tor", "tar", "kar," "jar", "ei", "ta", "la", "ra",
 "tara", "kara", "jara", "O", "Oo", "ho", "Ooo..","na", "er", "e", "a", "ar",
 "to", "r", "te", "hi", "he", "ami", 'amar', "amr", "ti","amader","k", "c", "d",
 "f", "g","h", "i", "j", "l", "m", "n", "O", "p", "q", "r","s", "t", "u", "v",
 "w", 'x', "y", "z","vai", "ai", "hy", "nai", "sb", "sob", "pabo", "onek", "onak",
 "hoy", "ase", "pase", "gulo", "ekta", "akta", "onek", "onk", "oi", "ou", "or",
 "ko", "kha", "ga", "gha", "dhor", "sor", "nor", "jodi", "tai", "jeno", "jodi",
 "emonki", "amonki", "emni", "amni", "tobe", "cai", "tomar", "apni", "apnader",
 "tumi", "tader", "jar", "jader", "eta", "ota", "oita", "iha", "uha", "jaha",
 "etai", "otai", "eita", "eitai", "kno", "kivabe", "kiser", "kar","kahar", "kobe",
 "kothay", "kake", "niye", "accha", "assa", "kokhon", "kemne", "kmne", "tumi", "tmi",
 "ache", "accha", "keno", "diye", "hobe", "hbe", "assa", "bar", "offer", "sim", "kono",
 "por", "trpr", "tarpor", "trpor", "jevabe", "apnar", "amr", "amar", "jai", "sob",
 "but", "kintu", "ace", "cai", "thakle", "age", "por", "pore", "pari", "nai", "nei",
 "jevabe", "jemon", "jodi", "sob", "tumi", "jodi", "ai", "eivabe", "oivabe", "kobe",
 "theke", "ek","1", "eto", "ato", "ber", "der", "naki", "eikhane", "eivabe", "kina"]
```

Figure 8: Stop-word's list

3. Tokenization

Then, using the built-in library functions, we tokenized the sentences. Tokenization means breaking the sentences into words and putting unique values for every distinct word. The reason is that machines cannot recognize the words. They can recognize numbers easily. Here, we showed a frequency table of the most frequent 50 words in Figure 9.

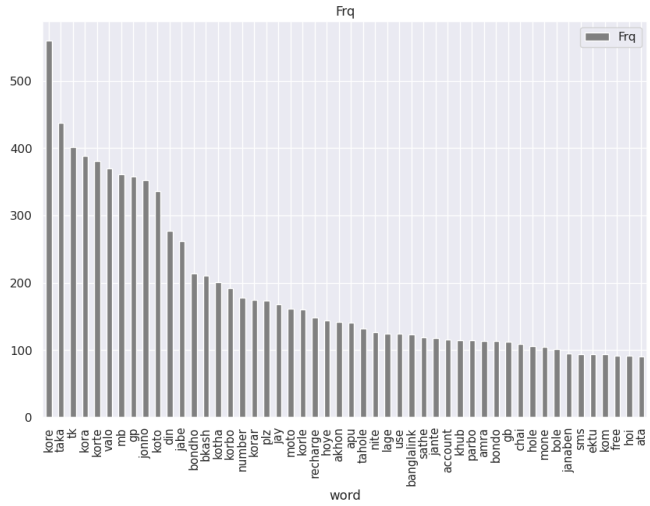

Figure 9: Frequency table of most frequent 50 words.

4. Stemming & Lemmatization

Here, we didn't do anything special. We just used the built-in libraries for English and completed the stemming and lemmatization. In stemming, we stem the words to extract the base form. In the same way, we do lemmatize the words to extract the root or lemma form of a word. Both are pretty similar.

D. Word Embedding

We need word embedding to represent the words and documents in a numeric vector form. Word embedding allows similar words to have the same representation by putting the word's real-valued vector in a lower dimensional space.

In our work, we tried two different types of word embedding. 1. Count Vectorizer and 2. Tf-IDF (Term frequency-inverse document frequency)
Between them, TF-IDF is working better.
We can divide TF-IDF into two parts. TF (Term Frequency) and IDF (Inverse Term Frequency).

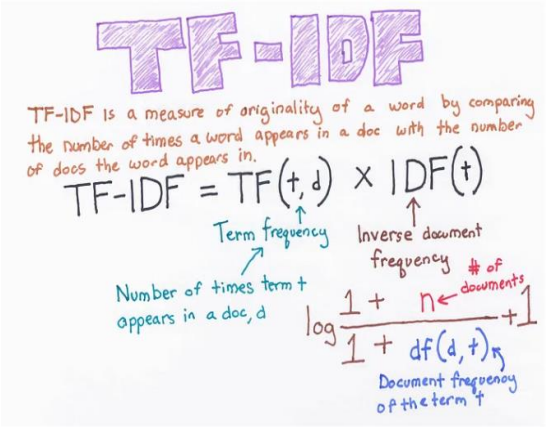

Figure 10: TF-IDF word embedding.

Here,

$$TF(i,j) = \frac{Term\ i\ frequency\ in\ document\ j}{\text{Total number of terms in document j}}$$

$$\mathbf{IDF(i) = \log(\frac{\text{Total documents}}{\text{Number of documents contains term i}})}$$

By merging them,

$TF\text{-}IDF(i,j) = TF(i,j) \times IDF(i)$

E. Splitting Dataset into Test and Train

We had 6188 rows in the dataset. We split it into test and train, where the test set contains 20% of the data and the train set contains 80% of the dataset. Here, 20% is approximately 1600 rows in the test set. On the other hand, 80% is approximately 4500 rows in the train set.

F. Train Models

We trained six different machine-learning models using the training dataset. Two of them were ensemble models, and the rest were individual machine-learning models. They were-

1. Random Forest,
2. Extra Trees,
3. Decision Tree,
4. Logistic Regression,
5. Naive Bayes, and
6. SVC (Support Vector Classifier).

1. Random Forest

It creates multiple sets from the dataset using random sampling. Then, it generates multiple decision trees from the different sets. As it is a bagging technique, it combines all the output by averaging as a single result.

2. Extra trees

It works like Random Forest, but it generates multiple decision trees for the original dataset instead of making different sets.

3. Decision Tree

It is a support hierarchical model that shows an algorithm that solely consists of conditional control statements by using a tree-like model of decisions and their potential outcomes, including chance event possibilities.

4. Logistic Regression

It works well for binary classification by using the nature of the sigmoid function.

5. Naive Bayes

Naive Bayes techniques are a type of algorithm that employs Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the class variable value.

6. SVC (Support Vector Classifier)

It is a particular application of the Support Vector Machine algorithm created with classification problems in mind. It has a soft margin and a hard margin. The hard margin is the fitted line, and the soft margin is to get rid of the outliers.

Cost function of SVC below:

$$(\theta) = C\sum_{i=1}^{m}\left[y^{(i)}cost_1(\theta^T x^{(i)}) + (1-y^{(i)})cost_0(\theta^T x^{(i)})\right] + \frac{1}{2}\sum_{i=1}^{n}\theta_i^2$$

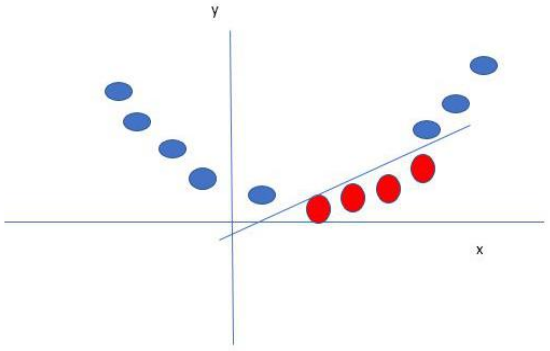

Figure 11: Mapping 1D data to 2D to become able to separate the two classes

G. Model Evaluation

Finally, we evaluated the performance of models and find out the best-performing model. We evaluated the models on the test dataset. For evaluation, we used accuracy, F1- score, precision, recall, confusion matrix, AUC score, and ROC curve.

IV. Result Analysis

A. Accuracy

Here is the accuracy table below for TF-IDF and Count Vectorizer word embedding:

	TF-IDF		**Count Vectorizer**
Models			
Random Forest	89.50%		87.25%
Extra Trees	92.25%		88.75%
Decision Tree	86.94		84.25%
Naïve Bayes	88.67%		81.33
Logistic Regression	86.25%		82.75%
****SVC**	**93.08%**		73.00%

So, from that table we have reached to the conclusion that SVC is working best among 6 models, after TF-IDF word embedding.

B. Precision, Recall and F1-Score

Here is the table below for our best-performing model SVC:

Label	Precision	Recall	F1-Score
0	0.88	1.00	0.93
1	1.00	0.86	

So, from the table we can say that the it is predicting both of the classes very well.

C. Confusion Matrix

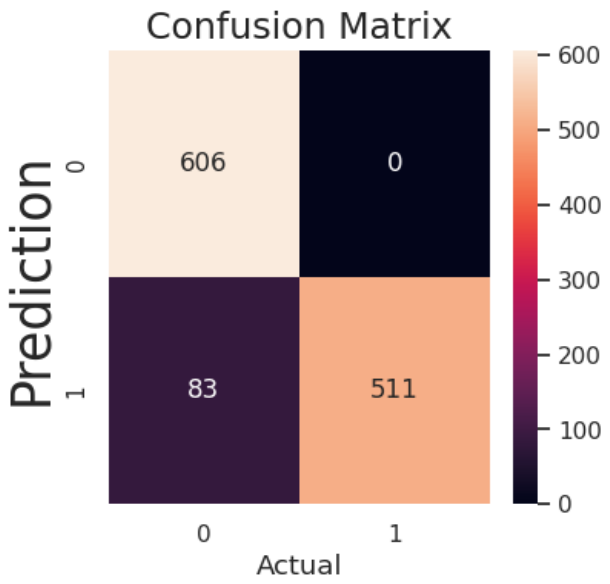

Figure 12: The confusion matrix.

From the Confusion matrix we can draw the conclusion that the model is working very well for toxic data. All of the toxic comments are predicted correctly.

D. ROC-AUC Score, ROC Curve

Here, the ROC-AUC score is 0.93.

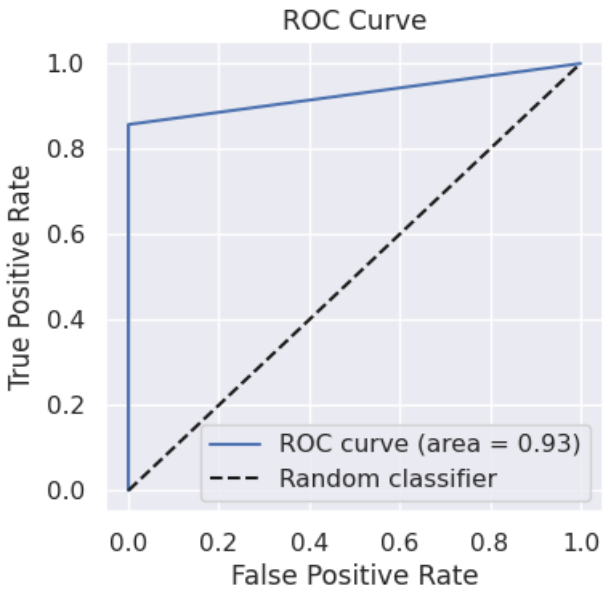

Figure 13: ROC curve

From the curve, we can make the statement that it is showing state-of-the-art performance.

V. Conclusion

Here, we showed different types of machine-learning approaches. Firstly, we analyzed the data. Then, we end up finding that we need to balance the dataset. So, we balance the data. Secondly, we tried two different types of word embedding techniques and got TF-IDF is working best. Thirdly, we trained six different models, which were machine-learning probability algorithms. Then, we evaluated those models using a test dataset. Finally, we reached the conclusion that after TF-IDF word embedding, SVC is working best with 93.08% accuracy.

VI. References

[1] "N. Dey, Md. S. Rahman, M. S. Mredula, A. S. M. S. Hosen, and I.-H. Ra, “Using Machine Learning to Detect Events on the Basis of Bengali and Banglish Facebook Posts,” Electronics, vol. 10, no. 19, p. 2367, Sep. 2021, doi: 10.3390/electronics10192367.".

[2] "] Rahul, H. Kajla, J. Hooda and G. Saini, "Classification of Online Toxic Comments Using Machine Learning Algorithms," 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 2020, pp. 1119-1123, doi: 10.11".

[3] "M. S. Hossain, N. Nayla and A. A. Rassel, "Product Market Demand Analysis Using NLP in Banglish Text with Sentiment Analysis and Named Entity Recognition," 2022 56th Annual Conference on Information Sciences and Systems (CISS), Princeton, NJ, USA, 2022, p".

[4] "Mostafa, Massrura, and M. Jamila, "From English to Banglish: Loanwords as opportunities and barriers?: Is English swamping the Bangla language?", English Today 28.2 (2012): 26-31.".

[5] "A. Hassan, M. R. Amin, A. K. A. Azad and N. Mohammed, "Sentiment analysis on bangla and romanized bangla text using deep recurrent models," 2016 International Workshop on Computational Intelligence (IWCI), Dhaka, Bangladesh, 2016, pp. 51-56, doi: 10.1109/".

[6] "R. A. Tuhin, B. K. Paul, F. Nawrine, M. Akter and A. K. Das, "An Automated System of Sentiment Analysis from Bangla Text using Supervised Learning Techniques," 2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS), Singapore".

[7] "Kabiraj, Shourov, S. Waheed, and Z. A. Khaled. "Transliteration from Banglish to Bengali Language Using Neural Machine Translation." Proceedings of the Fourth International Conference on Trends in Computational and Cognitive Engineering: TCCE 2022. Singap".

[8] "M. F. Mridha, M. A. H. Wadud, M. A. Hamid, M. M. Monowar, M. Abdullah-Al-Wadud and A. Alamri, "L-Boost: Identifying Offensive Texts From Social Media Post in Bengali," in IEEE Access, vol. 9, pp. 164681-164699, 2021, doi: 10.1109/ACCESS.2021.3134154.".

[9] "R. Basri, M. F. Mridha, M. A. Hamid and M. M. Monowar, "A Deep Learning based Sentiment Analysis on Bang-lish Disclosure," 2021 National Computing Colleges Conference (NCCC), Taif, Saudi Arabia, 2021, pp. 1-6, doi: 10.1109/NCCC49330.2021.9428849.".

[10] "N. Dey, M. Rahman, M. Mredula, A. Hosen, and I. Ra. (2021). “Using machine learning to detect events on the basis of Bengali and banglish Facebook posts”, in Electronics, 10(19), 2367.".

[11] "S. Hosain Sumit, M. Zakir Hossan, T. Al Muntasir and T. Sourov, "Exploring Word Embedding for Bangla Sentiment Analysis," 2018 International Conference on Bangla Speech and Language Processing (ICBSLP), Sylhet, Bangladesh, 2018, pp. 1-5, doi: 10.1109/ICBS".

[12] "M. S. Hossain, N. Nayla and A. A. Rassel, "Product Market Demand Analysis Using NLP in Banglish Text with Sentiment Analysis and Named Entity Recognition," 2022 56th Annual Conference on Information Sciences and Systems (CISS), Princeton, NJ, USA, 2022, p".