Hate Speech and Offensive Content Identification in Multiple Languages using machine learning algorithms

Dikshitha Vani V, B. Bharathi

Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam, Tamil Nadu 603110

Abstract

The freedom of expression on social media sites like Twitter and Facebook provides opportunities for people to voice out their opinions and concerns. At the same time, it has also become a tool for immense bullying and hateful comments online. AI tools are methods used to identify such comments automatically . These identification tools are evaluated by continuous experimentation with data sets. The HASOC track (Hate Speech and Offensive Content Identification) is dedicated to developing benchmark data for this purpose. This paper presents the HASOC task for Offensive Language Identification in Marathi. The data set was assembled from Twitter. This task has 3 subtasks. Subtask A is Offensive Language Detection where the goal is to discriminate between offensive and non-offensive posts. In subtask B, only the posts labeled as Offensive (OFF) in subtask A are included and the goal is to predict the type of offense as either Targeted Insult(TIN) or Untargeted (UNT). In subtask C, only posts that are either insults or threats (TIN) are considered in this third layer of annotation and classifies them on the target of offenses as Individual (IND), Group (GRP), and Other (OTH). In this work, our team ssncse_nlp have applied machine learning prediction algorithms - Random forest (RF), Support Vector Machine(SVM), Logistic Regression, and k nearest neighbors (KNN) classifier algorithms along with count vectorized features to the tweets for classification. Finally, the result shows that Random Forest predicts the labels for subtasks A and C more accurately than the other classifier models with a Macro F1 score of 0.9745 and 0.7929 while the Logistic Regression classifier predicts more accurately for subtask B with a Macro F1 of 0.6958.

1. Introduction

Social media is an active tool for people of all ages. They generate and consume a great deal of data. Social media has gained massive support and reach over the years because it helps people express themselves and their ideologies, showcase their talents, meet new people, connect themselves to others' stories, etc. On the other hand, it also means that people are openly critical of others and "impose" their ideologies on others. When ideologies clash, they introduce insulting, hurtful, derogatory or obscene language. Such objectionable content can even be a threat to democracy. Open societies need to find an acceptable way to react to such content without imposing rigid censorship regimes.

Forum for Information Retrieval Evaluation, December 9-13, 2022, India

[†] These authors contributed equally.

dikshithavani2010541@ssn.edu.in (D. V. V); bharathib@ssn.edu.in (B. Bharathi)

Online hate speech can be produced and distributed easily, at low cost and anonymously, while having the potential to reach a globally diverse audience in real time. The relative permanency of online content is also an issue when hate speech can reappear and (re)gain popularity over time.

As a consequence, many social media websites monitor user posts. This leads to a pressing demand for methods to automatically identify suspicious posts. Online communities, social media enterprises, and technology companies have been investing heavily in technology and processes to identify offensive language and prevent abusive social media behavior. However, there is increasing evidence that social media platforms still struggle to keep up with the demand for technology to identify offensive content, particularly for languages other than English. This paper aims to detect hate speech in the Marathi Language.

Marathi is an Indo-Aryan language predominantly spoken by Marathi people in the Indian state of Maharashtra. It is the official language of Maharashtra, and a co-official language in Goa state and the territory of Daman, Diu & Silvassa. It is one of the 22 scheduled languages of India, with more than 90 million speakers. Marathi ranks 10th in the list of languages with the most native speakers in the world. Marathi has the third largest number of native speakers in India, after Hindi and Bengali. The language has some of the oldest literature of all modern Indian languages.

In this paper, we aim to perform the classification of data based on a statistical analysis of test data using a count vectorizer. It outlines four classifier models along with a count vectorizer, namely random forest (RF), support vector machine (SVM), Logistic Regression, and K nearest neighbors (KNN).

The rest of the paper is organized as follows-Section 2 describes other related work on Hate Speech. The dataset for the shared task and machine learning algorithms used for this task are described in Section 3. Section 4 discusses the 4 classifier models used in this paper. Results are presented in Section 5. Section 6 emphasizes improvements that can be applied to the model. Section 7 concludes the paper.

2. Related Works

Identifying the vulgarity of comments on social media and classifying them has become an important field of study today to ensure security, safety, peace, and harmony among people. Using Classifier models coupled with a count vectorizer is one of the most commonly used models to detect hate speech. Ensemble models, a machine learning approach to combine multiple other models in the prediction process, is also commonly used to improve classification performance.

[1], [2], [3], [4], and [5] uses classical classifier models and ensemble models for the classifications of data(coupled with count vectorizers or TF-IDF vectorizers). The following

paragraph gives a brief description or names the models used in the papers listed above.

In [1], authors have used Ensemble Learning models such as Random Forest and AdaBoost coupled with count vectorizer for novel hate Speech Detection where Random Forest yielded 95% accuracy. They also used word cloud for displaying the most prominent tweets responsible for hateful sentiments. [2] discusses hate speech detection in Indonesian language. The authors have used five stand-alone classification algorithms namely Naïve Bayes, K-Nearest Neighbours, Maximum Entropy, Random Forest, and Support Vector Machines, and two ensemble methods namely hard voting and soft voting on Twitter hate speech database. They aimed to prove that the ensemble method can improve classification performance. In [3], the authors have devised a system that uses both machine learning and deep learning techniques to detect the offensive comments. [4] describes the usage of SVM and Naive Bayes classifier for Hate speech models and the results showed a classification accuracy of approximately 99% and 50% for SVM and NB respectively over their test set. Similarly [5] uses Logistic Regression and XGBoost classifiers.

Apart from the classifier models that are combined with count vectorizer or TF-IDF vectorizer, transformer models are also widely used for Hate Speech Detection. [6] uses multilingual pre-trained models. In this paper, for the detection of Hindi and Marathi languages, the MBERT model was used, and for the English language, the BERT model was used.

3. Proposed Methodology

The proposed system of detecting offensive content from the HASOC 2022 task 3 data is described in the following sections. The steps involved in the proposed system are as follows:

1. Data set exploration and preprocessing
2. Feature extraction
3. Model training and Testing

3.1 discusses the data set in detail, 3.2 discusses the need for feature extraction and finally 3.3 discusses the training and testing procedures.

3.1. Data Set exploration and Preprocessing

The data set given by the shared task organizers contains a training set consisting of 3103 instances and a testing set consisting of 510 instances. The training set contains the ID -identity of the user and Tweet while the test set consists of ID and tweet.

Each instance of the training set also contains up to 3 labels each corresponding to one of the following levels:
- Level (or sub-task) A: Offensive language identification;
- Level (or sub-task) B: Automatic categorization of offense types;
- Level (or sub-task) C: Offense target identification.

The training set is splitted into 3 CSV files for carrying out each of the subtasks separately. Subtask A contains all 3103 instances while subtask B and C contains 1068 and 740 instances respectively. The data set is split so that none of the tasks contain a null row entry in them. Each of these files contains ID, Tweet and label of corresponding level/task.

Our team has not done any preprocessing of the data set for Marathi language texts.
The texts can be preprocessed by removing stopwords from the text. Stopwords are words that do not provide any information for the classification but however increase the dimension of the matrix provided by the count vectorizer. This is explained in the section 6.

The data set has been retrieved from [7].

3.2. Feature extraction

Machines cannot understand characters and words. In order to use textual data for predictive modeling, words need to be encoded as integers or floating-point values for use as inputs in machine learning algorithms. This process is called feature extraction.

We have used Scikit-learn's count vectorizer to convert a collection of text documents to a vector of term/token counts. It enables the pre-processing of text data prior to generating the vector representation. This functionality makes it a highly flexible feature representation module for text.

The Count vectorizer is imported from sklearn.feature_extraction.text module. The training set is given to fit_transform function to fit the data into the model and test data set is used to transform the given set according to the fitted model from training set.

3.3. Model Training and Testing

The model is trained and tested using 4 Classifier models:

1. RandomForest
2. SVM classifier
3. Logistic Regression
4. KNN classifier

In the following subsection 3.3.1, a major pre-processing step: Label encoding is discussed. In the next subsection 3.3.2, to study these 4 different classifier models we brief in the importance of splitting the training data set.

3.3.1. Label Encoding

Label Encoding is a popular encoding technique for handling categorical variables. In this technique, each label is assigned a unique integer based on alphabetical ordering. Label

Encoding converts the labels into a numeric form to bring them into machine-readable form. Machine learning algorithms can then decide in a better way how those labels must be operated. It is an important preprocessing step for the structured dataset in supervised learning. This is achieved using LabelEncoder() from sklearn.preprocessing header. Since Scikit-Learn library is used, there no need for performing label encoding separately as the classifier object takes care of that by itself.

3.3.2. Data Split

The tweet field of the training data(features) and the label of subtask(labels) are used to perform a train and test data split.

This splitting is necessary to check the correctness of the proposed model. The splitted training set fits into a model based on the classifier. The tweets of the splitted test set are given to the fitted model to predict the labels of the subtask. The predicted results are then compared with the actual labels of the splitted test set using the metrics -> classification report, confusion matrix and accuracy score.
Empirical studies show that the best results are obtained if we use 20-30% of the data for testing, and the remaining 70-80% of the data for training. The paper "Why 70/30 or 80/20 Relation Between Training and Testing Sets: A Pedagogical Explanation"[1] by the University of Texas at El Paso discusses the same.

Each of these classifier models is discussed in the next section.

4. Classifier Models

This section is further divided into four sections. Section 4.1 describes Random Forest classifier model, section 4.2 describes Support vector machine classifier model, section 4.3 describes Logistic regression classifier model, and section 4.4 describes K nearest Neighbors classifier model.

4.1. Random Forest classifier

Random forest is imported from the Ensemble module. Ensemble means combining multiple models i.e a collection of models is used to make predictions rather than an individual model. Ensemble uses two types of methods:

1. Bagging– It creates a different training subset from sample training data with replacement and the final output is based on majority voting.
2. Boosting– It combines weak learners into strong learners by creating sequential models such that the final model has the highest accuracy.

The random forest algorithm is based on bagging principle.
Following are the steps involved in Random Forest Algorithm:

[1] Source: https://scholarworks.utep.edu/cgi/viewcontent.cgi?article=2202&context=cs_techrep

- n random records are taken from the data set.
- Individual decision trees are constructed for each sample.
- Each decision tree will generate an output.
- Final output is based on Majority Voting for Classification.

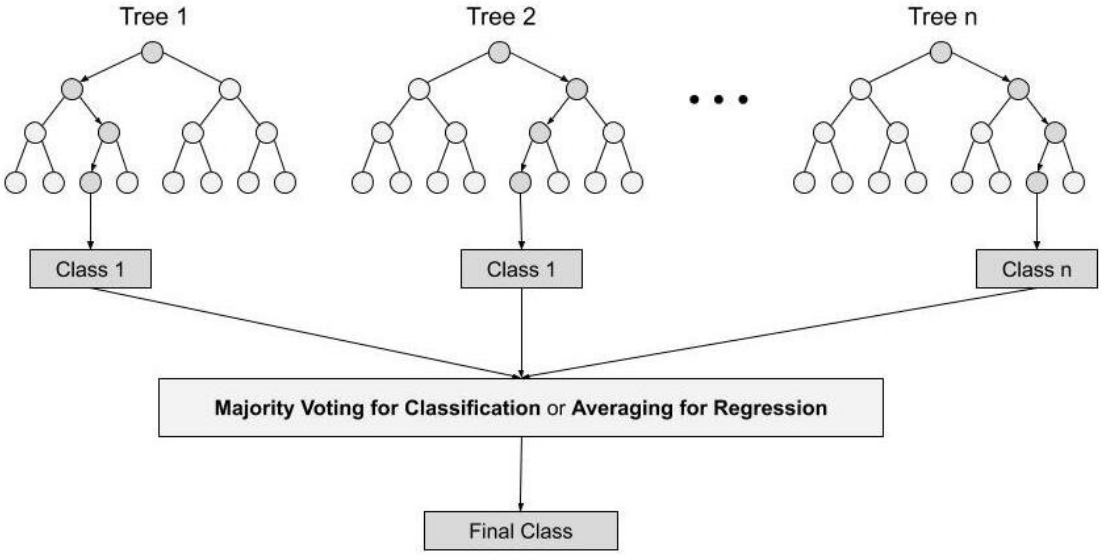

Figure 1: Steps involved in Random forest Classification[2]

The table 4.1 describes the hyperparameters used.

Hyperpameter	Function
n_estimators	The number of trees the algorithm builds before averaging the predictions
random_state	controls the randomness of the bootstrapping of the samples used when building trees

To train the model the splitted training set is fit into the regressor. The splitted test set is then used to predict the labels.

The table 1 compares the test prediction and actual results shown by the metrics Macro F1, precision, and accuracy.

4.2. Support Vector Machine

Support Vector Machine (SVM) is a supervised machine learning algorithm that can be used for classification problems. In this algorithm, data items are plotted as a point in n-dimensional space, n being the number of features, and the value of each feature is the value of a particular coordinate. Classification is performed by finding the hyper-plane that

[2]Source: https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest

Table 1
Classification Report - A comparison of test prediction with actual labels of the subtasks

Subtasks	Macro F1	Macro Precision	Accuracy
Subtask A	0.87	0.89	0.88
Subtask B	0.52	0.68	0.68
Subtask C	0.36	0.50	0.71

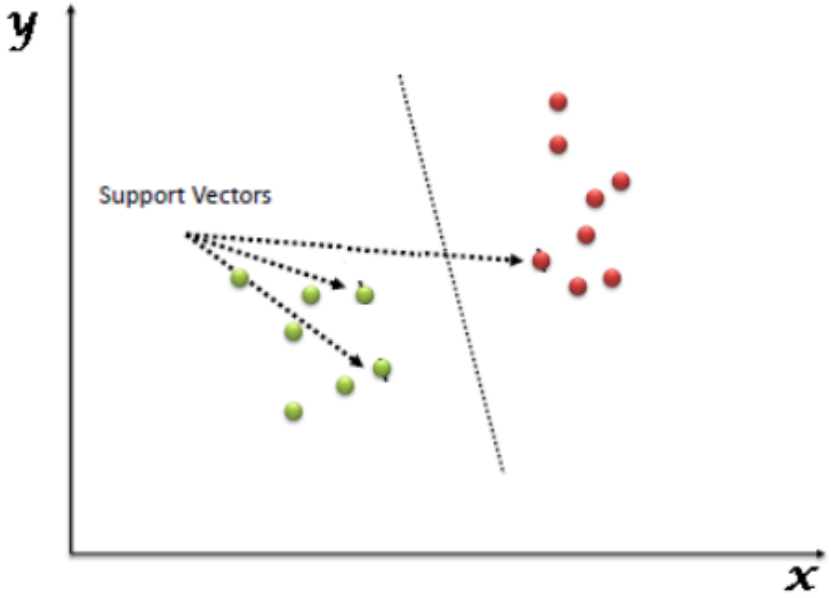

Figure 2: Hyperplane differentiating between 2 classes[3]

differentiates the two classes as shown in figure 2.

If the hyperplane classifies the dataset linearly then the algorithm is called SVC and the if algorithm separates the dataset by non-linear approach it is called SVM as depicted in figure 3

To train the model the splitted training data set is fit into the SVC classifier. The splitted test set is then used to predict the labels.

The table 2 is the comparison between the test prediction and actual results shown by the metrics Macro F1, precision and accuracy.

[3]Source: https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/
[4]Source: https://www.analyticsvidhya.com/blog/2021/06/support-vector-machine-better-understanding/

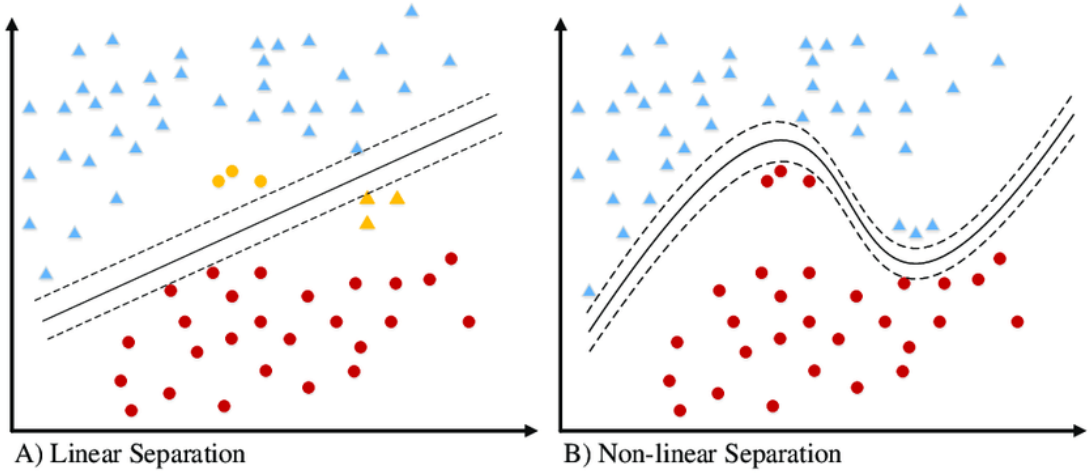

Figure 3: Difference between SVM and SVC classifiers[4]

Table 2
Classification Report - A comparison of test prediction with actual labels of the subtasks

Subtasks	Macro F1	Macro Precision	Accuracy
Subtask A	0.87	0.88	0.88
Subtask B	0.40	0.33	0.66
Subtask C	0.33	0.50	0.71

4.3. Logistic Regression

Logistic regression is a supervised classification algorithm where the target variable can take only discrete values for a given set of features. The model builds a regression model to predict the probability that a given data entry belongs to the category. Logistic regression models the data using the sigmoid function. [5]

Based on the number of categories, Logistic regression can be classified as:

1. Binomial: target variable can have only 2 possible types: “0” or “1” .
2. Multinomial: target variable can have 3 or more possible types which are not ordered like "Type A” vs “Type B” vs “Type C”.
3. Ordinal: it deals with target variables with ordered categories.Each category can be given a certain value like 0,1,2 etc. Example a test result can be classified as "poor","good","better" etc.

To train the model, the splitted training data set is fit into the Logistic Regression model. The splitted test set is then used to predict the labels.

[5]Source: https://www.geeksforgeeks.org/understanding-logistic-regression/
[6]Source: https://www.researchgate.net/publication/239269767_Artificial_Neural_Networks_in_Multivariate_Calibration/figures?lo=1

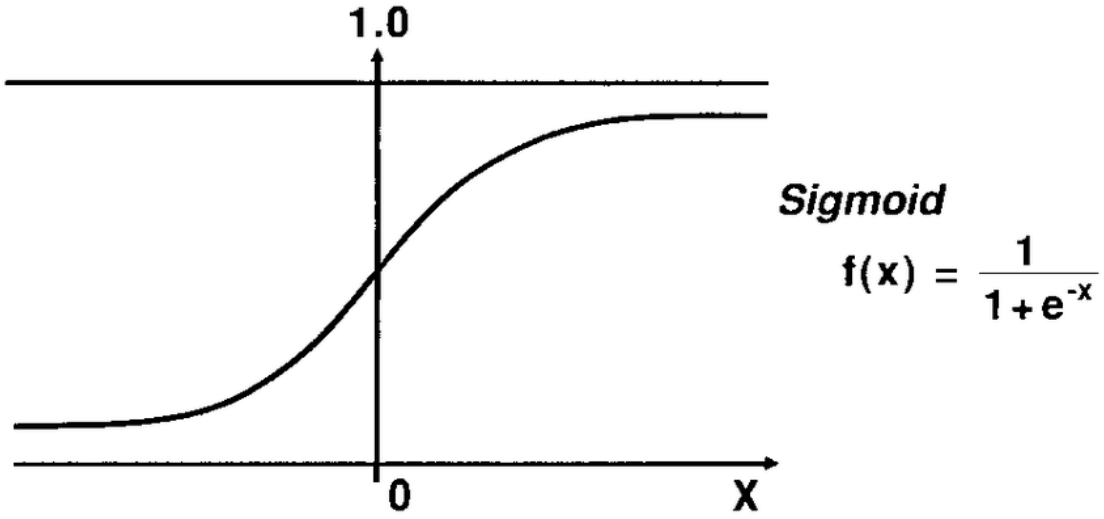

Figure 4: Sigmoid function[6]

The table 3 is the comparison between the test prediction and actual results shown by the metrics Macro F1, precision and accuracy.

Table 3
Classification Report - A comparison of test prediction with actual labels of the subtasks

Subtasks	Macro F1	Macro Precision	Accuracy
Subtask A	0.87	0.88	0.88
Subtask B	0.62	0.63	0.67
Subtask C	0.43	0.45	0.68

4.4. K-nearest Neighbors

K-nearest neighbors (KNN) is a supervised learning algorithm used for classification. KNN tries to predict the correct class for the test data by calculating the distance between the test data and all the training points. K number of points that are closet to the test data is selected. The algorithm calculates the probability of the test data belonging to the classes of 'K' training data and the class holds the highest probability will be selected.

The figure 5 depicts the different steps involved in the KNN classification

To train the model the splitted training data set is fit into the KNeighborsClassifier. The splitted test set is then used to predict the labels.

The table 4 is the comparison between the test prediction and actual results shown by the metrics Macro F1, precision and accuracy.

Out of the 4 classifiers, it is evident that for the given data set, the Random Forest classifier with Count vectorizer produces a higher accuracy, macro average of F1-Score and precision.

[7] Source: https://medium.com/swlh/k-nearest-neighbor-ca2593d7a3c4

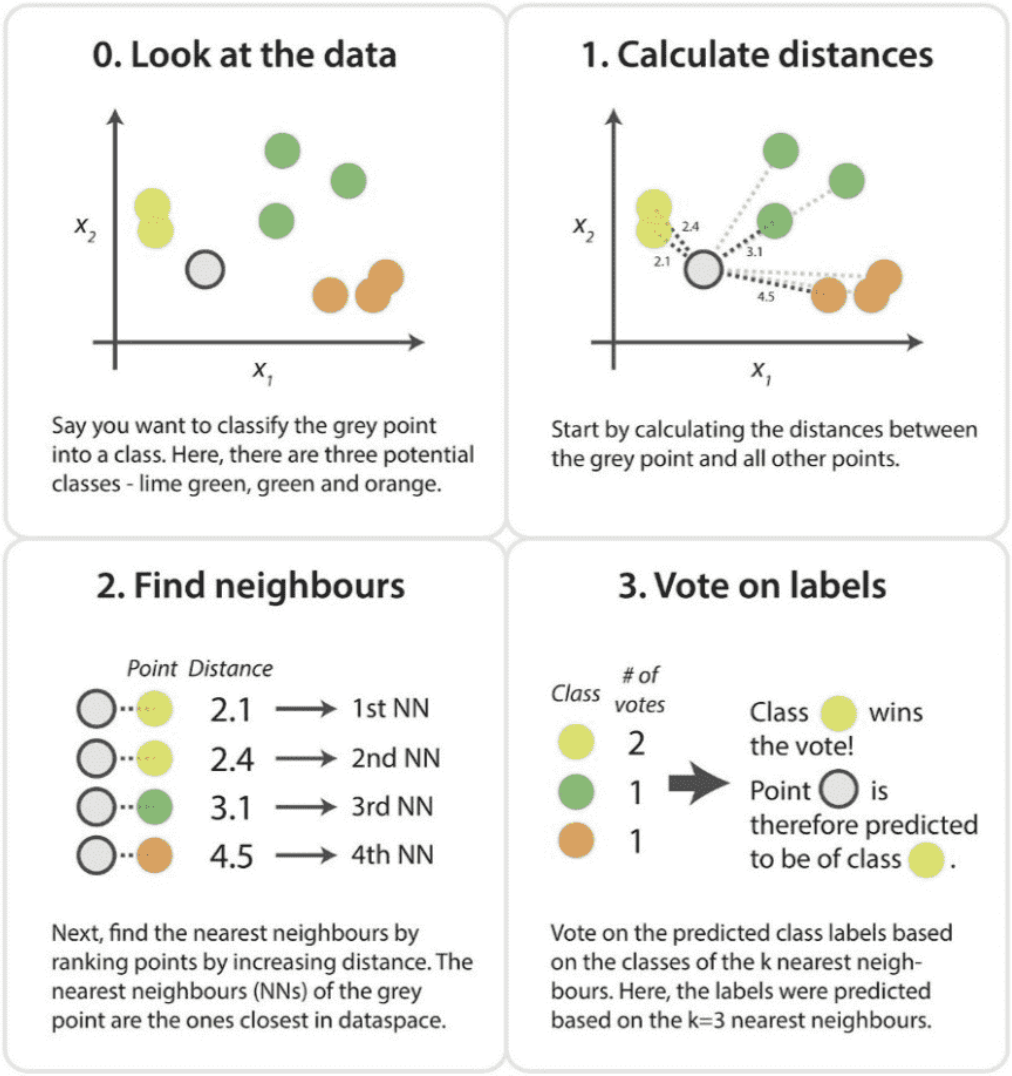

Figure 5: Steps in knn classification[7]

Hyperpameter	Function
n_neighbors	the tuning parameter/hyperparameter (k) which indicates the count of the nearest neighbors

5. Results

The entire training data set(without splitting) is used to fit the model and the Test data set is used to predict the labels from the fitted model to get the results. The results were copied to a csv file and submitted to the HASOC 2022 submission portal.

Table 4
Classification Report - A comparison of test prediction with actual labels of the subtasks

Subtasks	Macro F1	Macro Precision	Accuracy
Subtask A	0.87	0.88	0.88
Subtask B	0.55	0.58	0.64
Subtask C	0.38	0.37	0.65

Our team ssncse_nlp ranked 1 in the subtask A with a macro F1 and macro precision of 0.9745 and 0.9758 respectively using ***Random Forest*** classifier. In subtask B, our team ranked 3rd with a macro F1 and macro precision of 0.6958 and 0.7587 respectively using ***Logistic Regression*** classifier . In subtask C, our team ranked 2nd with a macro F1 and macro precision of 0.7929 and 0.7963 respectively using ***Random Forest*** classifier.

The overview of the HASOC 2022 task is given in [8] and [9]. The reference to the code is given in the foot note text as a git hub link[8].

The overall results(macro F1) are depicted in the table below:

Table 5
Results of HASOC[9]

Classifier Model	Subtask	Macro F1
RANDOM FOREST	A	**0.9745**
	B	0.6938
	C	**0.7929**
SUPPORT VECTOR MACHINE	A	0.8647
	B	0.5376
	C	0.3856
LOGISTIC REGRESSION	A	0.9587
	B	**0.6958**
	C	0.7867
K NEAREST NEIGHBORS	A	0.7316
	B	0.5018
	C	0.3986

***Inference*:** The above observations show that Random forest is a better choice. This is because Random Forest works well even for data points that are randomly distributed.

6. Ablation Study

Count vectorizer will consume much memory as it needs to store a vocabulary dictionary in memory. This problem can be reduced to some extent by the preprocessing of the data to remove the stopwords from the tweets. A few stopwords in Marathi language are as follows:

[8]Source: https://github.com/dikshu-02/HASOC2022-task3
[9]Source: https://hasocfire.github.io/submission/leaderboard.html

अधिक	अनेक	अशी	असलयाचे
असलेल्या	असा	असून	असे
आज	आणि	आता	आपल्या

Alternate is to use TF-IDF vectorizer. TF-IDF is better than count Vectorizers. This is because, TF-IDF not only focuses on the frequency of words present in the corpus, but also provides the importance of the words. We can then remove the words that are less important for analysis, hence making the model building less complex by reducing the input dimensions. [10] uses TF-IDF vectorizer for Social Network Hate speech Detection for Amharic language.

Another issue with these classifier models is that they don't take into account the context of a sentence. Hate speech is a very abstract and broad topic that is difficult to understand. The detection of hate speech depends on people's subjective understanding of what hate speech is. The model discussed in this paper counts the number of repeated words and finds a pattern while training and labels the data based on the count of specific words. This is misleading as the same words can be associated with negative sentences as well i.e. both "This is good" and "This is not good" counts equal numbers of "good", "this" and "is".
To ensure increased accuracy, pre-trained models for Marathi such as Roberta-base-mr can be used for sentiment analysis or context analysis. [11] have shown the usage of Marathi based transformer models and compares results of using a monolingual and multilingual model.

7. Conclusion

The need for scrutiny of comments online is becoming increasingly necessary to prevent cyber bullying and maintain the harmony of life. The models discussed in this paper are some of the ways to identify offensive tweets Once identified there can be many ways to prevent them(blocking the users, having a point-based system in Social media for reducing hate comments, giving warnings etc) which are not the scope of this paper. There is however further scope for improvement in the discussed models as seen in the ablation study.

References

[1] T. Turki, S. S. Roy, Novel hate speech detection using word cloud visualization and ensemble learning coupled with count vectorizer, Applied Sciences 12 (2022). URL: https://www.mdpi.com/2076-3417/12/13/6611. doi:10.3390/app12136611.

[2] M. A. Fauzi, A. Yuniarti, Ensemble method for indonesian twitter hate speech detection, Indonesian Journal of Electrical Engineering and Computer Science 11 (2018) 294–299.

[3] A. Anand, J. Golecha, B. Bharathi, B. Jayaraman, T. Mirnalinee, Machine learning based hate speech identification for english and indo-aryan languages (2021).
[4] D. Asogwa, C. Chukwuneke, N. Chigozie, G. Anigbogu, Hate speech classification using svm and naive bayes, 2022.
[5] M. K. A. Aljero, N. Dimililer, A novel stacked ensemble for hate speech recognition, Applied Sciences 11 (2021) 11684.
[6] A. Kalaivani, D. Thenmozhi, Multilingual hate speech and offensive language detection in english, hindi, and marathi languages (2021).
[7] M. Zampieri, T. Ranasinghe, M. Chaudhari, S. Gaikwad, P. Krishna, M. Nene, S. Paygude, Predicting the type and target of offensive social media posts in marathi, Social Network Analysis and Mining 12 (2022) 77. URL: https://doi.org/10.1007/s13278-022-00906-8. doi:10.1007/s13278-022-00906-8.
[8] Satapara, Shrey and Majumder, Prasenjit and Mandl, Thomas and Modha, Sandip and Madhu, Hiren and Ranasinghe, Tharindu and Zampieri, Marcos and North, Kai and Premasiri, Damith, Overview of the HASOC Subtrack at FIRE 2022: Hate Speech and Offensive Content Identification in English and Indo-Aryan Languages, in: FIRE 2022: Forum for Information Retrieval Evaluation, Virtual Event, 9th-13th December 2022, ACM, 2022.
[9] T. Ranasinghe, K. North, D. Premasiri, M. Zampieri, Overview of the HASOC subtrack at FIRE 2022: Offensive Language Identification in Marathi, in: Working Notes of FIRE 2022 - Forum for Information Retrieval Evaluation, CEUR, 2022.
[10] Z. Mossie, J.-H. Wang, Social network hate speech detection for amharic language, Computer Science & Information Technology (2018) 41–55.
[11] A. Velankar, H. Patil, R. Joshi, Mono vs multilingual bert for hate speech detection and text classification: A case study in marathi, arXiv preprint arXiv:2204.08669 (2022).