Pattern Recognition in Machine Learning

MAJOR PROJECT

PROJECT DESCRIPTION

Hate Speech Detection

The objective of this task is to build a machine learning model that can accurately detect hate speech. Hate speech is defined as any tweet with a racist or sexist sentiment associated with it. Given a dataset of tweets and their corresponding labels (0 for non-hate speech, 1 for hate speech), the task is to predict the labels for a test dataset of tweets.
The goal of this project is to develop a classifier that can accurately distinguish between tweets containing hate speech and those that do not. This model can be used to report tweets that could be harmful or offensive, making the online community safer and more welcoming for everyone.

PROJECT PIPELINE

1. Loading of the dataset and preprocessing:
    - Collect a dataset of tweets and their corresponding labels (0 for non-hate speech, 1 for hate speech).
    - Preprocess the data, including text cleaning (removing URLs, mentions, punctuation, and stop words), tokenisation, and stemming/lemmatization.

2. Feature engineering:
    - Transform the preprocessed text data into a numerical format that can be used as input to a machine learning model using the Tfidf vectoriser technique.

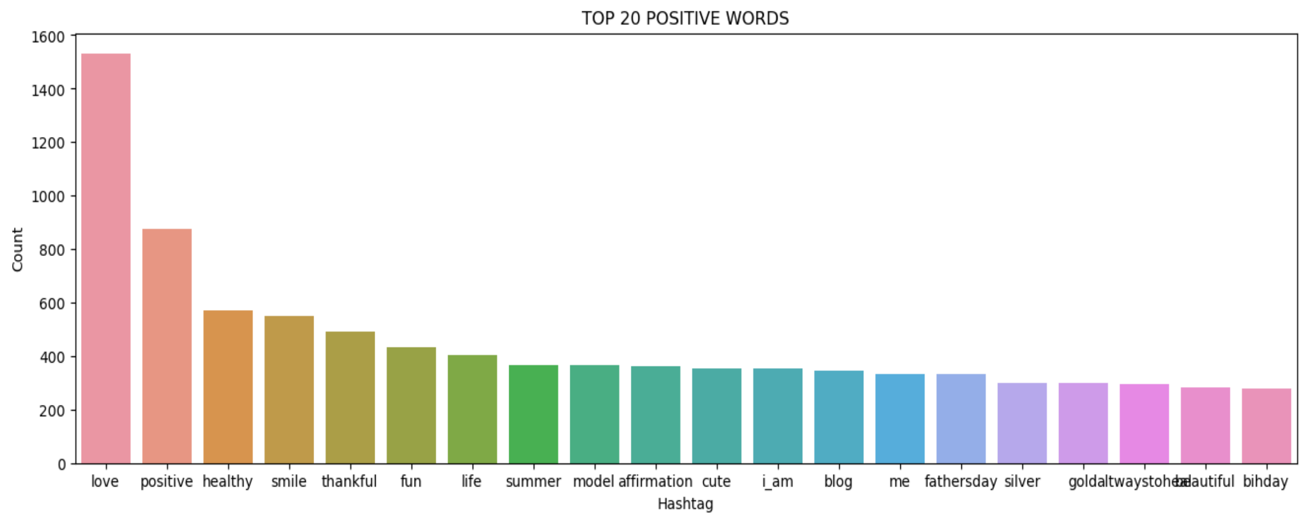
3. Model selection and training:
    - Split the preprocessed and transformed data into training and test sets.
    - Experiment with different models, such as random forests, support vector machines (SVMs), MLP (Neural Network), and Naive Bayes classifiers.
    - Train the selected model on the training set and evaluate its performance on the test set.

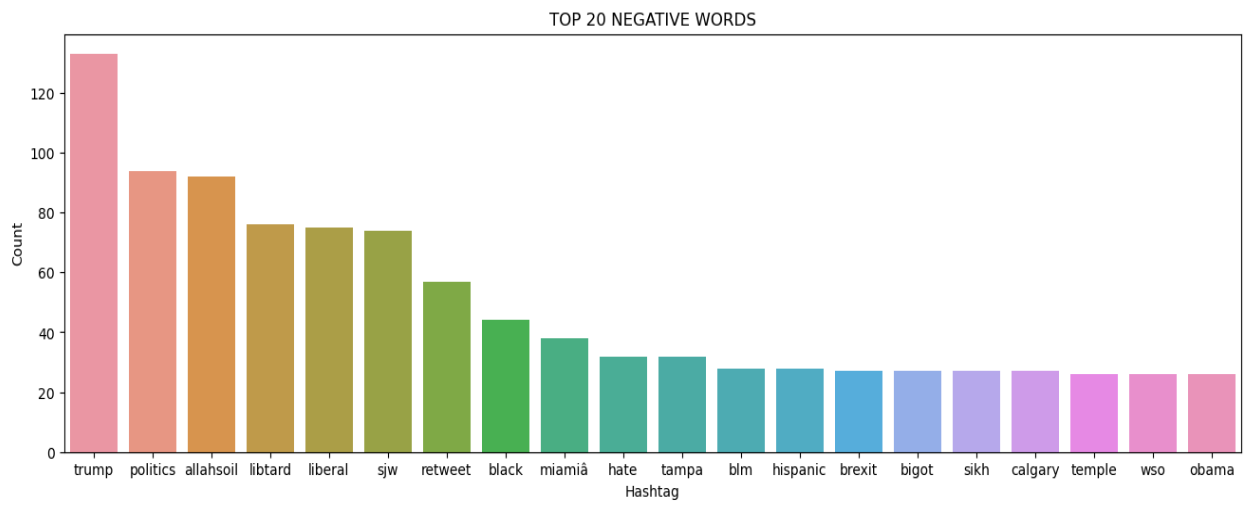
4. Model evaluation and testing:
    - Test the final model on a held-out test set to get an estimate of its performance on new, unseen data.
    - Evaluate the model's performance using metrics such as accuracy, precision, recall, and F1 score.

PRE-PROCESSING AND CLEANING OF THE DATASET

- Convert all text to lowercase: This ensures that words with different capitalization but the same meaning are treated the same.
- Remove URLs and mentions: URLs and mentions (i.e. references to other Twitter users) don't usually provide useful information for hate speech detection, so they can be removed.
- Remove punctuation: Punctuation marks such as commas, periods, and exclamation points are typically not informative for hate speech detection, so they can be removed.
- Remove stop words: Stop words are common words such as "the," "a," and "an" that are typically removed because they occur frequently but often do not carry much meaning.
- Tokenize the text: Tokenization involves breaking up the text into individual words, which can then be processed separately.
- Remove non-alphabetic characters: Removing non-alphabetic characters such as numbers and symbols can help reduce noise in the data.
- Stemming or lemmatization: Stemming and lemmatization are techniques for reducing words to their base form (e.g., "running" to "run") to reduce the number of unique words in the data and make it easier to process.

In the plots shown below , we have shown the list to top 20 positive and negative words in the given dataset





VECTORIZATION OF THE DATASET

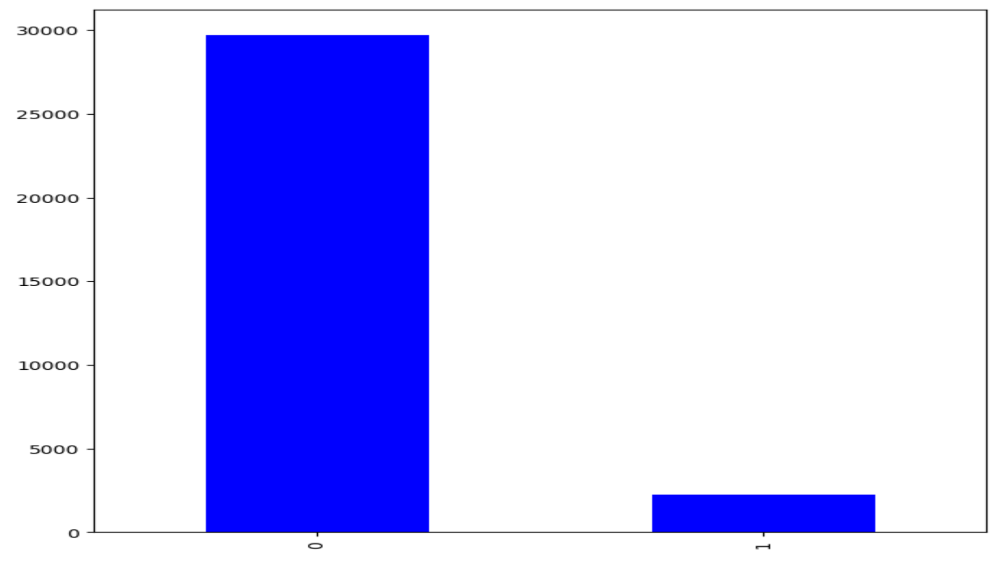
There are several types of vectorizer functions that can be used to convert text data into a numerical format that can be used as input to a machine-learning model. Here are a few examples:

- CountVectorizer
- TF-IDF Vectorizer
- HashingVectorizer
- Word2Vec

For hate speech detection, the **TF-IDF** vectorizer is a suitable choice because it is able to capture the importance of each word in a document based on its frequency in the document and across the entire corpus, which is important for identifying discriminatory language that may be unique to a particular document or common across many documents. Additionally, the sparsity of the resulting feature matrix can help reduce overfitting in the model and improve the interpretability of the results. Overall, the TF-IDF vectorizer is a popular choice for text classification tasks due to its effectiveness and simplicity.

OVERSAMPLING

Our dataset is a highly imbalanced as it contains most of the data having one types of class label and very less of the other type . This would affect the performed of our model trained on this dataset .
This high imbalance is evident from the bar plot shown below:



Thus there is a need for **oversampling** the dataset in order to train our model with equal instances of both class labels.

SMOTE :

Oversampling using techniques such as Synthetic Minority Over-sampling Technique (SMOTE) is a way to mitigate the effects of class imbalance by creating new synthetic samples for the minority class. SMOTE works by randomly selecting a minority class instance and introducing synthetic examples along the line segments connecting the k-nearest neighbors of the selected instance. The synthetic samples are created by interpolating between existing samples in the minority class, resulting in a larger and more balanced dataset.

In the context of our dataset, oversampling using SMOTE is necessary because it helps to balance the dataset and improve the performance of the classifier on the minority class (i.e. hate speech). Without oversampling, the classifier may not learn to identify hate speech effectively, and may instead focus on the majority class (i.e. non-hate speech) due to the imbalance in the dataset. By oversampling the minority class, the classifier is exposed to more instances of hate speech, which can help it learn more effectively and make more accurate predictions on new data.

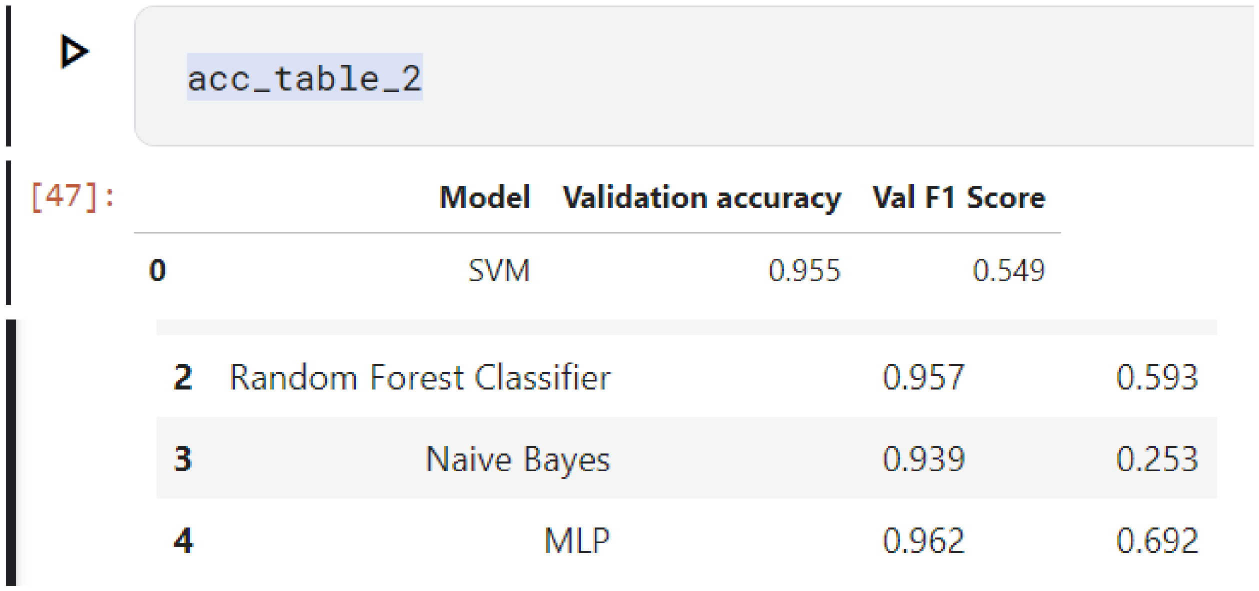
TRAINING MODELS ON THE DATASET

We will be using four models.

- SVM
- MLP
- Random Forest Classifier
- Naive Bayes Classifier

WITHOUT OVERSAMPLING

Here results are shown for different models which were trained without oversampling



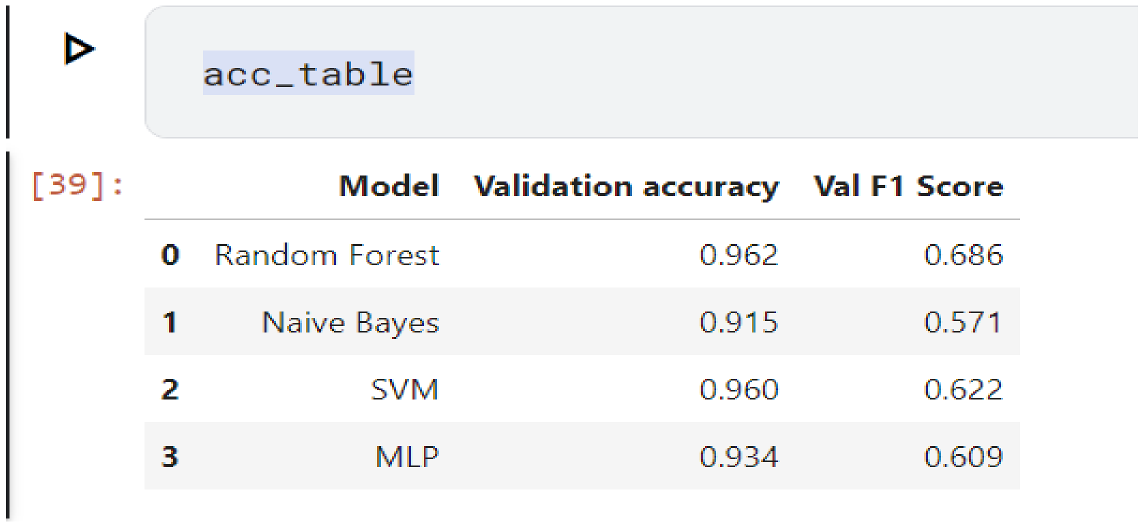
```
acc_table_2
```

[47]:

	Model	Validation accuracy	Val F1 Score
0	SVM	0.955	0.549
2	Random Forest Classifier	0.957	0.593
3	Naive Bayes	0.939	0.253
4	MLP	0.962	0.692

We see the values for f1-scores are coming out to be relatively low .
We try to oversample the data to train the model better .

WITH OVERSAMPLING



```
acc_table
```

[39]:

	Model	Validation accuracy	Val F1 Score
0	Random Forest	0.962	0.686
1	Naive Bayes	0.915	0.571
2	SVM	0.960	0.622
3	MLP	0.934	0.609

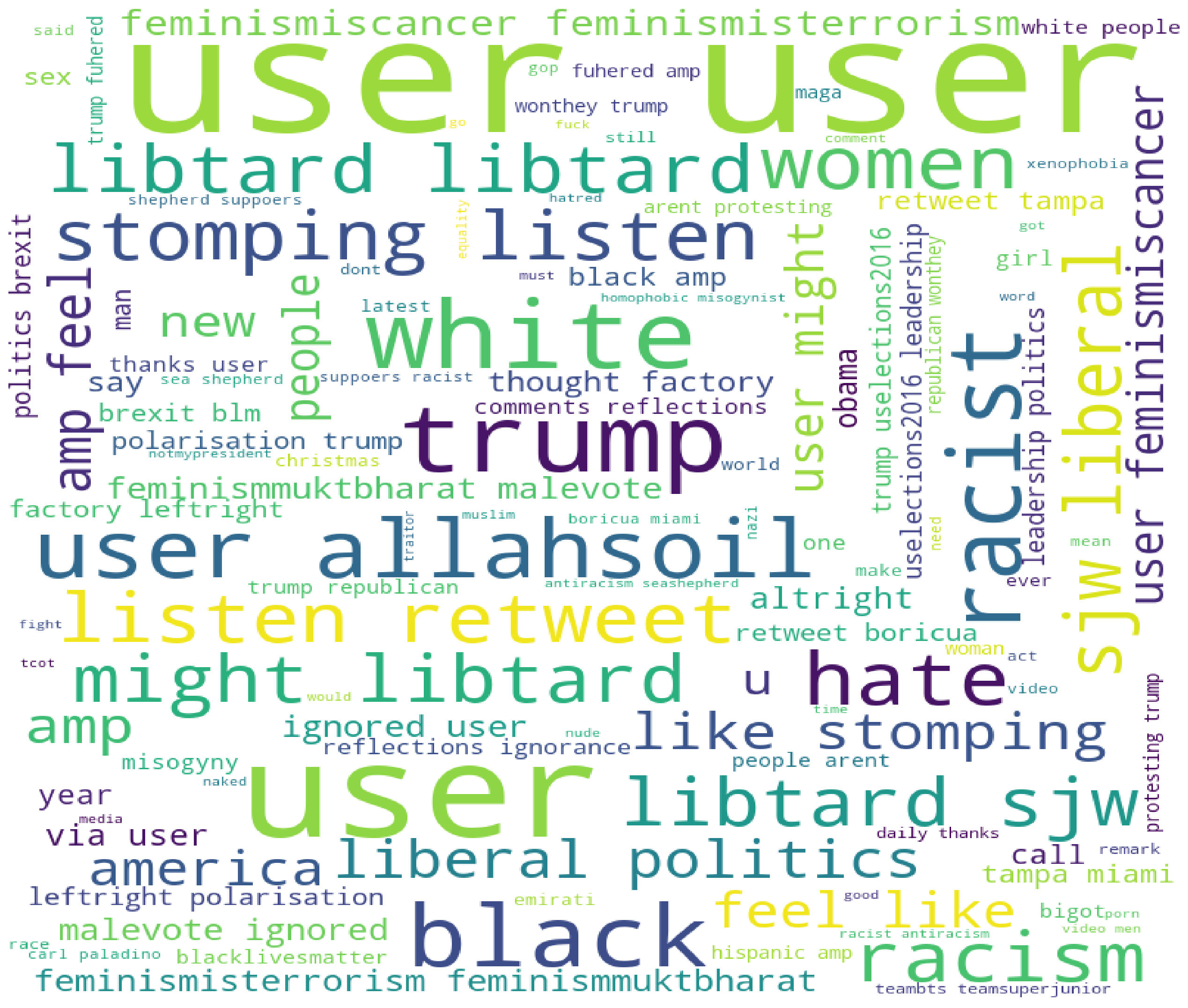
From the table shown (after oversampling) we can see the drastic increase in f1-scores for our models . (a high rise in case of naive bayes!) .
When we oversample the minority class, the F1 score tends to increase because the classifier is better able to identify instances of the minority class, resulting in fewer false negatives and a higher recall score. Additionally, oversampling can also lead to improved precision, as the classifier is less likely to incorrectly label instances of the minority class as the majority class.

Overall, oversampling can lead to significant improvements in the performance of classifiers for imbalanced datasets, particularly for problems like hate speech detection where the minority class is important and requires accurate identification.

RESULTS OF OUR BEST MODELS ON THE TEST DATASET:

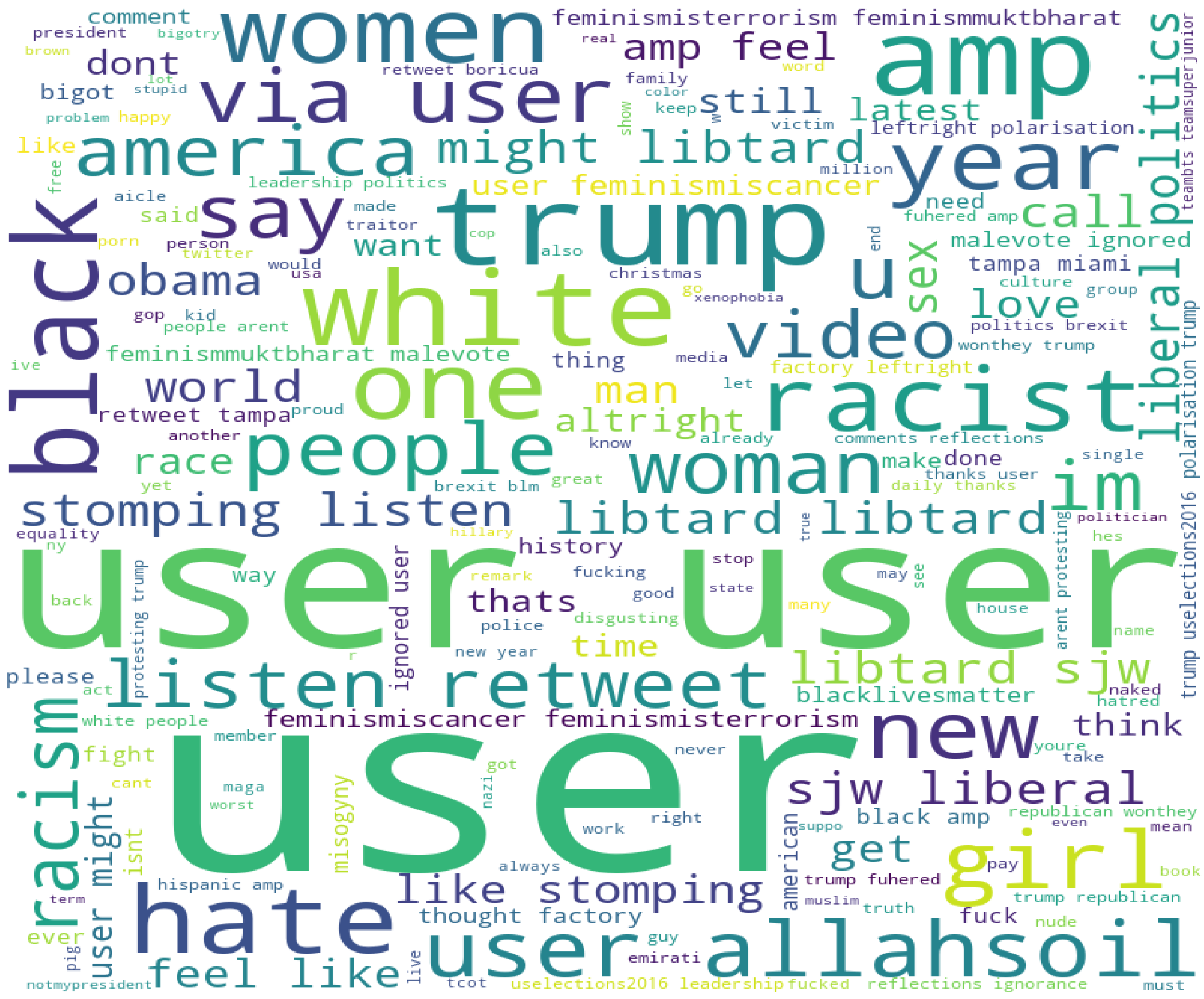
SVM :

Ou data segregated the hate words form the test dataset . The wordcloud feature showing the frequency of hate words are shown below:



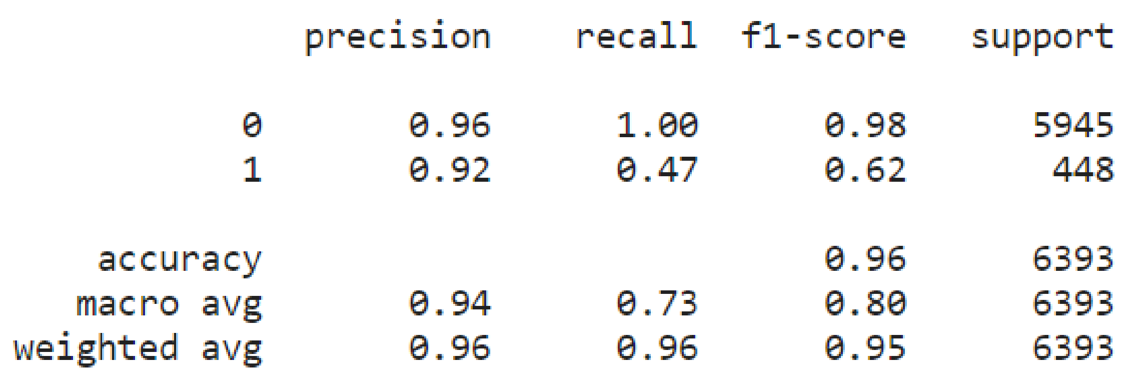
MLP:

In this model also , we segregated the hate words from the test dataset . Wordcloud is shown below:



Conclusion:

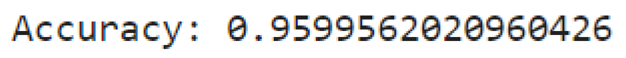
After training and testing the bunch of models and finding their accuracy metrics we can conclude that the SVM performed better than the rest of the models after it was trained oversampled data . The Classification report of SVM came oout to be :



```
              precision    recall  f1-score   support

           0       0.96      1.00      0.98      5945
           1       0.92      0.47      0.62       448

    accuracy                           0.96      6393
   macro avg       0.94      0.73      0.80      6393
weighted avg       0.96      0.96      0.95      6393
```

Accuracy score : 

```
Accuracy: 0.9599562020960426
```

CONTRIBUTIONS:

Rohan Raj (B21EE057) : Wrote the codes for the model training and finding the accuracy metric and used different techniques to analyze and enhance our model. Helped in preparing reports explaining our model better to the viewer .

Shah Keval Bhushanbhai (B21CS069) : Came up with different ideas to enhance our model. Pre-processed the dataset to visualize it better for the viewer. Gave ideas for preparing the reports.

Sajal Jain (B21EE058) : Found information about how we can tackle the problem statement and provided different resources that our team referred to while tackling the problem statement. Also helped with coding part .

Overall, our project was a team effort, with each team member contributing their own unique skills and expertise. By working together, we were able to build an effective and accurate model for hate speech detection and communicate our results clearly and effectively to our stakeholders.

Thank You