MALIGNATE COMMENT PREDICTION

Submitted by:

Karthik MSR

ACKNOWLEDGMENT

Following websites are used in completion of the project.

- https://stackoverflow.com
- https://scikit-learn.org
- https://machinelearningmastery.com
- https://www.geeksforgeeks.org
- https://pandas.pydata.org
- https://www.machinelearningplus.com

INTRODUCTION

❖ Business Problem Framing

- Our goal is to build a prototype of online hate and abuse comment classifier which can used to classify hate and offensive comments so that it can be controlled and restricted from spreading hatred and cyberbullying.

➢ Hate comments are divided into 6 categories which are:

- **Malignant:** It is the Label column, which includes values 0 and 1, denoting if the comment is malignant or not.
- **Highly Malignant:** It denotes comments that are highly malignant and hurtful.
- **Rude:** It denotes comments that are very rude and offensive.
- **Threat:** It contains indication of the comments that are giving any threat to someone.
- **Abuse:** It is for comments that are abusive in nature.
- **Loathe:** It describes the comments which are hateful and loathing in nature.

- Model will help in predicting the category of the comment and the platform can take the action accordingly.

❖ Conceptual Background of the Domain Problem

- The proliferation of social media enables people to express their opinions widely online. However, at the same time, this has resulted in the emergence of conflict and hate, making online environments uninviting for users. Although researchers have found that hate is a problem across multiple platforms, there is a lack of models for online hate detection, which we are trying to built a pattern out of the existing data.
- Online hate, described as abusive language, aggression, cyberbullying, hatefulness and many others has been identified as a major threat on online social media platforms. Social media platforms are the most prominent grounds for such toxic behaviour.
- There has been a remarkable increase in the cases of cyberbullying and trolls on various social media platforms. Many celebrities and influences are facing backlashes from people and have to come across hateful and offensive comments. This can take a toll on anyone and affect them mentally leading to depression, mental illness, self-hatred and suicidal thoughts.
- Internet comments are bastions of hatred and vitriol. While online anonymity has provided a new outlet for aggression and hate speech, machine learning can be used to fight it. The problem we sought to solve was the tagging of internet comments that are aggressive towards other users. This means that insults to third parties such as celebrities will be tagged as unoffensive, but “u are an idiot” is clearly offensive.
- Thus, the concept of cyberbullying and special departments to handle such cases are introduced across the countries. These prototypes can be helpful for both government bodies and platforms to control and take actions accordingly.

❖ Review of Literature

- Online hate, described as abusive language, aggression, cyberbullying, hatefulness and many others has been identified as a major threat on online social media platforms. Social media platforms are the most prominent grounds for such toxic behaviour.
- Hate comments also have a negative effect on receiver's mental health which is not a right sign of growth for the social media, there is a difference between criticism which help in the growth of that person and hate which effect the person mentally.
- Thus, there is a need to find and control such activities over the internet.

❖ Motivation for the Problem Undertaken

- Social media has become an integral part of our daily life where people care judged with just an image or video which they choose to show only the life they wish to show on multiple platforms.
- Online hate, described as abusive language, aggression, cyberbullying, hatefulness and many others has been identified as a major threat on online social media platforms. Have been growing greatly with the greater use and availability to multiple classes of people, where different classes of people have different opinions on same content leading to hate comments on them.
- Following will lead to mental health issues of the receiver is the motivation to undertake the project the create a prototype to detect and handle the hate in social media

Analytical Problem Framing

❖ Mathematical/ Analytical Modelling of the Problem

Following problem is a Multi class classification problem with six categorical

variables Malignant, Highly malignant, Rude, Threat, Abuse, Loathe which are denoted by '1' and the clean comments are denoted by '0'.

A comment can have more than one type of category for example it can malignant and rude at the same time.

• The algorithms used in predicting the comment categories are

- Keras - LSTM
- LSTM – Bidirectional
- GlobalMaxPool1D
- OneVsRestClassifier with LogisticRegression

- Techniques used to convert words into vectors or tokens are

  ➢ Vectors
  - TfidfVectorizer
  - TfidfTransformer
  - CountVectorizer

  ➢ Word Tokens
  - Tokenizer

• Removal Techniques

We have removed the comments where the words are less then 3 as they can't mean anything in general

Algorithm Justification

- LSTM is the one of the best models used in NLP for predictions
- Bidirectional LSTM is used to check if can improve the performance of the LSTM by using it in both forward and backward propagation.
- Word tokenizer word well for our model so we did not go for word2vec or glovec

❖ Data Sources and their formats

- Following Data contains 8 rows which include id, Comment text, and the six categorical tags for the comments.
- The data is not evenly distributed for the number of clean comments and hate comment with tags.
- The data includes train and test data set which has 159571 and 153000 columns respectively where the train data has

- Total comments = 159274
- Total clean comments = 143084
- Total Hate Comments = 16193
- Total tags = 35027
- Total number of Words = 158647 words
- Maximum length of the sentence = 1250 words

- The data has been collected from various social media platforms.

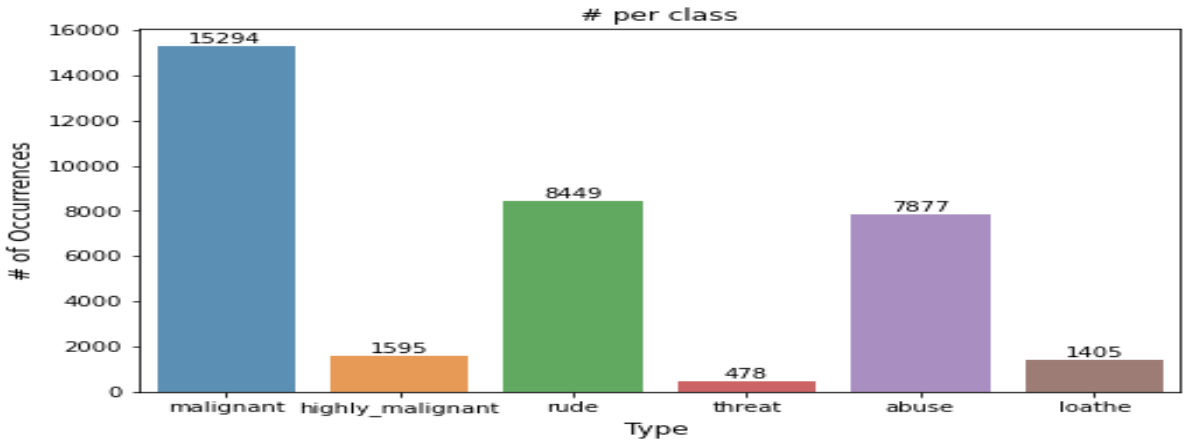
➢ Data Snapshot

```
df=pd.read_excel("malignate_train.xlsx")
df
```

	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...
159566	ffe987279560d7ff	":::::And for the second time of asking, when ...	0	0	0	0	0	0
159567	ffea4adeee384e90	You should be ashamed of yourself \n\nThat is ...	0	0	0	0	0	0
159568	ffee36eab5c267c9	Spitzer \n\nUmm, theres no actual article for ...	0	0	0	0	0	0
159569	fff125370e4aaaf3	And it looks like it was actually you who put ...	0	0	0	0	0	0
159570	fff46fc426af1f9a	"\nAnd ... I really don't think you understand...	0	0	0	0	0	0

159571 rows × 8 columns

➢ Following is the distribution of the Hate Comments



❖ Data Pre-processing Done

Following are the methods in data pre-processing

The comment can contain various types of noises which will disturb or decrease the affiances of the model in predicting so such noises are removed from the comments

- Removal of Html Tags
- Removal of Stop words
- Special Characters
- All the other Language reviews by holding only letters in English A-Z and a-z
- Converting all the words to lower case
- Removal of emoji's
- Removing all the punctuations
- Lemmatization of the Words to decrease the inconsistency where by decreasing the Features.

➢ Converting the words into tokens in Method 1

- We have converted the words into tokens using keras.preprocessing.text – Tokenizer. The word limit has been set to 70,000 words and Sentence limit is set to 500 words this is to decrease the load on the system.
- The following converted data has been used as the input for LSTM, Bidirectional LSTM.

➢ Converting the words into vectors in Method 2

- We have then converted the Sentence into separate values and values into vectors using Count Vectorizer and further buy labelling the vectors by using TFIDF Transformer.
- The total number of features that are set is 70,000.
- The following data is used as an input for Logistic Regression with OnevsRestClassifier.

❖ Data Inputs- Logic- Output Relationships

Method 1:

- Input of for the method 1 where we used LSTM as the Algorithm is Tokenised Word
- We have used texts_to_sequences by fitting Tokenizer from Comment text and transformed the text to sequence of numbers with 70,000 unique words and 500 as the maximum length on the sentence and padded the sequence using pad_sequence.
- Input layer has 500 inputs with combination of 70,000 tokenised words and output layer has 6 possible out puts with 0 means that comment doesn't belongs to the following category and 1 means the comment belongs to the category.
- Presence of combinations of tokens in a particular sequence will lead to the output prediction whether the comment belongs to a category or not.

Method 2:

- We converted words into vectors using TFIDF vectorizer and made the vectors as the input for the OnevsRestClassifier with Logistic regressor.
- Which follows a comparison approach where a category of the comment is compared to other categories and create a model on the set of features that are unique in determining the category.

❖ State the set of assumptions related to the problem under consideration

Following assumptions are made for the convenience in problem solving

- Total number of Words = 158647 words
- Maximum Length of the words = 1250
- The total number of word and Maximum length of the sentence has been assumed to be 70,000 words and 500 words as the max length.
- Threshold value to convert the output has been considered to be 0.5
- <0.5 = 0 and >=0.5 = 1

❖ Hardware and Software Requirements and Tools Used

Laptop- Windows 10 – i5 processor – 8 GB RAM.

Anaconda – Python3.

NumPy, Pandas, Seaborn, Pipeline

Algorithms – Logistic Regessor, LSTM, BiDirectional LSTM

Optimizer - Rmsprop, Adam

Loss Function - categorical_crossentropy, binary_crossentropy

Model Selection - train_test_split

Metrics – accuracy score, AUC

Model/s Development and Evaluation

❖ Identification of possible problem-solving approaches (methods)

➢ Following neural network algorithms have been used

- Keras.Preprocessing – To Preocess the text or words into tokens using Tokenizer

Embedding Layer

- Turns positive integers (indexes) into dense vectors of fixed size.This layer can only be used as the first layer in a model.
- LSTM – One directional for adjusting the weights in forward propagation
- Bidirectional LSTM – bidirectional to adjust the weights in both directions
- GlobalMaxPool1D - Global max pooling operation for 1D temporal data. Down samples the input representation by taking the maximum value over the time dimension.
- Conversion function – Sigmoid – To convert all the functional inputs to 0's and 1's according to their values

➢ Optimizer

- Adam optimization is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments.
- Rmsprop - Maintain a moving (discounted) average of the square of gradients
  Divide the gradient by the root of this average
- Loss Function - categorical_crossentropy, binary_crossentropy

➢ Following is used in Vectorizing data.

- CountVectorizer and TFIDF Transformer
- texts_to_sequences
- pad_sequences

➢ Following Metrices are used to analyse the best algorithm which fits the data.

- Metrics - accuracy_score, AUC ROC Score, Loss Function

➢ Following are used to split the data and select the best possible parameters for algorithms.

- Spilt Selection - train_test_split

➢ Following modules are used to plot and analyse the data.

- Plotting Modules – Seaborn, Matplotlib

❖ Testing of Identified Approaches (Algorithms)

- LSTM – One directional for adjusting the weights in forward propagation
- Bidirectional LSTM – bidirectional to adjust the weights in both directions
- GlobalMaxPool1D - Global max pooling operation for 1D temporal data. Down samples the input representation by taking the maximum value over the time dimension.

- OneVsRestClassifier - Also known as one-vs-all, this strategy consists in fitting one classifier per class. For each classifier, the class is fitted against all the other classes.
- logistic regressions are only binary classifiers, meaning they cannot handle target vectors with more than two classes. However, there are clever extensions to logistic regression to do just that. In one-vs-rest logistic regression (OVR) a separate model is trained for each class predicted whether an observation is that class or not (thus making it a binary classification problem).

➢ Testing Metrices

- **accuracy_score** - In multilabel classification, this function computes subset accuracy: the set of labels predicted for a sample must exactly match the corresponding set of labels in y_true
- **AUC - ROC** is a performance measurement for the classification problems at various threshold settings. ROC is a probability curve and AUC represents the degree or measure of separability. It tells how much the model is capable of distinguishing between classes.
- **Loss** - The purpose of loss functions is to compute the quantity that a model should seek to minimize during training.
- **Precision** - Computes the precision of the predictions with respect to the labels.
- The metric creates two local variables, true_positives and false_positives that are used to compute the precision. This value is ultimately returned as precision, an idempotent operation that simply divides true_positives by the sum of true_positives and false_positives.
- **Re-call** - Computes the recall of the predictions with respect to the labels.
- This metric creates two local variables, true_positives and false_negatives, that are used to compute the recall. This value is ultimately returned as recall, an idempotent operation that simply divides true_positives by the sum of true_positives and false_negatives.

➢ Loss Functions

Binary crossentropy

- Computes the cross-entropy loss between true labels and predicted labels. Use this cross-entropy loss for binary (0 or 1) classification applications.

❖ Run and evaluate selected models

Following is the First Model built

- Algorithm – LSTM
- Optimizer - Rmsprop
- Loss - Categorical_crossentropy
- Metrics – Accuracy and AUC

```
## Creating model
embedding_vector_features=20
model=Sequential()
model.add(Embedding(70000,embedding_vector_features,input_length=500))
model.add(LSTM(100))
model.add(Dense(num_classes,activation='sigmoid'))
model.compile(loss='categorical_crossentropy',optimizer='rmsprop',metrics=['accuracy','AUC'])
print(model.summary())
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 500, 20)           1400000
_________________________________________________________________
lstm (LSTM)                  (None, 100)               48400
_________________________________________________________________
dense (Dense)                (None, 6)                 606
=================================================================
Total params: 1,449,006
Trainable params: 1,449,006
Non-trainable params: 0
_________________________________________________________________
None
```

Results

```
#Fitting the model to the data
hist = model.fit(X_train, y_train, batch_size=128, epochs=2, validation_data=(X_val, y_val))
```

```
Epoch 1/2
834/834 [==============================] - 885s 1s/step - loss: 0.3529 - accuracy: 0.9874 - auc: 0.5067 - val_loss: 0.3336 - va
l_accuracy: 0.9940 - val_auc: 0.5034
Epoch 2/2
834/834 [==============================] - 809s 970ms/step - loss: 0.3302 - accuracy: 0.9940 - auc: 0.5024 - val_loss: 0.3307 -
val_accuracy: 0.9938 - val_auc: 0.5031
```

Second Model

- Algorithm – Bidirectional LSTM
- Optimizer - SDG
- Loss – binary_crossentropy
- Metrics – Accuracy and AUC

Bidirectional LSTM

```
model1 = keras.models.Sequential([keras.layers.Input(shape = (500)),
                                  keras.layers.Embedding(70000, 64),
                                  keras.layers.Bidirectional(LSTM(56)),
                                  keras.layers.Dense(128, activation = 'relu'),
                                  keras.layers.Dropout(0.1),
                                  keras.layers.Dense(28, activation = 'relu'),
                                  keras.layers.Dropout(0.1),
                                  keras.layers.Dense(num_classes, activation = 'sigmoid')])
model1.compile(loss='binary_crossentropy',optimizer='SGD',metrics=['accuracy','AUC'])
print(model1.summary())
```

```
WARNING:tensorflow:Please add `keras.layers.InputLayer` instead of `keras.Input` to Sequential model
to be used by Functional model.
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_3 (Embedding)      (None, 500, 64)           4480000
_________________________________________________________________
bidirectional_2 (Bidirection (None, 112)               54208
_________________________________________________________________
dense_8 (Dense)              (None, 128)               14464
_________________________________________________________________
dropout_6 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_9 (Dense)              (None, 28)                3612
_________________________________________________________________
dropout_7 (Dropout)          (None, 28)                0
```

Results

```
model1.fit(X_train, y_train, batch_size = 128, epochs = 2, validation_data=(X_val, y_val))
```

```
Epoch 1/2
834/834 [==============================] - 2345s 3s/step - loss: 0.5153 - accuracy: 0.1350 - auc: 0.5156 - val_loss: 0.1490 - v
al_accuracy: 0.9940 - val_auc: 0.6900
Epoch 2/2
834/834 [==============================] - 2366s 3s/step - loss: 0.1547 - accuracy: 0.8166 - auc: 0.6796 - val_loss: 0.1421 - v
al_accuracy: 0.9940 - val_auc: 0.7479
```

OneVsRest Classification with Logistic Regression

```
# Using pipeline for applying logistic regression and one vs rest classifier
LogReg_pipeline = Pipeline([
                ('clf', OneVsRestClassifier(LogisticRegression(solver='sag'), n_jobs=-1))])
for category in categories:
    print('**Processing {} comments...**'.format(category))

    # Training logistic regression model on train data
    LogReg_pipeline.fit(x_train, train[category])

    # calculating test accuracy
    prediction_train = LogReg_pipeline.predict(x_train)
    print('Train accuracy is {}'.format(accuracy_score(train[category], prediction_train)))
    print('Train AUC is {}'.format(metrics.roc_auc_score(train[category], prediction_train)))


    # calculating test accuracy
    prediction = LogReg_pipeline.predict(x_test)
    print('Test accuracy is {}'.format(accuracy_score(test[category], prediction)))
    print('Test AUC is {}'.format(metrics.roc_auc_score(test[category], prediction)))
    print("\n")

```

Results

```
**Processing malignant comments...**
Train accuracy is 0.961324232449256
Train AUC is 0.812330889072716
Test accuracy is 0.9541887282087772
Test AUC is 0.7801472086205209


**Processing highly_malignant comments...**
Train accuracy is 0.9909947888170345
Train AUC is 0.6193022924349617
Test accuracy is 0.9899755143042505
Test AUC is 0.5755698133309619


**Processing rude comments...**
Train accuracy is 0.9797023975029374
Train AUC is 0.8267521654223021
Test accuracy is 0.9752840968545299
Test AUC is 0.7908786955543148


**Processing threat comments...**
Train accuracy is 0.9973361078472702
Train AUC is 0.5496579573967428
Test accuracy is 0.9968817361823242
Test AUC is 0.5261018005074519


**Processing abuse comments...**
Train accuracy is 0.9731996304634455
Train AUC is 0.7694420276336419
Test accuracy is 0.9693405604503694
Test AUC is 0.730895182863145


**Processing loathe comments...**
Train accuracy is 0.9924388515664941
Train AUC is 0.6050712462658399
Test accuracy is 0.9920683088127576
Test AUC is 0.5713275122442902
```

Final Model

```
tf.keras.backend.clear_session()
input_layer = Input(shape = (500, ))
x = Embedding(70000, 200)(input_layer)
x = LSTM(60, return_sequences=True)(x)
x = GlobalMaxPool1D()(x)
x = Dropout(0.1)(x)
x = Dense(50, activation="relu")(x)
x = Dropout(0.1)(x)
output_layer = Dense(6, activation="sigmoid")(x)
model3 = Model(inputs=input_layer, outputs=output_layer)
model3.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_3 (InputLayer)         [(None, 500)]             0
_________________________________________________________________
embedding (Embedding)        (None, 500, 200)          14000000
_________________________________________________________________
lstm (LSTM)                  (None, 500, 60)           62640
_________________________________________________________________
global_max_pooling1d (Global (None, 60)                0
_________________________________________________________________
dropout (Dropout)            (None, 60)                0
_________________________________________________________________
dense (Dense)                (None, 50)                3050
_________________________________________________________________
dropout_1 (Dropout)          (None, 50)                0
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 306
=================================================================
```

Results

```
model4.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy','AUC','Precision','Recall'])
model4.fit(train_seq, train_labels, batch_size=128, validation_split=0.25, epochs = 2)
```

```
Epoch 1/2
934/934 [==============================] - 1406s 1s/step - loss: 0.1752 - accuracy: 0.6685 - auc: 0.7557 - precision: 0.3439 -
recall: 0.1262 - val_loss: 0.0525 - val_accuracy: 0.9941 - val_auc: 0.9778 - val_precision: 0.7793 - val_recall: 0.6645
Epoch 2/2
934/934 [==============================] - 1307s 1s/step - loss: 0.0484 - accuracy: 0.9689 - auc: 0.9825 - precision: 0.8064 -
recall: 0.6701 - val_loss: 0.0498 - val_accuracy: 0.9940 - val_auc: 0.9798 - val_precision: 0.7921 - val_recall: 0.6727
```

❖ Key Metrics for success in solving problem under consideration

➢ Testing Metrices

- accuracy_score - In multilabel classification, this function computes subset accuracy: the set of labels predicted for a sample must exactly match the corresponding set of labels in y_true
- AUC - ROC curve is a performance measurement for the classification problems at various threshold settings. ROC is a probability curve and AUC represents the degree or measure of separability. It tells how much the model is capable of distinguishing between classes.
- Loss - The purpose of loss functions is to compute the quantity that a model should seek to minimize during training.

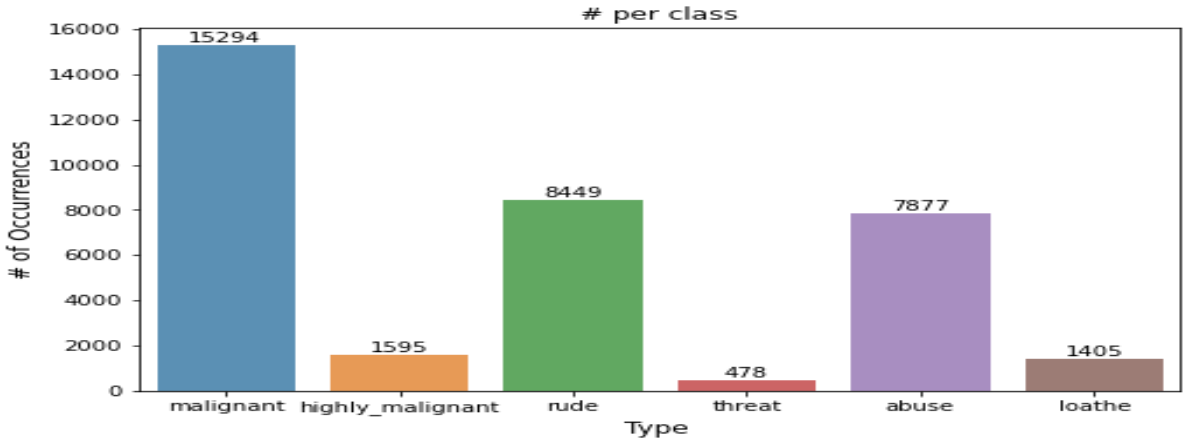
➢ Loss Functions

Binary crossentropy

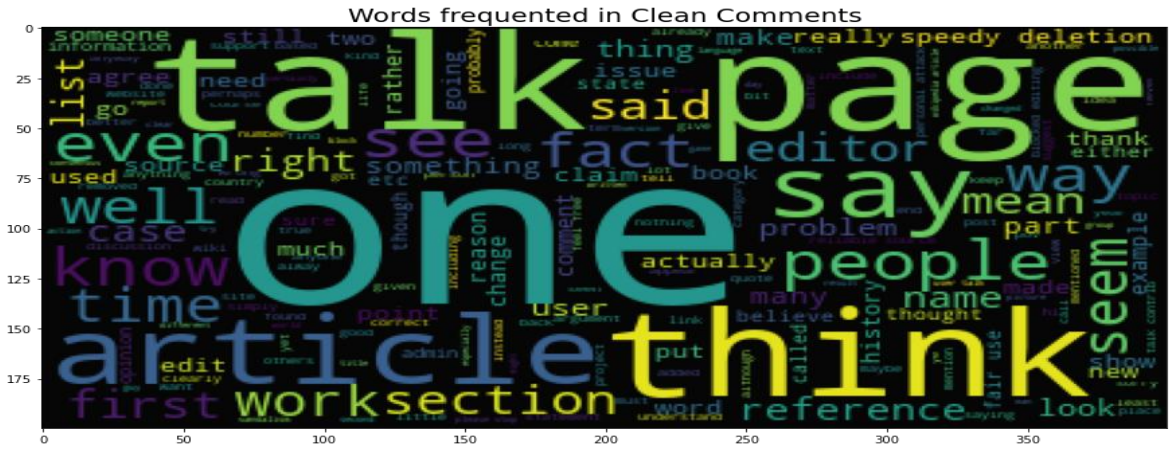
- Computes the cross-entropy loss between true labels and predicted labels. Use this cross-entropy loss for binary (0 or 1) classification applications.

❖ Visualizations

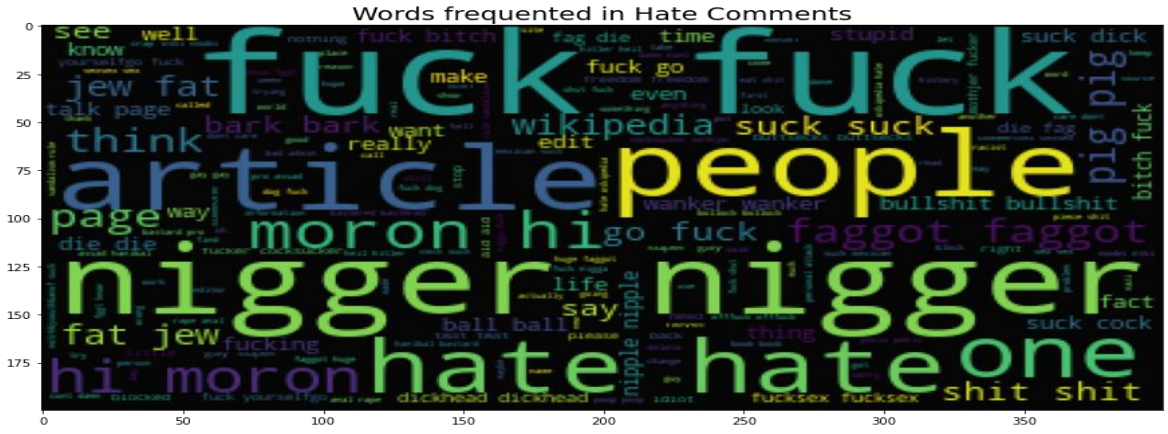
- ➢ Distribution of Hate Comments



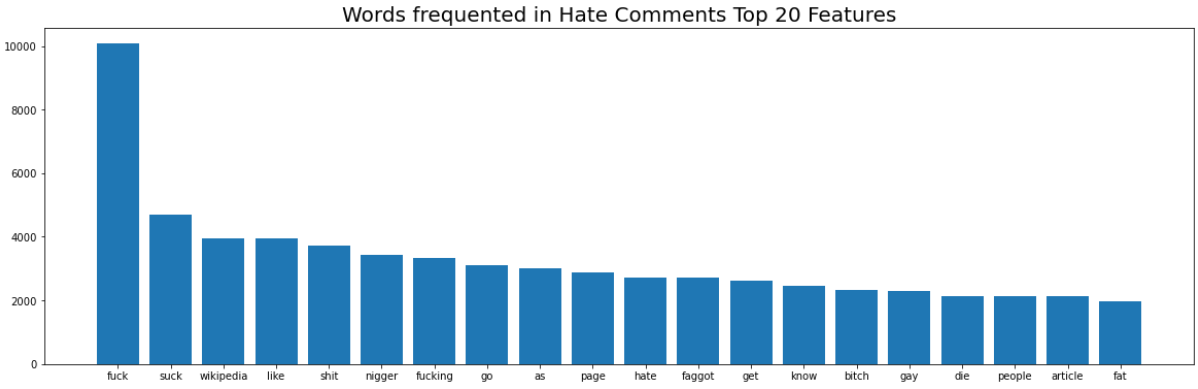
Following is the word cloud for Clean and Hate comments



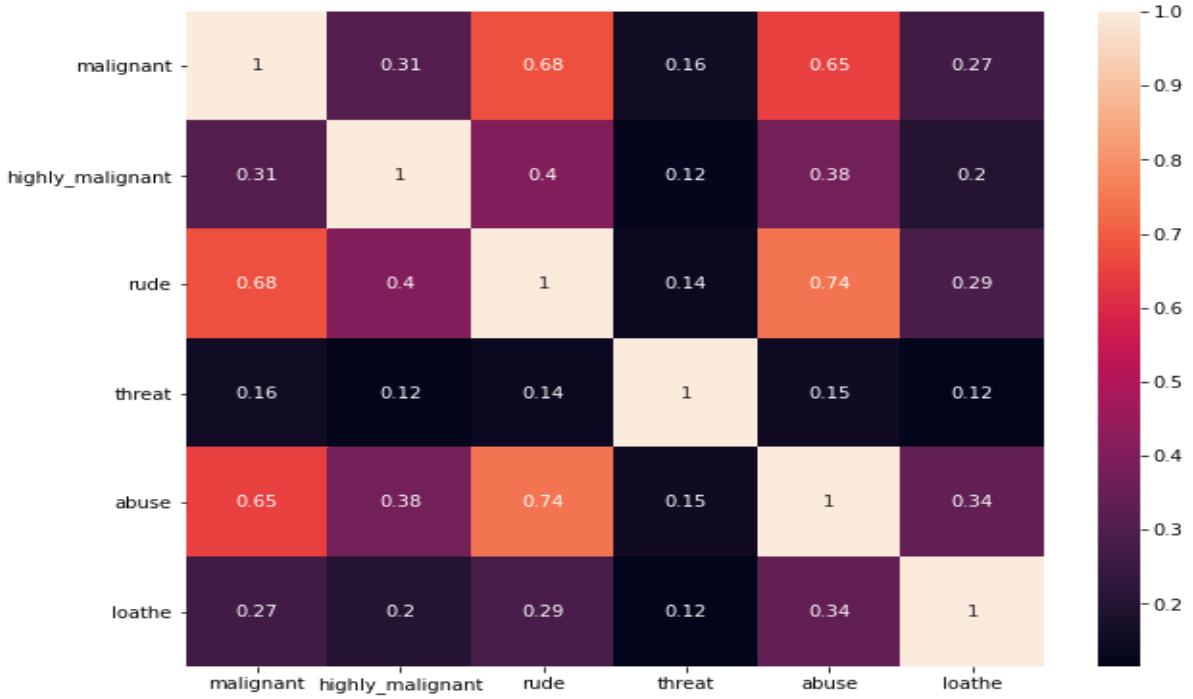
Hate Comments Word cloud



Word frequency of most common words in Hate comments



Correlation of the Hate Comments



❖ Interpretation of the Results

Hate comments in general follow a noticeable pattern which can be detected.

Following are the scores for Validation Dataset

Model	Parameters	Accuracy	Loss	AUC
LSTM	Categorical cross entropy, Rmprop	99.38%	0.3307	0.5031
Bidirectional LSTM	Binary Cross entropy, SDG	99.40%	0.1421	0.7479
Bidirectional LSTM GoldalMaxPool1D	Binary Cross entropy, Adam	99.41%	0.0498	0.9798
OnevsRest Logistic Regressor	solver= sag	97.84%	-	0.6613
XGB Classifier	use_label_encoder=False, eval_metric=logloss,objective =binary:logistic	98.05%		0.7077

- Final Model with best accuracy and Loss and AUC is Bidirectional LSTM with GlobalMaxPool1d
- Binary Entropy as the Loss Function
- Adam as the optimizer
- The Recall and Precision values for the final model is
- Recall – 67.27%
- Precision – 79.21%

For the Validation datasets

CONCLUSION

❖ Key Findings and Conclusions of the Study

We have been able to achieve the following scores using the final model.

Model	Parameters	Accuracy	Loss	AUC	Precision	Recall
Bidirectional LSTM GoldalMaxPool1D	Binary crossentropy, Adam	99.38%	0.0498	97.98%	79.21%	67.27%

- Hate Comments mostly follow a similar pattern which can be predicated and controlled with good accuracy.
- A Severe toxic comment is always toxic

❖ Learning Outcomes of the Study in respect of Data Science

- LSTM worked best with the NLP data where the multi class classification problem to output binary values determining the category.
- Adam Optimizer worked best with Keras LSTM Bidirectional
- Bidirectional LSTM with GlobalMaxPoll1d achieved the best scores with 70000 tokens and 500 max lengths of words as the inputs.
- Visualization helped in knowing the most generic words/features which play an important role in categorising the data
- Correlation helped in knowing the relation between the categories which effects which category.
- Data Cleaning helped in cleaning the noise from the data which greatly help the model in improving the precision and loss

❖ Limitations of this work and Scope for Future Work

There is a need in improving the recall and precision values

Recall Improvement –

Improving recall involves adding more accurately tagged text data to the tag in question. In this case, you are looking for the texts that should be in this tag but are not, or were incorrectly predicted (False Negatives). The best way to find these kinds of texts is to search for them using keywords.

Precision Improvement – Increasing the threshold will typically increase precision and decrease recall, and vice versa.