MALIGNANT COMMENTS CLASSIFIER PROJECT

Submitted by:

KONATALA MOHIT

ACKNOWLEDGMENT

I am grateful for the writer, developers and authority of the website pages geekforgeeks, wikipedia and stackoverflow which have helped me to refer for whenever there was need for guidance. Feature Engineering and usage of NLP helped me to get a better data to analyse and explore data. I am mostly grateful to DataTrained and Jose Portilla who have guided me in learning and enhancing my data science and NLP skills required to analyse and solve this project.

INTRODUCTION:

In this report I will be discussing about text classification of malignant comments by using nltk library and few of the machine learning models via python and its libraries. Social media platform is one most online surfed platform in today's world. Social media allows any individual to explore all the happening around the world via test data and visual clipping. It allows us to explore the content that we desire for and enlighten us with knowledge that is there to be grabbed. How much so ever the social media is a boon to us though it is equally bane to us as it leads to whole lot of vitriol and toxic data and comments. Malignant comments are the comments which consists of abusive words, aggressive phrase or the comments which are tend to be cyberbullying and such set of comments are recognized as harmful threat to the social media.

Famous celebrities and various content creators on social platform always tend to attract toxic comments and the toxicity is directly proportional to the growth of individual on the social media. Most of the time toxic comments tend to affect the target person mentally and it tends to the downfall of the individual wellbeing and mental stability. So, in order to deal with toxic comment first we need to detect the comments. To detect the malignant comments, one must have a good understanding of natural language processing, and usage of natural language toolkit to make the required data abled to feed it to a machine learning model.

Review of Literature:

For the better understanding of the data refer to Natural Language Processing & should know set of language and abusive words used in the data. Feature engineering is used to help in inspection of the comments before labelling it. Usage of Pipeline is required to simplify the process and required step by step process to clean and pre-process the data.

Undertaken Problem:

Objective of the project is to predict the label of the text comment i.e., if the comment is malignant or not by analysing various factors that affect the label outcome. Label prediction is one the many problems in Data Science as it ticks all the basic fundamentals required to analyse data and creating a machine learning model to predict the necessary outcome.

Mathematical/ Analytical Modelling of the Problem:

Label is the target data in the dataset and the variables present in the target data are in binary form, therefore Classification Machine Learning Models are used in this project. Classification is for estimating categorical form of data by using established relation between feature and target variables. Visual Analysis is used to study and observe the data.

Data Sources and their formats:

The source of the data is provided by the client. The data is provided in the csv format. Train data contains 159571 entries having 7 variables in the dataset and the test data contains 153164 entries having only 2 variables.

Data Pre-processing:

Data pre-processing has two main steps i.e., Data Cleaning and Data Transforming.

Data Cleaning:

Data Cleaning is one of the most important steps creating a machine model. If an uncleaned data is fed to a machine learning model, then the model will perform very poorly.

The Dataset does not contain any null values in it. Unnamed: 0 (serial number) is present in both the dataset as it unnecessary hence dropping this column. Except for comment text column all the other comments are in binary form. Comments text is in object string format.

Text Data has enormous junk words in which aren't useful in our label predict, therefore one must remove all the junk and unnecessary words present the comment text column.

Cleaning of the text data is as follows:

1. Word Tokenization.
2. Removal of punctuations.
3. Removal of stopwords.
4. Lemmatization.

Word Tokenization: with the help of nltk library we convert the string comment text into words so that is easier for processing to initiate. word_tokenize is the keyword used to convert a string into words.

Removal of punctuations: In English language usage of punctuation is common in order sound grammatically correct but in machine learning punctuation are not necessary so we remove the punctuation from the text. From string library, we call punctuation in order to remove the punctuations from the comments.

Removal of stop words: Same as the punctuations stop words are not necessary so we remove them by using stopwords keyword. This keyword already has all the commonly used stop words in our day-to-day life, although if one wants to add few more punctuation one can add those to this stopwords by modifying it. Punctuation and stop words are majority of the junk present in a text.

Lemmatization: It is process to simplify the verb form in order to decrease the uniqueness of words in the comments lot.

After performing these above stepped we join the words and make it to string format as before and add in to a list which is later converted into data frame and append to our existing data frame. Now we name it as clean comment text.

The whole above process is again repeated for the test set to the clean the data present in it.

Note: Label column is imbalanced.

Data Transforming:

After Data Cleaning the data must be transformed into numerical form as one can't feed ordinal data to machine learning model and also the data must be normalized as normalizing the data the machine learning model gives equal importance to all the data.

In this project I am using label token frequency inverse document frequency vectorizer to transform and fit the ordinal data present in the data set into numerical form. Tfidfvectorizer not convert the data into numerical form but has the ability to normalize the data. Before standardizing the data, we must split the data into train and test sets. Train Test Split method is to split the data into train and test data

After separating the train and test data we use Pipeline for model training and prediction as it makes the process simpler.

Data Inputs- Logic- Output Relationships:

After Separating the Input (feature) and output (target) data we split them into train and test division one part of the data is used to train the ML model and other part of the data to predict the output. When the train data is fed to machine learning model it generates an algorithm or simply put an equation that is applicable to all the data and when test data is fed to it implements the trained data equation to the current input values to predict the outcome. If the input has no outliers and is clean that there is no over or underfitting in the outcome.

Hardware and Software Requirements and Tools Used:

Hardware Required for Jupyter Notebook Software is as follows:

Memory and disk space required per user: 1GB RAM + 1GB of disk + .5 CPU core.

Server overhead: 2-4GB or 10% system overhead (whatever is larger), .5 CPU cores, 10GB disk space.

Port requirements: Port 8000 plus 5 unique, random ports per notebook

Libraries Used:

Pandas library: To frame raw data, visualize and perform task on it via other libraries.

Numpy library: To perform mathematical functions on the framed data numpy is used. In this project used it to location nan values and replace them with desired value also to find mean and standard values.

Matplotlib library: This library is used to visualize the data. Used to visualize univariate and bivariate analysis (pie plot, count plot and scatter plot) also to visualize outliers via box plot.

Seaborn library: heatmap to see co-relation between feature variable to arrest high collinearity.

Sklearn library: Imported this library to normalize the data, split the data into train and test data, various machine learning model and cross validation techniques.

Warnings library: To ignore filter warning shown while compiling block of codes

Pickle: To save the trained machine learning model.

Nltk: Imported this library to clean and preprocess the text data.

Algorithm Used:

1. XGB Classifier
2. Random Forest Classifier
3. K-Neighbors Classifier
4. Logistic Regression
5. Linear SVC

Model Training and Selection:

XGB Classifier:

XGBoost stands for "Extreme Gradient Boosting" and it is an implementation of gradient boosting trees algorithm. The XGBoost is a popular supervised machine learning model with characteristics like computation speed, parallelization, and performance

```
text_clf_3= Pipeline([('tfidf',TfidfVectorizer()),('log',XGBClassifier())])
text_clf_3.fit(X_train,y_train)
p3=text_clf_3.predict(X_test)
print(classification_report(p3, y_test))
```

```
[14:10:33] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1
0, the default evaluation metric used with the objective 'binary:logistic' w
t eval_metric if you'd like to restore the old behavior.
              precision    recall  f1-score   support

           0       0.99      0.96      0.97     29960
           1       0.57      0.90      0.70      1955

    accuracy                           0.95     31915
   macro avg       0.78      0.93      0.84     31915
weighted avg       0.97      0.95      0.96     31915
```

Random Forest Classifier:

A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting

```
text_clf_5= Pipeline([('tfidf',TfidfVectorizer()),('log',RandomForestClassifier())])
text_clf_5.fit(X_train,y_train)
p5=text_clf_5.predict(X_test)
print(classification_report(p5, y_test))
```

```
              precision    recall  f1-score   support

           0       0.99      0.95      0.97     30118
           1       0.54      0.91      0.68      1797

    accuracy                           0.95     31915
   macro avg       0.77      0.93      0.83     31915
weighted avg       0.97      0.95      0.96     31915
```

K-Neighbors Classifier:

KNN is a non-parametric method that, in an intuitive manner, approximates the association between independent variables and the continuous outcome by averaging the observations in the same neighbourhood

```
text_clf_4= Pipeline([('tfidf',TfidfVectorizer()),('log',KNeighborsClassifier())])
text_clf_4.fit(X_train,y_train)
p4=text_clf_4.predict(X_test)
print(classification_report(p4, y_test))
```

```
              precision    recall  f1-score   support

           0       1.00      0.92      0.96     31263
           1       0.18      0.86      0.30       652

    accuracy                           0.92     31915
   macro avg       0.59      0.89      0.63     31915
weighted avg       0.98      0.92      0.94     31915
```

Logistic Regression:

Logistic regression is a process of modelling the probability of a discrete outcome given an input variable. The most common logistic regression models a binary outcome, something that can take two values such as true/false, yes/no, and so on.

```
text_clf_2= Pipeline([('tfidf',TfidfVectorizer()),('log',LogisticRegression())])
text_clf_2.fit(X_train,y_train)
p2=text_clf_2.predict(X_test)
print(classification_report(p2, y_test))
```

```
              precision    recall  f1-score   support

           0       0.99      0.96      0.98     29905
           1       0.60      0.91      0.72      2010

    accuracy                           0.96     31915
   macro avg       0.80      0.93      0.85     31915
weighted avg       0.97      0.96      0.96     31915
```

Linear Support Vector Classification:

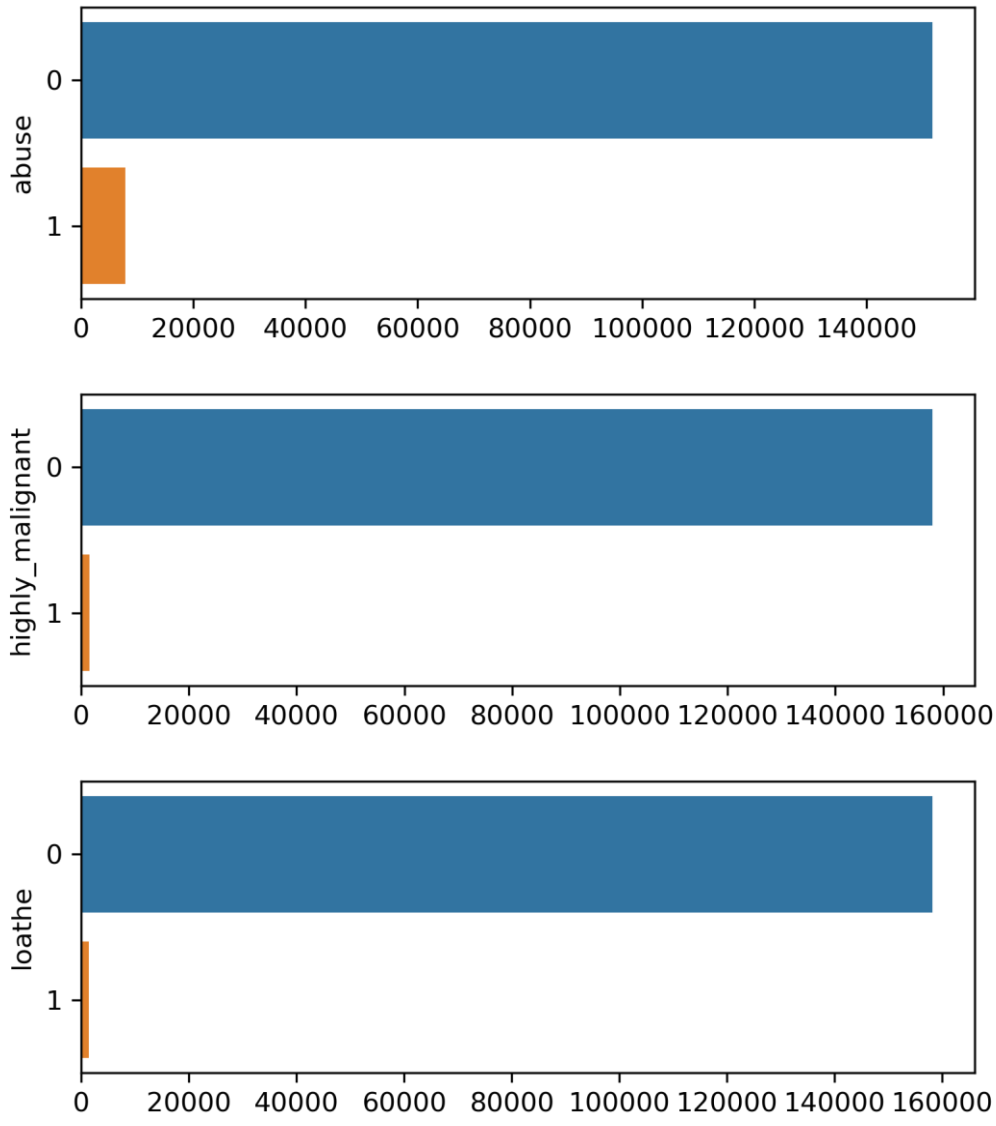
Similar to SVC with parameter kernel='linear', but implemented in terms of liblinear rather than libsvm, so it has more flexibility in the choice of penalties and loss function and should scale better to large number of samples.

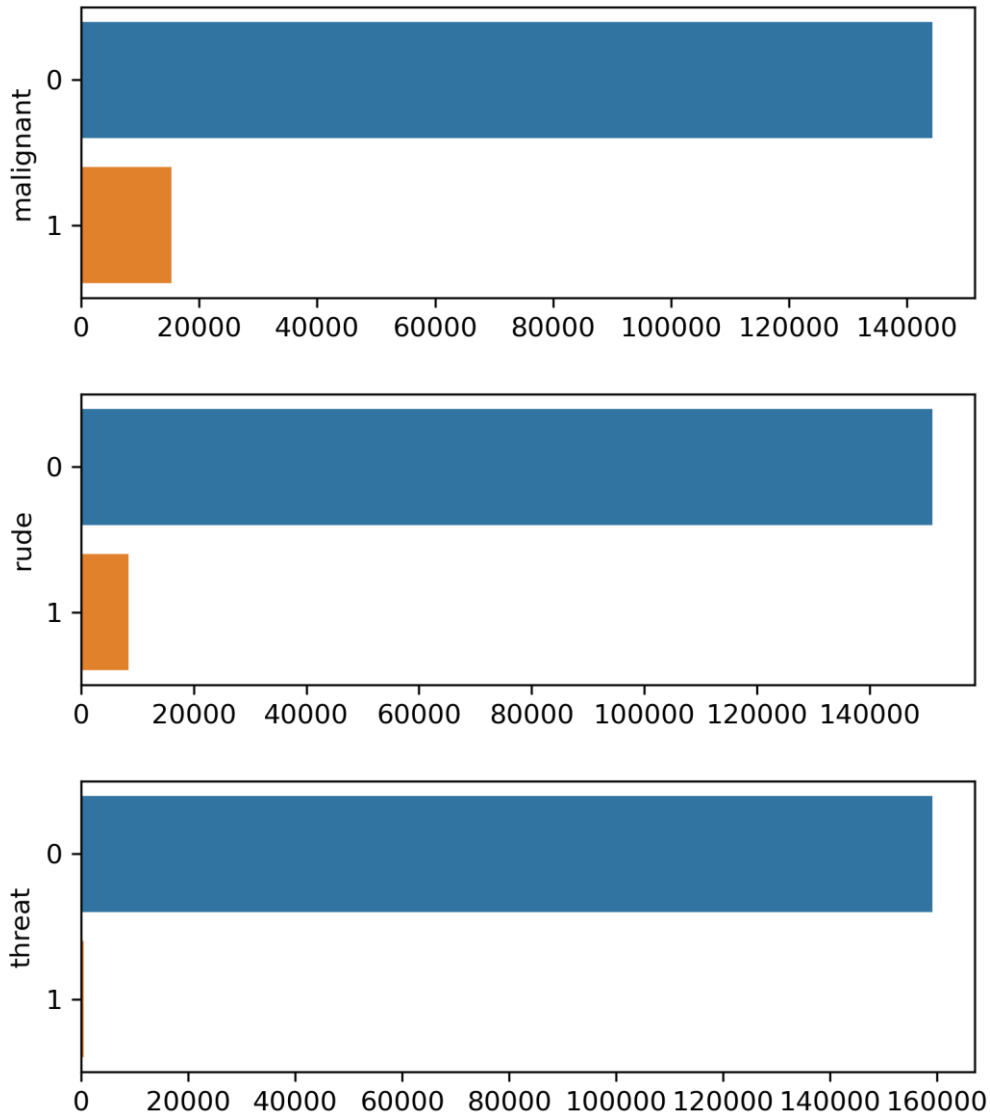
This class support both dense and sparse input and the multiclass support is handled according to a one-vs-the-rest scheme.

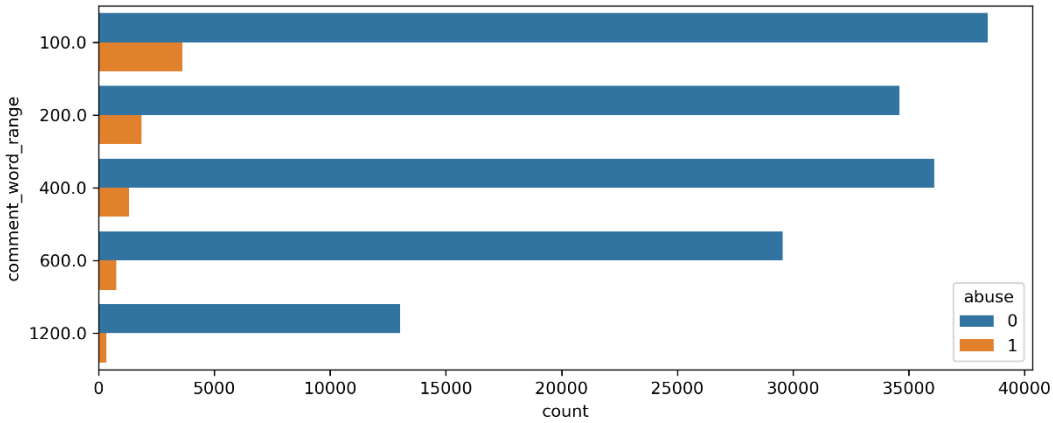
```
print(classification_report(p1, y_test))
```

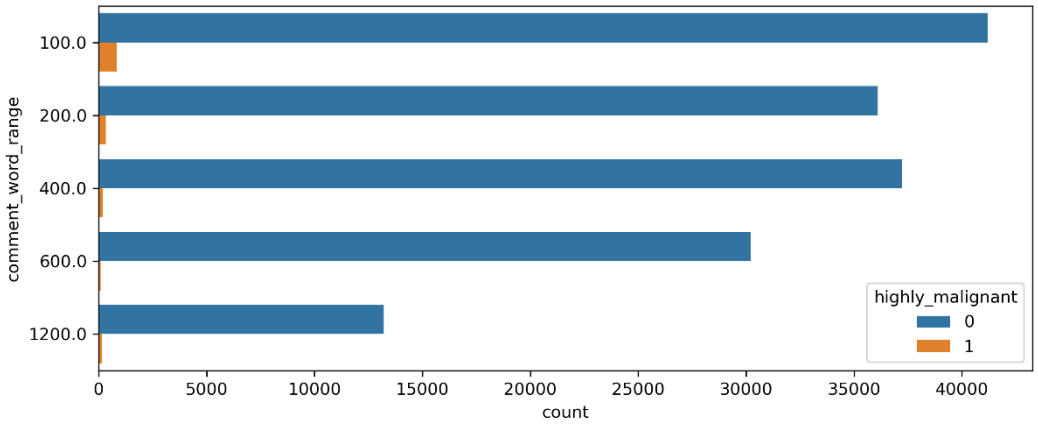
	precision	recall	f1-score	support
0	0.99	0.97	0.98	29539
1	0.67	0.86	0.76	2376
accuracy			0.96	31915
macro avg	0.83	0.92	0.87	31915
weighted avg	0.97	0.96	0.96	31915

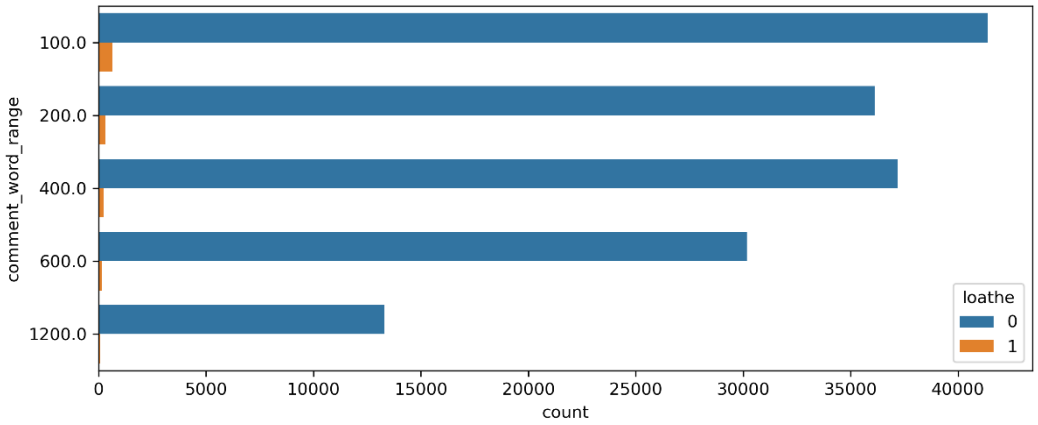
Visualization:

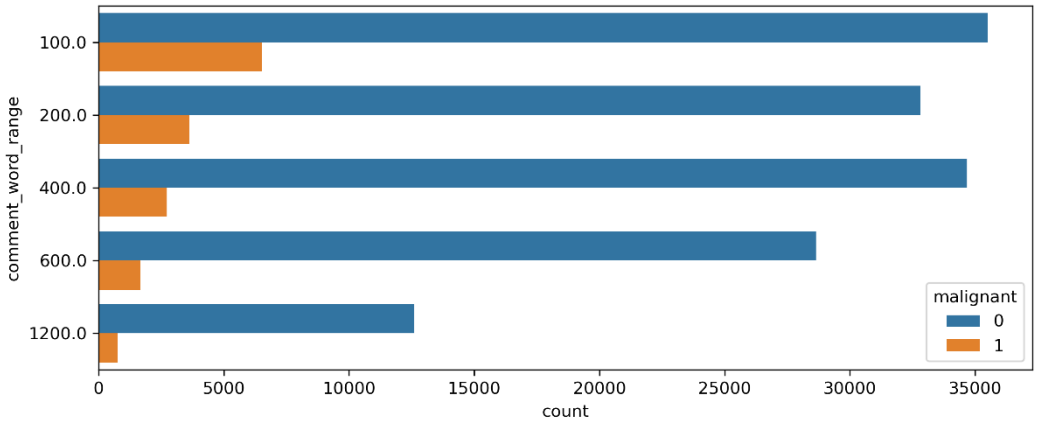


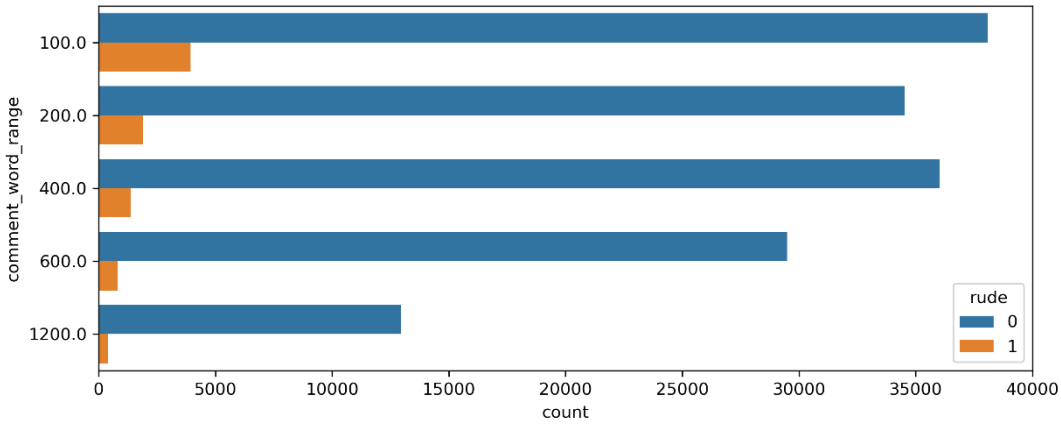

malignant
0
1
0
20000
40000
60000
80000
100000
120000
140000
rude
0
1
0
20000
40000
60000
80000
100000
120000
140000
threat
0
1
0
20000
40000
60000
80000
100000
120000
140000
160000

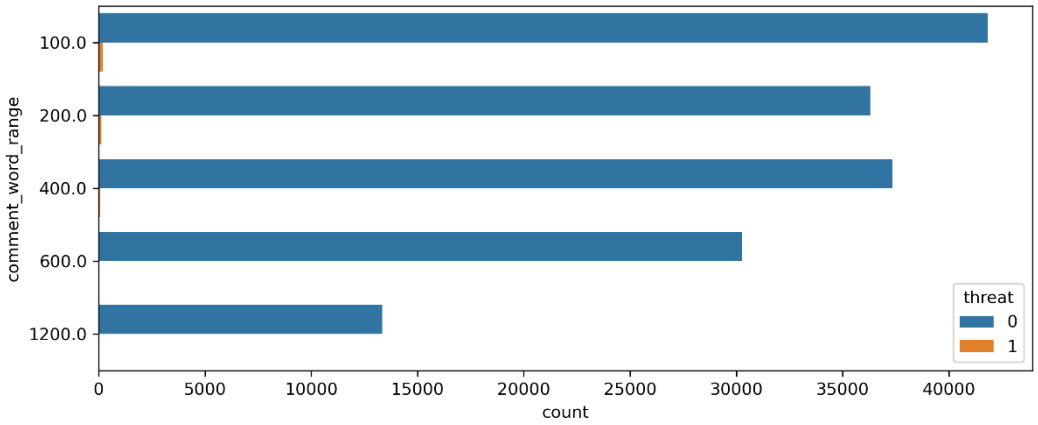

comment_word_range
100.0
200.0
400.0
600.0
1200.0
abuse
0
1
0
5000
10000
15000
20000
25000
30000
35000
40000
count

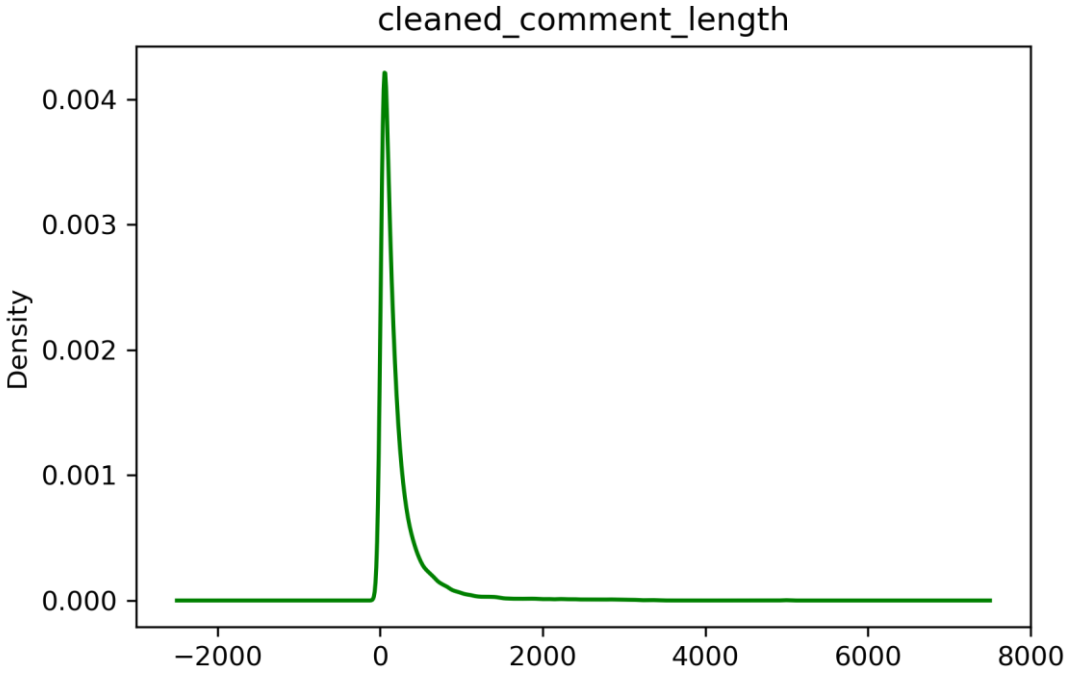

comment_word_range
100.0
200.0
400.0
600.0
1200.0
highly_malignant
0
1
0
5000
10000
15000
20000
25000
30000
35000
40000
count

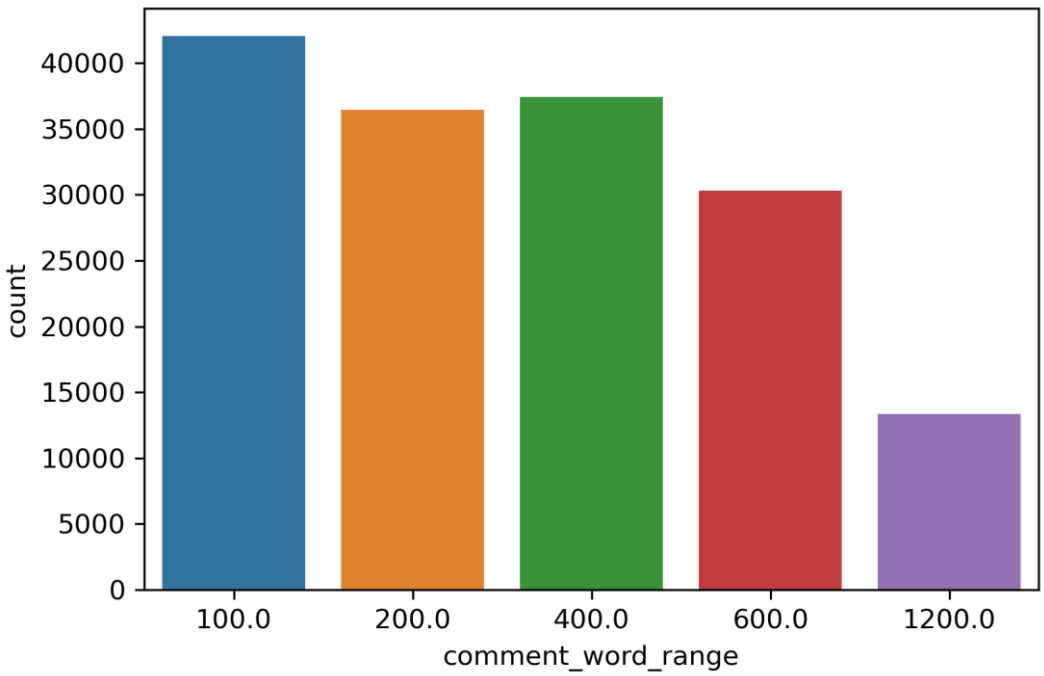

comment_word_range
100.0
200.0
400.0
600.0
1200.0
loathe
0
1
0
5000
10000
15000
20000
25000
30000
35000
40000
count

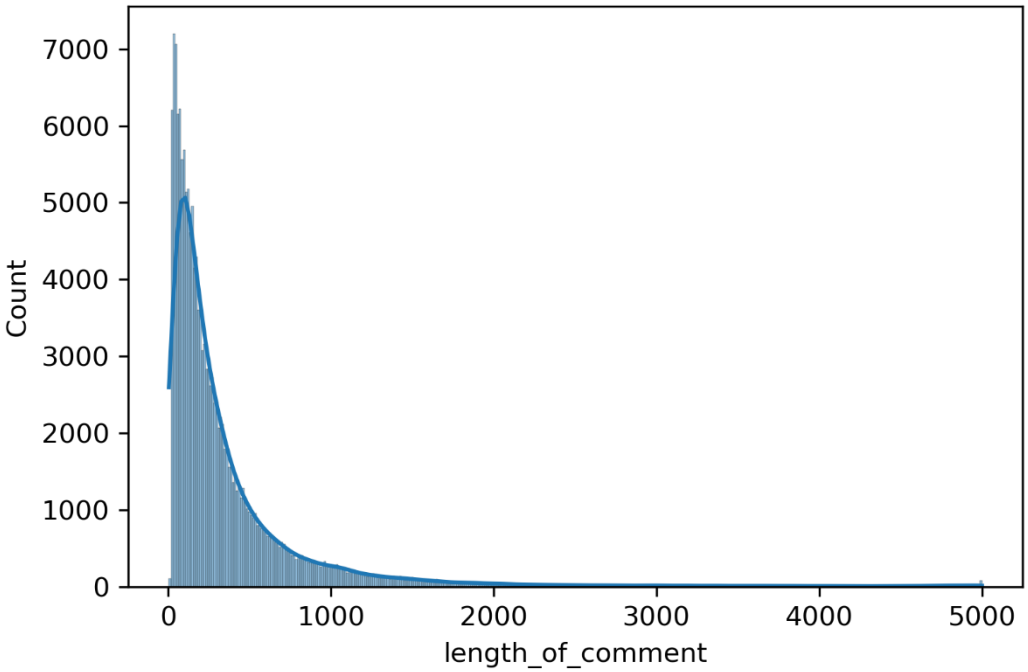

comment_word_range
100.0
200.0
400.0
600.0
1200.0
malignant
0
1
0
5000
10000
15000
20000
25000
30000
35000
count

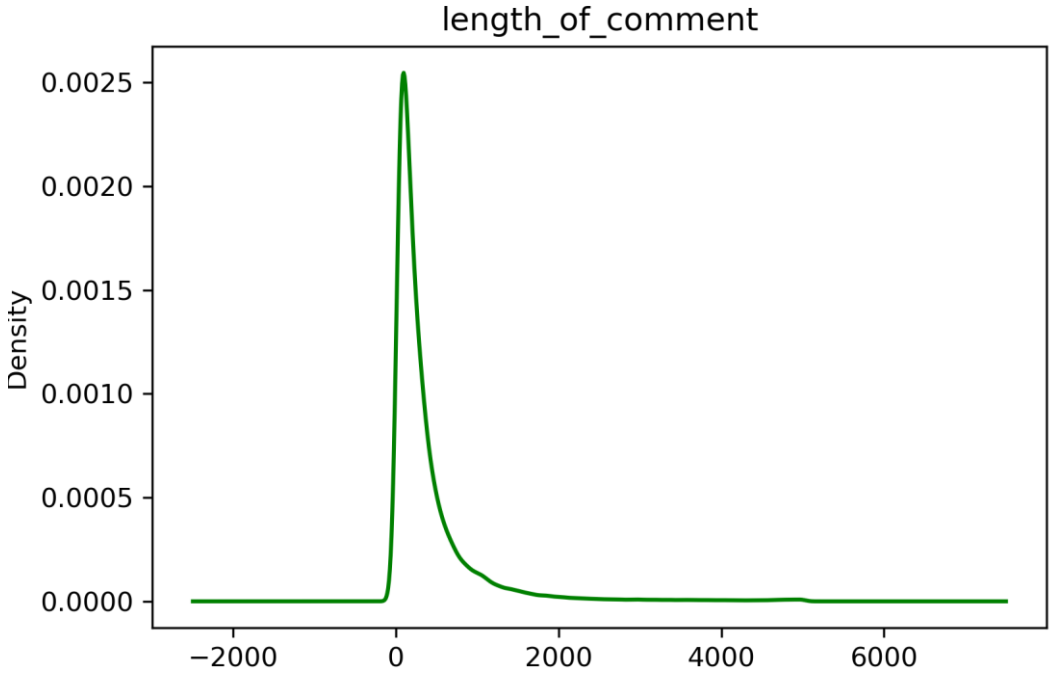

comment_word_range
100.0
200.0
400.0
600.0
1200.0
rude
0
1
0
5000
10000
15000
20000
25000
30000
35000
40000
count

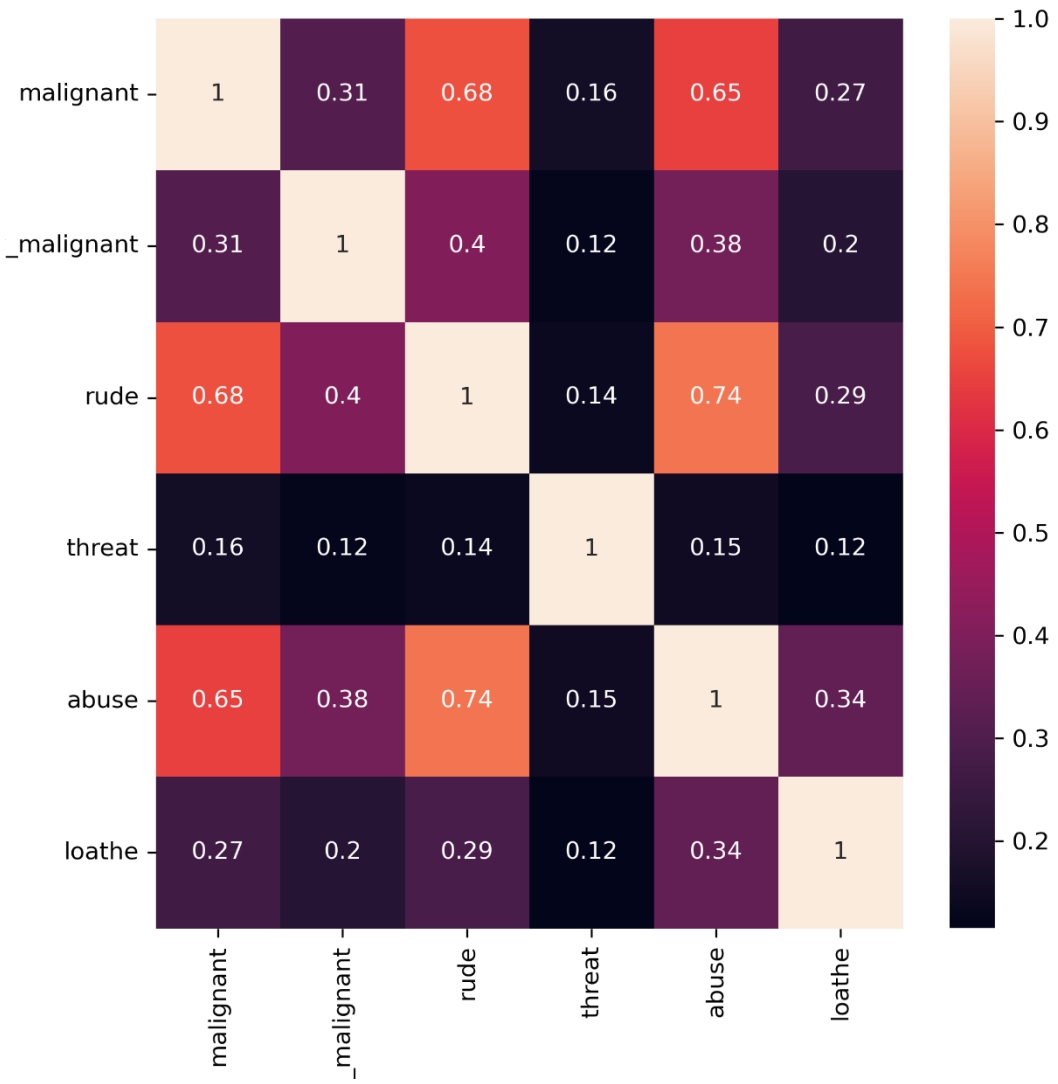

comment_word_range
100.0
200.0
400.0
600.0
1200.0
threat
0
1
0
5000
10000
15000
20000
25000
30000
35000
40000
count

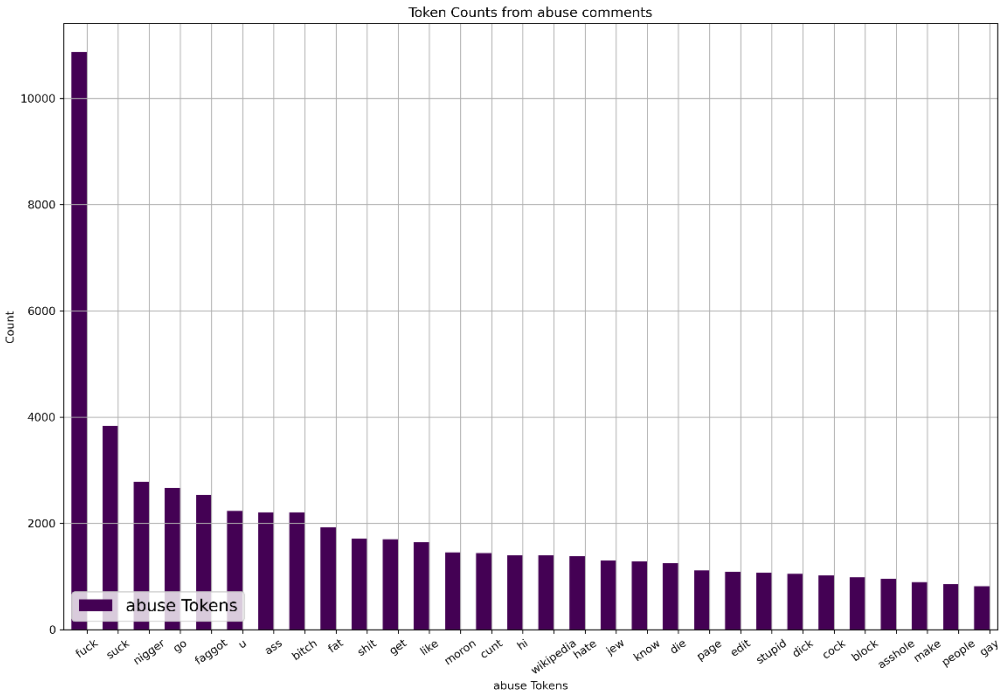

cleaned_comment_length
Density
0.004
0.003
0.002
0.001
0.000
−2000
0
2000
4000
6000
8000

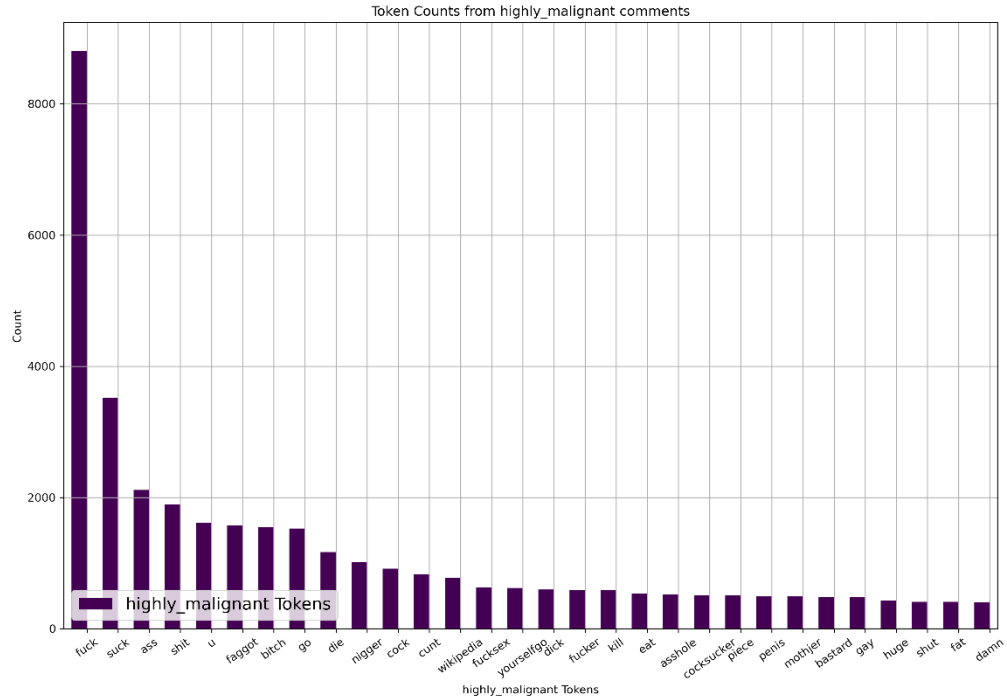

count
40000
35000
30000
25000
20000
15000
10000
5000
0
100.0
200.0
400.0
600.0
1200.0
comment_word_range

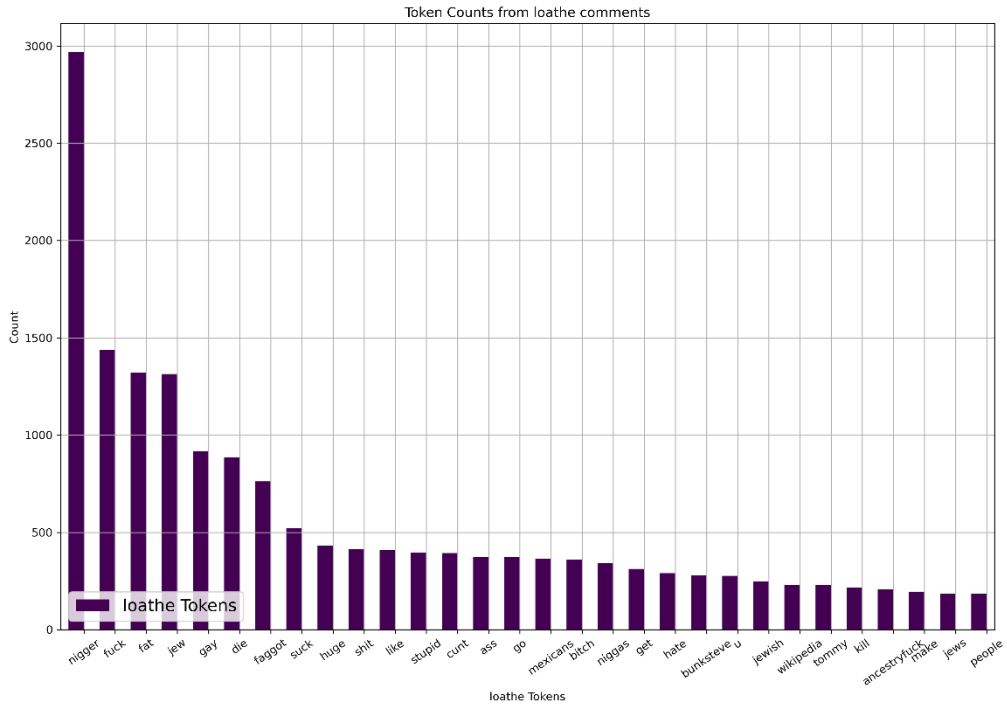

7000
6000
5000
4000
3000
2000
1000
0
Count
0
1000
2000
3000
4000
5000
length_of_comment

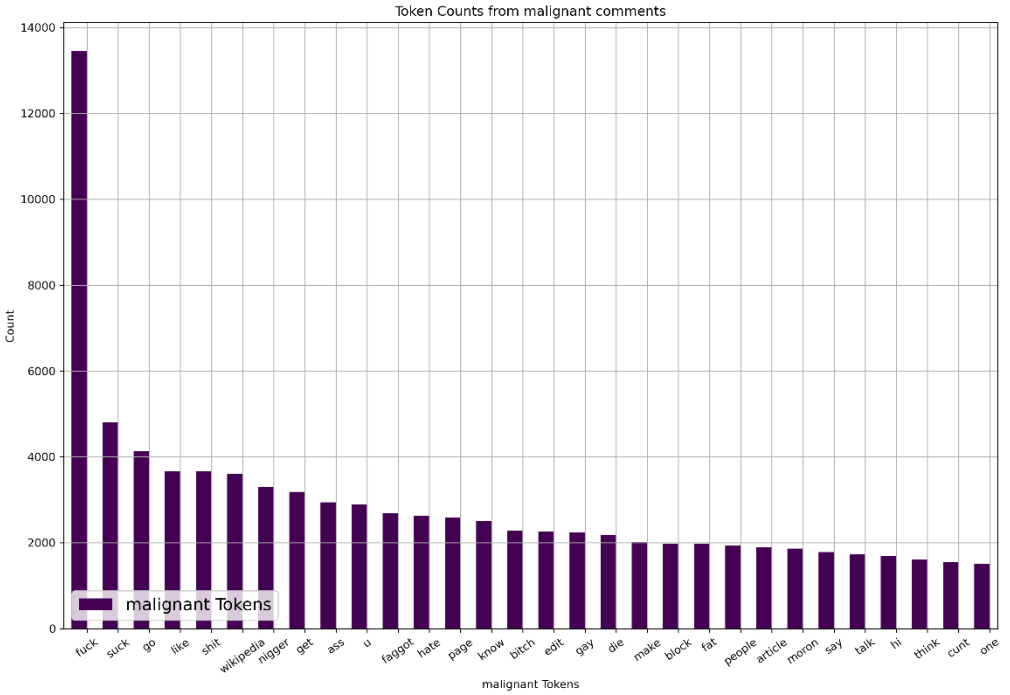

length_of_comment
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
Density
−2000
0
2000
4000
6000

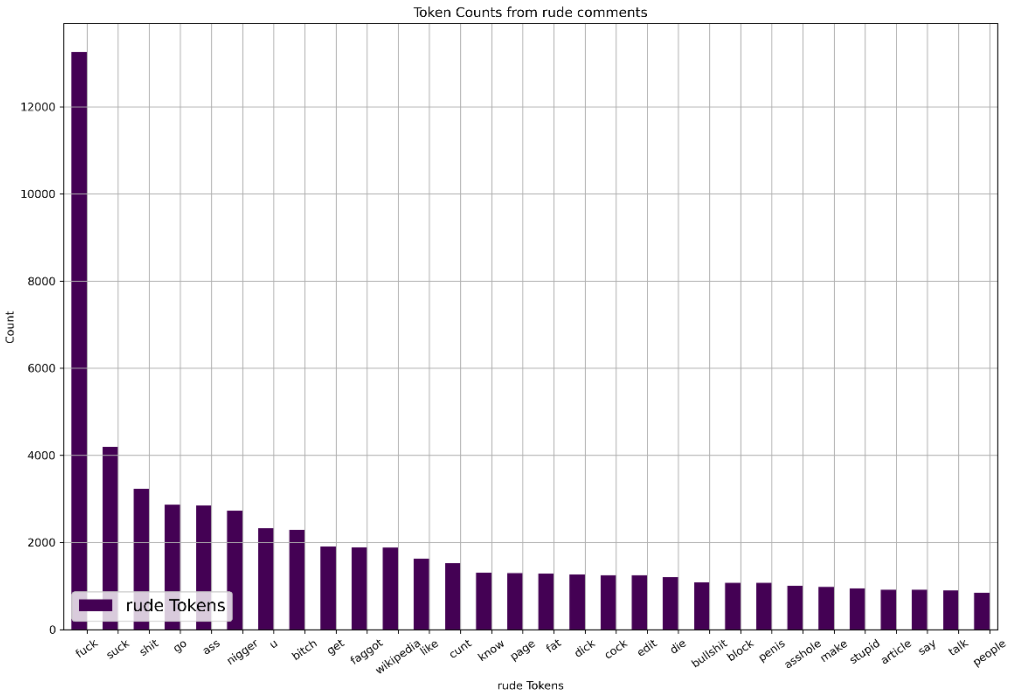

malignant
_malignant
rude
threat
abuse
loathe
1 0.31 0.68 0.16 0.65 0.27
0.31 1 0.4 0.12 0.38 0.2
0.68 0.4 1 0.14 0.74 0.29
0.16 0.12 0.14 1 0.15 0.12
0.65 0.38 0.74 0.15 1 0.34
0.27 0.2 0.29 0.12 0.34 1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2

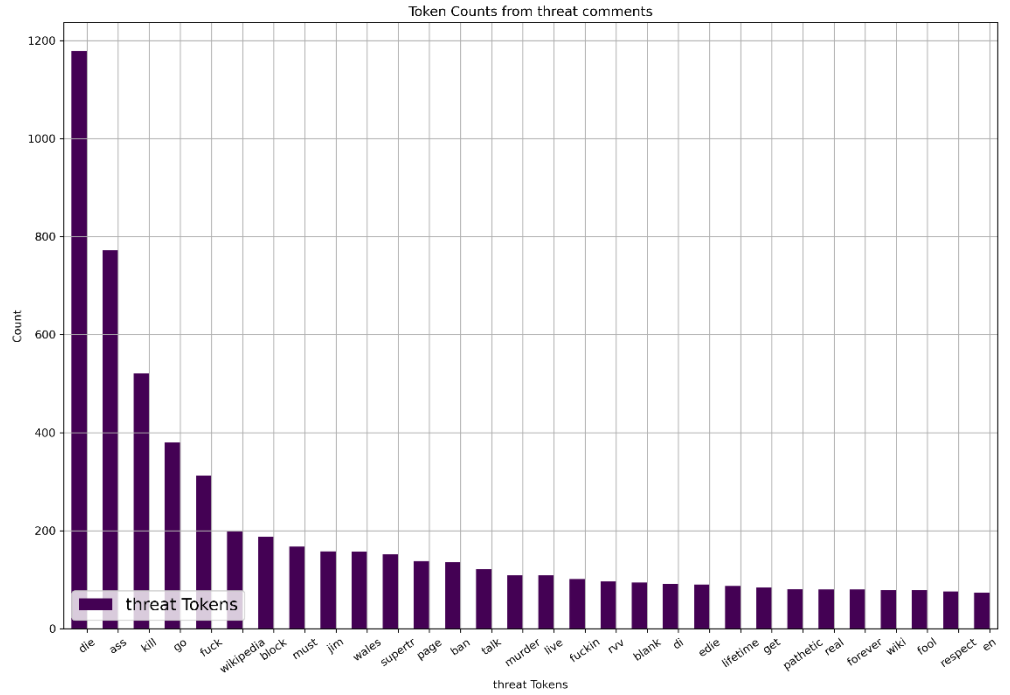

Token Counts from abuse comments
Count
10000
8000
6000
4000
2000
0
abuse Tokens
fuck
suck
nigger
go
faggot
u
ass
bitch
fat
shit
get
like
moron
cunt
hi
wikipedia
hate
jew
know
die
page
edit
stupid
dick
cock
block
asshole
make
people
gay
abuse Tokens

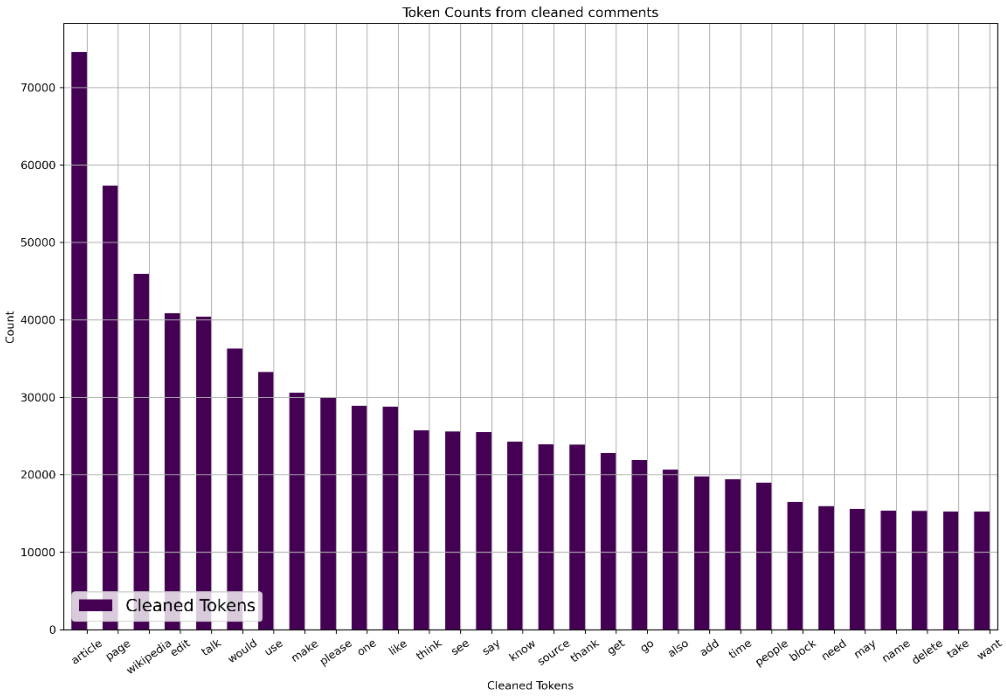

Token Counts from highly_malignant comments
Count
8000
6000
4000
2000
0
highly_malignant Tokens
fuck
suck
ass
shit
u
faggot
bitch
go
die
nigger
cock
cunt
wikipedia
fucksex
yourselfgo
dick
fucker
kill
eat
asshole
cocksucker
piece
penis
mothjer
bastard
gay
huge
shut
fat
damn
highly_malignant Tokens

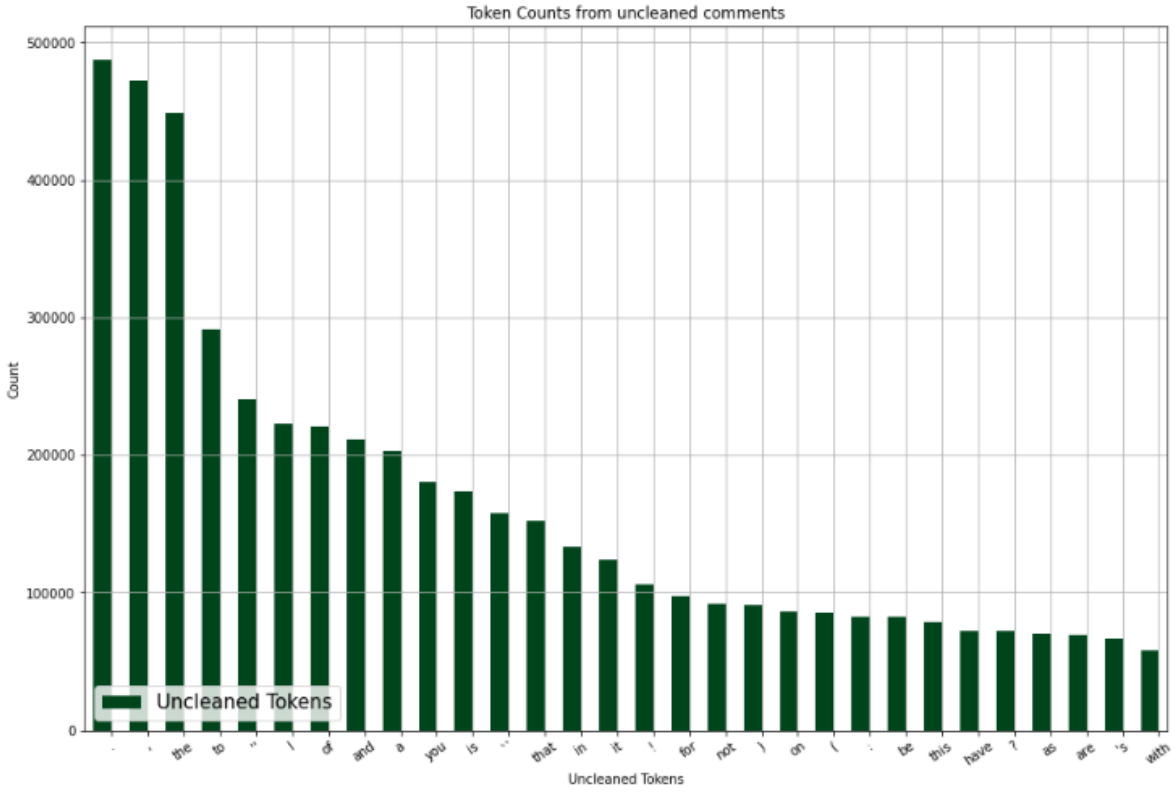

Token Counts from loathe comments
Count
loathe Tokens
nigger
fuck
fat
jew
gay
die
faggot
suck
huge
shit
like
stupid
cunt
ass
go
mexicans
bitch
niggas
get
hate
bunksteve
u
jewish
wikipedia
tommy
kill
ancestryfuck
make
jews
people

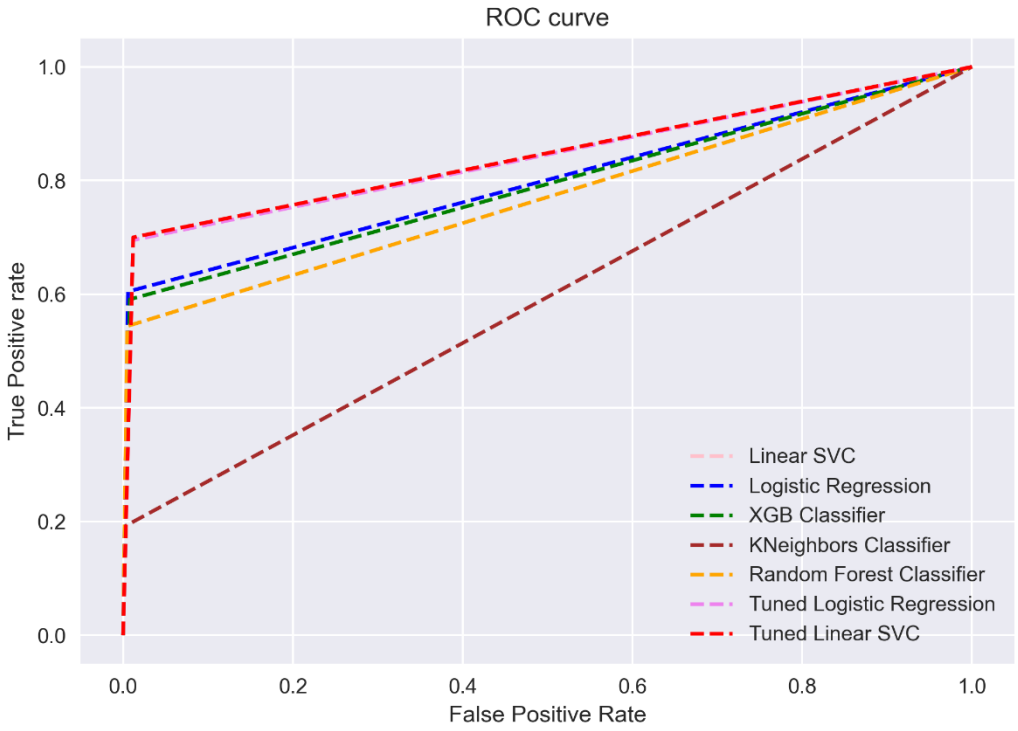

Token Counts from malignant comments
Count
malignant Tokens
fuck
suck
go
like
shit
wikipedia
nigger
get
ass
u
faggot
hate
page
know
bitch
edit
gay
die
make
block
fat
people
article
moron
say
talk
hi
think
cunt
one


Token Counts from rude comments
Count
rude Tokens
rude Tokens
fuck
suck
shit
go
ass
nigger
u
bitch
get
faggot
wikipedia
like
cunt
know
page
fat
dick
cock
edit
die
bullshit
block
penis
asshole
make
stupid
article
say
talk
people


Token Counts from threat comments
Count
threat Tokens
threat Tokens
die
ass
kill
go
fuck
wikipedia
block
must
jim
wales
supertr
page
ban
talk
murder
live
fuckin
rvv
blank
di
edie
lifetime
get
pathetic
real
forever
wiki
fool
respect
en


Token Counts from cleaned comments
Count
Cleaned Tokens
article
page
wikipedia
edit
talk
would
use
make
please
one
like
think
see
say
know
source
thank
get
go
also
add
time
people
block
need
may
name
delete
take
want
Cleaned Tokens


Token Counts from uncleaned comments
Count
Uncleaned Tokens
the
to
of
and
a
you
is
that
in
it
for
not
on
be
this
have
as
are
with
Uncleaned Tokens



Observations:

1. Malignant, highly malignant, rude, threat, abuse, loathe labelled as are very minor as majority of the data is non-malignant.
2. Malignant has maximum compared to all the other labels.
3. Comments constructed with less than 100 words have more malignant, highly malignant, rude, threat, abuse and loath comments and words with above 1200 has minimum set of it.
4. Malignant, rude and abuse comments are highly co-relative to each other.
5. In uncleaned comments most used words are punctuations and stop words.
6. In cleaned comments most used words is ‘article’, followed by page, Wikipedia, edit and talk.
7. Most comment have a word count of below 100 and minimum comments have a word count of above 1200.
8. Top words for malignant comments are ‘fuck’, ‘suck’, ‘go’, ‘shit’.
9. Top words for highly malignant, abuse and rude comments are same as malignant.
10. The top word for threat comment is ‘kill’ followed by ‘die’.

11. The top word for loathe comment is ‘nigger’.
12. Average comment length for uncleaned comment is 394 words.
13. Average comment length for cleaned comment is 238 words.

Results:

Out of all the Classifier models’ Linear SVC has performed well, KNN model is not at all suitable for this dataset. Logistic Regression has also performed well this model. XGB Classifier and Random Forest Classifier both are time consuming and have underperformed this dataset.

Conclusion:

In the whole dataset the top 15 words which were visualized must have the most influence on the label columns.

Outcome of the Study:

Visualizing data helped to explore and study the data. Data Cleaning helps in minimizing the overfitting created during model training and improves the model performance. Random Forest can neglect outliers even when the data is fed with outliers to the machine learning model. XGB Classifier is not worth the use in this type of problem as it required ample amount of time and has underperformed. Simplifying the data was the most challenge part in this project but it can be overcome if minimal domain knowledge in NLP is exploited into this project.

Limitations of this work and Scope for Future Work:

The label predicted are only limited to particular malignant comment. When we try to predict for label the machine learning model will fail. To improve the model efficiency, we can add more of failure label in order to balance the data and get much improved precision, recall and f1-score for the given model. Also by removing more unnecessary words from the data which are not affecting the output data the model can be improvised.