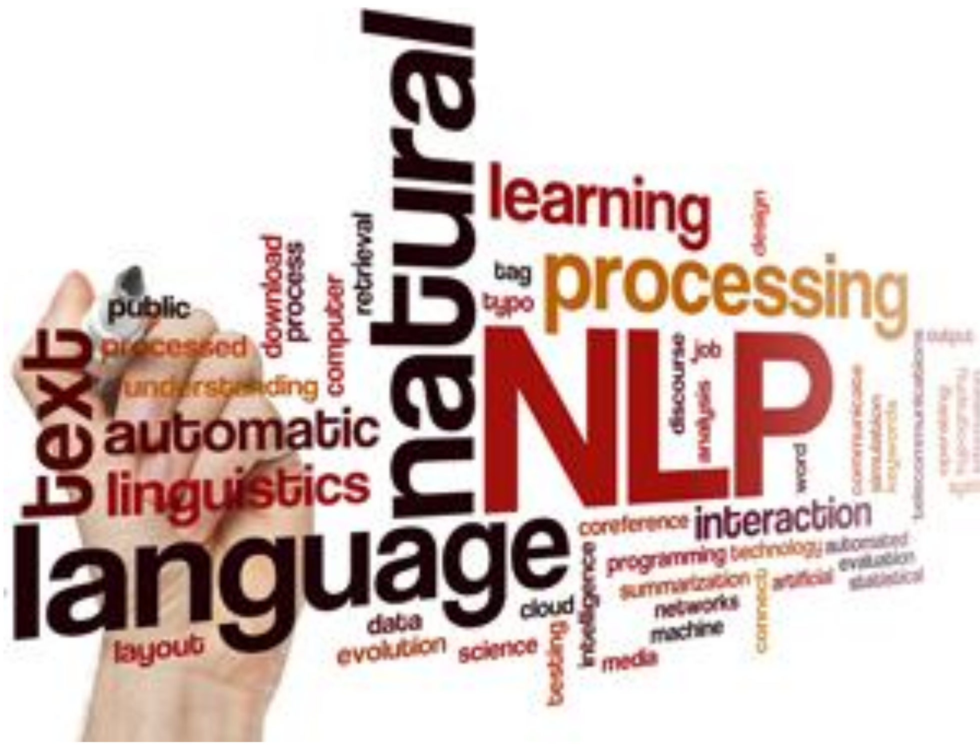
MALIGNANT COMMENTS CLASSIFICATION MODEL



Submitted by:
Prateek

ACKNOWLEDGEMENT

The internship opportunity I have with Flip Robo Technologies is a great chance for learning and professional development. I perceive this opportunity as a big milestone in my career development. I will strive to use gained skills acknowledge in the best possible way. I would like to extend my appreciation and thanks for the mentors from DataTrained and professionals from FlipRoboTechnologies who had extended their help and support.

References:

https://sklearn.org/supervised_learning.html#supervisedlearning
https://www.datacamp.com/community https://github.com/mxc19912008/Andrew-Ng-MachineLearning-Notes https://www.analyticsvidhya.com/blog/category/machinelearning/

INTRODUCTION

Business 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. 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 behavior. 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. 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.

Background of domain:

With the proliferation of smart devices and mobile and social network environments, the social side effects of these technologies, including cyberbullying through malicious comments and rumors, have become more serious. Malicious online comments have emerged as an unwelcome social issue worldwide. Both cyberbullying and malicious comments are increasingly viewed as a social problem due to their role in suicides and other real-world crimes. However, the online environment generally lacks a system of barriers to prevent privacy invasion, personal attacks, and cyberbullying, and the barriers that do exist are weak. Social violence as an online phenomenon is increasingly pervasive, a phenomenon manifesting itself through social divisiveness. Motivations for malicious comments identified in the study involved targeting people's mistakes. Conversely, most benevolent comments involved encouragement and compliments to help people in difficult or risky situations, showing malicious comments is a primary reason for degradation of online social networks. Moreover, the abolition of anonymity and intensification of punishment in social media can be effective in reducing malicious comments and rumors. However, potential violation of freedom of expression also risks trivializing the online social network itself. Because anonymous forms of freedom of expression have always been controversial in theoretical and normative spheres of social research. Careful consideration of any limiting of comments is necessary before a ban might be contemplated. Requiring true identities would cause them to be more careful and responsible. Our results also suggest providers of social media services can apply text filters to their systems. Because certain texts are used repeatedly in cyberbullying or malicious comments, providers should be persuaded to develop a system to detect certain texts and alert them as to when to possibly take action against the people posting them. Such a filtering function could reduce the number of all kinds of malicious comments. Conversely, social media service providers should consider posting lists that rank users most active in posting benevolent comments on their sites. Because people generally enjoy self-expression, these rankings could motivate more people to post positive comments as a way to develop a new social norm in which malicious comments are unwelcome and the people posting them are scorned.

MOTIVATION FOR PROBLEM UNDER TAKEN:

Based on data provided, a comment is assessed based on different factors. By building the model, we can assess which comments are highly likely to be hateful in varying degrees of hate thereby it will be useful for those people who are target of online hate comments by deleting those comments.

ANALYTICAL PROBLEM FRAMING MATHEMATICAL MODELLING OF PROBLEM:

Mathematical modeling is simply the method of implementing statistical analysis to a dataset where a Statistical Model is a mathematical representation of observed data. While analyzing the data, there are an array of statistical models we can choose to utilize. For the given project, we need to predict whether the given comment falls into the any category of hate. This is a classification problem. There are wide varieties of classification models like decision trees, random forests, nearest neighbor, Logistic Regression, etc.

DATA SOURCE AND BASIC INFO CHECK:

The data has been provided in different train and test datasets in a comma separated values(.csv) format.

1. The data will be loaded into pandas dataframe.

```
df = pd.read_csv(r'/content/drive/MyDrive/DataScience/malagnant comments/train.csv')
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

2. Checking basic info.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 159571 entries, 0 to 159570
Data columns (total 8 columns):
 #   Column            Non-Null Count   Dtype 
---  ------            --------------   ----- 
 0   id                159571 non-null  object
 1   comment_text      159571 non-null  object
 2   malignant         159571 non-null  int64 
 3   highly_malignant  159571 non-null  int64 
 4   rude              159571 non-null  int64 
 5   threat            159571 non-null  int64 
 6   abuse             159571 non-null  int64 
 7   loathe            159571 non-null  int64 
dtypes: int64(6), object(2)
memory usage: 9.7+ MB
```

3. We drop id column as it is of no use in model building. Then we check for duplicates in the data.

```
[ ] df.drop('id',axis=1,inplace=True)
```

Checking Duplicates

```
[ ] df.duplicated().sum()
```

```
0
```

4. We then check for Null Values.

```
df.isnull().sum()
```

```
comment_text        0
malignant           0
highly_malignant    0
rude                0
threat              0
abuse               0
loathe              0
dtype: int64
```

5. Checking basic description.

```
df.describe()
```

	malignant	highly_malignant	rude	threat	abuse	loathe
count	159571.000000	159571.000000	159571.000000	159571.000000	159571.000000	159571.000000
mean	0.095844	0.009996	0.052948	0.002996	0.049364	0.008805
std	0.294379	0.099477	0.223931	0.054650	0.216627	0.093420
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

```
df.dtypes
```

```
comment_text        object
malignant            int64
highly_malignant     int64
rude                 int64
threat               int64
abuse                int64
loathe               int64
dtype: object
```

This data set has around 1lakh,59thousand rows and 7 columns. Comment_text is object column whereas malignant, highly malignant, rude, threat, abuse and loathe are numeric columns.

DATA PRE-PROCESSING:

Data preprocessing is a technique of converting raw data into useful format.

1. Removing Numbers

- Removing Numbers

```
[ ] count = 0
    for i in range(len(df['comment_text'])):
        z = re.findall('[0-9]',df['comment_text'][i])
        if len(z)>1:
            count+=1
    print(count)

    45530
```

```
[ ] df['comment_text'][0]

    'Explanation\nWhy the edits made under my username Hardcore Metallica
    now.89.205.38.27'
```

```
[8] def remove_numbers(text):
        return re.sub("\d+", "", text)
```

```
[ ] df['comment_text'] = df['comment_text'].apply(remove_numbers)
```

```
[ ] df['comment_text'][0]

    'Explanation\nWhy the edits made under my username Hardcore Metallica
    now....'
```

2. Removing Emoji

- Removing Emojis

```
[ ] df['comment_text'][107479]

    'Ok maybe I was wrong? I'm sorry for what I did mate please don't take what I say seriously at the end of the day I'm only  and I have just lost my best friend to a car accident 😔 please forgive me mate 💜'
```

From above, we see that the text does contain Emoji's. Let's replace them with their meanings.

```
[9] !pip install emoji

    Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
    Requirement already satisfied: emoji in /usr/local/lib/python3.7/dist-packages (2.0.0)

[10] import emoji

[11] def remove_emoji(text):
         return emoji.demojize(text)

    df['comment_text'] = df['comment_text'].apply(remove_emoji)

                                   + Code   + Text

[ ] df['comment_text'][107479]

    'Ok maybe I was wrong? I'm sorry for what I did mate please don't take what I say seriously at the end of the day I'm only  and I have just lost my best friend to a car accident :pensive_face: please forgive me mate :purple_heart:
```

We see the Emojis have been removed

3. Chat Words Treatment

```
    'PITA':'Pain In The A..',
    'PRT':'Party',
    'PRW':'Parents Are Watching',
    'QPSA?':'Que Pasa?',
    'ROFL':'Rolling On The Floor Laughing',
    'ROFLOL':'Rolling On The Floor Laughing Out Loud',
    'ROTFLMAO':'Rolling On The Floor Laughing My A.. Off',
    'SK8':'Skate',
    'STATS':'Your sex and age',
    'ASL':'Age, Sex, Location',
    'THX':'Thank You',
    'TTFN':'Ta-Ta For Now!',
    'TTYL':'Talk To You Later',
    'U':'You',
    'U2':'You Too',
    'U4E':'Yours For Ever',
    'WB':'Welcome Back',
    'WTF':'What The F...',
    'WTG':'Way To Go!',
    'WUF':'Where Are You From?',
    'W8':'Wait...',
    '7K':'Sick:-D Laugher'}

] def chat_conversations(text):
    new_text = []
    for w in text.split():
        if w in chat_words:
            new_text.append(chat_words[w])
        else:
            new_text.append(w)
    return " ".join(new_text)

] df['comment_text'] = df['comment_text'].str.upper()

] df['comment_text'] = df['comment_text'].apply(chat_conversations)

] df['comment_text'][2],df['comment_text'][325]

 ("HEY MAN, i am REALLY NOT TRYING TO EDIT WAR. IT'S JUST THAT THIS GUY IS CONSTANTLY REMOVING RELEVANT INFORMATION AND
  "Laughing Out Loud CENSORSHIP. YOU'RE REALLY DISAPPOINTED THAT NPOV IS A CORE POLICY, AREN'T YOU? (TALK · CONTRIBS)")
```

4. Lower Casing

- Lower Casing

```
[ ] df['comment_text'] = df['comment_text'].str.lower()
    df['comment_text'].head(3)

    0    explanation why the edits made under my userna...
    1    d'aww! he matches this background colour i am ...
    2    hey man, i am really not trying to edit war. i...
    Name: comment_text, dtype: object
```

5. Removing Emails

```
def remove_emails(text):
    pattern = re.compile('[a-z0-9\.\-+_]+@[a-z0-9\.\-+_]+\.[a-z]+')
    return pattern.sub(r'',text)
```

```
df['comment_text'] = df['comment_text'].apply(remove_emails)
```

```
count = 0
for i in range(len(df)):
    z = re.findall('[a-z0-9\.\-+_]+@[a-z0-9\.\-+_]+\.[a-z]+',df['comment_text'][i])
    if len(z)>3:
        print(i,z)
        print()
        print('*'*50)
        count+=1
print(count)
```

```
0
```

6. Removing Weblinks

```
def remove_html_tags(text):
    pattern = re.compile('http\S+|www\S+')
    return pattern.sub(r'',text)
```

```
df['comment_text'] = df['comment_text'].apply(remove_html_tags)
```

```
count = 0
for i in range(len(df)):
    z = re.findall('http\S+|www\S+',df['comment_text'][i])
    if len(z)>3:
        print(i,z)
        print()
        print('*'*50)
        count+=1
print(count)
```

```
0
```

7. Removing Punctuations

- Removing Punctuations

```
[ ] df['comment_text'][0]

    'explanation why the edits made under my username hardcore metallica
    now....'
```

```
import string
exclude = string.punctuation+ ('·')+('•')
exclude
```

```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~·•'
```

```
[19] def remove_punctuations(text):
         return text.translate(str.maketrans('','',exclude))
```

```
[ ] df['comment_text'] = df['comment_text'].apply(remove_punctuations)
```

8. Removing Stop Words

Removing Stop Words

```
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
True
```

```
[21] from nltk.corpus import stopwords

     # Initialize the stopwords
     stoplist = stopwords.words('english')
```

```
df['comment_text'][1]
```

```
'D'aww! He matches this background colour I'm seemingly stuck with. Thanks.  (talk) 21:51, January 11, 2016 (UTC)'
```

```
[24] def remove_stopwords(text):
         new_text = []
         for words in text.split():
             if words in stoplist:
                 new_text.append('')
             else:
                 new_text.append(words)
         return " ".join(new_text)
```

```
[ ] for i in tqdm(range(len(df))):
        df['comment_text'][i] = remove_stopwords(df['comment_text'][i])
```

```
100%|██████████| 159571/159571 [04:47<00:00, 555.63it/s]
```

```
[ ] df['comment_text'][1]
```

```
'daww  matches  background colour   seemingly stuck  thanks talk january utc'
```

9. Lemmatization

Lemmatization

```
[ ] import nltk
    from nltk.stem import WordNetLemmatizer
```

```
[ ] lemmatiser = WordNetLemmatizer()
    nltk.download('wordnet')
```

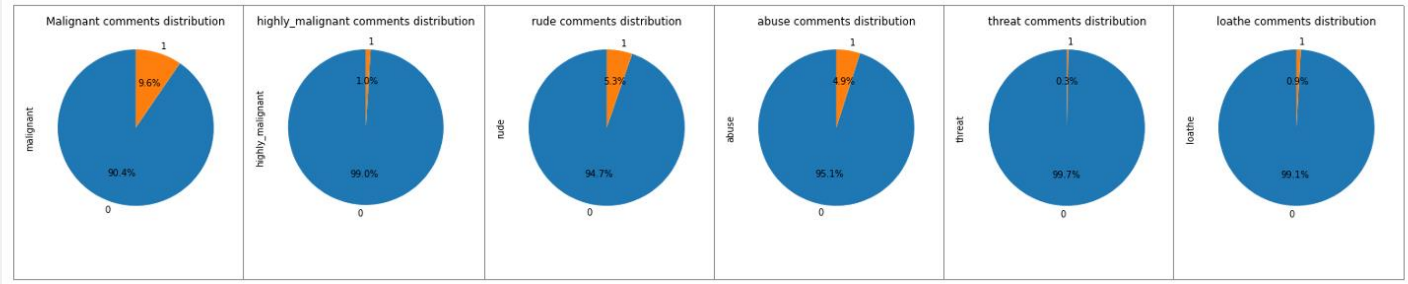
```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
True
```

```
[ ] for i in tqdm(range(len(df))):
        text = df['comment_text'][i]
        document = []
        token = nltk.word_tokenize(text)
        for words in token:
            word = lemmatiser.lemmatize(words,pos="v")
            document.append(word)
        df['comment_text'][i] = ' '.join(document)
```

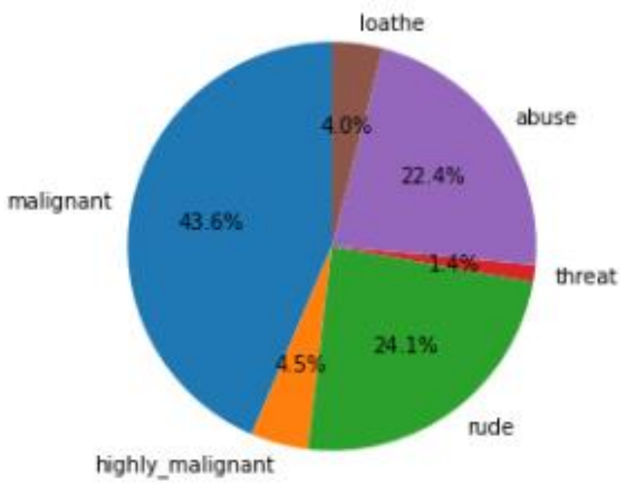
```
100%|██████████| 159571/159571 [05:01<00:00, 528.79it/s]
```

Visualization

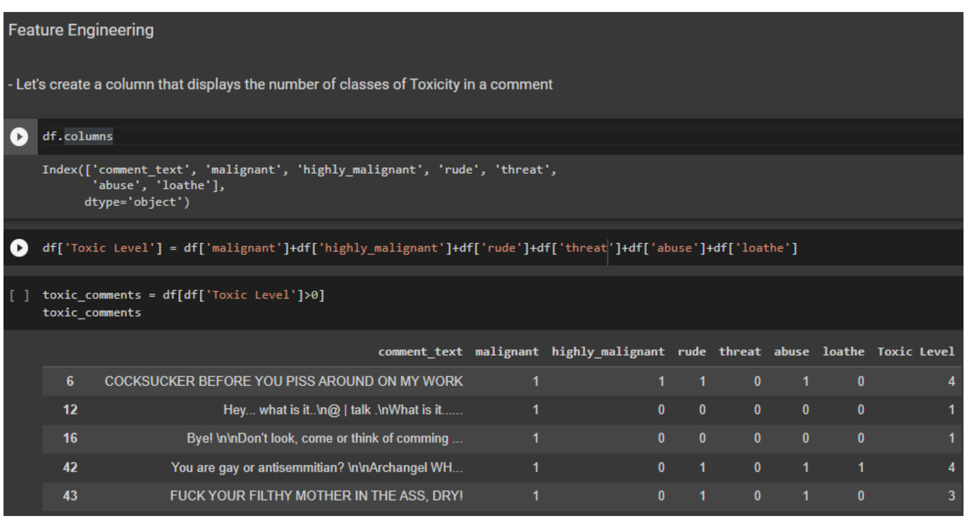
Checking the Distribution of Various Classes.



Checking the distribution of the toxic comments.



Creating a distribution, the represents the Toxic Comments level by adding the different classes together.



Feature Engineering

- Let's create a column that displays the number of classes of Toxicity in a comment

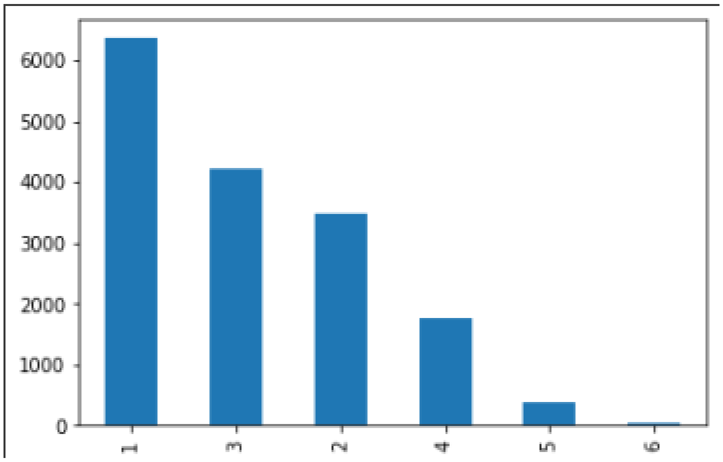
```
df.columns
```

```
Index(['comment_text', 'malignant', 'highly_malignant', 'rude', 'threat',
       'abuse', 'loathe'],
      dtype='object')
```

```
df['Toxic Level'] = df['malignant']+df['highly_malignant']+df['rude']+df['threat']+df['abuse']+df['loathe']
```

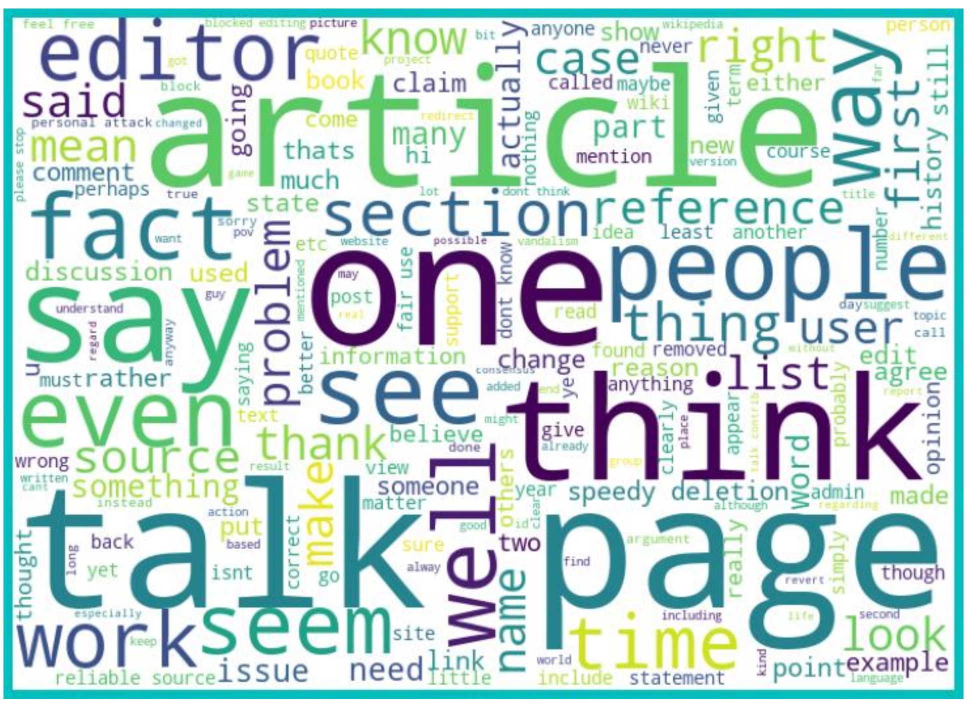
```
toxic_comments = df[df['Toxic Level']>0]
toxic_comments
```

	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	Toxic Level
6	COCKSUCKER BEFORE YOU PISS AROUND ON MY WORK	1	1	1	0	1	0	4
12	Hey... what is it..\n@ | talk .\nWhat is it......	1	0	0	0	0	0	1
16	Bye! \n\nDon't look, come or think of comming ...	1	0	0	0	0	0	1
42	You are gay or antisemmitian? \n\nArchangel WH...	1	0	1	0	1	1	4
43	FUCK YOUR FILTHY MOTHER IN THE ASS, DRY!	1	0	1	0	1	0	3



Word Cloud

Non-Toxic Comments



Malignant Comments

Highly Malignant Comments

Rude Comments

Threatening Comments

Abusive Comments

Loathe Comments

Hardware and Software's Used:

Software requirement:

Google Collab

Libraries and packages used:

- NumPy
- Pandas
- Sklearn
- Seaborn
- Matplotlib
- Nltk

Model/s Development and Evaluation

Problem-solving approach:

There were 3 approaches used with 4 Algorithms in each to test the best performing Model.

- TF-iDF
- TF-iDF(1-2 grams)
- Bag Of Words

Testing of Identified Approaches (Algorithms):

List of algorithms used:

- Logistic Regression
- SGDClassifier
- RandomForest Classifier
- ExtraTree Classifier

Run and evaluate selected models:

---> Using TFiDF

```
[49] from sklearn.feature_extraction.text import TfidfVectorizer
```

```
[50] tfidf = TfidfVectorizer(min_df=5)
     x_train_tfidf = tfidf.fit_transform(x_train)
     x_test_tfidf = tfidf.transform(x_test)
```

```
[ ] x_train_tfidf.shape

    (111615, 24177)
```

```
x_test_tfidf

<47836x24177 sparse matrix of type '<class 'numpy.float64'>'
        with 1187038 stored elements in Compressed Sparse Row format>
```

```
[ ] from sklearn.linear_model import LogisticRegression
    from sklearn.ensemble import ExtraTreesClassifier,RandomForestClassifier
    from sklearn.metrics import accuracy_score,classification_report
```

Final Model

```
from sklearn.linear_model import SGDClassifier

classifier1 = OneVsRestClassifier(SGDClassifier(class_weight='balanced'), n_jobs=-1)
classifier1.fit(x_train_tfidf, y_train)
predictions = classifier1.predict(x_test_tfidf)

accuracy = metrics.accuracy_score(y_test, predictions)
hamming = metrics.hamming_loss(y_test,predictions)
print("Accuracy :",accuracy)
print("AUC :",roc_auc_score(y_test,predictions))
print("Hamming loss ",hamming)
print("\nClassification Report")
print (metrics.classification_report(y_test, predictions))
```

```
Accuracy : 0.8627602642361402
AUC : 0.9022658296770233
Hamming loss  0.03733589765030521

Classification Report
              precision    recall  f1-score   support

           0       0.62      0.84      0.72      4627
           1       0.23      0.91      0.37       486
           2       0.65      0.89      0.75      2510
           3       0.16      0.70      0.26       153
           4       0.51      0.88      0.65      2348
           5       0.16      0.81      0.26       400

   micro avg       0.49      0.86      0.63     10524
   macro avg       0.39      0.84      0.50     10524
weighted avg       0.56      0.86      0.67     10524
 samples avg       0.06      0.08      0.06     10524
```

Key metrices for success in solving problem under consideration:

We have used 3 metrices for our problem

1. Accuracy Score

2. AUC_ROC Score
3. Hamming Loss

Comparing all the parameter, SGDClassifier had the highest AUC_ROC Score and hence was taken to be the final model.

Conclusions

Learning Challenges of the study with respect to Data Science:

1. The Challenges faced we Jupyter Notebook kept crashing.
2. While building a word2vec model, google collab would restart its runtime.
3. An not to forget data cleaning was the biggest challenge.