FLIP ROBO

Malignant Comments Classification

Submitted by:

Anjali Sunny

ACKNOWLEDGMENT

I would like to express my sincere gratitude to Flip Robo Technologies for supporting me throughout the internship and giving me the opportunity to explore the depth of Data Science by providing multiple projects like this, there are multiple people, organizations, youtubers, who guided me in this wonderful journey and few journals which helped me develop my models in this project. I would like to thank following people for the inspiration and help,

INTRODUCTION

- ## Business Problem Framing

  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.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.

- ## Conceptual Background of the Domain Problem

  Classification regarding toxicity has been intensively researched in past few years, largely in the context of social media data where researchers have applied various machine learning systems to try to tackle the problem of toxicity as well as related, so there are various factors affecting the comments and can be related to the emotion, figure of speeches, and sometimes sarcasms which is related to indirect taunting is also rude and as a human it'ssometimes hard for us to distinguish between sarcasms and real appreciation.

- ## Review of Literature

  From the research paper we got to know that the we can get different approaches related to this problem, so this problem is Multilabel classification problem In multi-label classification, data can belong to more than one label simultaneously. For example, in our case a comment may be malignant, rude, threat, abuse and loathe at the same time. It may also happen that the comment is non-toxic and hence does not belong to any of the six labels, so as per research papers we have seen that the effective models for these multilabel classification problems are onevsrest claissifier, Binary Relevance Method, classifier chain Method, Adaptation Algorithm (MLKNN: This is the adapted multi label version of K-nearest neighbours. Similar to this classification algorithm is the BRkNNaClassifier and BRkNNbClassifier which are based on K-Nearest Neighbours Method. Since our problem is somewhat similar to the page categorization problem, this algorithm is expected to give acceptable results. However, the time complexity involved is large and therefore it will be preferable to train it on smaller part of the dataset.), however deep learning and Bidirectional LSTM has provided results with 96% accuracy.

- ## Motivation for the Problem Undertaken

  This project was highly motivated project as it includes the real time problem of analysing toxic behaviour and providing us opportunity to explore a bit and contribute our efforts against cyberbullying which has been proven critical as this can take a toll on anyone and affect them mentally leading to depression, mental illness, self-hatred and suicidal thoughts.

Analytical Problem Framing

- Data Sources and their formats

The data set contains the training set, which has approximately 1,59,000 samples and the test set which contains nearly 1,53,000 samples. All the data samples contain 8 fields which includes 'Id', 'Comments', 'Malignant', 'Highly malignant', 'Rude', 'Threat', 'Abuse' and 'Loathe'.
The label can be either 0 or 1, where 0 denotes a NO while 1 denotes a YES. There are various comments which have multiple labels. The first attribute is a unique ID associated with each comment.
The data set includes:
-**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.
-**ID:**It includes unique Ids associated with each comment text given.
-**Comment text:** This column contains the comments extracted from various social media platforms.

```
<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
```

- ## Data Pre-processing Done

We have applied various methods for data preprocessing methods in this project interestingly we use wordNet, lemmatizer and porterStemmer to clean the words and removed special characters using Regexp Tokenizer and filter the words by removing stop words and then used lemmatizes and joined and return the filtered words, Used TFIDF vectorizer to convert those text into vectors, and split the data and into test and train and trained various Machine learning algorithms, but we also explored and implemented **keras.preprocessing.text** and **keras.preprocessing** as this class allows to vectorize a text corpus, by turning each text into either a sequence of integers (each integer being the index of a token in a dictionary) or into a vector where the coefficient for each token could be binary, based on word count, based on tf-idf

```
lemmatizer = WordNetLemmatizer()
stemmer = PorterStemmer()
def clean_text(text):
    text=str(text)
    text = text.lower()
    cleanr = re.compile('<.*?>')
    cleantext = re.sub(cleanr, '', text)
    rem_num = re.sub('[0-9]+', '', text)
    tokenizer = RegexpTokenizer(r'\w+')
    tokens = tokenizer.tokenize(rem_num)
    filtered_words = [w for w in tokens if len(w) > 2 if not w in stopwords.words('english')]
    stem_words=[stemmer.stem(w) for w in filtered_words]
    lemma_words=[lemmatizer.lemmatize(w) for w in stem_words]
    return " ".join(filtered_words)
df["comment_text"] = df["comment_text"].apply(lambda x: clean_text(x))
```

```
from keras.preprocessing.text import Tokenizer
tokenizer = Tokenizer(num_words=20000)
tokenizer.fit_on_texts(list(comments))
```

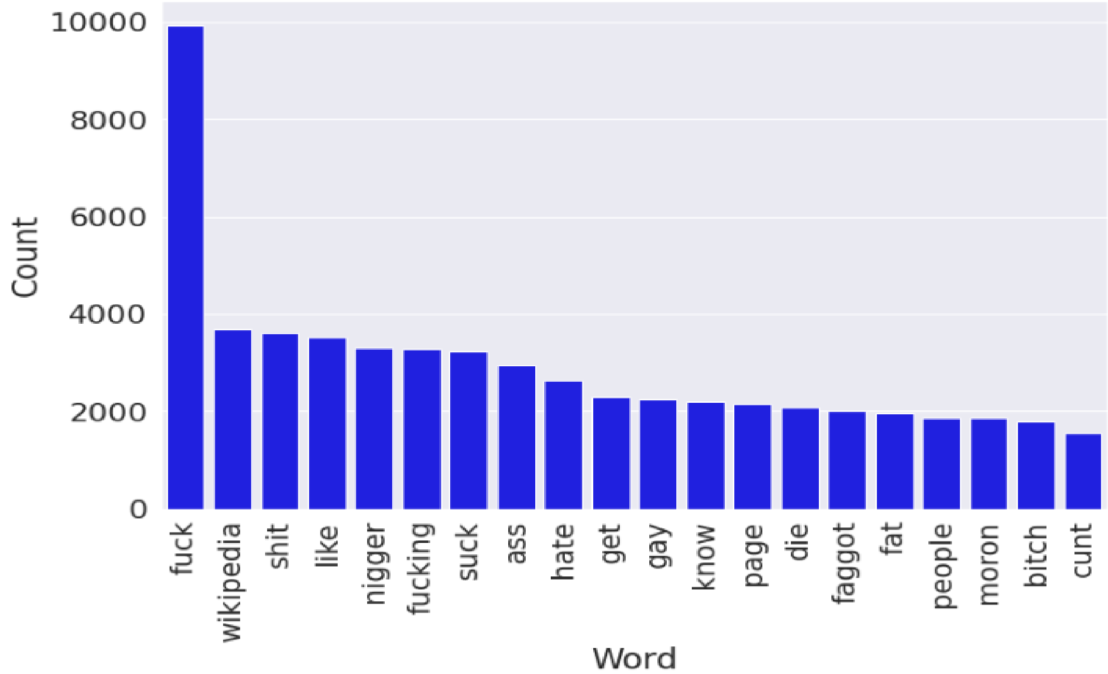
```
from keras.preprocessing import text, sequence

seq = tokenizer.texts_to_sequences(comments)
pad = sequence.pad_sequences(seq, maxlen=100)
```

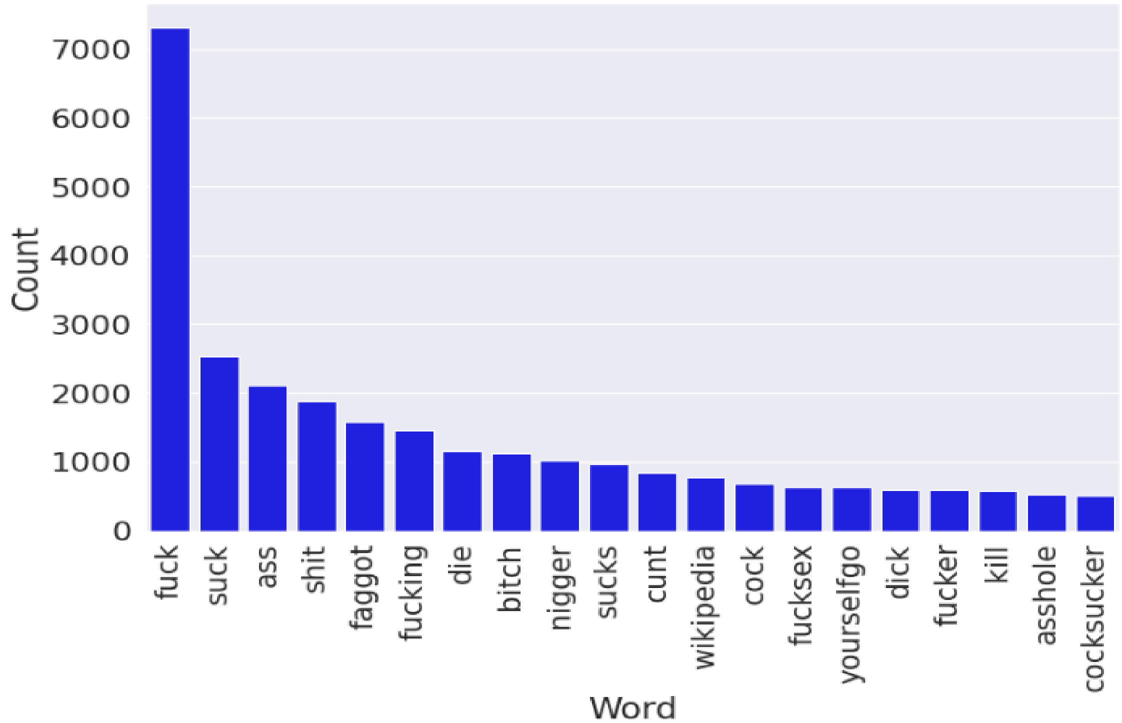
- ## Data Inputs- Logic- Output Relationships

For this data's input and output logic we will analyse words frequency for each label, so that we can get the which most 20 frequent words were used on that label categories.

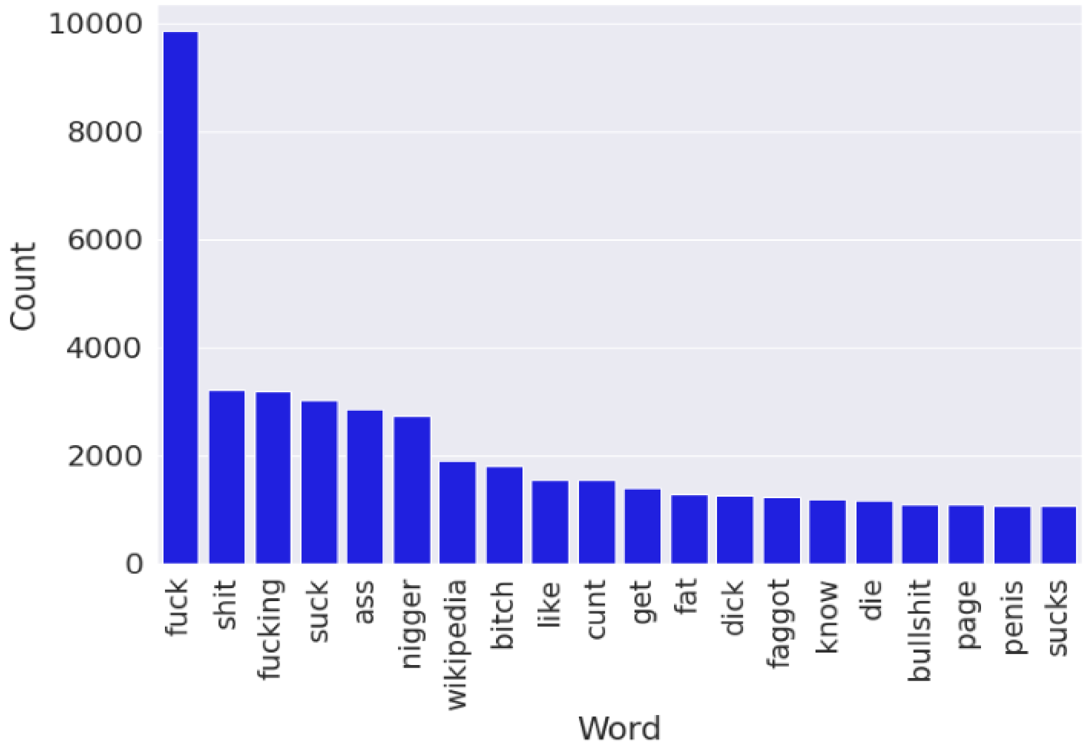
```
counter(data[data["malignant"] == 1], "comment_text", 20)
```



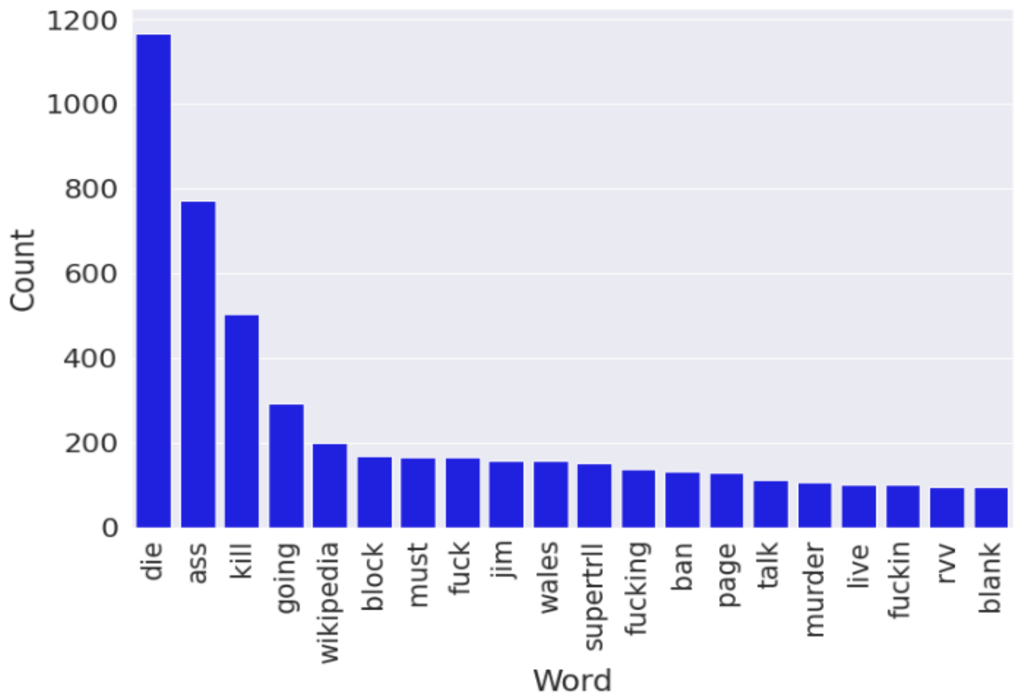

```
counter(data[data["highly_malignant"] == 1], "comment_text", 20)
```



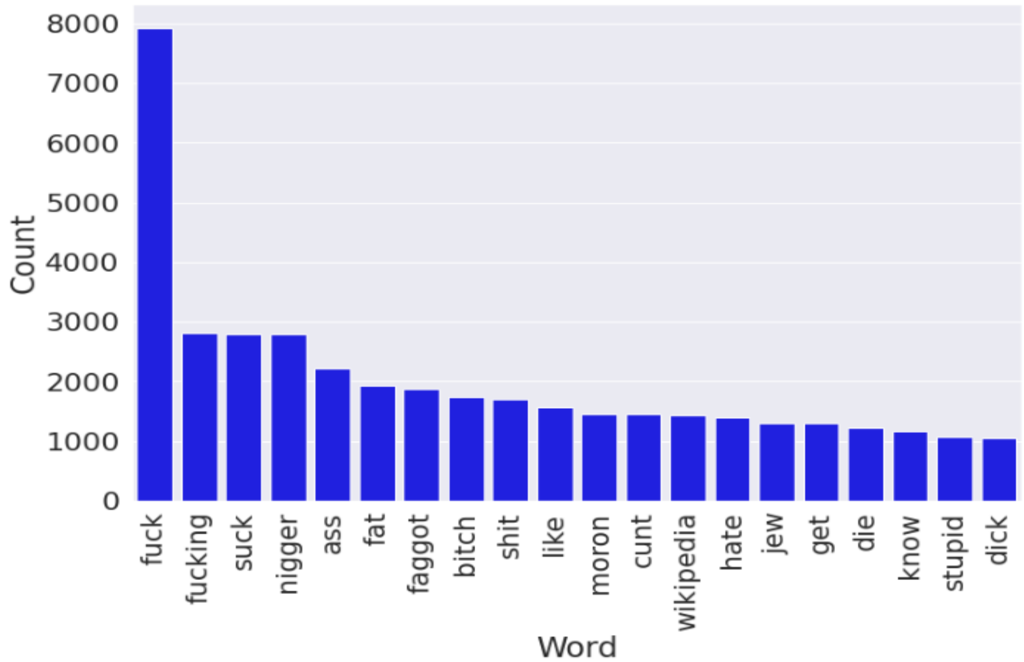
```
counter(data[data["rude"] == 1], "comment_text", 20)
```



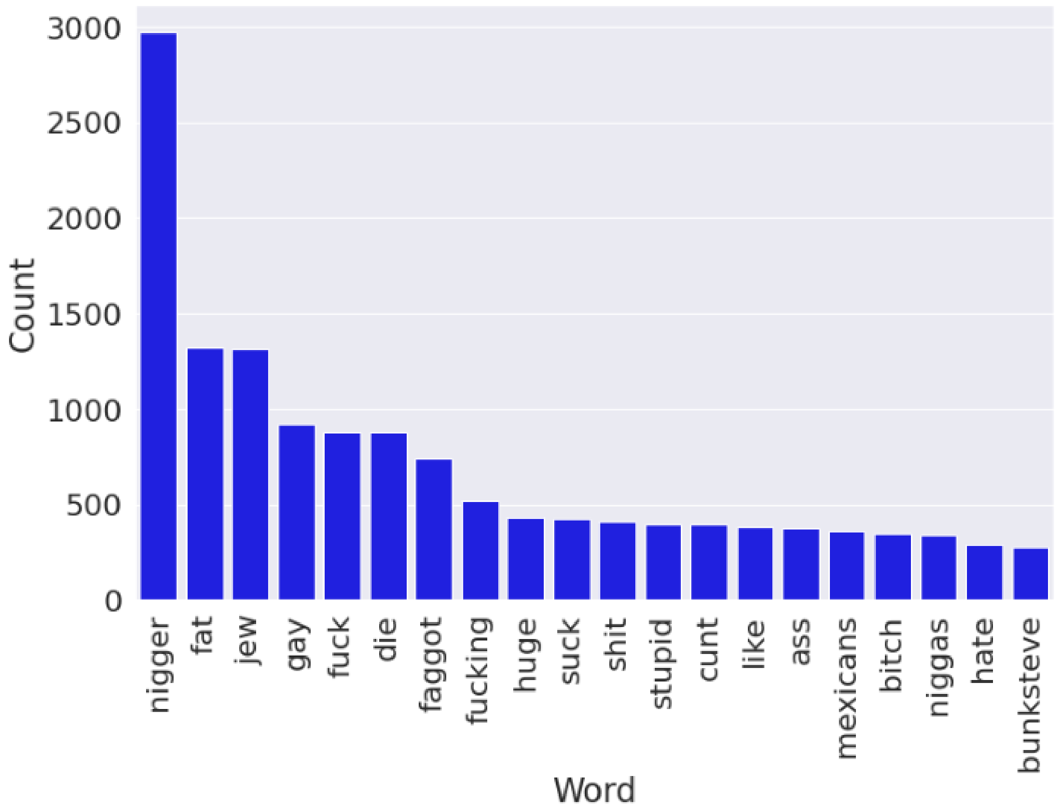

```
counter(data[data["threat"] == 1], "comment_text", 20)
```



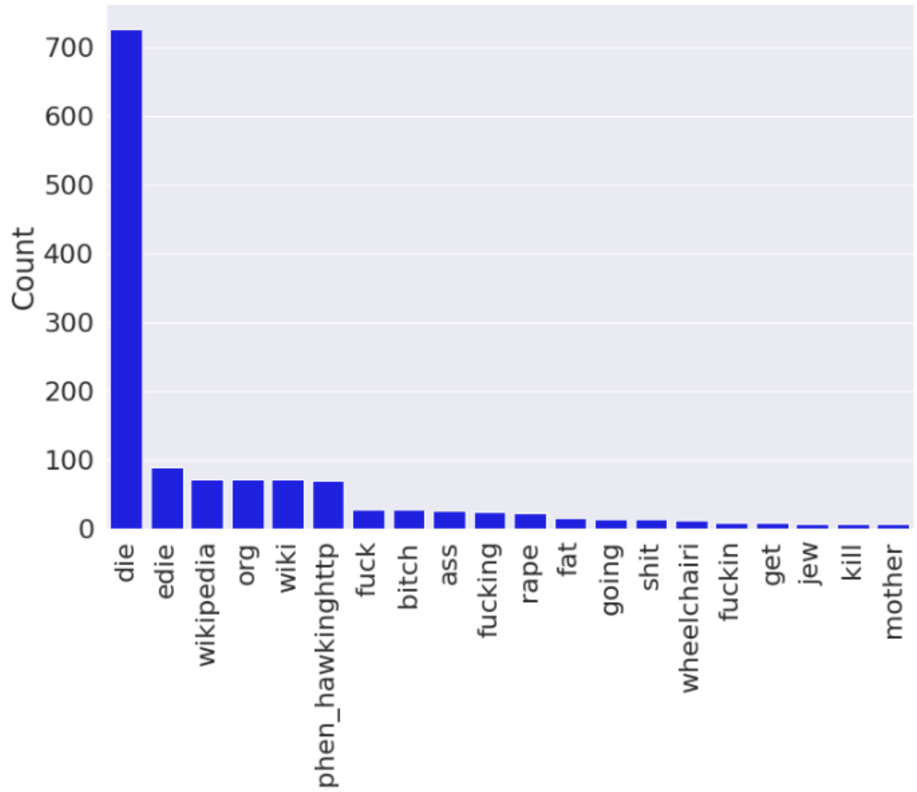
```
counter(data[data["abuse"] == 1], "comment_text", 20)
```



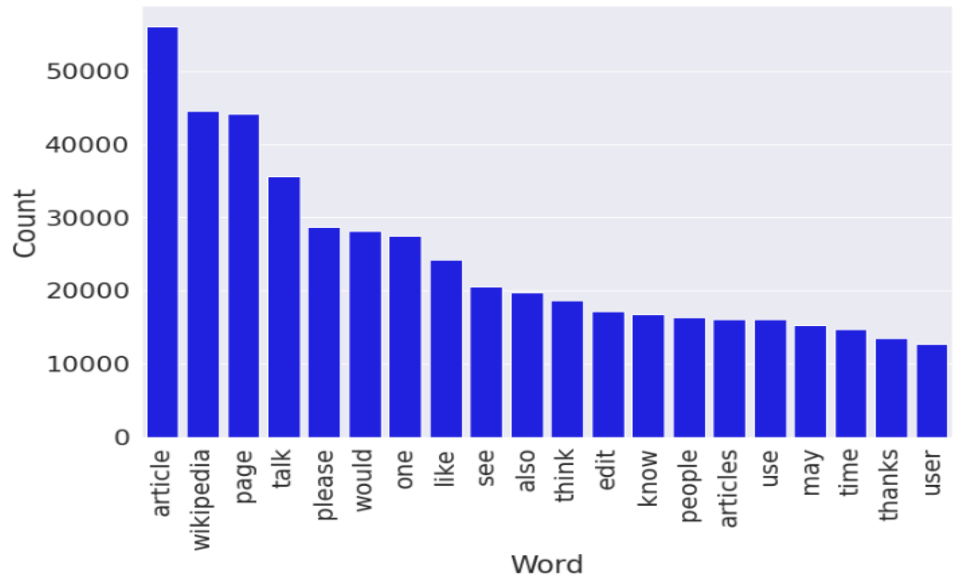

```
counter(data[data["loathe"] == 1], "comment_text", 20)
```



```
counter(data[(data["malignant"] == 1) & (data["highly_malignant"] == 1) & (data["rude"] == 1) & (data["threat"] == 1) & (data["ab
```




```
counter(data[(data["malignant"] == 0) & (data["highly_malignant"] == 0) & (data["rude"] == 0) & (data["threat"] == 0) & (data["ab
```



So from above sets of visualization we can see the most frequent words because of which that particular sentence are classified, and we can also see words which were categorized as ("malignant", "highly_malignant", "rude", "threat", "abuse", “loathe"), and words which were completely neutral.

- ## State the set of assumptions (if any) related to the problem under consideration

Only assumptions which were taken related to the problem was that we dropped the id column as it had high chance of overfitting as our models could have memories the results based on id, however shuffle was used but still we couldn’t take risk.

- ## Hardware and Software Requirements and Tools Used
- Hardware: 8GB RAM, 64-bit, i7 processor, and 12 GB RAM on Googlecolab(with TPU as runtime processing)
- Software: Excel, Jupyter Notebook, python 3.6.

Library Used:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
from nltk.corpus import wordnet
import string
import nltk
import ast
import re
from nltk import pos_tag
from nltk.corpus import stopwords
from nltk.tokenize import WhitespaceTokenizer
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.stem import WordNetLemmatizer,PorterStemmer
from nltk.tokenize import RegexpTokenizer
from nltk.corpus import wordnet
from nltk.corpus import sentiwordnet
nltk.download('stopwords')
nltk.download('wordnet')
from keras.preprocessing import text, sequence
from keras.preprocessing.text import Tokenizer
from keras.layers import Dense, Input, LSTM, Embedding, Dropout, Activation
from keras.layers import Bidirectional, GlobalMaxPool1D
from keras.models import Model
from keras import initializers, regularizers, constraints, optimizers, layers
from keras.callbacks import EarlyStopping, ModelCheckpoint
```

Model/s Development and Evaluation

- ## Identification of possible problem-solving approaches (methods)

We also found that the dataset was highly imbalanced as class 1 had 16225, and class 0 had 143346, so we down sampled class 0 to 16225 to make the data balanced using resample.

```
print("class0 count:",df_class0.shape)
print("class1 count:",df_class1.shape)
```

```
class0 count: (143346, 8)
class1 count: (16225, 8)
```

```
from sklearn.utils import resample
df_class0_downsampled = resample(df_class0,
                                 replace=False,
                                 n_samples=16225,
                                 random_state=42)
```

- ## Testing of Identified Approaches (Algorithms)

```
In [44]:
LogReg_pipeline = Pipeline([('clf', OneVsRestClassifier(LogisticRegression(solver='sag'), n_jobs=-1)),])
LogReg_pipeline.fit(x_train, y_train)
logisticRegression_prediction = LogReg_pipeline.predict(x_test)
calculate_metrics(y_test, logisticRegression_prediction)
calculate_kfold(LogReg_pipeline)
```

```
In [45]:
linsvc_pipeline = Pipeline([('lins', OneVsRestClassifier(LinearSVC(random_state=42))),])
linsvc_pipeline.fit(x_train, y_train)
linsvc_pipeline_prediction = linsvc_pipeline.predict(x_test)
calculate_metrics(y_test, linsvc_pipeline_prediction)
calculate_kfold(linsvc_pipeline)
```

```
In [47]:
from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import SGD
from keras.metrics import binary_accuracy

```

```
In [48]:
def get_model(n_inputs, n_outputs):
    model = Sequential()
    model.add(Dense(20, input_dim=n_inputs, kernel_initializer='he_uniform', activation='relu'))
    model.add(Dense(n_outputs, activation='sigmoid'))
    model.compile(loss='binary_crossentropy', optimizer='adam')
    return model
```

```
In [50]:
results = pd.DataFrame(data = {'accuracy':accuracy, 'precision': precision ,
                               'recall': recall,'f1_score': f1_score,
                               'kfold_mean': kfold_mean,'kfold_min': kfold_min,'kfold_max': kfold_max},
                      index = ['OnevsRest(LR)', 'OnevsRest(LSVC)'])
results
```

```
def get_model(n_inputs, n_outputs):
    model = Sequential()
    model.add(Dense(20, input_dim=n_inputs, kernel_initializer='he_uniform', activation='relu'))
    model.add(Dense(n_outputs, activation='sigmoid'))
    model.compile(loss='binary_crossentropy', optimizer='adam',metrics=['accuracy'])
    return model
```

```
n_inputs, n_outputs = x.shape[1], y.shape[1]
model_deep = get_model(n_inputs, n_outputs)
model_deep.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=10,batch_size=64,verbose=1)
```

```
def model_add():
    inputs = Input(shape=(100, ))
    x = Embedding(20000, 128)(inputs)
    x = Bidirectional(LSTM(80))(x)
    x = Dropout(0.1)(x)
    x = Dense(80, activation="relu")(x)
    x = Dropout(0.1)(x)
    outputs = Dense(6, activation="sigmoid")(x)
    model = Model(inputs=inputs, outputs=outputs)
    model.compile(loss='binary_crossentropy',
                  optimizer='adam',
                  metrics=['accuracy'])
    return model
model = model_add()
print(model.summary())
```

From the above we can see that the we have tried various methods and utilized pipelines in order to achieve a robust model, so we used OneVSRestclassifier on logistic regression and linear SVM, and used deep learning models and bidirectional LSTMwith embedding and achieve a better result.

- Key Metrics for success in solving problem under consideration

```
from sklearn import metrics
from sklearn.model_selection import cross_val_score
accuracy = []
precision = []
recall = []
f1_score = []
rocscore=[]
def calculate_metrics(y_test, y_pred):
    acc = metrics.accuracy_score(y_true = y_test, y_pred = y_pred)
    pre = metrics.precision_score(y_true = y_test, y_pred = y_pred,pos_label='positive',average='macro')
    rec = metrics.recall_score(y_true = y_test, y_pred = y_pred,pos_label='positive',average='macro')
    f1 = metrics.f1_score(y_true = y_test, y_pred = y_pred,pos_label='positive',average='macro')


    accuracy.append(acc)
    precision.append(pre)
    recall.append(rec)
    f1_score.append(f1)


kfold_min = []
kfold_mean = []
kfold_max = []
def calculate_kfold(estimator):
    accuracies = cross_val_score(estimator, x, y, cv = 20)
    kfold_min.append(accuracies.min())
    kfold_mean.append(accuracies.mean())
    kfold_max.append(accuracies.max())
```

From the above code snippet we can see the how we measured the metrices evaluation for the models which we used as OneVSRest classifier on logistic regression and linear SVC using pipelines and important metrices we needed to evaluate as it was multilabel classification model we need to use average as micro or macro but we used macro.

	accuracy	precision	recall	f1_score	kfold_mean	kfold_min	kfold_max
OnevsRest(LR)	0.626605	0.777604	0.294939	0.372374	0.614693	0.283600	0.943931
OnevsRest(LSVC)	0.647663	0.751523	0.387939	0.478026	0.642095	0.321208	0.951941

- Visualizations

  Malignanat

highly_malignant

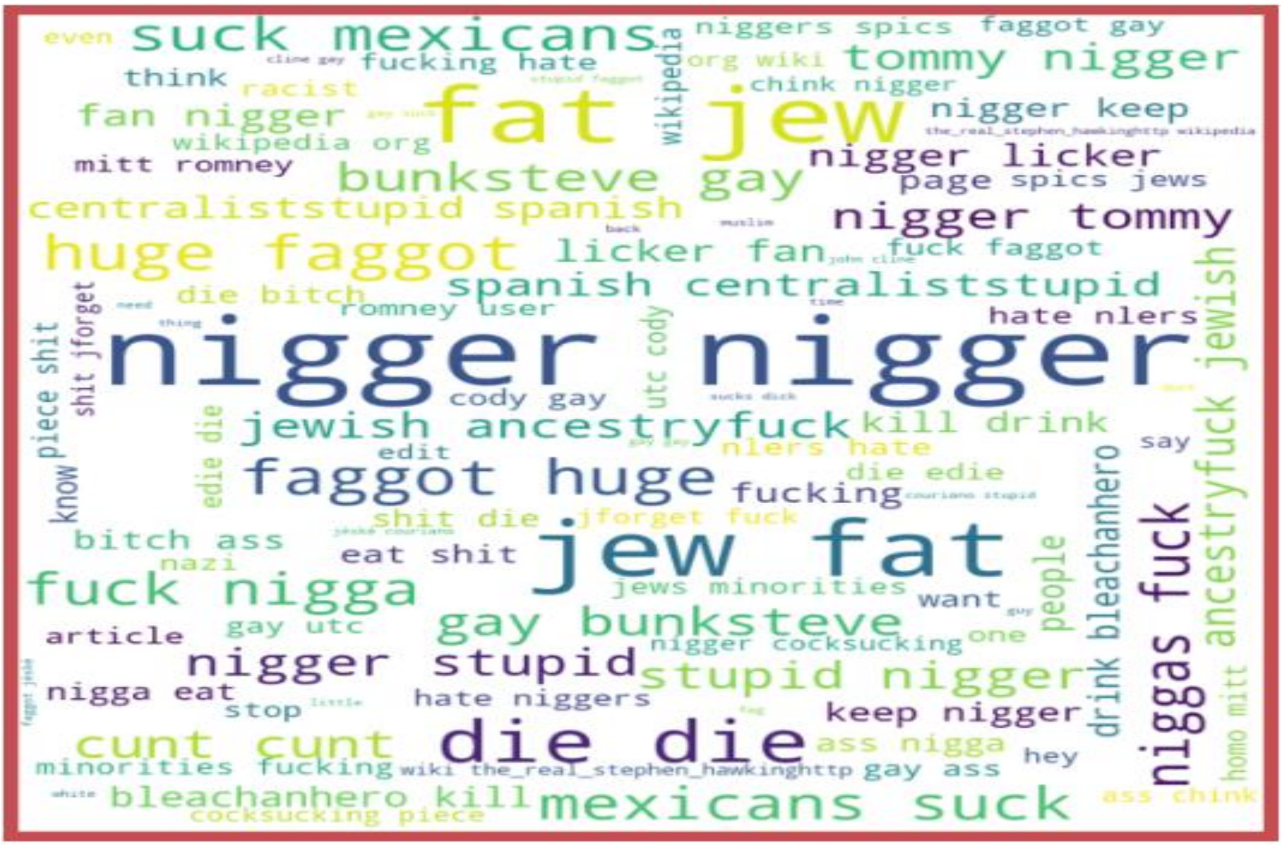
Rude


suck mexicans
fuck bitches
rape anal
think
stop
fucksex fucksex
sucks cock
talk page
buttsecks buttsecks
mothjer fucker
nigger nigger
eat shit
anal rape
niggas fuck
dog fuck
cocksucker mothjer
time
bullshit bullshit
nipple nipple
life
faggots faggots
jew fat
assad hanibalyou
hanibalyou bastard
cunt damn
bollocks bollocks
see
shut fuck
faggot huge
fuck ass
bitches fuck
fuck fuck
edit
say
stupid
freedom freedom
fat jew
fuck yourselfgo
wikipedia
piece shit
fuck bitch
mexicans suck
cock suck
ass ass
make
page
pro assad
sucks dick
dickhead dickhead
fuck nigga
twat twat
fuck shut
know
idiot idiot
fuck dog
penis penis
suck suck
please
one
bastard pro
yourselfgo fuck
huge faggot
hate hate
bitch fuck
boobs boobs
damn cunt
people
ass fuck
thing way
shit shit
small penis
really
fucker cocksucker
offfuck offfuck
poop poop
even
want
article
fucking
blocked
suck cock
die die
know fggt
chester marcolfuck
suck dick
bastered bastered

Threat

Abuse

Loathe



"malignant", "highly_malignant", "rude", "threat", "abuse","loathe"

Neutral(All Clean)

- Interpretation of the Results

From the above we visualization we can see that there are multiple words which are categories as multiple labels and we can see most frequent words which were used are on multiple labels, and we can also see most frequent words which were labelled as neutral.

CONCLUSION

- Key Findings and Conclusions of the Study

So, the key findings and conclusion we got from the whole analysis that there are few words which are focus on the same categories as the comment start going from bad to worse, and we keep them categories in multiple labels, as only because of this our Bidirectional LSTM using embedding worked so well, as on based on few words we can classify the whole comments.

- Learning Outcomes of the Study in respect of Data Science

There where al lot of learning outcomes we were able to see that how efficient and less time consuming keras text processing library can clean and vectorizes the comments using TFIDF, and also wanted to use all the binary relevance and adaptation algorithm, but due to memory limitation had to comment out those algorithm as kernel was continues to die of both local machine and googlecolab even tried 15 different times eliminating one algo at a time, just will keep working on that as I am very curious and keen to see the results of all other methods even using Naïve ByesmultinomialNB expecting a very good results, and have future work application of combining TF-IDF with sentiment features.