# *MALIGNANT COMMENTS CLASSIFICATION PROJECT*

**Submitted by:**
**NIPAM GOGOI**

## *ACKNOWLEDGMENT*

I would like to thank Flip Robo Technologies for providing me with the opportunity to work on this project from which I have learned a lot. I am also grateful to Miss Khushboo Garg for her constant guidance and support.

# 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. 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.

Therefore, 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.

## CONCEPTUAL BACKGROUND OF THE DOMAIN PROBLEM

Internet is one of the important inventions and a large number of persons are its users. These persons use this for different purposes. There are different social media platforms that are accessible to these users. Any user can make a post or spread the news through these online platforms. These platforms do not verify the users or their posts. So some of the users try to spread online hate. 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.

## *REVIEW OF LITERATURE*

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 inoffensive, 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.

There is a difference between the traditional and very famous multi-class classification, and the one which we will be using, which is the multi-label classification. In a multi-class classification, each instance is classified into one of three or more classes, whereas, in a multi-label classification, multiple labels (such as – toxic, severe-toxic, obscene, threat, insult or identity- hate) are to be predicted for the same instance.
Multiple ways are there to approach this classification problem. It can be done using –
Multi-label methods which belong to the problem transformation category: Label Power Set (LP), Binary Relevance (BR), BR+, and classifier chain.

Base and adapted algorithms like: J48 (Decision Tree), Naïve Bayes, k-Nearest-Neighbor (KNN), SMO (Support Vector Machines), and, BP-MLL neural networks.
Further, out of the total dataset used for experimenting these algorithms, 70% was used for training and 30% was used for testing. Each testing dataset was labelled and thus for each algorithm using the predictions and labels, calculation of metric such as hamming-loss, accuracy and log-loss was done. The final results have been complied on the basis of values obtained by algorithmic models in hamming-loss and log-loss combined.

## *ANALYTICAL PROBLEM FRAMING*

### Dataset description

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.

## *MODEL DEVELOPMENT AND EVALUATION*

### Exploratory Data Analysis

```python
train_df = pd.read_csv("train.csv")
```

```python
train_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

- There are 159571 rows and 8 columns in the entire dataset.

```python
train_df.isnull().any()    #checking if any null values
```

```
id                  False
comment_text        False
malignant           False
highly_malignant    False
rude                False
threat              False
abuse               False
loathe              False
dtype: bool
```

- Na values does not exist in the Dataset.

```
malignant = list(train_df.malignant)
highly_malignant = list(train_df.highly_malignant)
rude = list(train_df.rude)
threat= list(train_df.threat)
abuse = list(train_df.abuse)
loathe = list(train_df.loathe)
```

```
# Adding all the outputs. If the sum is 0 indicating non-offensive comments and sum > 0 indicating malignant comments
Target = []
for i,j,k,l,m,n in zip(malignant,highly_malignant,rude,threat,abuse,loathe):
    Target.append(i+j+k+l+m+n)
```

- Collecting all different types of outputs in a list

```
from collections import Counter
print(dict(Counter(Target)))
```

```
{0: 143346, 4: 1760, 1: 6360, 3: 4209, 2: 3480, 5: 385, 6: 31}
```

Here we can see that the dataset contains mostly outputs with 0 i.e. non-malignant comments and if Target > 1 indicating that the comment falls in more than one category such as rude, malignant, loathe, abuse etc.

```
#Changing Target with value more than 1 into 1 so that we have a binary classification problem:- malignant or non-malignant
Target2= []
for i in Target:
    if i == 0:
        Target2.append(0)
    else:
        Target2.append(1)
```

```
print(dict(Counter(Target2)))
```

```
{0: 143346, 1: 16225}
```

Here Label "1" are offensive comments and "0" are non-offensive.

- Defining function for word clouds.

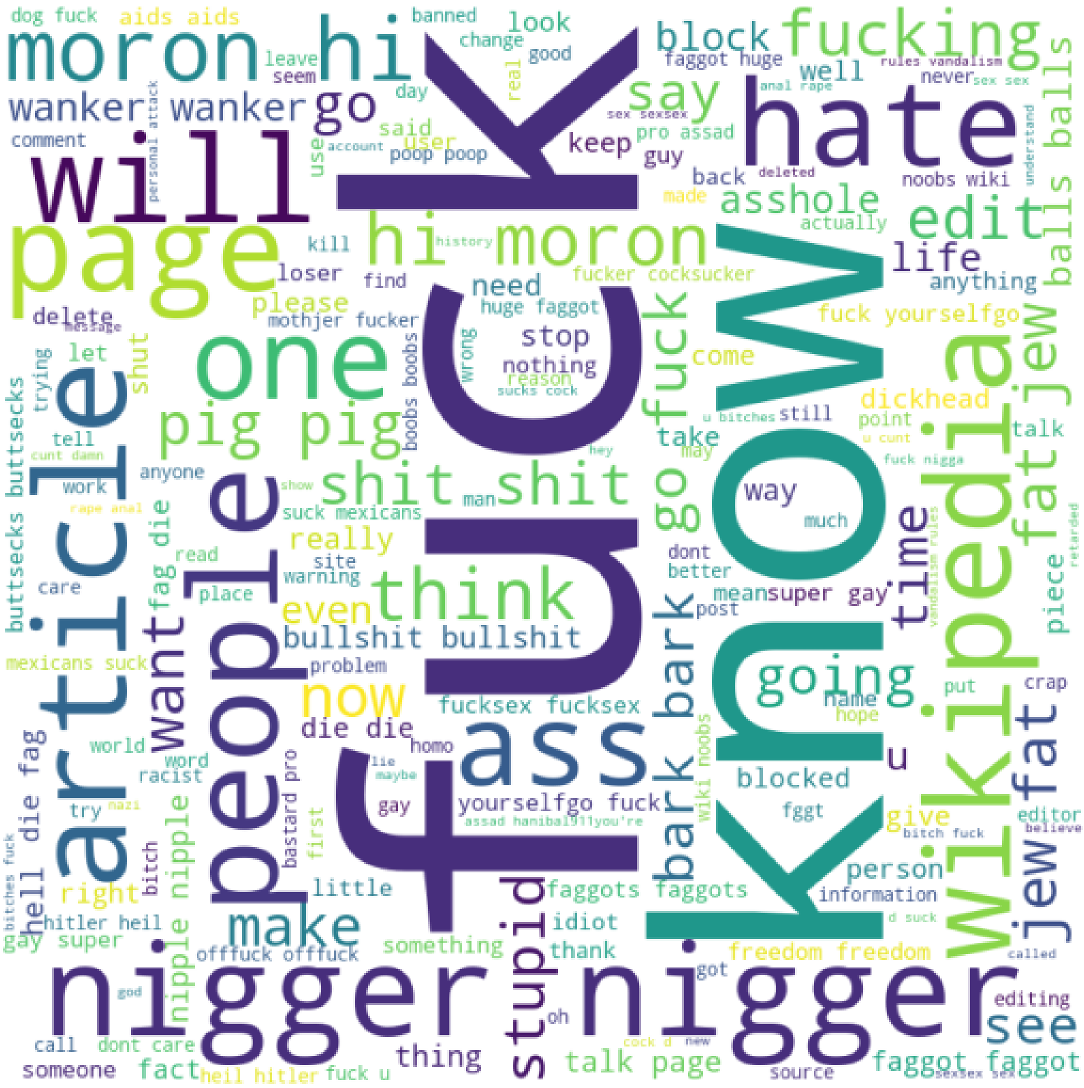
```
# Defining Function for Word Clouds
def Word_Cloud(str_List):
    comment_words = ''
    stopwords = set(STOPWORDS)

    # iterate through the csv file
    for val in str_List:

        # typecaste each val to string
        val = str(val)

        # split the value
        tokens = val.split()

        # Converts each token into lowercase
        for i in range(len(tokens)):
            tokens[i] = tokens[i].lower()

        comment_words += " ".join(tokens)+" "


    wordcloud = WordCloud(width = 800, height = 800,
                    background_color ='white',
                    stopwords = stopwords,
                    min_font_size = 10).generate(comment_words)

    # plot the WordCloud image
    plt.figure(figsize = (8, 8), facecolor = None)
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.tight_layout(pad = 0)

    plt.show()
```

- Plotting word cloud function.

```
#Plotting Word Cloud using the function defined for a sample of non-malignant comments with Target = 0
Word_Cloud(list(train_df[train_df.Target == 0].sample(n=15000).comment_text))
#Taking only 15000 samples for the plot since Target=0 has 143346 records which will take huge processing time for the plot
```

- Plotting word cloud function for malignant comments.

```python
# Plotting the Word-Cloud for all the malignant comments
Word_Cloud(list(train_df[train_df.Target != 0].comment_text))
```




- Plotting word cloud for all types of malignant comments.

```python
# Plotting Word Cloud for all types of Offensive comments
for cols in Malignant_Columns:
    print("Comment_Type :",str(cols).upper(),"    Count :",len(train_df[train_df[cols] != 0]))
    Word_Cloud(list(train_df[train_df[cols] != 0].comment_text))
    print("#####################################################")
    print()
```

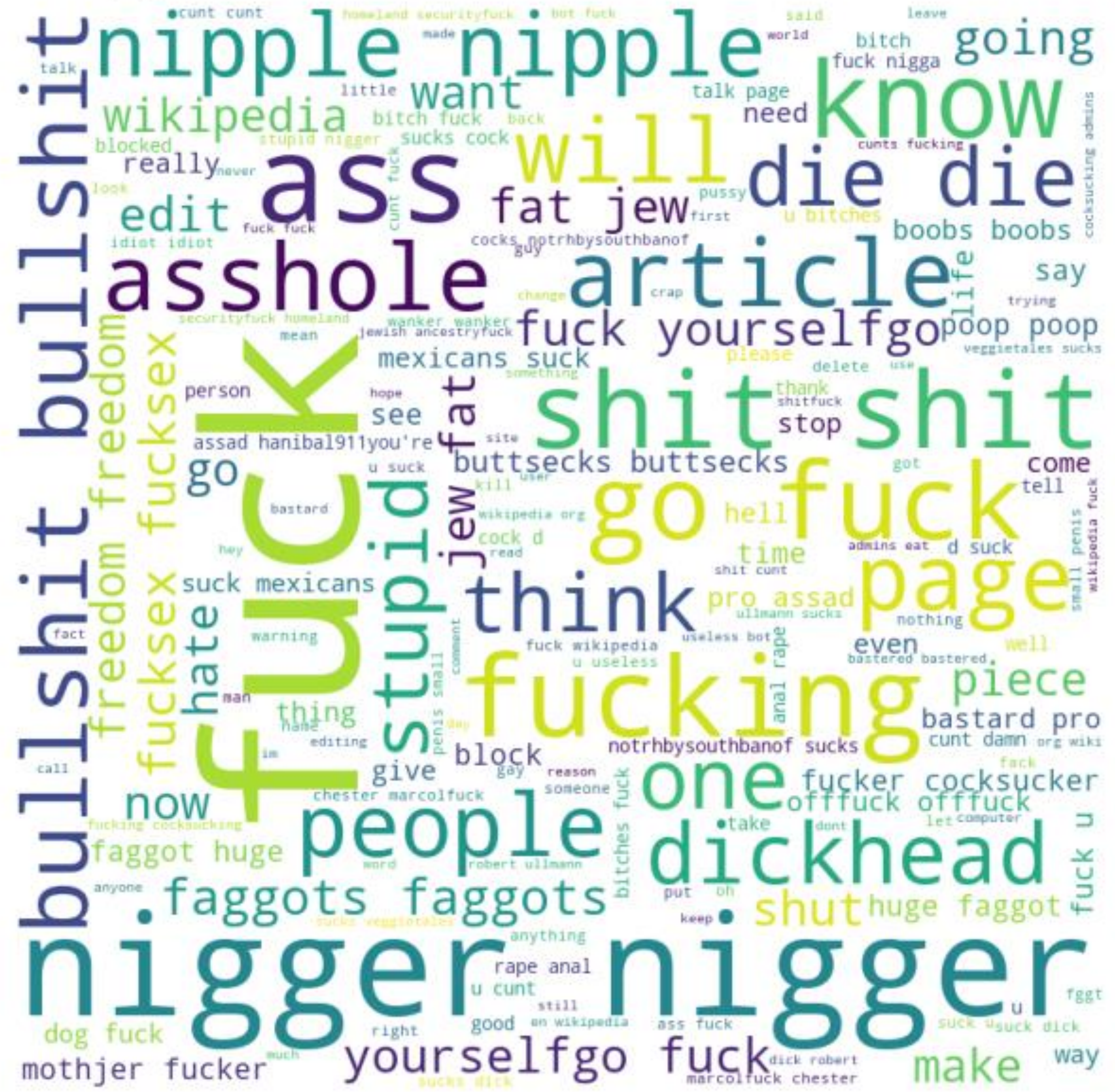
```
Comment_Type : MALIGNANT     Count : 15294
```


```
#####################################################
```

Comment_Type : HIGHLY_MALIGNANT     Count : 1595


################################################################

Comment_Type : RUDE     Count : 8449



Comment_Type : LOATHE      Count : 1405


*KEY METRICS FOR SUCCESS IN SOLVING PROBLEM UNDER CONSIDERATION*

- When it comes to the evaluation of a data science model's performance, sometimes accuracy may not be the best indicator.
- So, we have used f1 score as well as recall, precision to check the performance of the models.

## *MODEL TRAINING*

- Testing models for Malignant comments prediction

```
# Defining Input data and Output to be predicted
X = train_df.comment_text
y = train_df.Target
```

```
# Splitting the train_df into training and validation dataset
X_train,X_val,y_train,y_val=train_test_split(X,y,test_size=0.2,random_state=88)
```

Using TF-IDF to convert text data into numerical format for Machine Learning Models

```
vectorizer = TfidfVectorizer(min_df =1,stop_words='english',use_idf=True,analyzer='word',
                             ngram_range=(1,1),max_features=15000)
x_train = vectorizer.fit_transform(X_train)
x_val  = vectorizer.transform(X_val)
```

### Logistic Regression Model

```
logisticRegr = LogisticRegression(solver='liblinear',class_weight='balanced',random_state=5,tol=0.001,max_iter=1000)
logisticRegr.fit(x_train, y_train)
```

```
LogisticRegression(class_weight='balanced', max_iter=1000, random_state=5,
                   solver='liblinear', tol=0.001)
```

```
prediction = logisticRegr.predict(x_val)
```

```
print('\n','CONFUSION MATRIX','\n',confusion_matrix(y_val, prediction))
print('\n','ACCURACY','\n',accuracy_score(y_val, prediction))
print('\n','REPORT','\n',classification_report(y_val,prediction))
```

```
 CONFUSION MATRIX
 [[27295  1436]
 [  425  2759]]

 ACCURACY
 0.9416888610371299

 REPORT
               precision    recall  f1-score   support

           0       0.98      0.95      0.97     28731
           1       0.66      0.87      0.75      3184

    accuracy                           0.94     31915
   macro avg       0.82      0.91      0.86     31915
weighted avg       0.95      0.94      0.95     31915
```

- Four ML models were used to train the dataset using Sklearn, out of which the best was the Random Forest Model.

Random Forest Classifier

```
rand = RandomForestClassifier(n_estimators=100,criterion='entropy',max_features=None,class_weight='balanced')
rand.fit(x_train, y_train)
```

```
RandomForestClassifier(class_weight='balanced', criterion='entropy',
                       max_features=None)
```

```
prediction3 = rand.predict(x_val)
```

```
print('\n','CONFUSION MATRIX','\n',confusion_matrix(y_val, prediction3))
print('\n','ACCURACY','\n',accuracy_score(y_val, prediction3))
print('\n','REPORT','\n',classification_report(y_val,prediction3))
```

```
 CONFUSION MATRIX
 [[27881   850]
 [  859  2325]]

 ACCURACY
 0.9464515118282939

 REPORT
```

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.97 | 0.97 | 0.97 | 28731 |
| 1 | 0.73 | 0.73 | 0.73 | 3184 |
| accuracy | | | 0.95 | 31915 |
| macro avg | 0.85 | 0.85 | 0.85 | 31915 |
| weighted avg | 0.95 | 0.95 | 0.95 | 31915 |

Since it is an Unbalanced Dataset even though Random Forest Model has showed highest Accuracy the Logistic Regression Model is better with a higher F1 score. So we can take this model as the Final Model

- Checking for the performance of the model

```
# Plotting Word Cloud for all the comments predicted as Malignant
print("Count of Malignant Comments in the Test Dataset :",len(test_df[test_df["Label"] == "Malignant"]))
Word_Cloud(list(test_df[test_df["Label"] == "Malignant"].comment_text))
```

```
Count of Malignant Comments in the Test Dataset : 43305
```

## CONCLUSION

- Converted the problem into a binary classification problem. If the value in any of the columns: malignant, loathe, rude etc == 1 clubbed them together into one group and for rows with values == 0 counted them as non-malignent.
- Plotted word cloud for all the different types: 'malignant', 'highly_malignant', 'rude', 'threat','abuse', 'loathe'.
- Logistic Regression was the best model with 94% accuracy and best f1 score.