Malignant Comments Classifier Project

Submitted by:

Moncy Kurien

ACKNOWLEDGMENT

Referred the following articles:

1. https://towardsdatascience.com/journey-to-the-center-of-multi-label-classification-384c40229bff
2. https://www.kaggle.com/surekharamireddy/malignant-comment-classification-starter-notebook
3. https://medium.com/@nupurbaghel/toxic-comment-classification-f6e075c3487a

INTRODUCTION

- Business Problem Framing

The internet comments are quickly filling with hatred and abuse comments. 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. Our goal is to build a prototype of online hate and abuse comment classifier which can be 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

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. 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.

- Motivation for the Problem Undertaken

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. Although researchers have found that hate is a problem across multiple platforms, there is a lack of models for online hate detection.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

This is a multi-label classification problem. Any given record can have more than one class label associated to it. Hence, the Machine Learning Algorithms need to be used slightly differently. We are also dealing with text inputs. The text inputs need to be cleaned and converted as vectors.

- Data Sources and their formats

The data used for training and testing are the internet comments that contain text comments and are labelled with multiple labels.

The train dataset contains: id, comment_text, malignant, highly_malignant, rude, threat, abuse and loathe.

The test dataset contains: id and comment_text.

The dataset contains 2 columns:

1. 'id', an object data type – contains the unique id of each comment text.
2. 'comment_text', an object data type – Contains the actual original comment text.
3. Target Columns: 'malignant', 'highly_malignant', 'rude', 'threat', 'abuse', 'loathe' are binary variables – Indicates the type of toxicity of a toxic comment text.

Screen-Shot:

Train Dataset

	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

Test Dataset

	id	comment_text
0	00001cee341fdb12	Yo bitch Ja Rule is more succesful then you'll...
1	0000247867823ef7	== From RfC == \n\n The title is fine as it is...
2	00013b17ad220c46	" \n\n == Sources == \n\n * Zawe Ashton on Lap...
3	00017563c3f7919a	:If you have a look back at the source, the in...
4	00017695ad8997eb	I don't anonymously edit articles at all.

- Data Preprocessing Done
  1. Converted all the characters in the 'comment_text' column to lowercase.
  2. Replaced all email addresses, website links, currencies, phone numbers and any numbers with constant texts link

emailaddress, webaddress, currencyamount, phonenumber and numbr respectively.

3. Removed all non-alphabetic characters.
4. Replaced all multiple blank spaces with single blank space.
5. Removed the stopwords from the ‘reviews’ column.
6. Removed any patterns of repeating characters for unwanted number of times.
7. Saved the cleaned text as ‘cleaned_comment_text’.
8. The cleaned comment text is finally Lemmatized and saved as ‘lemmatized_clean_comment_text’.
9. Used the lemmatized cleaned ‘lemmatized_clean_comment_text’ field to create features using TFIDF with monograms, bigrams and trigrams. Max features restricted to 1,00,000 to compensate the computation power available.

- ## Data Inputs- Logic- Output Relationships

The input data consists of float values which are derived using TFIDF method from the ‘lemmatized_clean_comment_text’ column in the dataset.

The TFIDF method uses monograms, bigrams and trigrams to create the independent features for the model and the output contains numerical multiple classes.

The model approximates the function between the input and the output.

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

No restrictions on where the data is scraped.

- ## Hardware and Software Requirements and Tools Used

1. Google Colab
2. SKLEARN
3. TFIDFVECTORIZER
4. MATPLOTLIB
5. PANDAS

6. NUMPY
7. SKMULTILEARN

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)
    1. Clean the dataset using NLP approaches.
    2. Compare different models and identify the suitable model.
- Testing of Identified Approaches (Algorithms)
    1. SGDClassifier
    2. LogisticRegression
    3. MultinomialNB
    4. OneVsRestClassifier
    5. ClassificationChain
    6. BinaryRelevance
    7. LabelPowerset
- Run and Evaluate selected models

<u>Comparing OneVsRest models using Micro F1 metric:</u>

Sample Code:

```
from sklearn.linear_model import LogisticRegression
start = datetime.now()

classifier1 = OneVsRestClassifier(LogisticRegression(class_weight='balanced'), n_jobs=-1)
classifier1.fit(x_train, y_train)
predictions = classifier1.predict(x_val)

accuracy = metrics.accuracy_score(y_val, predictions)
hamming = metrics.hamming_loss(y_val,predictions)
print("Accuracy :",accuracy)
print("Hamming loss ",hamming)

precision = precision_score(y_val, predictions, average='micro')
recall = recall_score(y_val, predictions, average='micro')
f1 = f1_score(y_val, predictions, average='micro')

print("\nMicro-average quality numbers")
print("Precision: {:.4f}, Recall: {:.4f}, F1-measure: {:.4f}".format(precision, recall, f1))

models_performances.loc[len(models_performances.index)] = ['ovr-logistic regression', 'tfidf 1-3 gram', accuracy, hamming, precision, recall, f1]

precision = precision_score(y_val, predictions, average='macro')
recall = recall_score(y_val, predictions, average='macro')
f1 = f1_score(y_val, predictions, average='macro')

print("\nMacro-average quality numbers")
print("Precision: {:.4f}, Recall: {:.4f}, F1-measure: {:.4f}".format(precision, recall, f1))

print("\nClassification Report")
print (metrics.classification_report(y_val, predictions))
print("Time taken to run this cell :", datetime.now() - start)
```

Results:

	Model	Vectorizer	Accuracy	Hamming loss	Micro-Precision	Micro Recall	Micro F1
0	ovr-logistic regression	tfidf 1-1 gram	0.863512	0.035250	0.513568	0.866836	0.644998
1	ovr-sdg classifier - log loss	tfidf 1-1 gram	0.853736	0.042660	0.458607	0.857506	0.597606
2	ovr-sgd classifier - hinge loss	tfidf 1-1 gram	0.857528	0.039741	0.479231	0.874187	0.619081
3	ovr-logistic regression	tfidf 1-2 gram	0.874479	0.031344	0.548995	0.849025	0.666815
4	ovr-sgd classifier - log loss	tfidf 1-2 gram	0.857747	0.040300	0.474446	0.843794	0.607377
5	ovr-sgd classifier - hinge loss	tfidf 1-2 gram	0.865988	0.035845	0.508770	0.861040	0.639609
6	ovr-logistic regression	tfidf 1-3 gram	0.874416	0.031166	0.550865	0.846621	0.667447
7	ovr-sgd classifier - log loss	tfidf 1-3 gram	0.857246	0.040049	0.476232	0.842663	0.608545
8	ovr-sgd classifier - hinge loss	tfidf 1-3 gram	0.865079	0.035537	0.511326	0.858355	0.640878
9	ovr-logistic regression	tfidf 1-4 gram	0.874448	0.031177	0.550801	0.846056	0.667224
10	ovr-sgd classifier - log loss	tfidf 1-4 gram	0.858217	0.040091	0.475928	0.842663	0.608296
11	ovr-sgd classifier - hinge loss	tfidf 1-4 gram	0.866176	0.035506	0.511567	0.859627	0.641422
12	ovr-logistic regression	tfidf 2-4 gram	0.687984	0.088908	0.230047	0.599378	0.332484
13	ovr-sgd classifier - log loss	tfidf 2-4 gram	0.779508	0.078652	0.240395	0.522759	0.329341
14	ovr-sgd classifier - hinge loss	tfidf 2-4 gram	0.802507	0.065721	0.279472	0.493639	0.356891

```
[ ] models_performances[models_performances['Micro F1']==models_performances['Micro F1'].max()]
```

	Model	Vectorizer	Accuracy	Hamming loss	Micro-Precision	Micro Recall	Micro F1
6	ovr-logistic regression	tfidf 1-3 gram	0.874416	0.031166	0.550865	0.846621	0.667447

Observations:

1. The OveVsRest with Logistic Regression model is giving a better result when using TFIDF with 1-3 grams.

Hyper parameter tunned OneVsRest models:

Code:

```
from sklearn.model_selection import RandomizedSearchCV
from scipy import stats

st=datetime.now()

alpha = [0.000001,0.00001,0.0001,0.001,0.01,0.1,1,10,100,1000,10000]
penalty=['l1','l2']

params  = {"estimator__C":alpha,
           "estimator__penalty":penalty}

base_estimator = OneVsRestClassifier(LogisticRegression(class_weight='balanced'), n_jobs=-1)
rsearch_cv = RandomizedSearchCV(estimator=base_estimator, param_distributions=params, n_iter=10, cv=5, scoring='f1_micro', n_jobs=-1, verbose=0)
rsearch_cv.fit(x_train , y_train)

print("Time taken to perform hyperparameter tuning: ",datetime.now()-st)
print("Best estimator: ",rsearch_cv.best_estimator_)
print("Best Cross Validation Score: ",rsearch_cv.best_score_)
```

```
Time taken to perform hyperparameter tuning:  0:04:05.754985
Best estimator:  OneVsRestClassifier(estimator=LogisticRegression(C=10, class_weight='balanced',
                                                    dual=False, fit_intercept=True,
                                                    intercept_scaling=1,
                                                    l1_ratio=None, max_iter=100,
                                                    multi_class='auto',
                                                    n_jobs=None, penalty='l2',
                                                    random_state=None,
                                                    solver='lbfgs', tol=0.0001,
                                                    verbose=0, warm_start=False),
                    n_jobs=-1)
Best Cross Validation Score:  0.6893237611674382
```

Results

	Model	Vectorizer	Accuracy	Hamming loss	Micro-Precision	Micro Recall	Micro F1
0	hypertuned ovr-logistic regression	tfidf 1-1 gram	0.863512	0.035250	0.513568	0.866836	0.644998
1	hypertuned ovr-logistic regression	tfidf 1-2 gram	0.880307	0.027719	0.602034	0.736500	0.662513
2	hypertuned ovr-logistic regression	tfidf 1-3 gram	0.883879	0.026983	0.602055	0.795165	0.685265

```
tuned_models_performances[tuned_models_performances['Micro F1'] == tuned_models_performances['Micro F1'].max()]
```

	Model	Vectorizer	Accuracy	Hamming loss	Micro-Precision	Micro Recall	Micro F1
2	hypertuned ovr-logistic regression	tfidf 1-3 gram	0.883879	0.026983	0.602055	0.795165	0.685265

Comparing Binary Relevance techniques using Micro F1 metric:

Sample Code:

```
start = datetime.now()


parameters = [

    {
        'classifier': [GaussianNB()],
        'classifier__var_smoothing': [0.5,0.7,0.9]
    },
    {
        'classifier': [LogisticRegression(class_weight='balanced', penalty='l1')],
        'classifier__C': [0.01,0.1,1,10,100]
    },
    {
        'classifier': [MultinomialNB()],
        'classifier__alpha': [0.1,0.5,0.7,0.9]
    }
]



classifier = RandomizedSearchCV(estimator=BinaryRelevance(), param_distributions=parameters, n_iter=15, cv=5, scoring='f1_micro', n_jobs=-1, verbose=0)
classifier.fit(x_train, y_train)

print("Time taken to perform hyperparameter tuning: ",datetime.now()-start)
print("Best estimator: ",classifier.best_params_)
print("Best Cross Validation Score: ",classifier.best_score_)
```

```
Time taken to perform hyperparameter tuning:  0:09:15.179025
Best estimator:  {'classifier__alpha': 0.1, 'classifier': MultinomialNB(alpha=0.1, class_prior=None, fit_prior=True)}
Best Cross Validation Score:  0.6199729979276754
```

Results:

```
binary_relevance_models_performances
```

	Model	Vectorizer	Accuracy	Hamming loss	Micro-Precision	Micro Recall	Micro F1
0	binary relevance-MultinomialNB	tfidf 1-1 gram	0.912894	0.022576	0.857366	0.466497	0.604230
1	binary relevance-MultinomialNB	tfidf 1-2 gram	0.912173	0.023046	0.834885	0.468900	0.600525
2	binary relevance-MultinomialNB	tfidf 1-3 gram	0.911860	0.023040	0.833919	0.469890	0.601085

```
binary_relevance_models_performances[binary_relevance_models_performances['Micro F1'] == binary_relevance_models_performances['Micro F1'].max()]
```

	Model	Vectorizer	Accuracy	Hamming loss	Micro-Precision	Micro Recall	Micro F1
0	binary relevance-MultinomialNB	tfidf 1-1 gram	0.912894	0.022576	0.857366	0.466497	0.60423

Comparing Classifier Chain models using Micro F1 metric:

Sample Code:

```
from skmultilearn.problem_transform import ClassifierChain
from sklearn.linear_model import LogisticRegression

start = datetime.now()

classifier = ClassifierChain(LogisticRegression(class_weight='balanced'))

classifier.fit(x_train, y_train)
predictions = classifier.predict(x_val)

accuracy = metrics.accuracy_score(y_val, predictions)
hamming = metrics.hamming_loss(y_val,predictions)
print("Accuracy :",accuracy)
print("Hamming loss ",hamming)

precision = precision_score(y_val, predictions, average='micro')
recall = recall_score(y_val, predictions, average='micro')
f1 = f1_score(y_val, predictions, average='micro')

print("\nMicro-average quality numbers")
print("Precision: {:.4f}, Recall: {:.4f}, F1-measure: {:.4f}".format(precision, recall, f1))

classifier_chain_models_performances.loc[len(classifier_chain_models_performances.index)] = ['classifier chain - Logistic Regression','tfidf 1-1 gram', accuracy, hamming, precision, recall, f1]


precision = precision_score(y_val, predictions, average='macro')
recall = recall_score(y_val, predictions, average='macro')
f1 = f1_score(y_val, predictions, average='macro')

print("\nMacro-average quality numbers")
print("Precision: {:.4f}, Recall: {:.4f}, F1-measure: {:.4f}".format(precision, recall, f1))

print("\nClassification Report")
print (metrics.classification_report(y_val, predictions))
print("Time taken to run this cell :", datetime.now() - start)
```

Results:

```
classifier_chain_models_performances
```

	Model	Vectorizer	Accuracy	Hamming loss	Micro-Precision	Micro Recall	Micro F1
0	classifier chain - Logistic Regression	tfidf 1-1 gram	0.830331	0.088093	0.280607	0.885496	0.426166
1	classifier chain - sgd classifier	tfidf 1-1 gram	0.835501	0.099039	0.255400	0.877580	0.395653
2	classifier chain - sgd classifier log loss	tfidf 1-1 gram	0.836973	0.106277	0.241058	0.873622	0.377855
3	classifier chain - Logistic Regression	tfidf 1-2 gram	0.826915	0.092381	0.269778	0.879276	0.412878
4	classifier chain - sgd classifier	tfidf 1-2 gram	0.827855	0.107933	0.238999	0.879842	0.375891
5	classifier chain - sgd classifier log loss	tfidf 1-2 gram	0.811029	0.133835	0.202068	0.889454	0.329321
6	classifier chain - Logistic Regression	tfidf 1-3 gram	0.824315	0.094339	0.265160	0.877156	0.407219
7	classifier chain - sgd classifier	tfidf 1-3 gram	0.841203	0.099358	0.251930	0.857930	0.389488
8	classifier chain - sgd classifier log loss	tfidf 1-3 gram	0.826445	0.122043	0.215384	0.871643	0.345415

```
classifier_chain_models_performances[classifier_chain_models_performances['Micro F1'] == classifier_chain_models_performances['Micro F1'].max()]
```

	Model	Vectorizer	Accuracy	Hamming loss	Micro-Precision	Micro Recall	Micro F1
0	classifier chain - Logistic Regression	tfidf 1-1 gram	0.830331	0.088093	0.280607	0.885496	0.426166

<u>Comparing LabelPowerset models using Micro F1 metric:</u>

Sample code:

```
# using Label Powerset
from skmultilearn.problem_transform import LabelPowerset


start = datetime.now()

# initialize Label Powerset multi-label classifier
# with a gaussian naive bayes base classifier
classifier = LabelPowerset(SGDClassifier(class_weight='balanced',loss='log'))


classifier.fit(x_train, y_train)
predictions = classifier.predict(x_val)

accuracy = metrics.accuracy_score(y_val, predictions)
hamming = metrics.hamming_loss(y_val,predictions)
print("Accuracy :",accuracy)
print("Hamming loss ",hamming)

precision = precision_score(y_val, predictions, average='micro')
recall = recall_score(y_val, predictions, average='micro')
f1 = f1_score(y_val, predictions, average='micro')

print("\nMicro-average quality numbers")
print("Precision: {:.4f}, Recall: {:.4f}, F1-measure: {:.4f}".format(precision, recall, f1))

label_powerset_models_performances.loc[len(label_powerset_models_performances.index)] = ['Label powerset SGDClassifier log loss','tfidf 1-3 gram', accuracy, hamming, precision, recall, f1]


precision = precision_score(y_val, predictions, average='macro')
recall = recall_score(y_val, predictions, average='macro')
f1 = f1_score(y_val, predictions, average='macro')

print("\nMacro-average quality numbers")
print("Precision: {:.4f}, Recall: {:.4f}, F1-measure: {:.4f}".format(precision, recall, f1))

print("\nClassification Report")
print (metrics.classification_report(y_val, predictions))
print("Time taken to run this cell :", datetime.now() - start)
```

Results:

```
label_powerset_models_performances
```

	Model	Vectorizer	Accuracy	Hamming loss	Micro-Precision	Micro Recall	Micro F1
0	Label powerset Logistic Regression	tfidf 1-3 gram	0.695065	0.078124	0.280579	0.712751	0.402651
1	Label powerset SGDClassifier hinge loss	tfidf 1-3 gram	0.818079	0.064035	0.328982	0.705400	0.448701
2	Label powerset SGDClassifier log loss	tfidf 1-3 gram	0.894000	0.031547	0.589406	0.481340	0.529920
3	Label Powerset MultinomialNB	tfidf 1-3 gram	0.911014	0.025244	0.910858	0.351004	0.506735
4	Label Powerset GaussianNB	tfidf 1-3 gram	0.194423	0.256217	0.095256	0.698473	0.167648
5	Label Powerset RandomForestClassifier	tfidf 1-3 gram	0.895786	0.029537	0.662243	0.409104	0.505767

Final Model:

```
malignant_comments_classifier_pipeline = Pipeline([
                    ('tfidf_vectorizer', TfidfVectorizer(min_df=0.00009, max_features= 100000, smooth_idf=True, norm="l2", sublinear_tf=False, ngram_range=(1,3))),
                    ('ovr_logistic_regression', OneVsRestClassifier(estimator=LogisticRegression(C=10, class_weight='balanced',
                dual=False, fit_intercept=True,
                intercept_scaling=1,
                l1_ratio=None, max_iter=100,
                multi_class='auto',
                n_jobs=None, penalty='l2',
                random_state=None,
                solver='lbfgs', tol=0.0001,
                verbose=0, warm_start=False),
                n_jobs=-1))
])
```

Results:

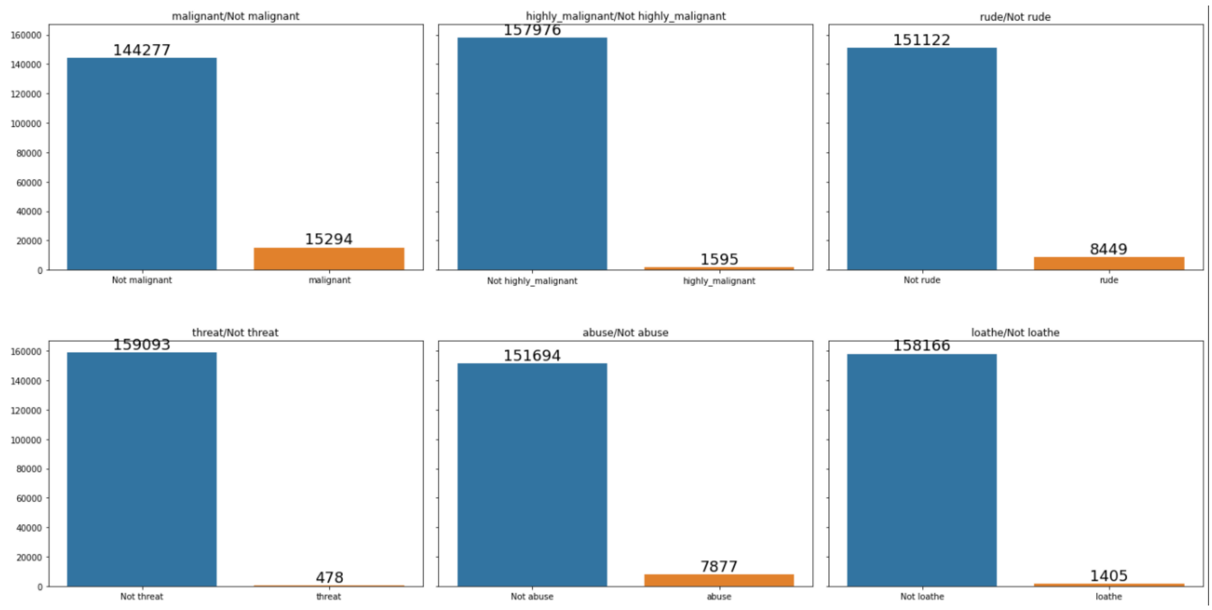
```
The F1 scores from the cross val score for the train data are: [0.67817332 0.68350208 0.68792225 0.66953528 0.68732864 0.67740736
 0.68194495 0.68064594 0.68379496 0.67789028].
The average of F1 scores from the cross val score for the train data are: 0.680814506664462.
The variance of F1 scores from the cross val score for the train data are: 2.6472297682420997e-05.
```

Observations:

1. The LogisticRegression model with OneVsRest technique is showing the best performance when combined with TFIDF of 1-3 grams vectorization.

- Key Metrics for success in solving problem under consideration
    1. The classes are all imbalance and we are dealing with a Multi-label classification problem. Hence, Micro F1 score is used as the key metric for evaluation.
    2. The Hamming Loss, Micro Precision and Micro Recall are also used as primary metrics.
    3. The Macro Precision, Macro Recall and Macro F1 scores are used as secondary metrics.
- Visualizations

1. Frequency of Toxic labels vs Clean comments:



Observations:

1.The plots shows that the train dataset contains large number of samples that are not toxic than the samples for any of the types of toxic.

2. Correlation between toxicity labels:

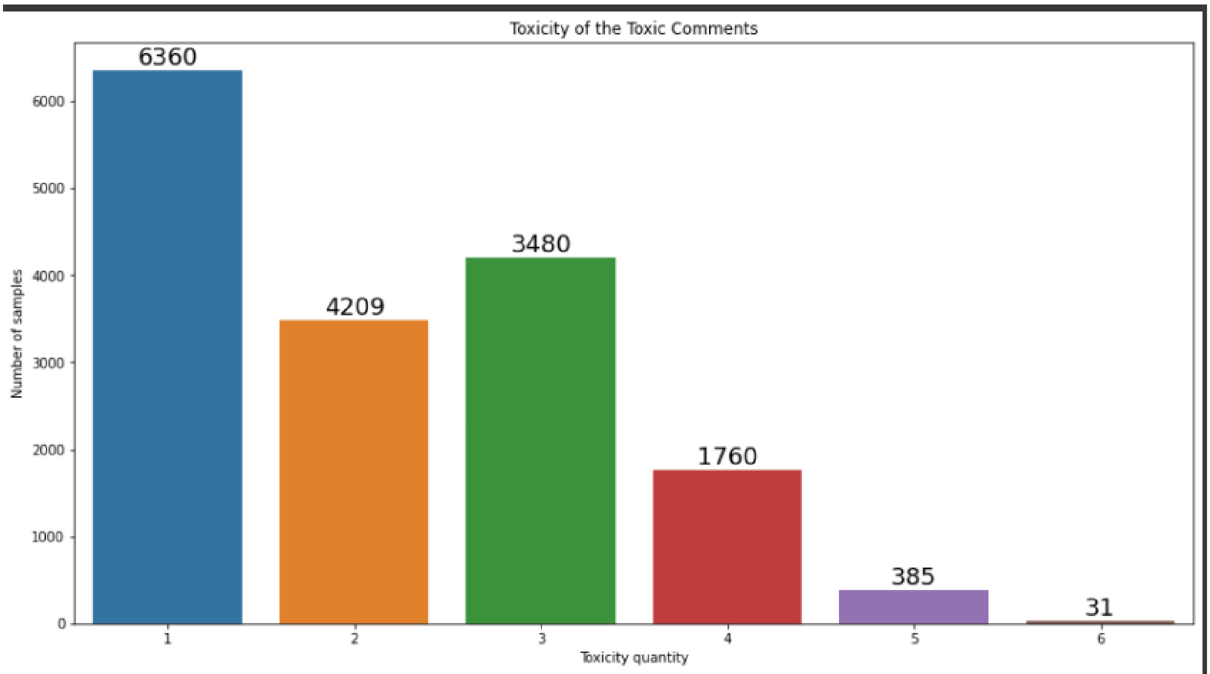
Observations:

1. The 'rude' and 'abuse' target variables have slightly strong positive correlation with each other with 0.74.

2. 'malignant' and 'rude' have positive correlation with each other.

3. 'malignant has a positive correlation with 'abuse' also.

3. Toxicity of the comments:

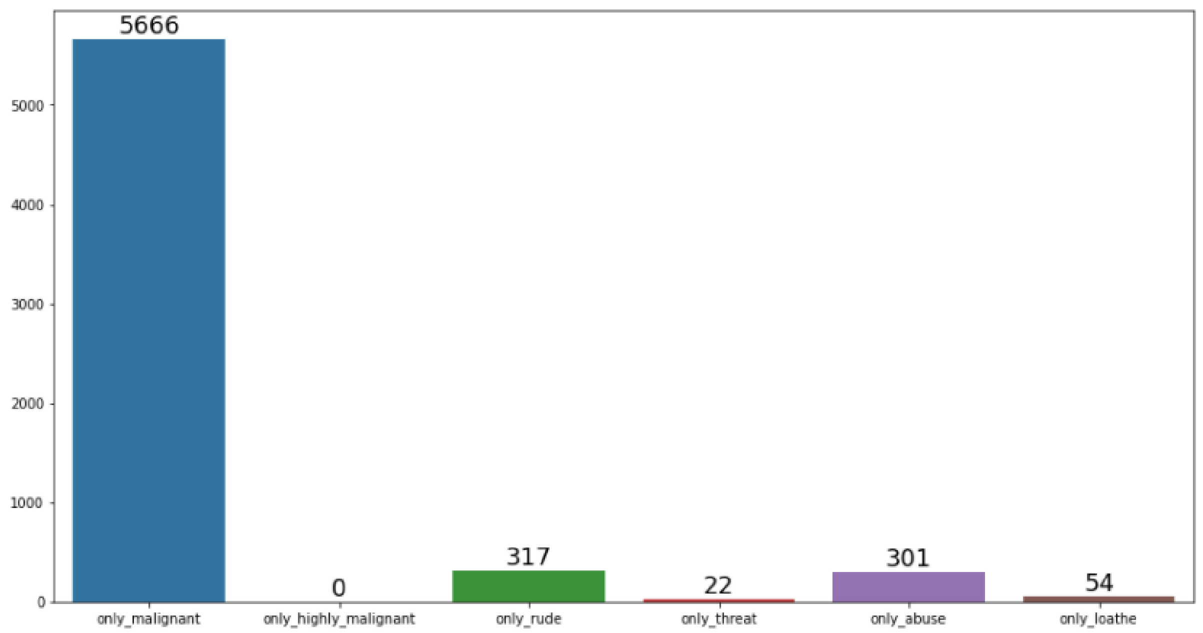



```
There are 6360 comments which have 1 type(s) of toxicity.
There are 4209 comments which have 2 type(s) of toxicity.
There are 3480 comments which have 3 type(s) of toxicity.
There are 1760 comments which have 4 type(s) of toxicity.
There are 385 comments which have 5 type(s) of toxicity.
There are 31 comments which have 6 type(s) of toxicity.
There are 143346 non toxic comments in the train data.
```

Observations:

1. About 6360 comments/samples have only one type of toxicity. Which means these many data points have only one label associated to them.
2. About 4209 data points have 2 labels associated to them.
3. About 3480 data points have 3 labels associated to them.
4. About 1760 data points have 4 labels associated to them.
5. About 385 data points have 5 labels associated to them.
6. About 31 data points have 6 labels associated to them.
7. About 143346 data points have no labels associated to them.

4. Looking at number of comments that have unique Toxicity.




```
Toxicity        Number of Samples in train dataset

========        ==================================
only_malignant           5666
only_highly_malignant       0
only_rude                 317
only_threat                22
only_abuse                301
only_loathe                54
dtype: int64
```

Observations:

1. There are 5666 comments in the train data that are associated to only malignant label.
2. There are no comments in the train data that is associated to only highly malignant label.
3. There are 317 comments in the train data that are associated to only rude label.
4. There are 22 comments in the train data that are associated to only threats label.
5. There are 301 comments in the train data that are associated to only abuses label.
6. There are 54 comments in the train data that are associated to only loathes label.

5. Comparing Ratings 1, 2, 3 4 and 5 word-clouds:

malignant

highly malignant

rude

threat

Abuse

loathe

Clean comments

- Interpretation of the Results
    1. The dataset contains mostly clean comments.
    2. Most of the toxic comments belong to 'malignant' label.
    3. All the highly malignant comments are also malignant. There are no comments that are only malignant.
    4. Most of the toxic comments have only one toxic label to it.

CONCLUSION

- Key Findings and Conclusions of the Study
    1. All of the toxic comments have negative words.
    2. More the data used for training the better the model performed.
    3. OneVsRest technique with Logistic Regression is able to perform well for the 1-3 grams TFIDF Vectorized data.
- Learning Outcomes of the Study in respect of Data Science

  Learned about the multi-label classification problem and the process to solve it.

- Limitations of this work and Scope for Future Work

  Computational power is the limitation faced in this project. The RAM memory in Google Colab was not enough for Adaptative Algorithms and the Word2Vec models.

  If there is enough computation power, using more data for training will give better results.

  Exploring ensemble techniques might give better results.