Malignant-Comments Classifiers Project

Submitted by:

Aditi Gupta

Internship 12

ACKNOWLEDGMENT

Gratitude takes three forms-"A feeling from heart, an expression in words and a giving in return". We take this opportunity to express our feelings.

I express my gracious gratitude to our project guide **Khushboo Garg**, SME of Flip Robo Technologies, for his valuable guidance and assistance. I am very thankful for his encouragement and inspiration that made project successful.

I express my gracious gratitude to our Trainer **Dr. Deepika Sharma**, Training Head of DataTrained - Data Analytics, Data Science Online Training, Bengaluru, for her valuable guidance and assistance throughout the PG Program. I am very thankful for her encouragement and inspiration that made project successful.

I would like to thank **Vishal**, In House Data Scientist of DataTrained - Data Analytics, Data Science Online Training, Bengaluru, for his constant support, encouragement, and guidance during project.

I would like to thank **Shankar**, In House Data Scientist of DataTrained - Data Analytics, Data Science Online Training, Bengaluru for his profound guidance throughout the training.

My special thanks to all the Instructors and Subject Matter Experts of DataTrained - Data Analytics, Data Science Online Training, Bengaluru, who helped me during the live sessions and during doubt clearing scenarios from which I received a lots of suggestions that improved the quality of the work.

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 behaviour. 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 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.

  This is a Multilabel Classification Problem.

  Multilabel classification

In multi-label classification, data can belong to more than one label simultaneously. For example, in our case a comment may be malignant, threat or loathe at the same time. It may also happen that the comment is positive/neutral and hence does not belong to any of the six labels. This is therefore a multi-label classification problem.

- Motivation for the Problem Undertaken

  The upsurge in the volume of unwanted comments called malignant comments has created an intense need for the development of more dependable and robust malignant comments filters. Machine learning methods of recent are being used to successfully detect and filter malignant comments. Build a model which can be used to predict in terms of a probability for comments to be malignant. In this case, Label '1' indicates that the comment is malignant, while, Label '0' indicates that the comment is not malignant.

- OBJECTIVE

  The objective of identification of comments are :

  - To give knowledge to the user about the bad comments and positive comments
  - To classify that comments are malignant or not.

- SCOPE OF THE PROJECT:
  - It provides sensitivity to the client and adapts well to the future malignant comments detection techniques.
  - It considers a complete message instead of single words with respect to its organization.
  - It increases Security and Control.
  - It reduces IT Administration Costs.
  - It also reduce Network Resource Costs.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

  Machine Learning is defined by Tom Mitchell in his book as “A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E”. Supervised learning is when the output is known for the corresponding inputs, and is also provided for the machine to learn.
  - EDA (Exploratory data analysis)
  - Data Preprocessing
  - Feature Extraction
  - Scoring & Metrics

- Data Sources and their formats

  The data is provided to us from our client database. It is hereby given to us for the exercise to improve the selection of comments for malignant or not malignant. It is given in the csv file format.

```
df=pd.read_csv('malignant_train.csv')
```

```
# Let's have a look at top 5 rows
df.head()
```

	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

```
# Similarly, reading test dataset
df_test=pd.read_csv('malignant_test.csv')
df_test.head()
```

	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 Set 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.

- Data Preprocessing Done

The dataset that will be used to train the model has some challenges. Text Cleaning is a very important step in machine learning because your data may contains a lot of noise and unwanted character such as punctuation, white space, numbers, hyperlink and etc.

Some standard procedures are:

- convert all letters to lower/upper case
- removing numbers

- removing punctuation
- removing white spaces
- removing hyperlink
- removing pos tags
- removing short words
- removing stop words such as *a, about, above, down, doing* and the list goes on… Sometimes, the extremely common word which would appear to be of very little value in helping select documents matching user need are excluded from the vocabulary entirely.
- Word lemmatization: Lemmatization is utilizing the dictionary of a particular language and tried to convert the words back to its base form. It will try to take into account of the meaning of the verbs and convert it back to the most suitable base form.
- Adding new column “Comment_length” which is the length of comment in number.
- Adding new column “Clean_comment_text” which is done by cleaning the column “comment-text”.
- Adding one more column “Clean_comment_text_length” which is count of length of comment column by mapping.

```
# Fuction to remove short words
def clean_text(text):
    text = text.lower()
    text = re.sub(r"what's", "what is ", text)
    text = re.sub(r"\'s", " ", text)
    text = re.sub(r"\'ve", " have ", text)
    text = re.sub(r"can't", "cannot ", text)
    text = re.sub(r"n't", " not ", text)
    text = re.sub(r"i'm", "i am ", text)
    text = re.sub(r"\'re", " are ", text)
    text = re.sub(r"\'d", " would ", text)
    text = re.sub(r"\'ll", " will ", text)
    text = re.sub(r"\'scuse", " excuse ", text)
    text = re.sub('\W', ' ', text)
    text = re.sub('\s+', ' ', text)
    text = text.strip(' ')
    return text
```

```
# function to filter using POS tagging. This will be called inside the below function
def get_pos(pos_tag):
    if pos_tag.startswith('J'):
        return wordnet.ADJ
    elif pos_tag.startswith('N'):
        return wordnet.NOUN
    elif pos_tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN
```

```
# Function for data cleaning.
def Processed_data(comments):
    # Replace email addresses with 'email'
    comments=re.sub(r'^.+@[^\.].*\.[a-z]{2,}$',' ', comments)

    # Replace 10 digit phone numbers (formats include paranthesis, spaces, no spaces, dashes) with 'phonenumber'
    comments=re.sub(r'^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$',' ',comments)

    # getting only words(i.e removing all the special characters)
    comments = re.sub(r'[^\w]', ' ', comments)

    # getting only words(i.e removing all the" _ ")
    comments = re.sub(r'[\_]', ' ', comments)

    # getting rid of unwanted characters(i.e remove all the single characters left)
    comments=re.sub(r'\s+[a-zA-Z]\s+', ' ', comments)

    # Removing extra whitespaces
    comments=re.sub(r'\s+', ' ', comments, flags=re.I)

    #converting all the letters of the review into lowercase
    comments = comments.lower()

    # splitting every words from the sentences
    comments = comments.split()

    # iterating through each words and checking if they are stopwords or not,
    comments=[word for word in comments if not word in set(STOPWORDS)]

    # remove empty tokens
    comments = [text for text in comments if len(text) > 0]

    # getting pos tag text
    pos_tags = pos_tag(comments)

    # considering words having length more than 3only
    comments = [text for text in comments if len(text) > 3]

    # performing lemmatization operation and passing the word in get_pos function to get filtered using POS
    comments = [(WordNetLemmatizer().lemmatize(text[0], get_pos(text[1])))for text in pos_tags]

   # considering words having length more than 3 only
    comments = [text for text in comments if len(text) > 3]
    comments = ' '.join(comments)
    return comments
```

```
# Adding new feature comment_length to store length of characters for train data
df['comment_length'] = df['comment_text'].apply(lambda x: len(str(x)))
df.head()
```

	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	comment_length
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0	264
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0	112
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0	233
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0	622
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0	67

```
# Adding new feature comment_length to store length of characters for test data
df_test['comment_length'] = df_test['comment_text'].apply(lambda x: len(str(x)))
df_test.head()
```

	id	comment_text	comment_length
0	00001cee341fdb12	Yo bitch Ja Rule is more succesful then you'll...	367
1	0000247867823ef7	== From RfC == \n\n The title is fine as it is...	50
2	00013b17ad220c46	" \n\n == Sources == \n\n * Zawe Ashton on Lap...	54
3	00017563c3f7919a	:If you have a look back at the source, the in...	205
4	00017695ad8997eb	I don't anonymously edit articles at all.	41

```
# Replacing short words with actual words in train and test data both
df['comment_text'] = df['comment_text'].map(lambda comments : clean_text(comments))
df_test['comment_text'] = df_test['comment_text'].map(lambda comments : clean_text(comments))
```

```
# cleaning the comments and storing them in a separate feature in train and test dataset both
df["clean_comment_text"] = df["comment_text"].apply(lambda x: Processed_data(x))
df_test["clean_comment_text"] = df_test["comment_text"].apply(lambda x: Processed_data(x))
```

```
# Adding new feature clean_comment_length to store length of characters in train and test dataset both
df['clean_comment_length'] = df['clean_comment_text'].apply(lambda x: len(str(x)))
df_test['clean_comment_length'] = df_test['clean_comment_text'].apply(lambda x: len(str(x)))
```

```
#checking the train dataset after preprocessing
df
```

	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	comment_length	clean_comment_text	clean_comment_length
0	0000997932d777bf	explanation why the edits made under my userna...	0	0	0	0	0	0	264	explanation edits username hardcore metallica ...	123
1	000103f0d9cfb60f	d aww he matches this background colour i am s...	0	0	0	0	0	0	112	match background colour seemingly stuck thanks...	64
2	000113f07ec002fd	hey man i am really not trying to edit war it ...	0	0	0	0	0	0	233	trying edit constantly removing relevant infor...	112
3	0001b41b1c6bb37e	more i cannot make any real suggestions on imp...	0	0	0	0	0	0	622	real suggestion improvement wondered section s...	315
4	0001d958c54c6e35	you sir are my hero any chance you remember wh...	0	0	0	0	0	0	67	hero chance remember page	25

```
#checking the test dataset after preprocessing
df_test
```

	id	comment_text	comment_length	clean_comment_text	clean_comment_length
0	00001cee341fdb12	yo bitch ja rule is more succesful then you wi...	367	bitch rule succesful whats hating mofuckas bit...	184
1	0000247867823ef7	from rfc the title is fine as it is imo	50	title fine	10
2	00013b17ad220c46	sources zawe ashton on lapland	54	source zawe ashton lapland	26
3	00017563c3f7919a	if you have a look back at the source the info...	205	look source information updated correct form g...	104
4	00017695ad8997eb	i do not anonymously edit articles at all	41	anonymously edit article	24
...	...	...	...	...	...
153159	fffcd0960ee309b5	i totally agree this stuff is nothing but too ...	60	totally agree stuff long crap	29
153160	fffd7a9a6eb32c16	throw from out field to home plate does it get...	198	throw field home plate faster throwing direct ...	85
153161	fffda9e8d6fafa9e	okinotorishima categories i see your changes a...	423	okinotorishima category change agree correct g...	212
153162	fffe8f1340a79fc2	one of the founding nations of the eu germany ...	502	founding nation germany return similar israel ...	275
153163	ffffce3fb183ee80	stop already your bullshit is not welcome here...	141	stop bullshit welcome fool think kind explinat...	54

153164 rows × 5 columns

- Hardware and Software Requirements and Tools Used

  Hardware : Since the computational aspect of the project is of importance to PANDA, it is important to know the hardware that was used in the evaluation process. The training and evaluation of the neural network model has been done on a Windows 10 computer using a quad-core CPU at i3.

  Software : anaconda 3 , windows 10 ,Microsoft office.

  Tools used : python , machine learning libraries, Nltk, Nlp libraries.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  From the given dataset it can be concluded that it is a **MultiLabel Classification** problem as six output columns “Malignant”,”Highly_Malignant”, “Rude“, “Threat”, “Abuse”, “Loathe” has binary output “0 & 1”. So for further analysis of the problem we have to import or call out the Classification related libraries in Python work frame. The different libraries used for the problem solving are sklearn - Scikit-learn is a free machine learning library for Python. It features various algorithms like random forests, and k-neighbours, and it also supports Python numerical and scientific libraries like NumPy and SciPy.

  1. **sklearn.tree - DecisionTreeClassifier**

     Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

     There are several advantages of using decision trees for predictive analysis:

     - Decision trees can be used to predict both continuous and discrete values i.e. they work well for both regression and classification tasks.
     - They require relatively less effort for training the algorithm.
     - They can be used to classify non-linearly separable data.
     - They're very fast and efficient compared to KNN and other classification algorithms.

     Decision tree learning is one of the predictive modelling approaches used in statistics, data mining and machine learning.

It uses a decision tree to go from observations about an item to conclusions about the item's target value.

2. **sklearn.KNeighborClassifier**

   K Nearest Neighbor(KNN) is a very simple, easy to understand, versatile and one of the topmost machine learning algorithms. KNN used in the variety of applications such as finance, healthcare, political science, handwriting detection, image recognition and video recognition. In Credit ratings, financial institutes will predict the credit rating of customers. In loan disbursement, banking institutes will predict whether the loan is safe or risky. In political science, classifying potential voters in two classes will vote or won't vote. KNN algorithm used for both classification and regression problems. KNN algorithm based on feature similarity approach.

   KNN is a non-parametric and lazy learning algorithm. Non-parametric means there is no assumption for underlying data distribution. In other words, the model structure determined from the dataset. This will be very helpful in practice where most of the real world datasets do not follow mathematical theoretical assumptions. Lazy algorithm means it does not need any training data points for model generation. All training data used in the testing phase. This makes training faster and testing phase slower and costlier. Costly testing phase means time and memory. In the worst case, KNN needs more time to scan all data points and scanning all data points will require more memory for storing training data.

3. **sklearn.ensemble**

   The goal of ensemble methods is to combine the predictions of several base estimators built with a given learning algorithm in order to improve generalizability / robustness over a single estimator. The sklearn.ensemble module includes two averaging algorithms based on randomized decision trees: the RandomForest algorithm and the Extra-Trees method. Both algorithms are perturb-and-combine techniques specifically designed for trees. This means a diverse set of classifiers is created

by introducing randomness in the classifier construction. The prediction of the ensemble is given as the averaged prediction of the individual classifiers. Boosting ensemble algorithms creates a sequence of models that attempt to correct the mistakes of the models before them in the sequence. Once created, the models make predictions which may be weighted by their demonstrated accuracy and the results are combined to create a final output prediction.

The different types of ensemble techniques are

i. **Random Forest Classifier** - Random Forest uses multiple decision trees as base learning models in the dataset. 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 in the dataset. The main concept of Random Forest is to combine multiple decision trees in determining the final result rather than relying on individual decision trees.

ii. **Extra Trees Classifier**- is an ensemble learning method fundamentally based on decision trees. Extra Trees Classifier, like Random Forest, randomizes certain decisions and subsets of data to minimize over-learning from the data and over fitting. This class implements a meta estimator that fits a number of randomized decision trees (a.k.a. extra-trees) on various subsamples of the dataset and uses averaging to improve the predictive accuracy and control overfitting.

- Methods used for evaluation

1. Accuracy_Score:

The most common metric for classification is accuracy, which is the fraction of samples predicted correctly as shown below:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Fraction predicted correctly

2. Log loss:

This is the loss function used in (multinomial) logistic regression and extensions of it such as neural networks, defined as the negative log-likelihood of the true labels given a probabilistic classifier's predictions. For a single sample with true label yt in {0,1} and estimated probability yp that yt = 1, the log loss is

$$-\log P(yt|yp) = -(yt \log(yp) + (1 - yt) \log(1 - yp))$$

3. Recall Score:

Recall (also known as sensitivity) is the fraction of positives events that you predicted correctly as shown below:

$$\text{Recall (Sensitivity)} = \frac{TP}{TP + FN}$$

Fraction of positives predicted correctly

4. Precision Score:

Precision is the fraction of predicted positives events that are actually positive as shown below:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Fraction of predicted positives that are actually positive

```python
from sklearn.metrics import f1_score, accuracy_score, log_loss, recall_score, precision_score

# classify function
from sklearn.model_selection import cross_val_score, train_test_split
def classify(model, x, y):
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
    # train the model
    model.fit(x_train, y_train)
    print("Accuracy:", model.score(x_test, y_test))
    pred = model.predict(x_test)
    print("Log Loss:", log_loss(y_test, pred))
    print("Recall:", recall_score(y_test, pred, average='micro'))
    print("Precision:", precision_score(y_test, pred, average='micro'))
```

- Results

```python
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
classify(model, x, y)
```

```
Accuracy: 0.8902710324298919
Log Loss: 1.2988274672416598
Recall: 0.5356234096692112
Precision: 0.6512547267102097
```

```python
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier()
classify(model, x, y)
```

```
Accuracy: 0.8932476891743694
Log Loss: 1.1103358888739179
Recall: 0.4072660446706248
Precision: 0.7042287949156686
```

```python
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
classify(model, x, y)
```

```
Accuracy: 0.913018956603478
Log Loss: 1.200646913283439
Recall: 0.5469324286118179
Precision: 0.8172792564427546
```

```python
from sklearn.ensemble import ExtraTreesClassifier
model = ExtraTreesClassifier()
classify(model, x, y)
```

```
Accuracy: 0.9131442895190349
Log Loss: 1.207874679185791
Recall: 0.5357647724059937
Precision: 0.8239130434782609
```

- Hyperparameter tuning

  RandomizedSearchCV - It is a library function that is a member of sklearn's model_selection package. It helps to loop through predefined hyper parameters and fit your estimator (model) on your training set. So, in the end, you can select the best parameters from the listed hyperparameters. RadomizedSearchCV combines an estimator with a grid search preamble to tune hyper-parameters. The method picks the optimal parameter from the grid search and uses it with the estimator selected by the user.

```
from sklearn.model_selection import RandomizedSearchCV
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```

```
rfc=RandomForestClassifier()
parameters={'bootstrap': [True, False],
 'max_depth': [10, 50, 100, None],
 'min_samples_leaf': [1, 2, 4],
 'min_samples_split': [2, 5, 10],
 'n_estimators': [100, 300, 500, 800, 1200]}

# Applying Randomized Search CV for hyperparameter tuning with scoring= "accuracy"
rand = RandomizedSearchCV(estimator = rfc, param_distributions = parameters,
                          n_iter = 10, cv = 3, verbose=2, random_state=42, n_jobs = -1,scoring='accuracy')
rand.fit(x_train,y_train)
rand.best_params_
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed: 52.0min finished
```

```
{'n_estimators': 500,
 'min_samples_split': 2,
 'min_samples_leaf': 1,
 'max_depth': 100,
 'bootstrap': False}
```

- Choosing Best model

Random Forest is giving highest accuracy with highest precision so we choose it as best model

```
RFC=RandomForestClassifier(n_estimators= 500,
                           min_samples_split= 2,
                           min_samples_leaf=1,
                           max_depth= 100,
                           bootstrap= False)
```

```
RFC.fit(x_train,y_train)
RFC.score(x_train,y_train)
pred=RFC.predict(x_test)
print('Accuracy Score:',accuracy_score(y_test,pred))
print('Log loss : ', log_loss(y_test,pred))
print("Recall:", recall_score(y_test, pred, average='micro'))
print("Precision:", precision_score(y_test, pred, average='micro'))
```

```
Accuracy Score: 0.9162462791790694
Log loss :  1.536268001619044
Recall: 0.5606446140797285
Precision: 0.844009363694403
```

- Cross Validation

  Cross validation helps to find out the over fitting and under fitting of the model. In the cross validation the model is made to run on different subsets of the dataset which will get multiple measures of the model. If we take 5 folds, the data will be divided into 5 pieces where each part being 20% of full dataset. While running the Cross validation the 1st part (20%) of the 5 parts will be kept out as a hold out set for validation and everything else is used for training data. This way we will get the first estimate of the model quality of the dataset. In the similar way further iterations are made for the second 20% of the dataset isheld as a hold out set and remaining 4 parts are used for training data during process. This way we will get the second estimate of the model quality of the dataset. These steps are repeated during the cross validation process to get the remaining estimate of the model quality. cross_val_score estimates the expected accuracy of the model on out-of-training data (pulled from the same underlying process as the training data). The benefit is that one need not set aside any data to obtain this metric, and we can still train the model on all of the available data.

```
# cross validation

from sklearn.model_selection import cross_val_score

scores=cross_val_score(RFC,x,y,cv=5)
print(scores)
print(scores.mean(),scores.std())
```

```
[0.91483628 0.91605565 0.91646299 0.91796704 0.9144576 ]
0.9159559137915405 0.001249815425296559
```

- Prediction on test data

```
def Tf_idf_test(text):
    tfid = TfidfVectorizer(min_df=2,max_features=62791,smooth_idf=False)
    return tfid.fit_transform(text)
```

```
x_testing_data=Tf_idf_test(df_test['clean_comment_text'])
```

```
x_testing_data.shape
```

```
(153164, 62791)
```

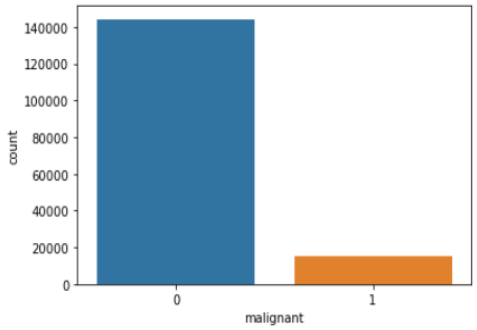
```
pred = RFC.predict(x_testing_data)
```

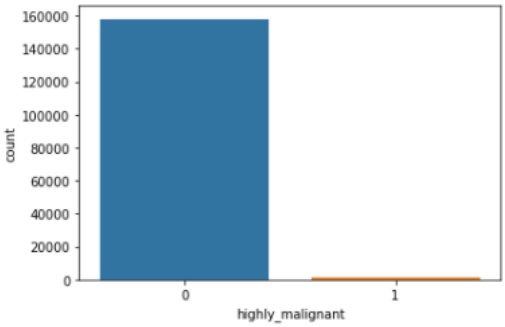
```
sub = pd.DataFrame(pred, columns=['malignant', 'highly_malignant', 'rude', 'threat', 'abuse', 'loathe'])
sub['id'] = df_test['id']
sub = sub[['id', 'malignant', 'highly_malignant', 'rude', 'threat', 'abuse', 'loathe']]
sub.head()
```

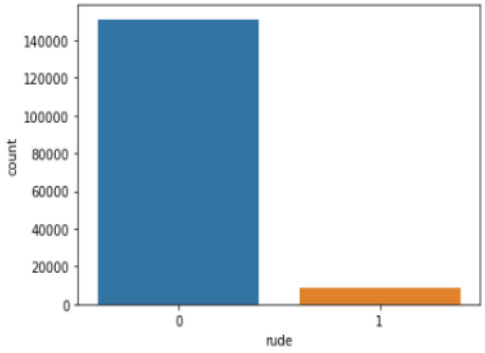
	id	malignant	highly_malignant	rude	threat	abuse	loathe
0	00001cee341fdb12	0	0	0	0	0	0
1	0000247867823ef7	1	0	0	0	0	0
2	00013b17ad220c46	0	0	0	0	0	0
3	00017563c3f7919a	0	0	0	0	0	0
4	00017695ad8997eb	0	0	0	0	0	0

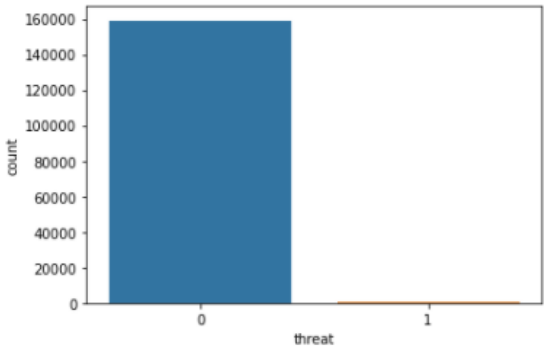
- Visualizations

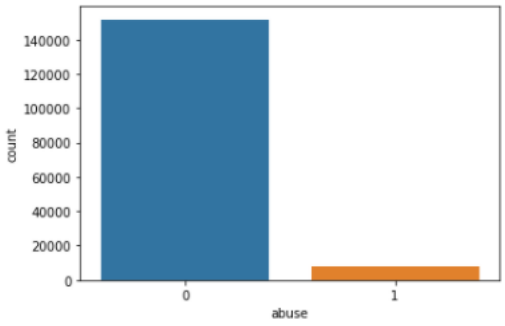
Count plot for Output Labels for 0,1

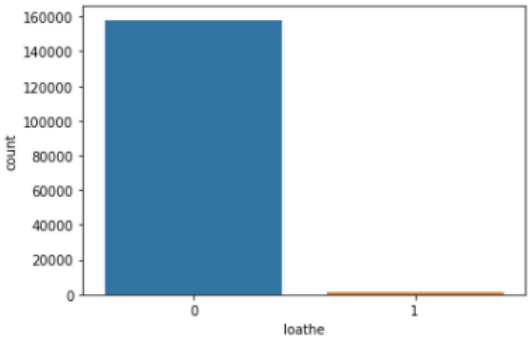




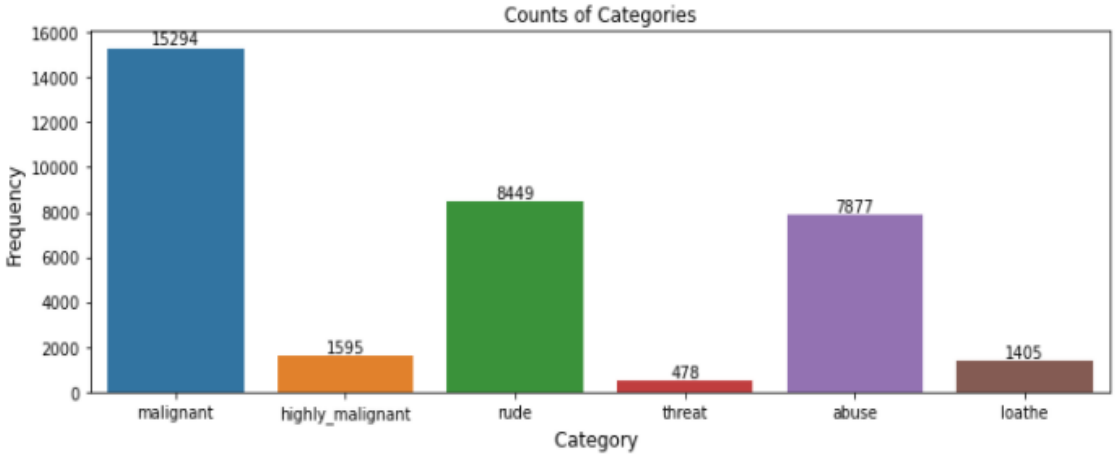




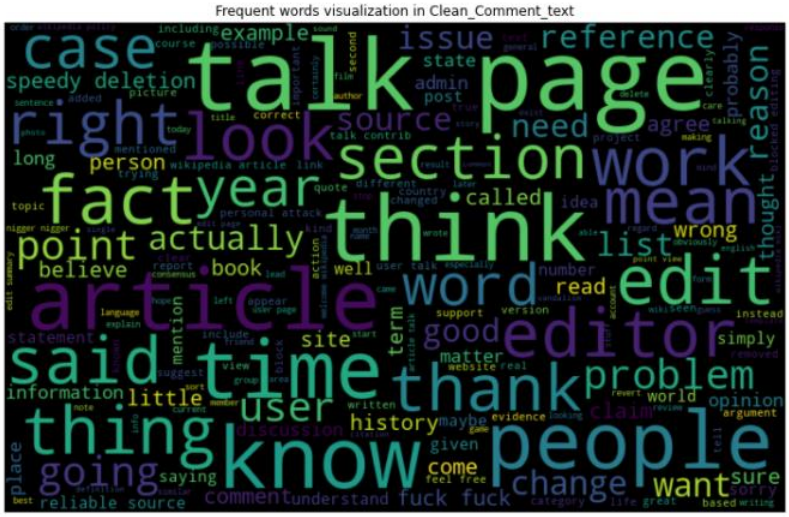




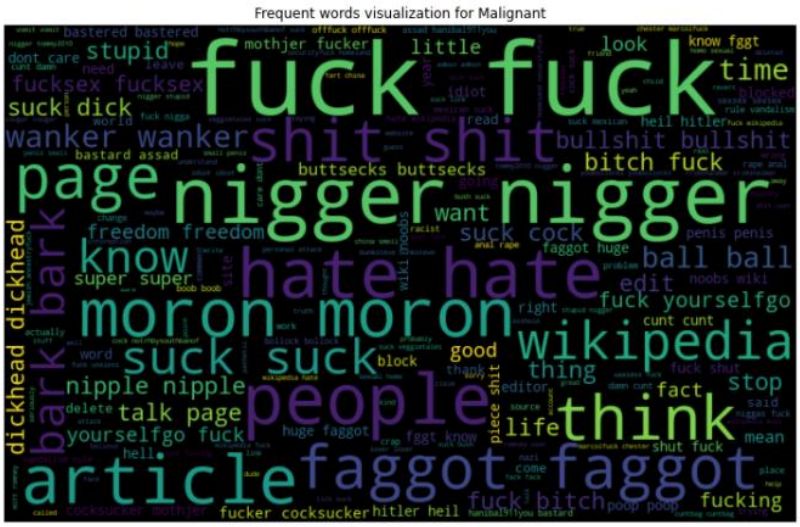
Count plot for all output category



Frequent word visualization for clean comment text



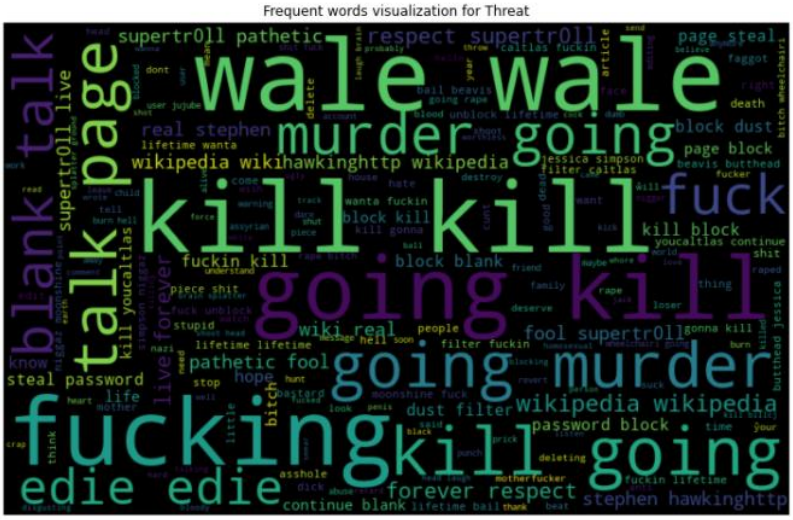
Frequent word visualization for Malignant Comments



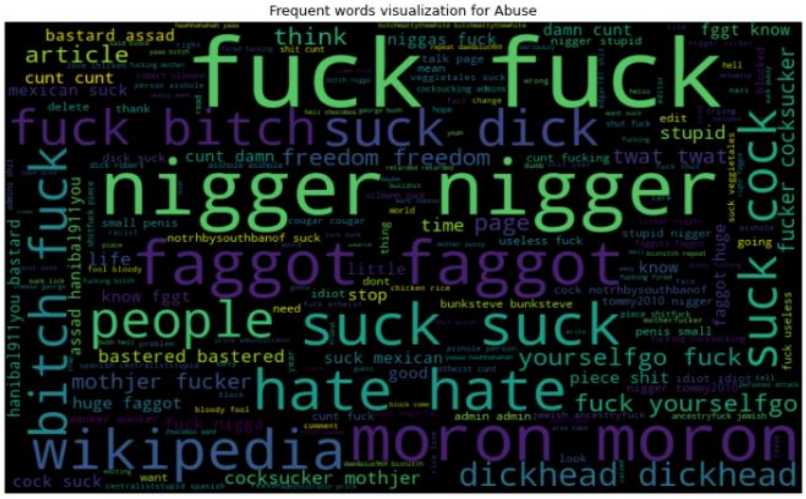
Frequent word visualization for Highly Malignant Comments

Frequent word visualization for Rude Comments

Frequent word visualization for Threat Comments



Frequent word visualization for Abuse Comments



Frequent word visualization for Loathe Comments

CONCLUSION

• Key Findings and Conclusions of the Study

This project proposed a Machine Learning Approach combined with Natural Language Processing for toxicity detection and its type identification in user comments. Finally, the best score of minimum log loss was achieved through Random Forest Classifier.

• Learning Outcomes of the Study in respect of Data Science

Through this project we were able to learn various Natural language processing techniques like lemmatization, stemming, removal of stopwords. We were also able to learn to convert strings into vectors through hash vectorizer. In this project we applied different evaluation metrics like log loss, hamming loss besides accuracy.

• Limitations of this work and Scope for Future Work

Some of the limitations can be:

- The model might not be able to understand sarcasm.
- Sometimes non negative comments can be wrongly classified as negative ones, leading to loss of constructive feedback or comments.