FLIP ROBO

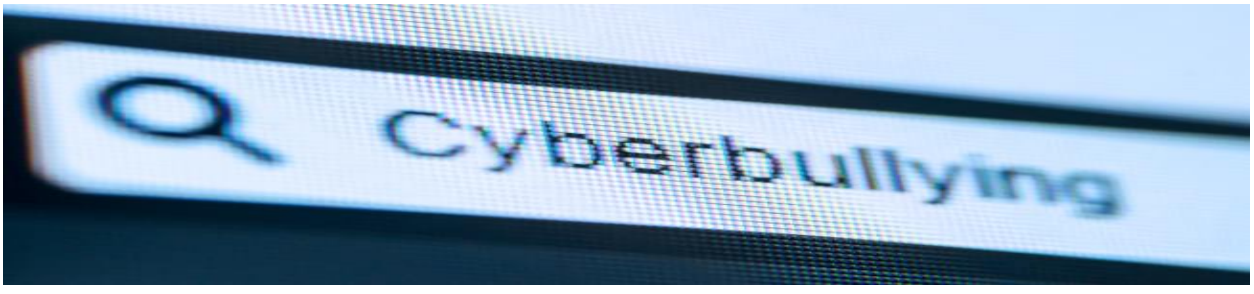
PROJECT REPORT

Submitted By

Aswini Arun Patil

NAME OF THE PROJECT

MALIGNANT COMMENTS CLASSIFIER PROJECT REPORT



ACKNOWLEDGMENT:

• Primarily I would like to thank God to being able to complete this project with success. Then I would like to express my special thanks of gratitude to my SME,

• And I am thankful I am part of flib rob technology of employee, who given me the golden opportunity to do this wonderful project on the given topic which is also help me in doing a lot of research and I came to know about so many new things, I am really thankful to flip robo.

DATE:11/07/2022

ASWINI A. PATIL

Data Science course

Institute: Data trained education

Internship: Flip Robo technology

@Bangalore

INTRODUCTION

- ## Business Problem Framing

  People may now express themselves broadly online because to the advent of social media. However, this has led in the growth of violence and hatred, making online environments unappealing to users. Despite the fact that academics have discovered that hatred is an issue across numerous platforms, there are no models for detecting online hate.

  Online hatred has been highlighted as a big problem on online social media platforms, and has been defined as abusive language, hostility, cyberbullying, hatefulness, and many other things. The most common venues for such toxic behaviors are social media platforms.

  On numerous social media sites, there has been a significant increase in incidences of cyberbullying and trolls. Many celebrities and influencers face blowback from the public and are subjected to nasty and disrespectful remarks. This may have a negative impact on anyone, resulting in sadness, mental disease, self-hatred, and suicide thoughts.

  Comments on the internet are hotbeds of hate and venom. Machine learning may be used to combat online anonymity, which has created a new venue for hostility and hate speech. The issue we were attempting to address was the labelling of internet remarks that were hostile to other users. This implies that insults directed towards third parties, such as celebrities, will be classified as non-offensive, whereas "u are an idiot" will be plainly offensive.

  Our objective is to create a prototype of an online hate and abuse comment classifier that can be used to categorise and manage hate and offensive remarks in order to prevent the spread of hatred and cyberbullying.

- **Conceptual Background of the Domain Problem**

  Online platforms and social media have evolved into places where individuals may freely discuss their beliefs without regard for race, and where people can share their thoughts and ideas with a large group of people.

  Through the creation of virtual networks and communities, social media is a computer-based technology that allows the exchange of ideas, opinions, and information. Social media is Internet-based by design, allowing people to share material quickly via electronic means. Personal information, documents, movies, and images are all included in the content. Users interact with social media using web-based software or applications on a computer, tablet, or smartphone.

  While social media is widely used in the United States and Europe, Asian nations such as India are at the top of the list. Social media is used by about 3.8 billion people.

  Some people or a motivated mob on this massive internet platform or online community willfully abuse others to prevent them from sharing their thoughts in a proper manner. They use filthy language to intimidate others, which is considered a form of ignominy in civilised society. When innocent people are intimidated by these mobs, they remain mute without saying anything. As a result, the disgusting mob's goal is excellently realized.

To address this issue, we are now developing a model that recognizes all foul language and foul terms, with the goal of preventing these mobs from using foul language in online communities or perhaps blocking them from using foul language altogether.

- **Review of Literature**

This project aims to build machine learning model that can classify good and offensive comments. So that offensive comments can be controlled and restricted from spreading hatred and cyber bullying

First, we do all the data pre-processing steps and do EDA to visualize the data graphically and after that we clean and prepare our data using Natural Language Processing (NLP), which is helpful in make a best machine learning model.

- **Motivation for the Problem Undertaken**

In current scenario, there has been increasing no. of cases of cyber bullying and trolls on social media platforms. My motivation for this project is to do some research on this thing and make a machine learning model. This helps clients to classify offensive comments and remove them from various platforms as well as take some legal action on offensive comments.

Analytical Problem Framing

- **Mathematical/ Analytical Modeling of the Problem**

- We use some mathematical, statistical and analytics approaches in this project. Which is described below:

    a. **Removing Stop Words**: Stop words are common words that structure a sentence. Words such as "I", "are", and "here" do not contribute to the sentiment (the rating in our case) of reviews. Hence, we decided to remove stopwords. We used NLTK's stopwords package to provide us with the list of stopwords.

    b. **Lemmatization:** Lemmatization is the process of converting a word to its base form. Lemmatization usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma. We applied this method on our comments text.

    c. **Term Frequency - Inverse Document Frequency (TF-IDF):** This approach to encoding is also fairly simplistic but carries more information than just the number of occurrences, like Bag-ofWords. TF-IDF is a numerical statistic that also reflects how important a word is to a document or paragraph. The scheme is split into two calculations, TF and IDF.

TF, or Term Frequency, is the number of times a term, *t*, appears divided by the number of terms in the document/paragraph, *d*.

$$tf_{t,d} = \frac{n_{t,d}}{Number\ of\ terms\ in\ the\ document}$$

Equation for Term Frequency

IDF, or Inverse Document Frequency, is a measure of how important the term is in the corpus. It's calculated by taking the log of the number of documents divided by the number of documents with the term *t* in it.

$$idf_t = \log \frac{number\ of\ documents}{number\ of\ documents\ with\ term\ 't'}$$

Equation for Inverse Document Frequency

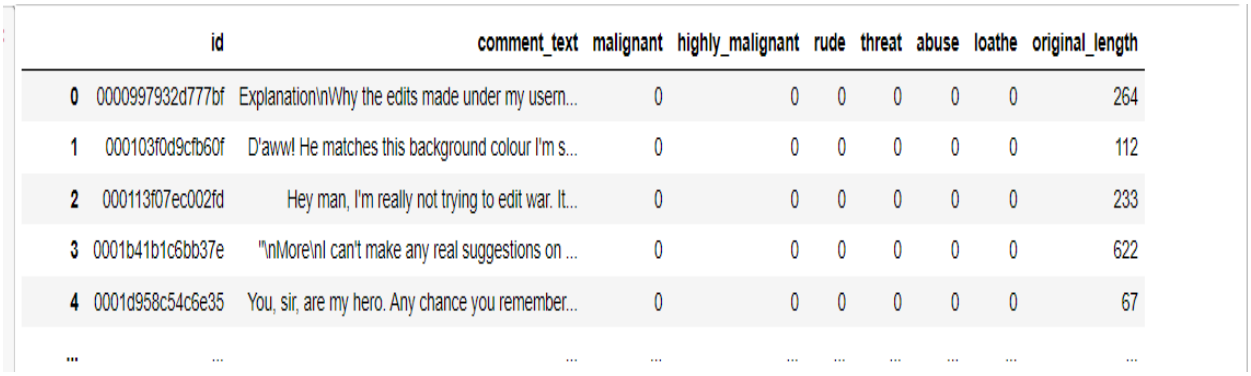
The TF-IDF encoding scheme outputs TF × IDF for each term in each review.

We convert our comments text into vectors using TF-IDF method.

d. **Synthetic minority oversampling technique (SMOTE):** In our dataset target variable is imbalanced and it provides inaccurate results during machine learning part. So, we used SMOTE method to cop up with that issue. SMOTE is an oversampling method and one of the most commonly used oversampling method to solve the imbalance classification problem. This method creates synthetic samples (not duplicate) of minority class. These synthetic records are generated by randomly selecting one or more of the k-nearest neighbors for each example in the minority class

- **Data Sources and their formats**

  In this project, client provides us two dataset, training dataset and testing dataset. Both datasets are in csv format.
  The dataset contains the training set, which has 1,59,571 samples and the test set which contains 1,53,164 samples. Training dataset contain 8 features which includes 'Id', 'Comments', 'Malignant', 'Highly malignant', 'Rude', 'Threat', 'Abuse' and 'Loathe'. While test dataset contains 2 features which include 'Id' and 'Comments'.
  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.
  Below is the snapshot of our training dataset:



	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	original_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
...	...	...	...	...	...	...	...	...	...

- **Data Preprocessing Done**

  For cleaning and preparation of the data, we use some techniques which are described below:

- We check null values present in the data, but there is no null value found in the dataset.
- There are six target features in this dataset so we combined them and make one target feature giving the name as label and also we do scaling for that feature.

- After data analysis, we clean our data using some NLP (Natural Language Processing) techniques. We convert comments text column in lower case and remove punctuation from that column like +, - , /, etc. because that type of punctuation badly affect our result. Then we remove digits from comments text.
- After that, we apply stop words and lemmatization techniques on comments text, which we already discussed above.
- At last, we use tf-idf and smote techniques for preparation of the data, which we also discussed above.

- **Data Inputs- Logic- Output Relationships**

  I have analyzed the input output logic with word cloud and I have word clouded the sentenced that as classified as foul language in every category. A tag/word cloud is a novelty visual representation of text data, typically used to depict keyword metadata on websites, or to visualize free form text. It's an image composed of words used in a particular text or subject, in which the size of each word indicates its frequency or importance.

  Code

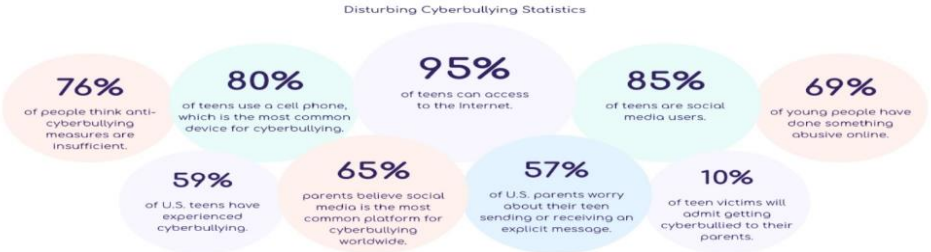
```
# WordCloud: Getting sense of loud words in each of the output labels.

cols = 3
rows = len(output_labels)//cols
if len(output_labels) % cols != 0:
    rows += 1

fig = plt.figure(figsize=(16,rows*cols*1.8))
fig.subplots_adjust(top=0.8, hspace=0.3)

p=1
for i in output_labels:
    word_cloud = WordCloud(height=650, width=800,
                           background_color="white",max_words=80).generate(' '.join(df.comment_text[df[i]==1]))
    ax = fig.add_subplot(rows,cols,p)
    ax.imshow(word_cloud)
    ax.set_title(f"WordCloud for {i} column",fontsize=14)
    for spine in ax.spines.values():
        spine.set_edgecolor('r')

    ax.set_xticks([])
    ax.set_yticks([])
    p += 1

fig.suptitle("WordCloud: Representation of Loud words in BAD COMMENTS",fontsize=16)
fig.tight_layout(pad=2)
plt.show()
```

Output:

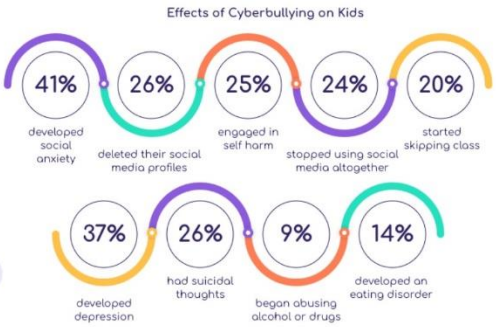
These are the comments that belongs to different type so which the help of word cloud we can see if there is abuse comment which type of words it contains and similar to other comments as well.

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

  Cyberbullying has become a growing problem in countries around the world. Essentially, cyberbullying doesn't differ much from the type of bullying that many children have unfortunately grown accustomed to in school. The only difference is that it takes place online.



Cyberbullying is a very serious issue affecting not just the young victims, but also the victims' families, the bully, and those who witness instances of cyberbullying. However, the effect of cyberbullying can be most detrimental to the victim, of course, as they may experience a number of emotional issues that affect their social and academic performance as well as their overall mental health.



- **Hardware and Software Requirements and Tools Used**

Hardware technology being used.
RAM : 8 GB
CPU : Intel(R) Core(TM) i3-7100U CPU @ 2.40GHz 2.40 GHz
Software technology being used.
Programming language : Python
Distribution : Anaconda Navigator
Browser based language shell : Jupyter Notebook
Libraries/Packages specifically being used.
Pandas, NumPy, matplotlib, seaborn, scikit-learn, pandas-profiling, missing no, NLTK

Model/s Development and Evaluation

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

I checked through the entire training dataset for any kind of missing values information and all these pre processing steps were repeated on the testing dataset as well.
Code:

```
df_train.isna().sum() # checking for missing values
```

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

- **Testing of Identified Approaches (Algorithms)**

  The classification algorithm that we used is:

  1) Gaussian Naïve Bayes
  2) Multinomial Naïve Bayes
  3) Logistic Regression
  4) Random Forest Classifier
  5) Linear Support Vector Classifier
  6) Ada Boost Classifier
  7) K Nearest Neighbors Classifier
  8) Decision Tree Classifier

  9)Bagging Classifier.

- **Run and Evaluate selected models**

  I created a classification function that included the evaluation metrics details for the generation of our Classification Machine Learning models.

```python
# 3. Training and Testing Model on our train dataset

# Creating a function to train and test model
def build_models(models,x,y,test_size=0.33,random_state=42):
    # spliting train test data using train_test_split
    x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=test_size,random_state=random_state)

    # training models using BinaryRelevance of problem transform
    for i in tqdm.tqdm(models,desc="Building Models"):
        start_time = timeit.default_timer()

        sys.stdout.write("\n=========================================================================================\n")
        sys.stdout.write(f"Current Model in Progress: {i} ")
        sys.stdout.write("\n=========================================================================================\n")

        br_clf = BinaryRelevance(classifier=models[i]["name"],require_dense=[True,True])
        print("Training: ",br_clf)
        br_clf.fit(x_train,y_train)

        print("Testing: ")
        predict_y = br_clf.predict(x_test)

        ham_loss = hamming_loss(y_test,predict_y)
        sys.stdout.write(f"\n\tHamming Loss   : {ham_loss}")

        ac_score = accuracy_score(y_test,predict_y)
        sys.stdout.write(f"\n\tAccuracy Score: {ac_score}")

        cl_report = classification_report(y_test,predict_y)
        sys.stdout.write(f"\n{cl_report}")

        end_time = timeit.default_timer()
        sys.stdout.write(f"Completed in [{end_time-start_time} sec.]")

        models[i]["trained"] = br_clf
        models[i]["hamming_loss"] = ham_loss
        models[i]["accuracy_score"] = ac_score
        models[i]["classification_report"] = cl_report
        models[i]["predict_y"] = predict_y
        models[i]["time_taken"] = end_time - start_time

        sys.stdout.write("\n=========================================================================================\n")

    models["x_train"] = x_train
    models["y_train"] = y_train
    models["x_test"] = x_test
    models["y_test"] = y_test

    return models
```

Code:

```python
# Preparing the list of models for classification purpose
models = {"GaussianNB": {"name": GaussianNB()},
          "MultinomialNB": {"name": MultinomialNB()},
          "Logistic Regression": {"name": LogisticRegression()},
          "Random Forest Classifier": {"name": RandomForestClassifier()},
          "Support Vector Classifier": {"name": LinearSVC(max_iter = 3000)},
          "Ada Boost Classifier": {"name": AdaBoostClassifier()},
          "K Nearest Neighbors Classifier": {"name": KNeighborsClassifier()},
          "Decision Tree Classifier": {"name": DecisionTreeClassifier()},
          "Bagging Classifier": {"name": BaggingClassifier(base_estimator=LinearSVC())},
         }

# Taking one forth of the total data for training and testing purpose
half = len(df)//4
trained_models = build_models(models,X[:half,:],Y[:half,:])
```

Output:

```
Building Models: 100% 9/9 [1:26:58<00:00, 756.96s/it]

=========================================================================================
Current Model in Progress: GaussianNB
=========================================================================================
Training:  BinaryRelevance(classifier=GaussianNB(), require_dense=[True, True])
Testing:

        Hamming Loss   : 0.21560957083175086
        Accuracy Score: 0.4729965818458033
              precision    recall  f1-score   support

           0       0.16      0.79      0.26      1281
           1       0.08      0.46      0.13       150
           2       0.11      0.71      0.19       724
           3       0.02      0.25      0.03        44
           4       0.10      0.65      0.17       650
           5       0.04      0.46      0.07       109

   micro avg       0.11      0.70      0.20      2958
   macro avg       0.08      0.55      0.14      2958
weighted avg       0.12      0.70      0.21      2958
 samples avg       0.05      0.07      0.05      2958
Completed in [27.4159962999999 sec.]
=========================================================================================
```

Observation:
From the above model comparison, it is clear that Linear Support Vector Classifier performs better with Accuracy Score: 91.35586783137106% and Hamming Loss: 1.9977212305355107% than the other classification models. Therefore, I am now going to use Linear Support Vector Classifier for further Hyperparameter tuning process. With the help of hyperparameter tuning process I will be trying my best to increase the accuracy score of our final classification machine learning model.

- **Key Metrics for success in solving problem under consideration**

Hyperparameter Tuning

```
# Choosing Linear Support Vector Classifier model

fmod_param = {'estimator__penalty' : ['l1', 'l2'],
              'estimator__loss' : ['hinge', 'squared_hinge'],
              'estimator__multi_class' : ['ovr', 'crammer_singer'],
              'estimator__random_state' : [42, 72, 111]
             }
SVC = OneVsRestClassifier(LinearSVC())
GSCV = GridSearchCV(SVC, fmod_param, cv=3)
x_train,x_test,y_train,y_test = train_test_split(X[:half,:], Y[:half,:], test_size=0.30, random_state=42)
GSCV.fit(x_train,y_train)
GSCV.best_params_
```

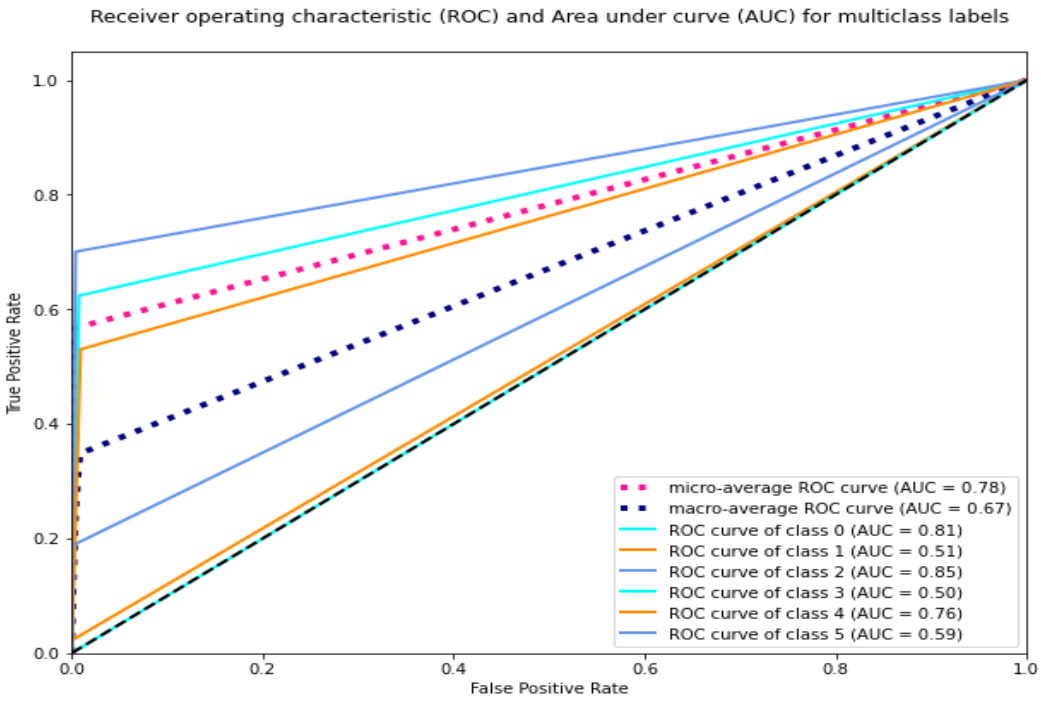
```
{'estimator__loss': 'hinge',
 'estimator__multi_class': 'ovr',
 'estimator__penalty': 'l2',
 'estimator__random_state': 42}
```

Final Classification Model details:

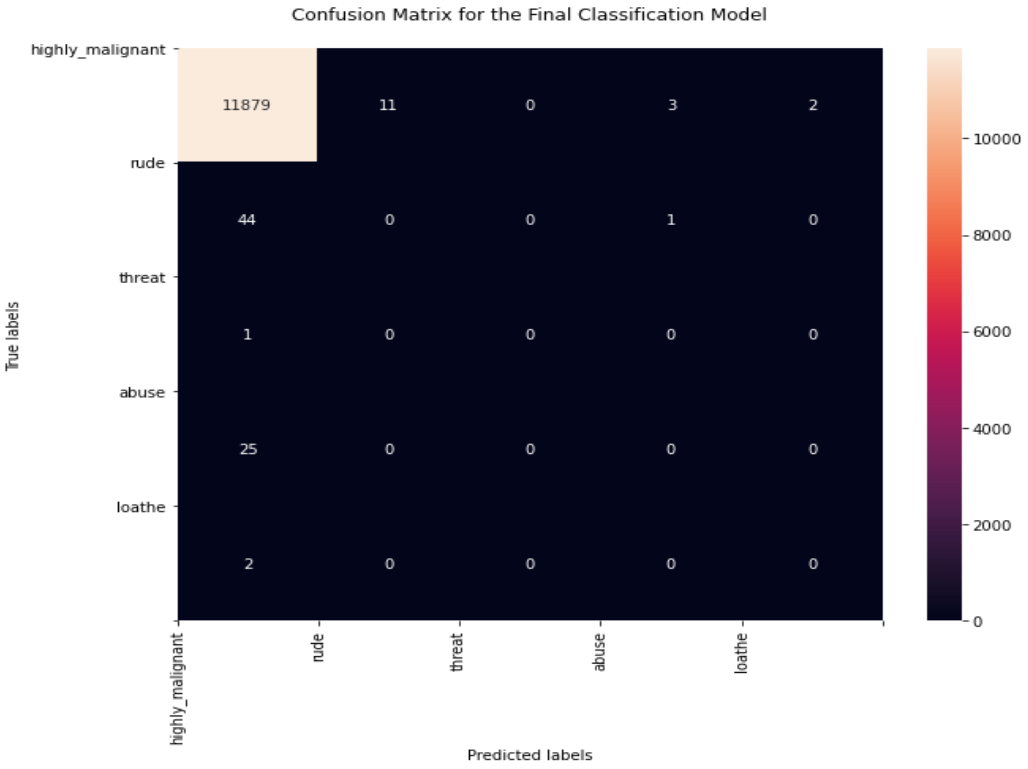
```
Final_Model = OneVsRestClassifier(LinearSVC(loss='hinge', multi_class='ovr', penalty='l2', random_state=42))
Classifier = Final_Model.fit(x_train, y_train)
fmod_pred = Final_Model.predict(x_test)
fmod_acc = (accuracy_score(y_test, fmod_pred))*100
print("Accuracy score for the Best Model is:", fmod_acc)
h_loss = hamming_loss(y_test,fmod_pred)*100
print("Hamming loss for the Best Model is:", h_loss)
```

```
Accuracy score for the Best Model is: 91.51069518716578
Hamming loss for the Best Model is: 1.9593917112299464
```

AUC ROC Curve for Final Model:



Confusion Matrix for Final Model



Saving the best model:

```
# selecting the best model
best_model = trained_models['Support Vector Classifier']['trained']

# saving the best classification model
joblib.dump(best_model,open('Malignant_comments_classifier.pkl','wb'))
```

Final predicted data frame:

	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe
0	yo bitch ja rule succes ever what hate sad mof...	0	0	0	0	0	0
1	rfc titl fine imo	0	0	0	0	0	0
2	sourc zaw ashton lapland	0	0	0	0	0	0
3	look back sourc inform updat correct form gues...	0	0	0	0	0	0
4	anonym edit articl	0	0	0	0	0	0
...	...	...	...	...	...	...	...
153159	total agre stuff noth long crap	0	0	0	0	0	0
153160	throw field home plate get faster throw cut ma...	0	0	0	0	0	0
153161	okinotorishima categori see chang agre correct...	0	0	0	0	0	0
153162	one found nation eu germani law return quit si...	0	0	0	0	0	0
153163	stop alreadi bullshit welcom fool think kind e...	0	0	0	0	0	0

153164 rows × 7 columns

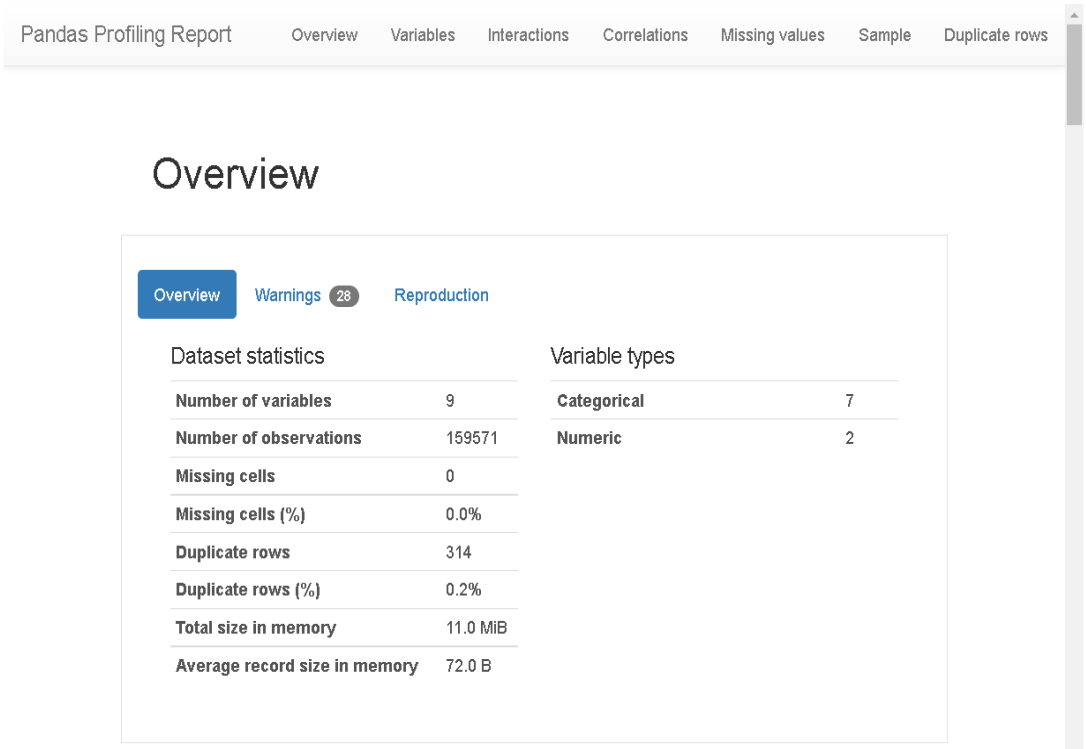
- # Visualizations

I used the pandas profiling feature to generate an initial detailed report on my data frame values. It gives us various information on the rendered dataset like the correlations, missing values, duplicate rows, variable types, memory size etc. This assists us in further detailed visualization separating each part one by one comparing and research for the impacts on the prediction of our target label from all the available feature columns.

Code:

```
pandas_profiling.ProfileReport(df)
```

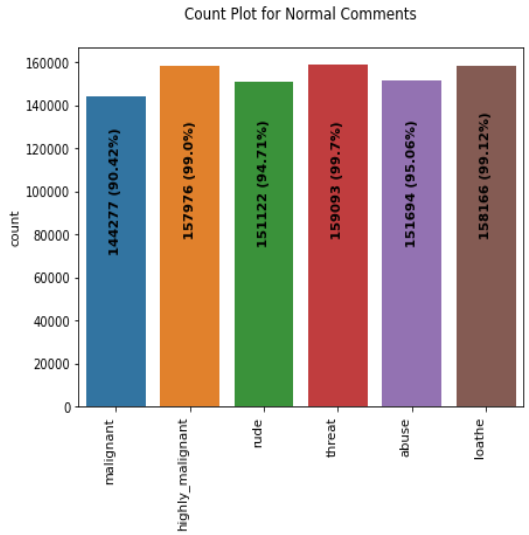
Output:

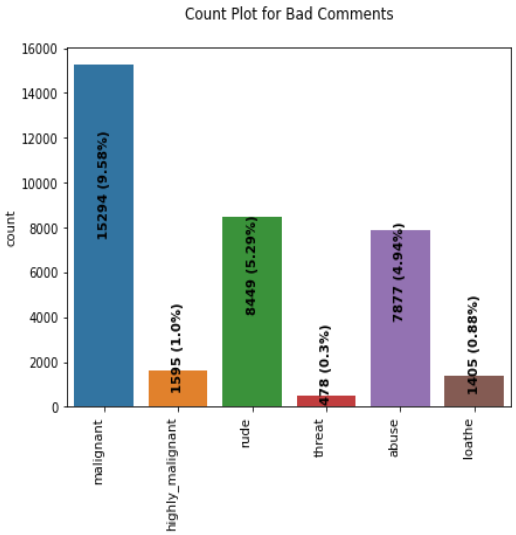


Code:

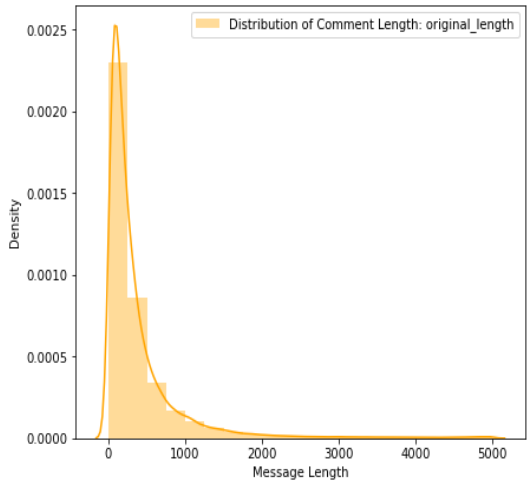
```
# comparing normal comments and bad comments using count plot

fig, ax = plt.subplots(1,2,figsize=(15,5))

for i in range(2):
    sns.countplot(data=df[output_labels][df[output_labels]==i], ax=ax[i])
    if i == 0:
        ax[i].set_title("Count Plot for Normal Comments\n")
    else:
        ax[i].set_title("Count Plot for Bad Comments\n")

    ax[i].set_xticklabels(output_labels, rotation=90, ha="right")
    p=0
    for prop in ax[i].patches:
        count = prop.get_height()
        s = f"{count} ({round(count*100/len(df),2)}%)"
        ax[i].text(p,count/2,s,rotation=90, ha="center", fontweight="bold")
        p += 1

plt.show()
```

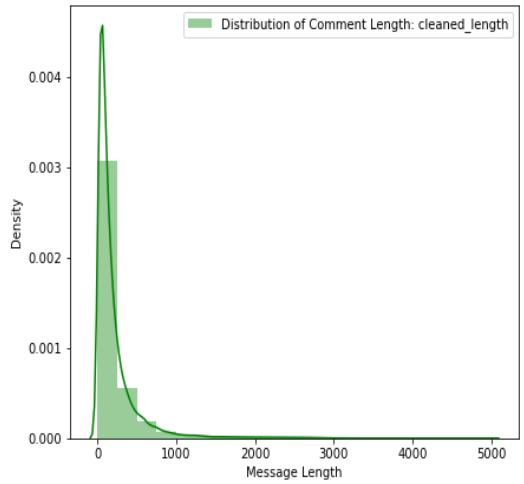
Output:





Code:

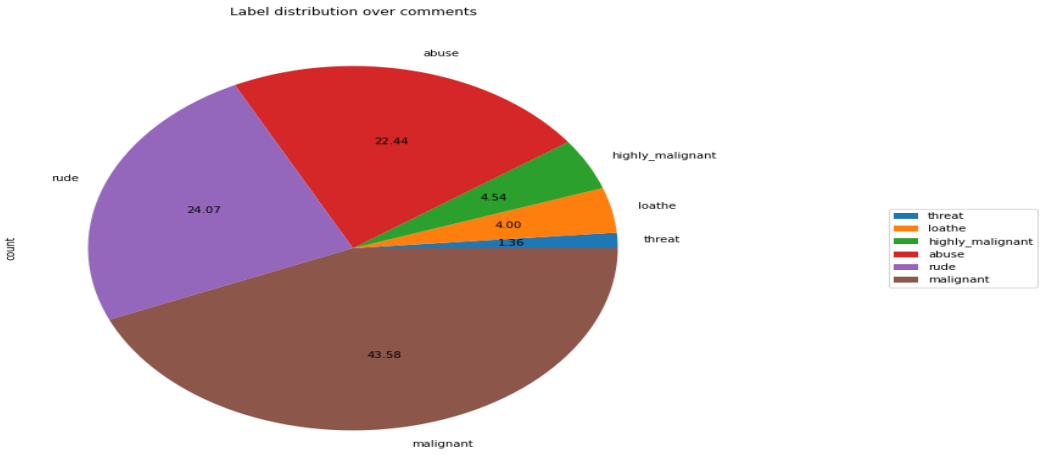




Code:

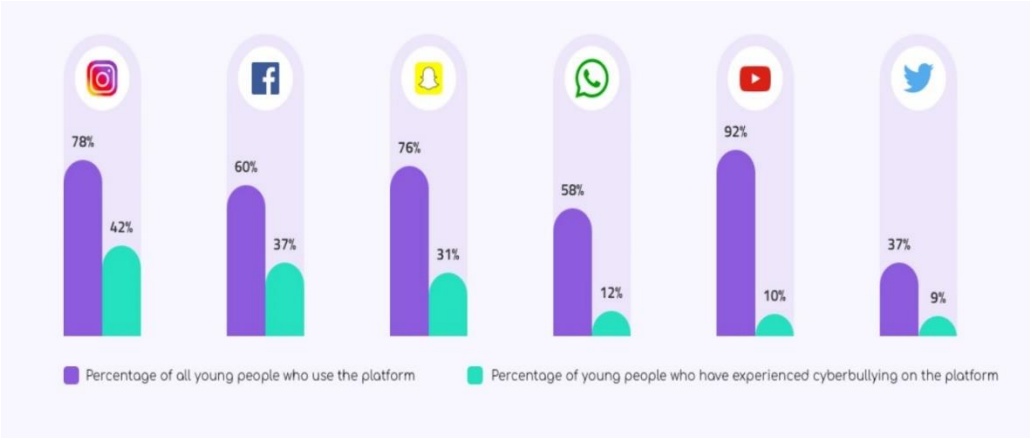
```
# Visualizing the label distribution of comments using pie chart

comments_labels = ['malignant', 'highly_malignant', 'rude', 'threat', 'abuse', 'loathe']
df_distribution = df_train[comments_labels].sum()\
                            .to_frame()\
                            .rename(columns={0: 'count'})\
                            .sort_values('count')

df_distribution.plot.pie(y = 'count', title = 'Label distribution over comments', autopct='%.2f', figsize = (15, 10))\
                            .legend(loc='center left', bbox_to_anchor=(1.3, 0.5))
```

Output:



CONCLUSION:

The finding of the study is that only few users over online use unparliamentary language. And most of these sentences have more stop words and are being quite long. As discussed before few motivated disrespectful crowds use these foul languages in the online forum to bully the people around and to stop them from doing these things that they are not supposed to do. Our study helps the online forums and social media to induce a ban to profanity or usage of profanity over these forums.



- **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.
My point of view from my project is that we need to use proper words which are respectful and also avoid using abusive, vulgar and worst words in social media. It can cause many problems which could affect our lives. Try to be polite, calm and composed while handling stress and negativity and one of the best solutions is to avoid it and overcoming in a positive manner.

- **Limitations of this work and Scope for Future Work**

  Problems faced while working in this project:

  - More computational power was required as it took more than 2 hours
  - Imbalanced dataset and bad comment texts
  - Good parameters could not be obtained using hyperparameter tuning as time was consumed more

  Areas of improvement:

  - Could be provided with a good dataset which does not take more time.
  - Less time complexity
  - Providing a proper balanced dataset with less errors.

Thank You ☺