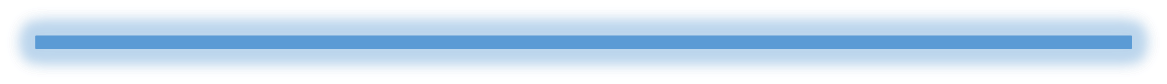


Malignant comment

Submitted by: Pailla kusaraju



*cknowledgements*

The completion of this theais atudy would not have been possible without the support and extensive knowledge of several website.

# I cannot begin without expressing my thanks to my #ME from



//Ip 6o6o Pec6no/oplea, Whuah&oo Mazg,

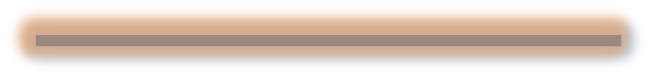
for the valuable expefience and insight in conducting and wfiting scientific reports. Thank you for always answefing all questions and giving me helpful practical suggestions.

Someone whose help cannot be overestimated ia my advisor from Career Coach, 6e. 6eepl6a #6a«ma



Thank you for always taking your time to support and guide me through the project. Without your extensive knowledge and insightful suggestions, the final result would not be what it ia today.

# Finally, I would like to express my sincere gratitude to my family for their constant encouragement and support throughout my time on this thesis.



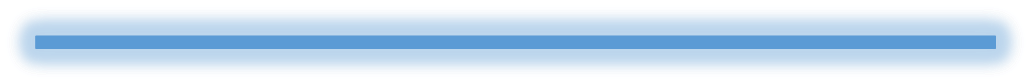
## The proliferation of social media enables people to express their opinions widely online. However, at the same time, this has resulted in the emergence of conflict and hate, making online environments uninviting for users. Although researchers have found that hate is a problem across multiple platforms, 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 unoffensive, 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.



### In this project, we will develop and evaluate the performance and predictability of trained and tested models based on comments which is provide by flip robo techknology. Once we get a good fit, we will apply on our test data.

In here we will use various classification algorithm to predict our target. Let's have an overview of the algorithms we will use for our predictions. To read more about these algorithms , just click on the algorithms name.

* [LogisticRegression](https://www.google.com/search?q=linear%2Bregression&rlz=1C1CHBF_enIN997IN998&oq&aqs=chrome.1.69i59i450l8.734952339j1j15&sourceid=chrome&ie=UTF-8):- Logistic regression analysis is valuable for predicting the likelihood of an event. It helps determine the probabilities between any two classes. In a nutshell, by looking at historical data, logistic regression can predict whether: An email is a spam.
* [DecisionTreeClassifier](https://www.google.com/search?q=about%2BDecisionTreeRegressor&rlz=1C1CHBF_enIN997IN998&ei=7kG5YoWNM6fA3LUPqcGy8AQ&ved=0ahUKEwiFvLry9Mz4AhUnILcAHamgDE4Q4dUDCA4&uact=5&oq=about%2BDecisionTreeRegressor&gs_lcp=Cgdnd3Mtd2l6EAM6BAgAEA1KBAhBGABKBAhGGABQAFjfEWDdFWgAcAF4AIABqQKIAZYLkgEDMi02mAEAoAEBwAEB&sclient=gws-wiz):- Decision trees **help you to evaluate your options**. Decision Trees are excellent tools for helping

**highly\_malignant** Binary column with labels for highly malignant text.

## **rude** Binary column with labels for comments that are rude in nat **threat** Binary column with labels for threatening context in the com **abuse** Binary column with labels with abusive behaviour.

**loathe** Label to comments that are full of loathe and hatred.



### For the purpose of the project the dataset has been preprocessed as follows:

 Checking shape of the dataframe

 Checking Missing Value

 Checking which type of data stored in each columns

 Text processing

 Plot Word cloud

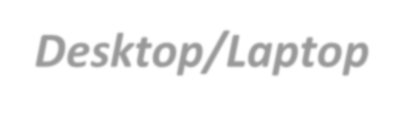
 Visualization

 Describing the dataset

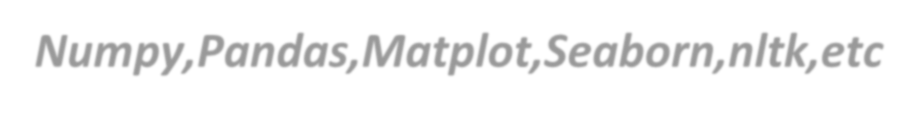
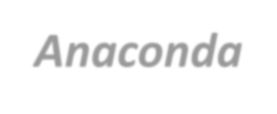
 Checking correlation and using heatmap for better understanding

We’ll now open a python 3 Jupyter Notebook and execute the following code snippet to load the dataset and remove the non- essential features. Recieving a success message if the actions were correclty performed.

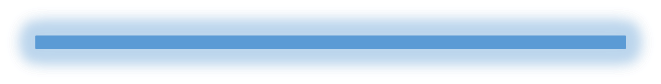
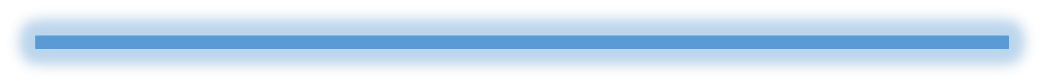
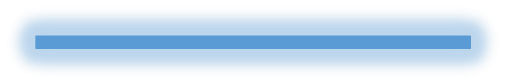




***Desktop/Laptop***



***Anaconda Numpy,Pandas,Matplot,Seaborn,nltk,etc***



**There are 2 primary ways of handling missing values:**

* + Deleting the Missing values:-Generally, this approach is not recommended. It is one of the quick and dirty techniques one can use to deal with missing values.
  + Imputing the Missing Values:- There are different ways of replacing the missing values
    - Replacing With Mean
    - Replacing With Mode
    - Replacing With Median,etc.
    - We are free from missing value otherwise it is very important step for model building



### I have analysed yhe 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 from 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.

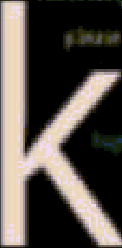
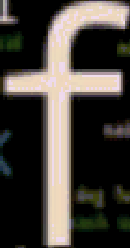
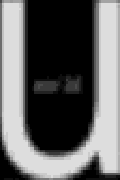




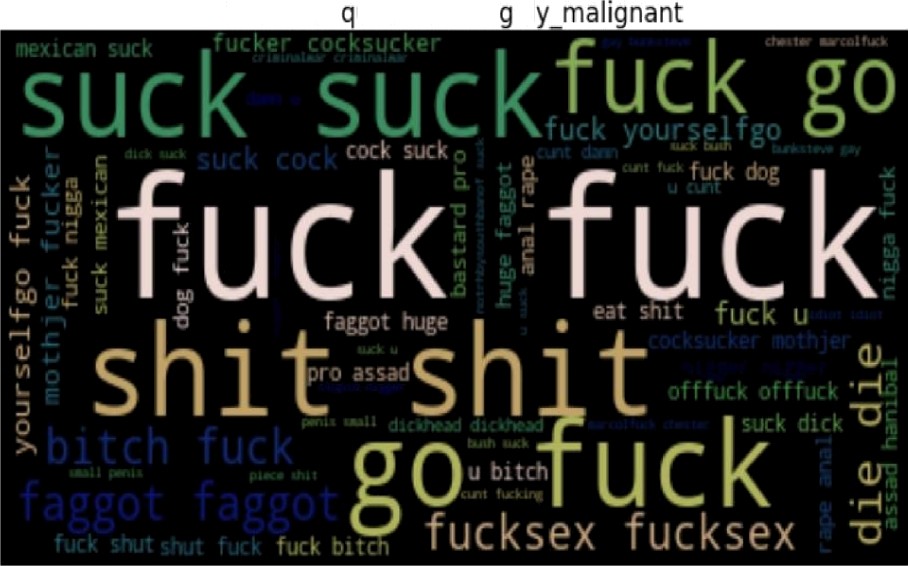
df m:df.loc[:,['comment text','malignant']]

we loud (df\_n, 'nazignant ' )



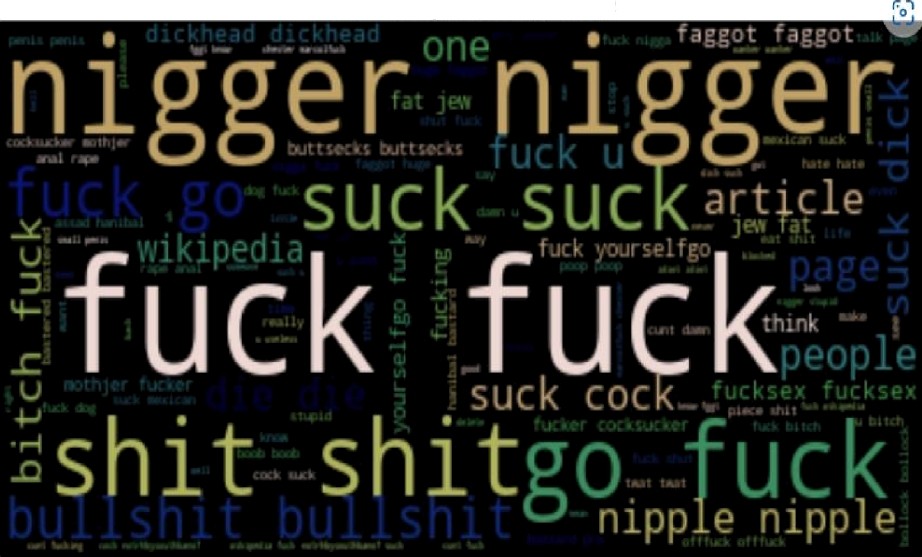
df\_hm\*df.loc{:,['comment\_text','highly\_mallgnant']]

wcloud(df\_hm,'highly\_malignant')



Words he uented in hi hl

df r:df.loc[:,['comment text','rude']] wcloud(df r,'rude')

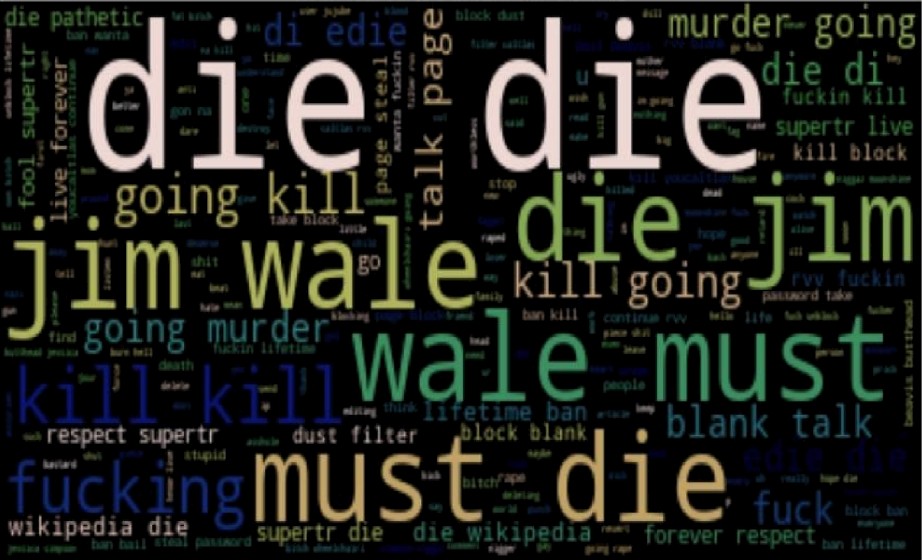


Words frequented in rude

df t:df.loc[:,['comment text','threat']]

uc loud (df I, ' threat )

Words frRquented in thrRat



df\_a:df.loc[:,['comment\_text','abuse']] wcloud(df\_a, 'abuse')

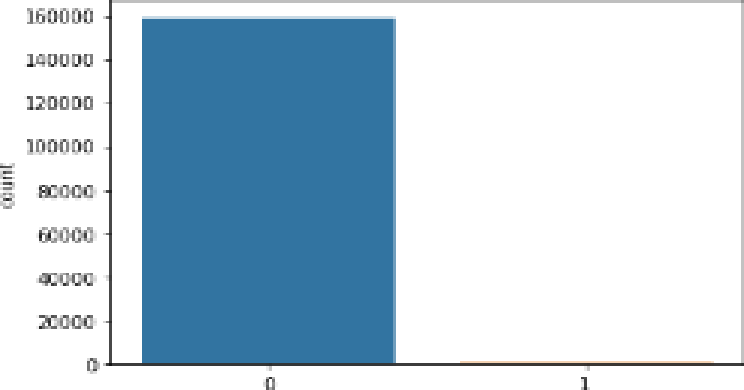
Words frequented in abuse



df T zdf. toe [ : , [ ' coouent text ' , ' 1oat he ' ] ] ucJoud (df\_1, ' I DBthe ' )

Words frequented in loathe

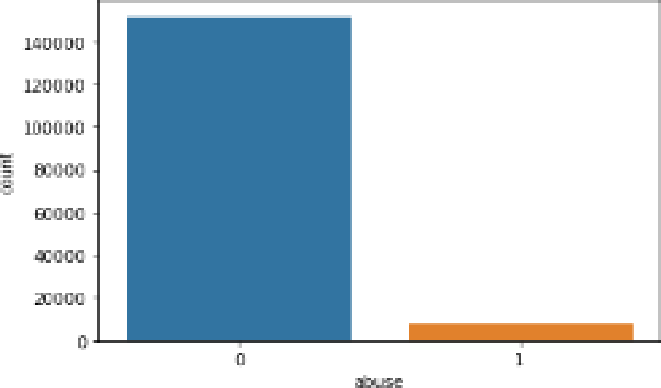
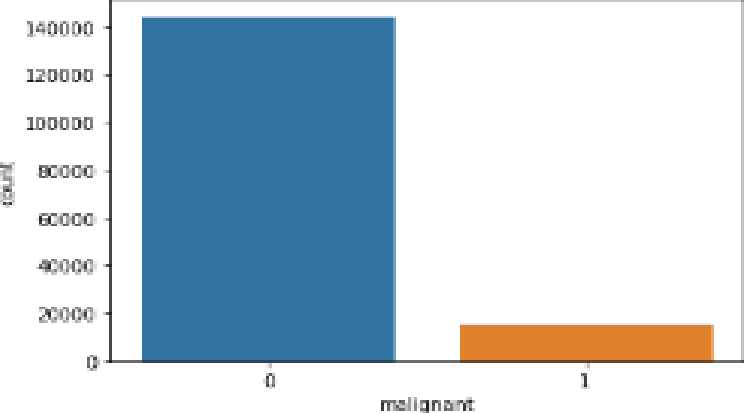


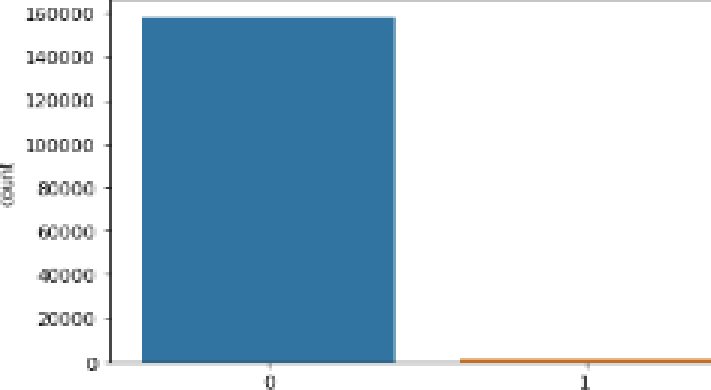
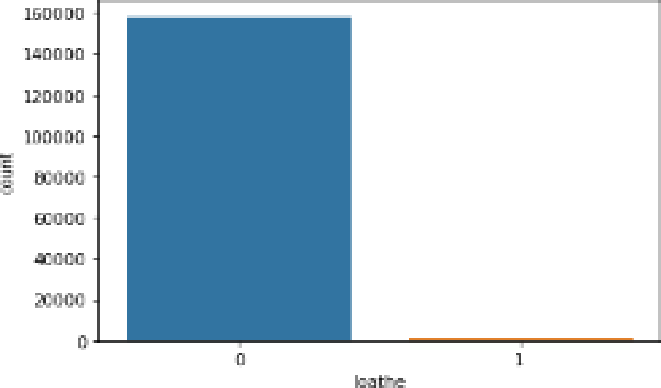


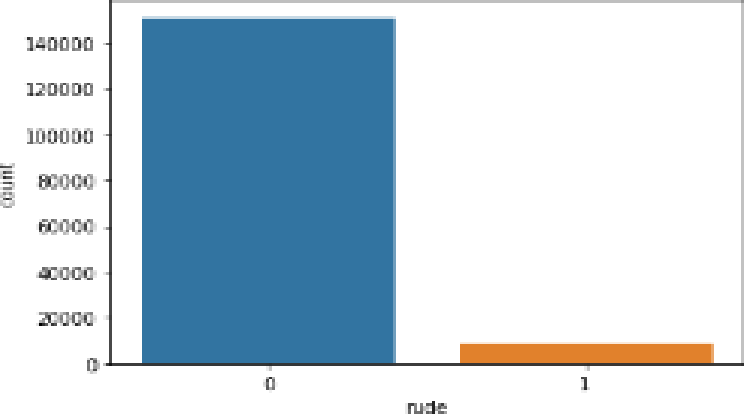
fE•at zdf. co1umns [ 1 : ] for cot 1n feat :

sns.countplot(df[col])

pit . sho\•f( )







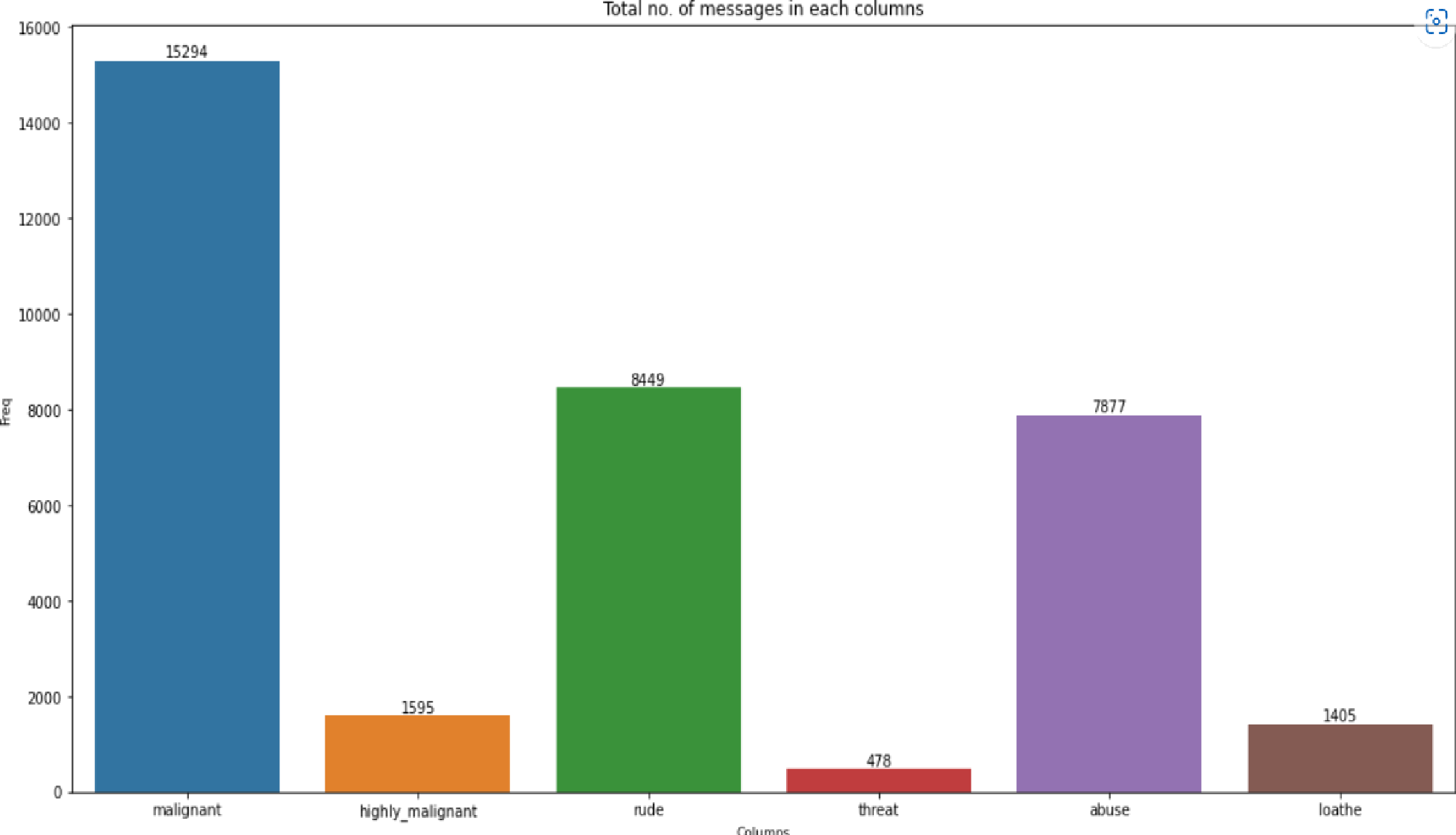
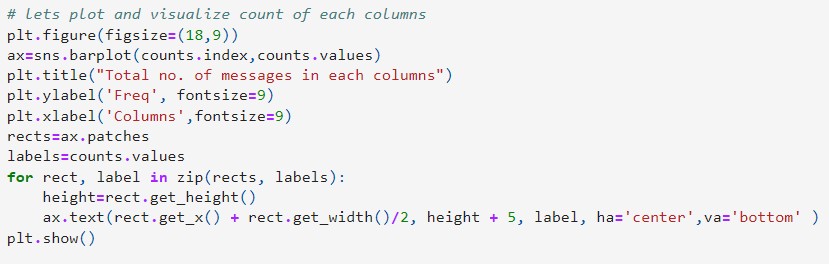


* Here in the first graph of malignant we can clearly observe that most of the messages are not malignant.
* In the second image we can clearly observe that there are very less highly malignant messages.
* Same in third picture there are few rude comments in the dataset.
* In 4th we can clearly see that there are very few cases/almost negligible of threat comments
* In 5th image we can clearly see that there are some messages with abusive language.
* While in the sixth image we can clearly see that there are very few cases of loathe messages.
* In 7th image we can see the no. of words in each rows
* In 8th image we can see the cleaned no. of remaining words in each row.









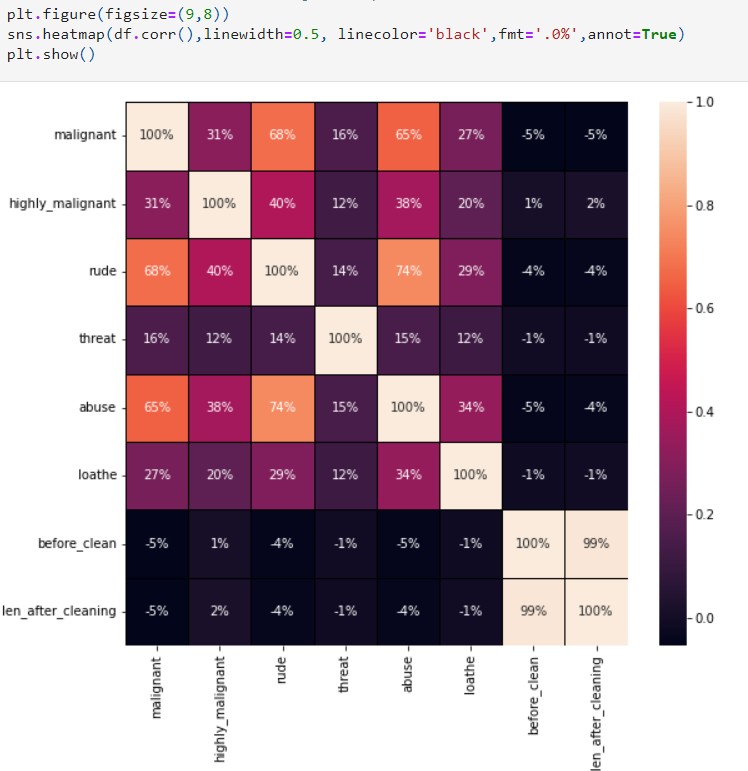
# *lets check the statisticot description of a I I the coLunns*

dT . desc vibe( )

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | malignant | highIy\_maIignant | rude | threat | abuse | laathe | before\_cIean | Ien\_after\_cIeaning |
| count | 159571.000000 | 159571.000000 | 159571.00000D | 159571.00D000 | 159571.000000 | 159571.000000 | 159571.000000 | 159571.000000 |
| mean | 0.095844 | 0.009996 | 0.05294B | 0.002996 | 0.@9364 | 0.rrgs0s | 394.138847 | 241.114238 |
| std | 0.294379 | 0.099477 | 0.223931 | 0.054650 | 0.216627 | 0.09J420 | 590.725381 | 377.602191 |
| min | 0.000000 | 0.000000 | 0.00000D | 0.00D000 | 0.000000 | 0.000000 | 5000000 | 0.000000 |
| 25'K | 0.000000 | 0.000000 | 0.00000D | 0.00D000 | 0.000000 | 0.000000 | 96.000000 | 56.000000 |
| 50'K | 0.000000 | 0.000000 | 0.00000D | 0.00D000 | 0.000000 | 0.000000 | 205000000 | 123.000000 |
| 75'K | 0.000000 | 0.000000 | 0.00000D | 0.00D000 | 0.000000 | 0.000000 | 436.000000 | 263.000000 |
| max | t000OJO | t000000 | 1.00000D | 100D000 | 1.000000 | t000000 | 5000.000000 | SOOOOOOOOO |



* Here we can see that only 2 values are present in all the columns i.e. 0 and 1.
* Low score of standard devaiation tells us that the data is not spreaded.
* there is difference in mean and median which tells us that some skewness is present.
* very low difference in 75% and max shows that there are no outliers present in the dataset.

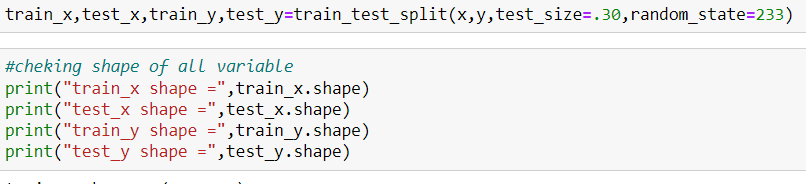








The [Train\_Test\_Split](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html) is a technique for evaluating the performance of a machine learning algorithm. It can be used for classification or regression problems and can be used for any supervised learning algorithm. The procedure involves taking a dataset and dividing it into two subsets. The first subset is used to fit the model and is referred to as the training dataset. The second subset is not used to train the model; instead, the input element of the dataset is provided to the model, then predictions are made and compared to the expected values. This second dataset is referred to as the test dataset.









classifier: LinearSVC

Jaccard score: 0.8478403520284093

Accuracy score: 0.9176554144385026

f1\_score: 0.9176554144385026

Precision : 0.9176554144385026

Recall: 0.9176554144385026

Hamming loss: 0.08234458556149733 Confusion matrix:

[[[ 2850 2018]

[ 257 42747]]

|  |  |
| --- | --- |
| [[45344 | 620] |
| [ 1573 | 335]] |
| [[46511 | 317] |
| [ 907 | 137]] |

[[45861 748]

[ 657 606]]

[[47134 210]

[ 431 97]]

[[47729 27]

[ 108 8]]

[[47861 2]

[ 9 0]]]

========================================

classifier: LogisticRegression Jaccard score: 0.8443875093910732

Accuracy score: 0.9156291778074866

f1\_score: 0.9156291778074866

Precision : 0.9156291778074866

Recall: 0.9156291778074866

Hamming loss: 0.08437082219251336 Confusion matrix:

[[[ 2130 2738]

[ 97 42907]]

|  |  |
| --- | --- |
| [[45521 | 443] |
| [ 1685 | 223]] |
| [[46676 | 152] |
| [ 991 | 53]] |
| [[45991 | 618] |
| [ 661 | 602]] |
| [[47261 | 83] |
| [ 481 | 47]] |
| [[47751 | 5] |
| [ 115 | 1]] |
| [[47863 | 0] |
| [ 9 | 0]]] |

========================================

classifier: MultinomialNB

Jaccard score: 0.8337546924078756

Accuracy score: 0.9093415775401069

f1\_score: 0.9093415775401069

Precision : 0.9093415775401069

Recall: 0.9093415775401069

Hamming loss: 0.09065842245989304 Confusion matrix:

[[[ 1271 3597]

[ 23 42981]]

|  |  |
| --- | --- |
| [[45799 | 165] |
| [ 1852 | 56]] |
| [[46823 | 5] |
| [ 1040 | 4]] |
| [[46038 | 571] |
| [ 774 | 489]] |
| [[47342 | 2] |
| [ 526 | 2]] |
| [[47756 | 0] |

[ 116 0]]

[[47863 0]

[ 9 0]]]

========================================

classifier: SGDClassifier

Jaccard score: 0.8368858277535829

Accuracy score: 0.9112007018716578

f1\_score: 0.9112007018716578

Precision : 0.9112007018716578

Recall: 0.9112007018716578

Hamming loss: 0.08879929812834225 Confusion matrix:

[[[ 1407 3461]

[ 10 42994]]

|  |  |
| --- | --- |
| [[45906 | 58] |
| [ 1899 | 9]] |
| [[46731 | 97] |
| [ 1001 | 43]] |
| [[46075 | 534] |
| [ 739 | 524]] |
| [[47268 | 76] |
| [ 484 | 44]] |
| [[47731 | 25] |
| [ 109 | 7]] |
| [[47863 | 0] |
| [ 9 | 0]]] |

========================================

classifier: LGBMClassifier

Jaccard score: 0.8471630042636931

Accuracy score: 0.9172585227272727

f1\_score: 0.9172585227272727

Precision : 0.9172585227272727

Recall: 0.9172585227272727

Hamming loss: 0.08274147727272728 Confusion matrix:

[[[ 2485 2383]

[ 161 42843]]

|  |  |
| --- | --- |
| [[45585 | 379] |
| [ 1750 | 158]] |
| [[46623 | 205] |
| [ 935 | 109]] |

[[45822 787]

[ 585 678]]

[[47207 137]

[ 428 100]]

[[47694 62]

[ 93 23]]

[[47855 8]

[ 9 0]]]

========================================

classifier: RandomForestClassifier Jaccard score: 0.8460589233379608

Accuracy score: 0.9166109625668449

f1\_score: 0.9166109625668449

Precision : 0.9166109625668449

Recall: 0.9166109625668449

Hamming loss: 0.08338903743315508 Confusion matrix:

[[[ 2429 2439]

[ 195 42809]]

|  |  |
| --- | --- |
| [[45646 | 318] |
| [ 1746 | 162]] |
| [[46562 | 266] |
| [ 886 | 158]] |

[[45775 834]

[ 599 664]]

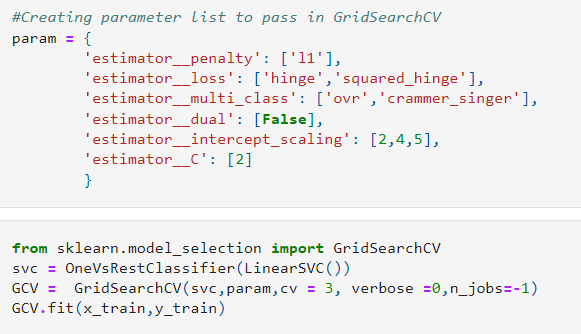
|  |  |
| --- | --- |
| [[47217 | 127] |
| [ 444 | 84]] |
| [[47748 | 8] |
| [ 113 | 3]] |
| [[47863 | 0] |
| [ 9 | 0]]] |

========================================

we get best accuracy score from **LinearSVC**









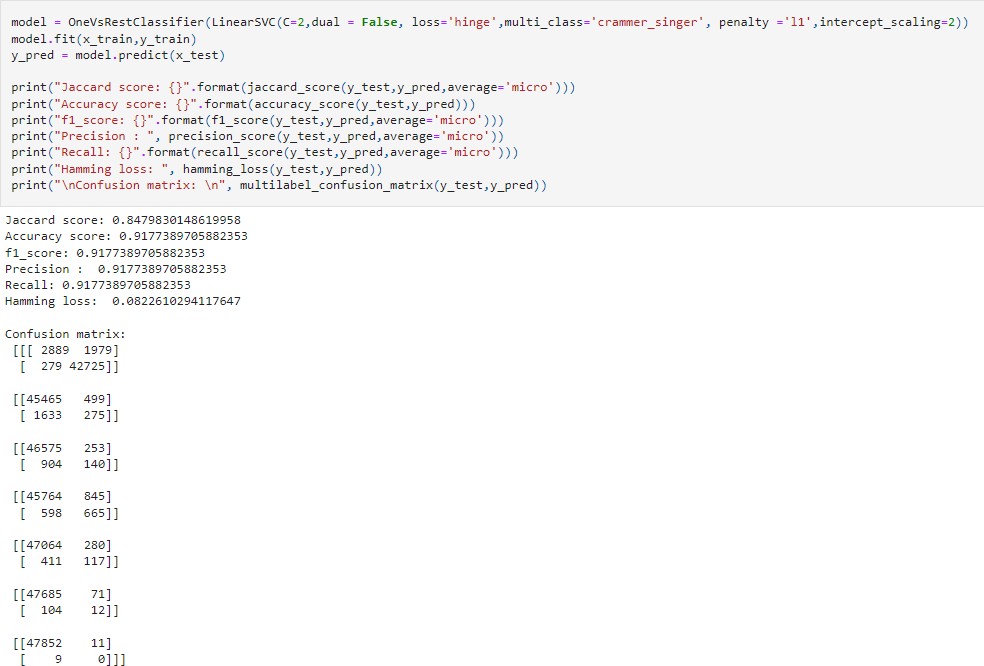
GridSearchCV(cv=3, estimator=OneVsRestClassifier(estimator=LinearSVC()), n\_jobs=-1,

param\_grid={'estimator C': [2], 'estimator dual': [False], 'estimator intercept\_scaling': [2, 4, 5],

'estimator loss': ['hinge', 'squared\_hinge'], 'estimator multi\_class': ['ovr', 'crammer\_singer'], 'estimator penalty': ['l1']})



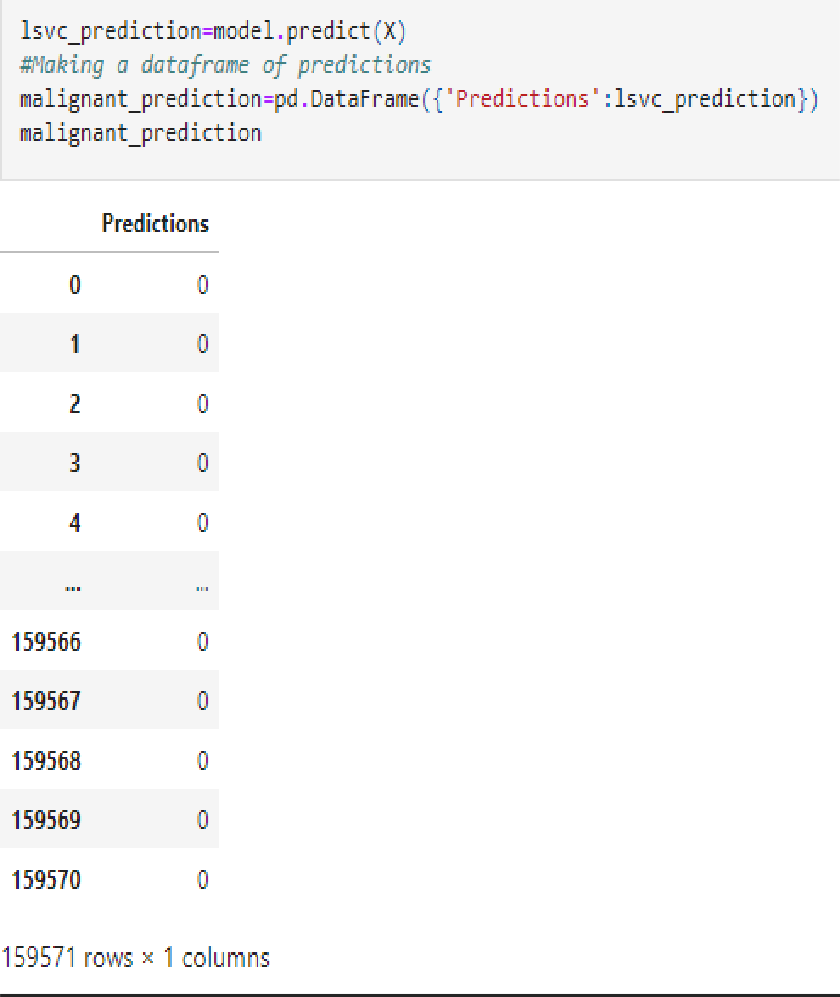


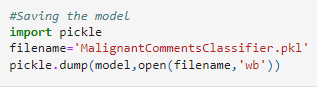


### Here we have successfully improved accuracy score from 91.76 to 91.77%.



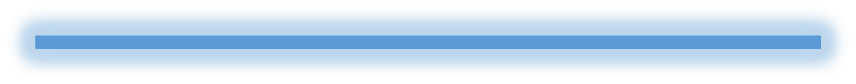








* LinearSVM and Random Forest models perform best.
* At first we get only 91.76% accuracy from LinearSVM But after parameter tuning we get 91.77% accuracy.There is no defference in accuracy score after parameter tuning.
* Using hyper parameter tunning we can improve our model accuracy, But here in this model the accuracy did not increased.
* It is always advised to all of us that atleast we need to use 5 Algorithm in order to figure out which one is performing best among them and we choose that one and we send that for hyper parameter tuning to know that best parameter .



For any of machine learning project my suggestion is first you have to understand the problem on ground level .if you don’t allow yourself to work with diligence .if you don’ t work harder anything that you are doing or will do , not only in case of machine learning but also in life cycle would be futile. Maybe, my endeavour assist you when ever you will get stuck

* For future improvements, following step we thought to took-

 Replacing model with a latest/different model

 Using other robust datasets

 More focus on NLP properties

* + It would seem that better performance might be achieved if multiple learners were combined.



I have also used few external resources that helped me to complete this project successfully.Below are the external resources that were used to fulfill my project.



