FLIP ROBO

MALIGNANT AND NON-MALIGNANT

Submitted By:

Sharukh Ansari

Acknowledgement

I’m extremely grateful to SME Khushboo Garg who give me an opportunity to work on this project and provide all the data required and information about the dataset which is crucial for completing this project and guided us in every possible manner. I’d also like to show my gratitude to live Data trained support who help me with this project and guided me in taking care of the skewness.

Introduction:

The rise of social media has allowed users to express their opinions widely, but this also resulted in the emergence of hate speech and conflict. This has led to the uninviting of users.

Online hate, which is often referred to as abusive language, is a major threat to the stability of social media platforms. There are many factors that contribute to this behaviour. The rise of online trolls and cyberbullying has been a distressing experience for many celebrities and influencers. It can lead to depression, self-hatred, and suicidal thoughts.

The rise of online hate speech has become an essential part of society. While it can be done through machine learning, it is also prone to abuse. We decided to create a system that will automatically tag offensive posts.

Business Understanding:

It is very crucial for any social platform to understand or having any algorithm which can be used to detect and understand the malignant and non-malignant comments and assign it to that particular column which it belongs to. So that any media or any online sites or social media remove that comment from their sites and stop cyberbullying.

Review Of Literature:

In today's world most of the people share their thoughts on social media this also increases the un stability and cyberbullying it can lead to hatred and suicidal-thoughts. To overcome this issue, we need to make sure that any statement which can lead to un stability.

Motivation behind undertaking of this problem:

In our country everyone has the right to express their thoughts and most of the people use social media to share their thought. But most of the people start abusing or posting malignant comment which can leads distressed among the people. So we have decided to build a model which can filter the malignant comment.

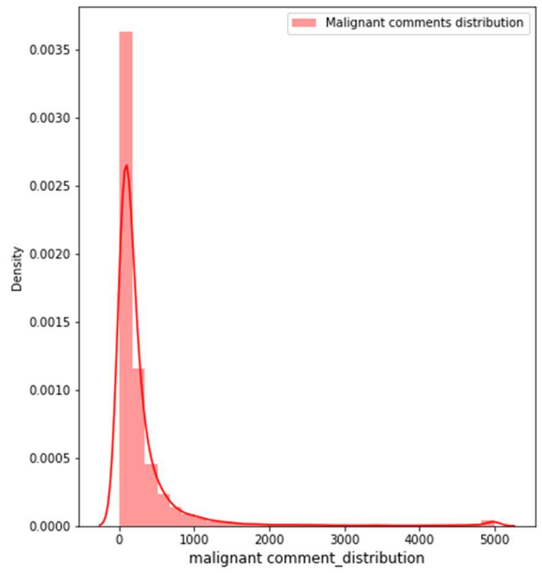
Mathematical Analysis:

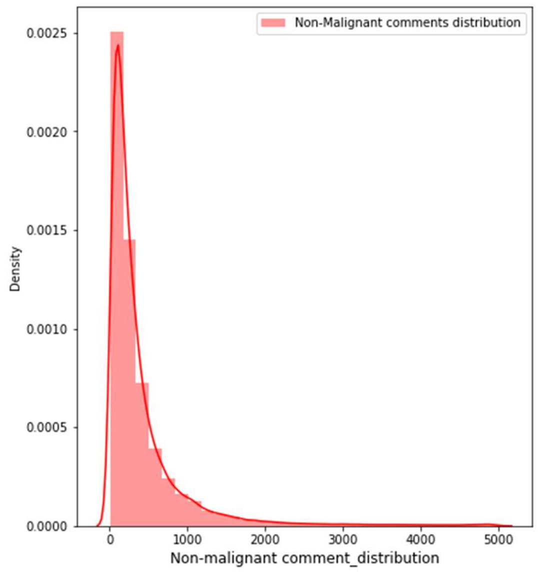
The dataset which we have 5 numerical columns i.e. loathe, rude, malignant, highly malignant and threat and one object datatype columns which have the comments. First we need to clean the dataset.

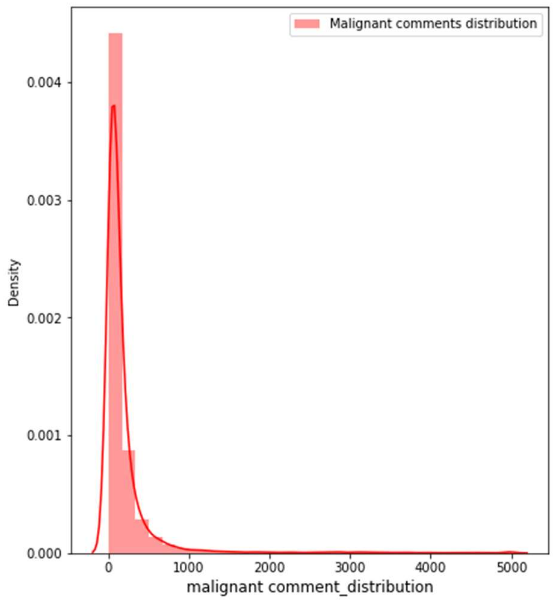
1. Firstly, we need to make sure that the data is converted to one case so I have converted all the comments to lower case
2. Regex Expression: - A regular expression is a sequence of characters that specifies a search pattern. Usually, such patterns are used by string-searching algorithms for "find" or "find and replace" operations on strings, or for input validation. We have

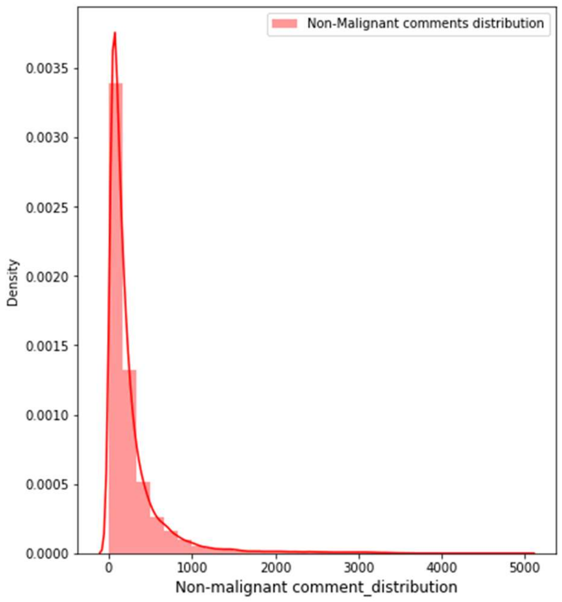
used to replace punchuation, and other special character which we need to remove from the dataset.

We have seen the distribution of data before and after cleaning and we have noticed that after cleaning the dataset the length of the comment reduced.









From above graphs we can see that there is huge density difference between malignant comments where as in non-malignant it is not that much affected.

Data Source and their format;

The data we received from the flip n robo has 7 columns i.e. id, comment text, loathe, rude, threat, malignant and highly malignant.

1. Id columns has all unique values for every comments.
2. Comment text: - It has the comments extracted from many sites.
3. Loathe, Rude, threat, malignant and highly malignant columns is like a target columns which we need to merge and make one columns in a such a way that any of the 5 columns has 1 then the target column append one else it append 0.

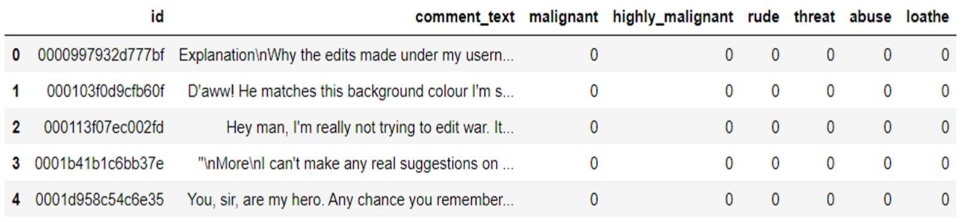
The dataset which we received is in csv format.

Data Description

The dataset contains seven columns. The total corpus of 159571 documents. The descriptive feature consists of text. The target features consist of two classes either 0 and 1.

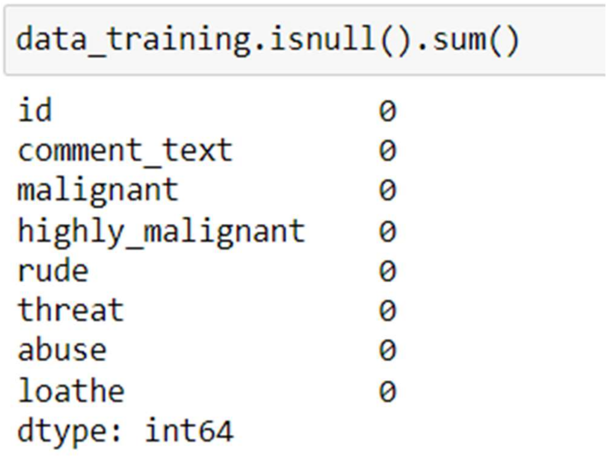
Data Preprocessing:

1. Understanding dataset is the most important thing.



	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

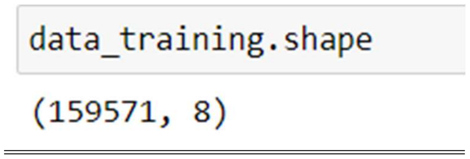
2. There is no null values present in the datasets.



```
data_training.isnull().sum()
```

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

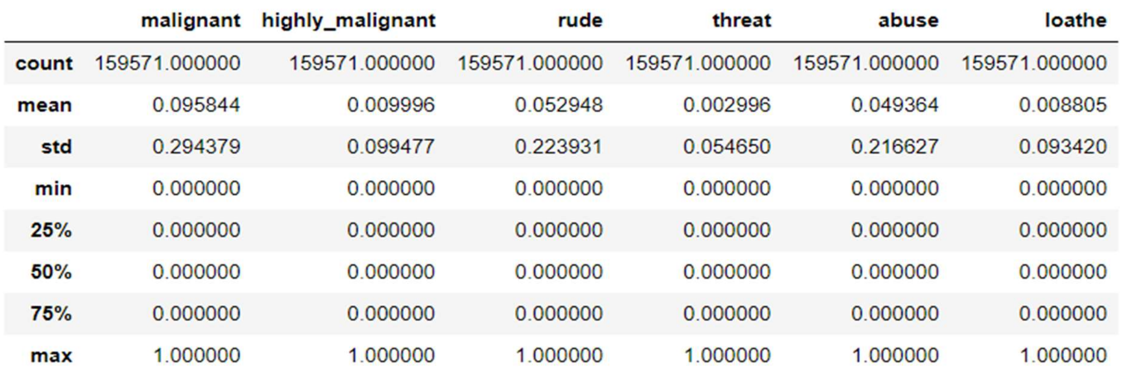
3. Checking shape of the dataset.



```
data_training.shape
```

```
(159571, 8)
```

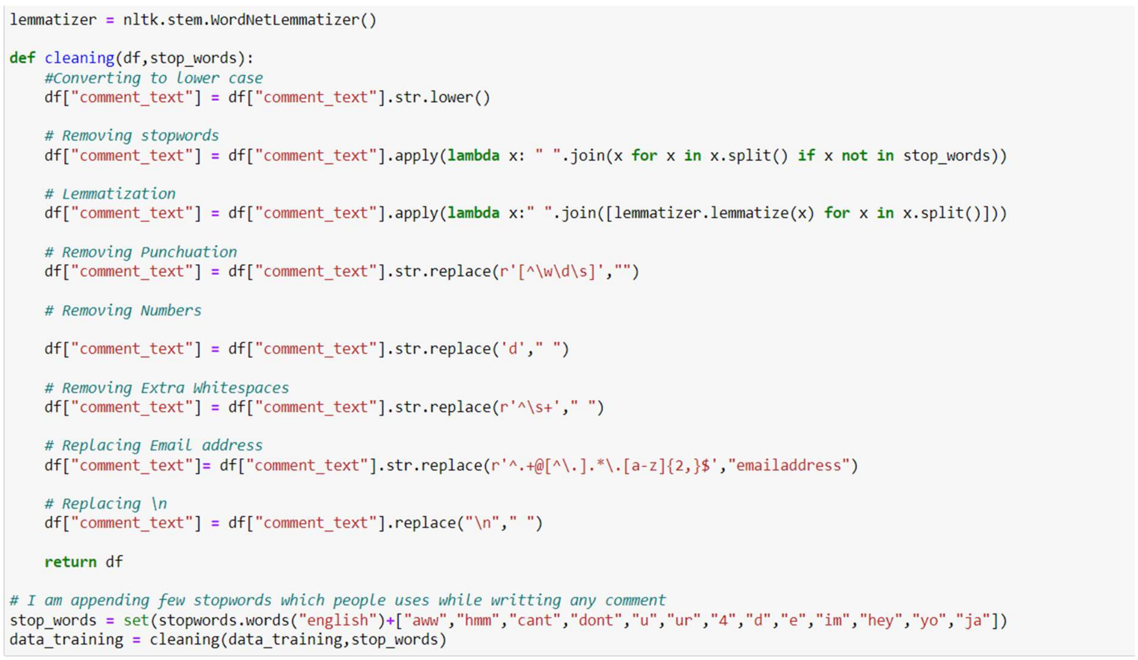
4. Checking basic info and statistics of the dataset.



	malignant	highly_malignant	rude	threat	abuse	loathe
count	159571.000000	159571.000000	159571.000000	159571.000000	159571.000000	159571.000000
mean	0.095844	0.009996	0.052948	0.002996	0.049364	0.008805
std	0.294379	0.099477	0.223931	0.054650	0.216627	0.093420
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

Data Cleaning:

We need to find and replace stop words, special characters and patterns from the datasets. We drop the stop words from the dataset because it doesn't carry any information from the dataset. Removing punctuation, and special character is very important because they did not carry any information.



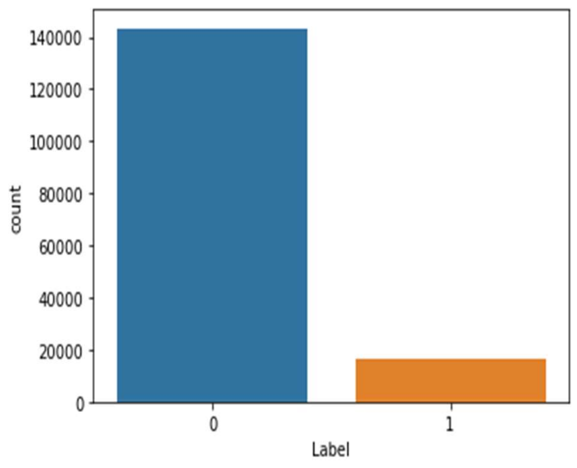
```python
lemmatizer = nltk.stem.WordNetLemmatizer()

def cleaning(df,stop_words):
    #Converting to lower case
    df["comment_text"] = df["comment_text"].str.lower()

    # Removing stopwords
    df["comment_text"] = df["comment_text"].apply(lambda x: " ".join(x for x in x.split() if x not in stop_words))

    # Lemmatization
    df["comment_text"] = df["comment_text"].apply(lambda x:" ".join([lemmatizer.lemmatize(x) for x in x.split()]))

    # Removing Punchuation
    df["comment_text"] = df["comment_text"].str.replace(r'[^\w\d\s]',"")

    # Removing Numbers

    df["comment_text"] = df["comment_text"].str.replace('d'," ")

    # Removing Extra Whitespaces
    df["comment_text"] = df["comment_text"].str.replace(r'^\s+'," ")

    # Replacing Email address
    df["comment_text"]= df["comment_text"].str.replace(r'^.+@[^\.].*\.[a-z]{2,}$',"emailaddress")

    # Replacing \n
    df["comment_text"] = df["comment_text"].replace("\n"," ")

    return df

# I am appending few stopwords which people uses while writting any comment
stop_words = set(stopwords.words("english")+["aww","hmm","cant","dont","u","ur","4","d","e","im","hey","yo","ja"])
data_training = cleaning(data_training,stop_words)
```

Data exploration

Percentage of non-malignant comments is 89.83211235124176 %

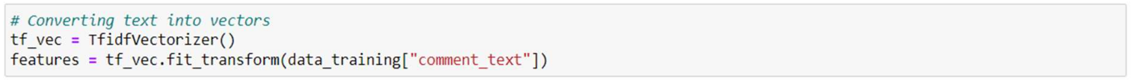
Percentage of Malignant comments is 10.167887648758233 %

The bar graph given below depicts the percentage of Ham and Spam emails in the given dataset. The blue bar represents the count of ham emails and the red bar shows the count of spam emails in the dataset.



Extracting features from comments:

TF-IDF :- **TF-IDF** stands for Term Frequency Inverse Document Frequency of records. It can be defined as the calculation of how relevant a word in a series or corpus is to a text. The meaning increases proportionally to the number of times in the text a word appears but is compensated by the word frequency in the corpus (data-set). It is used to convert text into vectors because the machine learning model only understand the vectors.



```
# Converting text into vectors
tf_vec = TfidfVectorizer()
features = tf_vec.fit_transform(data_training["comment_text"])
```

I have also used the same steps on test dataset which I have performed on training dataset.

Hardware and Software Requirement:

- Minimum core i5 or higher
- Minimum 8gb of ram
- Sufficient disk space Software and tools required:

Software Requirement

- Python is widely used in scientific and numeric computing
- SciPy is a collection of packages for mathematics, science, and engineering.
- Pandas is a data analysis and modelling library.

Modules or library required for project data analysis and visualization

- Pandas for data analysis and import
- NumPy to perform Mathematical operation.
- Nltk to perform text processing
- Seaborn and matplotlib to data visualization
- Scikit-learn: All the models, metrics and feature selection etc are present inside of that module. We import from this library according to our need

Modeling

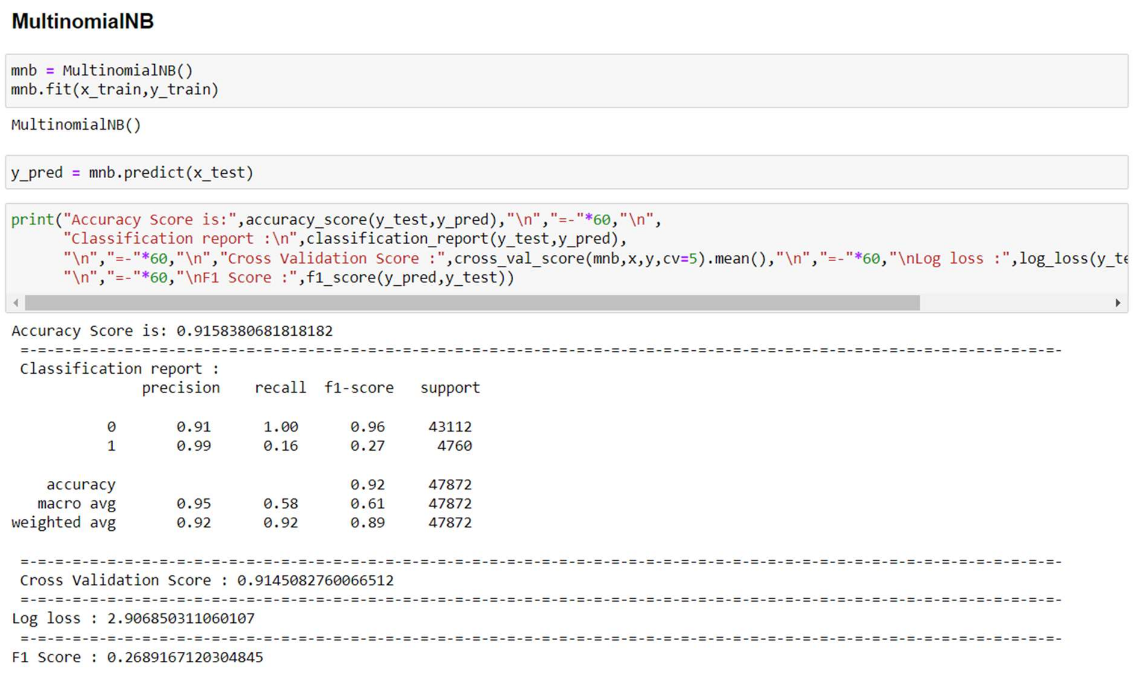
Models used: **Multinomial Naive Bayes, Logistic Regression, Decision Tree classifier and XGB Classifier**. Malignant and non-malignant classification done using traditional machine learning techniques.

Reason for choosing Multinomial Naive Bayes, Logistic Regression, Decision Tree classifier and XGB Classifier: All are good at handling large number of features; in the case of text classification each word is a feature and we have thousands and lakhs of words based on the vocabulary of the corpus. MultinomialNB and logistics works best with high dimensional data.

Multinomial Naive Bayes does not suffer from curse-of-dimensionality because it treats all features as independent of one another.

Splitting Training and Testing Data: Splitting the data into training and test datasets, where training data contains 70 percent and test data contains 30 percent.

Applying model: I trained the model for all the model without tuning hyperparameters as I got results with default parameter settings.



MultinomialNB

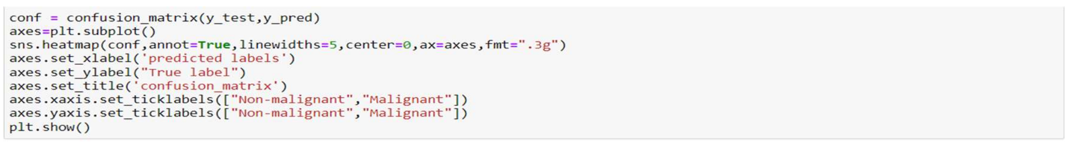
```
mnb = MultinomialNB()
mnb.fit(x_train,y_train)
```

```
MultinomialNB()
```

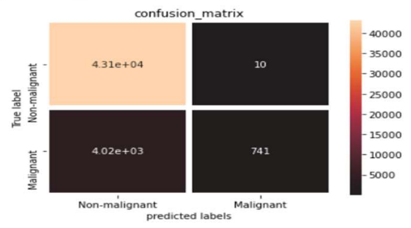
```
y_pred = mnb.predict(x_test)
```

```
print("Accuracy Score is:",accuracy_score(y_test,y_pred),"\n","=-"*60,"\n",
      "Classification report :\n",classification_report(y_test,y_pred),
      "\n","=-"*60,"\n","Cross Validation Score :",cross_val_score(mnb,x,y,cv=5).mean(),"\n","=-"*60,"\nLog loss :",log_loss(y_te
      "\n","=-"*60,"\nF1 Score :",f1_score(y_pred,y_test))
```

```
Accuracy Score is: 0.9158380681818182
 =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
 Classification report :
               precision    recall  f1-score   support

           0       0.91      1.00      0.96     43112
           1       0.99      0.16      0.27      4760

    accuracy                           0.92     47872
   macro avg       0.95      0.58      0.61     47872
weighted avg       0.92      0.92      0.89     47872

 =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
 Cross Validation Score : 0.9145082760066512
 =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Log loss : 2.906850311060107
 =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
F1 Score : 0.2689167120304845
```



```
conf = confusion_matrix(y_test,y_pred)
axes=plt.subplot()
sns.heatmap(conf,annot=True,linewidths=5,center=0,ax=axes,fmt=".3g")
axes.set_xlabel('predicted labels')
axes.set_ylabel("True label")
axes.set_title('confusion_matrix')
axes.xaxis.set_ticklabels(["Non-malignant","Malignant"])
axes.yaxis.set_ticklabels(["Non-malignant","Malignant"])
plt.show()
```



Logistic Regression

```
log_reg = LogisticRegression()
log_reg.fit(x_train,y_train)
```

```
LogisticRegression()
```

```
y_pred = log_reg.predict(x_test)
```

```
print("Accuracy Score is:",accuracy_score(y_test,y_pred),"\n","=-"*60,"\n",
      "Classification report :\n",classification_report(y_test,y_pred),
      "\n","=-"*60,"\n","Cross Validation Score :",cross_val_score(log_reg,x,y,cv=5).mean(),"\n","=-"*60,"\nLog loss :",log_loss(
      ,"\n","=-"*60,"\nF1 Score :",f1_score(y_pred,y_test))
```

```
Accuracy Score is: 0.9548379010695187
 =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
 Classification report :
               precision    recall  f1-score   support

           0       0.96      1.00      0.98     43112
           1       0.94      0.58      0.72      4760

    accuracy                           0.95     47872
   macro avg       0.95      0.79      0.85     47872
weighted avg       0.95      0.95      0.95     47872

 =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
 Cross Validation Score : 0.9534125861576659
 =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Log loss : 1.5598466262896251
 =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
F1 Score : 0.7198030067392431
```

DecisionTreeClassifier

```
dt =  DecisionTreeClassifier()
dt.fit(x_train,y_train)
```

```
DecisionTreeClassifier()
```

```
y_pred = dt.predict(x_test)
```

```
cv=cross_val_score(dt,x,y,cv=5).mean()
print("Accuracy Score is:",accuracy_score(y_test,y_pred),"\n","=-"*60,"\n",
      "Classification report :\n",classification_report(y_test,y_pred),
      "\n","=-"*60,"\n","Cross Validation Score :",cv,"\n","=-"*60,"\nLog loss :",log_loss(y_test,y_pred),"\n","=-"*60,
      "\nF1 Score :",f1_score(y_pred,y_test))
```

```
Accuracy Score is: 0.943829378342246
 =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
 Classification report :
               precision    recall  f1-score   support

           0       0.97      0.97      0.97     43112
           1       0.72      0.70      0.71      4760

    accuracy                           0.94     47872
   macro avg       0.85      0.84      0.84     47872
weighted avg       0.94      0.94      0.94     47872

 =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
 Cross Validation Score : 0.9393059965687394
 =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
Log loss : 1.940085854199444
 =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
F1 Score : 0.7134178834061601
```

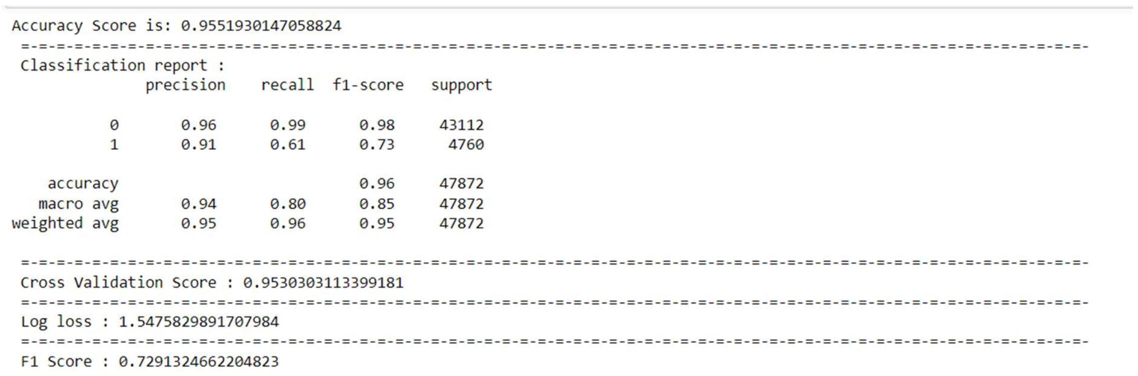
XGB Classifier

```
xgb = XGBClassifier(verbosity=0)
xgb.fit(x_train,y_train)
```

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
              gamma=0, gpu_id=-1, importance_type=None,
              interaction_constraints='', learning_rate=0.300000012,
              max_delta_step=0, max_depth=6, min_child_weight=1, missing=nan,
              monotone_constraints='()', n_estimators=100, n_jobs=8,
              num_parallel_tree=1, predictor='auto', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=0)
```

```
y_pred = xgb.predict(x_test)
```

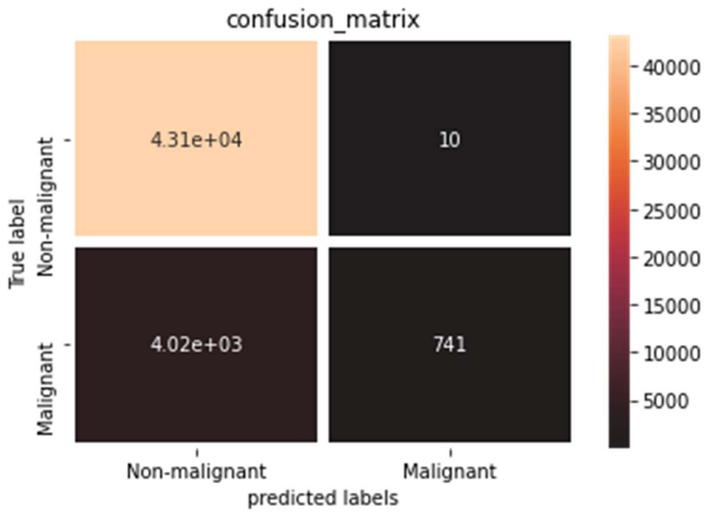
```
cv = cross_val_score(xgb,x,y,cv=5).mean()
print("Accuracy Score is:",accuracy_score(y_test,y_pred),"\n","=-"*60,"\n",
      "Classification report :\n",classification_report(y_test,y_pred),
      "\n","=-"*60,"\n","Cross Validation Score :",cv,"\n","=-"*60,"\n Log loss :",log_loss(y_test,y_pred),"\n","=-"*60,
      "\n F1 Score :",f1_score(y_pred,y_test))
```



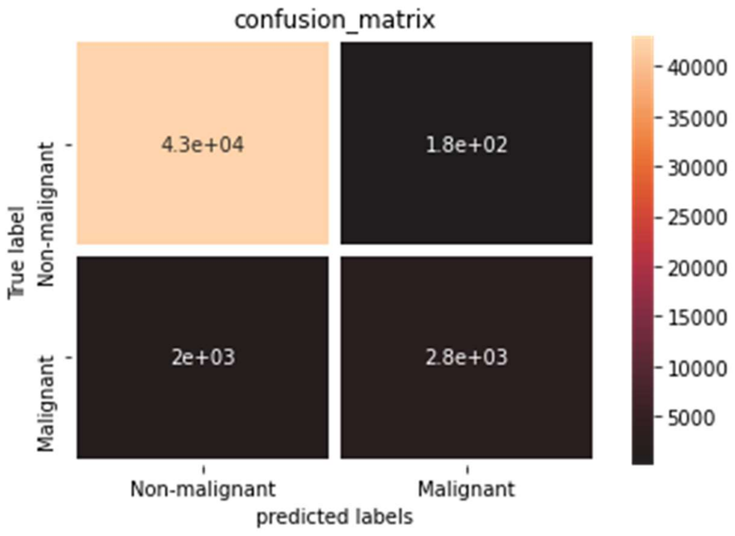
```
Accuracy Score is: 0.9551930147058824
 =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
 Classification report :
               precision    recall  f1-score   support

           0       0.96      0.99      0.98     43112
           1       0.91      0.61      0.73      4760

    accuracy                           0.96     47872
   macro avg       0.94      0.80      0.85     47872
weighted avg       0.95      0.96      0.95     47872

 =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
 Cross Validation Score : 0.9530303113399181
 =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
 Log loss : 1.5475829891707984
 =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
 F1 Score : 0.7291324662204823
```

Evaluation Metrics:

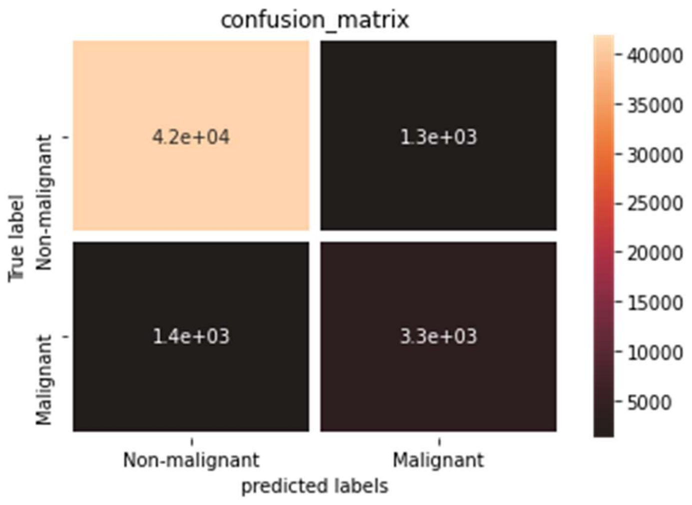
MultinomialNB Confusion Matrix:



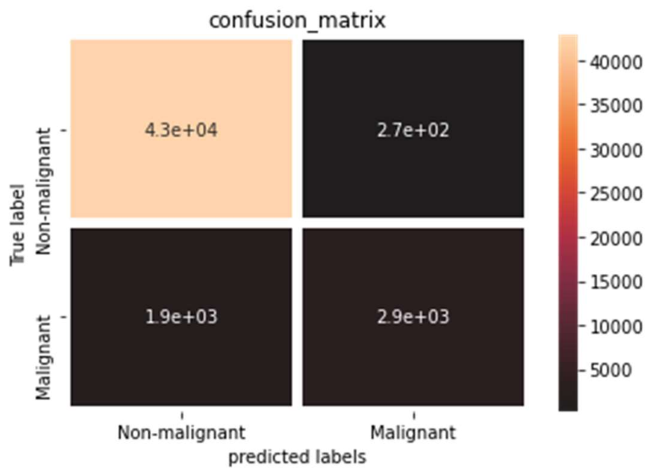
Logistic Regression:



DecisionTreeClassifier:

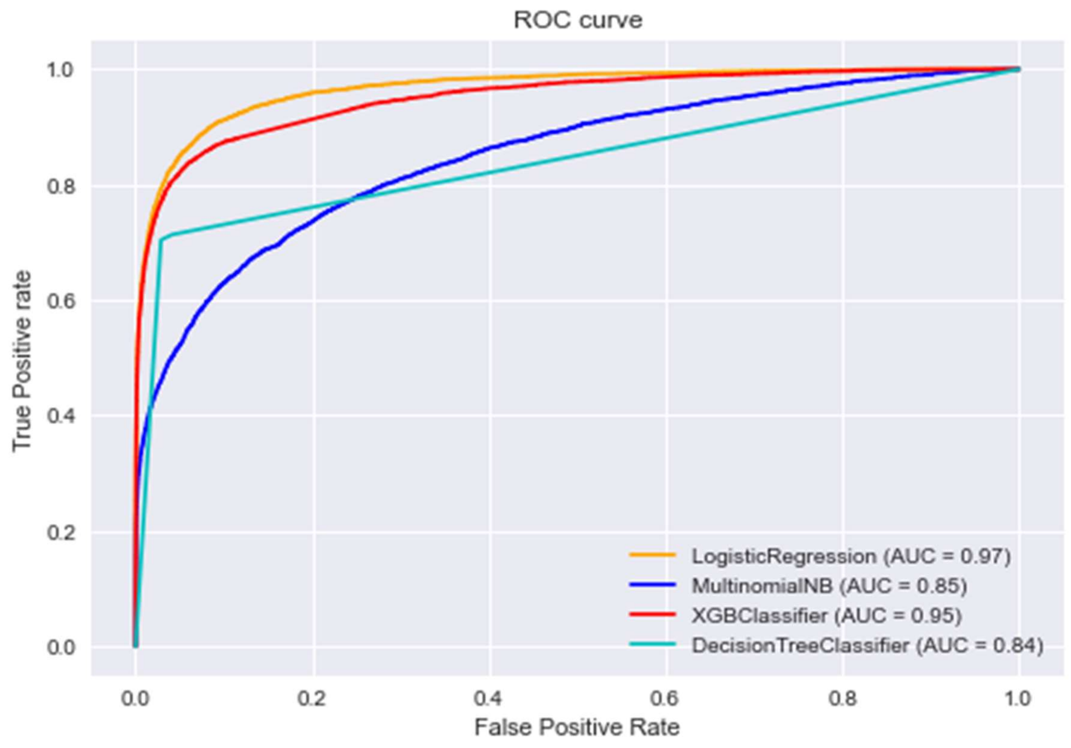


XGB Classifier:



I have also used the log loss, Cross validation and accuracy score for the evaluation of the model.

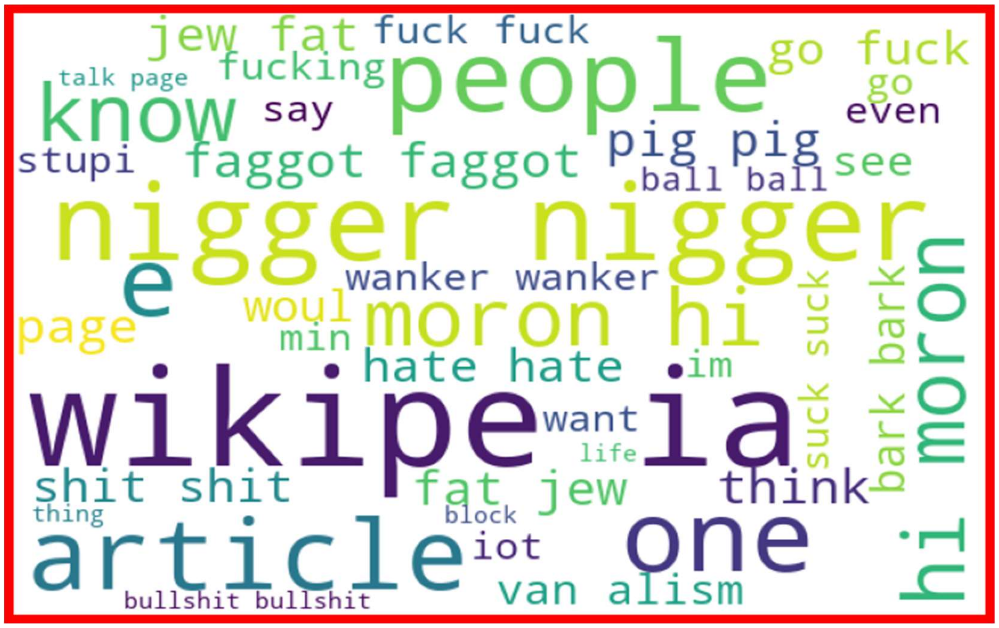
AUC-ROC CURVE



<u>Auc roc curve one of the important matrix to select the classification model. But we are selecting model on the basis of F1 score</u>

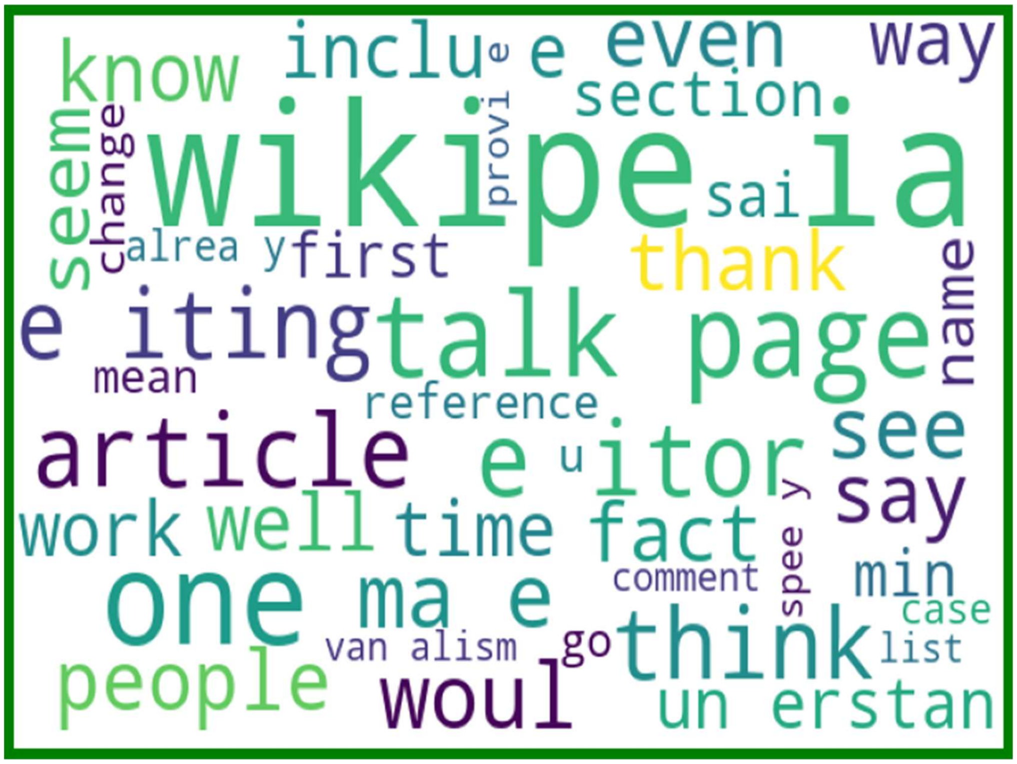
<u>Visualisation:</u>

Word Cloud for malignant comment



The above word cloud shows the most frequent word used in malignant document.

Word Cloud for non-malignant comment.



The above word cloud shows the most frequent word used in non-malignant document.

Interpretation of the Result:

- The documents which contain abusive word which tend to insult someone or spread hate is consider as malignant
- The document which do not have abusive word or context is consider as non-malignant

Conclusion:-

This model can find any comment which is malignant or not malign and gives us the output in the form of 1 and 0 respectively. This will help us to take care or find malign comments.

Future Work:-

We can use more models and work on large dataset and we can try multiple vectorization techniques and find out which model is working best.