FLIP ROBO

MALIGNANT COMMENTS CLASSIFICATION PROJECT

Submitted by:

Mekhala Misra

ACKNOWLEDGMENT

I took help from following websites:

1)Geek for geeks

2)Pandas documentation

3)towardsdatascience.com

4)data trained

INTRODUCTION

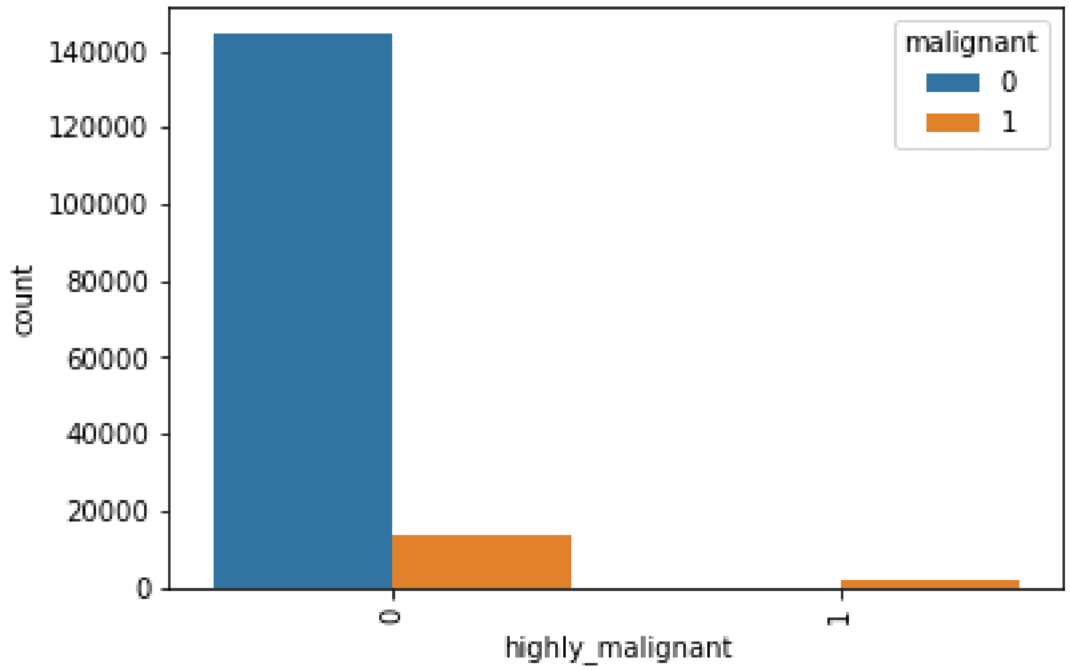
- Business Problem Framing

- 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 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.

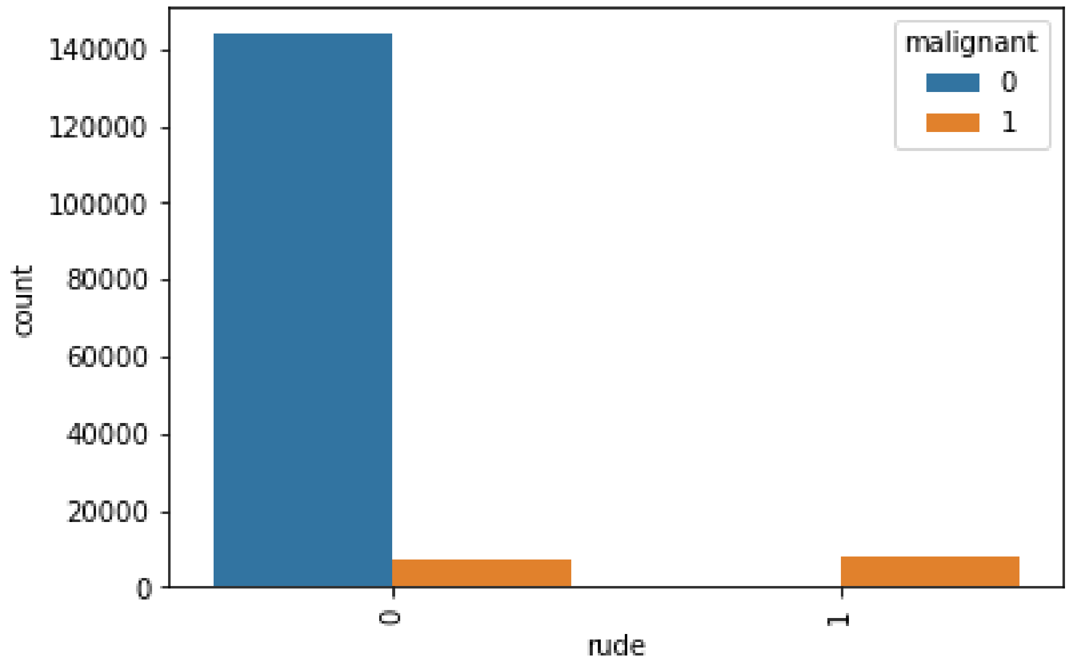
Review of Literature

Using various Count plot graph I came to certain conclusion:

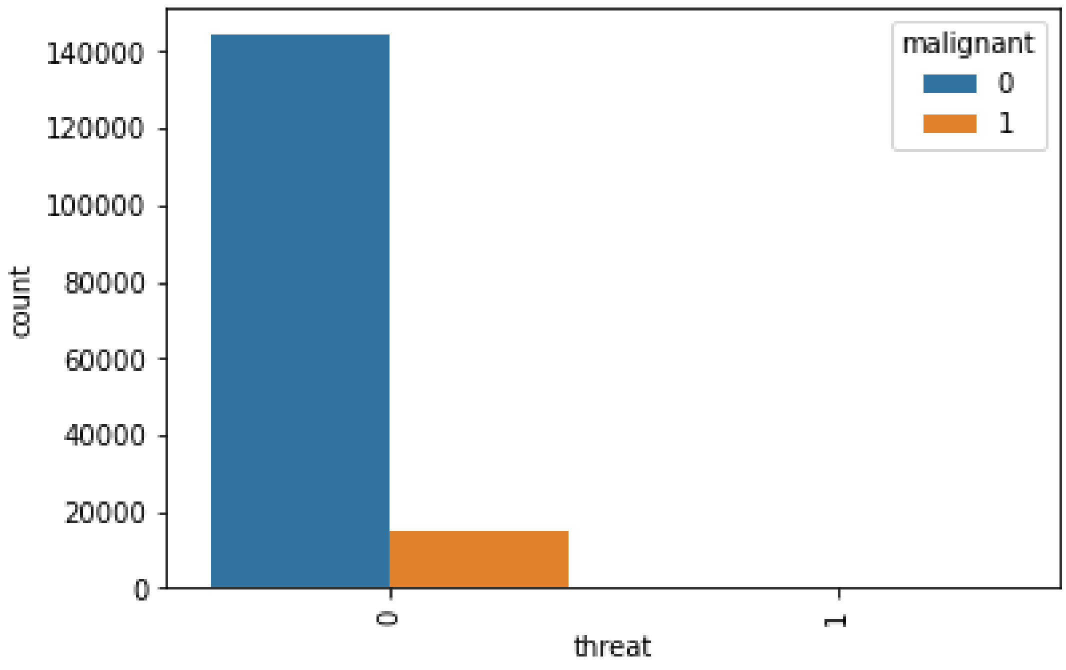
1) Here we can see that if highly_malignant is true then the comment has to be malignant but in few cases if highly_maglinant is not true still the comments are malignant .



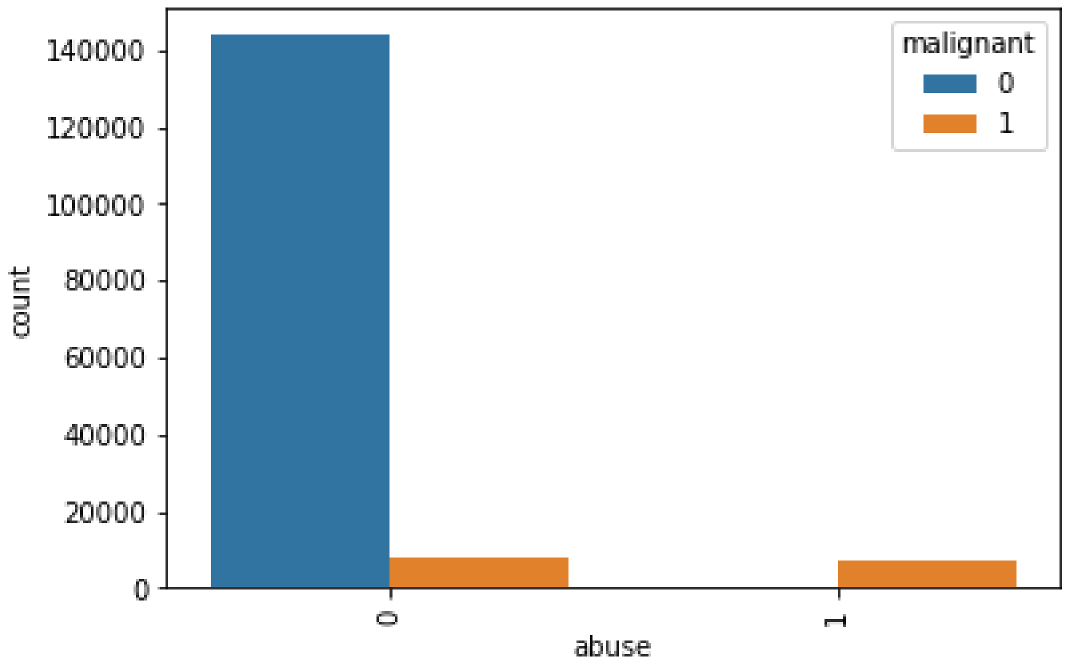
2) If the comment is rude then there are higher chances that it is malignant and vice-versa.



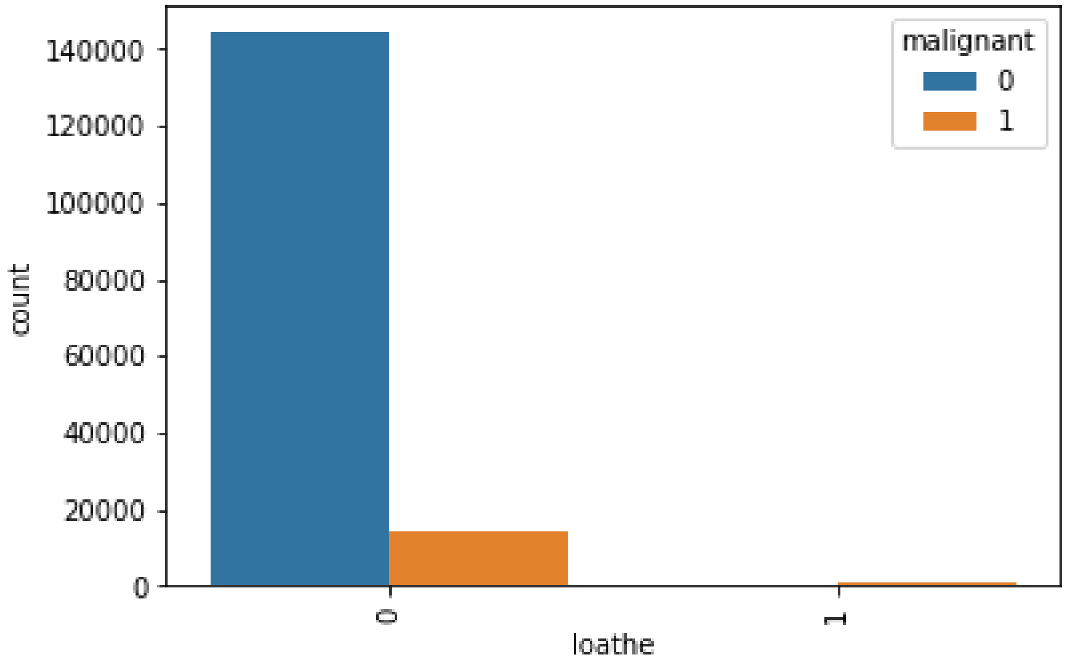
3) We could observe that the threat value is always 0 but still at some cases even if threat value is 0, malignant value is 1.



4) We could observe that at maximum times when abuse is 0, malignant is 0 as well.



5) We could observe that at maximum times when loathe is 0, malignant is 0 as well.



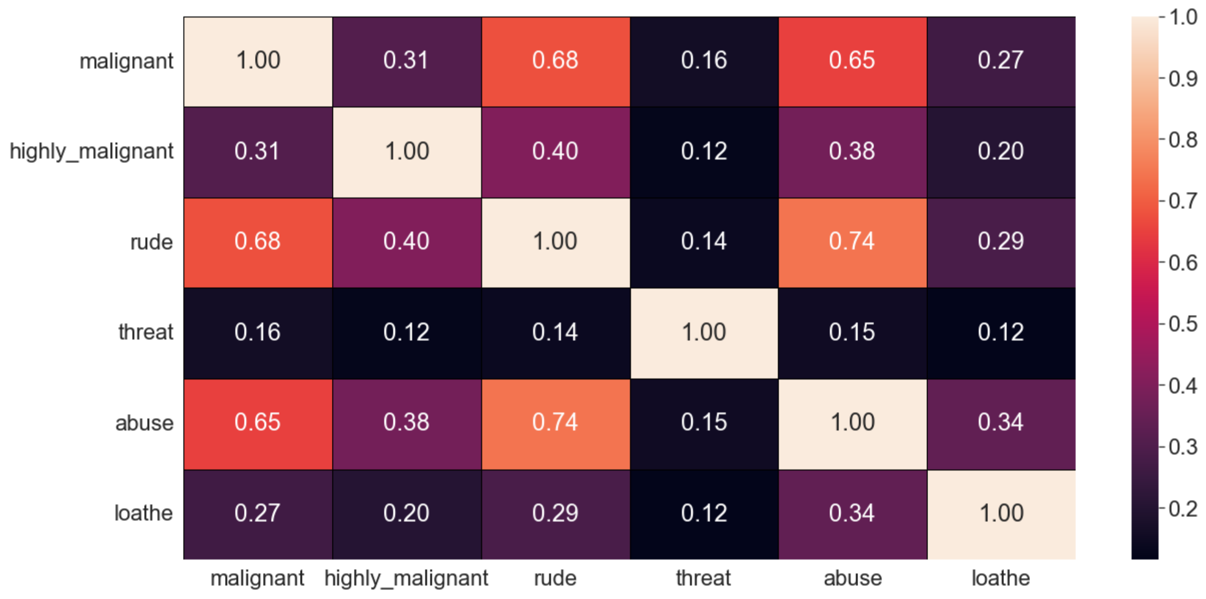
- Motivation for the Problem Undertaken

In todays time when there are lot of users of social media and that too of different age groups, its very important to keep track of type of comments that are floating on any social media. One cannot guess what kind of impact any comment puts on one selves and how serious actions could be taken by that person.
Few people are sporting and used to to social media trolls but how about the people who are new to it and takes malignant comments on there self respect.
So we much have some technology to prevent such kind of malignant comment from floating on any media.
This was the main motivation why I took this problem.

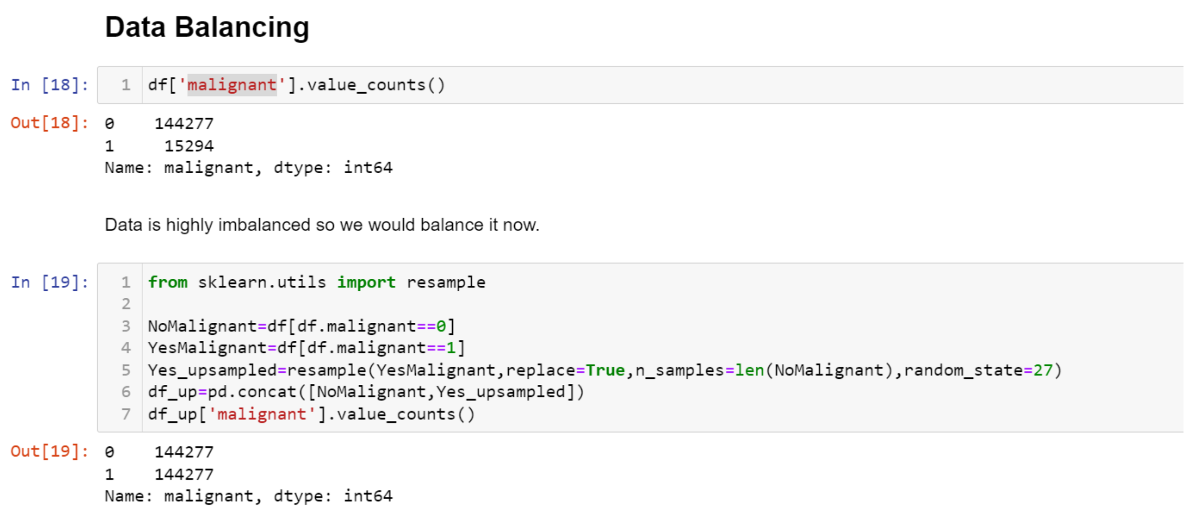
Analytical Problem Framing

- Mathematical/ Analytical Modelling of the Problem
    1) The dataset had no null values but **there is a feature called comment_text that contains the comment posted on any social media**, so we need to study this feature very hard and performed various feature engineering technique on it.

    2) We generated heatmap in order to check relation between features but could'nt found any multicollinearity problem here.



3) We found out that the data was imbalanced so we balanced the data by performing upsampling technique on it.



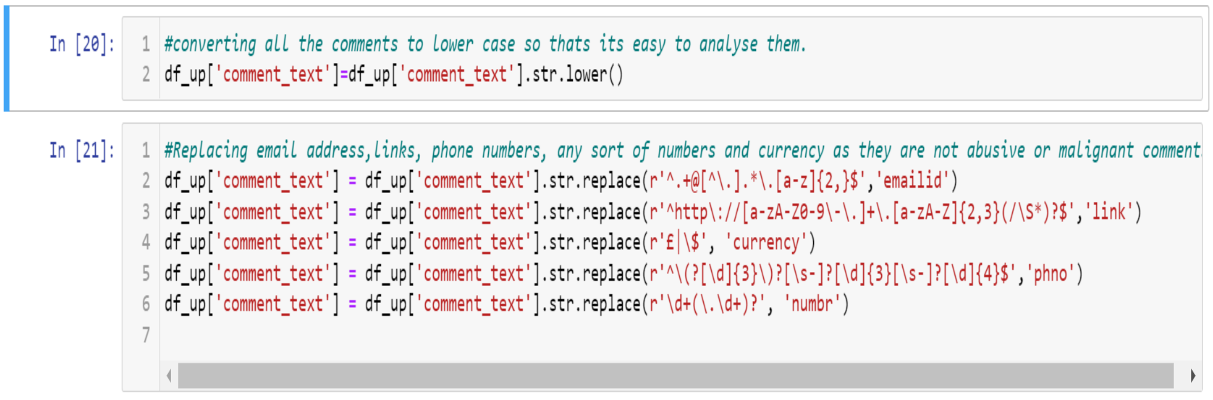
Data Balancing

```
In [18]: df['malignant'].value_counts()

Out[18]: 0    144277
         1     15294
         Name: malignant, dtype: int64
```

Data is highly imbalanced so we would balance it now.

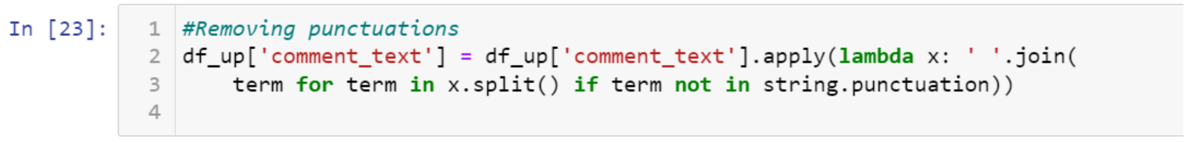
```
In [19]: from sklearn.utils import resample

         NoMalignant=df[df.malignant==0]
         YesMalignant=df[df.malignant==1]
         Yes_upsampled=resample(YesMalignant,replace=True,n_samples=len(NoMalignant),random_state=27)
         df_up=pd.concat([NoMalignant,Yes_upsampled])
         df_up['malignant'].value_counts()

Out[19]: 0    144277
         1    144277
         Name: malignant, dtype: int64
```

4) **Feature Engineering-**We have to fetch abusive, rude words from the comment so in order to do that we first converted entire comment_text feature to lower case then replaced all the phone numbers, email ids, URL's, any sort of number and currency by phno,emailed,link,numbr and currency respectively.



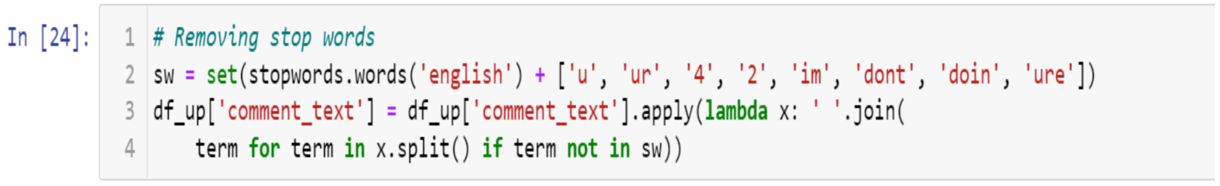
```
In [20]: #converting all the comments to lower case so thats its easy to analyse them.
         df_up['comment_text']=df_up['comment_text'].str.lower()

In [21]: #Replacing email address,links, phone numbers, any sort of numbers and currency as they are not abusive or malignant comment
         df_up['comment_text'] = df_up['comment_text'].str.replace(r'^.+@[^\.].*\.[a-z]{2,}$','emailid')
         df_up['comment_text'] = df_up['comment_text'].str.replace(r'^http\://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(/\S*)?$','link')
         df_up['comment_text'] = df_up['comment_text'].str.replace(r'£|\$', 'currency')
         df_up['comment_text'] = df_up['comment_text'].str.replace(r'^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$','phno')
         df_up['comment_text'] = df_up['comment_text'].str.replace(r'\d+(\.\d+)?', 'numbr')
```

5) Further we removed all kinds of punctuations from the feature comment_text.



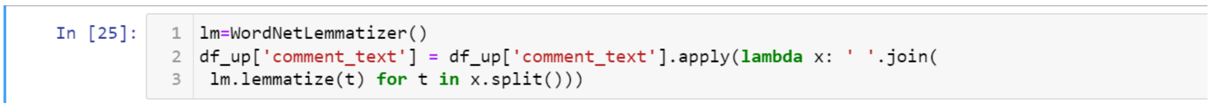
```
In [23]: #Removing punctuations
         df_up['comment_text'] = df_up['comment_text'].apply(lambda x: ' '.join(
             term for term in x.split() if term not in string.punctuation))
```

6) In order to fetch just the abusive words we also have to remove all kind of stop words.



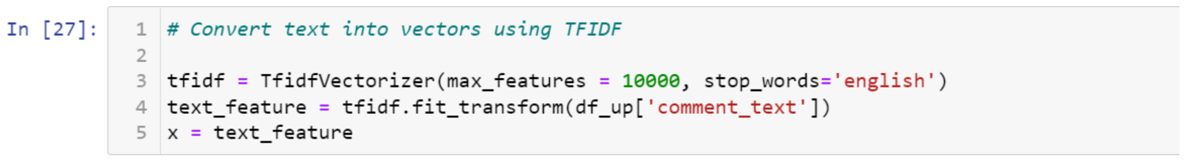
```
In [24]: # Removing stop words
         sw = set(stopwords.words('english') + ['u', 'ur', '4', '2', 'im', 'dont', 'doin', 'ure'])
         df_up['comment_text'] = df_up['comment_text'].apply(lambda x: ' '.join(
             term for term in x.split() if term not in sw))
```

7) In order to further analyse the comment_text using morphological analysis of the words called lemmatization.

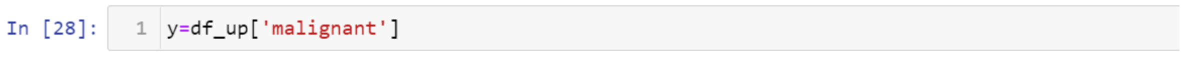


```
In [25]:  lm=WordNetLemmatizer()
          df_up['comment_text'] = df_up['comment_text'].apply(lambda x: ' '.join(
           lm.lemmatize(t) for t in x.split()))
```

8) Then finally we converted the comment_text into its vector form Term Frequency and Inverse Document Frequency method.



```
In [27]:  # Convert text into vectors using TFIDF

          tfidf = TfidfVectorizer(max_features = 10000, stop_words='english')
          text_feature = tfidf.fit_transform(df_up['comment_text'])
          x = text_feature
```



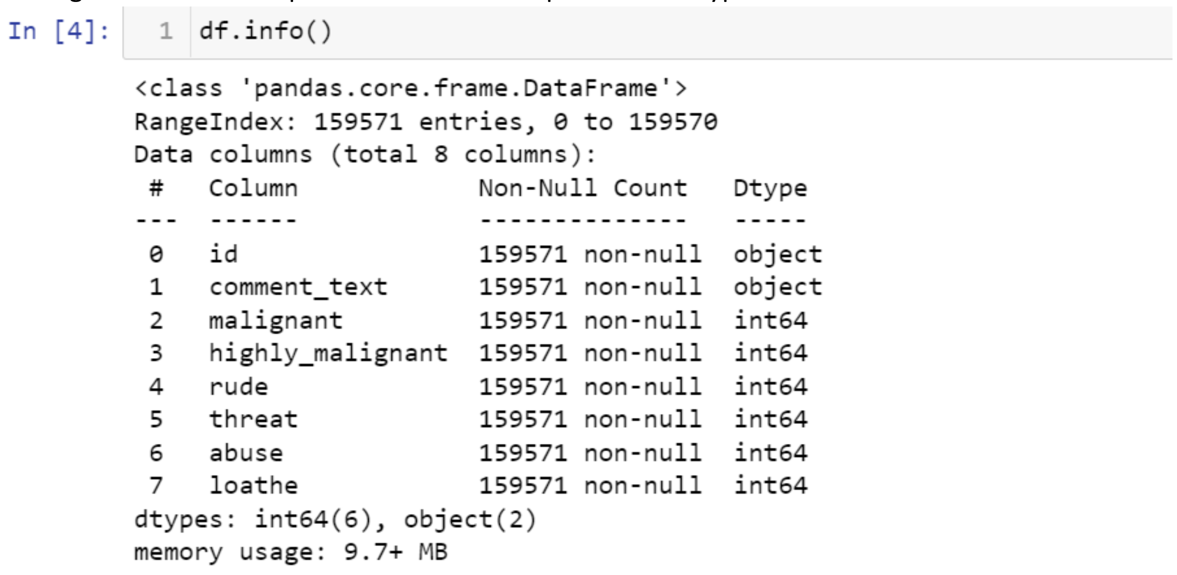
```
In [28]:  y=df_up['malignant']
```

9) We split the data into training set and testing set using train test split method.
10) On this train and test data we applied various models: logistic regression, decision tree classifier, random forest classifier , Knn classifier.
11) The best performing model is random forest as its confusion matrix, auc-roc curve and recall, f1-score is the best among all. Since the model is already giving its best accuracy so we won't apply hyperparameter tuning as it's a very big data so applying hyperparameter tuning is very time consuming and the scope of accuracy improvement is very less.
12) So, we saved our previous random forest classifier model.

- ## Data Sources and their formats

  Data set is provided by the FlipRobo technologies and it has 159571 rows and 8 columns.

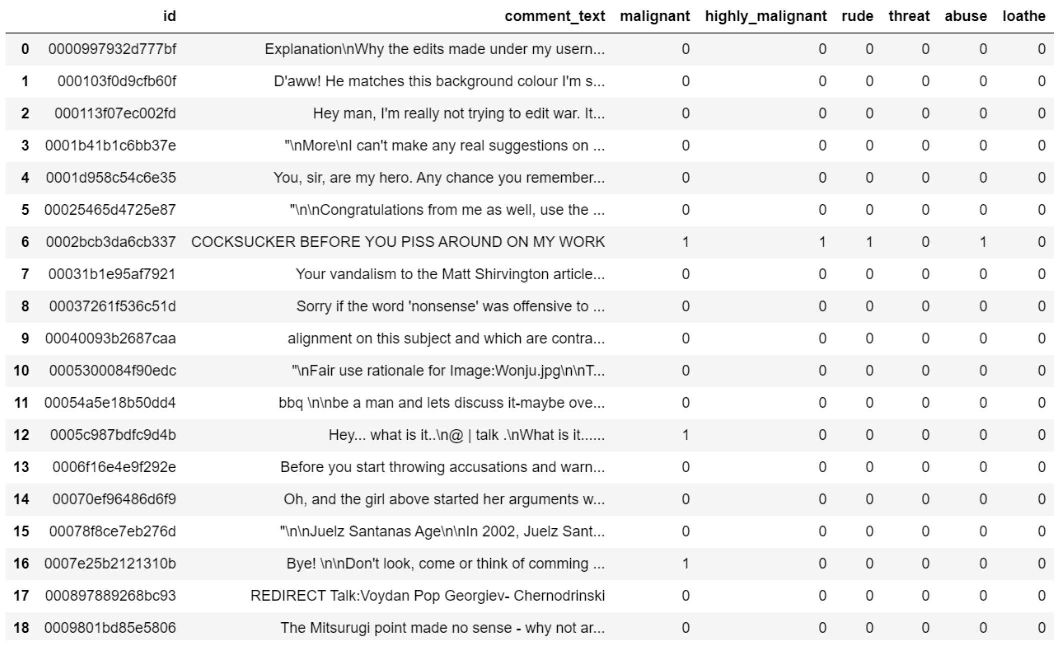
  Following are the columns present and there respective data types:



```
In [4]:   df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 159571 entries, 0 to 159570
Data columns (total 8 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   id                159571 non-null  object
 1   comment_text      159571 non-null  object
 2   malignant         159571 non-null  int64
 3   highly_malignant  159571 non-null  int64
 4   rude              159571 non-null  int64
 5   threat            159571 non-null  int64
 6   abuse             159571 non-null  int64
 7   loathe            159571 non-null  int64
dtypes: int64(6), object(2)
memory usage: 9.7+ MB
```

  Here is the glimpse of data:



	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
5	00025465d4725e87	"\n\nCongratulations from me as well, use the ...	0	0	0	0	0	0
6	0002bcb3da6cb337	COCKSUCKER BEFORE YOU PISS AROUND ON MY WORK	1	1	1	0	1	0
7	00031b1e95af7921	Your vandalism to the Matt Shirvington article...	0	0	0	0	0	0
8	00037261f536c51d	Sorry if the word 'nonsense' was offensive to ...	0	0	0	0	0	0
9	00040093b2687caa	alignment on this subject and which are contra...	0	0	0	0	0	0
10	0005300084f90edc	"\nFair use rationale for Image:Wonju.jpg\n\nT...	0	0	0	0	0	0
11	00054a5e18b50dd4	bbq \n\nbe a man and lets discuss it-maybe ove...	0	0	0	0	0	0
12	0005c987bdfc9d4b	Hey... what is it..\n@ | talk .\nWhat is it......	1	0	0	0	0	0
13	0006f16e4e9f292e	Before you start throwing accusations and warn...	0	0	0	0	0	0
14	00070ef96486d6f9	Oh, and the girl above started her arguments w...	0	0	0	0	0	0
15	00078f8ce7eb276d	"\n\nJuelz Santanas Age\n\nIn 2002, Juelz Sant...	0	0	0	0	0	0
16	0007e25b2121310b	Bye! \n\nDon't look, come or think of comming ...	1	0	0	0	0	0
17	000897889268bc93	REDIRECT Talk:Voydan Pop Georgiev- Chernodrinski	0	0	0	0	0	0
18	0009801bd85e5806	The Mitsurugi point made no sense - why not ar...	0	0	0	0	0	0

• Data Pre-processing Done

- The dataset had no null values but **there is a feature called comment_text that contains the comment posted on any social media**, so we need to study this feature very hard and performed various feature engineering technique on it.

- We generated heatmap in order to check relation between features but could'nt found any multicollinearity problem here.

- We found out that the data was imbalanced so we balanced the data by performing upsampling technique on it.

- **Feature Engineering-**We have to fetch abusive, rude words from the comment so in order to do that we first converted entire comment_text feature to lower case then replaced all the phone numbers, email ids, URL's, any sort of number and currency by phno,emailed,link,numbr and currency respectively.

- Further we removed all kinds of punctuations from the feature comment_text.

- In order to fetch just the abusive words we also have to remove all kind of stop words.

- In order to further analyse the comment_text using morphological analysis of the words called lemmatization.

- Then finally we converted the comment_text into its vector form Term Frequency and Inverse Term Frequency method.

- We split the data into training set and testing set using train test split method.
- On this train and test data we applied various models: logistic regression, decision tree classifier, random forest classifier , Knn classifier.
- The best performing model is random forest as its confusion matrix, auc-roc curve and recall, f1-score is the best among all. Since the model is already giving its best accuracy so we won't apply hyperparameter tuning as it's a very big data so applying hyperparameter tuning is very time consuming and the scope of accuracy improvement is very less.
- So, we saved our previous random forest classifier model.

- ## Hardware and Software Requirements and Tools Used

  We imported following packages:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import OrdinalEncoder
from sklearn.preprocessing import LabelEncoder
from statsmodels.stats.outliers_influence import variance_inflation_factor
from scipy.stats import zscore
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import power_transform
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn import metrics
from sklearn.metrics import roc_curve,auc,classification_report
from sklearn.feature_extraction.text import TfidfVectorizer
import warnings
warnings.filterwarnings('ignore')
from nltk.stem import WordNetLemmatizer
import nltk
from nltk.corpus import  stopwords
import string
```

  Further we downloaded stopwords and wordnet module as well.

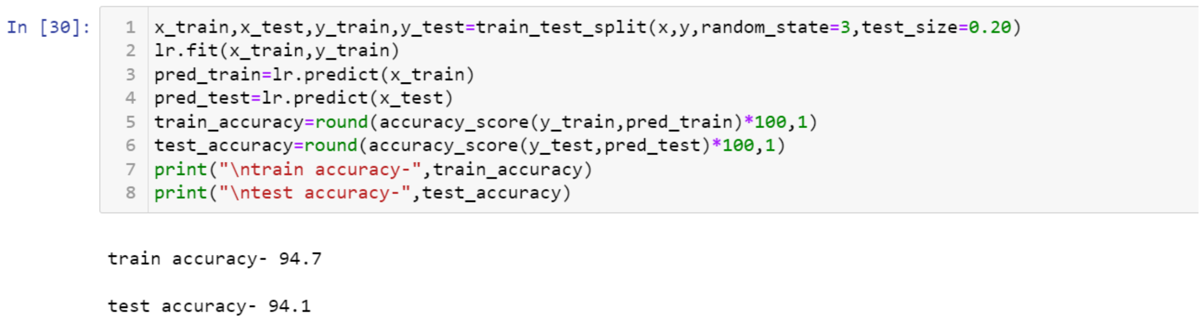
Model/s Development and Evaluation

- Testing of Identified Approaches (Algorithms)
- We applied Logistic regression, Decision Tree classifier,random forest classifier, knn classifier on the clean data and got 94.1 %,96.7%, 98.9%,85.2% accuracy respectively.
- Depending on the model accuracy, confusion matrix, auc-roc curve and classification report we opted random forest classifier.
- Since random forest classifier is giving best accuracy already so I did not apply hyper parameter tuning on dataset as the dataset is very large and time consuming and scope of accuracy improvement is not much high. So, we saved our previous random forest classifier model.

- **Run and Evaluate selected models**

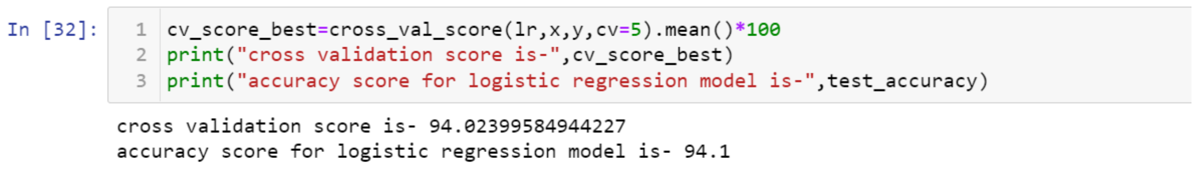
1) **Logistic Regression**

We applied this algorithm and found the train accuracy to be 94.7% and test accuracy to be 94.1%.



```
In [30]:
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=3,test_size=0.20)
lr.fit(x_train,y_train)
pred_train=lr.predict(x_train)
pred_test=lr.predict(x_test)
train_accuracy=round(accuracy_score(y_train,pred_train)*100,1)
test_accuracy=round(accuracy_score(y_test,pred_test)*100,1)
print("\ntrain accuracy-",train_accuracy)
print("\ntest accuracy-",test_accuracy)
```

```

train accuracy- 94.7

test accuracy- 94.1
```



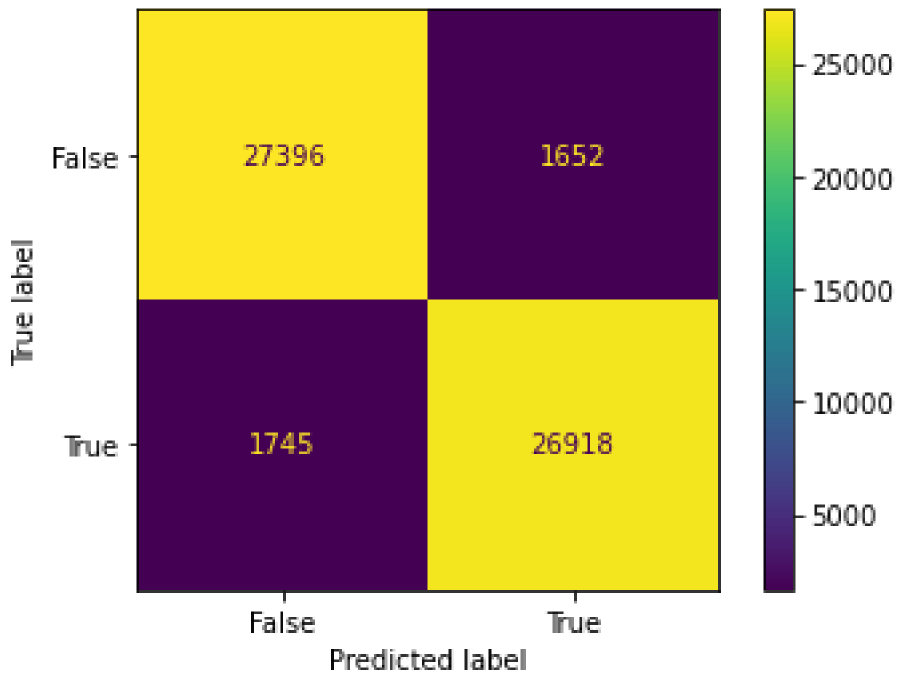
```
In [32]:
cv_score_best=cross_val_score(lr,x,y,cv=5).mean()*100
print("cross validation score is-",cv_score_best)
print("accuracy score for logistic regression model is-",test_accuracy)
```

```
cross validation score is- 94.02399584944227
accuracy score for logistic regression model is- 94.1
```

The test accuracy for logistic regression is 94.1% and its cv score is 94.02% thus making us sure that the model is not overfitted.

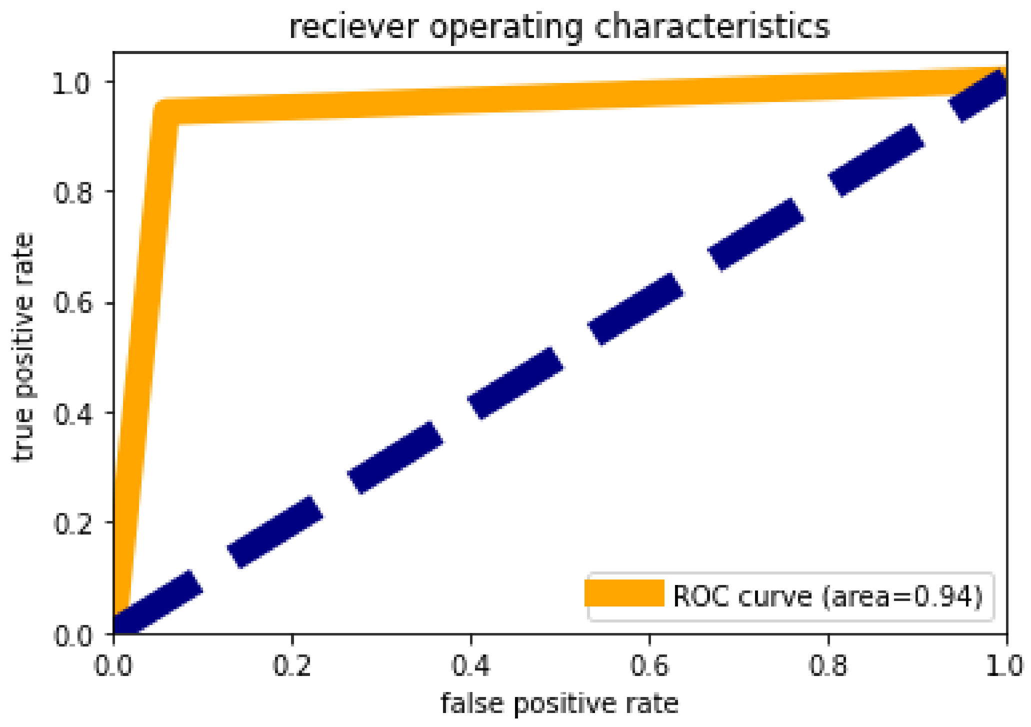
We also tested this models on other metrics:

i) Confusion matrix:



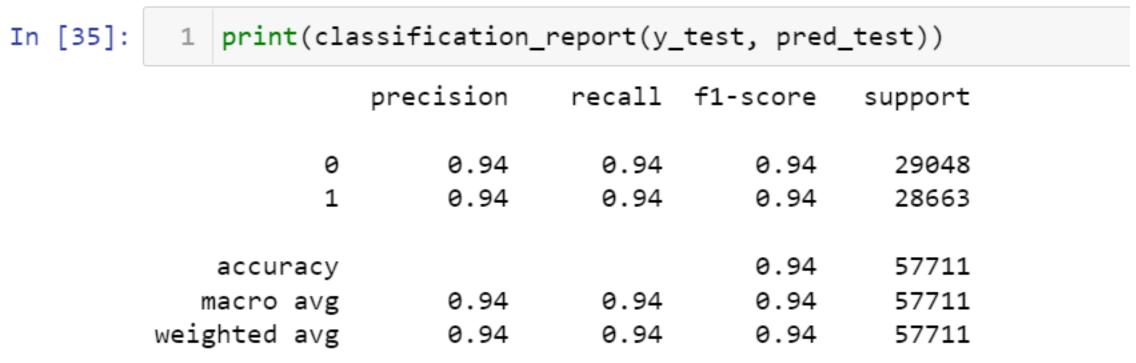
Model is good in predicting both the classes

ii) AUC-ROC Curve:



We observed that the area under the curve is 94% that means 94% times model is predicting accurately and rest all other time it gives wrong prediction.

iii) Classification Report:



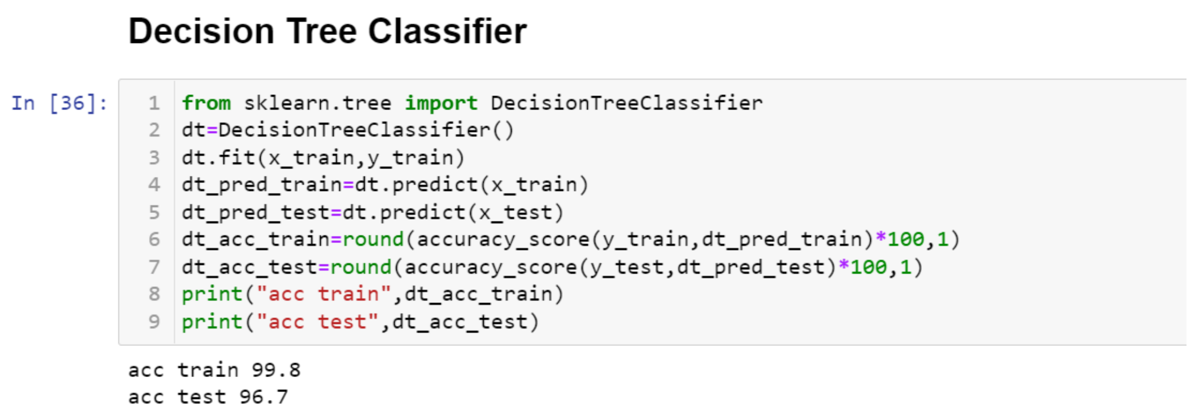
```
In [35]:
print(classification_report(y_test, pred_test))
```

```
              precision    recall  f1-score   support

           0       0.94      0.94      0.94     29048
           1       0.94      0.94      0.94     28663

    accuracy                           0.94     57711
   macro avg       0.94      0.94      0.94     57711
weighted avg       0.94      0.94      0.94     57711
```

Precision, recall and f1-score is same foe both the classes.

2) Decision Tree Classifier

We applied this algorithm and found the train accuracy to be 99.8% and test accuracy to be 96.7%.



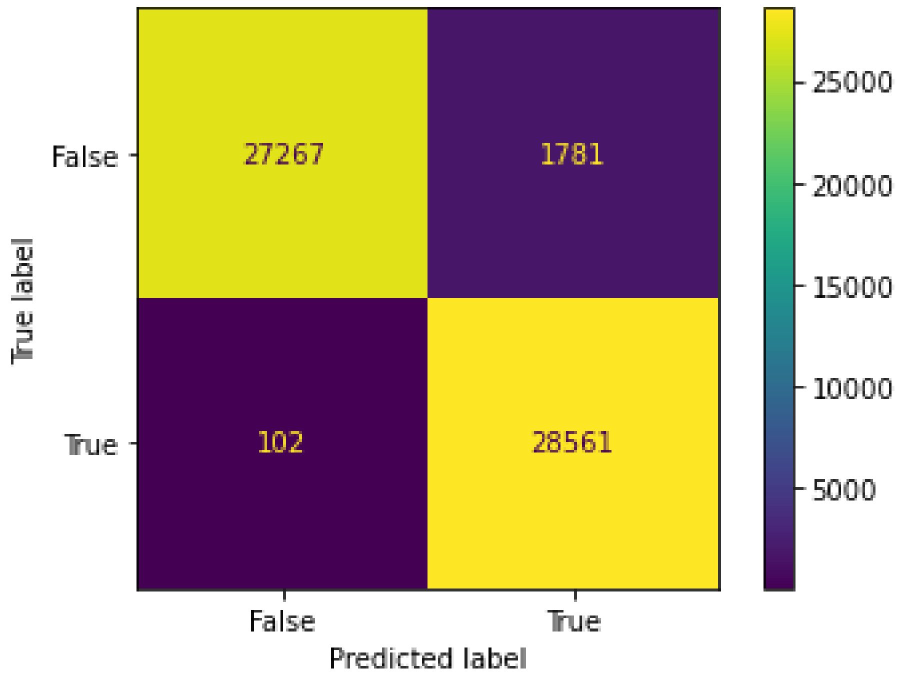
Decision Tree Classifier

```
In [36]:
from sklearn.tree import DecisionTreeClassifier
dt=DecisionTreeClassifier()
dt.fit(x_train,y_train)
dt_pred_train=dt.predict(x_train)
dt_pred_test=dt.predict(x_test)
dt_acc_train=round(accuracy_score(y_train,dt_pred_train)*100,1)
dt_acc_test=round(accuracy_score(y_test,dt_pred_test)*100,1)
print("acc train",dt_acc_train)
print("acc test",dt_acc_test)
```

```
acc train 99.8
acc test 96.7
```

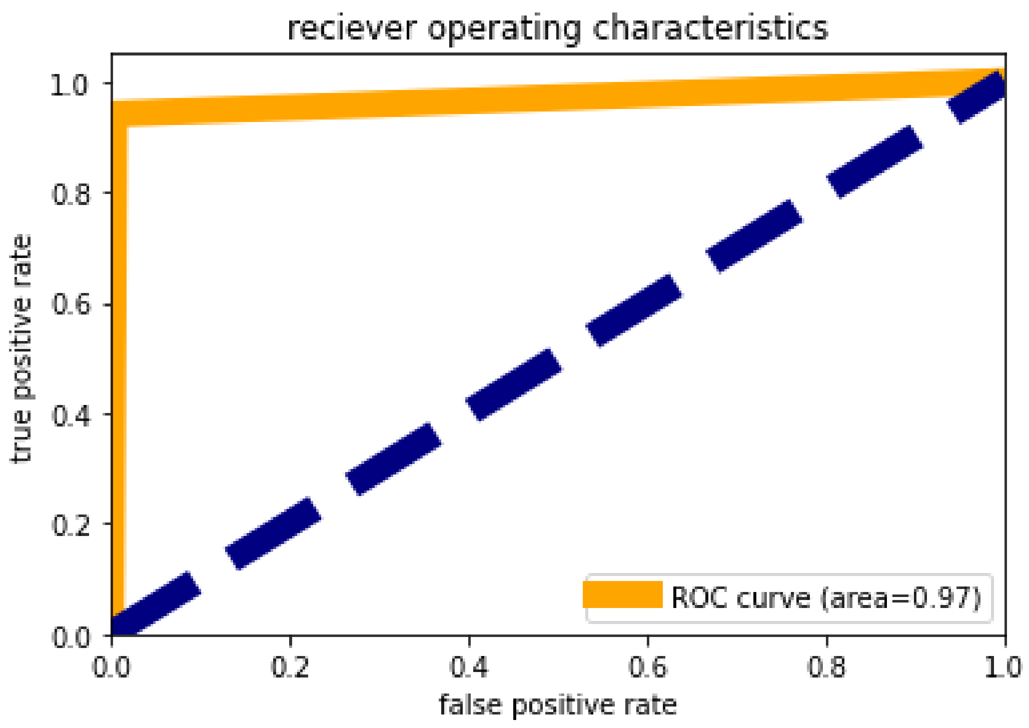
We also tested this model on other metrics:

i) Confusion matrix:



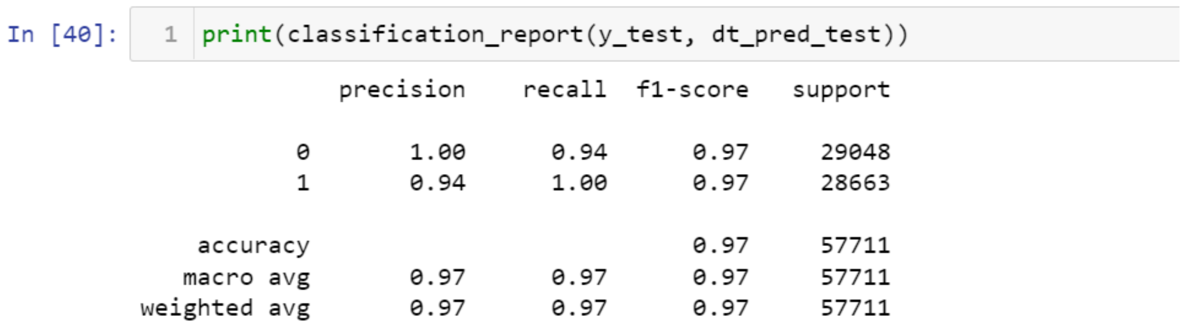
Model is bad in predicting false classes but good in predicting true classes.

ii) AUC-ROC Curve:



We observed that the area under the curve is 97% that means 97% times model is predicting accurately and rest all other time it gives wrong prediction. This accuracy is a good one lets check on other metrics as well.

iii) Classification Report:



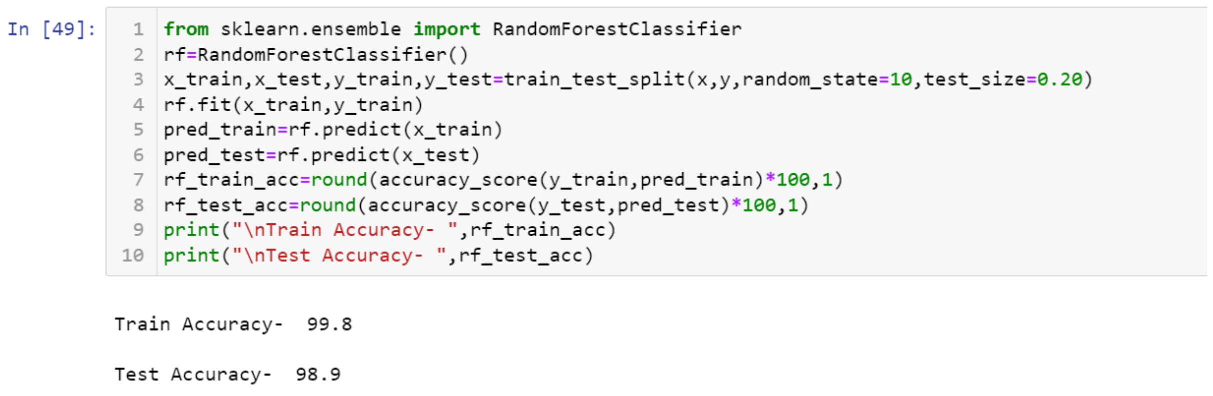
```
In [40]:  print(classification_report(y_test, dt_pred_test))
```

```
              precision    recall  f1-score   support

           0       1.00      0.94      0.97     29048
           1       0.94      1.00      0.97     28663

    accuracy                           0.97     57711
   macro avg       0.97      0.97      0.97     57711
weighted avg       0.97      0.97      0.97     57711
```

Model is poor in recalling class 1 i.e., True than class 0 i.e. False, although precision for class 1 is higher than class 0. F1-score of class 0 is similar to that of class 1.

3) **Random Forest Classifier**

We applied this algorithm and found the train accuracy to be 99.8% and test accuracy to be 98.9%.

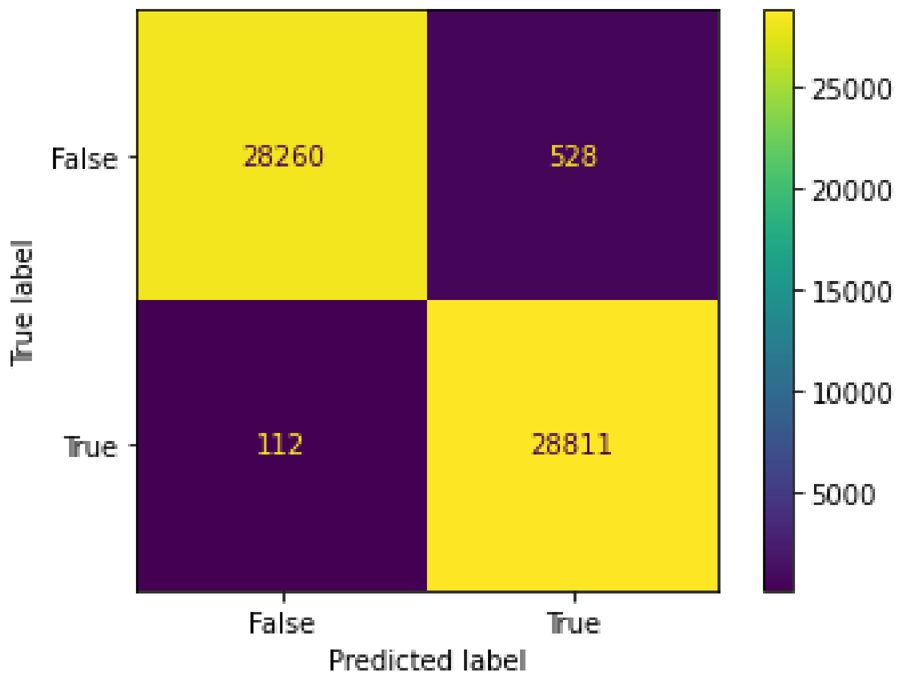


```
In [49]:
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier()
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=10,test_size=0.20)
rf.fit(x_train,y_train)
pred_train=rf.predict(x_train)
pred_test=rf.predict(x_test)
rf_train_acc=round(accuracy_score(y_train,pred_train)*100,1)
rf_test_acc=round(accuracy_score(y_test,pred_test)*100,1)
print("\nTrain Accuracy- ",rf_train_acc)
print("\nTest Accuracy- ",rf_test_acc)
```

```
Train Accuracy-  99.8

Test Accuracy-  98.9
```

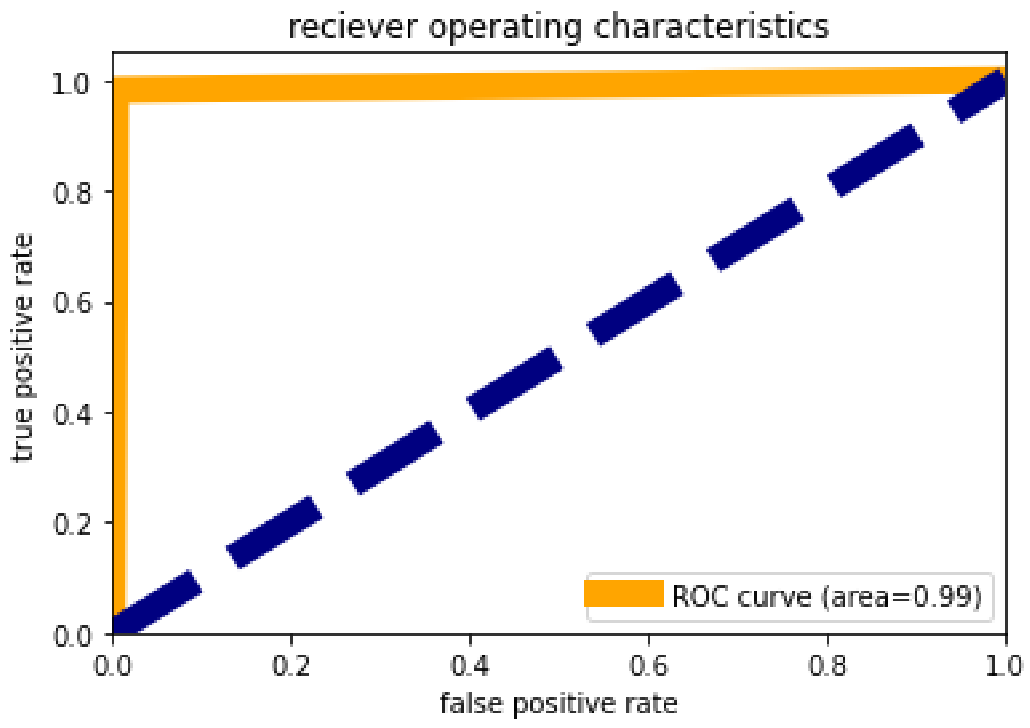
We also tested this model on other metrics:

i) Confusion matrix:



Model is equally good in predicting both the classes.

ii) AUC-ROC Curve:



We observed that the area under the curve is 99% that means 99% times model is predicting accurately and rest all other time it gives wrong prediction. This accuracy is a good one let's check on other metrics as well.

iii) Classification Report:

```
In [53]:  print(classification_report(y_test, pred_test))
```

```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99     28788
           1       0.98      1.00      0.99     28923

    accuracy                           0.99     57711
   macro avg       0.99      0.99      0.99     57711
weighted avg       0.99      0.99      0.99     57711
```

Model is poor in recalling class 1 i.e., True than class 0 i.e. False, although precision for class 1 is higher than class 0. F1-score of class 0 is similar to that of class 1.

4) Knn Classifier

We applied this algorithm and found the train accuracy to be 91.1% and test accuracy to be 85.2%.

```
In [43]:
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier()
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=1,test_size=0.20)
knn.fit(x_train,y_train)
pred_train=knn.predict(x_train)
pred_test=knn.predict(x_test)
knn_train_acc=round(accuracy_score(y_train,pred_train)*100,1)
knn_test_acc=round(accuracy_score(y_test,pred_test)*100,1)
print("\nTrain Accuracy- ",knn_train_acc)
print("\nTest Accuracy- ",knn_test_acc)
```

```
Train Accuracy-  91.1

Test Accuracy-  85.2
```

Cross Validation Score

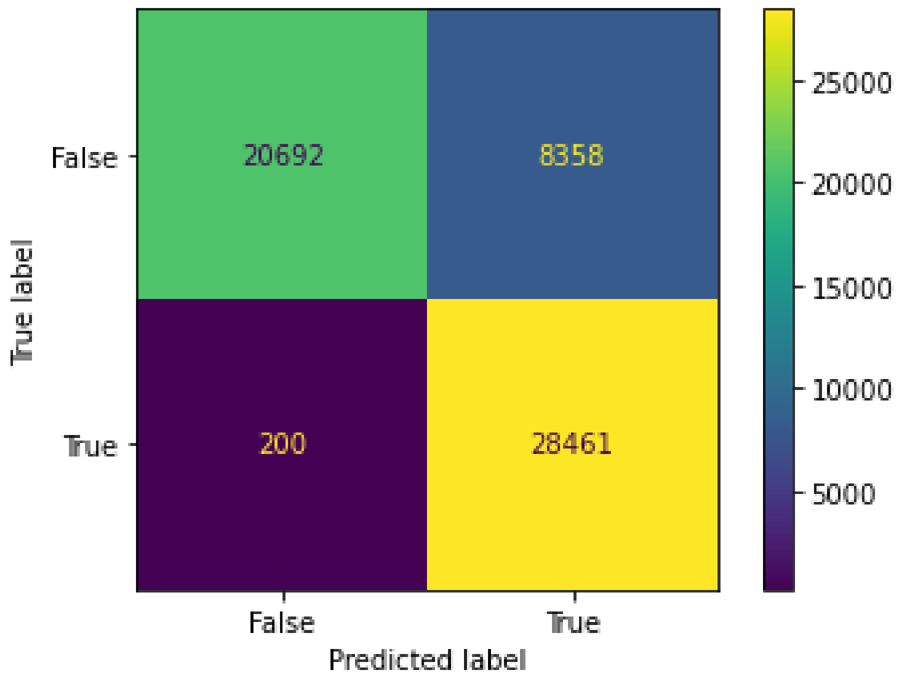
```
In [47]:
cv_score_best_knn=cross_val_score(knn,x,y,cv=11).mean()*100
print("cross validation score is-",cv_score_best_knn)
print("accuracy score for Knn classifier model is-",knn_test_acc)
```

```
cross validation score is- 86.37585674357695
accuracy score for Knn classifier model is- 85.2
```

The test accuracy for logistic regression is 85.2% and its cv score is 86.3% thus making us sure that the model is not overfitted, although decision tree classifier is prone to overfitting.

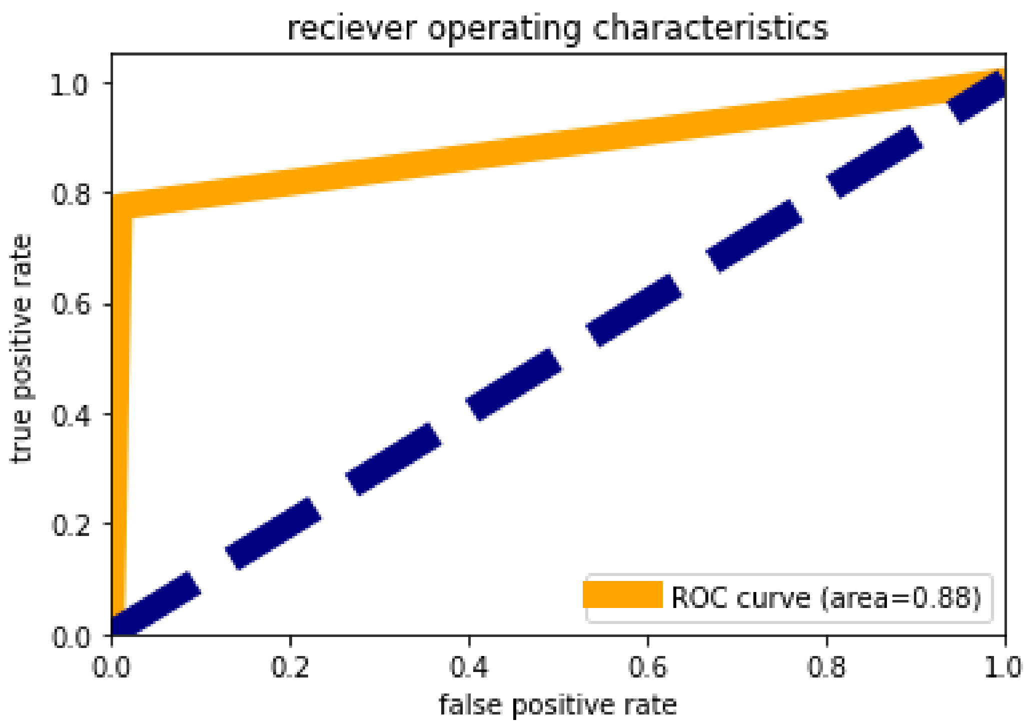
We also tested this model on other metrics:

i) Confusion matrix:



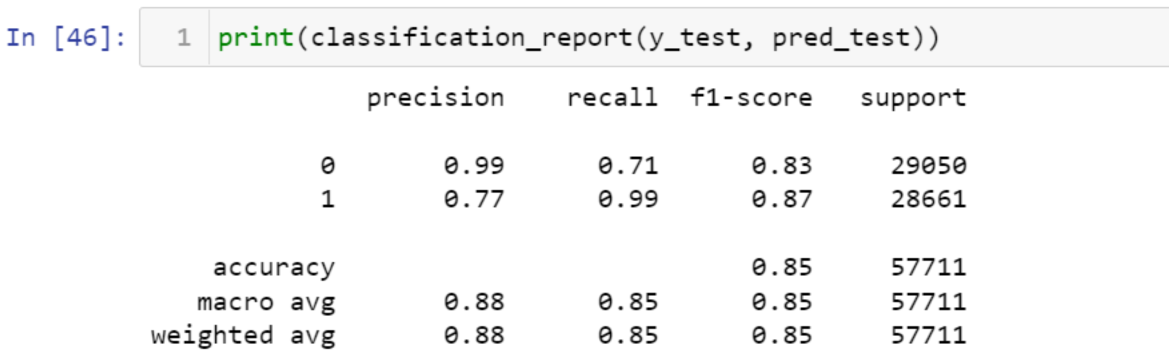
Model is good in predicting false classes but bad in predicting true as well as false classes. Although we can also see that the samples for true classes and samples for false classes are also equal.

ii) AUC-ROC Curve:



We observed that the area under the curve is 88% that means 88% times model is predicting accurately and rest all other time it gives wrong prediction. This accuracy is a good one let’s check on other metrics as well.

iii) Classification Report:



```
In [46]:  print(classification_report(y_test, pred_test))
```

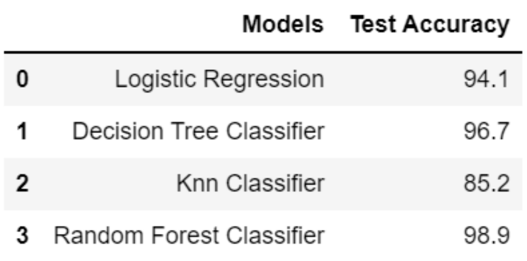
```
              precision    recall  f1-score   support

           0       0.99      0.71      0.83     29050
           1       0.77      0.99      0.87     28661

    accuracy                           0.85     57711
   macro avg       0.88      0.85      0.85     57711
weighted avg       0.88      0.85      0.85     57711
```

Model is poor in recalling class 1 i.e., True than class 0 i.e. False, although precision for class 1 is higher than class 0. F1-score of class 0 is similar to that of class 1.

Conclusion for Model

There are three models with least difference between accuracy and cv score but the best performing model is random forest as its confusion matrix, auc-roc curve and recall, f1-score is the best among all. Since the model is already giving its best accuracy so we wont apply hyperparameter tuning as it’s a very big data so we will save our random forest model.

:



	Models	Test Accuracy
0	Logistic Regression	94.1
1	Decision Tree Classifier	96.7
2	Knn Classifier	85.2
3	Random Forest Classifier	98.9