MALIGNANT COMMENTS CLASSIFIER PROJECT

PROBLEM STATEMENT:

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.

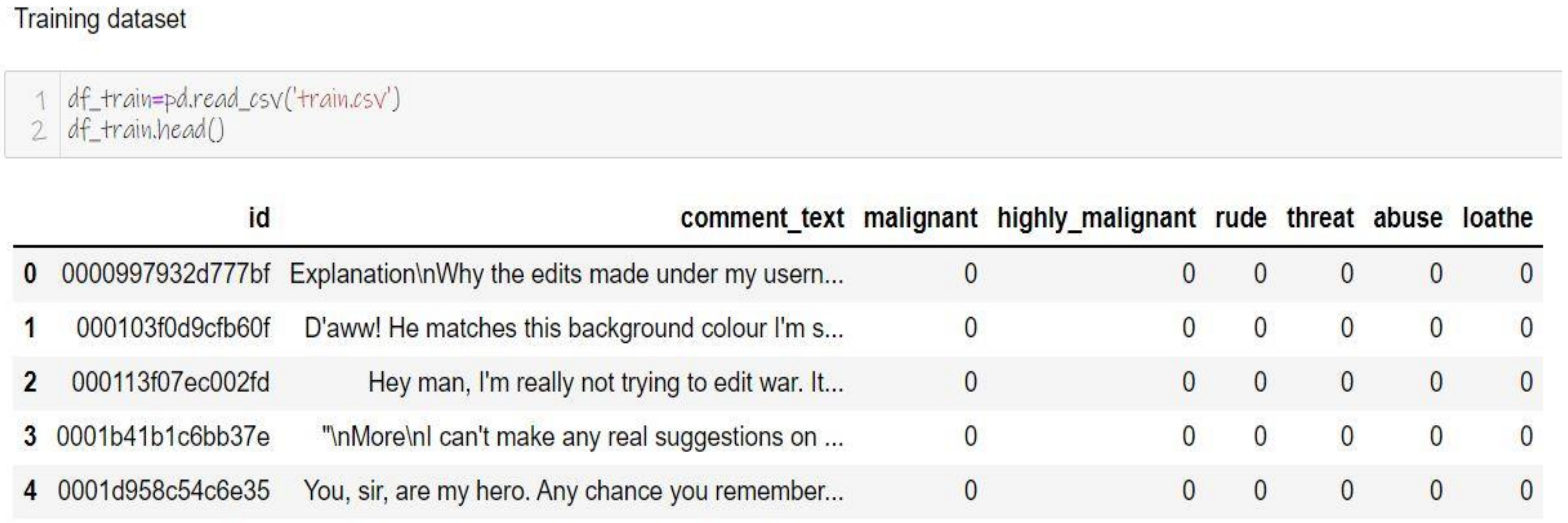
Our goal is to build a prototype of online hate and abuse comment classifier which can used to classify hate and offensive comments so that it can be controlled and restricted from spreading hatred and cyberbullying.

UNDERSTANDING:

The project aims towards the social media which enables people to express their opinions widely online. It passionately and relentlessly malevolent type of work which means aggressively malicious in nature which can hurt others sentiment. 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.

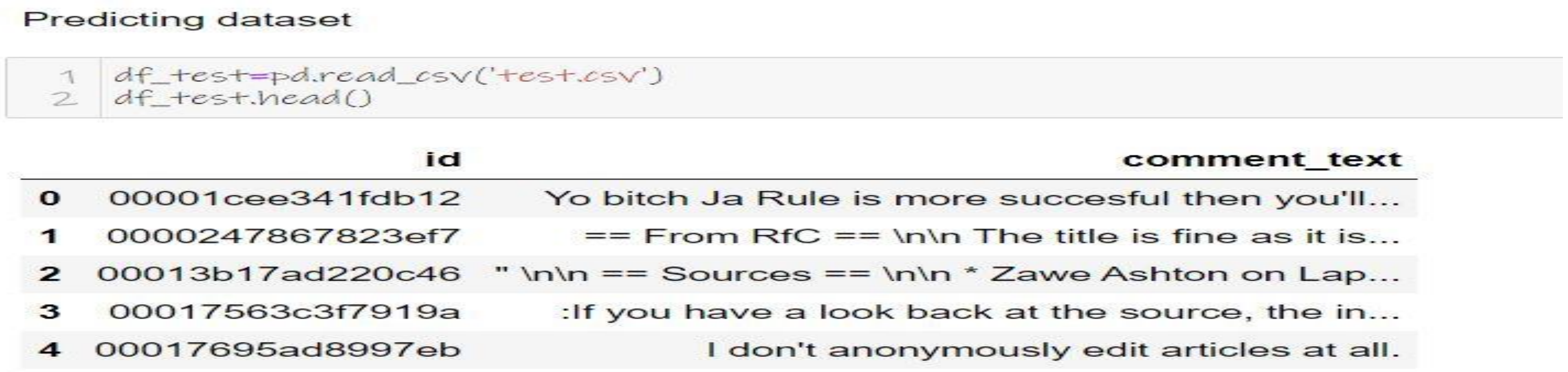
EDA STEPS AND VISUALIZATIONS:

1. Data Collection


Training dataset

```
df_train=pd.read_csv('train.csv')
df_train.head()
```

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

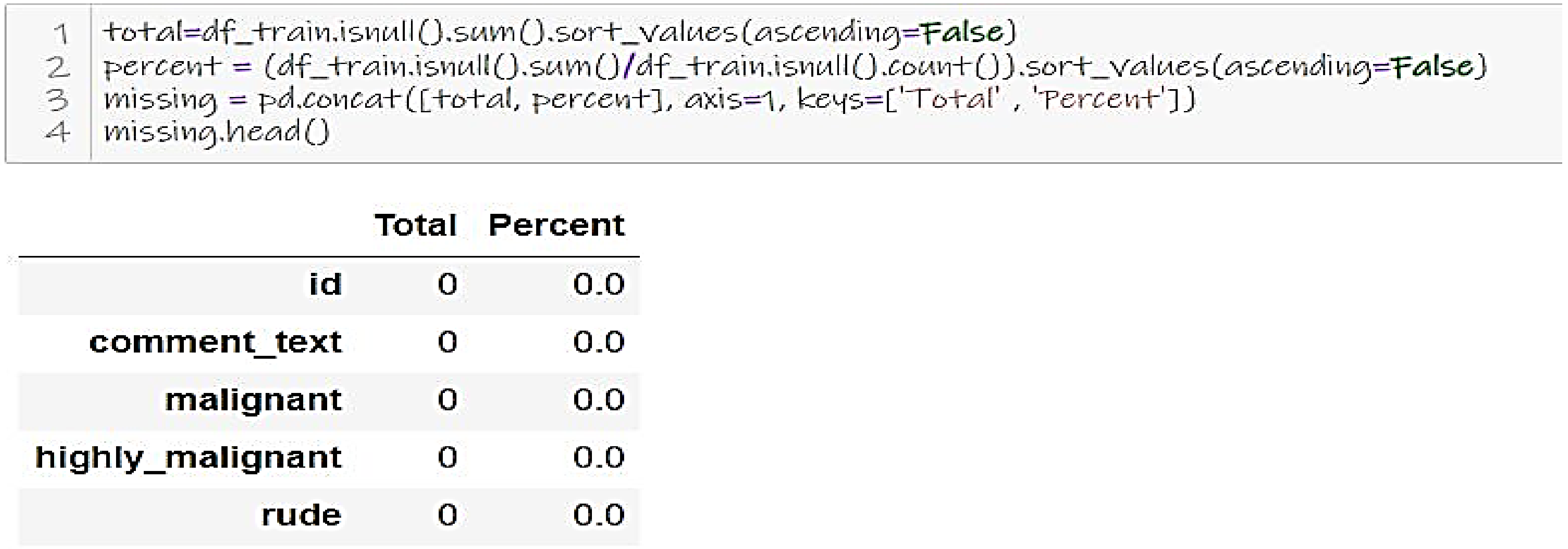


Predicting dataset

```python
df_test=pd.read_csv('test.csv')
df_test.head()
```

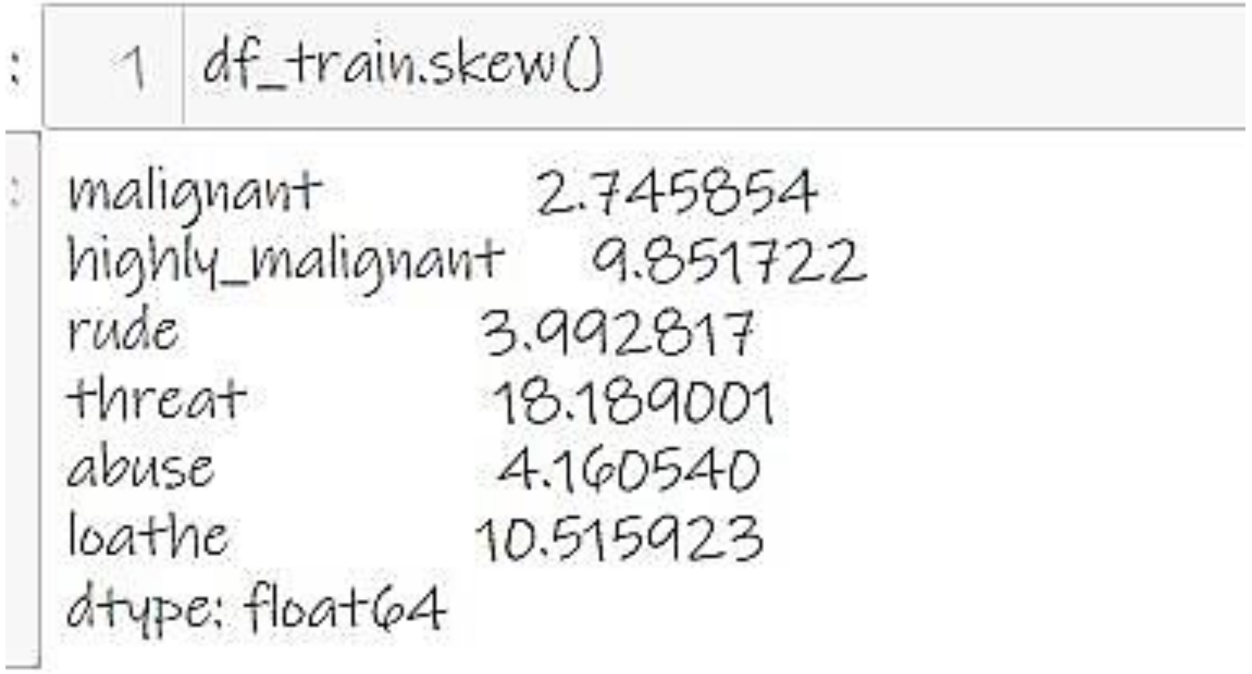
	id	comment_text
0	00001cee341fdb12	Yo bitch Ja Rule is more succesful then you'll...
1	0000247867823ef7	== From RfC == \n\n The title is fine as it is...
2	00013b17ad220c46	" \n\n == Sources == \n\n * Zawe Ashton on Lap...
3	00017563c3f7919a	:If you have a look back at the source, the in...
4	00017695ad8997eb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

2. Data Cleaning



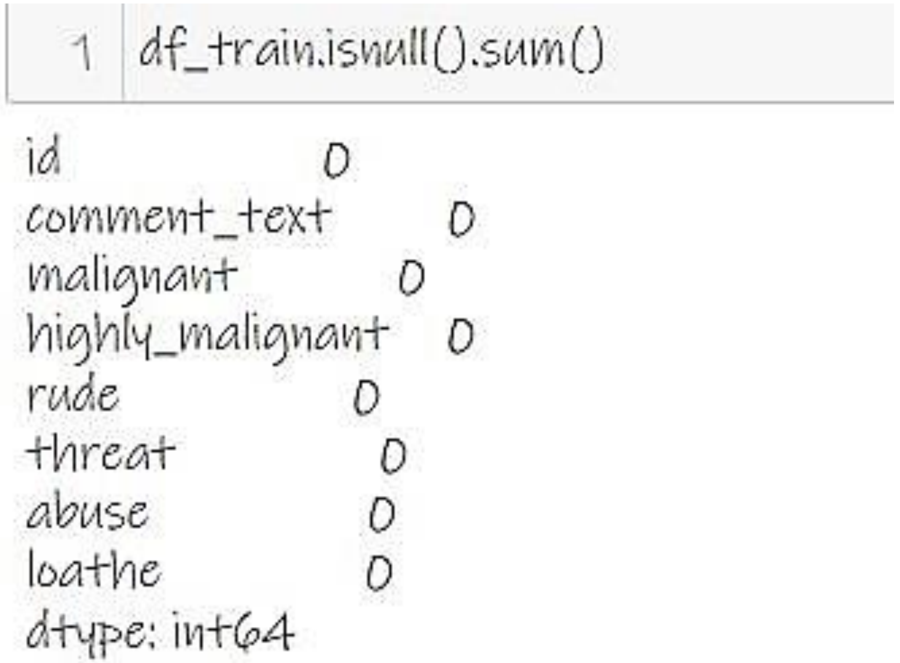
```python
total=df_train.isnull().sum().sort_values(ascending=False)
percent = (df_train.isnull().sum()/df_train.isnull().count()).sort_values(ascending=False)
missing = pd.concat([total, percent], axis=1, keys=['Total' , 'Percent'])
missing.head()
```

	Total	Percent
id	0	0.0
comment_text	0	0.0
malignant	0	0.0
highly_malignant	0	0.0
rude	0	0.0



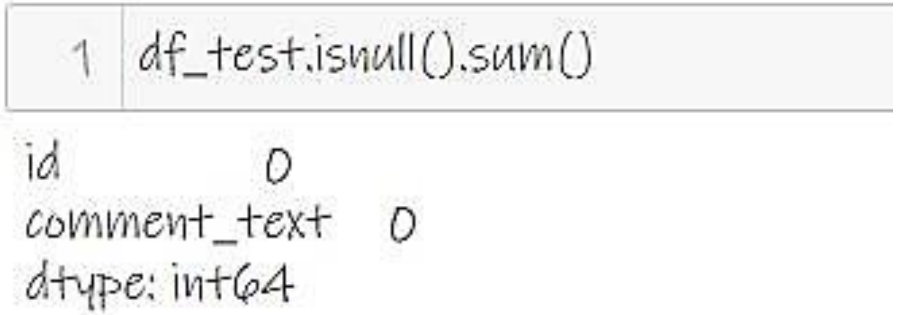
```
df_train.skew()
```

```
malignant            2.745854
highly_malignant     9.851722
rude                 3.992817
threat              18.189001
abuse                4.160540
loathe              10.515923
dtype: float64
```



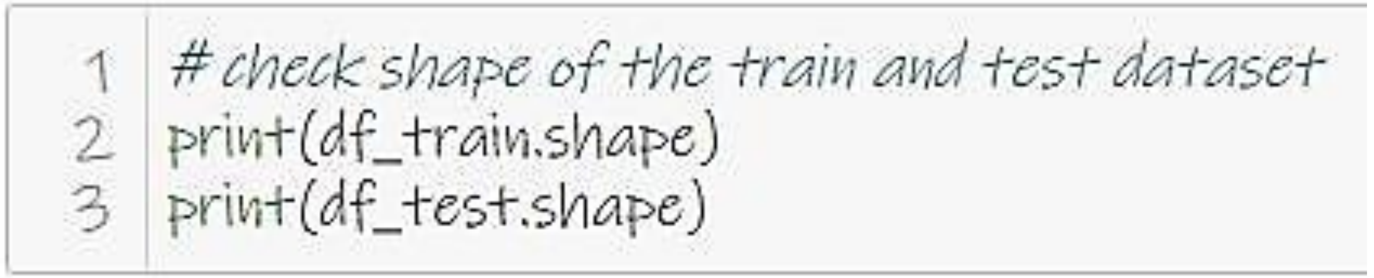
```
df_train.isnull().sum()
```

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



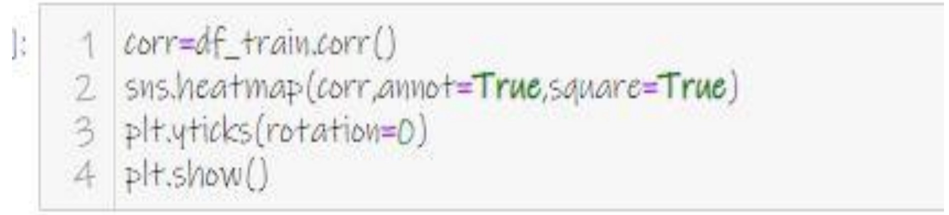
```
df_test.isnull().sum()
```

```
id              0
comment_text    0
dtype: int64
```

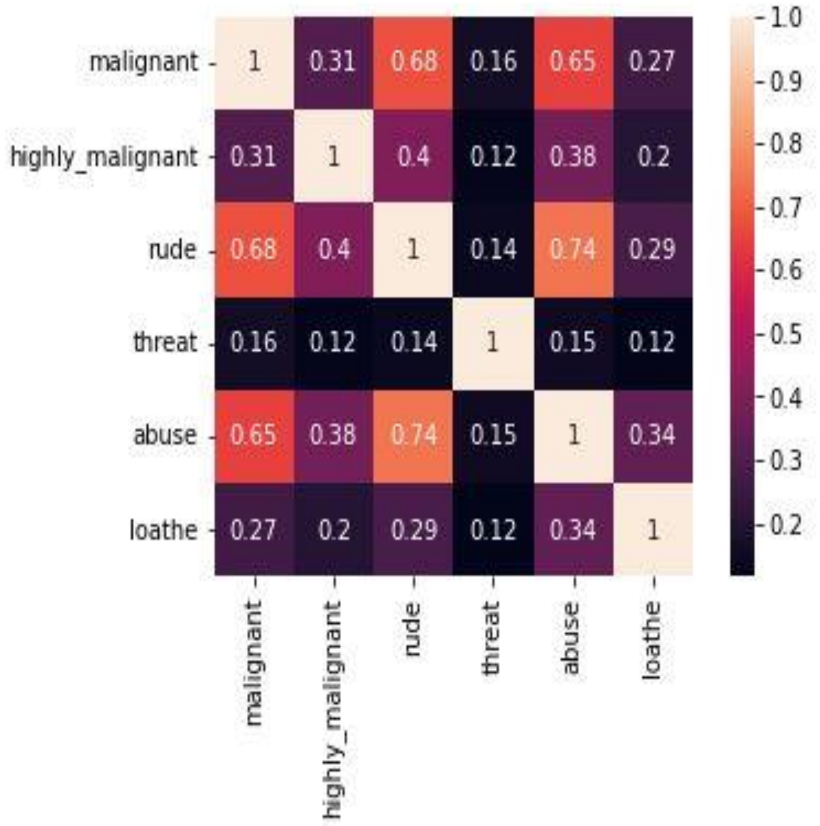


```
# check shape of the train and test dataset
print(df_train.shape)
print(df_test.shape)
```

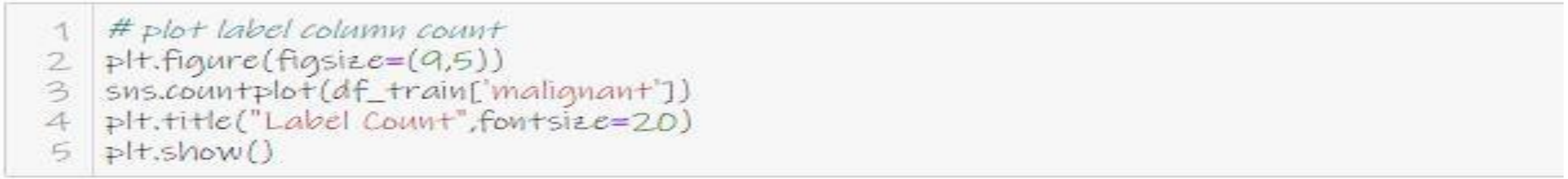
```
(159571, 8)
(153164, 2)
```



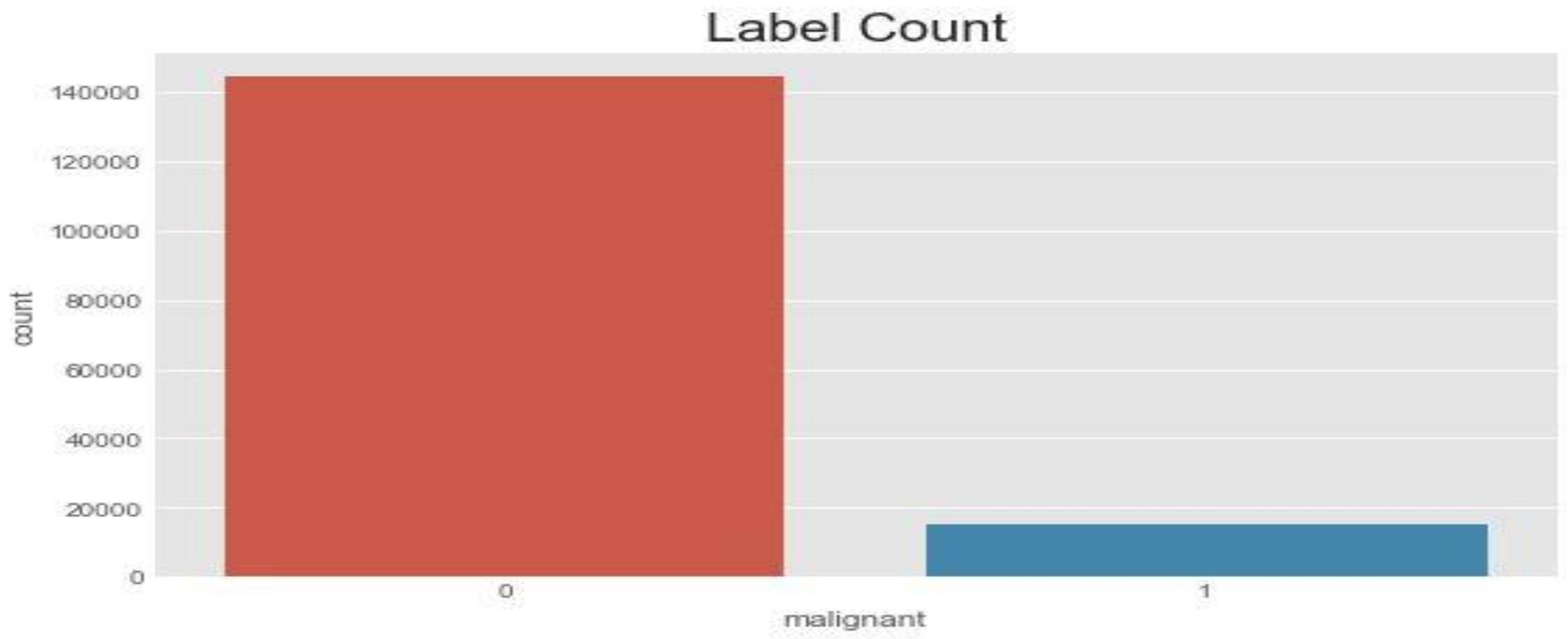
```
corr=df_train.corr()
sns.heatmap(corr,annot=True,square=True)
plt.yticks(rotation=0)
plt.show()
```



	malignant	highly_malignant	rude	threat	abuse	loathe
malignant	1	0.31	0.68	0.16	0.65	0.27
highly_malignant	0.31	1	0.4	0.12	0.38	0.2
rude	0.68	0.4	1	0.14	0.74	0.29
threat	0.16	0.12	0.14	1	0.15	0.12
abuse	0.65	0.38	0.74	0.15	1	0.34
loathe	0.27	0.2	0.29	0.12	0.34	1

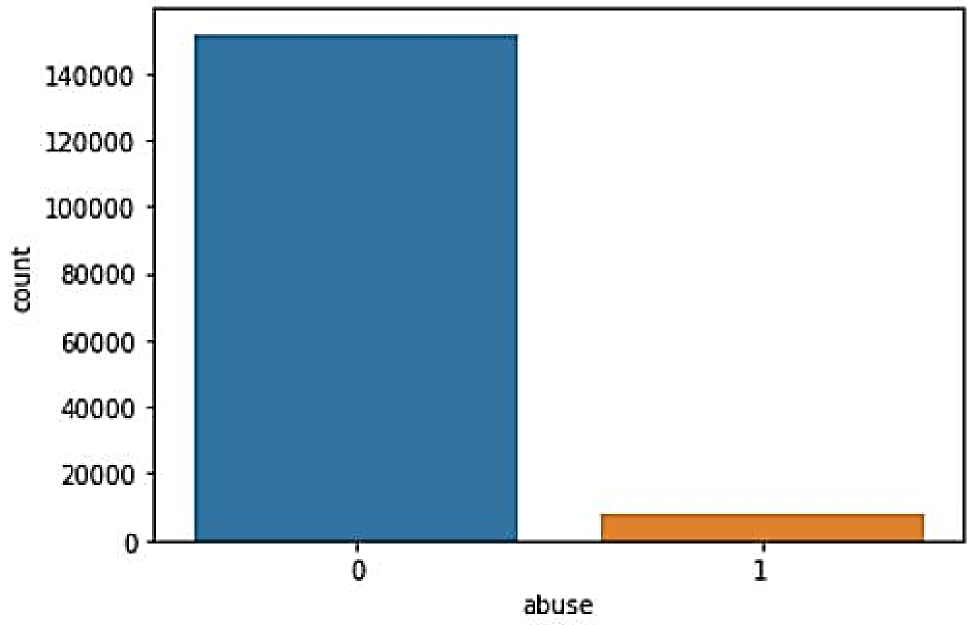


```
# plot label column count
plt.figure(figsize=(9,5))
sns.countplot(df_train['malignant'])
plt.title("Label Count",fontsize=20)
plt.show()
```



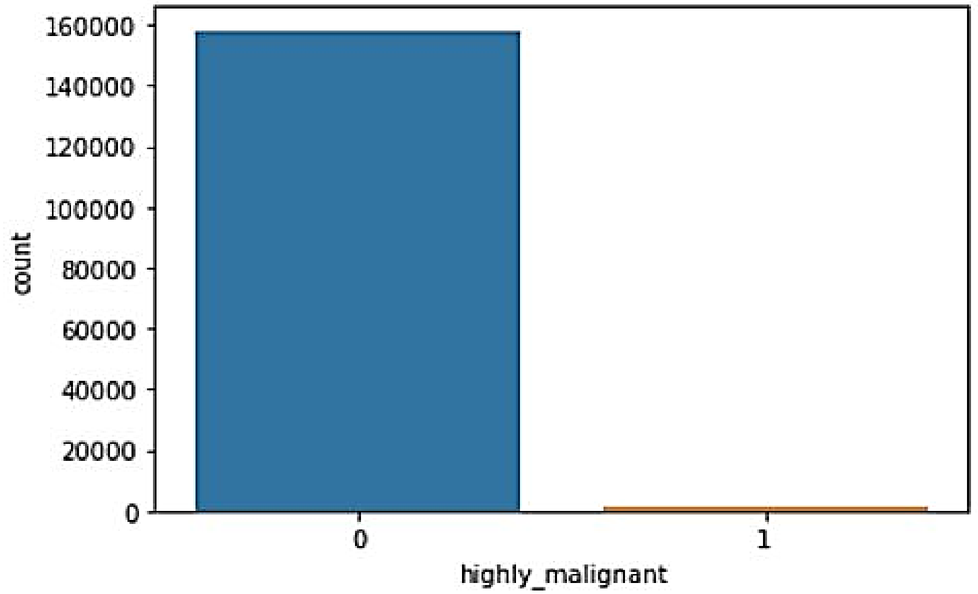
abuse

```
0    151694
1      7877
Name: abuse, dtype: int64
```



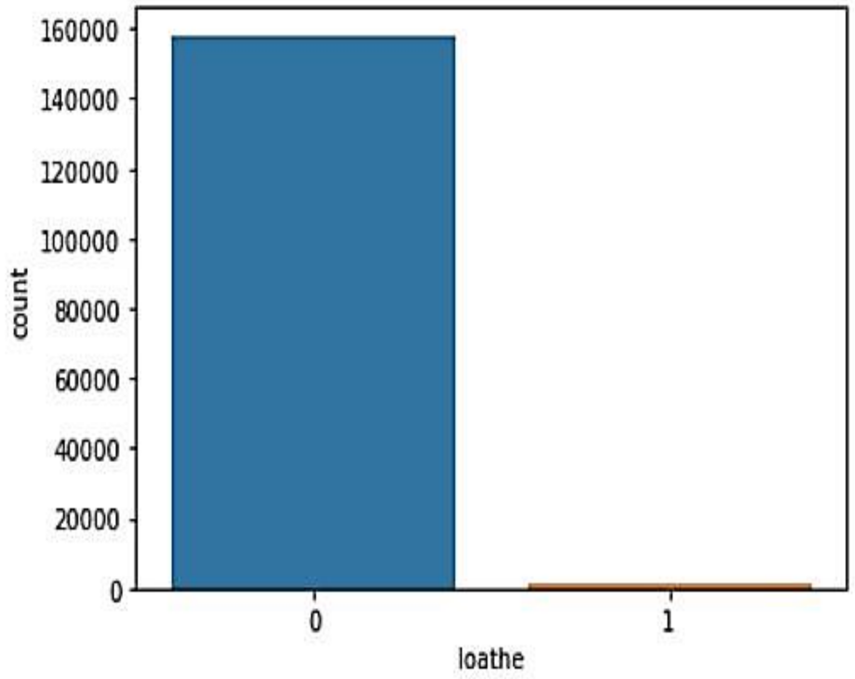
highly_malignant

```
0    157976
1      1595
Name: highly_malignant, dtype: int64
```



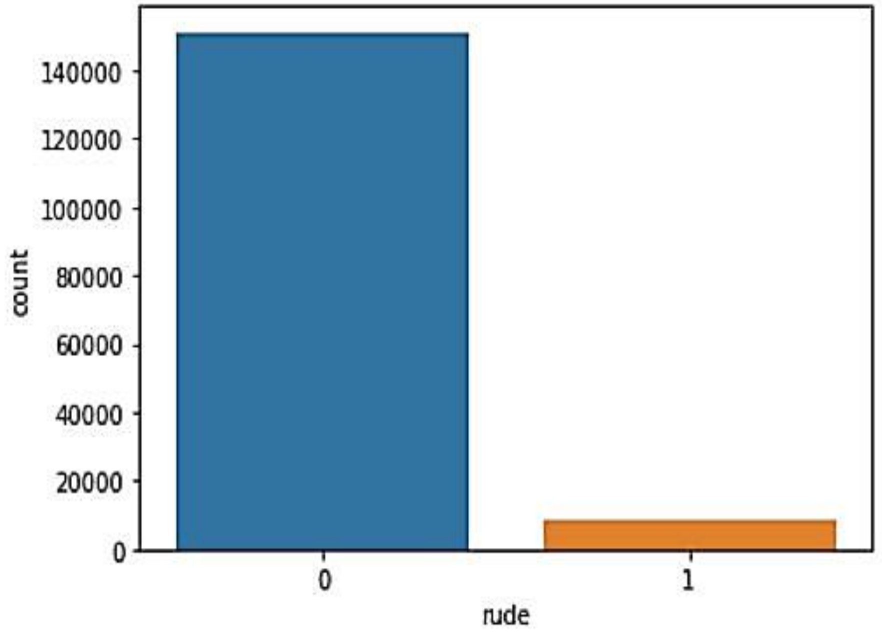
loathe

```
0    158166
1      1405
Name: loathe, dtype: int64
```



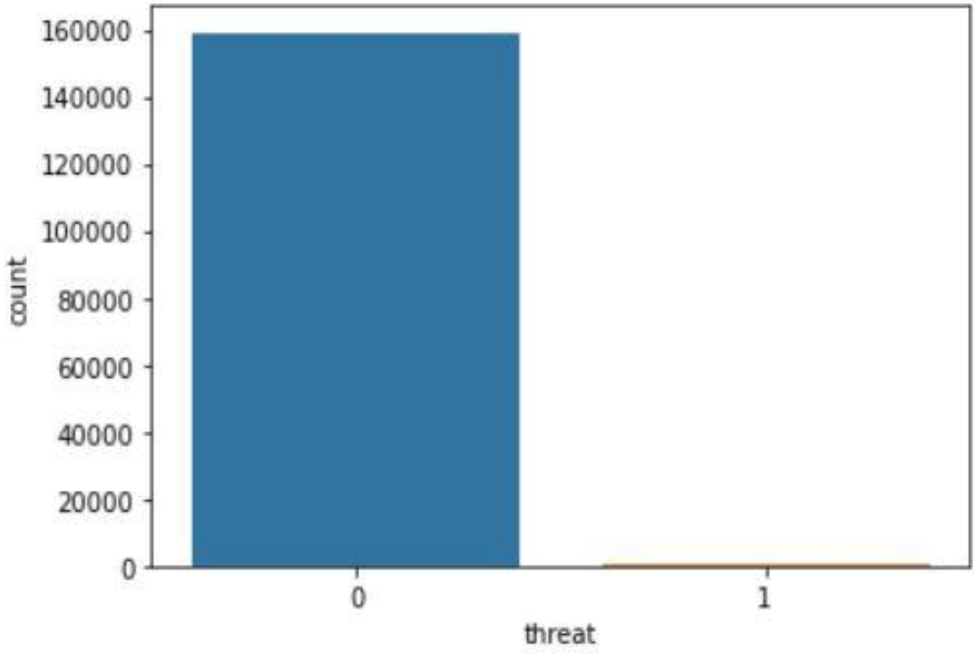
rude

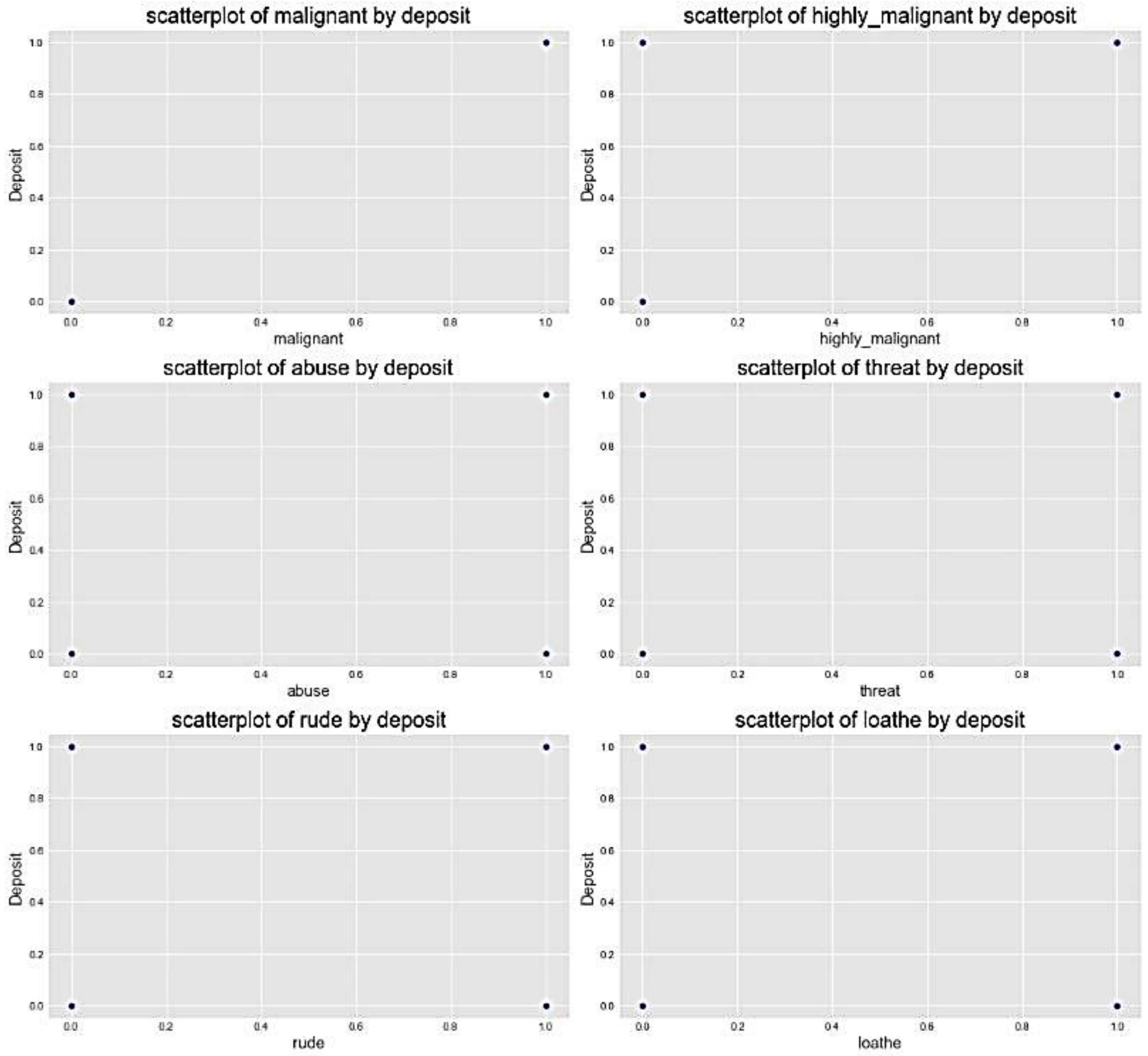
```
0    151122
1      8449
Name: rude, dtype: int64
```

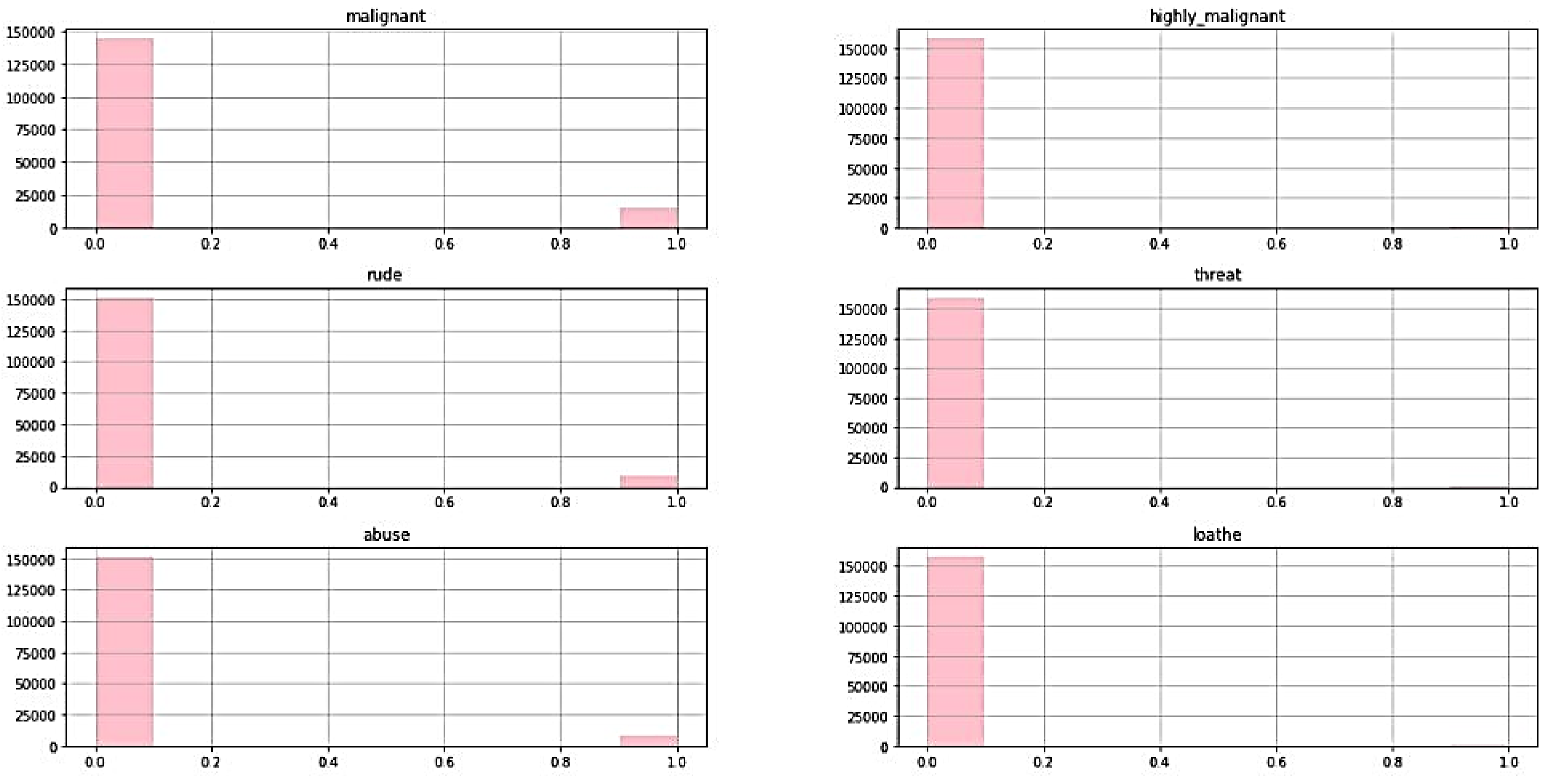


threat

```
0    159093
1       478
Name: threat, dtype: int64
```





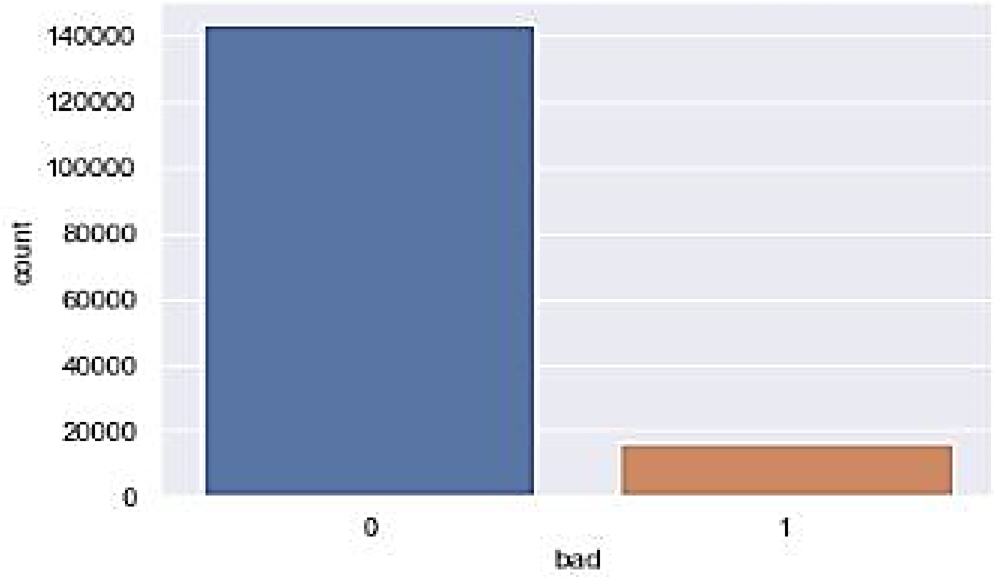


malignant
highly_malignant
rude
threat
abuse
loathe
0
25000
50000
75000
100000
125000
150000
0.0
0.2
0.4
0.6
0.8
1.0

```
target_data = df_train[target_cols]
df_train['bad']=df_train[target_cols].sum(axis=1)
print(df_train['bad'].value_counts())
df_train['bad'] = df_train['bad'] > 0
df_train['bad'] = df_train['bad'].astype(int)
print(df_train['bad'].value_counts())
```

```
0   143346
1     6360
3     4209
2     3480
4     1760
5      385
6       31
Name: bad, dtype: int64
0   143346
1    16225
Name: bad, dtype: int64
```

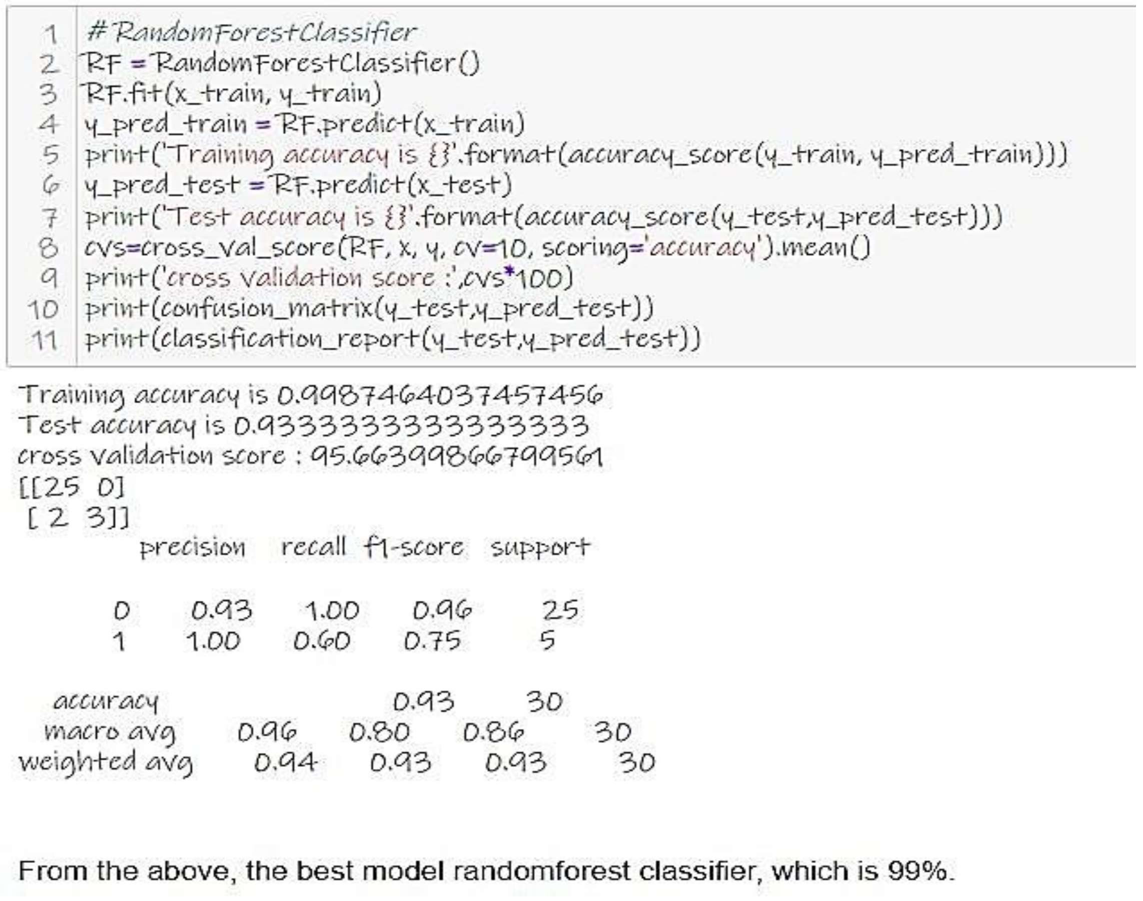
```
sns.set()
sns.countplot(x="bad",data=df_train)
plt.show()
```



STEPS AND ASSUMPTIONS TO COMPLETE THE PROJECT:

- The final model of the independent variables and dependent variables are exactly vary with the variables and offers a significant performance boost over the logistic regression model, about accuracy of training model 93%. So far in the abstract about F1 scores.
- This model has 93% accuracy. The model performed at predicting a malignant comment is recall. It achieved a recall score of 0.60; 60% of the actual malignant comments as malignant. If we used recall as a training objective, it would classify every comment as malignant and reach 100% recall and make every clean comment a false positive.
- It's necessary then to strike a balance between precision and recall.
- False positives waste time, while fast negatives allow malignant to fall through the cracks.
- This model is robust enough for this application and it offers a large advantage over both the standard approach of human flagging for review and an out-of-the-box model.
- Of the comments would be submitted to a moderator review by the model, 60% are malignant.
- An effective tool that would both save moderators time and efficiently catch comments that may otherwise fall through the cracks, this is the most important and the most time consuming.
- Manually collecting data daily is efficient to do work with the research data. Each moderator could have a big impact on reducing malignant in the Wikipedia community.

MODEL DASHBOARD

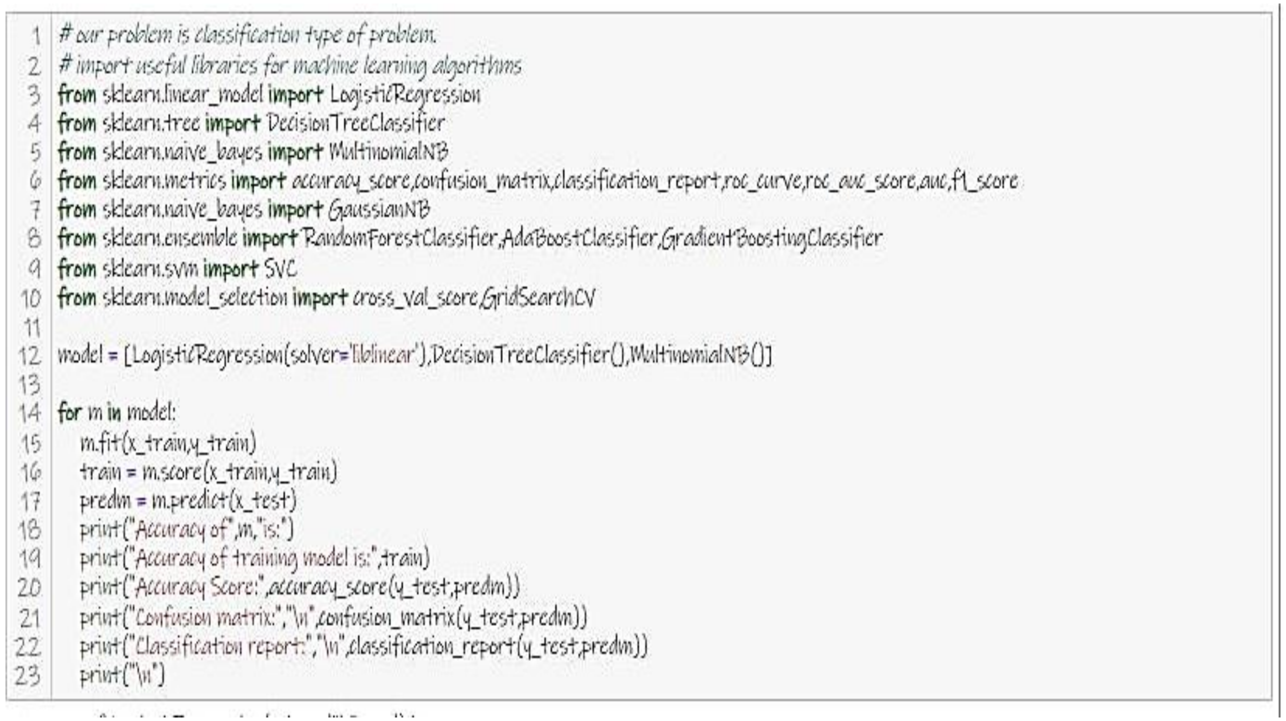


```
#RandomForestClassifier
RF = RandomForestClassifier()
RF.fit(x_train, y_train)
y_pred_train = RF.predict(x_train)
print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
y_pred_test = RF.predict(x_test)
print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
cvs=cross_val_score(RF, x, y, cv=10, scoring='accuracy').mean()
print('cross validation score :',cvs*100)
print(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))
```

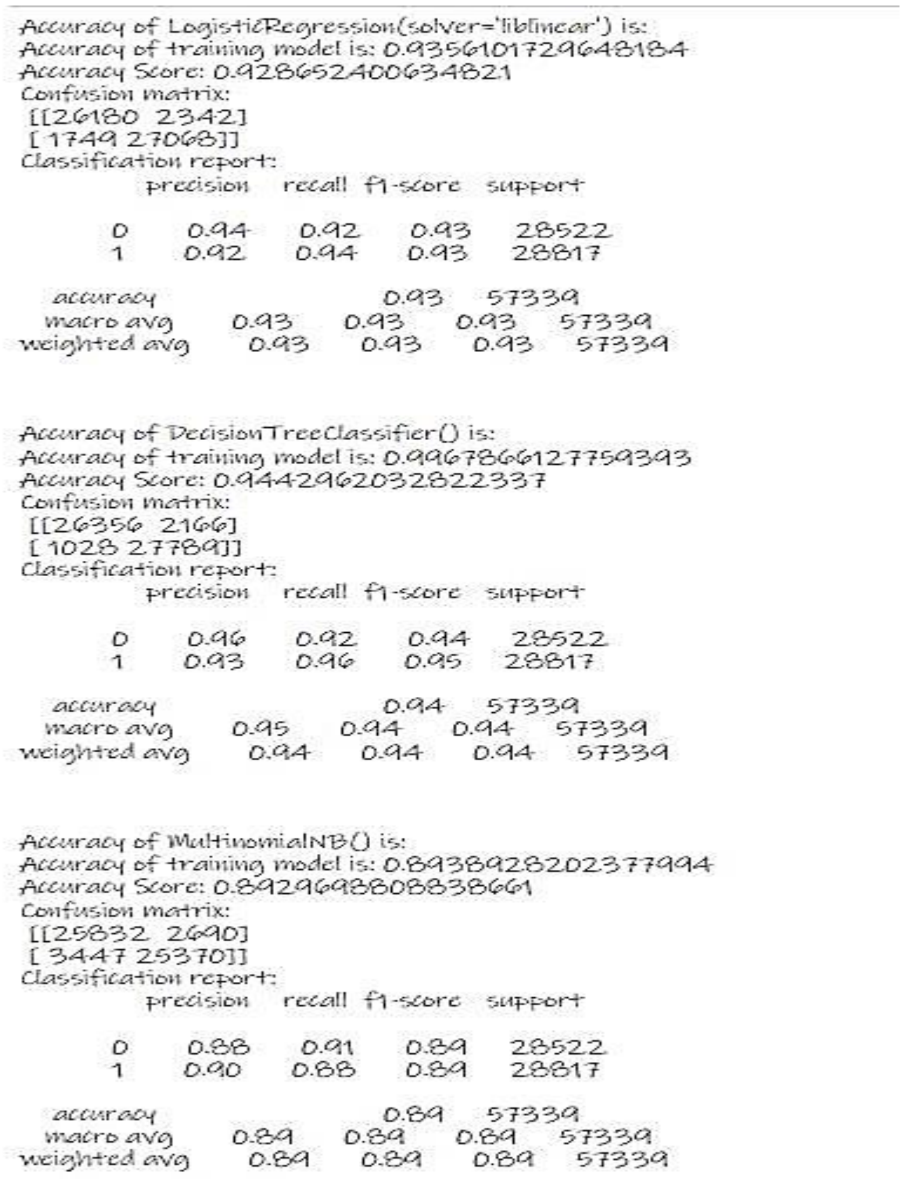
```
Training accuracy is 0.9987464037457456
Test accuracy is 0.9333333333333333
cross validation score : 95.66399866799561
[[25  0]
 [ 2  3]]
              precision    recall  f1-score   support

           0       0.93      1.00      0.96        25
           1       1.00      0.60      0.75         5

    accuracy                           0.93        30
   macro avg       0.96      0.80      0.86        30
weighted avg       0.94      0.93      0.93        30
```

From the above, the best model randomforest classifier, which is 99%.

FINALIZED MODEL:

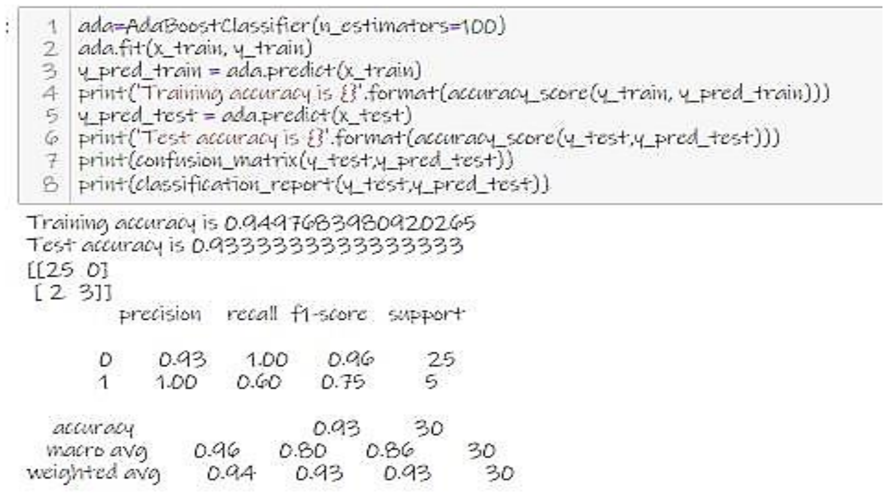


```python
# our problem is classification type of problem.
# import useful libraries for machine learning algorithms
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,roc_curve,roc_auc_score,auc,f1_score
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier,AdaBoostClassifier,GradientBoostingClassifier
from sklearn.svm import SVC
from sklearn.model_selection import cross_val_score,GridSearchCV

model = [LogisticRegression(solver='liblinear'),DecisionTreeClassifier(),MultinomialNB()]

for m in model:
    m.fit(x_train,y_train)
    train = m.score(x_train,y_train)
    predm = m.predict(x_test)
    print("Accuracy of",m,"is:")
    print("Accuracy of training model is:",train)
    print("Accuracy Score:",accuracy_score(y_test,predm))
    print("Confusion matrix:","\n",confusion_matrix(y_test,predm))
    print("Classification report:","\n",classification_report(y_test,predm))
    print("\n")
```



```
Accuracy of LogisticRegression(solver='liblinear') is:
Accuracy of training model is: 0.9356101729648184
Accuracy Score: 0.928652400634821
Confusion matrix:
 [[26180  2342]
 [ 1749 27068]]
Classification report:
               precision    recall  f1-score   support

           0       0.94      0.92      0.93     28522
           1       0.92      0.94      0.93     28817

    accuracy                           0.93     57339
   macro avg       0.93      0.93      0.93     57339
weighted avg       0.93      0.93      0.93     57339


Accuracy of DecisionTreeClassifier() is:
Accuracy of training model is: 0.9967866127759393
Accuracy Score: 0.9442962032822337
Confusion matrix:
 [[26356  2166]
 [ 1028 27789]]
Classification report:
               precision    recall  f1-score   support

           0       0.96      0.92      0.94     28522
           1       0.93      0.96      0.95     28817

    accuracy                           0.94     57339
   macro avg       0.95      0.94      0.94     57339
weighted avg       0.94      0.94      0.94     57339


Accuracy of MultinomialNB() is:
Accuracy of training model is: 0.8938928202377994
Accuracy Score: 0.8929698808838661
Confusion matrix:
 [[25832  2690]
 [ 3447 25370]]
Classification report:
               precision    recall  f1-score   support

           0       0.88      0.91      0.89     28522
           1       0.90      0.88      0.89     28817

    accuracy                           0.89     57339
   macro avg       0.89      0.89      0.89     57339
weighted avg       0.89      0.89      0.89     57339
```

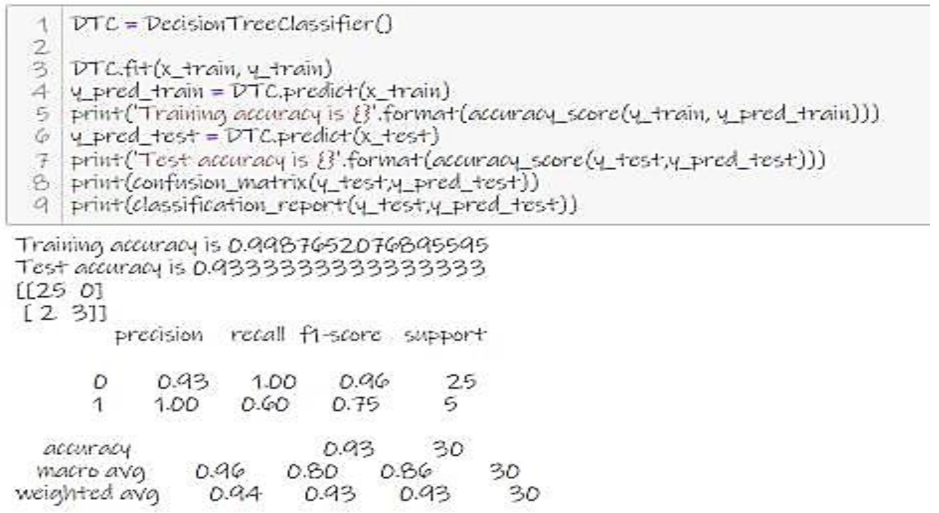
AdaBoostClassifier



```
ada=AdaBoostClassifier(n_estimators=100)
ada.fit(x_train, y_train)
y_pred_train = ada.predict(x_train)
print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
y_pred_test = ada.predict(x_test)
print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
print(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))
```

```
Training accuracy is 0.9497683980920265
Test accuracy is 0.9333333333333333
[[25  0]
 [ 2  3]]
              precision    recall  f1-score   support

           0       0.93      1.00      0.96        25
           1       1.00      0.60      0.75         5

    accuracy                           0.93        30
   macro avg       0.96      0.80      0.86        30
weighted avg       0.94      0.93      0.93        30
```

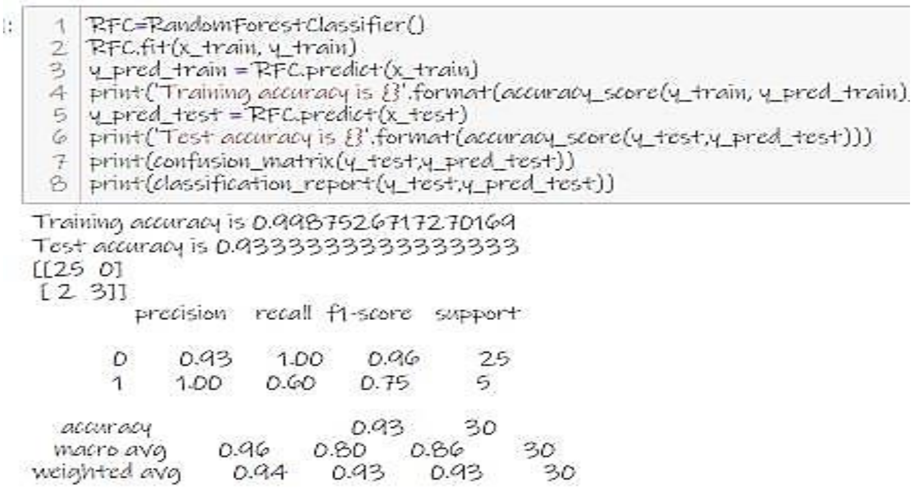
Decision Tree Classifier



```
DTC = DecisionTreeClassifier()

DTC.fit(x_train, y_train)
y_pred_train = DTC.predict(x_train)
print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
y_pred_test = DTC.predict(x_test)
print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
print(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))
```

```
Training accuracy is 0.9987652076895595
Test accuracy is 0.9333333333333333
[[25  0]
 [ 2  3]]
              precision    recall  f1-score   support

           0       0.93      1.00      0.96        25
           1       1.00      0.60      0.75         5

    accuracy                           0.93        30
   macro avg       0.96      0.80      0.86        30
weighted avg       0.94      0.93      0.93        30
```

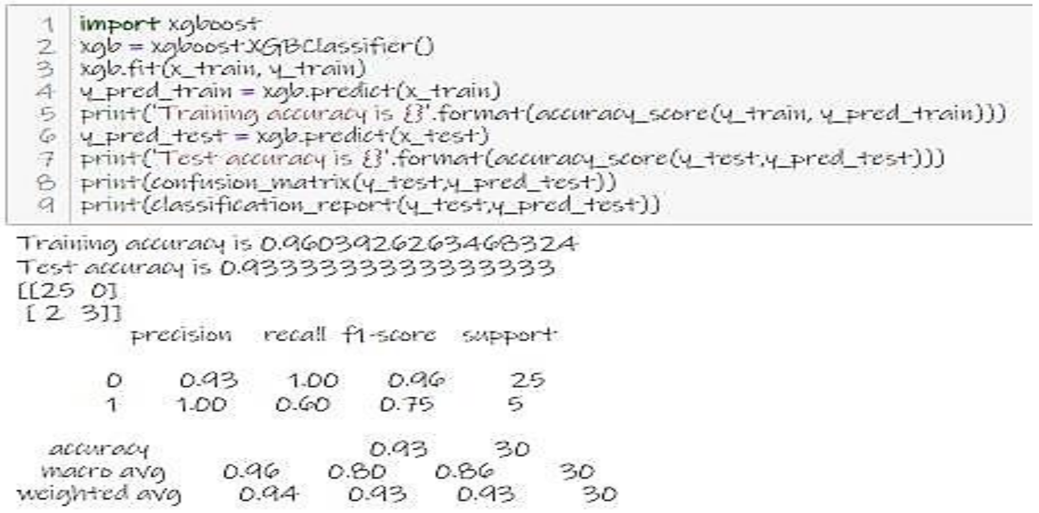
Random Forest Classifier



```
RFC=RandomForestClassifier()
RFC.fit(x_train, y_train)
y_pred_train = RFC.predict(x_train)
print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)
y_pred_test = RFC.predict(x_test)
print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
print(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))
```

```
Training accuracy is 0.9987526717270169
Test accuracy is 0.9333333333333333
[[25  0]
 [ 2  3]]
              precision    recall  f1-score   support

           0       0.93      1.00      0.96        25
           1       1.00      0.60      0.75         5

    accuracy                           0.93        30
   macro avg       0.96      0.80      0.86        30
weighted avg       0.94      0.93      0.93        30
```

XGBOOST



```
import xgboost
xgb = xgboost.XGBClassifier()
xgb.fit(x_train, y_train)
y_pred_train = xgb.predict(x_train)
print('Training accuracy is {}'.format(accuracy_score(y_train, y_pred_train)))
y_pred_test = xgb.predict(x_test)
print('Test accuracy is {}'.format(accuracy_score(y_test,y_pred_test)))
print(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))
```

```
Training accuracy is 0.9603926263468324
Test accuracy is 0.9333333333333333
[[25  0]
 [ 2  3]]
              precision    recall  f1-score   support

           0       0.93      1.00      0.96        25
           1       1.00      0.60      0.75         5

    accuracy                           0.93        30
   macro avg       0.96      0.80      0.86        30
weighted avg       0.94      0.93      0.93        30
```

CONCLUSION

I would like to conclude here that by doing research in the topic by through some points:

- Analyze the problem and purpose a useful solution
- Explore the dataset to get better picture of how the labels are distributed, how they correlate with each other
- Develop an objective that fits a practical use case and addresses the major class imbalance
- Create a baseline score with a simple logistic regression classifier
- Explore the effectiveness of multiple machine learning algorithms
- Select the best model based on a balance of performance and efficiency
- Tune the model parameters to maximize performance
- Build the final model with the best performing algorithm and parameters and test it on a holdout subset of the data
- And the final model offered about 93% performance gain over the initial benchmark model, which makes it an effective solution to the problem.
- By using multiple models, a sort of divide the conquer method where the problem is divided into multiple smaller, contextual problems. While the solution generalizes to the entire dataset, no one solution will be able to generalize perfectly to the diverse variety of inputs from Internet users. By training models on different situations, like a model that's only been trained on short or long comments, to only detect whether a comment is malignant when profanity is present by use a simple decision tree to feel comments into the model that would be most effective.

THANK YOU