MALIGNANT COMMENTS CLASSIFICATION PROJECT

Submitted by:

Akash Kanjwani

ACKNOWLEDGMENT

During the process of completing this project, I have referred following materials for which I owe them great gratitude.

1. For theoretical knowledge https://towardsdatascience.com/
2. Data trained video tutorials.
3. Scikit-learn https://scikit-learn.org/stable/
4. Machine Learning for Dummies by John Mueller and Luca Massaron - Easy to understand for a beginner book.
5. Geeksforgeeks. https://www.geeksforgeeks.org/

Besides that all the observation, creations of the models and graphs done by self help.

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.

Data Set Description

The data set contains the training set, which has approximately 1,59,000 samples and the test set which contains nearly 1,53,000 samples. All the data samples contain 8 fields which includes 'Id', 'Comments', 'Malignant', 'Highly malignant', 'Rude', 'Threat', 'Abuse' and 'Loathe'.

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.

The data set includes:

- **Malignant:** It is the Label column, which includes values 0 and 1, denoting if the comment is malignant or not.
- **Highly Malignant:** It denotes comments that are highly malignant and hurtful.
- **Rude:** It denotes comments that are very rude and offensive.
- **Threat:** It contains indication of the comments that are giving any threat to someone.
- **Abuse:** It is for comments that are abusive in nature.
- **Loathe:** It describes the comments which are hateful and loathing in nature.
- **ID:** It includes unique Ids associated with each comment text given.
- **Comment text:** This column contains the comments extracted from various social media platforms.

Analytical Problem Framing

Mathematical/ Analytical Modelling of the Problem

1) The size of table is 159571 × 8 i.e. no. of rows are 159571 and no. of columns are 8.

2) Out of 8 columns 6 columns are numeric type and 2 columns are object type.

3) Null values are not present in the data set as we can see in this seaborn heatmap, so there is no need to adopt imputation technique.

4) In case of object data type, we will apply the NLP technique to convert the values in the numeric format.

Because this project is based on Natural language processing that is why we will have to adopt NLP technique such as Word Net Lemmatizer, Stop words, Vectorization etc.

Data Sources and their formats

Data has been provided by the Flip Robo technology.

Train 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
...	...	...	...	...	...	...	...	...
159566	ffe987279560d7ff	":::::And for the second time of asking, when ...	0	0	0	0	0	0
159567	ffea4adeee384e90	You should be ashamed of yourself \n\nThat is ...	0	0	0	0	0	0
159568	ffee36eab5c267c9	Spitzer \n\nUmm, theres no actual article for ...	0	0	0	0	0	0
159569	fff125370e4aaaf3	And it looks like it was actually you who put ...	0	0	0	0	0	0
159570	fff46fc426af1f9a	"\nAnd ... I really don't think you understand...	0	0	0	0	0	0

159571 rows × 8 columns

Test Data –

	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.
...	...	...
153159	fffcd0960ee309b5	. \n i totally agree, this stuff is nothing bu...
153160	fffd7a9a6eb32c16	== Throw from out field to home plate. == \n\n...
153161	fffda9e8d6fafa9e	" \n\n == Okinotorishima categories == \n\n I ...
153162	fffe8f1340a79fc2	" \n\n == ""One of the founding nations of the...
153163	ffffce3fb183ee80	" \n :::Stop already. Your bullshit is not wel...

153164 rows × 2 columns

```
train.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
```

```
train.iloc[:,2:].sum()
```

```
malignant           15294
highly_malignant     1595
rude                 8449
threat                478
abuse                7877
loathe               1405
dtype: int64
```

```
train.isnull().sum()
```

```
id                  0
comment_text        0
malignant           0
highly_malignant    0
rude                0
threat              0
abuse               0
```

Data Pre-processing Done

Because the project is based on NLP that is why we have adopted some NLP technique like Vectorization, Lemmatization, stop words.

```
import string
```

```
train['comment_text']=train['comment_text'].str.lower()
```

```
train.drop(['id'],axis=1,inplace=True)
```

```
train['comment_text']=train['comment_text'].str.replace("won't","will not")
```

```
train['comment_text']=train['comment_text'].str.replace("can't","can not")
```

```
train['comment_text']=train['comment_text'].str.replace("weren't","were not")
```

```
train['comment_text']=train['comment_text'].str.replace(r"\'t","not")
train['comment_text']=train['comment_text'].str.replace(r"\'re","are")
train['comment_text']=train['comment_text'].str.replace(r"\'d","would")
train['comment_text']=train['comment_text'].str.replace(r"\'ll","will")
train['comment_text']=train['comment_text'].str.replace(r"\'t","not")
train['comment_text']=train['comment_text'].str.replace(r"\'ve","have")
train['comment_text']=train['comment_text'].str.replace(r"\'m","am")
```

```
train['comment_text']=train['comment_text'].replace("[^a-zA-Z]"," ",regex=True)
```

```
from nltk.stem import WordNetLemmatizer
```

```
stop_words = set(stopwords.words('english') + ['u', 'ü', 'ur', '4', '2', 'im', 'dont', 'doin', 'ure'])
train['comment_text'] = train['comment_text'].apply(lambda x: ' '.join(term for term in x.split() if term not in stop_words))

lem=WordNetLemmatizer()
train['comment_text'] = train['comment_text'].apply(lambda x: ' '.join(lem.lemmatize(t) for t in x.split()))
```

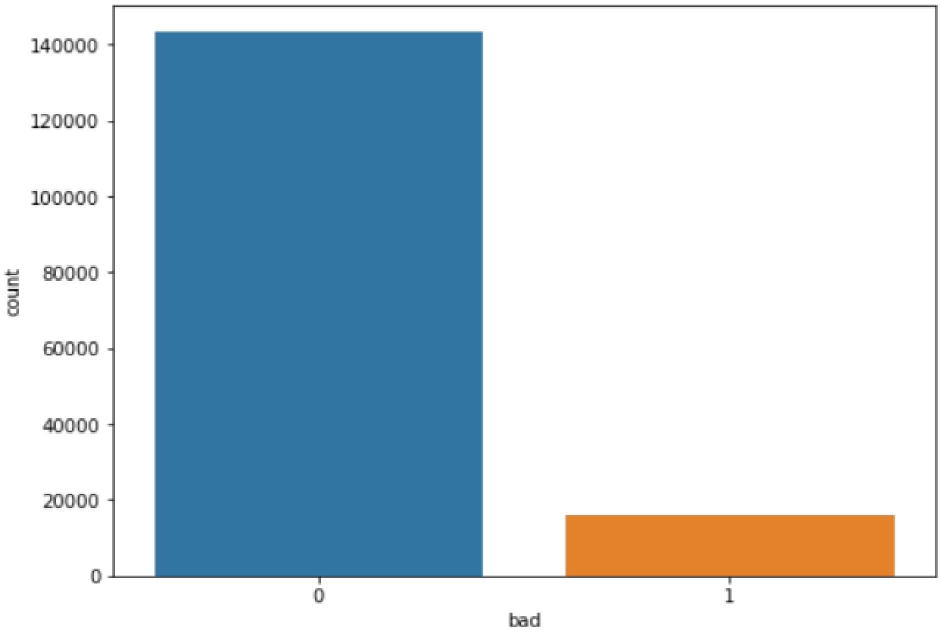
```
cols_output = ['malignant','highly_malignant','rude','threat','abuse','loathe']
```

```
target_data = train[cols_output]WordNetLemmatizer

train['bad']=train[cols_output].sum(axis=1)
print(train['bad'].value_counts())
train['bad']=train['bad']>0
train['bad']=train['bad'].astype(int)
print(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
```

```
plt.figure(figsize=[8,6])
sns.countplot(train['bad'])
plt.show()
```



Hardware and Software Requirements and Tools Used

Anaconda Navigator

Jupyter Notebook

Language-Python

Many lib.-------

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np

import warnings

warnings.filterwarnings('ignore')

import sklearn

from sklearn.linear_model import Logistic Regression

from sklearn.model_selection import
train_test_split,GridSearchCV,cross_val_score

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier,

AdaBoostClassifier,

GradientBoostingClassifier

import xgboost as xg

from sklearn.metrics

import mean_squared_error, mean_absolute_error, r2_score

Pandas- For making data frame

Matplotlib and seaborn- For data visualization

Numpy- For numerical python

From metrice – Classification Report , Confusion metrix , Accuracy score -For checking the model accuracy.

Ensamble- For boosting and bagging

Cross_Val_Score- For cross validation

Algorithms

• Logistic Regression

• Decision Tree Classifier

For Bagging and boosting:

• Random Forest Classifier

• Gradient Bossting Classifier

- AdaBoost Classifier
- XgBoost Classifier

```
from sklearn.feature_extraction.text import TfidfVectorizer
tf_vec=TfidfVectorizer(max_features = 10000,stop_words='english')
features= tf_vec.fit_transform(train['comment_text'])
x=features
```

```
from sklearn.model_selection import train_test_split
```

```
y=train['bad']
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=42,test_size=.20)
```

```
y_train.shape,y_test.shape
```

```
((127656,), (31915,))
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix,classification_report
```

Decision Tree Classifier

```
df = DecisionTreeClassifier()
```

```
df.fit(x_train,y_train)
```

```
DecisionTreeClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
df.score(x_test,y_test)
```

```
0.9411561961460129
```

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

```
print("Accuracy Score :",accuracy_score(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
Accuracy Score : 0.9411561961460129
[[27805   866]
 [ 1012  2232]]
              precision    recall  f1-score   support

           0       0.96      0.97      0.97     28671
           1       0.72      0.69      0.70      3244

    accuracy                           0.94     31915
   macro avg       0.84      0.83      0.84     31915
weighted avg       0.94      0.94      0.94     31915
```

Random Forest Classifier

[338]:
```
rf = RandomForestClassifier()
rf.fit(x_test,y_test)
pred=rf.predict(x_test)
print("Accuracy Score :",accuracy_score(y_test,pred))
print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))
```

```
Accuracy Score : 0.9992480025066584
[[28669     2]
 [   22  3222]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     28671
           1       1.00      0.99      1.00      3244

    accuracy                           1.00     31915
   macro avg       1.00      1.00      1.00     31915
weighted avg       1.00      1.00      1.00     31915
```

Logistic Regression

[339]:
```
lm=LogisticRegression()
lm.fit(x_test,y_test)
pred=lm.predict(x_test)
print("Accuracy Score :",accuracy_score(y_test,pred))
print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))
```

```
Accuracy Score : 0.9521854927150243
[[28633    38]
 [ 1488  1756]]
              precision    recall  f1-score   support

           0       0.95      1.00      0.97     28671
           1       0.98      0.54      0.70      3244

    accuracy                           0.95     31915
   macro avg       0.96      0.77      0.84     31915
weighted avg       0.95      0.95      0.95     31915
```

Gredient Boosting Classifier

[340]:
```
gb = GradientBoostingClassifier()
gb.fit(x_test,y_test)
pred=gb.predict(x_test)
print("Accuracy Score :",accuracy_score(y_test,pred))
print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))
```

```
Accuracy Score : 0.945542848190506
[[28633    38]
 [ 1700  1544]]
              precision    recall  f1-score   support

           0       0.94      1.00      0.97     28671
           1       0.98      0.48      0.64      3244

    accuracy                           0.95     31915
   macro avg       0.96      0.74      0.81     31915
weighted avg       0.95      0.95      0.94     31915
```

Adaboost Classifier

```
ad=AdaBoostClassifier()
ad.fit(x_train,y_train)
pred=ad.predict(x_test)
print("Accuracy Score :",accuracy_score(y_test,pred))
print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))
```

```
Accuracy Score : 0.9459815133949554
[[28436   235]
 [ 1489  1755]]
              precision    recall  f1-score   support

           0       0.95      0.99      0.97     28671
           1       0.88      0.54      0.67      3244

    accuracy                           0.95     31915
   macro avg       0.92      0.77      0.82     31915
weighted avg       0.94      0.95      0.94     31915
```

Xgboost Regressior

```
import xgboost
xgb = xgboost.XGBClassifier()
xgb.fit(x_train, y_train)
pred=xgb.predict(x_test)
print("Accuracy Score :",accuracy_score(y_test,pred))
print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))
```

```
Accuracy Score : 0.9543788187372709
[[28497   174]
 [ 1282  1962]]
              precision    recall  f1-score   support

           0       0.96      0.99      0.98     28671
           1       0.92      0.60      0.73      3244

    accuracy                           0.95     31915
   macro avg       0.94      0.80      0.85     31915
weighted avg       0.95      0.95      0.95     31915
```

```
test.drop(["id"],axis=1,inplace=True)
```

```
test=tf_vec.fit_transform(test['comment_text'])
test
```

```
<153164x10000 sparse matrix of type '<class 'numpy.float64'>'
	with 2940344 stored elements in Compressed Sparse Row format>
```

```
from sklearn.model_selection import cross_val_score
```

```
name=[df,rf,lm,gb,ad,xgb]
```

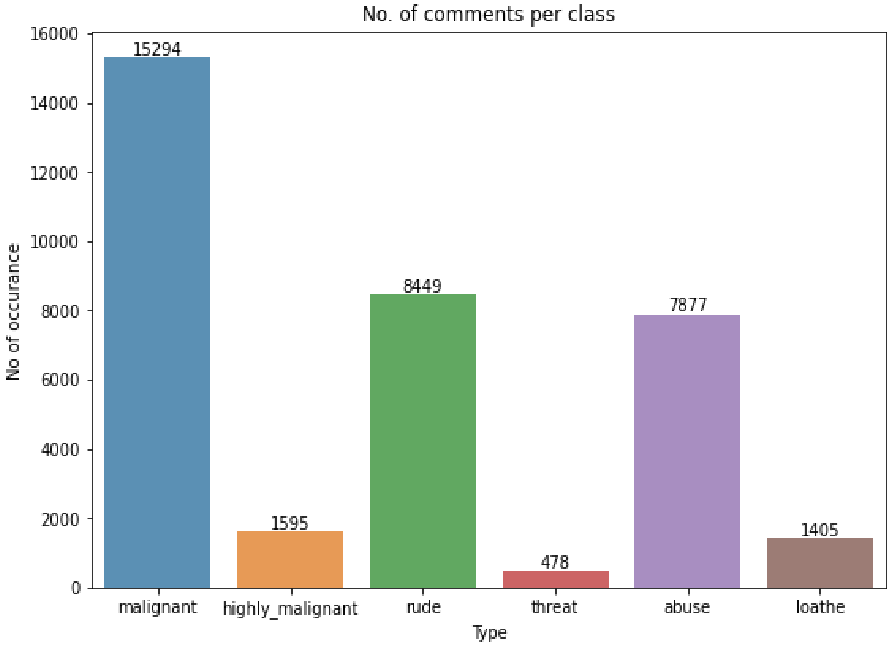
```
for i in name:
    cvs=cross_val_score(i,x,y,cv=9)
    print('Cross validation of',i,"is:",cvs.mean())
```

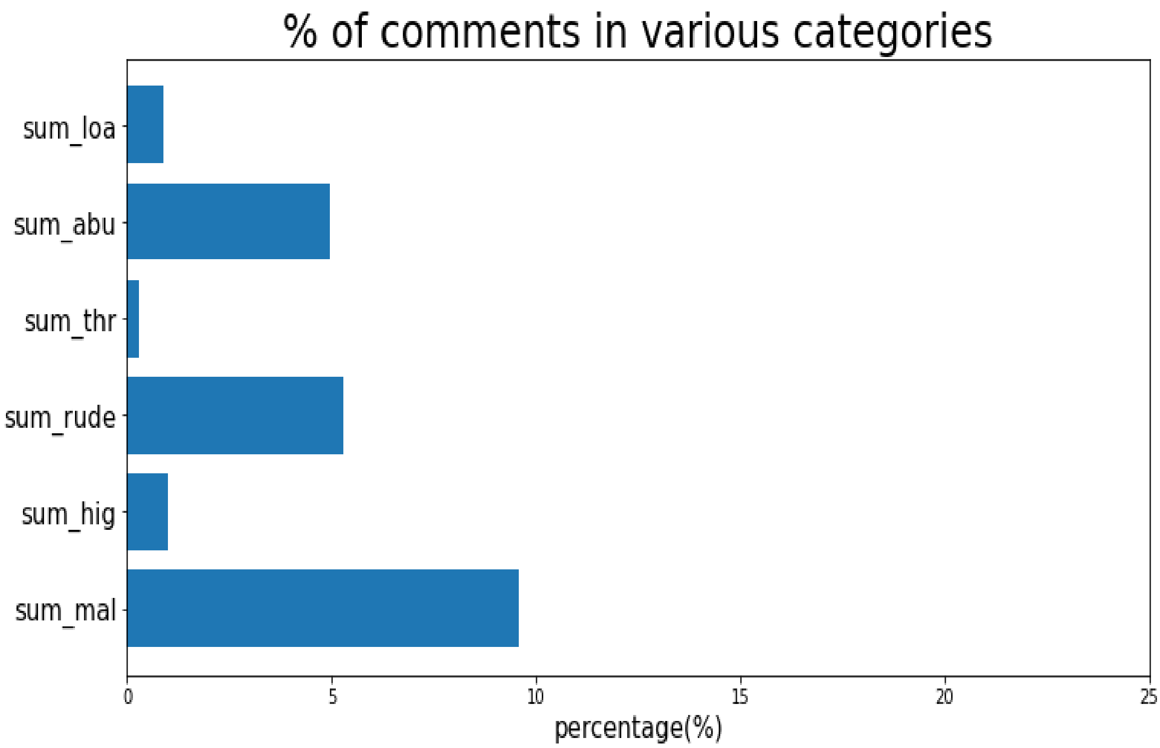
```
Cross validation of DecisionTreeClassifier() is: 0.9416497908853635
Cross validation of RandomForestClassifier() is: 0.9563454522877449
Cross validation of LogisticRegression() is: 0.956658778136424
Cross validation of GradientBoostingClassifier() is: 0.9404591032494918
Cross validation of AdaBoostClassifier() is: 0.9458234801811268
Cross validation of XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
              importance_type=None, interaction_constraints='',
              learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
              max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
              missing=nan, monotone_constraints='()', n_estimators=100,
              n_jobs=0, num_parallel_tree=1, predictor='auto', random_state=0,
              reg_alpha=0, reg_lambda=1, ...) is: 0.9539953924943801
```

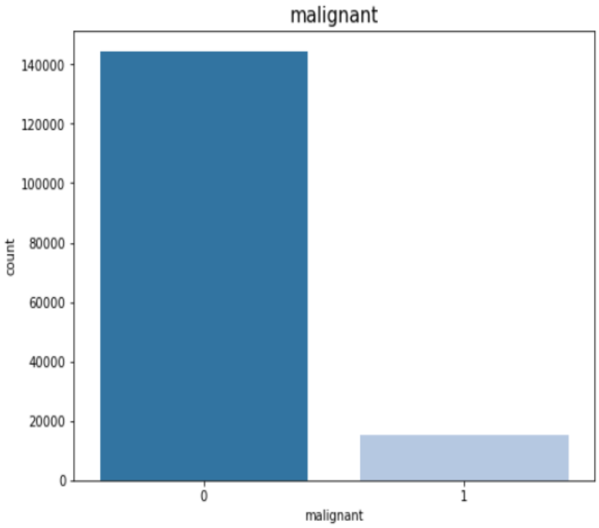
```
predict=rf.predict(test)
predict
```

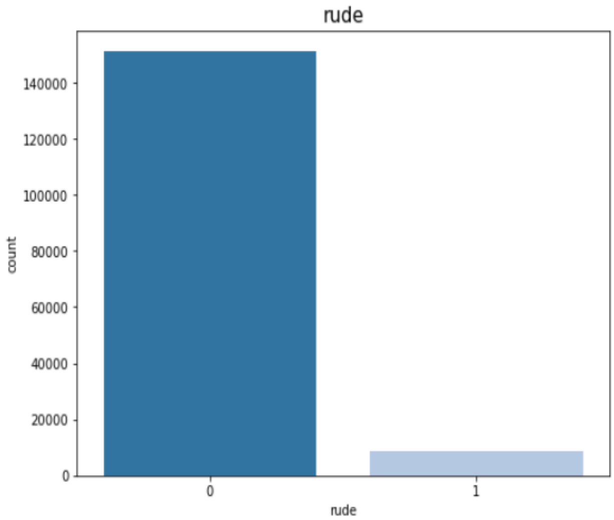
```
array([0, 0, 0, ..., 0, 1, 0])
```

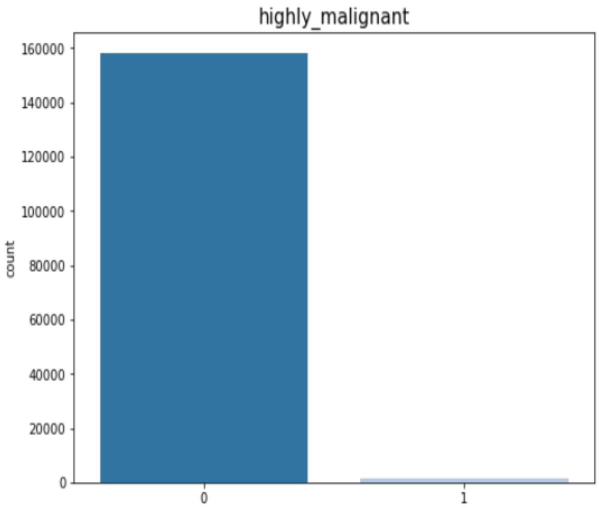
Visualization

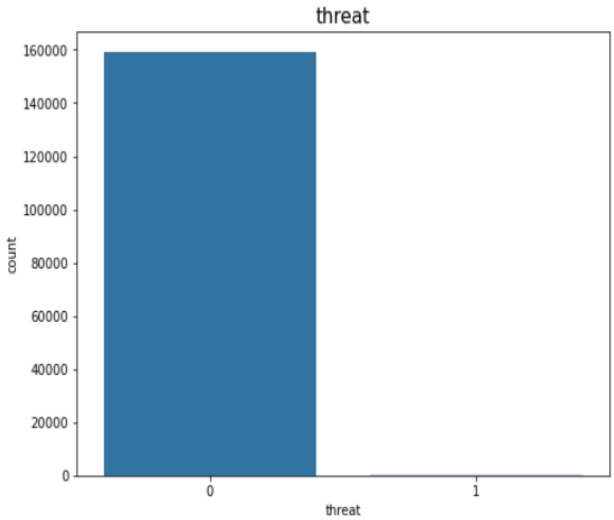


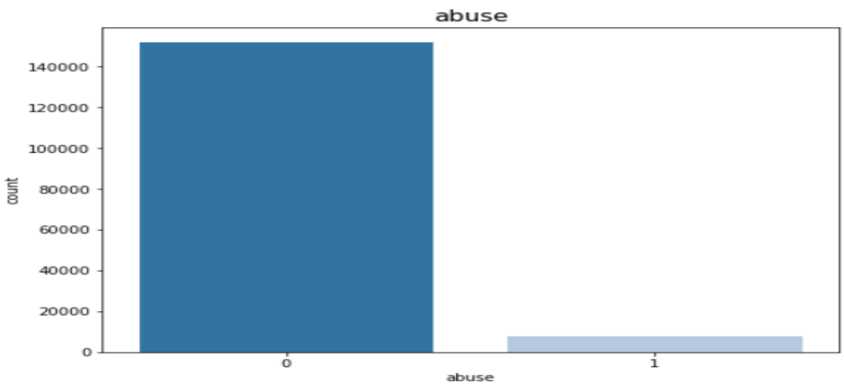


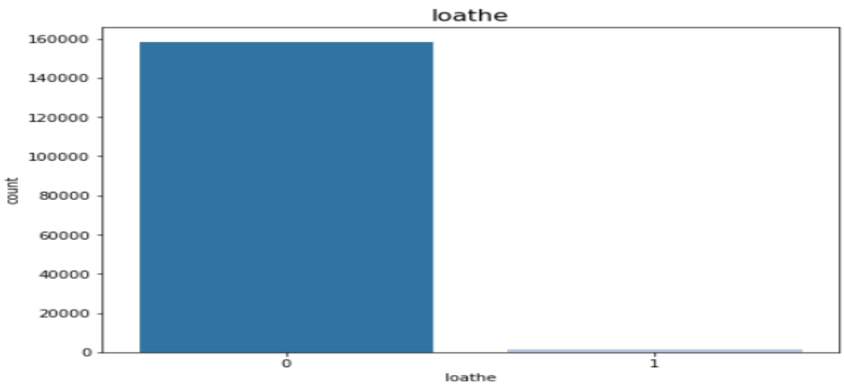

malignant
count
140000
120000
100000
80000
60000
40000
20000
0
0
1
malignant


rude
count
140000
120000
100000
80000
60000
40000
20000
0
0
1
rude


highly_malignant
count
160000
140000
120000
100000
80000
60000
40000
20000
0
0
1


threat
count
160000
140000
120000
100000
80000
60000
40000
20000
0
0
1
threat


abuse
count
140000
120000
100000
80000
60000
40000
20000
0
0
1
abuse


loathe
count
160000
140000
120000
100000
80000
60000
40000
20000
0
0
1
loathe

Finally we have selected the Random forest classifier model because it is giving the highest accuracy as compare to other models.

Accuracy score- 99.92 %

Cross validation score – 95.63 %