FLIP ROBO

A

Data science

Project

on

"Malignant Comments Classifier"

Submitted by:

Santosh Arvind Dharam

ACKNOWLEDGMENT

I feel great pleasure to present the Project entitled "Malignant Comments Classifier". But it would be unfair on our part if I do not acknowledge efforts of some of the people without the support of whom, this Project would not have been a success. First and for most I am very much thankful to my respected SME 'Swati Mahaseth' for his leading guidance in this Project. Also he has been persistent source of inspiration to me. I would like to express my sincere thanks and appreciation to 'flip robo' for their valuable support. Most importantly I would like to express our sincere gratitude towards my Friend & Family for always being there when I needed them most.

Mr. Santosh Arvind Dharam

INTRODUCTION

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, cyber bullying, 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 cyber bullying 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.

PROBLEM STATEMENT

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 un offensive, 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 cyber bullying.

Analytical Problem Framing

EDA steps:

1) import necessary libraries:

first we will import all the necessary libraries which will be usefull for analysis of data

```
In [1]: #import all libraries
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        import warnings
        warnings.filterwarnings('ignore')
        from sklearn.metrics import r2_score,mean_absolute_error,mean_squared_error
        from sklearn.linear_model import LogisticRegression,Lasso,LinearRegression
        from sklearn.neighbors import KNeighborsRegressor
        from sklearn.svm import SVR
        from sklearn.tree import DecisionTreeRegressor
        from sklearn.ensemble import AdaBoostRegressor,GradientBoostingRegressor
        from sklearn.preprocessing import LabelEncoder,StandardScaler
        from sklearn.model_selection import train_test_split,GridSearchCV
        from sklearn.decomposition import PCA
        from scipy.stats import zscore
        from sklearn.model_selection import cross_val_score
```

in this case we have to import all the necessary library that are useful for data analysis in jupyter notebook

2) Extract the dataset in jupyter notebook:

lets extract the train and test data from excel file

```
In [2]: train=pd.read_excel("C:\\Users\\SAI BABA\\Desktop\\train.xlsm")
        train.head()
```

Out[2]:

	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

```
In [3]: test=pd.read_excel("C:\\Users\\SAI BABA\\Desktop\\test.xlsm")
        test.head()
```

Out[3]:

	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.

```
In [4]: train.shape
Out[4]: (159571, 8)
```

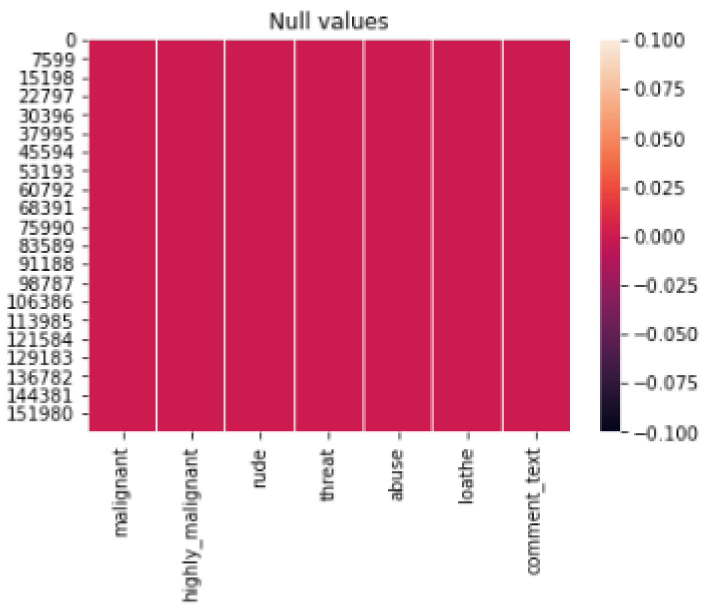
Data is extracted for further analysis in jupyter notebook data contains a 159571 rows and 8 columns some columns contains categorical data and some contains numerical .

3) checking null values:

In this case we have to find out the null values present in our data set. if it is there it is required to remove it. in our data set has no null values it is also shown by heat map

```
In [16]: train.isnull().sum()

Out[16]: malignant           0
         highly_malignant    0
         rude                0
         threat              0
         abuse               0
         loathe              0
         comment_text        0
         dtype: int64
```

```
In [17]: sns.heatmap(train.isnull())
         plt.title("Null values")
         plt.show()
```



4) Data information:

In [6]:
```
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
```

data contains 2 object type of columns and 6 int type of data

data contains 2 object type of columns and 6 integer type of dataset some categorical columns and some numerical data

5)Encoding the Dataset:

In [8]:
```
from sklearn.preprocessing import OrdinalEncoder
```

In [9]:
```
enc=OrdinalEncoder()
```

In [10]:
```
from sklearn.preprocessing import LabelEncoder
```

In [11]:
```
from nltk.stem import WordNetLemmatizer
import nltk
from nltk.corpus import  stopwords
import string
```

In [12]:
```
# Convert all messages to lower case
train['comment_text'] = train['comment_text'].str.lower()

# Replace email addresses with 'email'
train['comment_text'] = train['comment_text'].str.replace(r'^.+@[^\.].*\.[a-z]{2,}$',
                                 'emailaddress')

# Replace URLs with 'webaddress'
train['comment_text'] = train['comment_text'].str.replace(r'^http\://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(/\S*)?$',
                                  'webaddress')

# Replace money symbols with 'moneysymb' (£ can by typed with ALT key + 156)
train['comment_text'] = train['comment_text'].str.replace(r'£|\$', 'dollers')

# Replace 10 digit phone numbers (formats include paranthesis, spaces, no spaces, dashes) with 'phonenumber'
train['comment_text'] = train['comment_text'].str.replace(r'^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$',
                                  'phonenumber')


# Replace numbers with 'numbr'
train['comment_text'] = train['comment_text'].str.replace(r'\d+(\.\d+)?', 'numbr')
```

As the dataset contains some object type of columns and it is need to convert it for further analysis .so we have converted it from object to float or integer. some replacement in dataset is also required, we have did it .column comment text consist of object type of data it is converted to continuous data.

In [13]:
```
le=LabelEncoder()
label=le.fit_transform(train["comment_text"])
label
train=train.drop("comment_text",axis='columns')
train["comment_text"]=label
```

In [14]:
```
train
```

Out[14]:

	id	malignant	highly_malignant	rude	threat	abuse	loathe	comment_text
0	0000997932d777bf	0	0	0	0	0	0	74031
1	000103f0d9cfb60f	0	0	0	0	0	0	69080
2	000113f07ec002fd	0	0	0	0	0	0	81840
3	0001b41b1c6bb37e	0	0	0	0	0	0	35592
4	0001d958c54c6e35	0	0	0	0	0	0	157171
...	...	...	...	...	...	...	...	...
159566	ffe987279560d7ff	0	0	0	0	0	0	44139
159567	ffea4adeee384e90	0	0	0	0	0	0	156286
159568	ffee36eab5c267c9	0	0	0	0	0	0	129924
159569	fff125370e4aaaf3	0	0	0	0	0	0	59267
159570	fff46fc426af1f9a	0	0	0	0	0	0	31080

159571 rows × 8 columns

In [15]:
```
train.drop(['id'],axis=1,inplace=True)
```

We will also drop column id as shown above

6) Data Description:

Data consist of total 159571 rows

In [18]:
```
train.describe()
```

Out[18]:

	malignant	highly_malignant	rude	threat	abuse	loathe	comment_text
count	159571.000000	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	79688.965338
std	0.294379	0.099477	0.223931	0.054650	0.216627	0.093420	45990.152206
min	0.000000	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	39864.500000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	79705.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	119491.500000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	159159.000000

it gives total count,mean,std,mini to max range of each column which is shown in above table

It shows the total count of data along with mean and std deviation along with the mini and maximum values of data

7)Data Correlation:

If we want to found out the correlation between dataset that can be found by correlation.

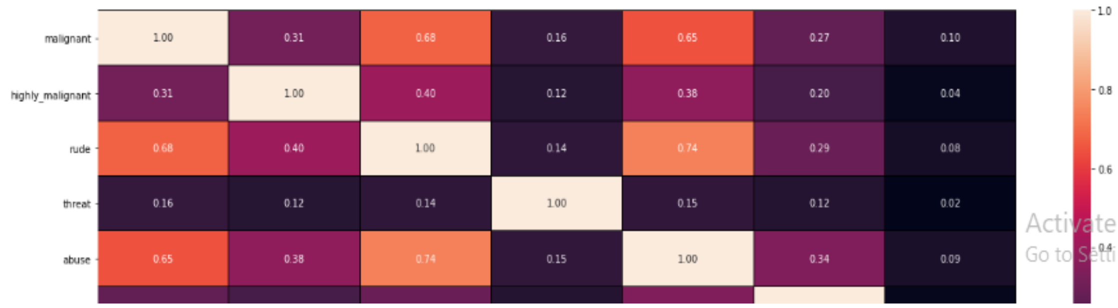
In [19]:
```
train.corr()
```

Out[19]:

	malignant	highly_malignant	rude	threat	abuse	loathe	comment_text
malignant	1.000000	0.308619	0.676515	0.157058	0.647518	0.266009	0.103724
highly_malignant	0.308619	1.000000	0.403014	0.123601	0.375807	0.201600	0.039877
rude	0.676515	0.403014	1.000000	0.141179	0.741272	0.286867	0.075431
threat	0.157058	0.123601	0.141179	1.000000	0.150022	0.115128	0.021269
abuse	0.647518	0.375807	0.741272	0.150022	1.000000	0.337736	0.087155
loathe	0.266009	0.201600	0.286867	0.115128	0.337736	1.000000	0.034670
comment_text	0.103724	0.039877	0.075431	0.021269	0.087155	0.034670	1.000000

In [20]:
```
#heat map
plt.figure(figsize=(22,7))
sns.heatmap(train.corr(),annot=True,linewidths=0.1,linecolor='black',fmt='0.2f')
```

Out[20]: <AxesSubplot:>



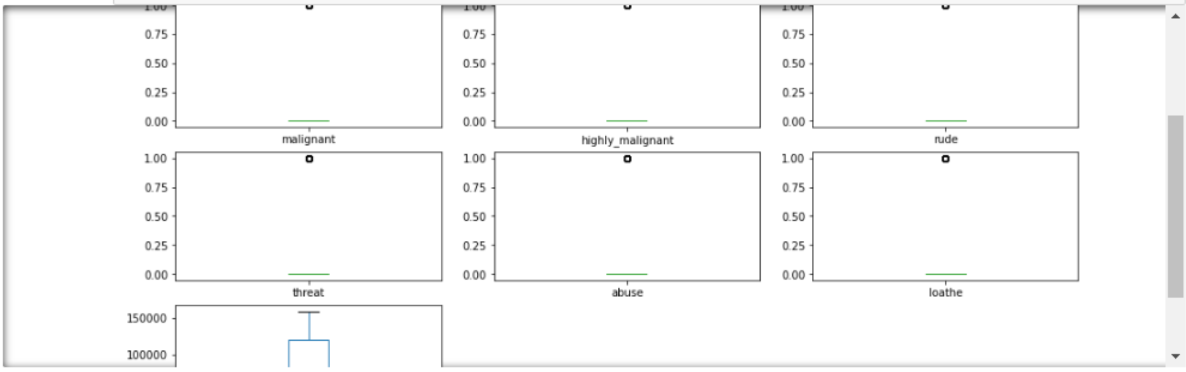
It shows that some data is having positive correlation and some data is having negative correlation .also it shows that % contribution of data.

8)visualization:

Lets we will see some visualization

Box Plot:

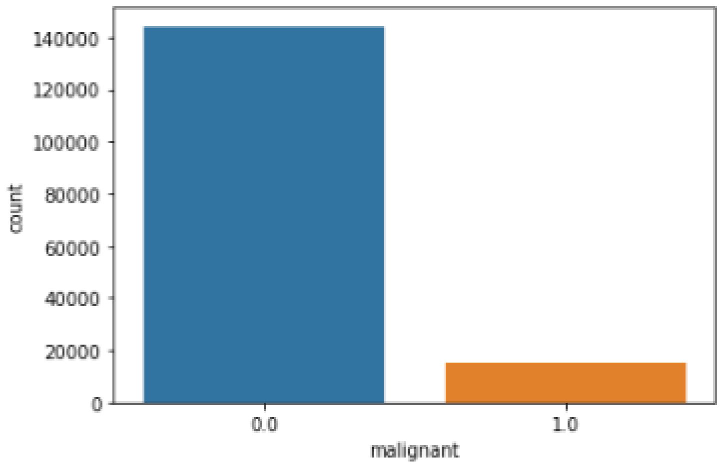
```
In [22]: #plotting the boxplot of each column to check the outliers
         train.plot(kind='box',subplots = True,layout=(4,3),figsize = (15,10))
```



It shows that whether outliers present in our dataset or not

```
In [31]: #detail insights of each column
         col=['malignant','highly_malignant','loathe','rude','abuse','threat']
         for i in col:
             print(i)
             print("\n")
             print(train[i].value_counts())
             sns.countplot(train[i])
             plt.show()
```

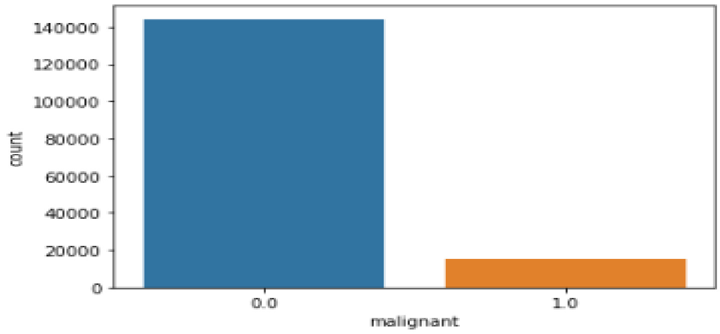
```
1.0     15294
Name: malignant, dtype: int64
```



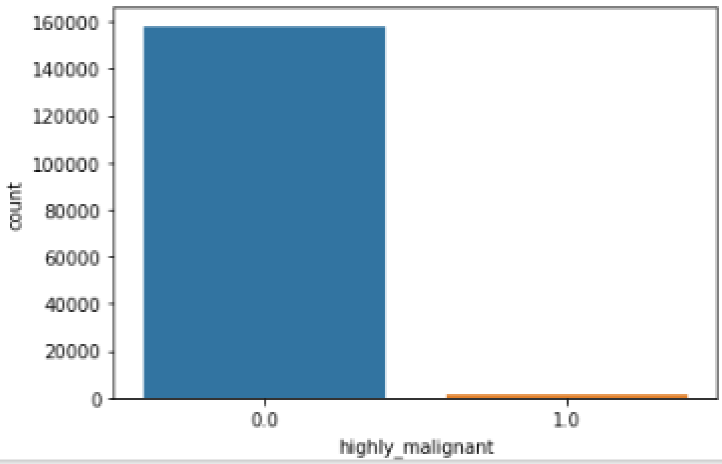
it gives the contribution of malignant non malignant

In [31]:
```
#detail insights of each column
col=['malignant','highly_malignant','loathe','rude','abuse','threat']
for i in col:
    print(i)
    print("\n")
    print(train[i].value_counts())
    sns.countplot(train[i])
    plt.show()
```

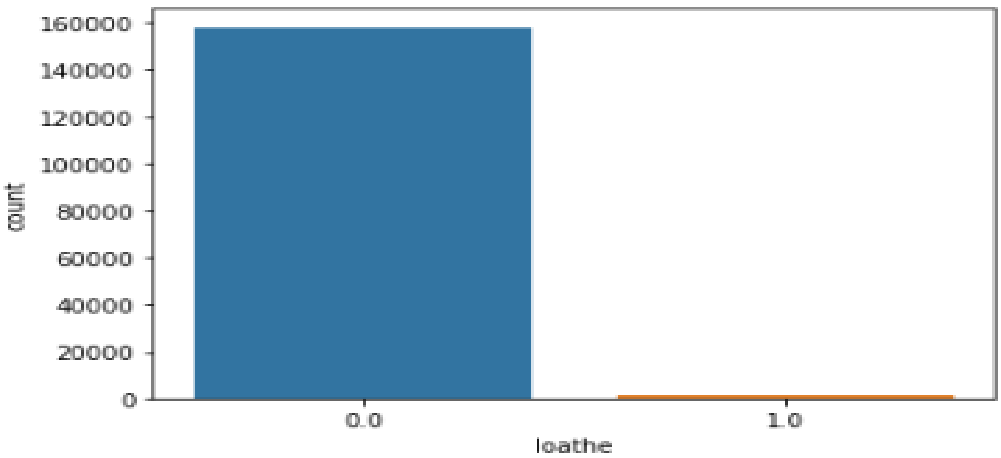
```
1.0      15294
Name: malignant, dtype: int64
```



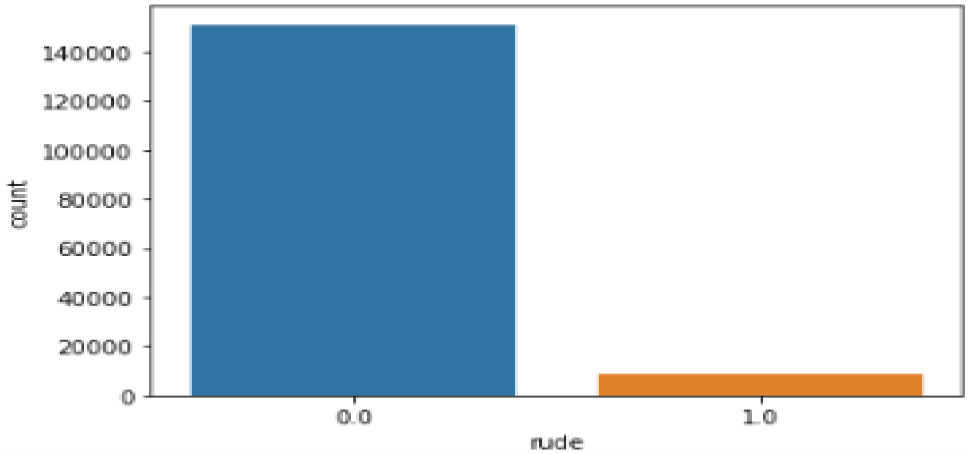

```
0.0     157976
1.0       1595
Name: highly_malignant, dtype: int64
```



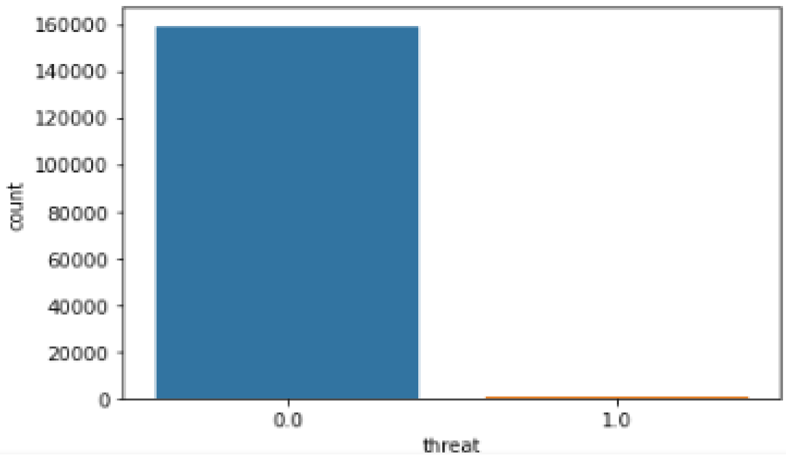

```
0.0     158166
1.0       1405
Name: loathe, dtype: int64
```



```
1.0       8449
Name: rude, dtype: int64
```

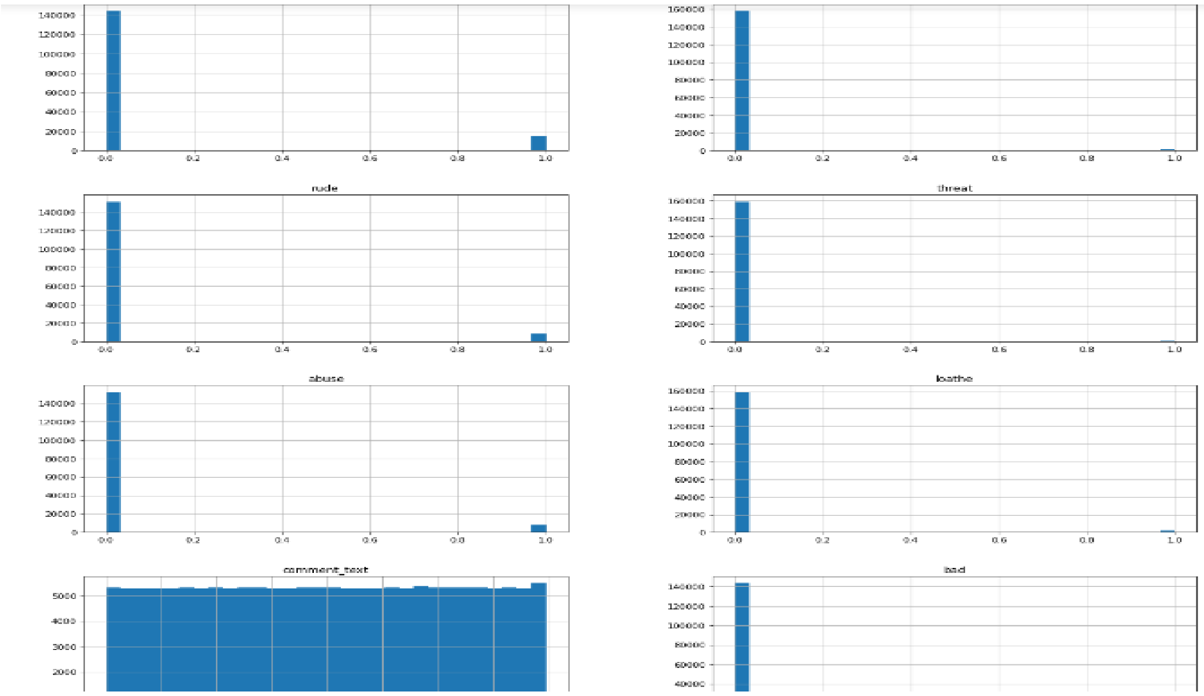



```
0.0    159093
1.0       478
Name: threat, dtype: int64
```



9) histogram:

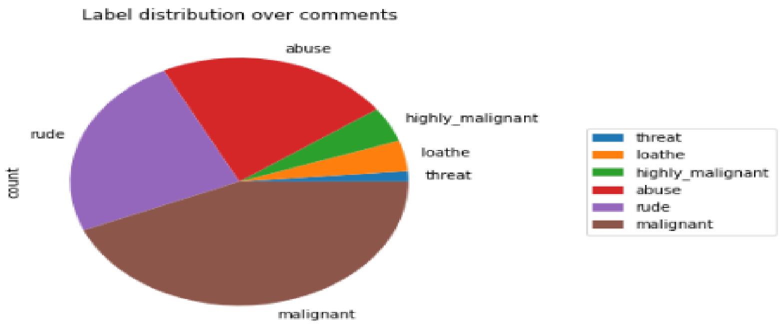
histogram for univariate analysis and checking the Normal Distribution



It shows that data is normally distributed in each column.

10)Distribution plot:

```
In [32]: #Label distribution over comments
         cols_target = ['malignant','highly_malignant','rude','threat','abuse','loathe']
         df_distribution = train[cols_target].sum()\
                                     .to_frame()\
                                     .rename(columns={0: 'count'})\
                                     .sort_values('count')

         df_distribution.plot.pie(y='count',
                                         title='Label distribution over comments',
                                         figsize=(5, 5))\
                                     .legend(loc='center left', bbox_to_anchor=(1.3, 0.5))

Out[32]: <matplotlib.legend.Legend at 0x2d30d56ad60>
```



it shows the label distribution over the comments for different column

11)distribution of data into x and y:

We have devided the data into x and y for further analysis

```
In [35]: #assign the value of x and y for training and testing phase
         x = train.drop(columns=['malignant'])
         y = train[["malignant"]]
         print(x.shape)
         print(y.shape)

         (159571, 7)
         (159571, 1)
```

```
In [36]: #Standardize the value of x so that mean will 0 and SD will become 1 , and make the data as normal distributed
         sc = StandardScaler()
         sc.fit_transform(x)
         x = pd.DataFrame(x,columns=x.columns)
```

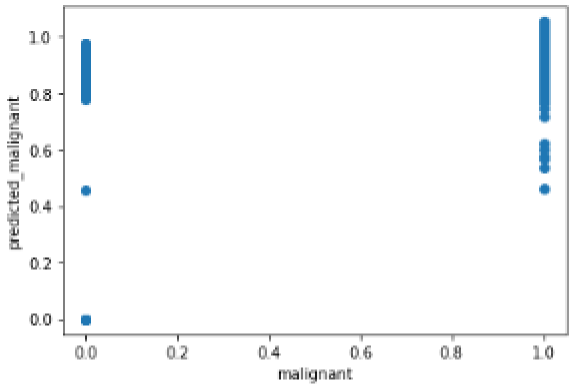
12) use of multiple algorithm:

```
model = [DecisionTreeRegressor(),KNeighborsRegressor(),AdaBoostRegressor(),LinearRegression(),GradientBoostir
max_r2_score = 0
for r_state in range(1,100):
    train_x,test_x,train_y,test_y = train_test_split(x,y,random_state = r_state,test_size = 0.24)
    for i in model:
        i.fit(train_x,train_y)
        pre = i.predict(test_x)
        r2_sc = r2_score(test_y,pre)
        print("R2 score correspond to random state " ,r_state ,"is", r2_sc)
        if r2_sc> max_r2_score:
            max_r2_score=r2_sc
            final_state = r_state
            final_model = i

print()
print()
print()
print()
print("max R2 score correspond to random state " ,final_state , "is" , max_r2_score ,"and model is",final_moc
```

```
R2 score correspond to random state  97 is 0.12392234187398021
R2 score correspond to random state  97 is 0.9378166133155523
R2 score correspond to random state  97 is 0.9370978619942096
R2 score correspond to random state  97 is 0.9427557221444546
R2 score correspond to random state  98 is 0.8912992539907431
R2 score correspond to random state  98 is 0.1255870433802012
R2 score correspond to random state  98 is 0.9370332124950239
R2 score correspond to random state  98 is 0.936707548864184
R2 score correspond to random state  98 is 0.9412595948390408
R2 score correspond to random state  99 is 0.8860178616659579
R2 score correspond to random state  99 is 0.10797274587409722
R2 score correspond to random state  99 is 0.9365242590001608
R2 score correspond to random state  99 is 0.936360197352472
R2 score correspond to random state  99 is 0.9412901148005866




max R2 score correspond to random state  70 is 0.9489085007545783 and model is GradientBoostingRegressor()
```

13) Scatter plot:

```
In [53]: #checking the diff between actual and predicted value using graph
         plt.scatter(x=test_y,y=pred)
         plt.xlabel('malignant')
         plt.ylabel('predicted_malignant')

Out[53]: Text(0, 0.5, 'predicted_malignant')
```



Graph shows the actual verses predicted data

14) Cross validation:

cross validation

```
In [57]: from sklearn.model_selection import cross_val_score

In [58]: for i in range(2,10):
             cv=cross_val_score(gbr,x,y,cv=i)
             print(gbr,cv.mean())

GradientBoostingRegressor() 0.9420471766662655
GradientBoostingRegressor() 0.9428206223948165
GradientBoostingRegressor() 0.9424446765968846
GradientBoostingRegressor() 0.9425840735830862
GradientBoostingRegressor() 0.9426183994283427
GradientBoostingRegressor() 0.9426492880953853
GradientBoostingRegressor() 0.9426805382721519
GradientBoostingRegressor() 0.9426630218819506
```

We have done the cross validation of model developed ,it also aacuracy upto 94.28%

15)Hyperparameter Tunning:

Lets we will do hyperparameter tunning by GridSearchCV

```
In [62]: from sklearn.model_selection import GridSearchCV
         from sklearn.model_selection import cross_val_score
         import warnings
         from sklearn.linear_model import Lasso
         warnings.filterwarnings('ignore')
```

```
In [63]: parameters={'alpha':[.0001,0.001,.01,.1,1,10],'random_state':list(range(0,30))}
         ls=Lasso()
         clf=GridSearchCV(ls,parameters)
         clf.fit(x_train,y_train)
         print(clf.best_params_)

         {'alpha': 0.0001, 'random_state': 0}
```

```
In [64]: from sklearn.metrics import r2_score
```

```
In [65]: ls=Lasso(alpha=0.0001,random_state=0)
         ls.fit(x_train,y_train)
         ls.score(x_train,y_train)
         pred_ls=ls.predict(x_test)
         lss=r2_score(y_test,pred_ls)
         lss

Out[65]: 0.9444120333281292
```

It also shows the good accuracy upto 94.44%

16) Saving Model:

Lets we will save the model

```
saving model

In [54]: import pickle

In [55]: #saving model to the local file system
         filename='malignant_comment_classifier.pickle'
         pickle.dump(gbr,open(filename,'wb'))

In [56]: filename

Out[56]: 'malignant_comment_classifier.pickle'
```

Successfully we have saved model by pickle. Dump

17)conclusion

```
In [61]: #Lets predict data
         y_pred=dt.predict(x_test)
         y_pred

Out[61]: array([0., 0., 0., ..., 0., 0., 0.])
```

We observed that data is cleaned now also some encoding is done now data is uniformly distributed in column. it is also observed from histogram, we have saved model and also predicted the result with help of saved model , model is ready for the future data prediction.