MALIGNANT COMMENTS CLASSIFICATION

Submitted by:

SHUBHAM MALIK

ACKNOWLEDGMENT

I would like to thank flip robo for providing me opportunity to work on this project. And would also thanks to my SME for helping me out in the completion of this project. Here I have taken help from the two research papers and from net surfing by help of these I got benefit to understand the problem and data we received from flip robo technology, and I have made my effort to solve this problem statement.

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 behavior.
- 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.

- Conceptual Background of the Domain Problem

In the past few years its seen that the cases related to social media hatred have increased exponentially. The social media is turning into a dark venomous pit for people now a days. Online hate is the result of difference in opinion, race, religion, occupation, nationality etc.

• In social media the people spreading or involved in such kind of activities uses filthy languages, aggression, images etc. to offend and gravely hurt the person on the other side. This is one of the major concerns now.
• The result of such activities can be dangerous. It gives mental trauma to the victims making their lives miserable. People who are not aware of mental health online hate or cyber bullying become life threatening for them. Such cases are also at rise. It is also taking its toll on religions. Each day we can see an incident of fighting between people of different communities or religions due to offensive social media posts.
• Online hate, described as abusive language, aggression, cyberbullying, hatefulness, insults, personal attacks, provocation, racism, sexism, threats, or toxicity has been identified as a major threat on online social media platforms. These kinds of activities must be checked for a better future.

- Review of Literature

There has been a remarkable increase in the cases of cyberbullying and trolls on various social media platforms. Because now a day people are focusing on the social media from which they get to now every which are going around in the world. Every people in the world have their own opinion towards a specific task like here we take an example of political party as I have worked on scraping data of YouTube of a political news where I came across to many comments which are hateful and offensive which can affect any one mentally and leads to depression, mental illness, self-hatred and fights among people so in this project we are working to identify the comments.

- Motivation for the Problem Undertaken

This project was provided by FlipRobo as a part of my internship program the main objective behind solving this real time problem is to put forward my skill to solve these problem. However, the motivation for taking this project was that it is relatively a new field of research. Here we have many options but less concrete solutions. The main motivation 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.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

We have calculated the mean, standard deviation with the help of the describe function. The describe function applies basic statistical computations on the dataset like extreme values, count of data points standard deviation etc. we get details of all the columns by the help of this we can find the outlier present in the data by seeing the mean and 50% value of the given values. We can fill null values in the data set so that they do not make any problem in building the model. With the help of this we get proper justification of the data set we have on which we are working. Here we are dealing with one main text columns which held some importance of the data and others shows the multiple types of behavior inferred from the text. I prefer to select on focus more on the words which has great value of importance in the context. This converts the important words proper vectors with some weights.

- Data Sources and their formats

After loading the training dataset into Jupiter Notebook using Pandas and there are eight columns named as: “id, comment text, “malignant, highly malignant, rude, threat, abuse, loathe”. There are 8 columns in the dataset provided:
The description of each of the column is given below:

- 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 pla

```
[ ]: # To fetching the datatypes of the columns
traindata.dtypes
```

```
Out[8]: id                  object
comment_text        object
malignant            int64
highly_malignant     int64
rude                 int64
threat               int64
abuse                int64
loathe               int64
dtype: object
```

a) Data Preprocessing Done

Data pre-processing means that making data correct by removing the unwanted data which help us to achieve highly accurate results. If the quality of data is good, then output of the result is also good. Hence if quality increase then the result of the model automatically increases. Some of the factors which can affect the data are :

i Incomplete

ii. Noisy

iii. Inconsistent data etc.

a) Incomplete data – It can occur due to many reasons. Due to some misunderstanding the data cannot be correct. And other way the machine may not work proper due to some error.

b) Noisy data – noisy data can be described as a faulty data which contain some error in it. It may be by human or by machine defects other way can be transferring of data.

c) Inconsistent data – inconsistent data can be defined as we take example of the bank; we made a transaction the amount of money is deducted from our

account but not added to the other account that means there is some error in the database or the dataset.

There are many stages involved in data preprocessing:
1. Data cleaning
2. Data integration
3. Data transformation
4. Data reduction etc.

There are many steps which are performed for the data cleaning in this model building. They are as follows
Checking the null values in the data set.
Checking for the outliers we find outlier in 12 columns we applied z-score on these columns to remove the outliers because outlier harm the result of our model which we have built. Then we check for the skewness of the data some of the columns are skewed then we removed skewness with the help of power transform. Then moving towards the model building.

b) Data Inputs- Logic- Output Relationships

After loading all the required libraries, we loaded the data into our jupyter notebook. Then we uploaded our dataset

```
# fetching data from computer
import pandas as pd
traindata=pd.read_csv(r"C:\Users\malik\Downloads\Malignant-Comments-Classifier-Project--1-\Malignant Comments Classifier Proj
traindata.head()
```

t[3]:

	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

```
# fetching data from computer
import pandas as pd
testdata=pd.read_csv(r"C:\Users\malik\Downloads\Malignant-Comments-Classifier-Project--1-\Malignant Comments Classifier Proje
testdata.head()
```

t[4]:

	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.

Then we have checked the number of abuse, threat, malignant, high malignant etc.

```
]: cols=['malignant', 'highly_malignant', 'rude', 'threat','abuse', 'loathe',]
for col in cols:
    print("Number of value_counts of {} : {}".format(col, traindata[col].nunique()))
    print(traindata[f'{col}'].value_counts())
```

```
Number of value_counts of malignant : 2
0    144277
1     15294
Name: malignant, dtype: int64
Number of value_counts of highly_malignant : 2
0    157976
1      1595
Name: highly_malignant, dtype: int64
Number of value_counts of rude : 2
0    151122
1      8449
Name: rude, dtype: int64
Number of value_counts of threat : 2
0    159093
1       478
Name: threat, dtype: int64
Number of value_counts of abuse : 2
0    151694
1      7877
Name: abuse, dtype: int64
Number of value_counts of loathe : 2
0    158166
1      1405
Name: loathe, dtype: int64
```

After that we have counted the various parameters to understand them like:

Firstly we checked for the null values present in our data set and we find out that there were no null values in our data set and after that we checked for the data type of our columns because different data types do not work in model building so we convert them here we have two columns in object format.

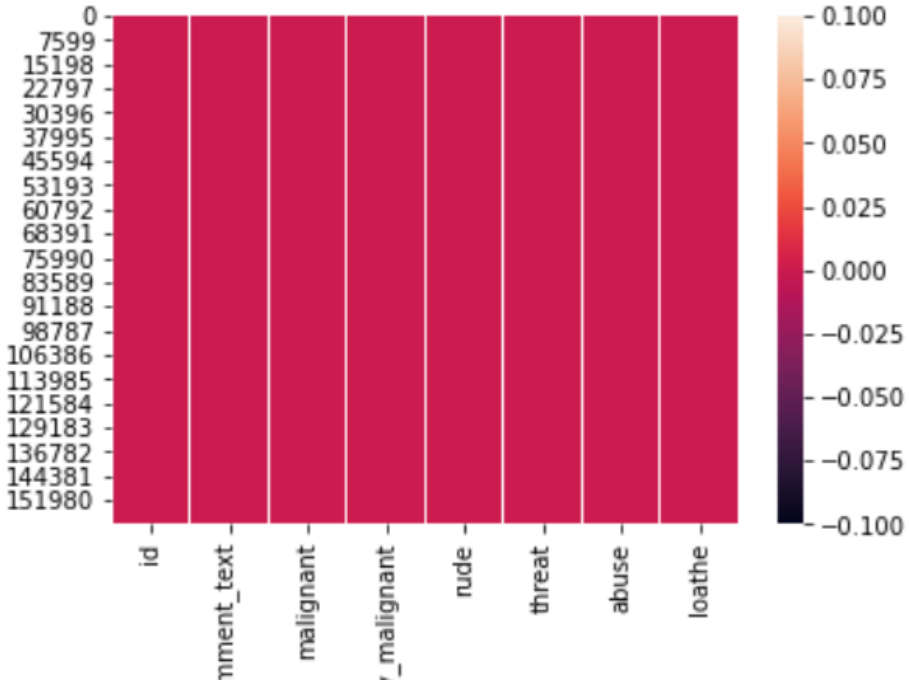
```
traindata.isna().sum()
```

[11]:
```
id                  0
comment_text        0
malignant           0
highly_malignant    0
rude                0
threat              0
abuse               0
loathe              0
dtype: int64
```

```
# To fetching the datatypes of the columns
traindata.dtypes
```

[9]:
```
id                  object
comment_text        object
malignant            int64
highly_malignant     int64
rude                 int64
threat               int64
abuse                int64
loathe               int64
dtype: object
```

```
#checking null values using heatmap
sns.heatmap(traindata.isnull());
```



Then we have dropped the id column which is not useful to us in model building

```
# Dropping column 'id' since it's of no use
traindata.drop(['id'],axis=1,inplace=True)
```

Then we have counted the number of unique values in the data set we are working on

```
#Checking unique values in each column
for i in features:
    print('Number of unique values in {} : {}'.format(i, traindata[i].nunique()))
```

```
Number of unique values in comment_text : 159571
Number of unique values in malignant : 2
Number of unique values in highly_malignant : 2
Number of unique values in rude : 2
Number of unique values in threat : 2
Number of unique values in abuse : 2
Number of unique values in loathe : 2
```

c) Data Preprocessing Done

After loading all the required libraries, we loaded the data into our Jupiter notebook.

```
#Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import zscore
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings('ignore')
```

For Data pre-processing we did some data cleaning, where we used wordNetlemmatizerto clean the words and removed special characters using Regexp Tokenizer and filter the words by removing stop words and then used lemmatizes and joined and return the filtered words. Used TFIDF vectorizer to convert those text into vectors and split the data and into test and train and trained various Machine learning algorithms.

```
#Creating a function to filter using POS tagging.

def get_pos(pos_tag):
    if pos_tag.startswith('J'):
        return wordnet.ADJ
    elif pos_tag.startswith('N'):
        return wordnet.NOUN
    elif pos_tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN
```

```
def Processed_data(comments):
    # Replace email addresses with 'email'
    comments=re.sub(r'^.+@[^\.].*\.[a-z]{2,}$',' ', comments)

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

    # getting only words(i.e removing all the special characters)
    comments = re.sub(r'[^\w]', ' ', comments)

    # getting only words(i.e removing all the" _ ")
    comments = re.sub(r'[\_]', ' ', comments)

    # getting rid of unwanted characters(i.e remove all the single characters left)
    comments=re.sub(r'\s+[a-zA-Z]\s+', ' ', comments)

    # Removing extra whitespaces
    comments=re.sub(r'\s+', ' ', comments, flags=re.I)

    #converting all the letters of the review into lowercase
    comments = comments.lower()

    # splitting every words from the sentences
    comments = comments.split()

    # iterating through each words and checking if they are stopwords or not,
    comments=[word for word in comments if not word in set(STOPWORDS)]

    # remove empty tokens
    comments = [text for text in comments if len(text) > 0]

    # getting pos tag text
    pos_tags = pos_tag(comments)

    # considering words having length more than 3only
    comments = [text for text in comments if len(text) > 3]

    # performing lemmatization operation and passing the word in get_pos function to get filtered using POS ...
    comments = [(WordNetLemmatizer().lemmatize(text[0], get_pos(text[1])))for text in pos_tags]

   # considering words having length more than 3 only
    comments = [text for text in comments if len(text) > 3]
    comments = ' '.join(comments)
    return comments
```

```
traindata.head(5)
```

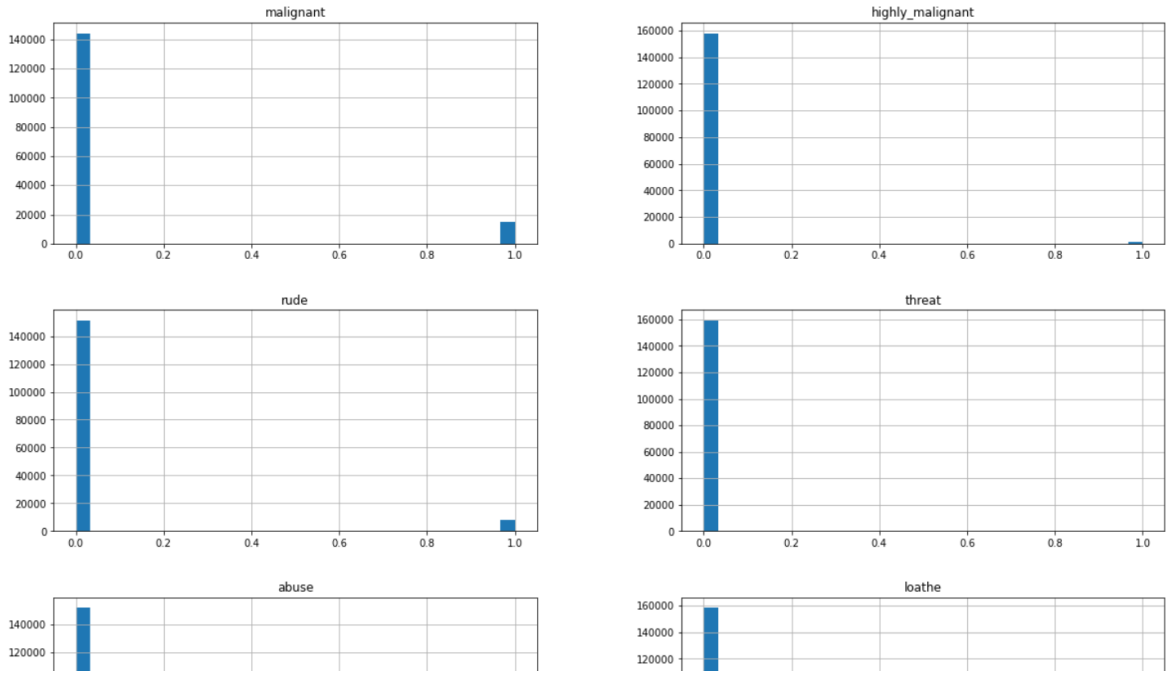
8]:

	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	comment_length	label
0	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0	264	0
1	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0	112	0
2	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0	233	0
3	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0	622	0
4	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0	67	0

Visualization performed

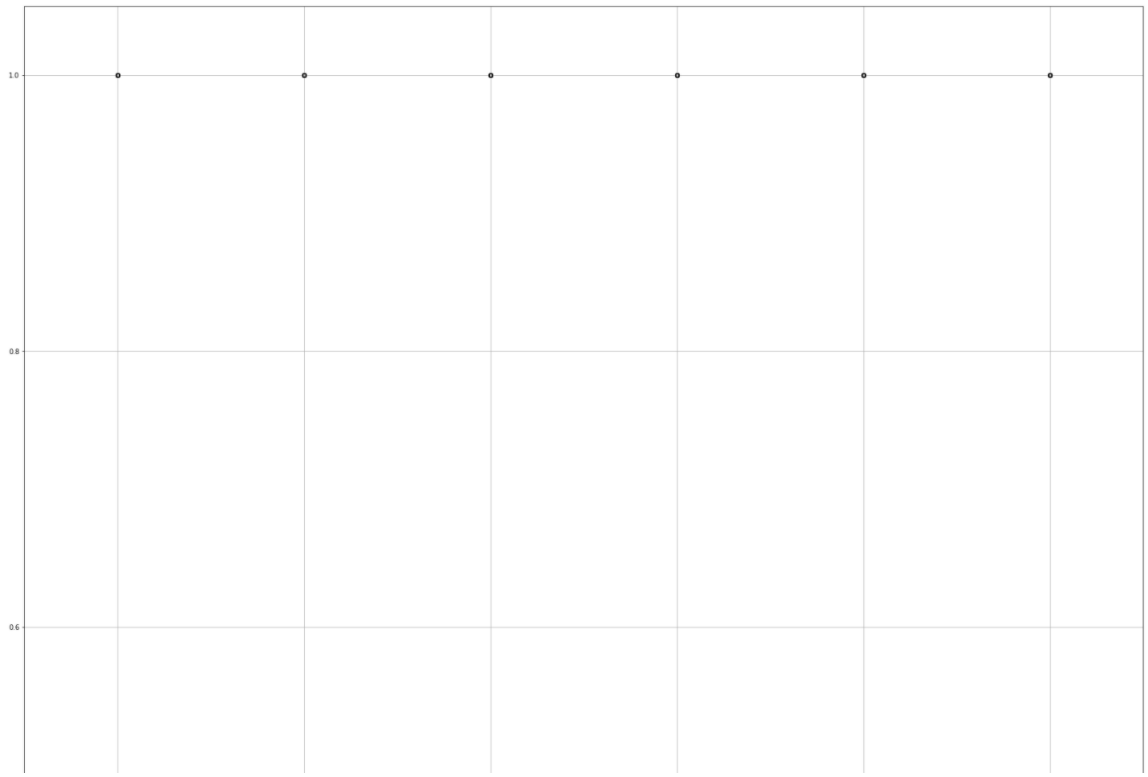
```
traindata.hist(figsize=(20,20),grid=True,layout=(4,2),bins=30)
```

```
16]: array([[<AxesSubplot:title={'center':'malignant'}>,
        <AxesSubplot:title={'center':'highly_malignant'}>],
       [<AxesSubplot:title={'center':'rude'}>,
        <AxesSubplot:title={'center':'threat'}>],
       [<AxesSubplot:title={'center':'abuse'}>,
        <AxesSubplot:title={'center':'loathe'}>],
       [<AxesSubplot:>, <AxesSubplot:>]], dtype=object)
```



Checking for outliers

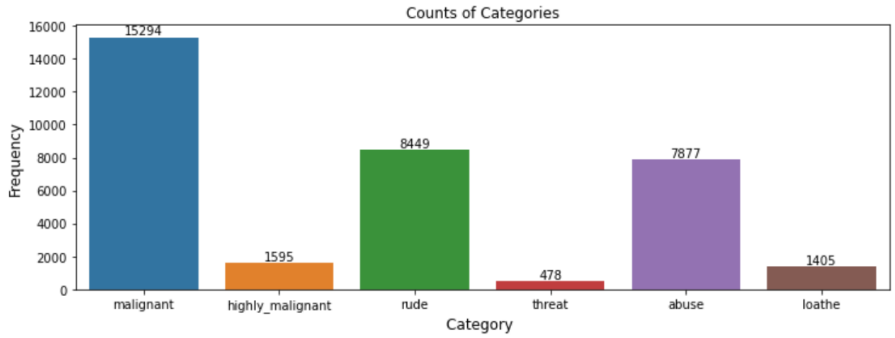
```
# plotting boxplot to identify outliers
traindata.boxplot(figsize=[30,50])
plt.subplots_adjust(bottom=0.25)
plt.show()
```



DATA INPUTS- LOGIC- OUTPUT RELATIONSHIPS

EDA was performed by creating valuable insights using various visualization libraries.

```
# Let's plot the counts of each category

plt.figure(figsize=(12,4))
ax = sns.barplot(counts.index, counts.values)
plt.title("Counts of Categories")
plt.ylabel('Frequency', fontsize=12)
plt.xlabel('Category ', fontsize=12)
rects = ax.patches
labels = counts.values
for rect, label in zip(rects, labels):
    height = rect.get_height()
    ax.text(rect.get_x() + rect.get_width()/2, height + 5, label, ha='center', va='bottom')
plt.show()
```



Checking for correlation

```
# Let's plot the correlation chart

traindata.corr().style.background_gradient(cmap='YlGnBu')
```

:4]:

	malignant	highly_malignant	rude	threat	abuse	loathe
malignant	1.000000	0.308619	0.676515	0.157058	0.647518	0.266009
highly_malignant	0.308619	1.000000	0.403014	0.123601	0.375807	0.201600
rude	0.676515	0.403014	1.000000	0.141179	0.741272	0.286867
threat	0.157058	0.123601	0.141179	1.000000	0.150022	0.115128
abuse	0.647518	0.375807	0.741272	0.150022	1.000000	0.337736
loathe	0.266009	0.201600	0.286867	0.115128	0.337736	1.000000

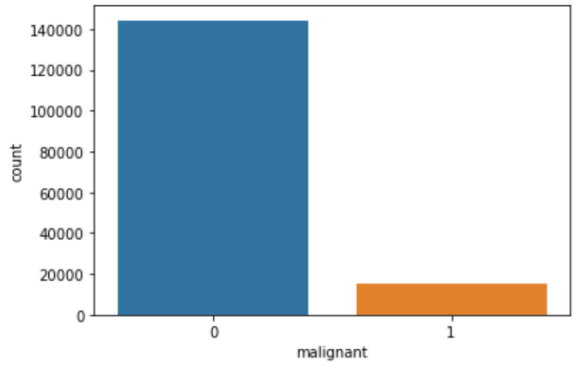
Plotting heatmap

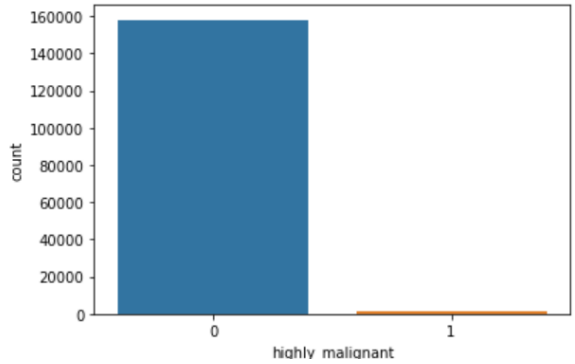
```
# Let's view the Correlation heatmap among variables

plt.figure(figsize=(8,6))
sns.heatmap(traindata.corr())
```

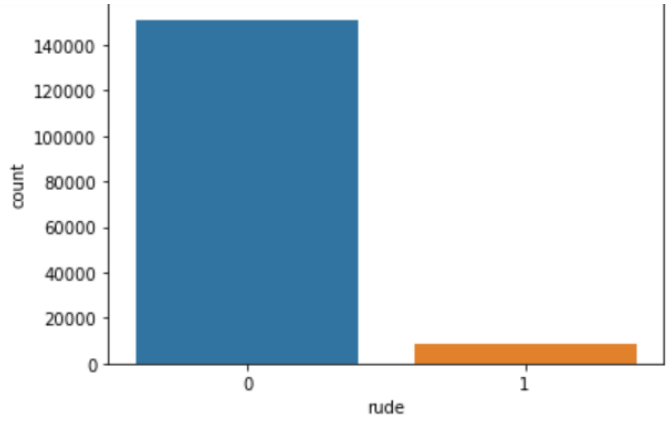
25]: <AxesSubplot:>

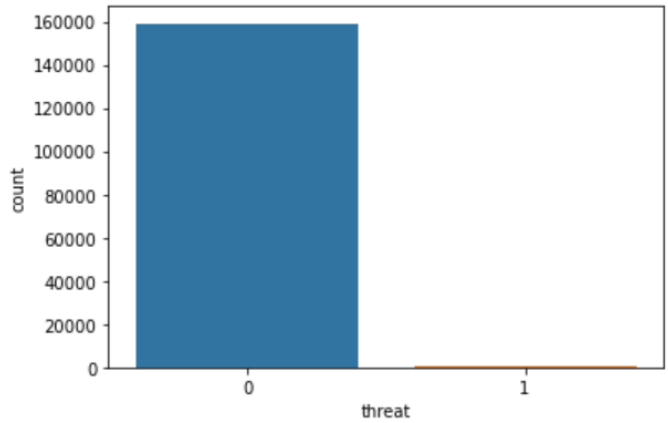
Counting values for each features that it comes under 0 or 1

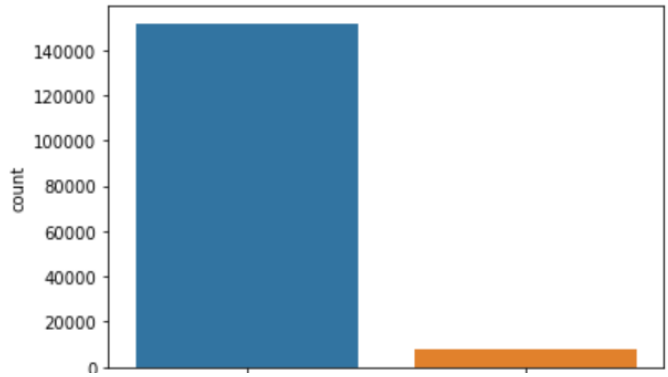
```
for i in features:
    sns.countplot(traindata[i])
    plt.show()
```











- Hardware and Software Requirements and Tools Used

Here software used is Jupiter notebook in which we build the model in python. This is a built-in platform we must start by importing some of the library which are used in the model building. Some are as follows:

i. Importing NumPy-this is a numerical python in which numeric calculation are done.

ii. Importing pandas- pandas are used to perform different operation from uploading or retrieving the file to make changes in file.

iii. Matplotlib- these are used for the visualization technique to plot the histogram, boxplot etc.

iv. Seaborn – these are used for the visualization technique by the help of which we plot heat map.

There are more libraries like

i. Classification report
ii. Confusion matrix
iii. Standard scalar
iv. Accuracy score

Model/s Development and Evaluation

- For this we have used different method for solving this problem for the analysis purpose we use visualization that is boxplot, seaborn and matplotlib to identify the pattern of the data. We find that data is starting from zero which is good then we applied boxplot for outliers' detection we find that 3 columns are having the outliers. Outliers are not good for the model building. Then with the help of z-score we remove the outliers. By the help of the visualization, we find the features related to price. Then after we find the correlation of the data each column correlation with itself. we find that some of them are highly corelated. After that we started building the model, we split data set in two form label and features using train test split.
- Testing of Identified Approaches (Algorithms)

  List of algorithms used are:

  I. Logistic Regression
  II. Decision Tree classifier

III. Random Forest classifier
IV. K Neighbors classifier

- Run and evaluate selected models

Now we need to separate the train dataset into X and y values. We have converted the train data set in the form of array and split it in the form of x and y

```
: # Splitting the training dataset into x and y
  x=Tf_idf_train(traindata['comment_text'])
  x.shape
```

```
[44]: (159571, 52750)
```

```
: y = traindata['label'].values
  y.shape
```

```
[45]: (159571,)
```

Let's split the master dataset into training and test set with an 75%-25% ratio. it is a function in sklearn model selection for splitting data arrays into two subsets for training data and testing data. With this function, you don't need to divide the dataset manually. By default, sklearn train_test_split will make random partitions for the two subsets. However, you can also specify a random state for the operation. It gives four outputs x train, x test, y train and y test. The x train and x test contain the training and testing predictor variables while y train and y test contains the training and testing target variable

```
#splitting the data into training and testing
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30)
```

a) Logistic regression

Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is binary, which means there would be only two possible classes

```
#agorithm use to predict the model
from sklearn.linear_model import LogisticRegression

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

```
47]: LogisticRegression()
```

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

```
accuracy=accuracy_score(y_test,y_pred)
accuracy
```

```
49]: 0.9547125668449198
```

b) Random forest classifier

As we know that a forest is made up of trees and more trees means more robust forest. Similarly, a random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting. It is an ensemble method which is better than a single decision tree because it reduces the over-fitting by averaging the result.

```
#agorithm use to predict the model
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf.fit(x_train,y_train)
predrf = rf.predict(x_test)
```

```
print(accuracy_score(y_test,predrf))
```

```
0.9440173796791443
```

```
print(confusion_matrix(y_test,predrf))
```

```
[[42925    81]
 [ 2599  2267]]
```

```
print(classification_report(y_test,predrf))
```

```
              precision    recall  f1-score   support

           0       0.94      1.00      0.97     43006
           1       0.97      0.47      0.63      4866

    accuracy                           0.94     47872
   macro avg       0.95      0.73      0.80     47872
weighted avg       0.95      0.94      0.94     47872
```

c) Decision tree classifier algorithm

Decision trees can be constructed by an algorithmic approach that can split the dataset in different ways based on different conditions. The two main entities of a tree are decision nodes, as parent node where the data is split and leaves or child node, where we get the outcome

```
#using dtc algoritm
from sklearn.tree import DecisionTreeClassifier
clf=DecisionTreeClassifier()
clf.fit(x_train,y_train)
```

```
]: DecisionTreeClassifier()
```

```
clf.score(x_train,y_train)
```

```
]: 0.9997582789460963
```

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

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

```
]: 0.9376044451871658
```

d) K Neighbors classifier

Out of all the machine learning algorithms I have come across, KNN algorithm has easily been the simplest to pick up. Despite its simplicity, it has proven to be incredibly effective at certain tasks (as you will see in this article).And even better? It can be used for both classification and regression problems! KNN algorithm is by far more popularly used for classification problems, however. I have seldom seen KNN being implemented on any regression task.

```
#k nearest neighbour algorithm
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(
n_neighbors = 22,
metric = 'minkowski', p = 2)
classifier.fit(x_train, y_train)
```

```
]: KNeighborsClassifier(n_neighbors=22)
```

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

```
acc = accuracy_score(y_test, y_pred )
```

```
print(accuracy_score(y_test,predrf))
```

```
0.9440173796791443
```

e) Hyperparametric tunning

Model improvement can be done by choosing the different methods of machine learning that we come up with. There are two methods that we have used in this model -first type is the model learning by the help of training and after that predicting the result and the second method is hyperparameters it a random model that cannot be determined by the data we tried to optimize tuning techniques using gridsearchCV using lasso model.

Hyperparameter tuning might not improve the model every time. For instance, when we tried to tune the model further which is shown below in the figure.

```
]:  #implementing lasso regression
    from sklearn.model_selection import GridSearchCV
    from sklearn.linear_model import Lasso
    lasso = Lasso()
```

```
]:  parameters = {"alpha":[16, 15, 33, 41, 12, 22, 16, 51, 14, 21]}
    lasso_regression = GridSearchCV(lasso, parameters, scoring='neg_mean_squared_error', cv=5)
    lasso_regression.fit(x_test, y_test)
```

```
:[65]: GridSearchCV(cv=5, estimator=Lasso(),
                    param_grid={'alpha': [16, 15, 33, 41, 12, 22, 16, 51, 14, 21]},
                    scoring='neg_mean_squared_error')
```

```
]:  print(lasso_regression.best_params_)
```

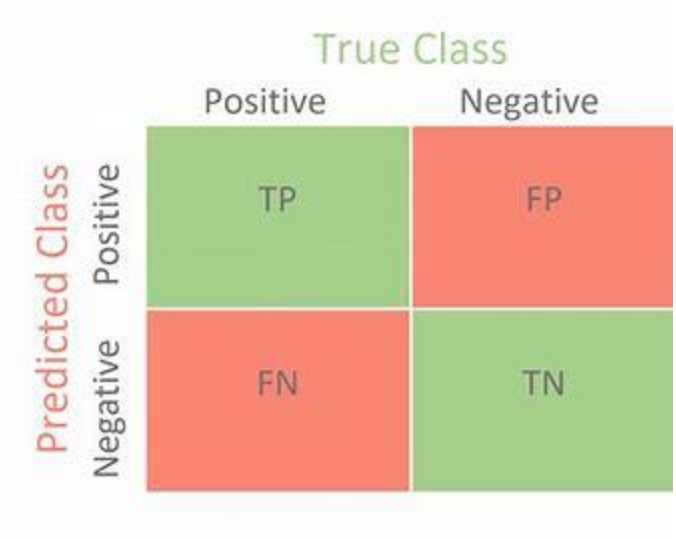
```
{'alpha': 16}
```

```
]:  print(lasso_regression.best_score_)
```

```
-0.0913507465373949
```

CONCLUDING REMARKS:

After applying the machine learning algorithms, we concluded that it provides the accuracy of the model is 95% without losing any data. The best accuracy is given by the random forest classifier algorithm in this algorithm it identifies the decision tree because forest contain n number of trees according to it check for each iteration and provide the best result to us. The main aim of this project was to detect the auto insurance fraud. In industry insurance fraud is a big problem. It's difficult to identify the fraud claim. Machine Learning is in a unique position to help the Auto Insurance industry with this problem. Here our model predicts 42932 true positive cases out of 43004positive cases and 2353true negative cases out of 4868 cases. It predicts 72 false positive cases out of 43004positive cases and 2515 false negative cases out of 4868cases. It gives the f1 score of 95%.



TN/True Negative: the cases were negative and predicted negative.

TP/True Positive: the cases were positive and predicted positive.

FN/False Negative: the cases were positive but predicted negative.

TN/True Negative: the cases were negative but predicted positive.

Finally saving the model