MALIGNANT COMMENT CLASSIFIER PROJECT

Prepared by
Sonali Daga

Data Science Intern at Flip Robo Technologies

SME Name:

SWATI MAHASETH

Acknowledgement

It is my deepest pleasure and gratification to present this report. Working on this project was an incredible experience that has given me a very informative knowledge regarding the data analysis process.

All the required information and dataset are provided by **Flip Robo Technologies** (Bangalore) that helped me to complete the project.

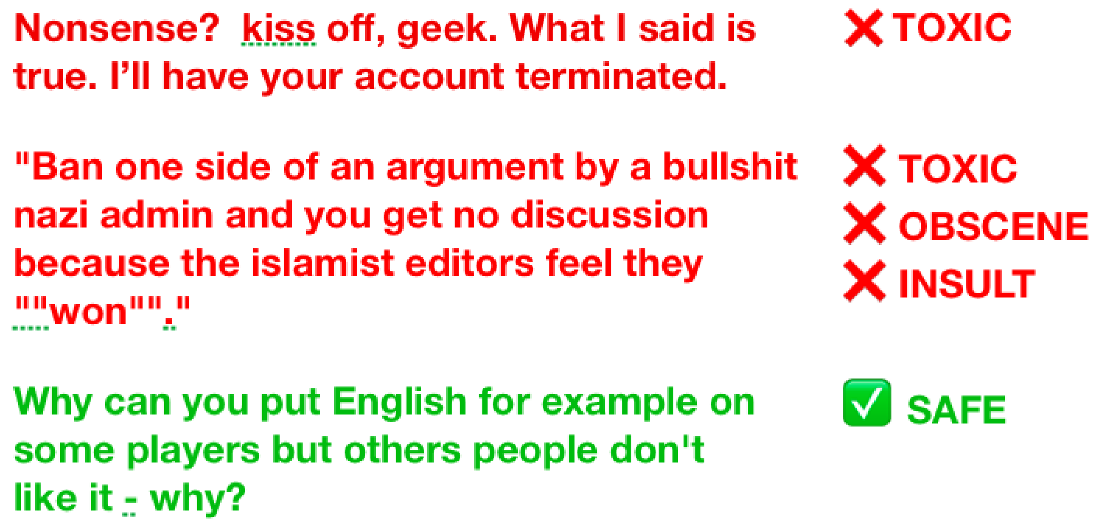
I want to thank my SME

SWATI MAHASETH for giving the dataset and instructions to perform the complete case study process.

INTRODUCTION

Problem Statement: 😁 😐 😠

The background for the problem originates from the multitude of online forums, where-in people participate actively and make comments. As the comments sometimes may be abusive, insulting or even hate-based, it becomes the responsibility of the hosting organizations to ensure that these conversations are not of negative type. The task was thus to build a model which could make prediction to classify the comments into various categories. Consider the following examples:



Nonsense? kiss off, geek. What I said is true. I'll have your account terminated.	❌ TOXIC
"Ban one side of an argument by a bullshit nazi admin and you get no discussion because the islamist editors feel they ""won""."	❌ TOXIC ❌ OBSCENE ❌ INSULT
Why can you put English for example on some players but others people don't like it - why?	✅ SAFE

The exact problem statement was thus as below:

Given a group of sentences or paragraphs, used as a comment by a user in an online platform, classify it to belong to one or more of the following categories — toxic, severe-toxic, obscene, threat, insult or identity-hate with either approximate probabilities or discrete values (0/1).

Introduction:

Online forums and social media platforms have provided individuals with the means to put forward their thoughts and freely express their opinion on various issues and incidents. In some cases, these online comments contain explicit language which may hurt the readers. Comments containing explicit language can be classified into myriad categories such as Toxic, Severe Toxic, Obscene, Threat, Insult, and Identity Hate. The threat of abuse and harassment means that many people stop expressing themselves and give up on seeking different opinions.

To protect users from being exposed to offensive language on online forums or social media sites, companies have started flagging comments and blocking users who are found guilty of using unpleasant language. Several Machine Learning models have been developed and deployed to filter out the unruly language and protect internet users from becoming victims of online harassment and cyberbullying.

Objective:

The main purpose of building this model is to prevent the abusive comment which in turn will Detroit the mindset of an individual or people, now-a-days a lot of abusive and lethargic comment can be seen on various social media platform which create a negative environment among the people and community, so to stop this type of activity a machine learning model is built to identify the malignant text and filter it out as soon as it encounters it.

Review of Literature:

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 Using Machine Learning Techniques. In this we are investigating the application of supervised machine learning techniques to predict the comments. The predictions are based on historical data collected from websites like twitter etc. Different techniques. To build a model for predicting the comments we have used Supervised machine learning.

Motivation:

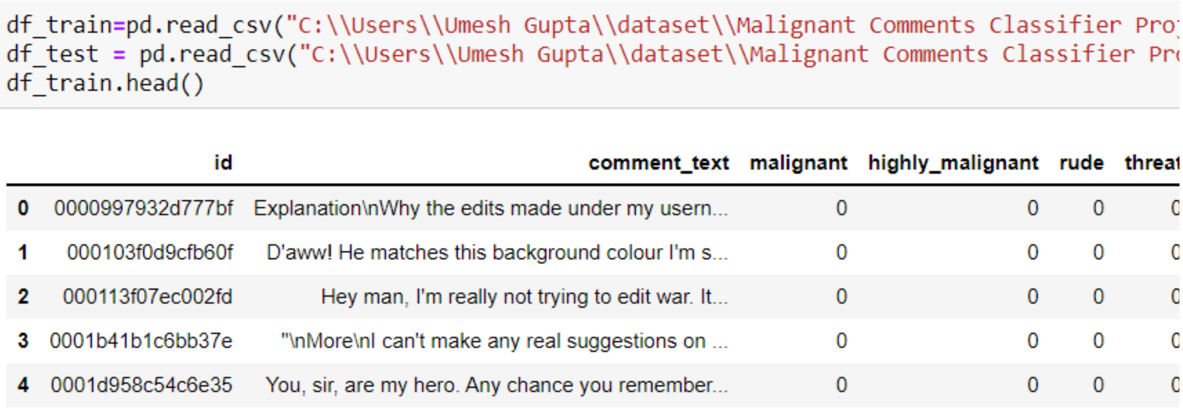
Online platforms when used by normal people can only be comfortably used by them only when they feel that they can express themselves freely and without any reluctance. If they come across any kind of a malignant or toxic type of a reply which can also be a threat or an insult or any kind of harassment which makes them uncomfortable, they might defer to use the social media platform in future. Thus, it becomes extremely essential for any organization or community to have an automated system which can efficiently identify and keep a track of all such comments and thus take any respective action for it, such as reporting or blocking the same to prevent any such kind of issues in the future.

Description of dataset

I had a dataset of 159571 rows and 8 columns for training and 153164 rows and 2 columns for testing. I observed that every 1 in 10 samples was toxic, every 1 in 50 samples was obscene and insulting, but the occurrences of sample being severe-toxic, threat and identity hate was extremely rare.
The dataset consists of the following fields-

- **id**: An 8-digit integer value, to get the identity of the person who had written this
comment
- **comment text**: A multi-line text field which contains the unfiltered comment.
- **malignant**: binary label which contains 0/1 (0 for no and 1 for yes).
- **highly malignant**: binary label which contains 0/1.
- **rude**: binary label which contains 0/1.
- **threat**: binary label which contains 0/1.
- **abuse**: binary label which contains 0/1.
- **loathe**: binary label which contains 0/1

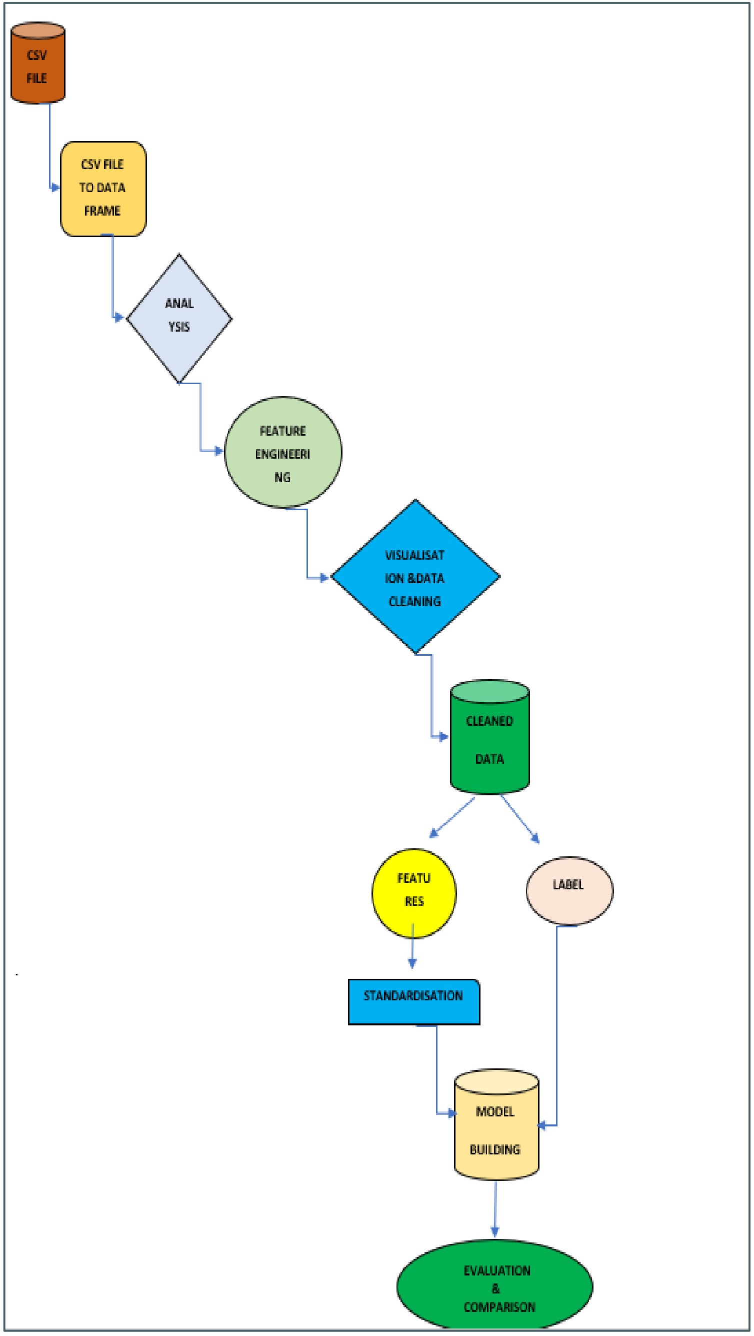
Out of these fields, the comment text field will be pre-processed and fitted into different classifiers to predict whether it belongs to one or more of the labels/outcome variables (i.e., **malignant**, **highly malignant**, **loathe**, threat, **abuse** and **rude**). We have a total of 159571 samples of comments and labelled data, which can be loaded from train.csv file. The first 5 samples are as follows.



```
df_train=pd.read_csv("C:\\Users\\Umesh Gupta\\dataset\\Malignant Comments Classifier Pro
df_test = pd.read_csv("C:\\Users\\Umesh Gupta\\dataset\\Malignant Comments Classifier Pr
df_train.head()
```

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

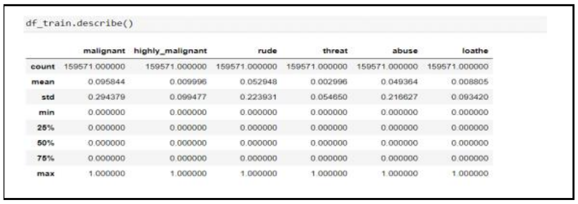
Architecture of dataset



Data pre processing

Data processing and feature engineering are crucial in machine learning to build a prediction model. Furthermore, a model cannot be made without some data processing. For instance, as shown in the experiment, the model could not be trained before handling the missing values and converting the text in the dataset into numerical values. Hence, from the experiment, we saw that pre-processing the data does improve the prediction accuracy.

- **info**: it is used to give info about not null value and datatype of features. here datatype of comments is (object) which is been taken care where as others have int datatype.

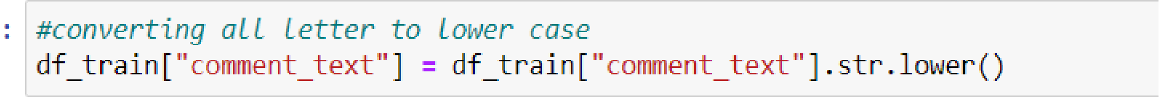


```
df_train.describe()
```

	malignant	highly_malignant	rude	threat	abuse	loathe
count	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
std	0.294379	0.099477	0.223931	0.054650	0.216627	0.093420
min	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
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

- **Describe**: The describe method is used for calculating some statistical data like **percentile, mean** and **std** of the numerical valuesof the Series or Data Frame. It analyses both numeric and object series and also the Data Frame column sets of mixed data types.it also give info about distribution of data.

- **Converting upper case to lower case**: It converts all the upper-case text in the comment to lower case, it is an important technique as it helps in cleaning the data. Quite recently, one of readers found that different variation in input capitalization (e.g., 'Canada' vs. 'Canada') gave him different types of output or no output at all. This was probably happening because the dataset had mixed-case occurrences of the word 'Canada' and there was insufficient evidence for the neural-network to effectively learn the weights for the less common version. This type of issue is bound to happen when your dataset is fairly small, and lowercasing is a great way to deal with sparsity issues.

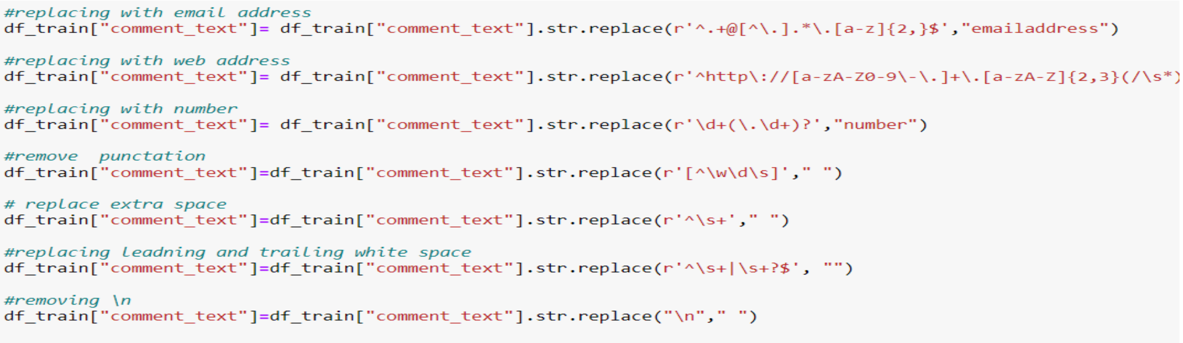


```
: #converting all letter to lower case
  df_train["comment_text"] = df_train["comment_text"].str.lower()
```

- **Text Normalisation**: As I was now certain that there are no missing records in my data, I decided to start with data pre-processing. Firstly, I decided to normalize the text data since comments from online forums usually contain inconsistent language, use of special characters in place of letters (e.g., @rgument), as well as the use of numbers to represent letters (e.g., n0t). To tackle such inconsistencies in data, I decided to use ***Regex.*** The text normalization steps that I performed are listed below: -

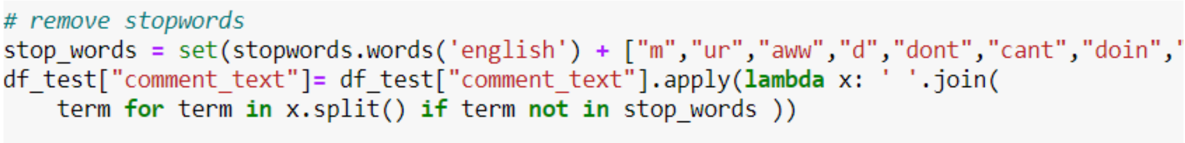
- Removing Characters in between Text.
- Removing Repeated Characters.
- Converting data to lower-case.

- Removing Punctuation.
- Removing unnecessary white spaces in between words.
- Removing “\n”.
- Removing Non-English characters.



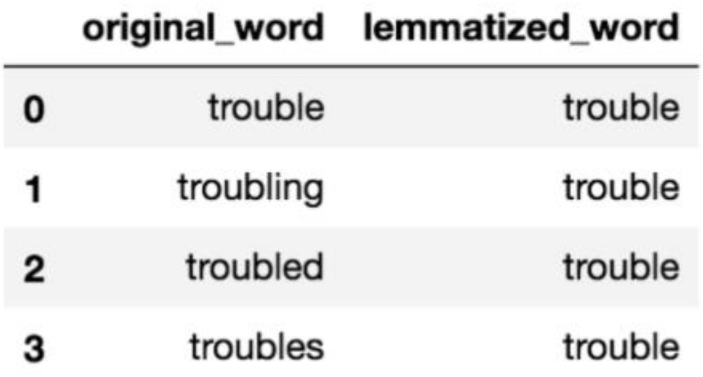
```
#replacing with email address
df_train["comment_text"]= df_train["comment_text"].str.replace(r'^.+@[^\.].*\.[a-z]{2,}$',"emailaddress")

#replacing with web address
df_train["comment_text"]= df_train["comment_text"].str.replace(r'^http\://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(/\s*)

#replacing with number
df_train["comment_text"]= df_train["comment_text"].str.replace(r'\d+(\.\d+)?',"number")

#remove  punctation
df_train["comment_text"]=df_train["comment_text"].str.replace(r'[^\w\d\s]'," ")

# replace extra space
df_train["comment_text"]=df_train["comment_text"].str.replace(r'^\s+'," ")

#replacing leadning and trailing white space
df_train["comment_text"]=df_train["comment_text"].str.replace(r'^\s+|\s+?$', "")

#removing \n
df_train["comment_text"]=df_train["comment_text"].str.replace("\n"," ")
```

❖ **Stop word Removal:** *Stop words are those words that are frequently used in both written and verbal communication and thereby do not have either a positive/negative impact on our statement.* Stop words are a set of commonly used words in a language. Examples of stop words in English are “a”, “the”, “is”, “are” and etc. The intuition behind using stop words is that, by removing low information words from text, we can focus on the important words instead. For example, in the context of a search system, if your search query is “what is text processing?”, you want the search system to focus on surfacing documents that talk about text pre-processing over documents that talk about what is. This can be done by preventing all words from your stop word list from being analysed. Stop words are commonly applied in search systems, text classification applications, topic modelling, topic extraction and others.

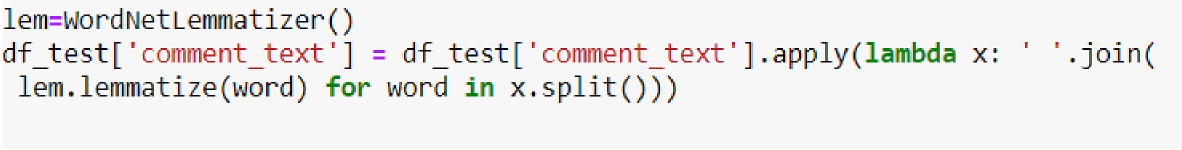


```
# remove stopwords
stop_words = set(stopwords.words('english') + ["m","ur","aww","d","dont","cant","doin",'
df_test["comment_text"]= df_test["comment_text"].apply(lambda x: ' '.join(
    term for term in x.split() if term not in stop_words ))
```

❖ **Lemmatization:** Lemmatization on the surface is very similar to stemming, where the goal is to remove inflections and map a word to its root form. The only difference is that, lemmatization tries to do it the proper way. It doesn’t just chop things off, it actually transforms words to the actual root. For example, the word “better” would map too “good”. It may use a dictionary such as word net for mapping or some special rule-based approaches. Here is an example of lemmatization in action using a WordNet-based approach:



	original_word	lemmatized_word
0	trouble	trouble
1	troubling	trouble
2	troubled	trouble
3	troubles	trouble



```
lem=WordNetLemmatizer()
df_test['comment_text'] = df_test['comment_text'].apply(lambda x: ' '.join(
 lem.lemmatize(word) for word in x.split()))
```

- **Tf-Idf vectorization:** In NLP, tf-idf is an important measure and is used by algorithms like cosine similarity to find documents that are similar to a given search query.

  - What is Term Frequency (tf): tf is the number of times a term appears in a particular document. So, it’s specific to a document. A few of the ways to calculate tf is given below: -

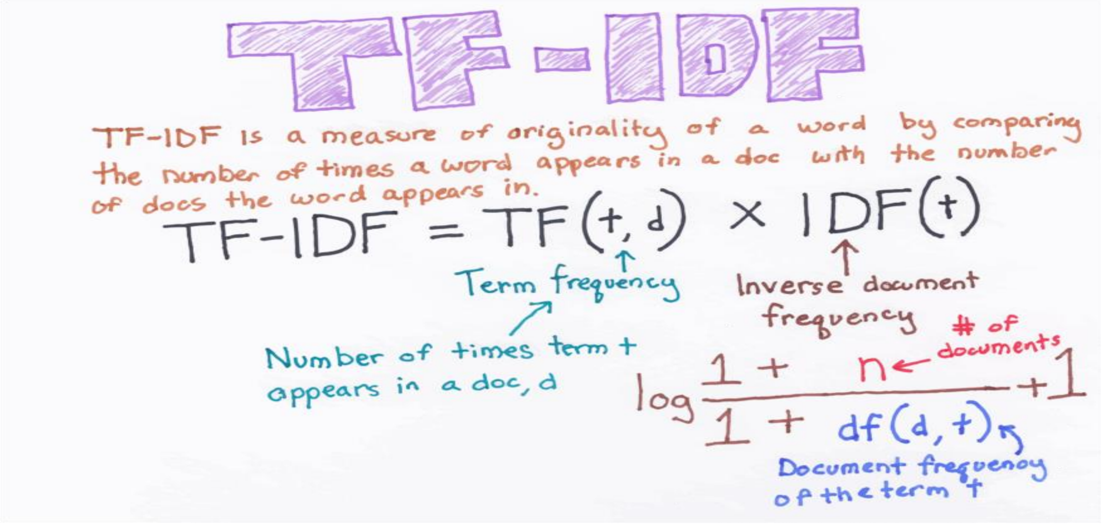
  tf(t) = No. of times term ‘t’ occurs in a document.

- **Inverse Document Frequency (idf):idf is a measure of how common or rare a term is across the entire corpus of documents. So, the point to note is that it’s common to all the documents. If the word is common and appears in many documents, the idf value (normalized) will approach 0 or else approach 1 if it’s rare. A few of the ways we can calculate idf value for a term is given below:**

idf(t) = $\log_e$ [(1+n) / (1 + df(t))] + 1 (default i: e smooth_idf = True)

and

idf(t) = $\log_e$ [n / df(t)] + 1 (when smooth_idf = False)



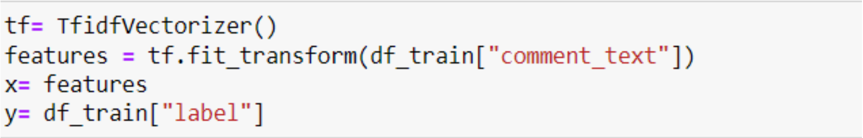
TF-IDF

TF-IDF is a measure of originality of a word by comparing the number of times a word appears in a doc with the number of docs the word appears in.

$$\text{TF-IDF} = \text{TF}(t,d) \times \text{IDF}(t)$$

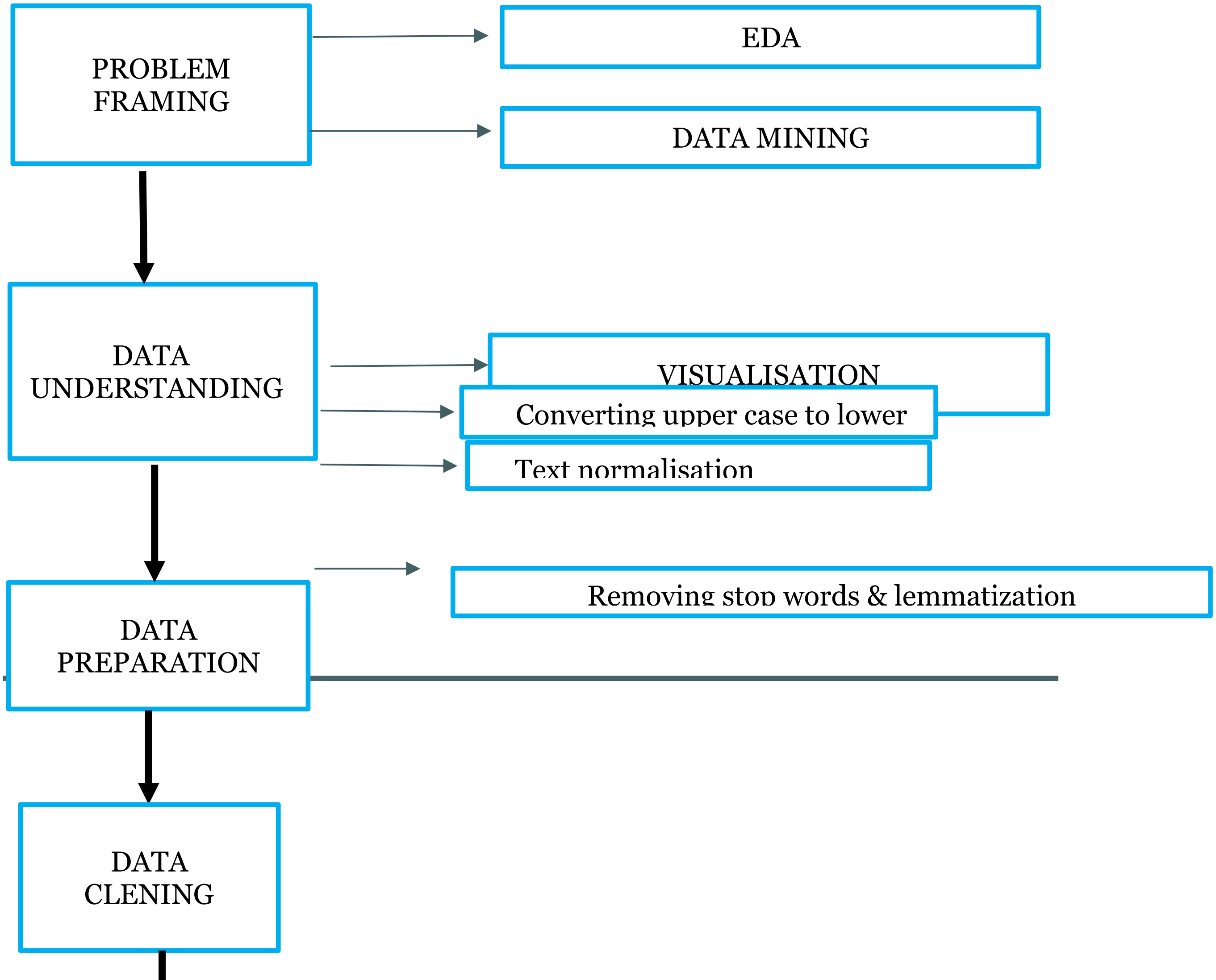
$$\log \frac{1 + n}{1 + df(d,t)} + 1$$

Term frequency
Number of times term t appears in a doc, d
Inverse document frequency
# of documents
Document frequency of the term t



```
tf= TfidfVectorizer()
features = tf.fit_transform(df_train["comment_text"])
x= features
y= df_train["label"]
```

Model development and evaluation



PROBLEM FRAMING

EDA

DATA MINING

DATA UNDERSTANDING

VISUALISATION

Converting upper case to lower

Text normalisation

DATA PREPARATION

Removing stop words & lemmatization

DATA CLENING

Above is the architecture which shows the important steps involve in data pre-processing and model building. Firstly, I analys [illegible] datatype of certain columns is not correct so I corrected that and with the help of feature engineering I have created numerical column out of object column as our machine learning model will not understand text, then for understanding I have used plots to visualise and understand data. After imputation comes data cleaning which is done by detecting outliers, checking skewness and removing unwanted columns which does not add value to model accuracy. After data cleaning splitting of data into test and train and then doing normalisation of data so that it does not get biased toward a particular feature. Finally comes model building, I have used 4 classifier algorithm () which fits best in this dataset and predict good accuracy. After finding my best model with the help of accuracy and cross validation score comes hyper parameter tuning for best fit model.

Accuracy score

Hardware requirement: -

- Minimum core i5 or higher
- Minimum 8gb of ram

Software and tools required:

- Python is widely used in scientific and numeric computing:
- SciPy is a collection of packages for mathematics, science, and engineering.
- Pandas is a data analysis and modelling library.

Modules or library required for project data analysis and visualization:

- **Pandas** for data analysis and import
- **NumPy** to perform Mathematical operation
- **Seaborn** and **matplotlib** to data visualization
- **Scikit-learn:** All the models, metrics and feature selection etc are present inside of that module. We import from this library according to our need.

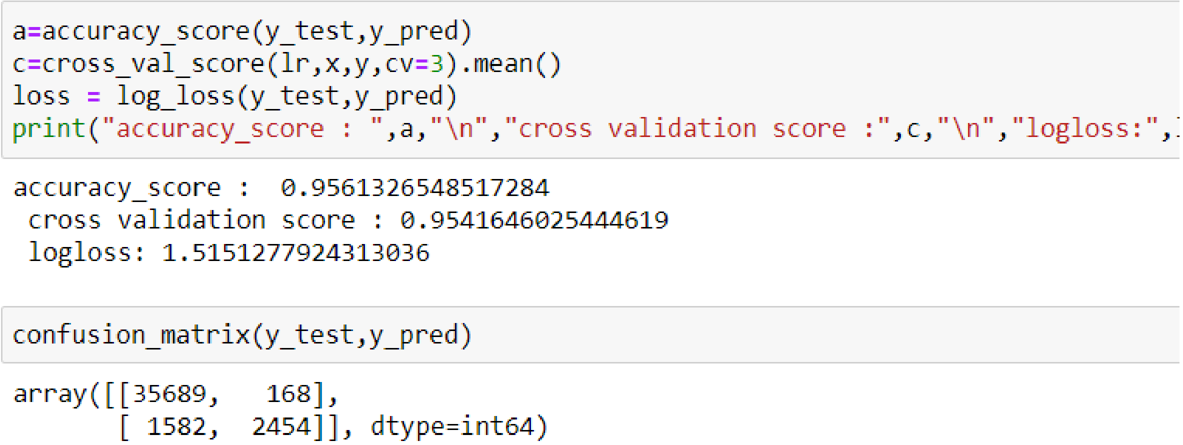
- **Train-Test split**: There are two primary phases in the system:

1. **Training phase:** The system is trained by using the data in the data set and fits a model (line/curve) based on the algorithm chosen accordingly.

2. **Testing phase:** the system is provided with the testing data, and it is tested for its working. The accuracy is checked. And therefore, the data that is used to train the model or test it, must be appropriate. The system is designed to detect and predict price of used car and hence appropriate algorithms must be used to do the two different tasks

Model Building: I use 4 different algorithms for model building:

Logistic Regression:



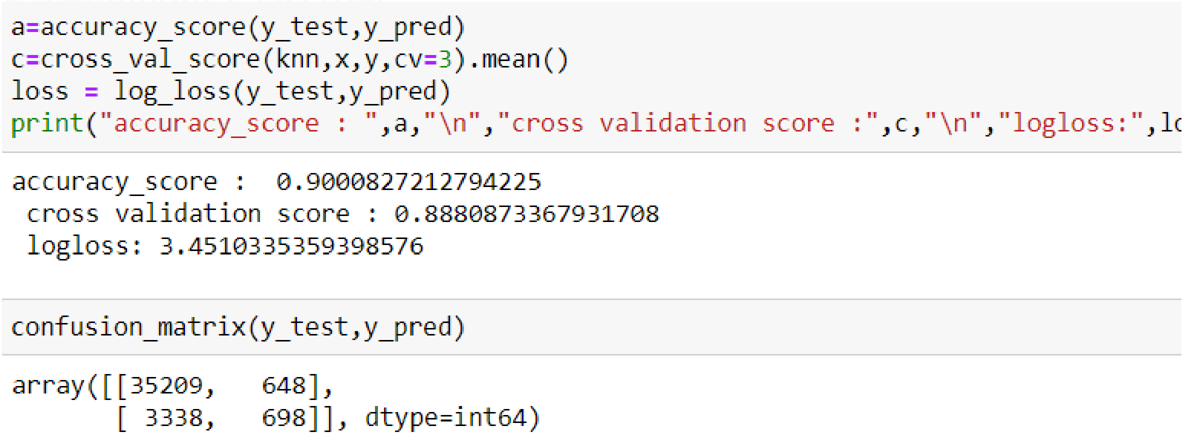
```
a=accuracy_score(y_test,y_pred)
c=cross_val_score(lr,x,y,cv=3).mean()
loss = log_loss(y_test,y_pred)
print("accuracy_score : ",a,"\n","cross validation score :",c,"\n","logloss:",l
```

```
accuracy_score :  0.9561326548517284
 cross validation score : 0.9541646025444619
 logloss: 1.5151277924313036
```

```
confusion_matrix(y_test,y_pred)
```

```
array([[35689,   168],
       [ 1582,  2454]], dtype=int64)
```

KNeighbour Classifier:



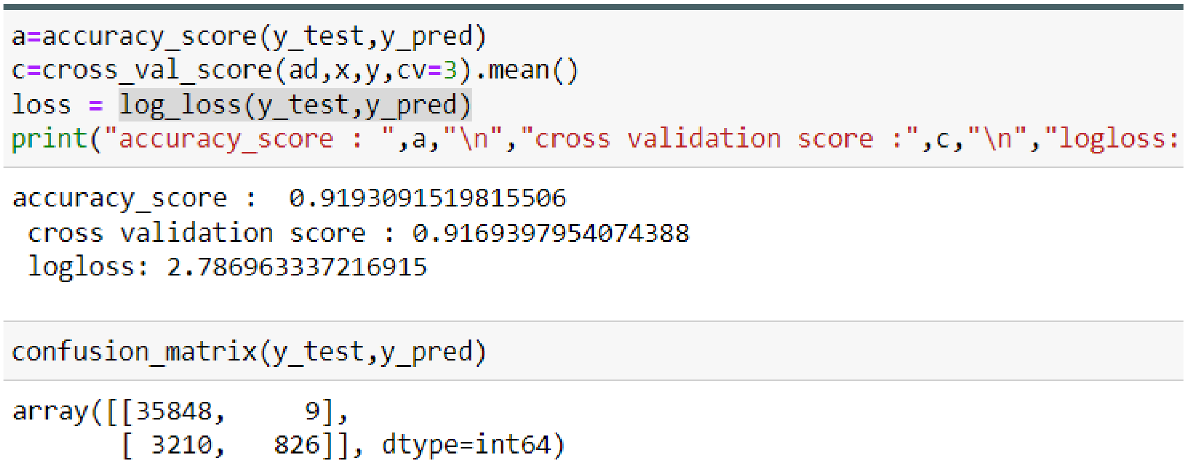
```
a=accuracy_score(y_test,y_pred)
c=cross_val_score(knn,x,y,cv=3).mean()
loss = log_loss(y_test,y_pred)
print("accuracy_score : ",a,"\n","cross validation score :",c,"\n","logloss:",lo
```

```
accuracy_score :  0.9000827212794225
 cross validation score : 0.8880873367931708
 logloss: 3.4510335359398576
```

```
confusion_matrix(y_test,y_pred)
```

```
array([[35209,   648],
       [ 3338,   698]], dtype=int64)
```

MultinomialNB:



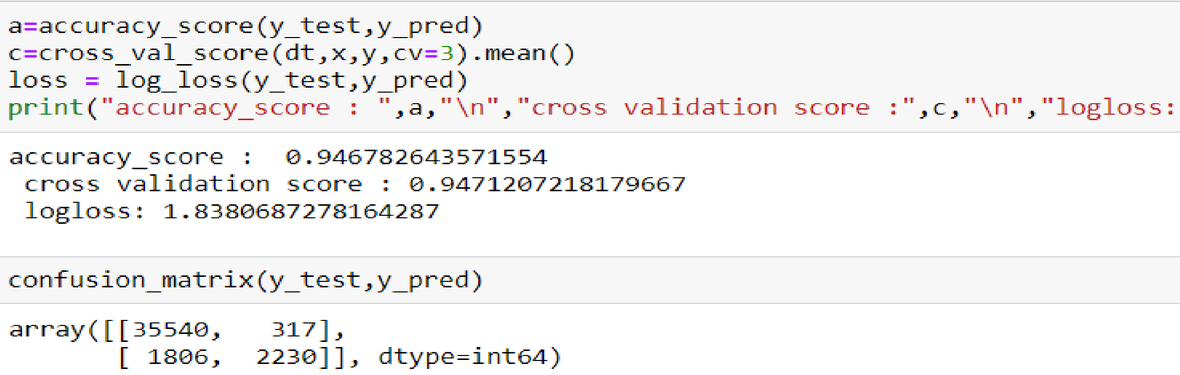
```
a=accuracy_score(y_test,y_pred)
c=cross_val_score(ad,x,y,cv=3).mean()
loss = log_loss(y_test,y_pred)
print("accuracy_score : ",a,"\n","cross validation score :",c,"\n","logloss:
```

```
accuracy_score :  0.9193091519815506
 cross validation score : 0.9169397954074388
 logloss: 2.786963337216915
```

```
confusion_matrix(y_test,y_pred)
```

```
array([[35848,     9],
       [ 3210,   826]], dtype=int64)
```

AdaBoost Classifier:



```
a=accuracy_score(y_test,y_pred)
c=cross_val_score(dt,x,y,cv=3).mean()
loss = log_loss(y_test,y_pred)
print("accuracy_score : ",a,"\n","cross validation score :",c,"\n","logloss:
```

```
accuracy_score :  0.946782643571554
 cross validation score : 0.9471207218179667
 logloss: 1.8380687278164287
```

```
confusion_matrix(y_test,y_pred)
```

```
array([[35540,   317],
       [ 1806,  2230]], dtype=int64)
```

Accuracy score:

Accuracy is one metric for evaluating classification models. Informally, accuracy is the fraction of predictions our model got right. Formally, accuracy has the following definition:

Accuracy=Number of correct predictions /Total number of predictions

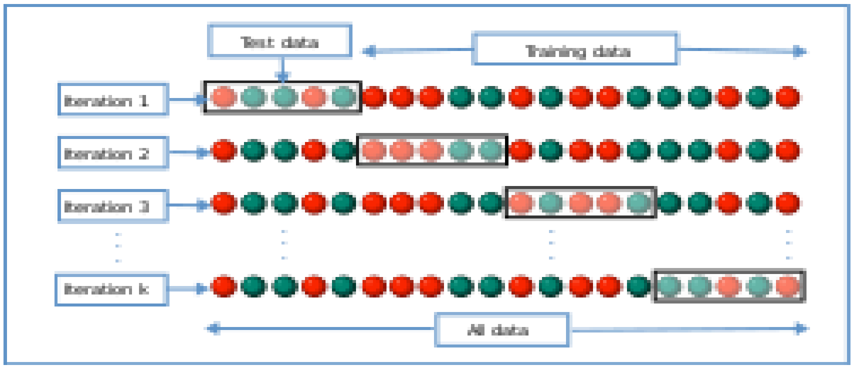
For binary classification, accuracy can also be calculated in terms of positives and negatives as follows:

$$\text{Accuracy}=(TP+TN)/(TP+TN+FP+FN)$$

Where *TP* = True Positives, *TN* = True Negatives, *FP* = False Positives, and *FN* = False Negatives

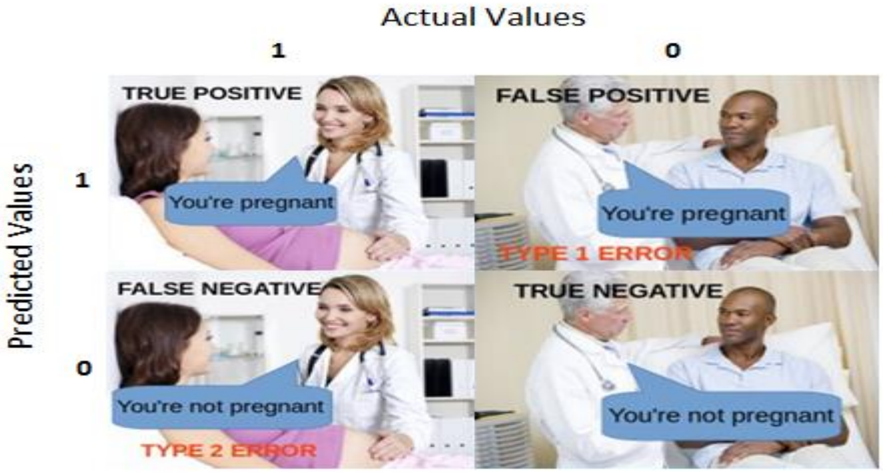
Cross Validation Score:

Cross-validation is a statistical method used to estimate the skill of machine learning models. It is commonly used in applied machine learning to compare and select a model for a given predictive modelling problem because it is easy to understand, easy to implement, and results in skill estimates that generally have a lower bias than other methods. The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation. When a specific value for k is chosen, it may be used in place of k in the reference to the model, such as k=10 becoming 10-fold cross-validation.



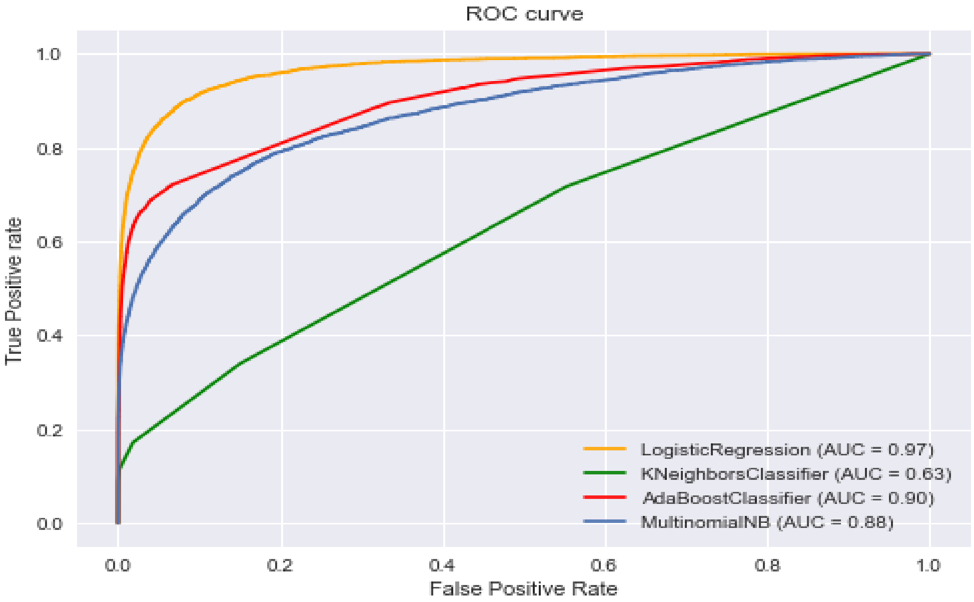
Confusion matrix:

A confusion matrix is a table that is often used to **describe the performance of a classification model** (or "classifier") on a set of test data for which the true values are known. The confusion matrix itself is relatively simple to understand, but the related terminology can be confusing. Well, it is a performance measurement for machine learning classification problem where output can be two or more classes. It is a table with 4 different combinations of predicted and actual values. It is extremely useful for measuring Recall, Precision, Specificity, Accuracy, and most importantly AUC-ROC curves.
Let’s understand TP, FP, FN, TN in terms of pregnancy analogy.



Log loss: Log Loss is the **most important classification metric based on probabilities**. ... For any given problem, a lower log-loss value means better predictions.

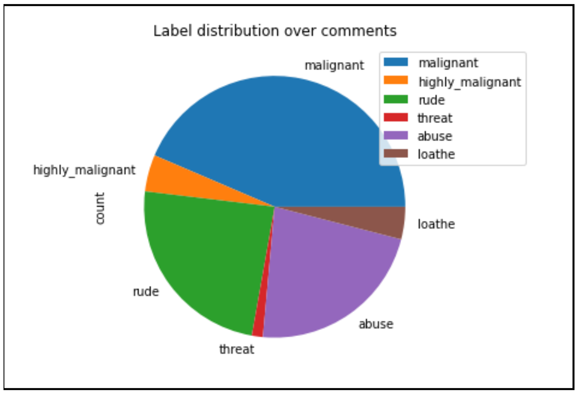
AUC-ROC CURVE: AUC - ROC curve is a performance measurement for the classification problems at various threshold settings. ROC is a probability curve and AUC represents the degree or measure of separability. It tells how much the model is capable of distinguishing between classes. Higher the AUC, the better the model is at predicting 0 classes as 0 and 1 classes as 1.



Visualisation:

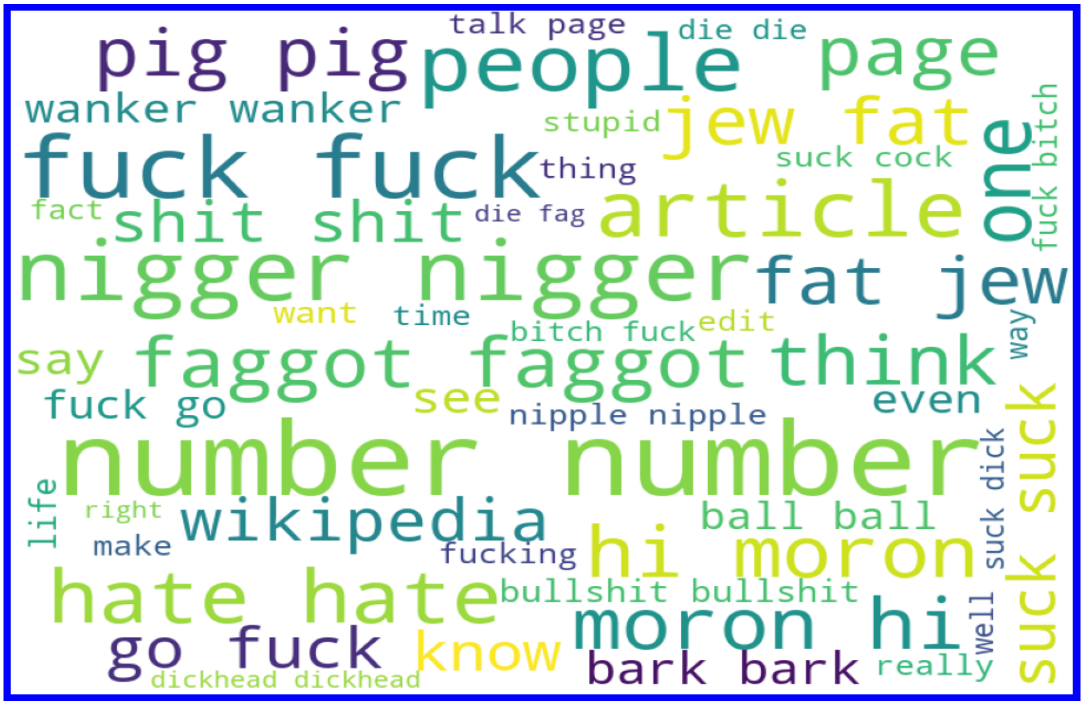
Data visualization is the discipline of trying to understand data by placing it in a visual context so that patterns, trends and correlations that might not otherwise be detected can be exposed.

This pie plot gives us the information about the no of comment which are classified as malignant, highly-malignant, abuse, loathe, threat. It gives us the information about the distribution of comments which are malignant into various sections.



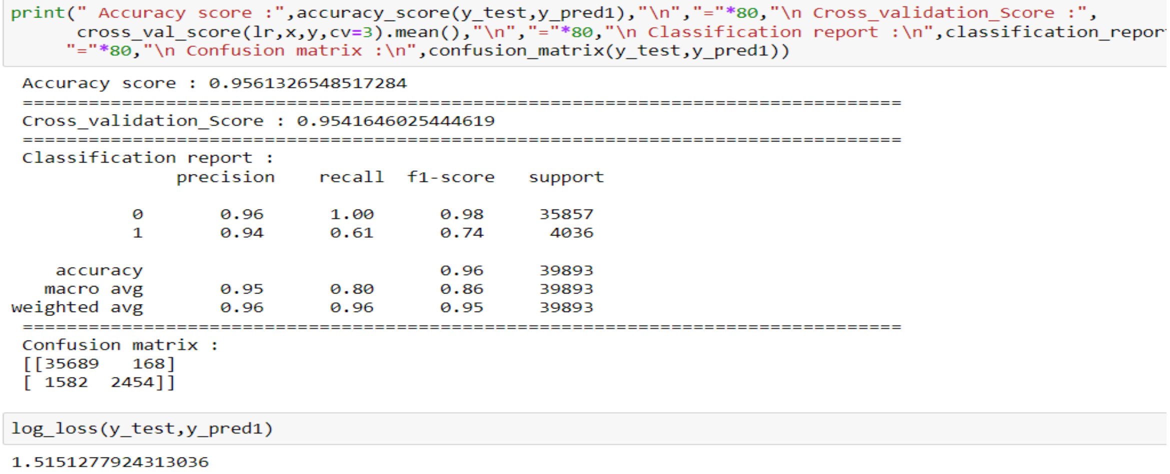
Word Cloud:

Word Cloud is a data visualization technique used for representing text data in which the size of each word indicates its frequency or importance. Significant textual data points can be highlighted using a word cloud. Word clouds are widely used for analyzing data from social network websites.



RESULT:

Many machine learning algorithms are used to predict. However, the prediction accuracy of these algorithms depends heavily on the given data when training the model. If the data is in bad shape, the model will be overfitted and inefficient, which means that data pre-processing is an important part of this experiment and will affect the final results. Thus, multiple combinations of pre-processing methods need to be tested before getting the data ready to be used in training. After analyzing every model logistic regression shows good accuracy and cv with least difference and on doing hyper parameter tuning it accuracy reaches to 96%.



```
print(" Accuracy score :",accuracy_score(y_test,y_pred1),"\n","="*80,"\n Cross_validation_Score :",
      cross_val_score(lr,x,y,cv=3).mean(),"\n","="*80,"\n Classification report :\n",classification_repor
     "="*80,"\n Confusion matrix :\n",confusion_matrix(y_test,y_pred1))
```

```
 Accuracy score : 0.9561326548517284
 ================================================================================
 Cross_validation_Score : 0.9541646025444619
 ================================================================================
 Classification report :
               precision    recall  f1-score   support

           0       0.96      1.00      0.98     35857
           1       0.94      0.61      0.74      4036

    accuracy                           0.96     39893
   macro avg       0.95      0.80      0.86     39893
weighted avg       0.96      0.96      0.95     39893
 ================================================================================
 Confusion matrix :
 [[35689   168]
 [ 1582  2454]]
```

```
log_loss(y_test,y_pred1)
```

```
1.5151277924313036
```

CONCLUSION:

Communication is one of the basic necessities of everyone’s life. People need to talk and interact with one another to express what they think. Over the years, social media and social networking have been increasing exponentially due to an upsurge (rise) in the use of the internet. Flood of information arises from online conversation on a daily basis, as people are able to discuss, express themselves and express their opinion via these platforms. While this situation is highly productive and could contribute significantly to the quality of human life, it could also be destructive and enormously dangerous. The responsibility lies on the social media administration, or the host of organization to control and monitor these comments.

This research work focuses on developing a model that would automatically classify a comment as either malignant or non-malignant using logistic regression. Therefore, this study aims to develop a multi-headed model to detect different types of malignant comment like threats, rude, abusive, and loathe. By collecting and preprocessing malignant comments for training and testing using term frequency- inverse document frequency (TF-IDF) algorithm, developing a multi-headed model will detect different types of malignant comment using logistic regression to train the dataset, and evaluate the model using confusion metrics

LIMITATION:

As per my understanding and study of dataset I presented the best model with good accuracy, this is the best model until someone came with an extraordinary approach and technique. Moreover, this study will not cover all classification algorithms, instead, it is focused on the chosen algorithm, starting from the basic classification techniques to the advanced ones.

FUTURE WORK:

In future this machine learning model may bind with various website which can provide real time data for price prediction. Also, we may add large historical data of flight price which can help to improve accuracy of the machine learning model. We can build an android app as user interface for interacting with user. For better performance, we plan to judiciously design deep learning network structures, use adaptive learning rates and train on clusters of data rather than the whole dataset.

REFRENCE:

https://www.researchgate.net/publication/338170211_Identification_and_Classification_of_Toxic_Comments_on_Social_Media_using_Machine_Learning_Techniques.

https://scholar.smu.edu/cgi/viewcontent.cgi?article=1134&context=datasciencereview