FLIP ROBO

Malignant Comments Classification

A Project Report by:

Archit Khanduja

Acknowledgement

The project would not have been built without the constant support from **DataTrained** and **Fliprobo** teams.

https://www.nltk.org/

https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

https://www.geeksforgeeks.org/removing-stop-words-nltk-python/

Introduction

Business problem Framing

Millions of people use various channels of Social Media daily and it is now the part of their daily routine, where, they can freely express themselves. This freedom of speech is sometimes taken for granted by a few people, who post abusive, racist or insensitive comments on Social Media.

It is difficult for social media companies to incorporate humans to figure out and filter malignant comments. If we can train a Machine Learning model upon historical data of such comments, we will be able to build a model and classify whether a comment is malignant or not.

Conceptual Background of the Domain Problem

This problem proposed will be solved using Natural Language Processing first wherein we will be using techniques to identify the comments which are malignant. Then we will be using Machine Learning Classification algorithms to train our model as per the data available and check our model performance using the appropriate measures.

Motivation for the Problem Undertaken

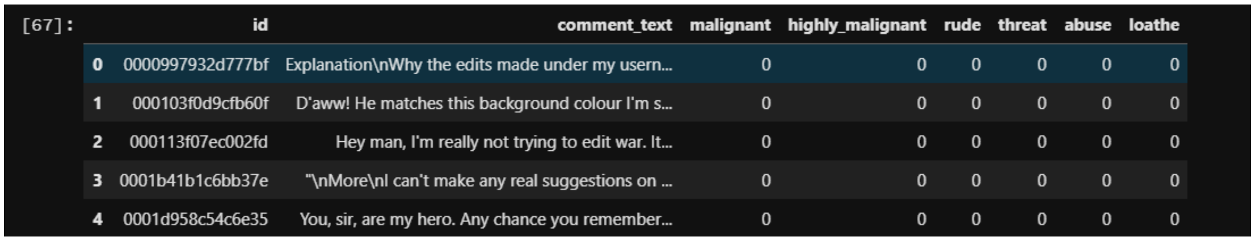
There have been a huge surge in cyberbullying and cybercrimes since people have started talking about political and religious issues online. Also, people have been posting racist, sexist and other forms of abusive comments and it is very necessary to filter out and remove these comments on a timely basis so that the cyberbullying cases can be identified and there can be a strict action on this.

To pursue this in an automated manner without much human intervention, we will use NLP to identify the malignant comments and train a ML model to classify them.

Analytical Problem Framing

Mathematical/ Analytical Modelling of the Problem

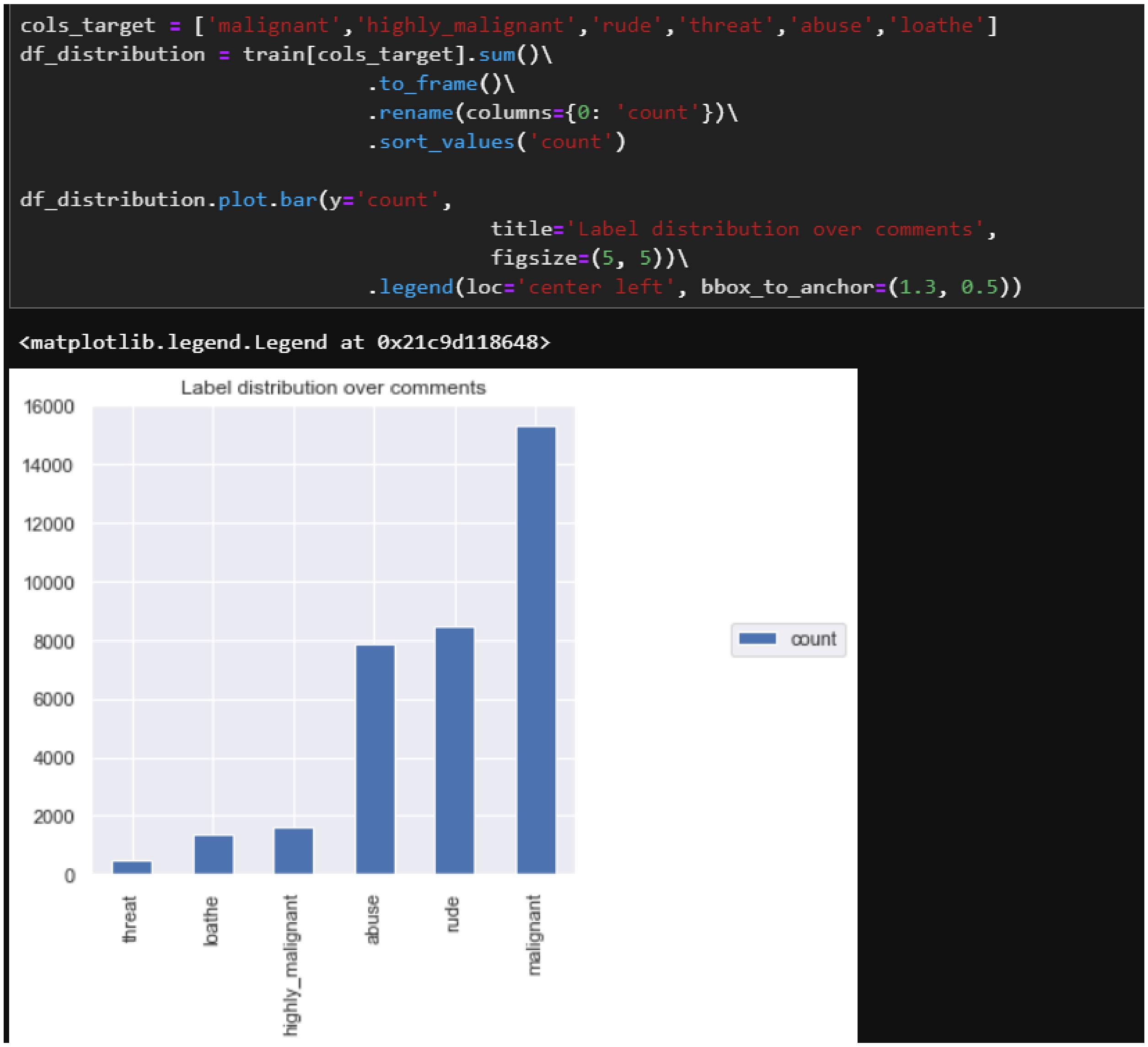
In this problem, the dataset that has been provided to us has **159571 Rows and 8 columns** as shown below:



[67]:

	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

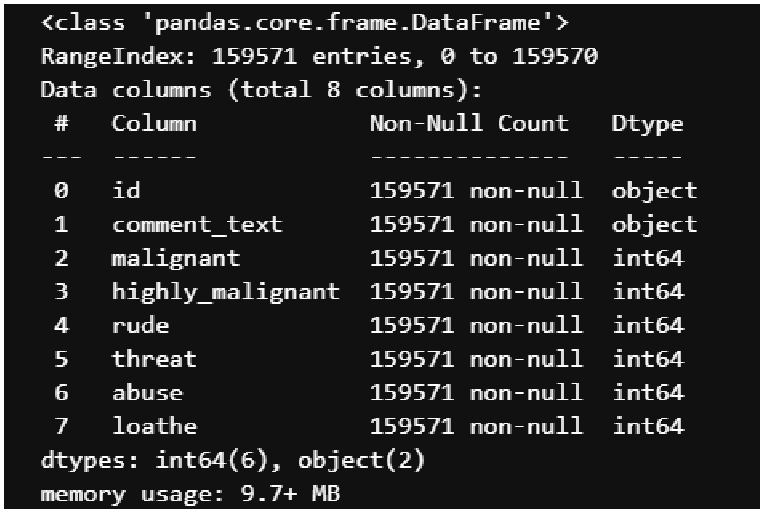
Also, we have multiple labels and multiple target variables involved in our case. Distribution of various labels can be seen below:



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

```
<matplotlib.legend.Legend at 0x21c9d118648>
```

Data Sources and their Formats



```
<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
```

Following are the data sources along with their formats:

id : A unique id aligned with each comment text. Its datatype is object.

comment_text: It includes the comment text. Its datatype is object.

malignant: It is a column with binary values depicting which comments are malignant in nature. Its datatype is int.

highly_malignant: Binary column with labels for highly malignant text. Its datatype is int.

rude: Binary column with labels for comments that are rude in nature. Its datatype is int.

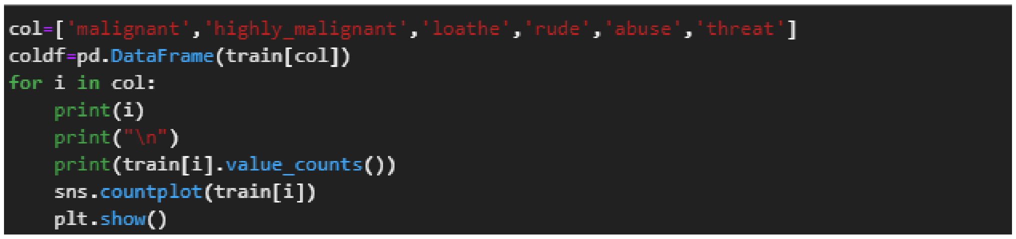
threat: Binary column with labels for threatening context in the comments. Its datatype is int.

abuse: Binary column with labels with abusive behaviour. Its datatype is int.

loathe: Label to comments that are full of loathe and hatred. Its datatype is int.

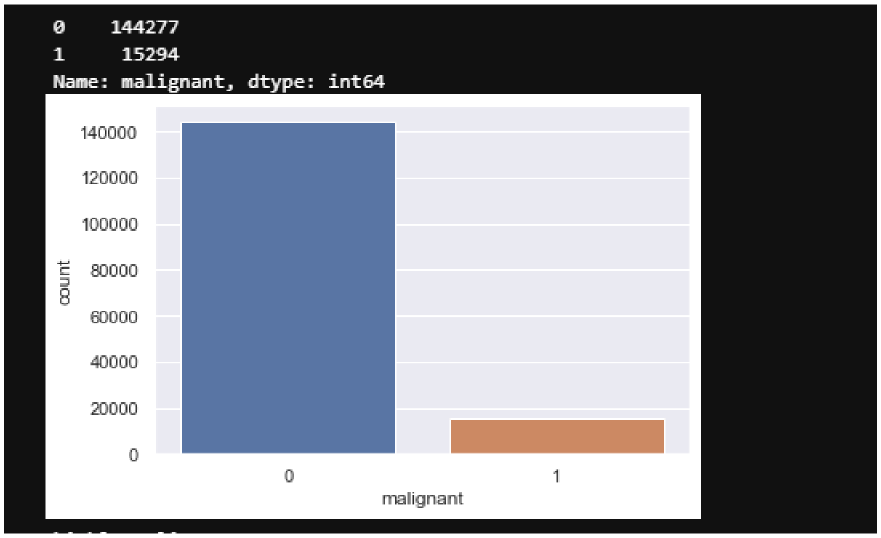
EDA and Data Visualisation

Firstly, we visualised various target columns that we have in our dataset by the following code:



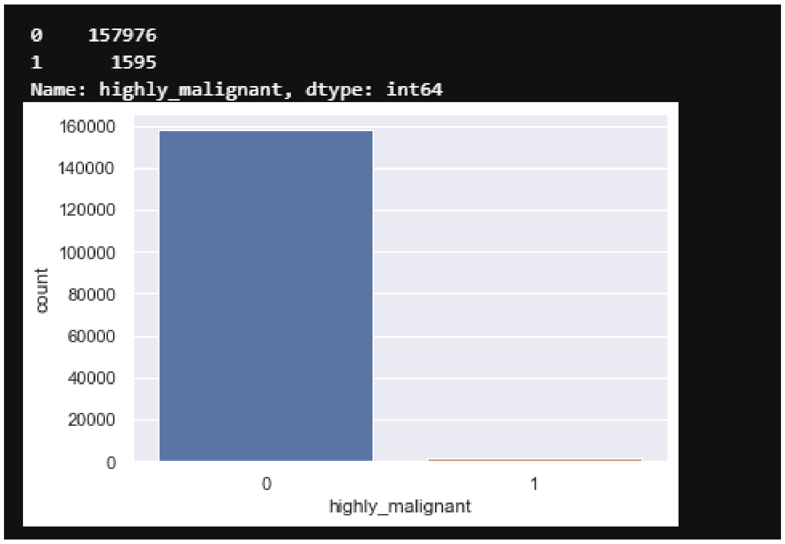
```
col=['malignant','highly_malignant','loathe','rude','abuse','threat']
coldf=pd.DataFrame(train[col])
for i in col:
    print(i)
    print("\n")
    print(train[i].value_counts())
    sns.countplot(train[i])
    plt.show()
```

1. Malignant



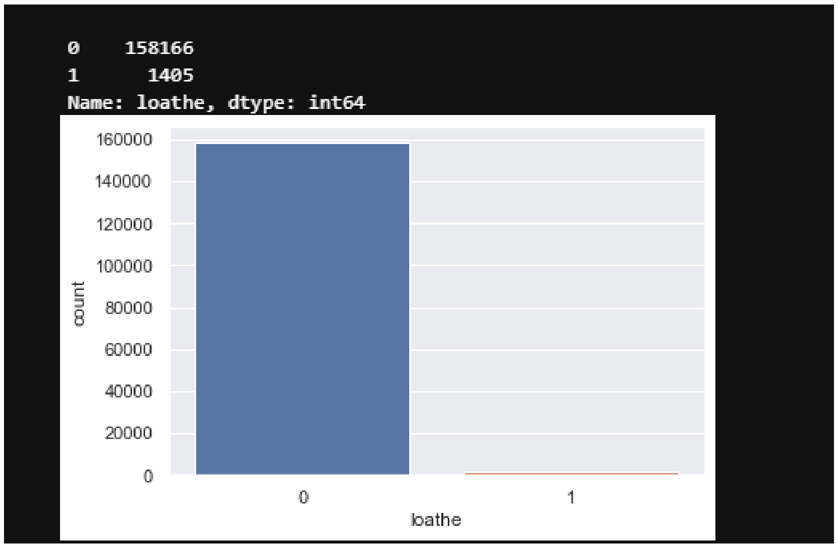
```
0    144277
1     15294
Name: malignant, dtype: int64
```

2. Highly Malignant



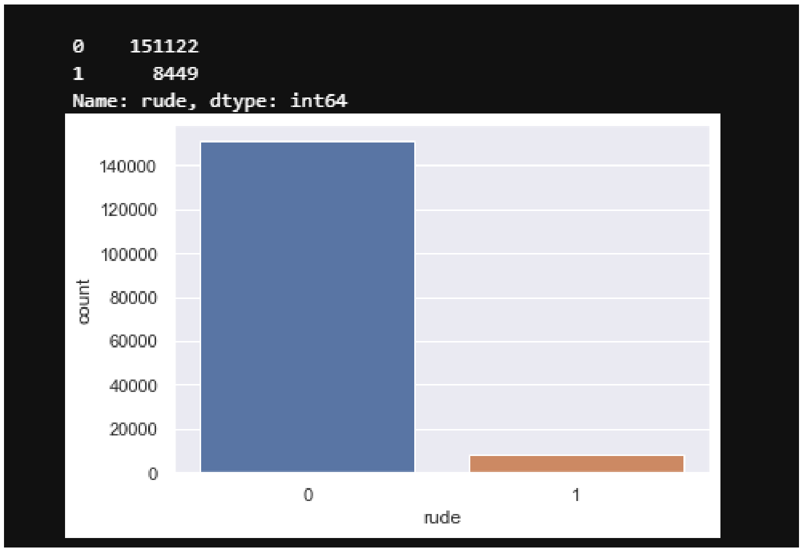
```
0    157976
1      1595
Name: highly_malignant, dtype: int64
```

3. Loathe



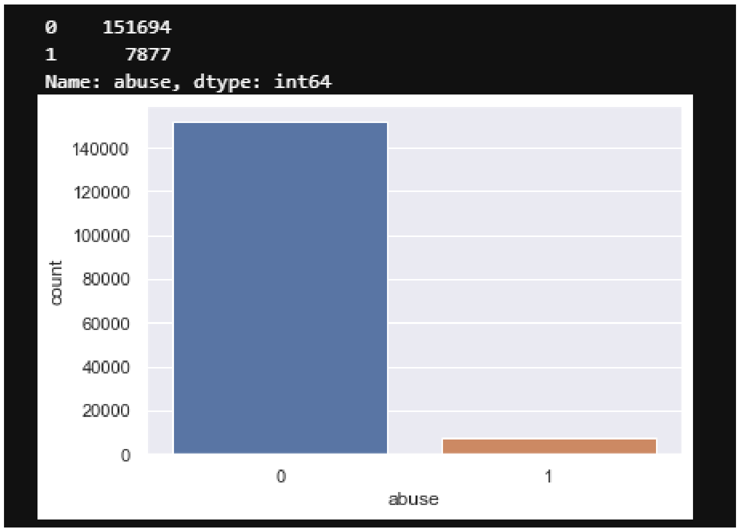
```
0    158166
1      1405
Name: loathe, dtype: int64
```

4. Rude



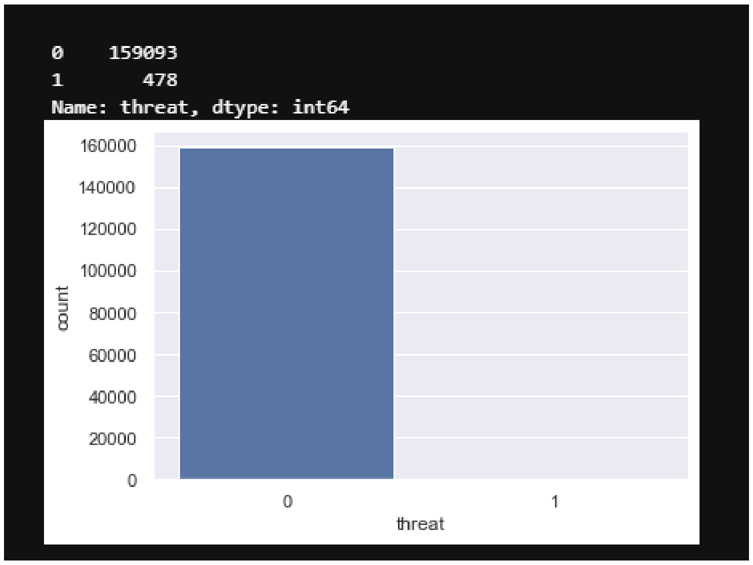
```
0    151122
1      8449
Name: rude, dtype: int64
```

5. Abuse



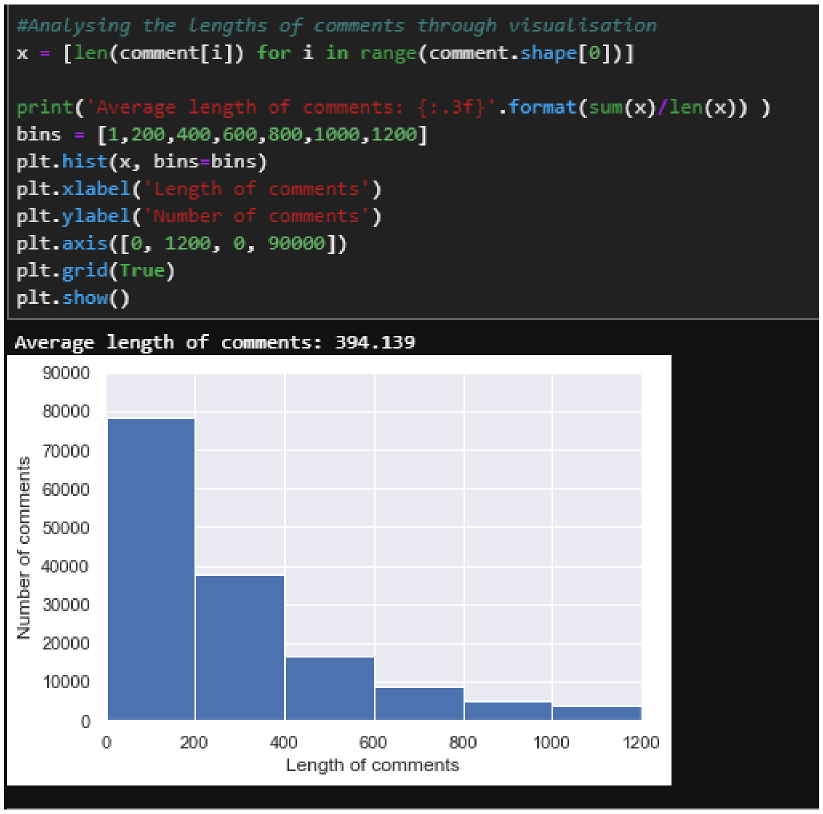
```
0    151694
1      7877
Name: abuse, dtype: int64
```

6. Threat



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

Analysis of Length of comments through Visualisation



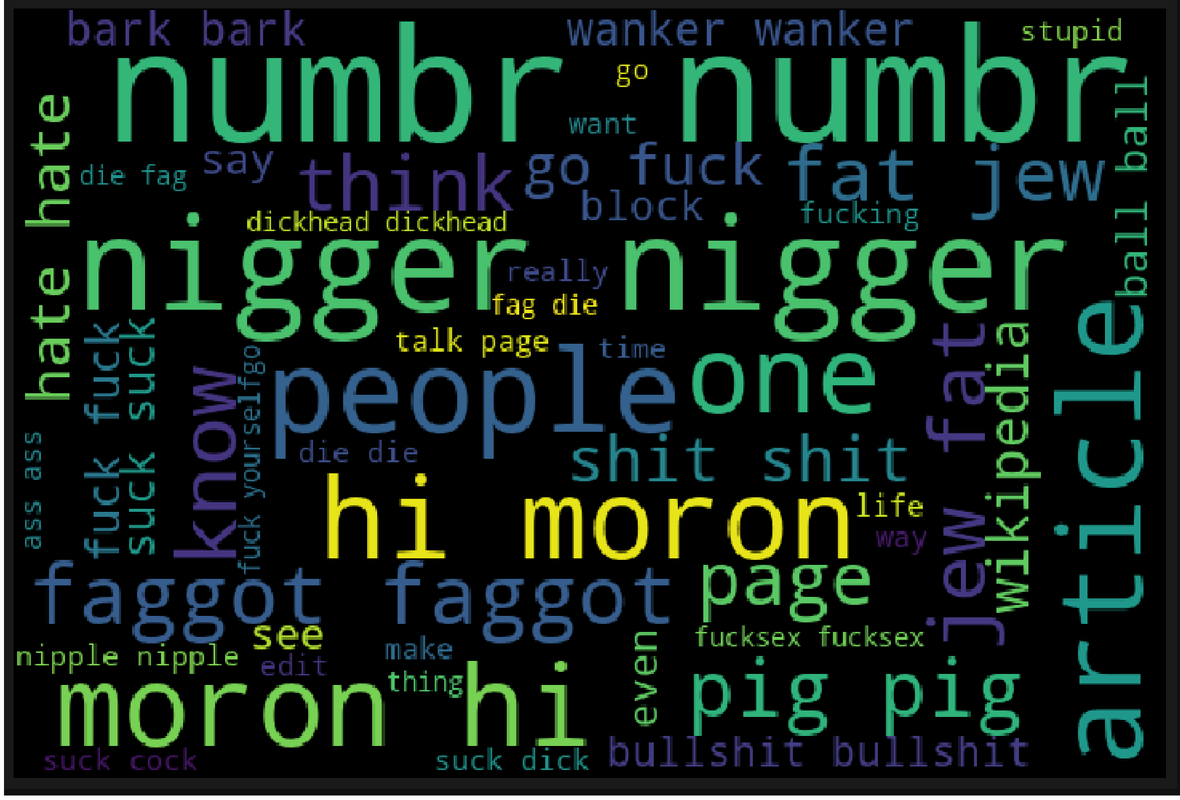
```
#Analysing the lengths of comments through visualisation
x = [len(comment[i]) for i in range(comment.shape[0])]

print('Average length of comments: {:.3f}'.format(sum(x)/len(x)) )
bins = [1,200,400,600,800,1000,1200]
plt.hist(x, bins=bins)
plt.xlabel('Length of comments')
plt.ylabel('Number of comments')
plt.axis([0, 1200, 0, 90000])
plt.grid(True)
plt.show()
```

```
Average length of comments: 394.139
```

The average length of comments was found to be 394.

Getting the sense of Loud Words through visualisation

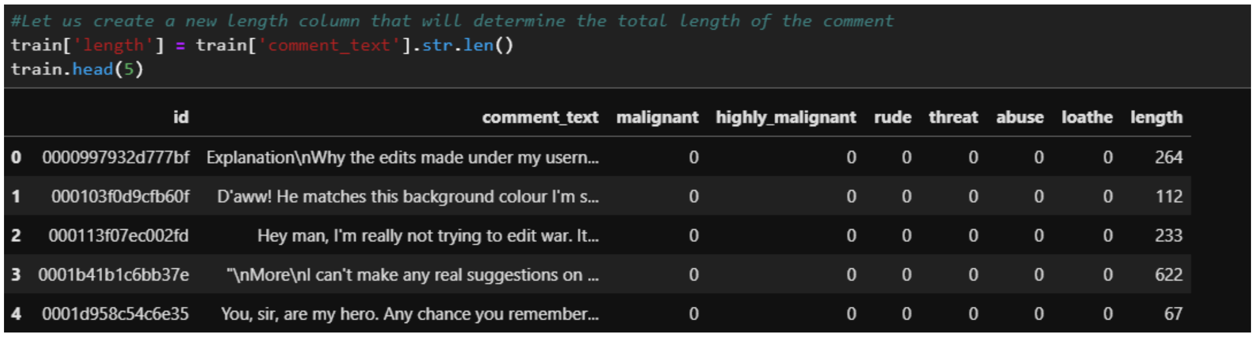
```
#Getting sense of loud words which are offensive
from wordcloud import WordCloud
hams = train['comment_text'][train['malignant']==1]
spam_cloud = WordCloud(width=600,height=400,background_color='black',max_words=50).generate(' '.join(hams))
plt.figure(figsize=(10,8),facecolor='k')
plt.imshow(spam_cloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```



Data Preprocessing Done

Since this is an NLP problem, we needed to do a lot of NLP based preprocessing. The first thing that we did was to separate our comment column and determined the length of the comments. It was important because too much lengthy comments can increase our training time and decrease our accuracy.

So first, we extracted a new feature of comment length as shown below:



```
#Let us create a new length column that will determine the total length of the comment
train['length'] = train['comment_text'].str.len()
train.head(5)
```

	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	length
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0	264
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0	112
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0	233
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0	622
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0	67

Then, we went on and removed punctuations and other special characters, followed by the removal of stop words. Also, before removing the stop words, we split the comments into individual words. We then used stemming and lemmatizing in order to reduce more set of words from our comments by identifying the similar set of words and bringing them under the umbrella of 1 single word.

Post that, we extracted another column which has the length of shorter comments after the data preprocessing has been done.

Original Length 62893130

Clean Length 43575187

Also, before we could send our model for a ML Algorithm, we converted it into a vector form by TFIDF Vectorizer so that our model can read the data in numerical form.

Assumptions

1. Attributes are independent of each other (low or no multicollinearity).
2. Objects in the target variable are identically distributed.
3. A linear classifier will assume that the decision boundaries are linear.
4. In case of logistic regression, we assume that there is a linear relationship between the logit of the outcome and each predictor variables.

Hardware and Software Requirements and Tools Used

For the building of this model, an MSI Machine with Intel CORE i7 7th Gen processor and an 8 GB RAM was used.

The programming language used was Python. The compiler that was used was Anaconda Navigator. The programs were run in Jupyter Notebook environments.

The libraries used were as follows: numpy, pandas, matplotlib, seaborn, sklearn, scipy, eli5

Model/s Development and Evaluation

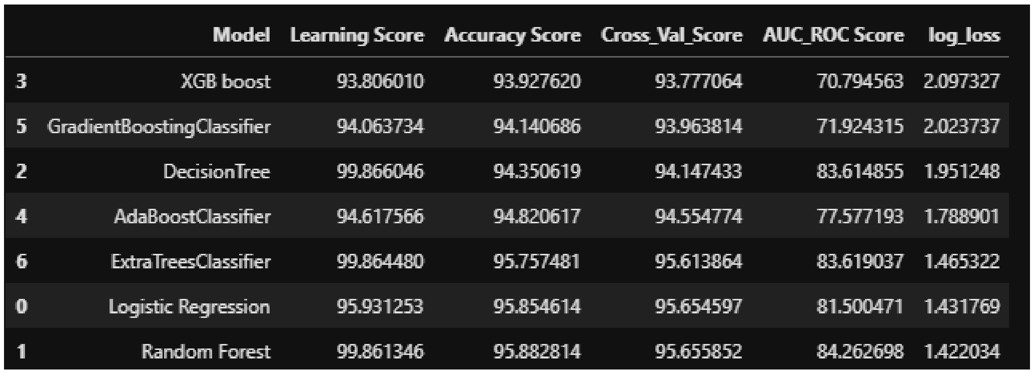
Identification of possible problem-solving approaches

Upon doing some research on such problems, it was identified that for such problems, RF models are the best as they are non-linear tree based models. I had an intuition of RF model to be the one, but still I tested my dataset on the following algorithms:

1. Logistic Regression
2. Decision Trees
3. Random Forest
4. XGBOOST
5. Gradient Boosting
6. Adaboost
7. Extra Trees Classifier

Run and Evaluate selected models

Different models were tried upon after doing the train test split. Following is the modelwise dataframe obtained after running all the models:



	Model	Learning Score	Accuracy Score	Cross_Val_Score	AUC_ROC Score	log_loss
3	XGB boost	93.806010	93.927620	93.777064	70.794563	2.097327
5	GradientBoostingClassifier	94.063734	94.140686	93.963814	71.924315	2.023737
2	DecisionTree	99.866046	94.350619	94.147433	83.614855	1.951248
4	AdaBoostClassifier	94.617566	94.820617	94.554774	77.577193	1.788901
6	ExtraTreesClassifier	99.864480	95.757481	95.613864	83.619037	1.465322
0	Logistic Regression	95.931253	95.854614	95.654597	81.500471	1.431769
1	Random Forest	99.861346	95.882814	95.655852	84.262698	1.422034

Key Metrics for success in solving problem under consideration

The key matrices used in solving the problem were applied in the same code of model building. Those matrices were Accuracy Score, Cross validation Score, AUC ROC Score, log_loss and learning score.

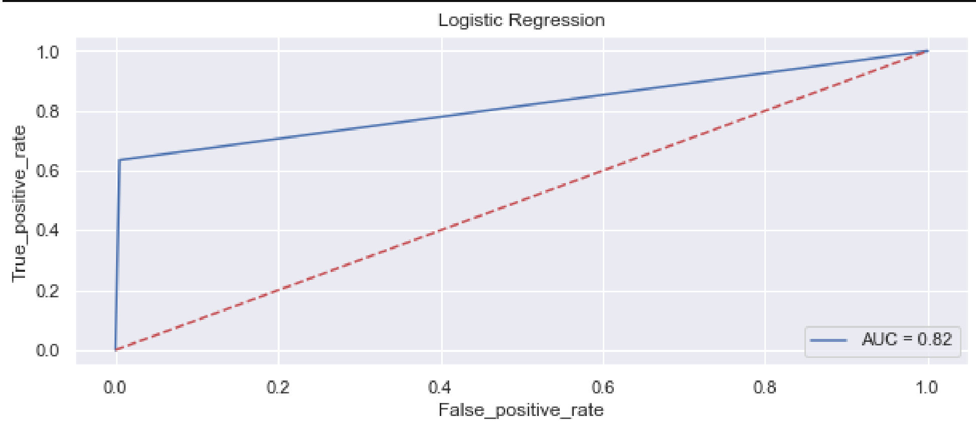
Cross validation score was used to create a more generic model so that it performs well under different circumstances and in various permutations and combinations of data.

Also, as we can see, log loss score is also inversely proportional to accuracy score and it is closer to zero in case of RF and Extra Trees. It has helped us in strengthening our conclusion of cross val scores.

In addition to this, AUC ROC score is one of the key metric for evaluation as it tells us how capable the model is in distinguishing between the positive and negative classes. It means that it observes the True Positive Rate and False Positive Rate for users who paid the loan and are falsely marked as defaulters.

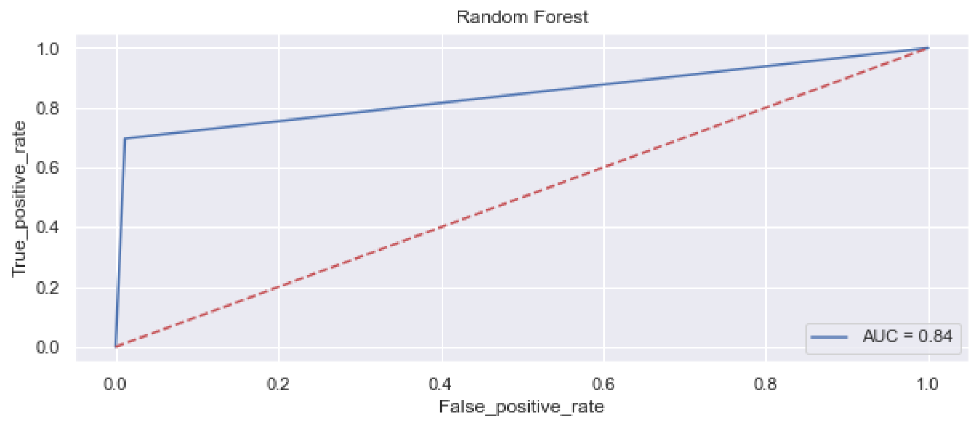
Visualizations and Interpretations

1. AUC ROC curve
   i) Logistic Regression



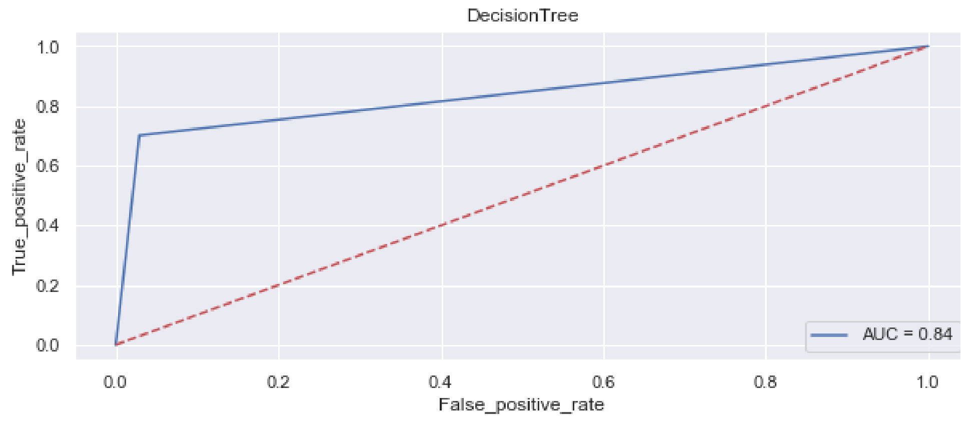
The area under the curve is 0.82, which means that 82% of the predictions by the model are correct. Closer the line is towards the Y axis, more is the area under the curve.

   ii) Random Forest



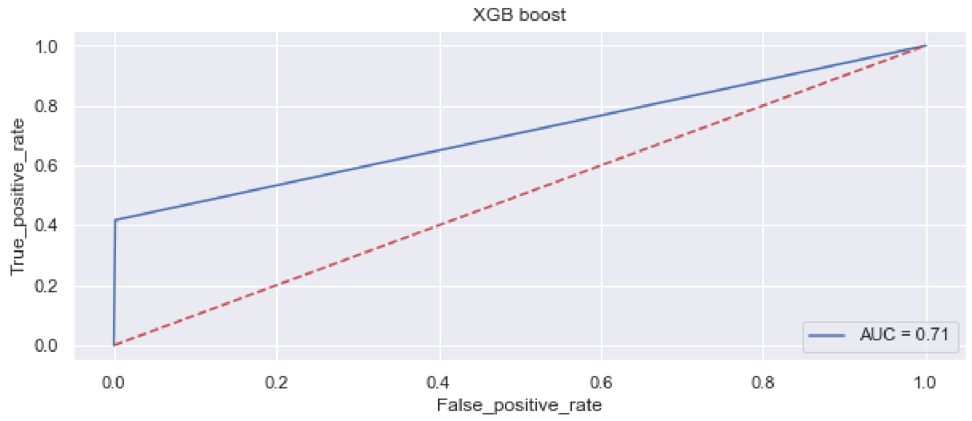
The area under the curve is 0.84, which means that 84% of the predictions by the model are correct.

   iii) Decision Tree



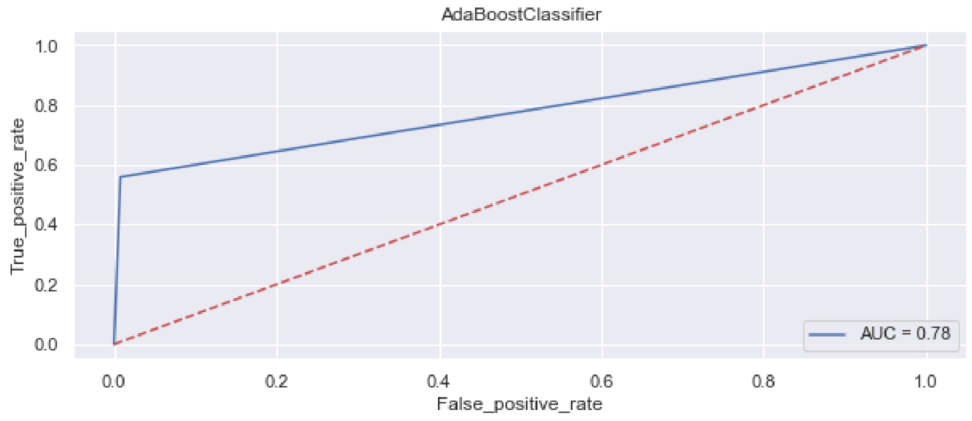
The area under the curve is 0.84, which means that 84% of the predictions by the model are correct.

iv) XG Boost



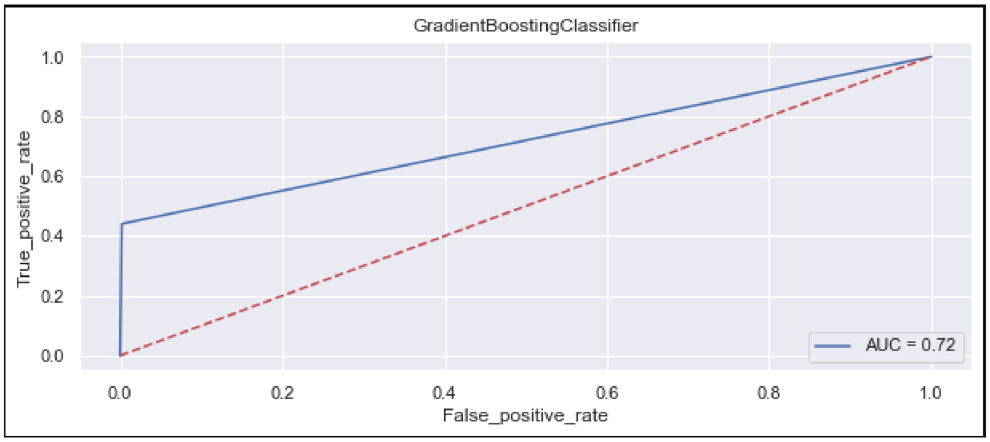
The area under the curve is 0.71, which means that 71% of the predictions by the model are correct.

v) ADA Boost



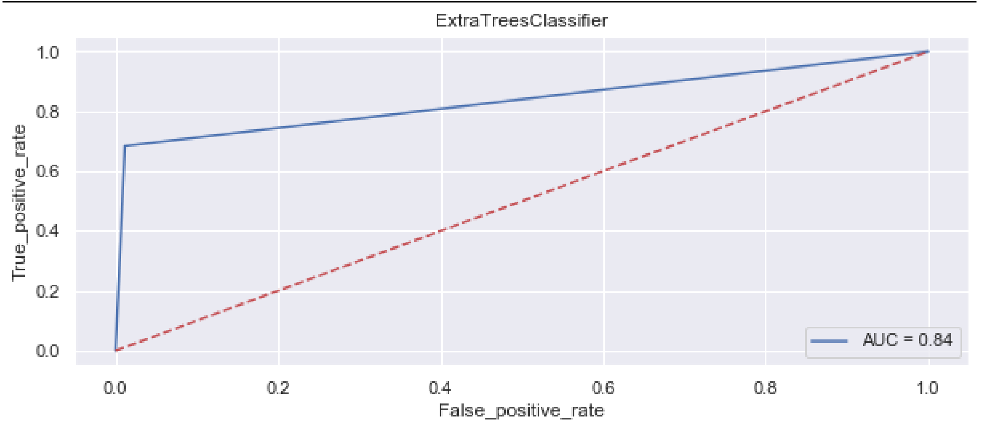
The area under the curve is 0.78, which means that 78% of the predictions by the model are correct.

vi) Gradient Boosting



The area under the curve is 0.72, which means that 72% of the predictions by the model are correct.

vii) Extra Trees Classifier



The area under the curve is 0.84, which means that 84% of the predictions by the model are correct.

Conclusion

Key Findings and Conclusions of the Study

After doing the whole study, following were the key findings:

1. As a social media channel, we were able to filter out most of the comments as malignant with an accuracy score of 95% and AUC ROC score of 84%, which are pretty decent numbers.
2. Also, the log loss score was minimised in the RF model.

Limitations of this work and Scope for Future Work

There would be a lot of comments which will not be classified in this. For example, comments posted in languages other than English will not be classified by the model because the model was trained only in English language.

Moreover, we would not be able to identify sarcasm and figure of speech, which again can be a malignant comment but our machine cannot understand the same.