FLIP ROBO

Malignant Comment Classifier Project

Submitted by:
Sushil Joshi

ACKNOWLEDGMET

I express my sincere gratitude to Flip Robo Technologies for giving me the opportunity to work on this project on Malignant Comment Classifier using machine learning algorithms and NLTK suite of libraries and also, for providing me with the requisite datasets for training and testing prediction accuracies of the models. I acknowledge my indebtedness to the authors of the papers titled: “Toxic Comment Classification” and “Machine learning methods for toxic comment classification: a systematic review” for providing me with invaluable knowledge and insights into what constitute as malignant and benign comments and the role of natural language processing tools and techniques in identifying them and in helping build models to classify input comments as malignant and benign.

INTRODUCTION

Business Problem Framing

With the proliferation of social media there has been an emergence of conflict and hate, making online environments uninviting for users. There is a lack of models for online hate detection. Online hate, described as abusive language, aggression, cyberbullying, hatefulness and many others has been identified as a major threat on online social media platforms. Social media platforms are the most prominent grounds for such toxic behaviour. 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

Predictive modelling, Classification algorithms are some of the machine learning techniques used along with the various libraries of the NLTK suite for Classification of comments.

Using NLTK tools, the frequencies of malignant words occurring in textual data were estimated and given appropriate weightage, whilst filtering out words, and other noise which do not have any impact on the semantics of the comments and reducing the words to their base lemmas for efficient processing and accurate classification of the comments.

Review of Literature

Two research papers titled: “Toxic Comment Classification” by Sara Zaheri and “Machine learning methods for toxic comment classification: a systematic review” by Darko Androcec were reviewed and studied to gain insights into the nature of malignant comments, their impact on social media platforms and the various methods that are employed for training models to detect, identify and classify them.

Motivation for the Problem Undertaken

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. Automatic recognition of malignant comments on online forums, and social media serves as a useful provision for moderators of public platforms as well as users who could receive warnings and filter unwanted contents. The need of advanced methods and techniques to improve identification of different types of comments posted online motivated the current project.

Analytical Problem Framing

Mathematical/ Analytical Modeling of the Problem

Various Classification analysis techniques were used to build Classification models to determine whether an input Message content is benign or malignant. Machine Learning Algorithms such as Multinomial Naïve Bayes and Complement Naïve Bayes were employed which are based on the Bayes Theorem:

P(message is malignant | message content) = P(message content | malignant). P(malignant) / P(message content)

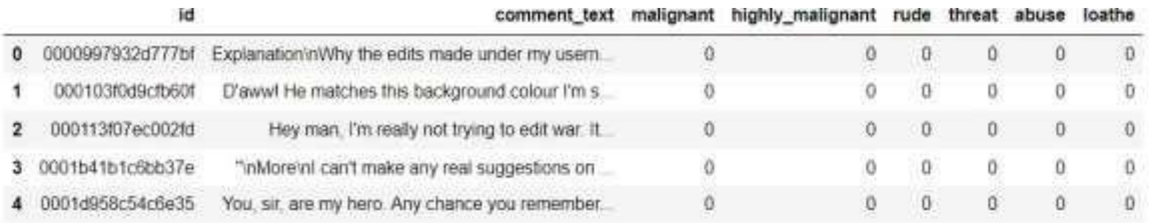
The probability of message being Malignant, knowing that Message Content has occurred could be calculated. Event of "Message Content" represents the evidence and "Message is Malignant", the hypothesis to be approved. The theorem runs on the assumption that all predictors/features are independent and the presence of one would not affect the other.

The approach to classify a comment as malignant would depend on training data labelled as various categories of malignant messages and benign messages.

Data Sources and their formats

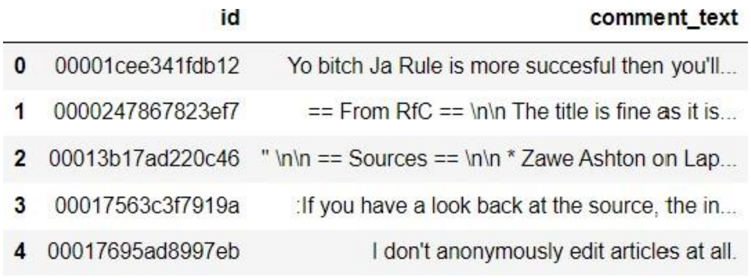
The data set contains the training set, which has approximately 1,59,000 samples and the test set which contains nearly 1,53,000 samples. All the data samples contain 8 fields which includes „Id", „Comments", „Malignant", „Highly malignant", „Rude", „Threat", „Abuse" and „Loathe".

The label can be either 0 or 1, where 0 denotes a NO while 1 denotes a YES. There are various comments which have multiple labels. The first attribute is a unique ID associated with each comment.



	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

Figure 1 Train Dataset



	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.

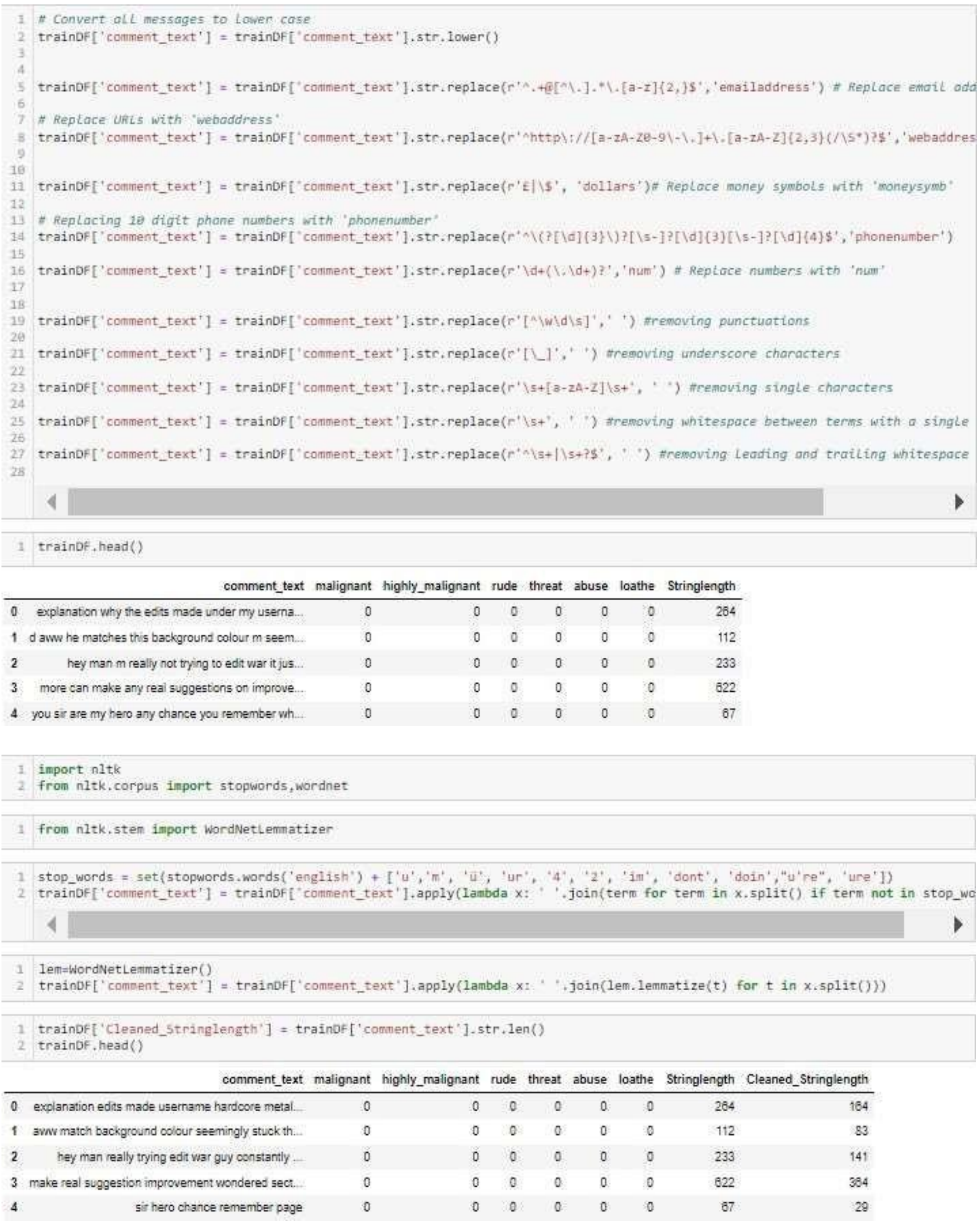
Figure 2 Test Dataset

The data set includes:

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

The dataset was checked to see if there were any null values or random characters present. None were found.

Column: **ID** was dropped since they don't contribute to building a good model for predicting the target variable values.



```
# Convert all messages to lower case
trainDF['comment_text'] = trainDF['comment_text'].str.lower()


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

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


trainDF['comment_text'] = trainDF['comment_text'].str.replace(r'£|\$', 'dollars')# Replace money symbols with 'moneysymb'

# Replacing 10 digit phone numbers with 'phonenumber'
trainDF['comment_text'] = trainDF['comment_text'].str.replace(r'^\(?[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$','phonenumber')

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


trainDF['comment_text'] = trainDF['comment_text'].str.replace(r'[^\w\d\s]',' ') #removing punctuations

trainDF['comment_text'] = trainDF['comment_text'].str.replace(r'[\_]',' ') #removing underscore characters

trainDF['comment_text'] = trainDF['comment_text'].str.replace(r'\s+[a-zA-Z]\s+', ' ') #removing single characters

trainDF['comment_text'] = trainDF['comment_text'].str.replace(r'\s+', ' ') #removing whitespace between terms with a single

trainDF['comment_text'] = trainDF['comment_text'].str.replace(r'^\s+|\s+?$', ' ') #removing leading and trailing whitespace

```

```
trainDF.head()
```

	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	Stringlength
0	explanation why the edits made under my userna...	0	0	0	0	0	0	264
1	d aww he matches this background colour m seem...	0	0	0	0	0	0	112
2	hey man m really not trying to edit war it jus...	0	0	0	0	0	0	233
3	more can make any real suggestions on improve...	0	0	0	0	0	0	622
4	you sir are my hero any chance you remember wh...	0	0	0	0	0	0	67

```
import nltk
from nltk.corpus import stopwords,wordnet
```

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

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

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

```
trainDF['Cleaned_Stringlength'] = trainDF['comment_text'].str.len()
trainDF.head()
```

	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe	Stringlength	Cleaned_Stringlength
0	explanation edits made username hardcore metal...	0	0	0	0	0	0	264	164
1	aww match background colour seemingly stuck th...	0	0	0	0	0	0	112	83
2	hey man really trying edit war guy constantly ...	0	0	0	0	0	0	233	141
3	make real suggestion improvement wondered sect...	0	0	0	0	0	0	622	364
4	sir hero chance remember page	0	0	0	0	0	0	67	29

The train and test dataset contents were then converted into lowercase. Punctuations, unnecessary characters etc were removed, currency symbols, phone numbers, web urls, email addresses etc were replaced with single words. Tokens that contributed nothing to semantics of the messages were removed as Stop words. Finally retained tokens were lemmatized using WordNetLemmatizer().

The string lengths of original comments and the cleaned comments were then compared.

Data Inputs- Logic- Output Relationships

The comment tokens so vectorised using TfidVectorizer are input and classified as benign(0) or malignant(1) as output by classification models.

State the set of assumptions (if any) related to the problem under consideration

The comment content made available in Train and Test Dataset is assumed to be written in English Language in the standard Greco-Roman script. This is so that the Stopword package and WordNetLemmatizer can be effectively used.

Hardware and Software Requirements and Tools Used

Hardware Used:

- Processor: Intel core i3-2348M, 2.3GHz
- Physical Memory: 4.0GB
- GPU: NVIDIA GeForce 710M, 2GB . Software Used:

- Windows 10 Operating System□
- Anaconda Package and Environment Manager: Anaconda is a distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment. The distribution includes data science packages suitable for Windows and provides a host of tools and□ environment for conducting Data Analytical and Scientific works. Anaconda provides all the necessary Python packages and libraries for Machine learning projects.
- Jupyter Notebook: The Jupyter Notebook is an open-source web application that allows data scientists to create and share documents that integrate live code,

equations, computational output, visualizations, and other multimedia resources, along with explanatory text in a single document.□

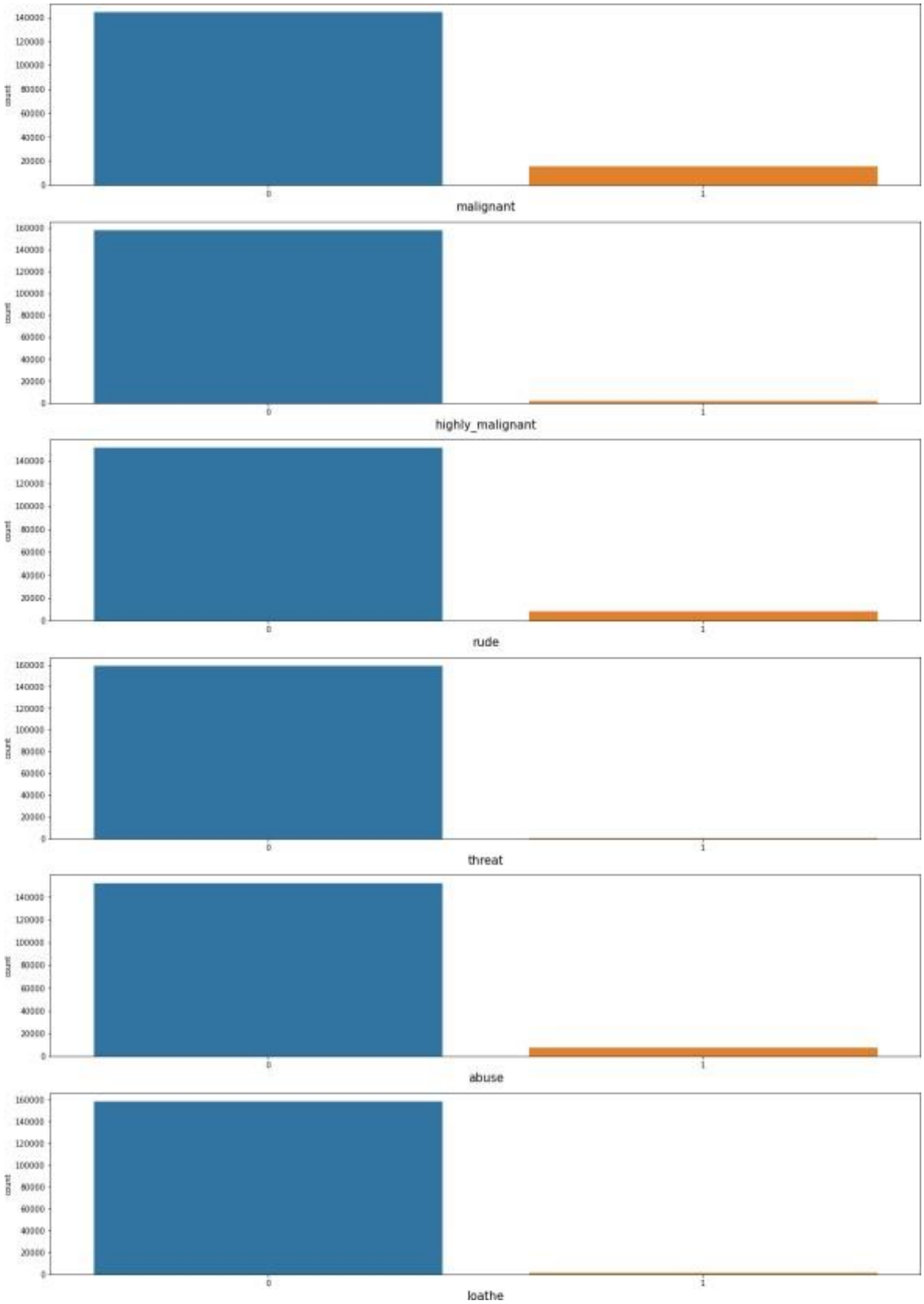
- Python3: It is open source, interpreted, high level language and provides great approach for object-oriented programming. It is one of the best languages used for Data Analytics And Data science projects/application. Python provides numerous libraries to deal with mathematics, statistics and scientific function.□
- Python Libraries used:□ ○ Pandas: For carrying out Data Analysis, Data Manipulation, Data Cleaning etc o Numpy: For performing a variety of operations on the datasets.

  ○ matplotlib.pyplot, Seaborn: For visualizing Data and various relationships between Feature and Label Columns ○ sklearn for Modelling Machine learning algorithms,
  Evaluation metrics, Data Transformation etc
  ○ imblearn.over_sampling: To employ SMOTE technique for balancing out the classes. ○ re, string: To perform regex operations ○ Wordcloud: For Data Visualization ○ NLTK: To use various Natural Language Processing Tools.

Exploratory Data Analysis Visualizations

Barplots, Countplots,Distplots,WordClouds were used to visualise the data of all the columns and their relationships with Target variable.

Analyzing the Feature Columns



From the graphs about it is observed that majority of the comments are benign.

Unprocessed vs Cleaned string lengths

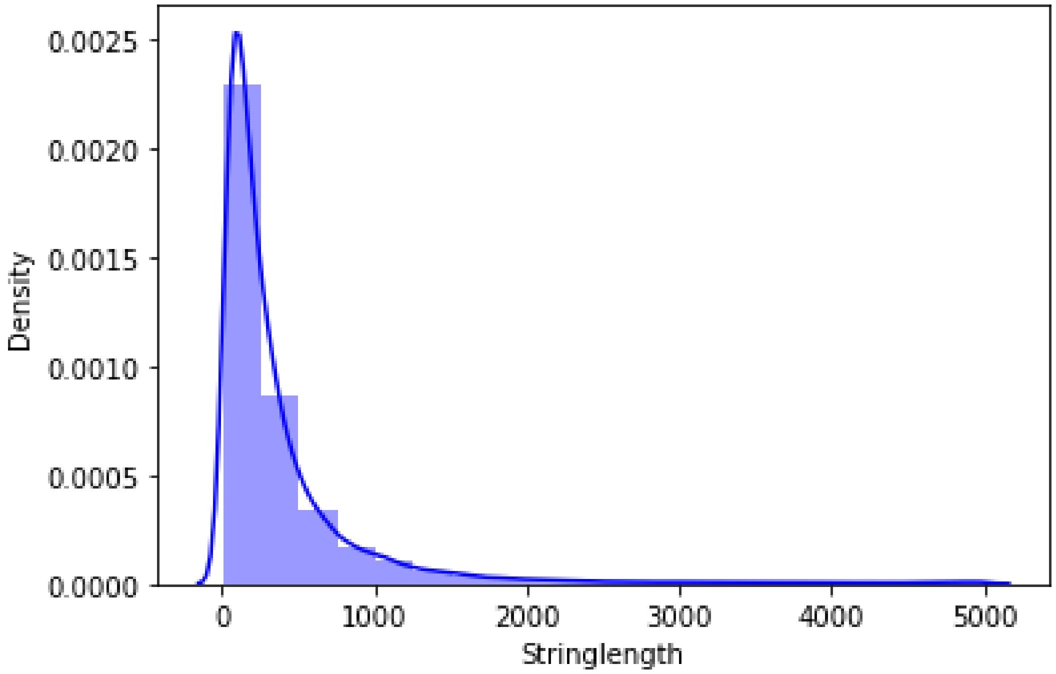


Figure 3 String Length of unprocessed comments

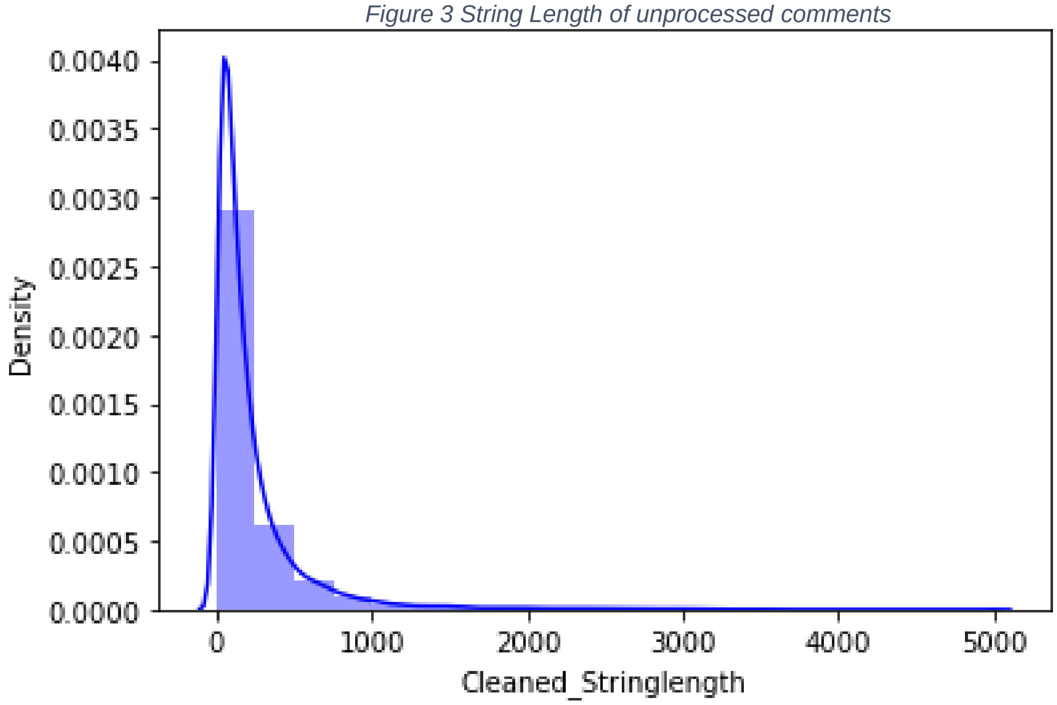
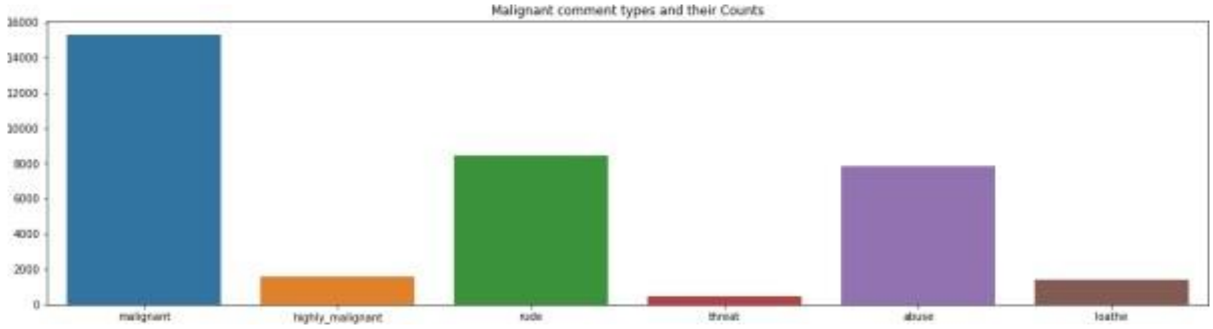


Figure 4 Cleaned Comments String Length

Above graphs show that the string length of comments was drastically brought down after processing.



The above graph shows the composition of toxic comments, of which majority are malignant followed by rude comments, abusive comments, highly malignant comments, hateful comments and threats.

Word Clouds of the most frequent words under various categories of Malignant Comments

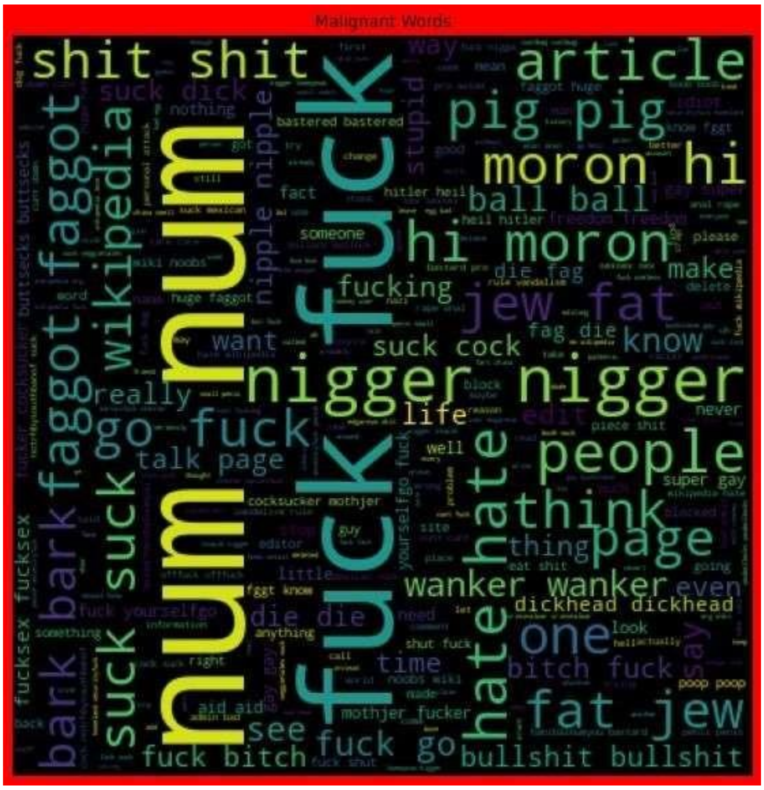


Figure 5 Malignant Words

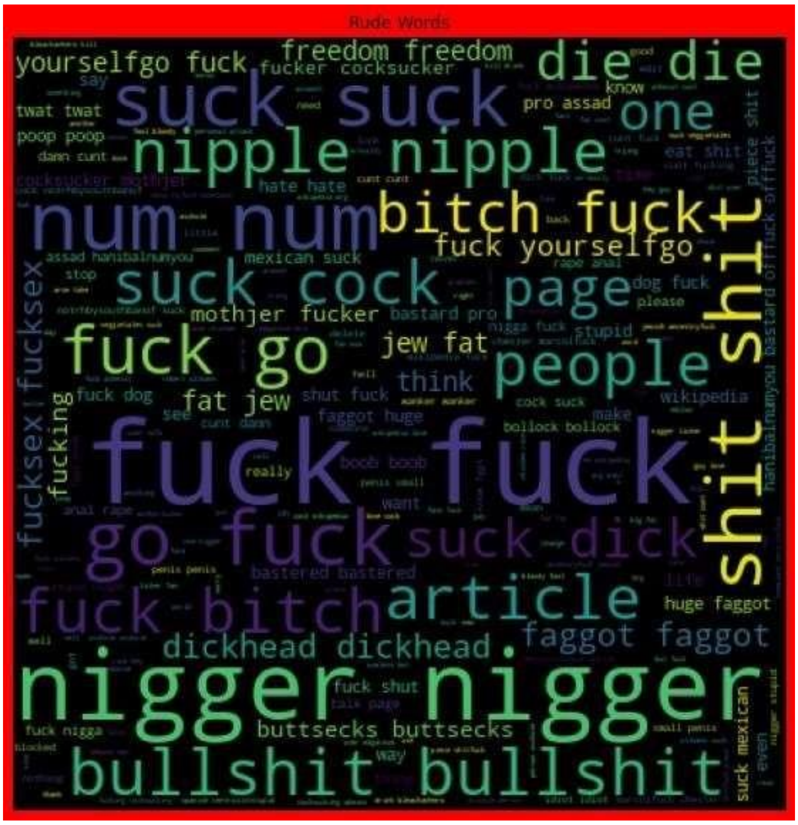


Figure 6 Rude Words

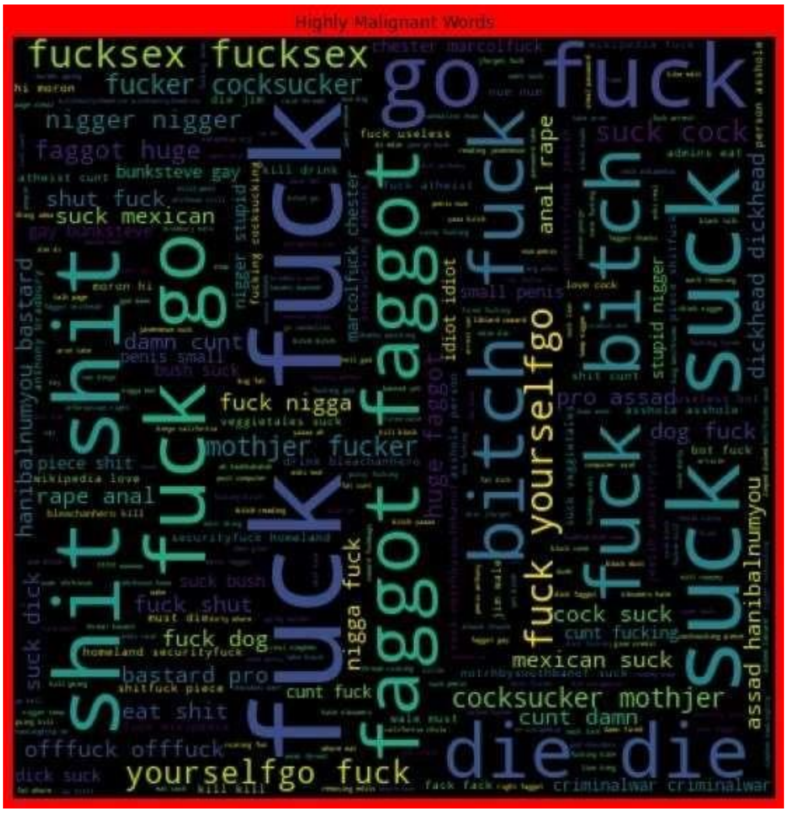


Figure 7 Highly Malignant Words

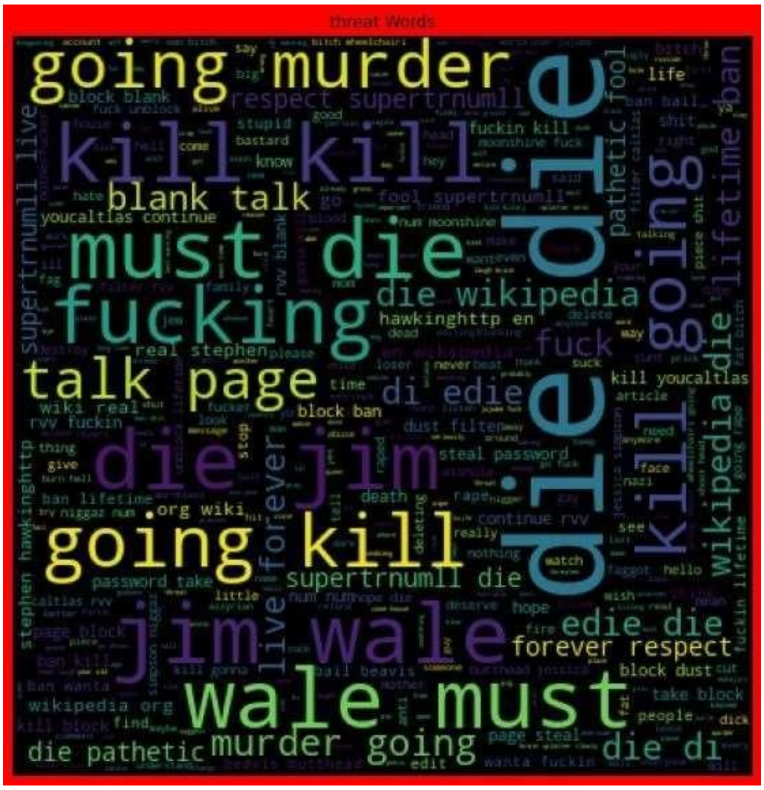


Figure 8 Threat Words

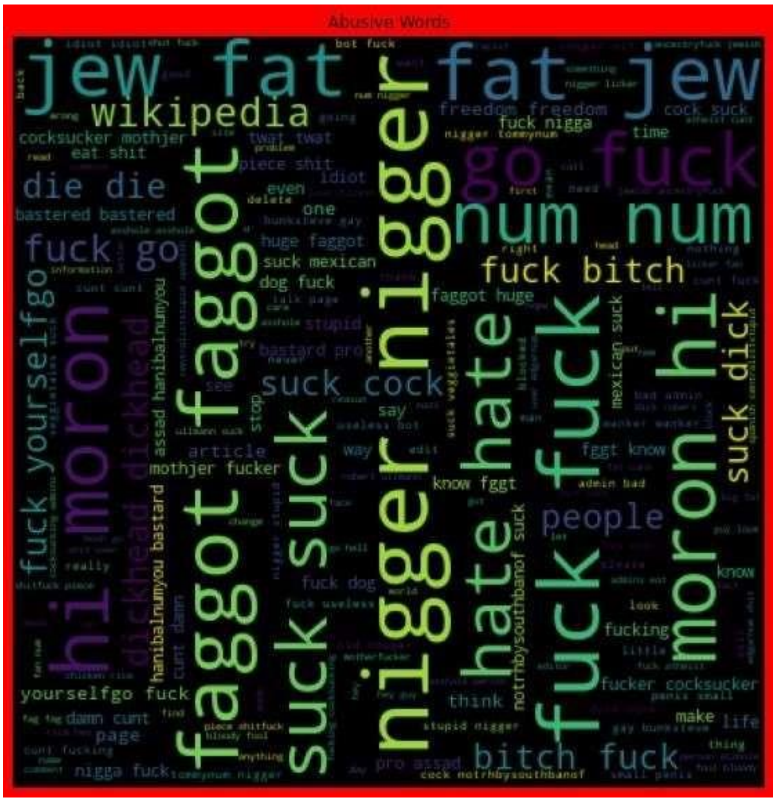


Figure 9 Abusive Words

Hateful Words
nigger spic one
spanish centraliststupid
page
bunksteve gay
ancestryfuck jewish
wikipedia org
gay bunksteve
fan num
centraliststupid spanish
jew minority
racist
nlnumers hate
nigger nigger
fucking hate
think
keep nigger
jewish ancestryfuck
article
mexican suck
num nigger
suck dick
jforget fuck
spic jew hate nlnumers
kill drink
fuck nigga
nigger tommynum
drink bleachanhero
bitch
utc cody
gay utc
fat jew
faggot huge
num num
know
cody gay
tommynum nigger
nigga eat
jew fat
huge faggot
cunt cunt
mitt romney
fuck faggot
die di wikipedia
licker fan
nigger keep
edit
suck mexican
die bitch
bleachanhero kill
nigga fuck
romney user
people
nazi
piece shit
die die
nigger licker
stupid nigger
fucking
nigger stupid
homosexual

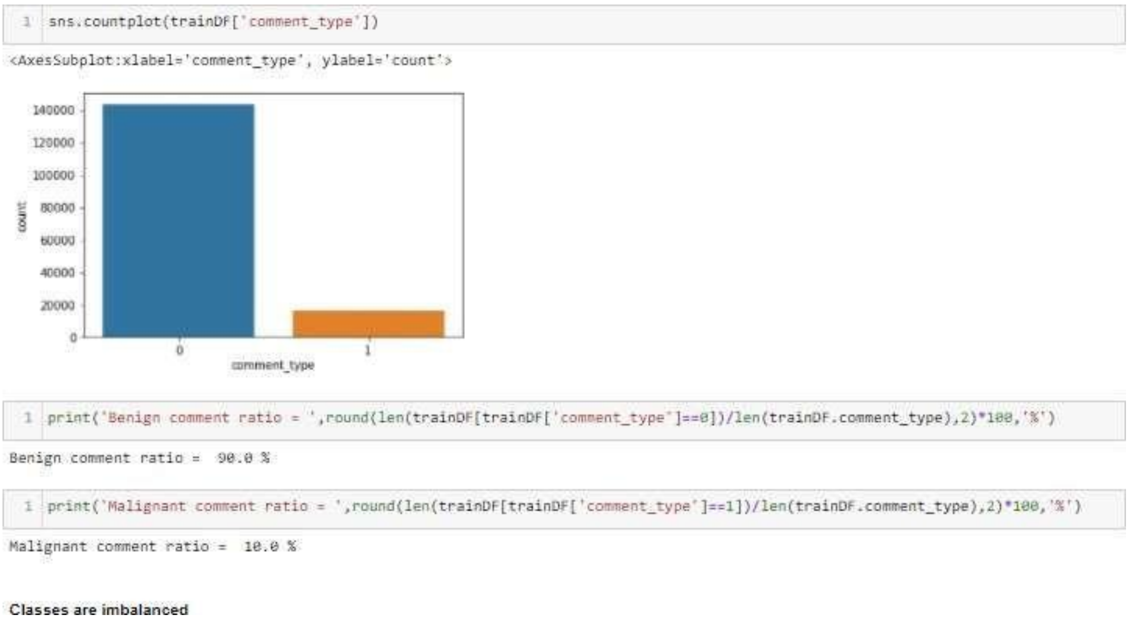
Figure 10 Hateful Words

Feature Engineering

The comments data could belong to more than one label simultaneously(rude comments are at the same time malignant and in some cases can also be deemed hateful, abusive comments are hateful and can be highly malignant at the same time, threats are highly malignant too etc.)

Since each of the categories had very small data available to work with, a new column: „comment_type‟ was created which only had binary classes: 0 which represented all the benign comments and 1 which represented all the comments which fell under malignant,highly malignant,abusive,hateful,rude,threat features. This column acted as Target Label column for malignant comment classification.

Visualising data in Target column



```
sns.countplot(trainDF['comment_type'])
```

```
<AxesSubplot:xlabel='comment_type', ylabel='count'>
```

```
print('Benign comment ratio = ',round(len(trainDF[trainDF['comment_type']==0])/len(trainDF.comment_type),2)*100,'%')
```

```
Benign comment ratio =  90.0 %
```

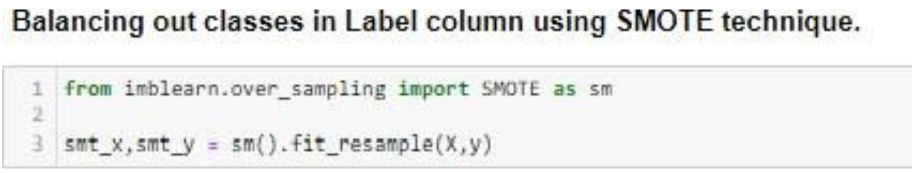
```
print('Malignant comment ratio = ',round(len(trainDF[trainDF['comment_type']==1])/len(trainDF.comment_type),2)*100,'%')
```

```
Malignant comment ratio =  10.0 %
```

Classes are imbalanced

The classes appear to be imbalanced with 90% of comments being benign (0) and only 10% being malignant (1).

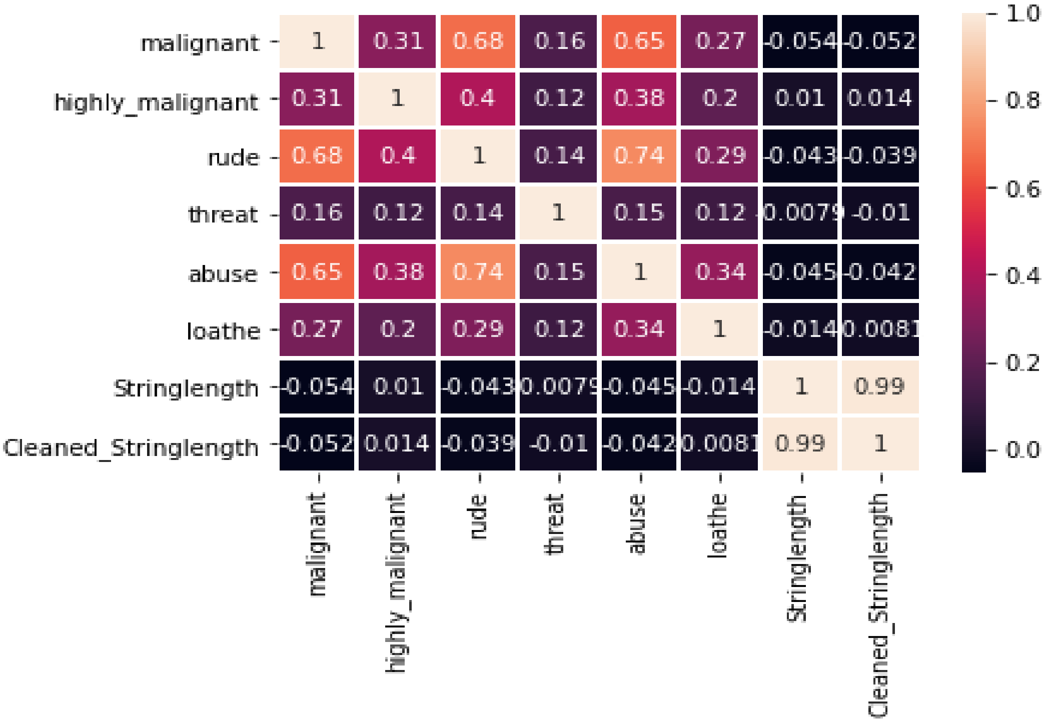
Smote Technique was used to balance out the classes

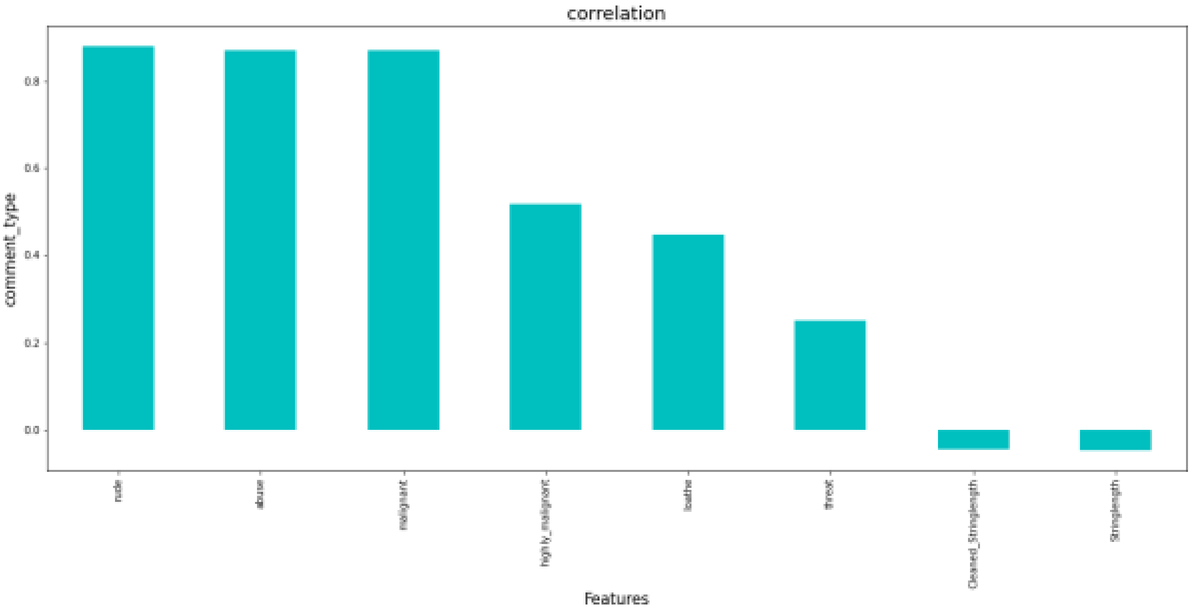


Balancing out classes in Label column using SMOTE technique.

```
from imblearn.over_sampling import SMOTE as sm

smt_x,smt_y = sm().fit_resample(X,y)
```

Finding Correlation





From the graphs above it is observed that columns: Rude,Abuse, Malignant have highest positive correlation with comment_type.

Model/s Development and Evaluation

Identification of possible problem-solving approaches (methods)

The model algorithms used were as follows:

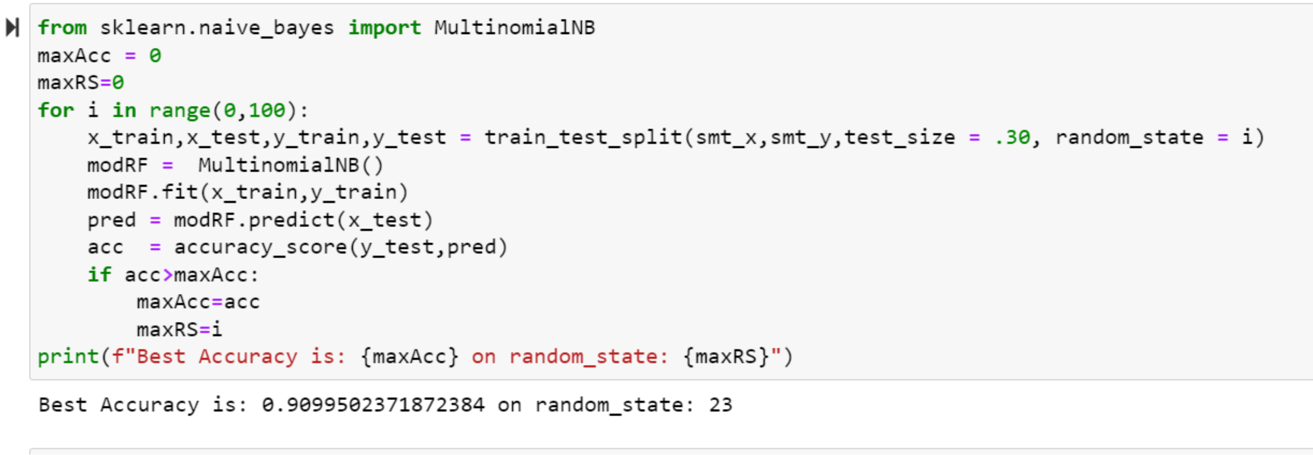
- Logistic Regression: It is a classification algorithm used to find the probability of event success and event failure. It is used when the dependent variable is binary(0/1, True/False, Yes/No) in nature. It supports categorizing data into discrete classes by studying the relationship from a given set of labelled data. It learns a linear relationship from the given dataset and then introduces a non-linearity in the form of the Sigmoid function. It not only provides a measure of how appropriate a predictor(coefficient size)is, but also its direction of association (positive or negative).
- Multinomial Naïve Bayes Classifier: Multinomial Naive Bayes algorithm is a probabilistic learning method that is mostly used in Natural Language Processing (NLP). The algorithm is based on the Bayes theorem. It calculates the probability of each tag for a given sample and then gives the tag with the highest probability as output.
- XGBClassifier: XGBoost uses decision trees as base learners; combining many weak learners to make a strong learner. As a result it is referred to as an ensemble learning method since it uses the output of many models in the final prediction. It uses the power of parallel processing and supports regularization.
- RandomForestClassifier: A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. A random forest produces good predictions that can be understood easily. It reduces overfitting and can handle large datasets efficiently. The random forest algorithm provides a higher level of accuracy in predicting outcomes over the decision tree algorithm.
- Complement Naïve Bayes Classifier: Complement Naive Bayes is somewhat an adaptation of the standard Multinomial Naive Bayes algorithm. Complement Naive Bayes is particularly suited to work with imbalanced datasets. In complement Naive Bayes, instead of

calculating the probability of an item belonging to a certain class, we calculate the probability of the item belonging to all the classes.

- Passive Aggressive Classifier: Passive-Aggressive algorithms do not require a learning rate and are called so because if the prediction is correct, keep the model and do not make any changes. i.e., the data in the example is not enough to cause any changes in the model. If the prediction is incorrect, make changes to the model. i.e., some change to the model may correct it.□

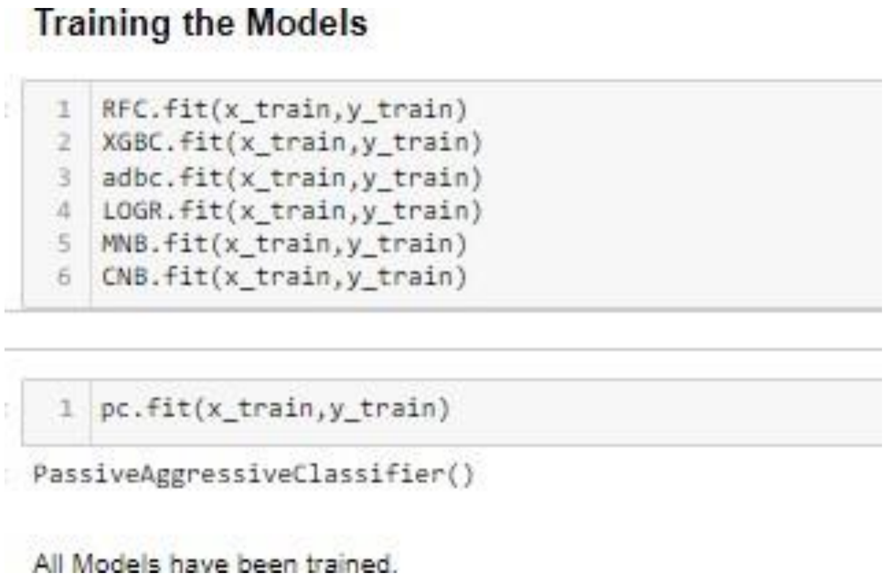
- AdaBoost Classifier: The basis of this algorithm is□ the Boosting main core: give more weight to the misclassified observations. the meta-learner adapts based upon the results of the weak classifiers, giving more weight to the misclassified observations of the last weak learner. The individual learners can be weak, but as long as the performance of each weak learner is better than random guessing, the final model can converge to a strong learner (a learner not influenced by outliers and with a great generalization power, in order to have strong performances on unknown data).

Best Random state was found to be 23



```
from sklearn.naive_bayes import MultinomialNB
maxAcc = 0
maxRS=0
for i in range(0,100):
    x_train,x_test,y_train,y_test = train_test_split(smt_x,smt_y,test_size = .30, random_state = i)
    modRF =  MultinomialNB()
    modRF.fit(x_train,y_train)
    pred = modRF.predict(x_test)
    acc  = accuracy_score(y_test,pred)
    if acc>maxAcc:
        maxAcc=acc
        maxRS=i
print(f"Best Accuracy is: {maxAcc} on random_state: {maxRS}")
```

```
Best Accuracy is: 0.9099502371872384 on random_state: 23
```



Training the Models

```
RFC.fit(x_train,y_train)
XGBC.fit(x_train,y_train)
adbc.fit(x_train,y_train)
LOGR.fit(x_train,y_train)
MNB.fit(x_train,y_train)
CNB.fit(x_train,y_train)
```

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

```
PassiveAggressiveClassifier()
```

All Models have been trained.

Analyzing Accuracy of The Models

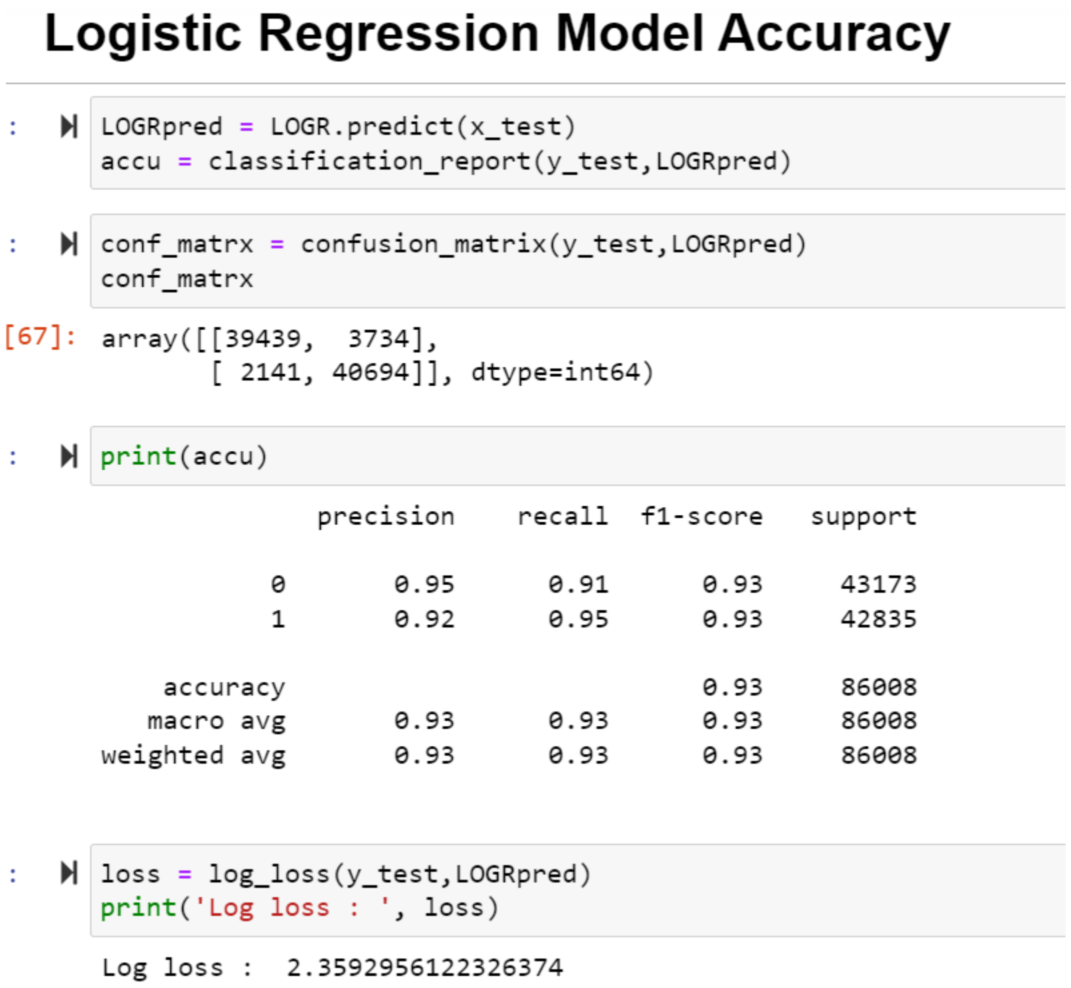
Classification Report consisting of Precision,Recall, Support and F1-score were the metrics used to evaluate the Model Performance.

Precision is defined as the ratio of true positives to the sum of true and false positives. Recall is defined as the ratio of true positives to the sum of true positives and false negatives.The F1 is the weighted harmonic mean of precision and recall. The closer the value of the F1 score is to 1.0, the better the expected performance of the model is. Support is the number of actual occurrences of the class in the dataset. It doesn‟t vary between models; it just diagnoses the performance evaluation process.

Log Loss quantifies the accuracy of a classifier by penalizing false classifications.

Model Cross Validation

Cross validation is a technique for assessing how the statistical analysis generalises to an independent data set.It is a technique for evaluating machine learning models by training several models on subsets of the available input data and evaluating them on the complementary subset of the data. Using cross-validation, there are high chances that we can detect over-fitting with ease. Model Cross Validation scores were then



Logistic Regression Model Accuracy

```
LOGRpred = LOGR.predict(x_test)
accu = classification_report(y_test,LOGRpred)
```

```
conf_matrx = confusion_matrix(y_test,LOGRpred)
conf_matrx
```

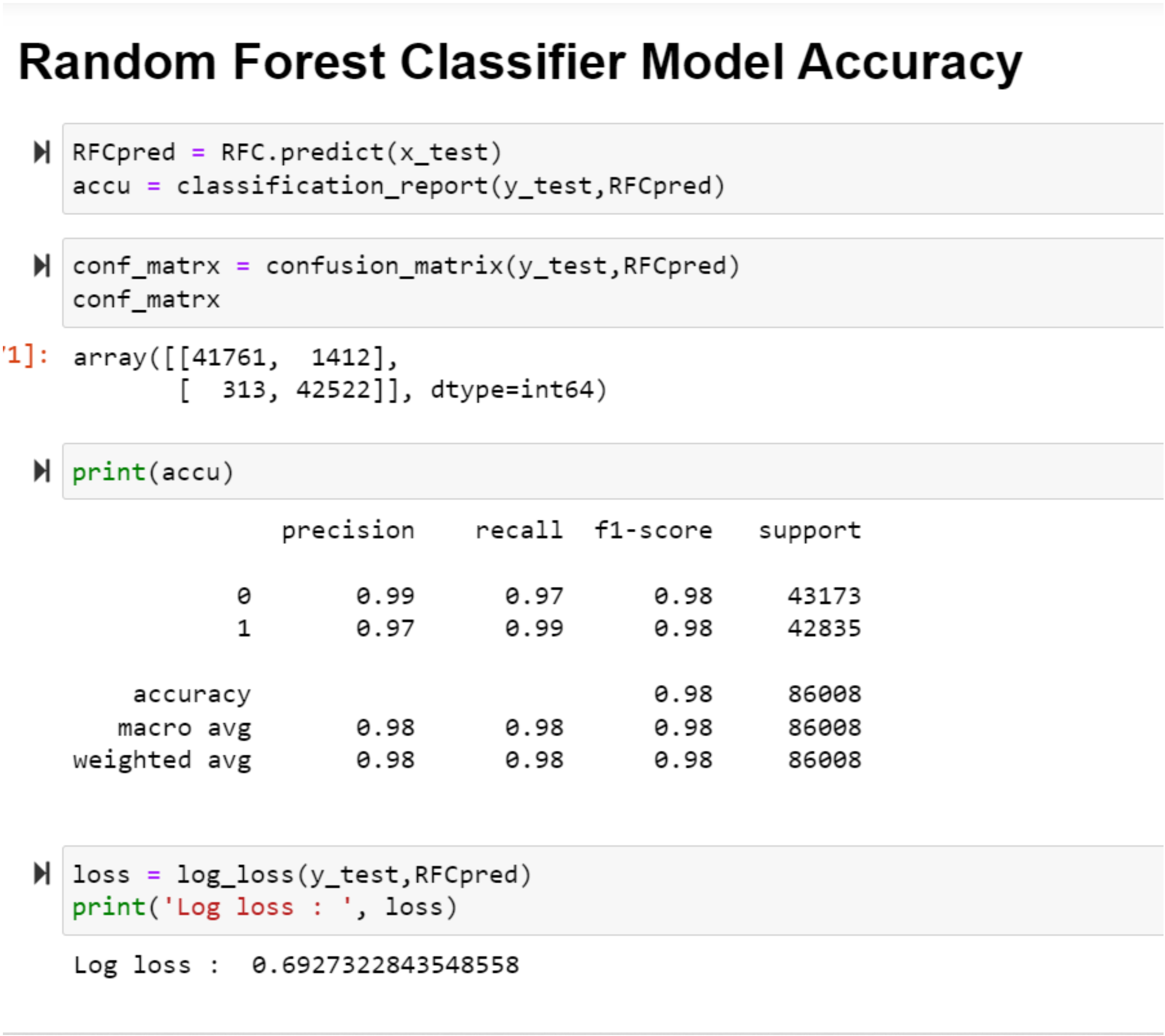
```
[67]: array([[39439,  3734],
              [ 2141, 40694]], dtype=int64)
```

```
print(accu)
```

```
              precision    recall  f1-score   support

           0       0.95      0.91      0.93     43173
           1       0.92      0.95      0.93     42835

    accuracy                           0.93     86008
   macro avg       0.93      0.93      0.93     86008
weighted avg       0.93      0.93      0.93     86008
```

```
loss = log_loss(y_test,LOGRpred)
print('Log loss : ', loss)
```

```
Log loss :  2.3592956122326374
```



Random Forest Classifier Model Accuracy

```
RFCpred = RFC.predict(x_test)
accu = classification_report(y_test,RFCpred)
```

```
conf_matrx = confusion_matrix(y_test,RFCpred)
conf_matrx
```

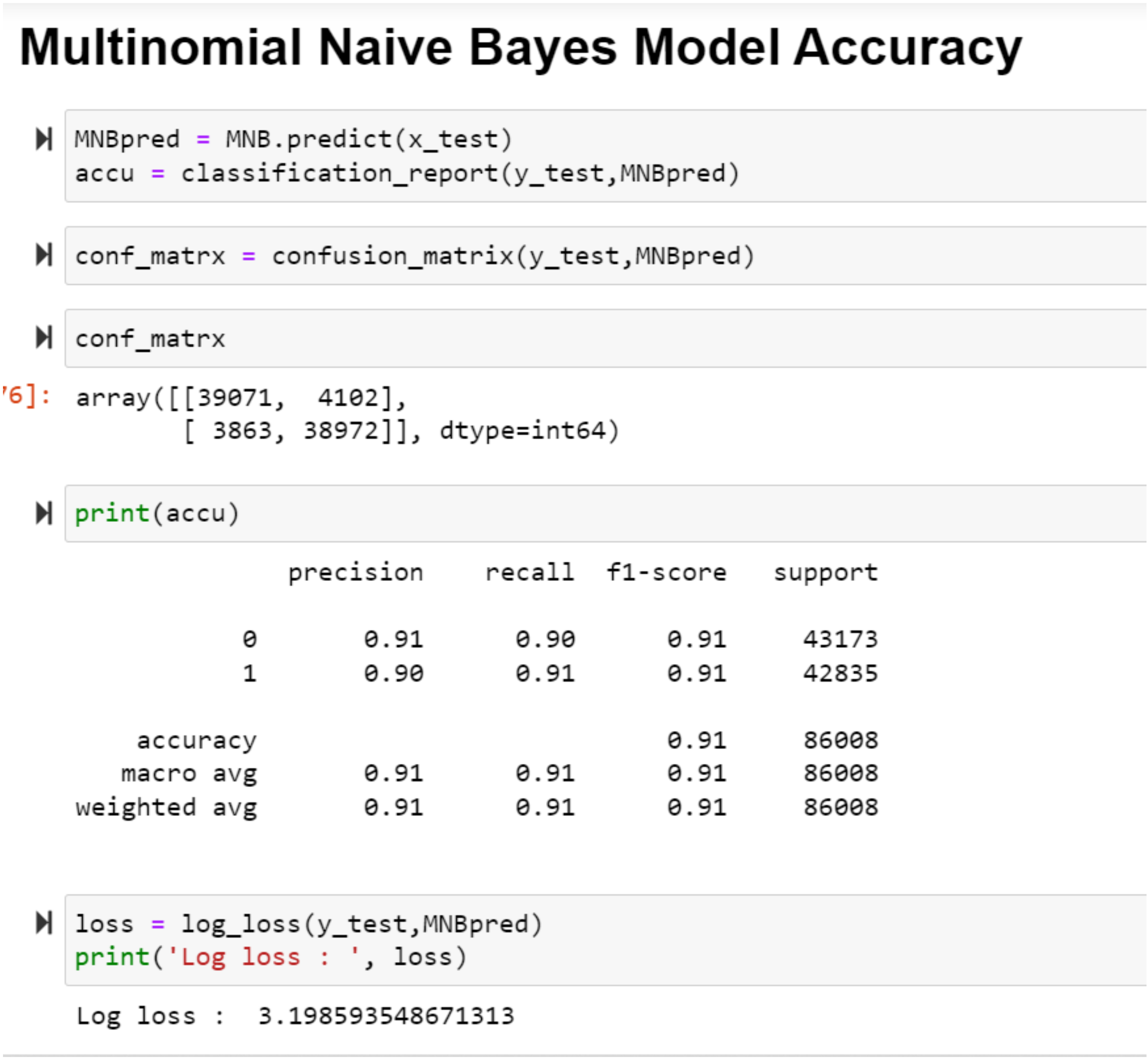
```
'1]: array([[41761,  1412],
            [  313, 42522]], dtype=int64)
```

```
print(accu)
```

```
              precision    recall  f1-score   support

           0       0.99      0.97      0.98     43173
           1       0.97      0.99      0.98     42835

    accuracy                           0.98     86008
   macro avg       0.98      0.98      0.98     86008
weighted avg       0.98      0.98      0.98     86008
```

```
loss = log_loss(y_test,RFCpred)
print('Log loss : ', loss)
```

```
Log loss :  0.6927322843548558
```



Multinomial Naive Bayes Model Accuracy

```
MNBpred = MNB.predict(x_test)
accu = classification_report(y_test,MNBpred)
```

```
conf_matrx = confusion_matrix(y_test,MNBpred)
```

```
conf_matrx
```

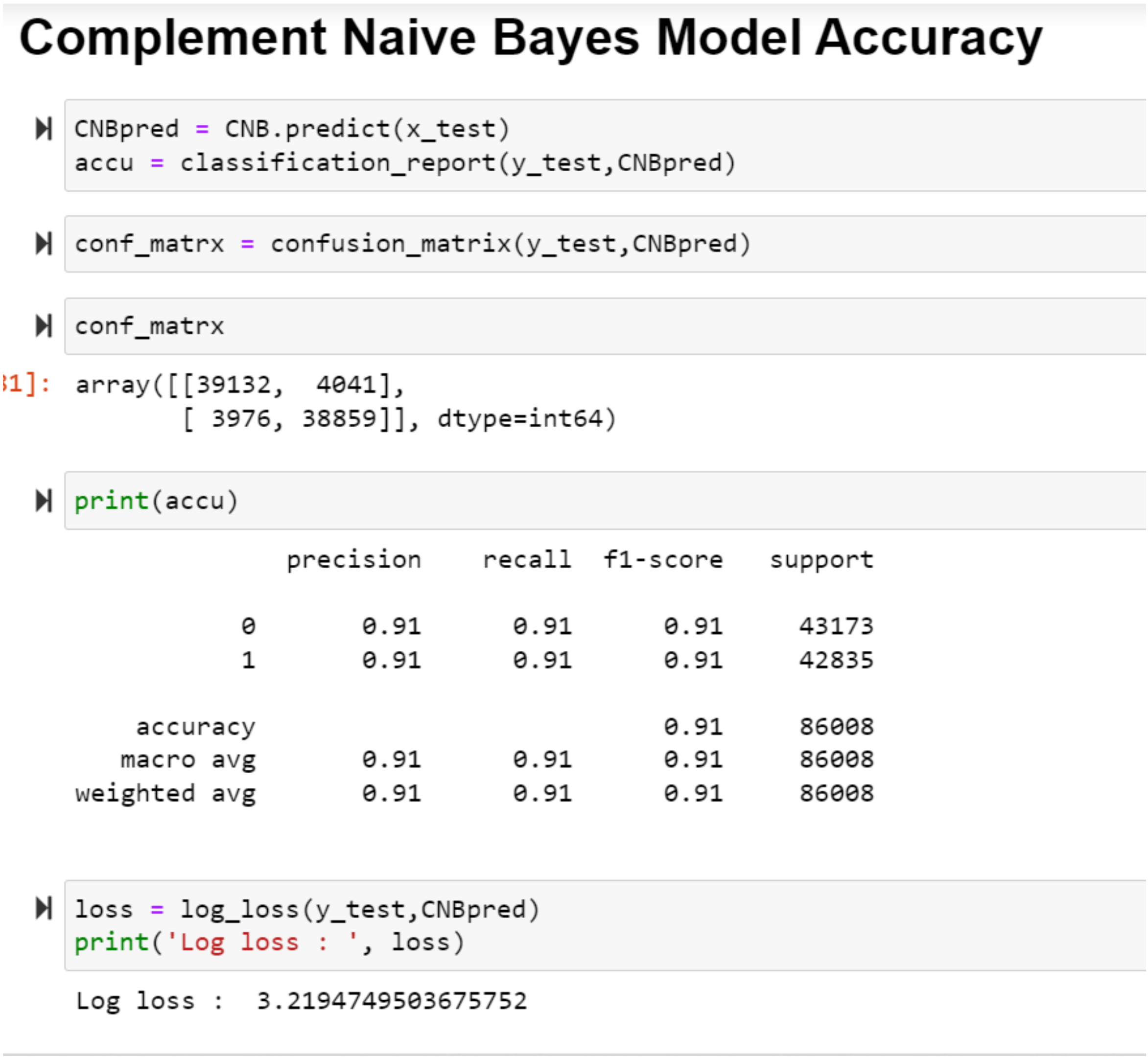
```
'6]: array([[39071,  4102],
            [ 3863, 38972]], dtype=int64)
```

```
print(accu)
```

```
              precision    recall  f1-score   support

           0       0.91      0.90      0.91     43173
           1       0.90      0.91      0.91     42835

    accuracy                           0.91     86008
   macro avg       0.91      0.91      0.91     86008
weighted avg       0.91      0.91      0.91     86008
```

```
loss = log_loss(y_test,MNBpred)
print('Log loss : ', loss)
```

```
Log loss :  3.198593548671313
```



Complement Naive Bayes Model Accuracy

```
CNBpred = CNB.predict(x_test)
accu = classification_report(y_test,CNBpred)
```

```
conf_matrx = confusion_matrix(y_test,CNBpred)
```

```
conf_matrx
```

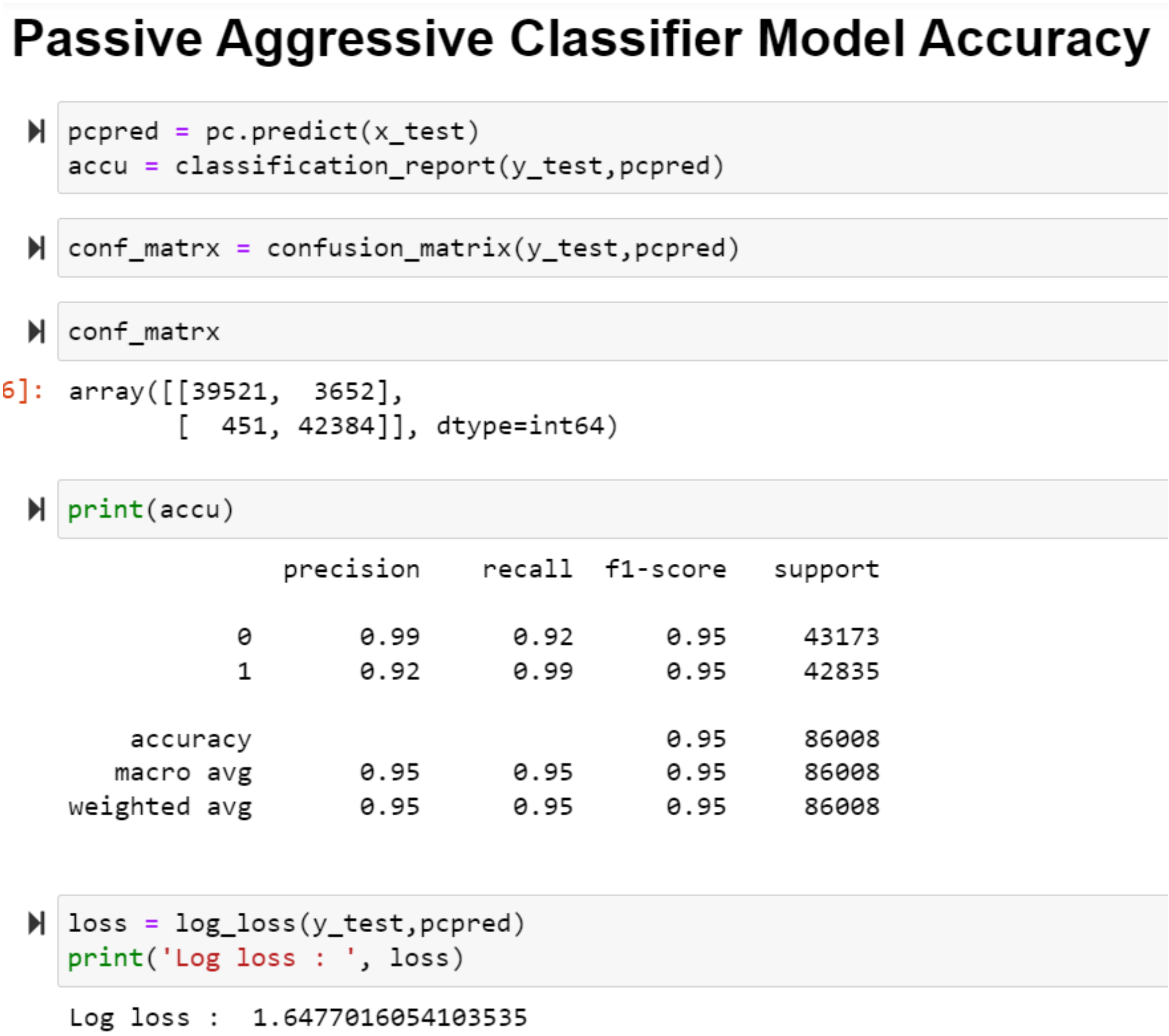
```
1]: array([[39132,  4041],
           [ 3976, 38859]], dtype=int64)
```

```
print(accu)
```

```
              precision    recall  f1-score   support

           0       0.91      0.91      0.91     43173
           1       0.91      0.91      0.91     42835

    accuracy                           0.91     86008
   macro avg       0.91      0.91      0.91     86008
weighted avg       0.91      0.91      0.91     86008
```

```
loss = log_loss(y_test,CNBpred)
print('Log loss : ', loss)
```

```
Log loss :  3.2194749503675752
```



Passive Aggressive Classifier Model Accuracy

```
pcpred = pc.predict(x_test)
accu = classification_report(y_test,pcpred)
```

```
conf_matrx = confusion_matrix(y_test,pcpred)
```

```
conf_matrx
```

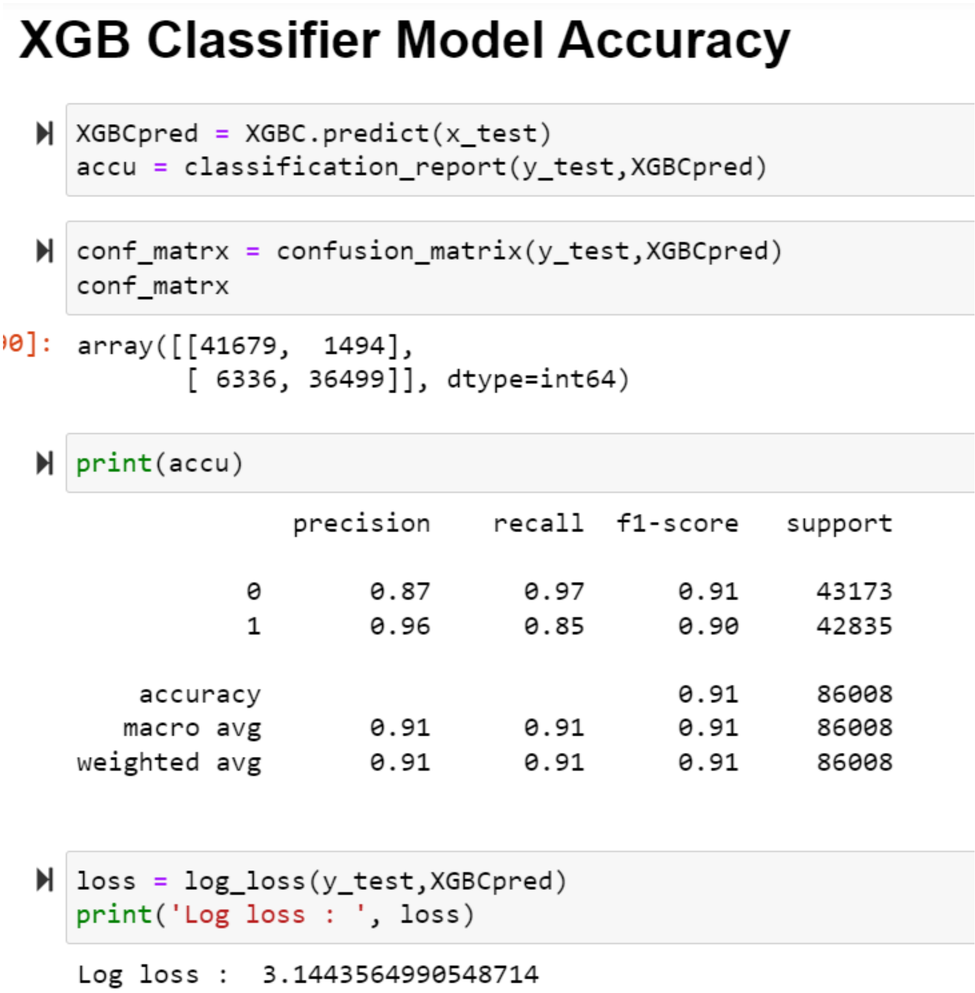
```
6]: array([[39521,  3652],
           [  451, 42384]], dtype=int64)
```

```
print(accu)
```

```
              precision    recall  f1-score   support

           0       0.99      0.92      0.95     43173
           1       0.92      0.99      0.95     42835

    accuracy                           0.95     86008
   macro avg       0.95      0.95      0.95     86008
weighted avg       0.95      0.95      0.95     86008
```

```
loss = log_loss(y_test,pcpred)
print('Log loss : ', loss)
```

```
Log loss :  1.6477016054103535
```



XGB Classifier Model Accuracy

```
XGBCpred = XGBC.predict(x_test)
accu = classification_report(y_test,XGBCpred)
```

```
conf_matrx = confusion_matrix(y_test,XGBCpred)
conf_matrx
```

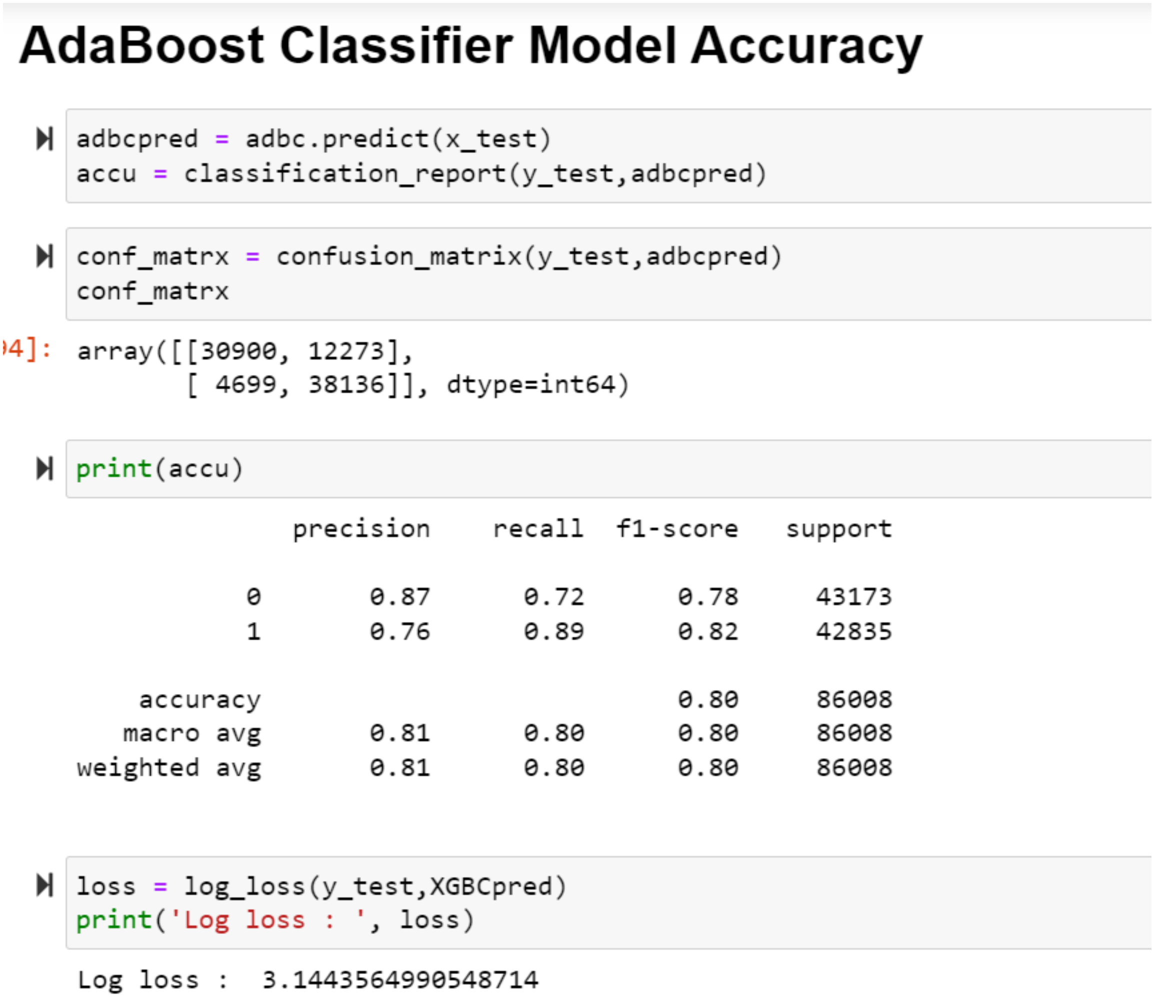
```
0]: array([[41679,  1494],
           [ 6336, 36499]], dtype=int64)
```

```
print(accu)
```

```
              precision    recall  f1-score   support

           0       0.87      0.97      0.91     43173
           1       0.96      0.85      0.90     42835

    accuracy                           0.91     86008
   macro avg       0.91      0.91      0.91     86008
weighted avg       0.91      0.91      0.91     86008
```

```
loss = log_loss(y_test,XGBCpred)
print('Log loss : ', loss)
```

```
Log loss :  3.1443564990548714
```



AdaBoost Classifier Model Accuracy

```
adbcpred = adbc.predict(x_test)
accu = classification_report(y_test,adbcpred)
```

```
conf_matrx = confusion_matrix(y_test,adbcpred)
conf_matrx
```

```
4]: array([[30900, 12273],
           [ 4699, 38136]], dtype=int64)
```

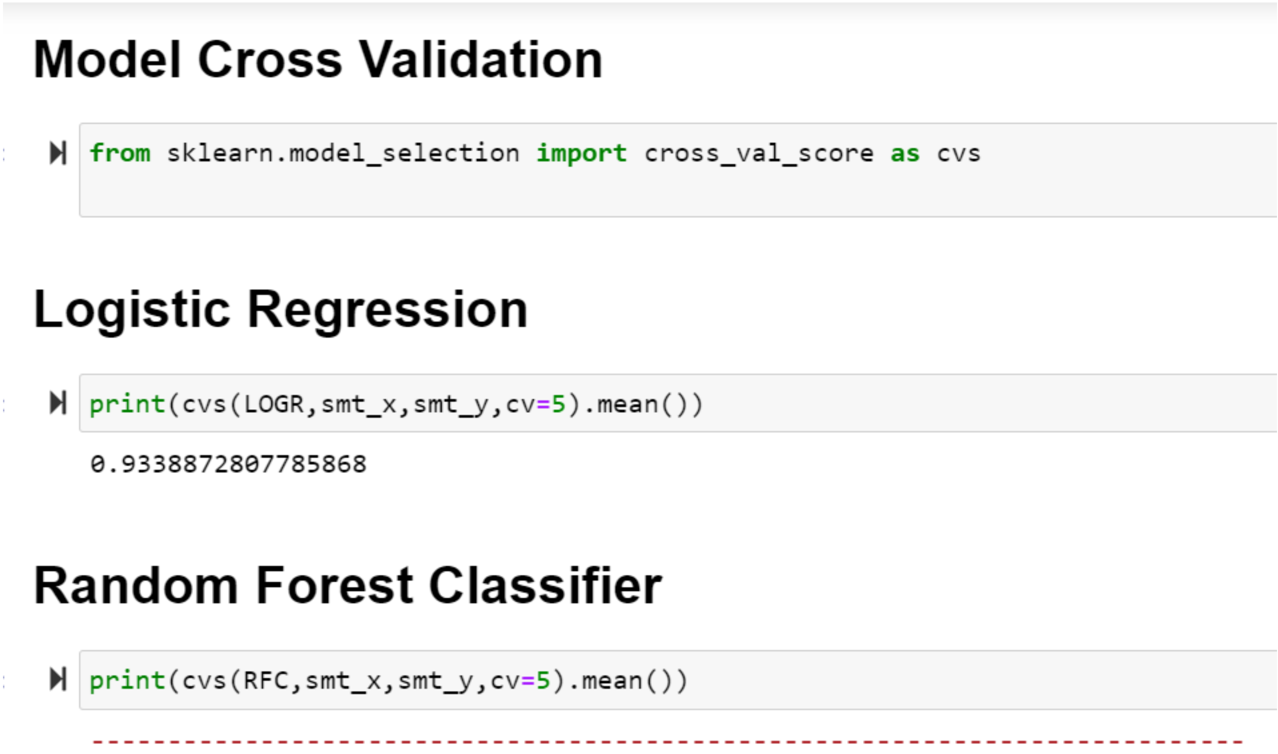
```
print(accu)
```

```
              precision    recall  f1-score   support

           0       0.87      0.72      0.78     43173
           1       0.76      0.89      0.82     42835

    accuracy                           0.80     86008
   macro avg       0.81      0.80      0.80     86008
weighted avg       0.81      0.80      0.80     86008
```

```
loss = log_loss(y_test,XGBCpred)
print('Log loss : ', loss)
```

```
Log loss :  3.1443564990548714
```

obtained for assessing how the statistical analysis generalises to an independent data set. The models were evaluated by training several models on subsets of the available input data and evaluating them on the complementary subset of the data.



Model Cross Validation

```
from sklearn.model_selection import cross_val_score as cvs
```

Logistic Regression

```
print(cvs(LOGR,smt_x,smt_y,cv=5).mean())
```

```
0.9338872807785868
```

Random Forest Classifier

```
print(cvs(RFC,smt_x,smt_y,cv=5).mean())
```

```
---------------------------------------------------------------------------
```