MALIGNANT COMMENT CLASSIFICATION

Submitted by:
LAKSHITA KAIN
DATA SCIENCE INTERN

ACKNOWLEDGMENT

I would like to express my deepest appreciation to all those who provided me the possibility to complete this report. A special gratitude I give DataTrained and FlipRobo, whose contribution in stimulating suggestions and encouragement, helped me to coordinate my project. Furthermore I would also like to acknowledge with much appreciation the crucial role of the our SME of Mr Keshav Bansal.

1. INTRODUCTION

Internet offers a global ecosystem for social interactions. Modern life revolves around the network, with its status updates, news feeds, comment chains, political advocacy, omnipresent reviews, rankings and ratings.

People have freedom of speech while keeping their anonymity. Someone posts something, and people start commenting negative feedbacks which is far from constructive, it's demotivating and lead to discouragement. Sometimes, it's plain discrimination on the basis of body(body-shaming), social background or just on the basis of race or community. The people who relish this online freedom are called trolls, a term that originally came from a fishing method online thieves use to find victims.

A Pew Research Center survey published found that 70% of 18-to-24-year-olds who use the Internet had experienced harassment, and 26% of women that age said they'd been stalked online. A 2014 study published in the psychology journal Personality and Individual Differences found that the approximately 5% of Internet users who self-identified as trolls scored extremely high in the dark tetrad of personality traits: narcissism, psychopathy, Machiavellianism and, especially, sadism. The pandemic and indirectly the sudden increase in unemployment have made things even worse. Trolls have more time to hide behind the screens and bring people down.

1.1 PROBLEM DEFINITION:

Sentiment Analysis is the task of analysing people's opinions in textual data (e.g., product reviews, movie reviews, or tweets), and extracting their polarity and viewpoint. The task can be cast as either a binary or a multi-class problem. Binary sentiment analysis classifies texts into positive and negative classes, while multi-class sentiment analysis classifies texts into fine-grained labels or multi-level intensities.

1.2 OBJECTIVE:

The goal of this project is to use deep learning techniques to identify the level toxicity of a comment which could be used to help deter users from posting potentially hurtful comments, engage in more sophisticated arguments and make internet a safer place.

Our dataset consists of 6 labels namely highly malignant, malignant, abuse, threat, loathe and rude. This project aims to implement various Machine Learning algorithms and deep learning algorithms like Multilayer perceptron(MLP), Long Short Term Memory Networks, Multinomial Naïve Bayes, Logistic Regression, Random Forest Classifier , Linear SVC and Adaptive Boosting.

2. Literature Review

2.1 Machine Learning Algorithms

Machine learning-based systems are growing in popularity in research applications in most disciplines. Considerable decision-making knowledge from data has been acquired in the broad area of machine learning, in which decision-making tree-based ensemble techniques are recognized for supervised classification problems. Thus, classification is
an essential form of data analysis in data mining that formulates models while describing significant data classes. Accordingly, such models estimate categorical class labels, which can provide users with an enhanced understanding of the data at large resulted in significant advancements in classification accuracy. Motivated by the preceding literature, we evaluated a large number of machine learning algorithms in our work on credit risk in micro-lending. A set of algorithms that performed well in numerical experiments with real data is explained in more details below.

2.1.2 Logistic Regression

Logistic regression is named for the function used at the core of the method, the logistic function.

The logistic function, also called the sigmoid function was developed by statisticians to describe properties of population growth in ecology, rising quickly and maxing out at the carrying capacity of the environment. It's an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits.

1 / (1 + e^-value)

Where e is the base of the natural logarithms (Euler's number or the EXP() function in your spreadsheet) and value is the actual numerical value that you want to transform. Below is a plot of the numbers between -5 and 5 transformed into the range 0 and 1 using the logistic function.

Input values (x) are combined linearly using weights or coefficient values (referred to as the Greek capital letter Beta) to predict an output value (y). A key difference from linear regression is that the output value being modeled is a binary values (0 or 1) rather than a numeric value.

y = e^(b0 + b1*x) / (1 + e^(b0 + b1*x))

Where y is the predicted output, b0 is the bias or intercept term and b1 is the coefficient for the single input value (x). Each column in your input data has an associated b coefficient (a constant real value) that must be learned from your training data.

2.1.2 AdaBoost

Adaptive boosting (AdaBoost) is an ensemble algorithm incorporated by Freund and

Schapire (1997), which trains and deploys trees in time series. Since then, it evolved as a popular

boosting technique introduced in various research disciplines. It merges a set of weak classifiers to build and boost a robust classifier that will improve the decision tree's performance and improve accuracy.
The mathematical presentation of the AdaBoost classifier at a high level is described of the following:

Consider that training data (x1 and y1), . . . , (xm and ym) are the xi 2 X, yi 2 {1, +1}. Then the parameters of AdaBoost classifier are initialized: D1(i) = 1/m for i = 1, . . . , m. For t = 1, . . . , T:
• Train weak learner using distribution Dt.
• Attain weak hypothesis ht: X ! {1, +1}.
• Aim: select ht with low weighted error:

$$\epsilon = \Pr_i \sim D_t[h_t(x_i) \neq y_i]$$

Select $\alpha_t = 1$

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$$

Hence, for i = 1, . . . ,m:

$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Zt is a normalization factor (chosen so that Dt + 1 will be a distribution).
Which generates an output of the final hypothesis as following:
H(x) = sign
.
Therefore, H is estimated as a weighted majority vote of the weak hypotheses ht while assigning each hypothesis a weight *a*t (Schapire et al. 2013).

2.1.3 Random Forest Classifier

The random forest classifier is an ensemble method algorithm of decision trees wherein each tree depends on randomly selected samples trained independently, with a similar distribution for all the trees in the forest (Breiman 2001). Hence, a random forest is a classifier incorporating a collection of tree-structured classifiers that decrease overfitting, resulting in an increase in the overall accuracy (Geurts et al. 2006). As such, random forest's accuracy differs based on the strength of each tree classifier and their dependencies.

$$r_N(X,\beta) = \frac{\sum_{i=1}^{N} y_i 1_{x_j} \in A_N(X,\beta)}{\sum_{i=1}^{N} 1_{x_j} \in A_N(X,\beta)} 1_{L_N}$$

where LN = åN
i=1 1xj 2 AN(x, *b*) 6= 0. We can achieve the estimate of rN with respect to the parameter *b* by taking the expectation of rN (Addo et al. 2018).

2.1.4 Linear SVC

The objective of a Linear SVC (Support Vector Classifier) is to fit to the data provided and return a "best fit" hyperplane that divides, or categorizes the data.
Similar to SVC with parameter kernel='linear', but implemented in terms of liblinear rather than libsvm, so it has more flexibility in the choice of penalties and loss functions and should scale better to large numbers of samples.

This class supports both dense and sparse input and the multiclass support is handled according to a one-vs-the-rest scheme.

2.1.5 Multinomial Naive Bayes

Naive Bayes is a learning algorithm that is frequently employed to tackle
text classification problems. It is computationally very efficient and easy to implement. Multinomial NB is a probablistic algorithm that is mostly used in NLP.

Bayes theorem, formulated by Thomas Bayes, calculates the probability of an event occurring based on the prior knowledge of conditions related to an event. It is based on the following formula:

$P(A|B) = P(A) * P(B|A)/P(B)$

Where we are calculating the probability of class A when predictor B is already provided.

P(B) = prior probability of B

P(A) = prior probability of class A

P(B|A) = occurrence of predictor B given class A probability

2.2 Deep Learning Algorithms

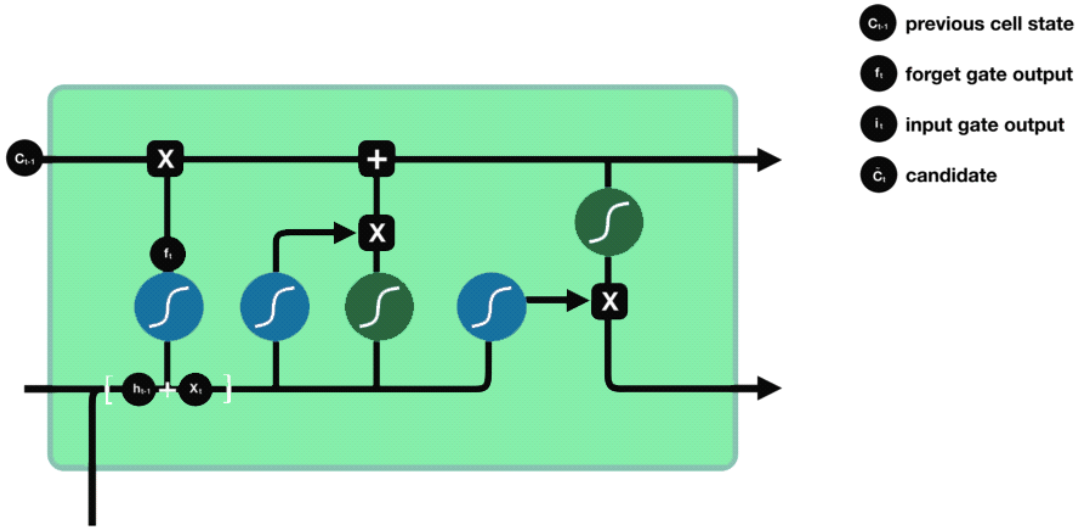
Deep learning is an important element of data science, which includes statistics and predictive modeling. It is extremely beneficial to data scientists who are tasked with collecting, analyzing and interpreting large amounts of data; deep learning makes this process faster and easier.

At its simplest, deep learning can be thought of as a way to automate predictive analytics. While traditional machine learning algorithms are linear, deep learning algorithms are stacked in a hierarchy of increasing complexity and abstraction.
In traditional machine learning, the learning process is supervised. The advantage of deep learning is the program builds the feature set by itself without supervision. Unsupervised learning is not only faster, but it is usually more accurate.

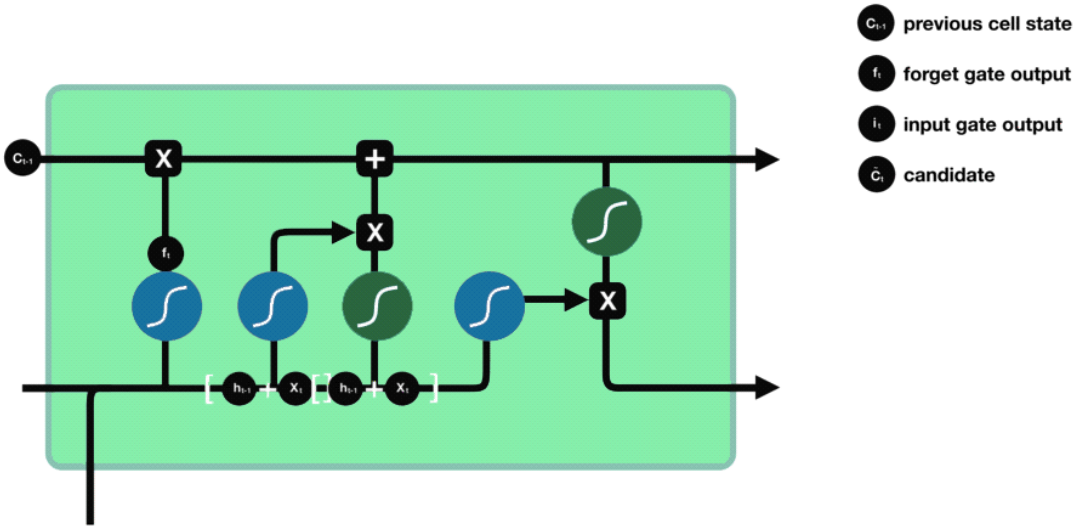
2.2.1 Long Short Term Memory Model

Tradition neural networks suffer from short-term memory. Also, a big drawback is the vanishing gradient problem. (While backpropagation the gradient becomes so small that it tends to 0 and such a neuron is of no use in further processing.) LSTMs efficiently improves performance by memorizing the relevant information that is important and finds the pattern. Having a good hold over memorizing certain patterns LSTMs perform fairly better. As with every other NN, LSTM can have multiple hidden layers and as it passes through every layer, the relevant information is kept and all the irrelevant information gets discarded in every single cell. LSTM have 3 main gates:

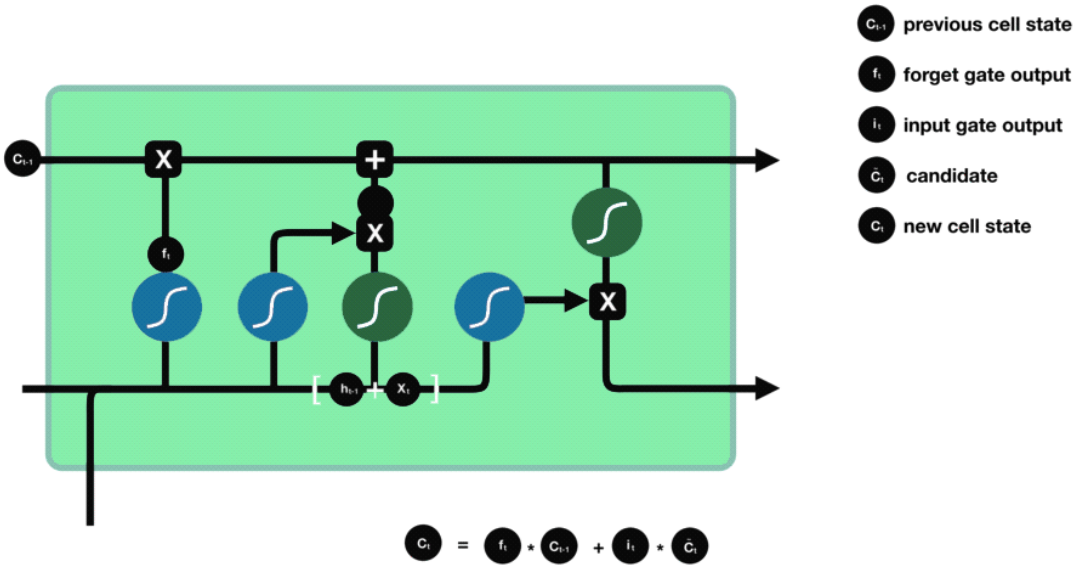
FORGET GATE: This gate is responsible for deciding which information is kept for calculating the cell state and which is not relevant and can be discarded. ht-1(information from previous hidden layer) and x1(information from current cell) are the 2 inputs given to the Forget gate. They are passed through a sigmoid function and the ones tending towards 0 are discarded, and others are passed further to calculate the cell state.



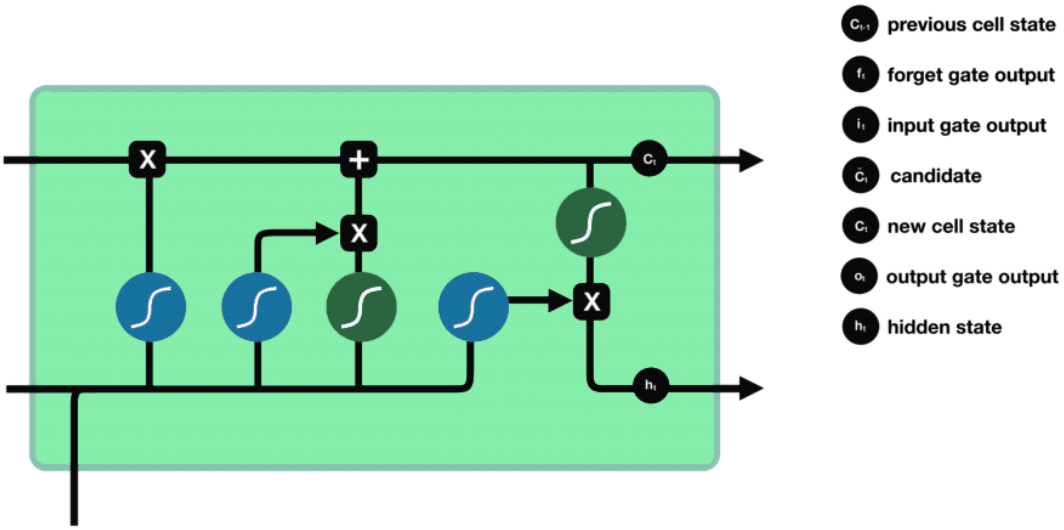
INPUT GATE: Input Gate updates the cell state and decides which information is important and which is not. As forget gate helps to discard the information, the input gate helps to find out important information and store certain data in the memory that relevant. ht-1 and xt are the inputs that are both passed through sigmoid and tanh functions respectively. tanh function regulates the network and reduces bias.



CELL STATE: All the information gained is then used to calculate the new cell state. The cell state is first multiplied with the output of the forget gate. This has a possibility of dropping values in the cell state if it gets multiplied by values near 0. Then a pointwise addition with the output from the input gate updates the cell state to new values that the neural network finds relevant.



OUTPUT GATE: The last gate which is the Output gate decides what the next hidden state should be. ht-1 and xt are passed to a sigmoid function. Then the newly modified cell state is passed through the tanh function and is multiplied with the sigmoid output to decide what information the hidden state should carry.


previous cell state
forget gate output
input gate output
candidate
new cell state
output gate output
hidden state

3. Methodology

3.1 About Dataset

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.

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.

3.2 Text Preprocessing

3.2.1 Lower casing

3.2.2 Text may contain numbers, special characters, and unwanted spaces. Depending upon the problem we face, we may or may not need to remove these special characters and numbers from text. However, for the sake of explanation, we will remove all the special characters, numbers, and unwanted spaces from our text.

3.2.3 Tokenization: Splitting sentences into text tokens.

3.3.3 Lemmatization: The final pre-processing step is the lemmatization. In lemmatization, we reduce the word into dictionary root form. For instance, "cats" is converted into "cat". Lemmatization is done in order to avoid creating features that are semantically similar but syntactically different. For instance, we don't want two different features named "cats" and "cat", which are semantically similar, therefore we perform lemmatization.

3.3.4 TFID: The bag of words approach works fine for converting text to numbers. However, it has one drawback. It assigns a score to a word based on its occurrence in a particular document. It doesn't take into account the fact that the word might also

be having a high frequency of occurrence in other documents as well. TFIDF resolves this issue by multiplying the term frequency of a word by the inverse document frequency. The TF stands for "Term Frequency" while IDF stands for "Inverse Document Frequency".

The term frequency is calculated as:
Term frequency = (Number of Occurrences of a word)/(Total words in the document)
And the Inverse Document Frequency is calculated as:

IDF(word) = Log((Total number of documents)/(Number of documents containing the word))

3.3 Training–Test Set Split

For all the models fitted in this study, we split the balanced data into 70% for the training set and 30% for the testing set (validation).

4. Hardware and Software Requirements and Tools Used

4.1 Software Requirements :

- Python
- Anaconda (Jupyter Notebook)
- Python Libraries (Scikit - Learn, Keras, Tensorflow, Pandas, Numpy , etc)

4.2 Hardware Requirements :

- Minimum 4 GB RAM
- Intel Core-i3 or above processor

5. Results

In this section, I present analytic results of the various machine learning models adopted in this research. All model diagnostic metrics in this paper are based on the validation/test set.

5.1 Prediction Accuracy

The idea here was to determine which model performs best with our data, and as a first step, we considered each model's overall out-of-sample prediction accuracy on the
test set. Note that as a rule of thumb, it is advisable to use the global f1-scores for model comparison instead of the accuracy metric; however, in our case, the two metrics were the same for all classifiers.
the least performing model in terms of prediction accuracy was the Random Forest Classifier .However, note that the best performing models were the machine learning classifiers (Logistic Regression, Linear SVC, and Multinomial NB).

5.2 Hamming Loss

In multi-label classification, a misclassification is no longer a hard wrong or right. A prediction containing a subset of the actual classes should be considered better than a prediction that contains none of them.Hamming Loss measures between 0 to 1. Lower the Hamming loss better the model performance.

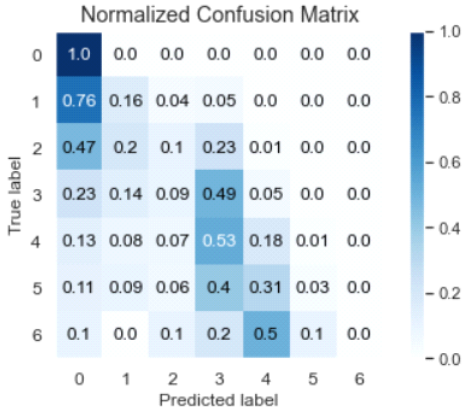
We use Hamming loss/Hamming score metrics. does that by exclusive or (XOR) between the actual and predicted labels and then average across the dataset. The Hamming Loss counts the number of labels for which our prediction is wrong normalizing it.

$$HammingLoss(\widehat{y}, y) = \frac{1}{NL}\sum_{n=1}^{N}\sum_{i=1}^{L} 1_{\widehat{y}_i \neq y_i}$$

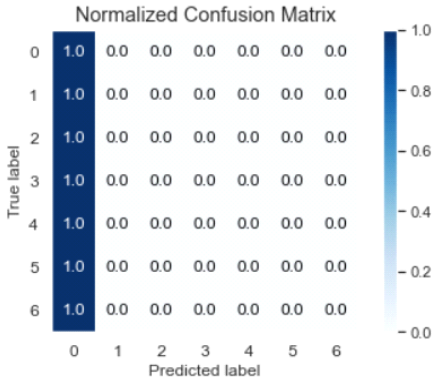
	Prediction Accuracy (%)	Hamming Loss	Cross - Validation Score (%)
Logistic Regression	91.82	0.0817	91.96
Random Forest Classifier	89.72	0.1028	89.83
Linear SVC	91.67	0.0833	91.9
Multinomial NB	90.57	0.1028	90.81
Adaptive Boosting Regressor	90.71	0.0920	90.79
LSTM	82.39	0.1016	-
LSTM with dropout	89.83	0.1657	-

5.3 Confusion Matrix

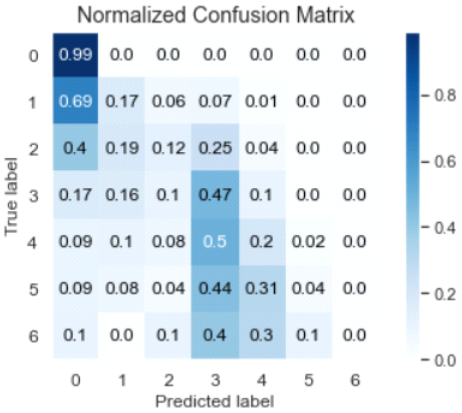
There is a need to look at the confusion matrix to assess a classification model's quality of classification. For an ideal confusion matrix, we expect to get values only on the leading/principal diagonal, since they represent correct classification; values off-diagonal are those that were misclassified.



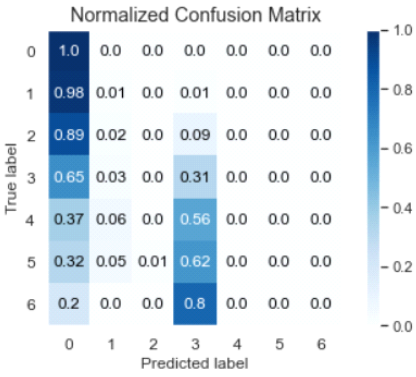
Logistic Regression



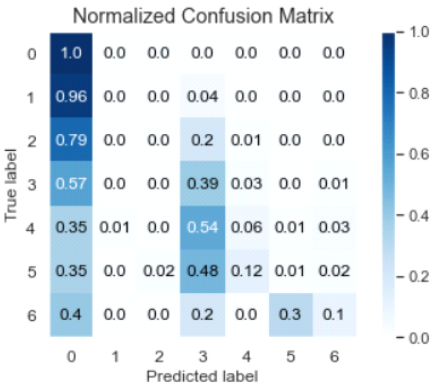
Random Forest Classifier



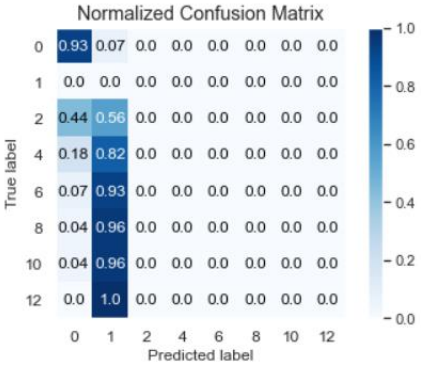
Linear SVC



Multinomial NB



Adaboost Classifier



LSTM Model

5.4 Classification Report

In this subsection, we examine each classifier's precision, recall, and f1-score.
Note that precision is the ratio of predicted values of a risk class that actually belong to that class to all values. In any of the confusion matrices above, it is the ratio of the values on the leading diagonal to the sum of all values in that column. Recall (true positive rate), on the other hand, is a ratio of the actual values of a risk class that were actually predicted as belonging to that class. Precision and recall usually have inverse relations, and the f1-score is a metric that measures both precision and recall together; it presents a combined picture of both precision and recall. It is a harmonic mean of the two metrics. Support is the actual number of occurrences of a particular risk class in the data set (usually the validation data). The accuracy parameter is simply the overall predictive power of the classifier. It is simply the ratio of the sample data that the classification model correctly classified. In each of the confusion matrices above, the sum of all elements on the principal diagonal is divided by the sum of all elements in the confusion matrix to obtain each classifier's accuracy. The micro-average metric is the arithmetic mean of the precision, recall, and f1-scores, while the weighted average computes the weighted average of the precision, recall, and f1-scores.

Classification Logistic Regression :

	precision	recall	f1-score	support
0	0.95	1.00	0.97	42950
1	0.36	0.16	0.22	1947
2	0.30	0.10	0.15	1051
3	0.49	0.49	0.49	1275
4	0.43	0.18	0.25	527
5	0.43	0.03	0.05	112
6	0.00	0.00	0.00	10
accuracy			0.92	47872
macro avg	0.42	0.28	0.30	47872
weighted avg	0.89	0.92	0.90	47872

Classification Random Forest Classifier :

	precision	recall	f1-score	support
0	0.90	1.00	0.95	42950
1	0.00	0.00	0.00	1947
2	0.00	0.00	0.00	1051
3	0.00	0.00	0.00	1275
4	0.00	0.00	0.00	527
5	0.00	0.00	0.00	112
6	0.00	0.00	0.00	10
accuracy			0.90	47872
macro avg	0.13	0.14	0.14	47872
weighted avg	0.80	0.90	0.85	47872

Classification MLinear SVC :

	precision	recall	f1-score	support
0	0.95	0.99	0.97	42950
1	0.34	0.17	0.23	1947
2	0.27	0.12	0.16	1051
3	0.45	0.47	0.46	1275
4	0.32	0.20	0.25	527
5	0.15	0.04	0.06	112
6	0.00	0.00	0.00	10
accuracy			0.92	47872
macro avg	0.36	0.28	0.30	47872
weighted avg	0.89	0.92	0.90	47872

Classification Multinomial Naive Bayes Classifier :

	precision	recall	f1-score	support
0	0.92	1.00	0.96	42950
1	0.11	0.01	0.01	1947
2	0.22	0.00	0.00	1051
3	0.45	0.31	0.37	1275
4	0.50	0.00	0.01	527
5	0.00	0.00	0.00	112
6	0.00	0.00	0.00	10
accuracy			0.91	47872
macro avg	0.31	0.19	0.19	47872
weighted avg	0.85	0.91	0.87	47872

Classification Adaptive Boost Classifier :

	precision	recall	f1-score	support
0	0.92	1.00	0.96	42950
1	0.18	0.00	0.01	1947
2	0.00	0.00	0.00	1051
3	0.42	0.39	0.41	1275
4	0.33	0.06	0.10	527
5	0.08	0.01	0.02	112
6	0.03	0.10	0.05	10
accuracy			0.91	47872
macro avg	0.28	0.22	0.22	47872
weighted avg	0.85	0.91	0.87	47872

5.5. Tuning of Hyperparameters

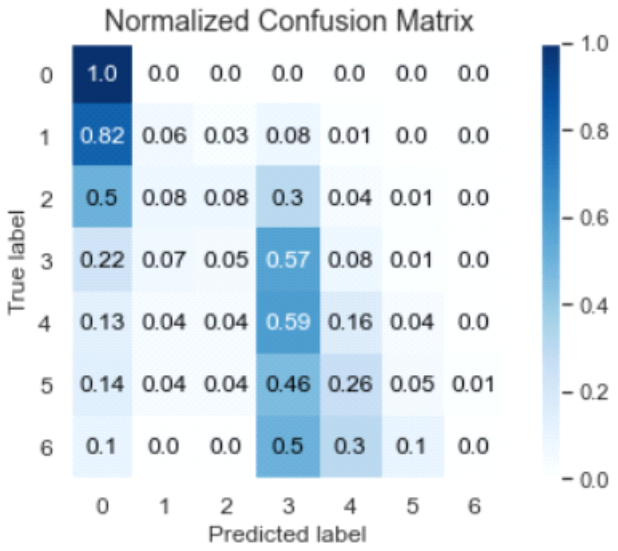
In this subsection, we present the optimal hyperparameters to tune our best model i.e. Linear Support Vector Classifier model with the following hyperparameters:

```
params = {'penalty': ['l1','l2'],
          'loss':['hinge','squared_hinge'],
          'max_iter':[1000,2000,3000],
          'multi_class': ['ovr','crammer_singer'],
          'fit_intercept' : ['True', 'False'],
          'random_state': [10,20,50,82,4]}
```

```
{'random_state': 20,
 'penalty': 'l2',
 'multi_class': 'ovr',
 'max_iter': 1000,
 'loss': 'hinge',
 'fit_intercept': 'True'}
```

We've deployed Randomized Search CV for Hyperparameter Tuning of our model with 3 cross validations and 20 iterations each which is equivalent to 60 fits.

It was found that our model was overfitted and the final tuned model gave us the prediction accuracy of 91.67%.



Confusion Matrix of our Tuned Linear Support Vector Classifier

Classification Linear SVC model Classifier :

	precision	recall	f1-score	support
0	0.95	1.00	0.97	42950
1	0.34	0.06	0.10	1947
2	0.34	0.08	0.12	1051
3	0.46	0.57	0.51	1275
4	0.29	0.16	0.20	527
5	0.12	0.05	0.07	112
6	0.00	0.00	0.00	10
accuracy			0.92	47872
macro avg	0.36	0.27	0.28	47872
weighted avg	0.88	0.92	0.89	47872

Classification Report of our Tuned Linear Support Vector Classifier

It was found that our model was overfitted and the final tuned model gave us the prediction accuracy of 91.67%.

6. Conclusions :

This research evaluated the malicious comment text classification using machine learning and deep learning techniques. Using real data, we compared the various machine learning algorithms' accuracy by performing detailed experimental analysis while classifying the text into 7 categories.
Generally, Linear Support vector Classification machine learning algorithms have shown a better performance with our real-life data than others, and the most performing models are all ensemble classifiers.It was found that the Random Forest classifier generated the least accurate predictions.

7. Future Scope:

The future of sentiment analysis is going to continue to dig deeper, far past the surface of the number of likes, comments and shares, and aim to reach, and truly understand, the significance of social media interactions and what they tell us about the consumers behind the screens. This forecast also predicts broader applications for sentiment analysis – brands will continue to leverage this tool, but so will individuals in the public eye, governments, nonprofits, education centers and many other organizations.

Sentiment analysis is getting better because social media is increasingly more emotive and expressive. A short while ago, Facebook introduced “Reactions,” which allows its users to not just ‘Like’ content, but attach an emoticon, whether it be a heart, a shocked face, angry face, etc. To the average social media user, this is a fun, seemingly silly feature that gives him or her a little more freedom with their responses. But, to anyone looking to leverage social media data for sentiment analysis, this provides an entirely new layer of data that wasn’t available before. Every time the major social media platforms update themselves and add more features, the data behind those interactions gets broader and deeper.