Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

INFORMATION SECURITY MANAGEMENT

WINTER SEMESTER 2022-23

PROJECT

Team

B K Anirudh – 19BIT0348
Manam Lakshmi Nivas – 19BIT0189

Under the guidance of

Prof. Priya V

Cyberbullying Detection: Identifying Hate Speech Using Machine Learning

ABSTRACT

Bullying has been from the dawn of time; it's only that the methods of bullying have evolved over time, from physical to cyberbullying. There are eight varieties of cyberbullying, according to Williard (2004), including harassment, denigration, impersonation, and so on. Although social media sites have been available for nearly two decades, there haven't been many effective tactics to combat social bullying, and it has become one of the most concerning concerns in recent years. We study strategies to detect hate speech in social media while distinguishing it from general profanity in this project, which includes a thorough assessment of some previous research on cyberbullying detection tools.We plan to use supervised classification algorithms and a manually annotated open source dataset to build lexical baselines for this effort. This paper compares and contrasts numerous Supervised algorithms, including both standard and ensemble techniques. The results demonstrate that Ensemble supervised methods have the potential to outperform traditional supervised approaches. A variety of potential research directions are also discussed.

KEYWORDS

Machine Learning **.** Cyberbullying **.** Super- vised **.** Ensemble **.** Hate Speech **.** Natural Language Processing

INTRODUCTION

Hate speech refers to statements made with the intent of inspiring hatred toward a particular group, such as a community, religion, or race. This speech could be relevant or not, but it will almost surely result in violence. The rise of violence against minorities around the world, including mass shootings, lynchings, and ethnic cleansing, has been linked to online hate speech.

As a result of the massive increase in user-generated web material, particularly on social media networks, the number of hate speech is continuously expanding. Cyberbullying detection research has grown in recent years, owing in part to the expansion of cyberbullying throughout social media and its harmful influence on the younger generation. A increasing body of research on automated methods for identifying cyberbullying is available. The elements of a cyberbullying exchange are determined using machine learning and natural language processing techniques, and cyberbullying is detected automatically by comparing textual input to the specified properties.

The domain of an utterance, its discourse context, as well as context consisting of co-occurring media objects (e.g. images, videos, audio), the exact time of posting and world events at this time, the identity of the author, and the targeted recipient are all factors that influence what is considered a hate speech message.

This project provides a comprehensive and well-organized review of automatic hate speech identification, as well as a systematic comparison of a few of the existing strategies and an informative assessment of some published research on cyberbullying detection methods.

RELATED WORK

Paper	Algorithm	Advantages	Issues	Metrics used
Social Media Cyberbullying Detection using Machine Learning	SVM and Neural Network	The detection rate is high in both SVM and Neural Network algorithm	Takes time to detect the traffic characteristics with a large proportion.	Accuracy
Deep Learning Algorithm for Cyberbullying Detection	CNN-CB is based on convolutional neural network (CNN)	CNN-CB will eliminate the need for feature engineering and it has better prediction than traditional cyberbullying detection approaches.	This Model is not a silver bullet solution to all types of cyber bullying attacks	i)Effective arrival time ii)Waiting time
Online Social Network Bullying Detection Using Intelligence Techniques	Adaboost and Genetic algorithm.	The algorithms uses genetic operators like crossover and mutation for optimizing the parameters and obtain precise type of cyberbullying activity.	Only slowloris attack was considered, Only slow header and slow POST were examined.	i)Precision ii)Recall iii)F measure
Cyberbullying Detection Using Machine Learning	To detect cyberbullying activities from social network data. Naive Bayes classifier is used	Navie bayes algorithm based on Machine Learning (ML) and Cyberbullying Detection Algorithm and bullying Detection Algorithm based on Machine Learning.	The accuracy of this custom classifier will be poor in the server's infancy	i)Accuracy rate ii)Sensitivity rate iii)Specificit y rate

Cyberbullying detection: advanced preprocessing techniques & deep learning architecture for Roman Urdu data	RNN-LSTM, RNN-BiLSTM and CNN models	The epochs executions, model layers and tuning hyperparameters to analyze and uncover cyberbullying textual patterns in Roman Urdu.	Takes time to detect the traffic characteristics with a large proportion.	F1 score for non-cyberbullying content prediction was 90% whereas for cyberbullying content, the score was 67% only.
XBully: Cyber bullying Detection within a Multi-Modal Context	XBully –Cyber bullying detection framework that models multi-modal social media data in a collaborative way.	Among different social media sessions, multi-modal social media data is been modeled as a heterogeneous network by exploiting co-existence and neighborhood relations and aim at learning the embedding representation for nodes in the network.	XBully technique achieves better detection performance than the variants without the embedding refinement component. This result highlights the benefit of collaboratively refining the embeddings by integrating information from similar nodes during the learning process.	Two common evaluation metrics are calculated - Macro F1 and Micro F1. A macro-average computes the metric independently for each class and then takes the average as the output. Whereas a micro-average will aggregate contributions of all classes to compute the average metric.
Text Mining Techniques for Cyberbullying Detection	Deep learning approach like CNN And other methods like 2.Naive Bayes 3.SMV 4. Desision Tree	By using machine learning method and deep learning method for cyberbullying has a good detection rate for the current popular attacks.	Only slowloris attack was considered and takes time.	It was found that the best accuracy was achieved when a deep learning approach is used

				especially when CNN used than naive bayes.
Improved Cyber bullying Detection Using Gender Information	Support Vector Machine(SVM) classifier using WEKA	Used a moderator especially in the fora that are mostly used by teenagers is a common thing.A corpus is used with posts written by both male and female users as per the datasets	The gender-specific approach improved the baseline by 39% in precision, 6% in recall, and 15% in F-measure.	The main focus of the technical studies which have been conducted on cyber bullying detection is mainly on the content of the text written by the users but not the user's information.
Cyber Bullying Detection on Social Media using Machine Learning	A bidirectional deep learning model called BERT.	BERT method is both fast and reliable, which makes it very suitable for real-time detection of cyberbullying.	Obtained result but it is not upto the mark like 91.05%	The accuracy fri BERT is better than SMV and Navie bayes
Cyberbullying Detection Through Sentiment Analysis	SA model for identifying cyberbullying texts in Twitter social media. Support Vector Machines (SVM) and Naive Bayes (NB) are used in this model as supervised machine learning	The results of the experiments conducted on this model showed encouraging outcomes when a higher n-grams language model	i)Need improved hit rate among attack classes ii)Automatic parameter calibration is missing	the results showed that SVM classifiers have better performance measures than NB classifier
Cyberbullying	Hybrid algorithm:	Takes time but it can scan more data	Takes time and all algorithms	novel neural network has

Detection: Hybrid Models Based on Machine Learning and Natural Language Processing Techniques.	novel neural network, XGBoost,Naive Bayes,Logistic Regression and SMV	form dataset	will not give perfect score	given better result than SVM, NaiveBayes, XGBoost , Logistic Regression.

DATASET

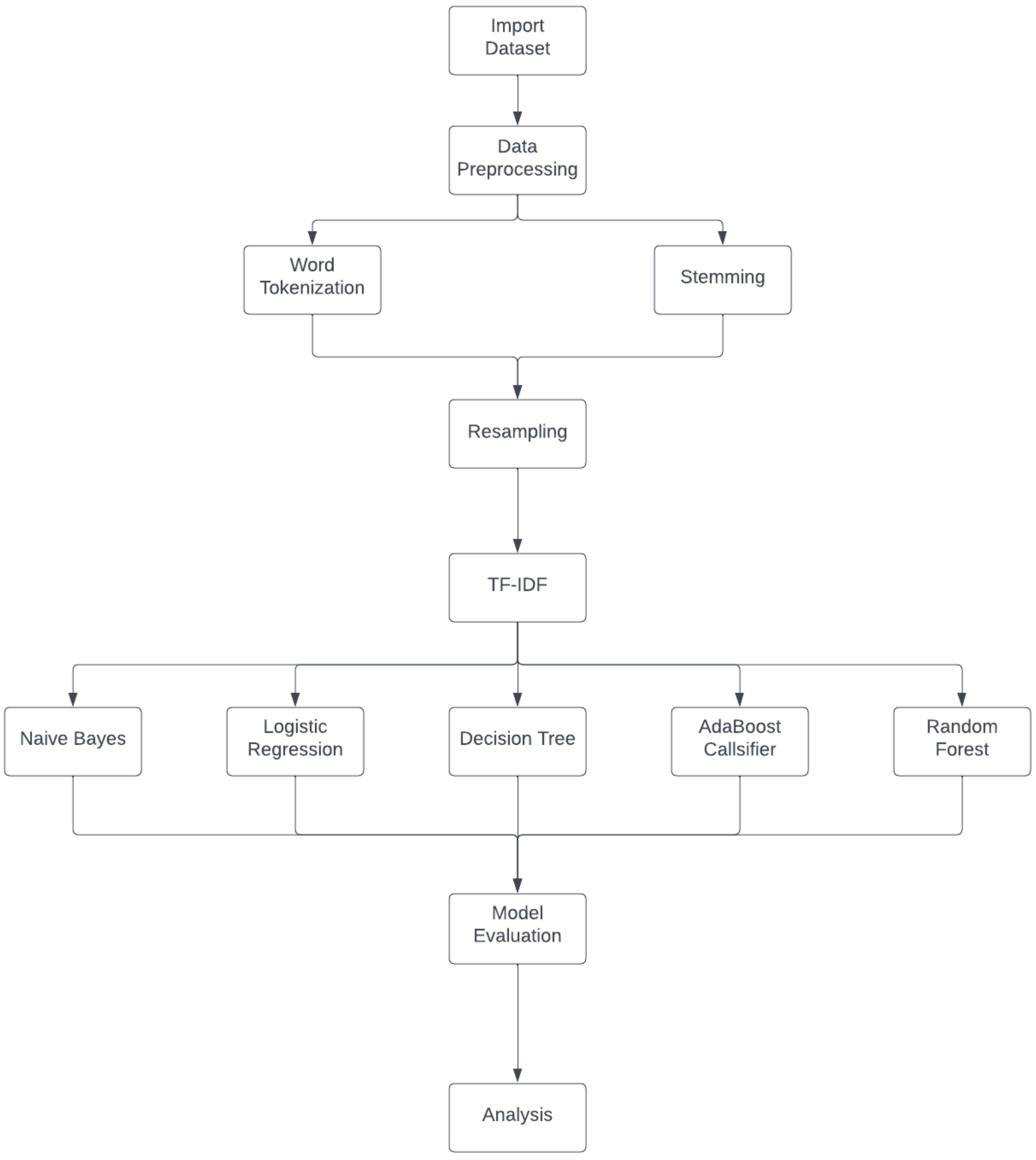
For the final results, we used Dataturks' Tweet Dataset for Cybertroll Detection, which we received from Kaggle. Because of the gravity of the problem we were attempting to solve, it was critical to select a dataset that was thorough, dependable, relevant, and concise. While we looked at a lot of additional datasets, many of them had missing properties, were of poor quality, or had data that was useless after manual assessment. So, after experimenting with a variety of other open-source datasets, we settled on this one because it seemed to fit all of the criteria.

Here is the Detailed Description of the dataset:
1) It is a partially manually labelled dataset.
2) Total Instances: 20001
The tweet and label attributes are included in the dataset [0 refers to No, while 1 corresponds to Yes]. The dataset was saved as a json file. The initial set of fields in the annotation attribute were removed and filled with the label values to simplify the next step because the dataset's fields were relatively simple to comprehend. The table shows the number of occurrences for each class.

	Twitter
Total Instances	20001
Cyber Bullying Instances	7822
Non Cyber Bullying Instances	12179

Link: https://www.kaggle.com/datasets/dataturks/dataset-for-detection-of-cybertrolls?select=Dataset+for+Detection+of+Cyber-Trolls.json

ARCHITECTURE DIAGRAM



DATA PREPROCESSING

1) Word Tokenization*:* A token is a single item that functions as a phrase or paragraph's building blocks. Word Tokenization breaks down our text into individual words in a list.

2)A token is a single item that functions as a phrase or paragraph's building blocks. Word Tokenization breaks down our text into individual words in a list.

3)*To remove punctuation, we save only non-punctuation characters, which can be verified using string. a lack of punctuation*

4)Stemming is a linguistic normalising procedure in which words are reduced to their root words. To acquire the stemmed tokens, we use the nltk.stem.porter.PorterStemmer class. For example, the words "connection," "connected," and "connecting" can all be reduced to the single word "connect."

5)Digit removal: *We also filtered out any numeric content as it doesn't contribute to cyberbullying.*

6)*The next step was to extract features so that they could be used with machine learning techniques, which we did using Python's sklearn library and the TF-IDF Transform. The TF-IDF is a statistical measure of a word's importance that is calculated by multiplying the number of times a word appears in a document by the inverse document frequency of the word.*

7) Because the data was skewed, resampling on the training data was required. The data was first split into Training and Test in an 80:20 ratio, and then resampling was done on the training data.

- We used oversampling of the minority class since we had plenty of data to work with. This means that if the majority class has 1,000 examples and the minority class only has 100, this strategy will oversample the minority class to give it 1,000.
- For all the "not majority" classes, which in our case was only one, the RandomOverSample function from the imblearn package is used for oversampling.

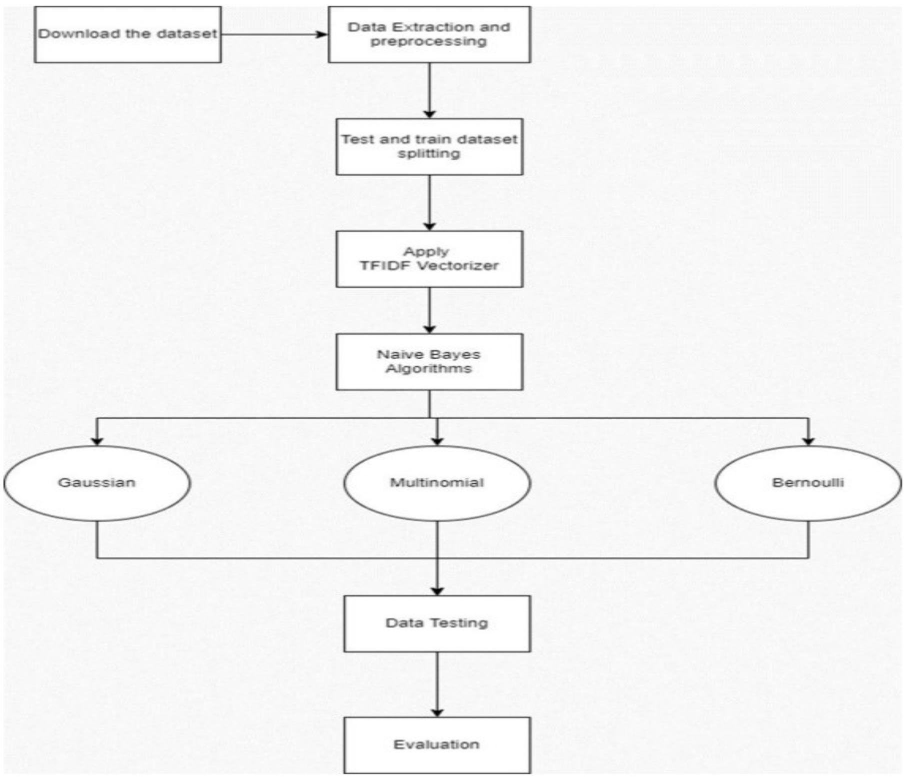
After resampling, the training data had 9750 CB & NON-CB instances.

DESCRIPTION OF METHODS USED

Gaussian Naive Bayes

The Naive Bayes classifiers are a set of classification algorithms based on mathemat- ics' Bayes' Theorem. The Bayes' theorem, in simple terms, describes the likelihood of an event based on prior knowledge of conditions that could be relevant to the event. It is a family of algorithms that share a similar idea, namely that each pair of features being classified is independent of the others. For binary (two-class) and multi-class classification issues, Naive Bayes is a classification algorithm. When stated with binary or categorical input values, the technique is the easiest to grasp.It's termed naive Bayes because the probabilities for each hypothesis are simplified to make them more tractable to calculate.

Naive Bayes can be extended to real-valued attributes by assuming a Gaussian distribution, which is the most frequent assumption. Gaussian Naive Bayes is the name given to this extension of Naive Bayes. There are also Multinomial naive Bayes and Bernoulli naive Bayes, in addition to Gaussian naive Bayes. We chose the Gaussian Naive Bayes method since it is the most common and one of the easiest to apply because all we have to do is estimate the mean and standard deviation from the training data.



Logistic Regression

In a dataset, regression analysis is a predictive modelling technique that examines the relationship between the target or dependent variable and the independent variable. When the target and independent variables have a linear or non-linear connection and the target variable has continuous values, regression analysis techniques are applied. Regression analysis entails establishing the best fit line, which is a line that goes through all of the data points with the least amount of distance between them.

When the dependent variable is discrete, one of the types of regression analysis techniques employed is logistic regression. For instance, 0 or 1, true or false, and so forth. A sigmoid curve denotes the relationship between the target variable and the independent variable by transferring any real value to a value between 0 and 1. We chose Logistic Regression because our data set was huge and the target variables had about equal occurrences of values. Furthermore, there was no association between the dataset's independent variables.

Decision Tree Classifier

A Decision Tree is created by posing a series of questions about the data set. After each response, a follow-up question is asked until a decision about the record's class label is reached. A decision tree, which is a hierarchical structure consisting of nodes and directed edges, can be used to organise a series of questions and their alternative solutions. Root, Internal, and Leaf nodes are the three types of nodes.

Each leaf node in a decision tree is given a class label. The root and other internal nodes, which are non-terminal nodes, contain attribute test requirements to distinguish records with various properties. We start at the root of the tree and split the data on the characteristic that yields the most information gain using the decision method.

We can then continue this splitting operation at each child node in an iterative process until the leaves are pure. This indicates that all of the samples at each leaf node are of the same class.

Adoboost Classifier

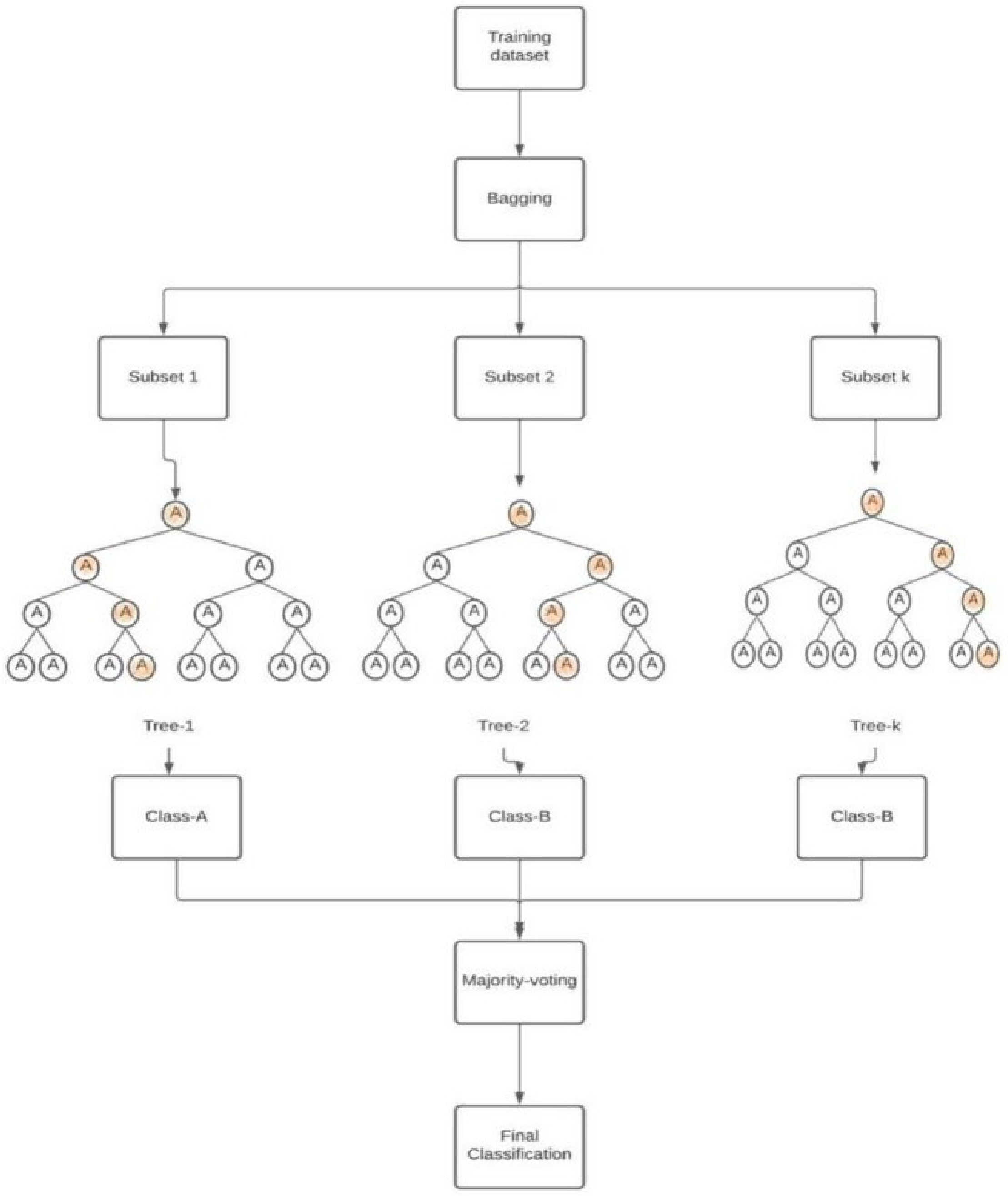
AdaBoost is a method for creating iterative ensembles. The general idea underlying boosting methods is to train predictors in order, with each one attempting to correct the one before it. The Ad- aBoost classifier creates a strong classifier by merging numerous low-performing classifiers, resulting in a high-accuracy, high-performance classifier. Adaboost's core principle is to establish the weights of classifiers and train the data sample in each iteration so that reliable predictions of uncommon observations may be made.

Any machine learning method that accepts weights on the training set can be used as a basic classifier.

AdaBoost and Random Forest are similar in that they both add up the predictions given by each decision tree inside the forest to determine the final classification. However, there are some slight differences. The decision trees in AdaBoost have a depth of one (i.e. 2 leaves). Furthermore, the predictions provided by each decision tree have variable effects on the model's ultimate forecast. In the AdaBoost algorithm, instead of taking the average of the predictions given by each decision tree in the forest (or the majority in the case of classification), each decision tree contributes a different amount to the final prediction.

Random Forest

The Random Forest Classifier, as the name suggests, is made up of a huge number of individual decision trees that work together as an ensemble. Each tree in the random forest produces a class prediction, and the class with the most votes becomes the prediction of our model. Because the trees shelter each other from their individual errors, they can yield ensemble forecasts that are more accurate than any of the individual predictions. The Bagging method is used to diversify models by allowing each individual tree to sample from the dataset at random with replacement.



ANALYSIS

We used Naive Bayes(Gaussian), Logistic regression, and J48 Decision Tree as conventional approaches for our supervised learning methodology analysis. We used AdaBoost and RandomForest Classifiers as Ensemble techniques. According to our findings, the Gaussian Naive Bayes classifier performed the worst, while the Random Forest Classifier delivered the greatest results across the board.
The Random Forest classifier performed the best, which was unsurprising. The Decision Tree classifier outperformed the Naive Bayes and Logistic Regression classifiers. Because it is an extension of the Decision Tree classifier, averaging out results from numerous recursions of the same, the Random Forest Classifier came out on top in all performance criteria, which was expected.
The below are the metrics used to assess model performance:

$$F - Measure = \frac{(2 \times Precison \times Recall)}{Precision + Recall}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

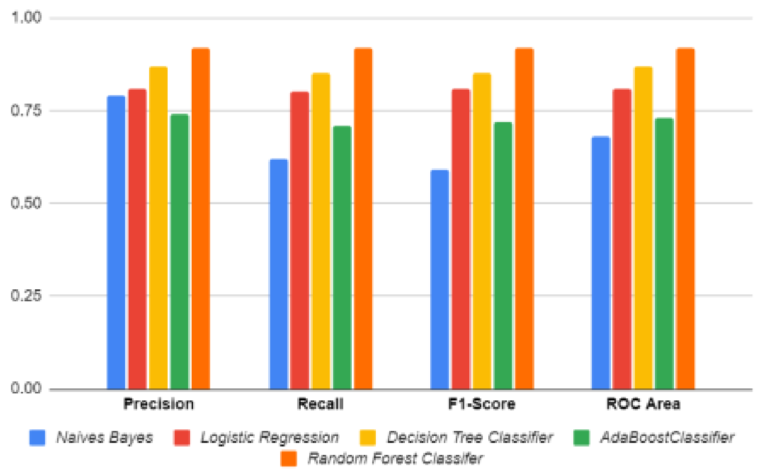
$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

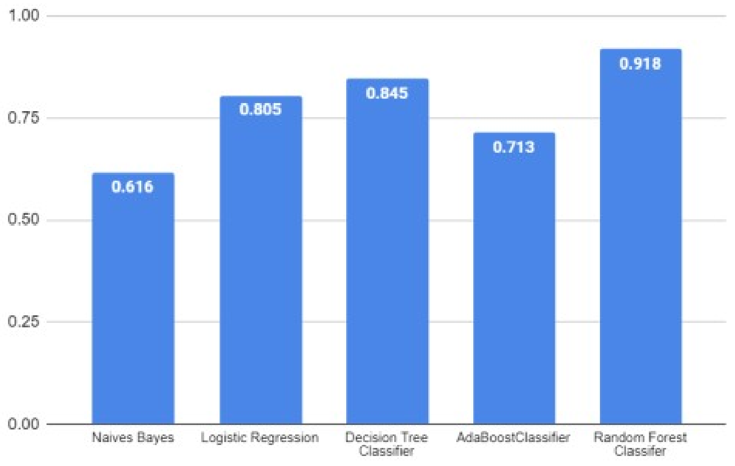
ROC area, Which denotes the area under the curve formed by plotting TP rates.

TP = No.of True Positives
TN = No.of True Negatives
FP = No.of False Positives
FP = No.of False Negatives

Traditional Supervised Learning used:
NaiveBayes, Logistic Regression and J48 Decision Trees classifier.
The Ensemble Learning Methods used:
AdaBoost and Random Forest classifier.



Precision, recall, F1- score and ROC area



Accuracy

RESULTS

Supervised Traditional Methods

	NaiveBayes	Regression	DecisionTree
Accuracy	0.62	0.80	0.85
Precision	0.79	0.81	0.88
Recall	0.62	0.80	0.85
F1-Score	0.59	0.81	0.85
ROC area	0.68	0.81	0.87

Supervised Ensemble Methods

	AdaBoost	Random Forest
Accuracy	0.71	0.92
Precison	0.74	0.92
Recall	0.71	0.92
F1 – Score	0.72	0.92
ROC area	0.73	0.92

CONCLUSIONS

In this research, we compare and contrast numerous Supervised algorithms, as well as various Supervised Ensemble approaches. Random Forest classifier had the best overall performance, with an accuracy of around 92 percent. The Ensemble techniques performed as well as or better than the Supervised methods, but all of the ensemble methods had a high True positive rate for the cyberbullying class, which is significantly better. The Naive Bayes algorithm performed the poorest, with only 61 percent accuracy.

We examined our technique in this paper and compared it to other works in the "Related Work" section. We also noticed that none of the previous studies we looked at employed any semi-supervised approaches, most likely because they aren't as popular or effective as supervised methods and don't produce comparable results.

The absence of tagged datasets and academics' failure to take a comprehensive approach to cyberbullying when developing detection methods is also something that has to be addressed. Cyberbul- lying detection research has two major obstacles. Another issue was a lack of resources, which prevented us from analysing the performance of SVM (Support Vector Machine) and Multi Layer Perceptron (Neural Networks) classifiers. They were, nevertheless, acknowledged in our research as a source of information.

FUTURE SCOPE

Because the number of characteristics in this scenario might be fairly significant, as illustrated in our example, future work on cyberbullying could benefit from using Dimensionality Reduction. PCA (Principal Component Analysis) and LDA (Linear Discriminant Analysis) are two common algorithms for this purpose that can be quite useful in machine learning, especially when dealing with thousands of features. Principal Components Analysis is one of the most used dimensionality reduction approaches, and it can assist improve the output of the classifier in addition to making feature manipulation easier. The goal is to examine each one's benefits and drawbacks, as well as to compare and contrast their results individually and in combination.

Link: https://drive.google.com/drive/folders/1LLEpZcxYvyVkg--hEZv36DLLhUkDQ7wD?usp=sharing

CODE SNAPSHOTS

```
In [ ]: url = 'https://drive.google.com/uc?export=download&id=12fBlhsa5GIdtme1jT3KlPPIgIdjzqhv1'
        df = pd.read_json(url, lines= True,orient='columns')
        df.head

Out[41]: <bound method NDFrame.head of                                                   content  ... extras
         0                               Get fucking real dude.  ...    NaN
         1       She is as dirty as they come  and that crook ...  ...    NaN
         2       why did you fuck it up. I could do it all day...  ...    NaN
         3       Dude they dont finish enclosing the fucking s...  ...    NaN
         4       WTF are you talking about Men? No men thats n...  ...    NaN
         ...                                                 ...  ...    ...
         19996    I dont. But what is complaining about it goi...  ...    NaN
         19997   Bahah  yeah i&;m totally just gonna&; get pis...  ...    NaN
         19998       hahahahaha >:) im evil mwahahahahahahahaha  ...    NaN
         19999            What&;s something unique about Ohio? :)  ...    NaN
         20000              Who is the biggest gossiper you know?  ...    NaN

         [20001 rows x 3 columns]>
```

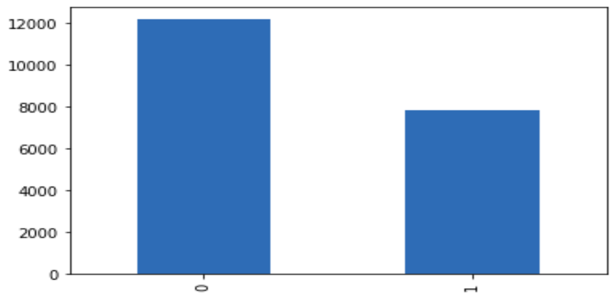
```
In [ ]: df.drop(['extras'],axis = 1,inplace = True)
        df

Out[43]:
```

	content	annotation
0	Get fucking real dude.	1
1	She is as dirty as they come and that crook ...	1
2	why did you fuck it up. I could do it all day...	1
3	Dude they dont finish enclosing the fucking s...	1
4	WTF are you talking about Men? No men thats n...	1
...	...	...
19996	I dont. But what is complaining about it goi...	0
19997	Bahah yeah i&;m totally just gonna&; get pis...	0
19998	hahahahaha >:) im evil mwahahahahahahahaha	0
19999	What&;s something unique about Ohio? :)	0
20000	Who is the biggest gossiper you know?	0

20001 rows × 2 columns

```
In [ ]: df.shape

Out[44]: (20001, 2)
```

```
In [ ]: df['annotation'].value_counts().sort_index().plot.bar()

Out[45]: <matplotlib.axes._subplots.AxesSubplot at 0x7f5c4d4dd048>
```



```
In [ ]: #Biasness
        print("PosiNon cyber trollingtive: ", df.annotation.value_counts(
        print("Cybertrolling: ", df.annotation.value_counts()[1]/len(df.a

        PosiNon cyber trollingtive:  60.89195540222989 %
        Cybertrolling:  39.10804459777012 %
```

In [ ]:

```
nltk.download('stopwords')
stop = stopwords.words('english')

regex = re.compile('[%s]' % re.escape(string.punctuation))

    return regex.sub('', s)

df ['content_without_stopwords'] = df['content'].apply(lambda x: ' '.join([word for word in x.split() if word not
df ['content_without_puncs'] = df['content_without_stopwords'].apply(lambda x: regex.sub('',x))
del df['content_without_stopwords']
del df['content']
df
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[47]:

	annotation	content_without_puncs
0	1	Get fucking real dude
1	1	She dirty come crook Rengel Dems fucking corru...
2	1	fuck up I could day too Lets hour Ping later s...
3	1	Dude dont finish enclosing fucking showers I h...
4	1	WTF talking Men No men thats menage thats gay
...	...	...
19996	0	I dont But complaining going do
19997	0	Bahah yeah im totally gonna get pissed talking...
19998	0	hahahahaha im evil mwahahahahahahahahaha
19999	0	Whats something unique Ohio
20000	0	Who biggest gossiper know

20001 rows × 2 columns

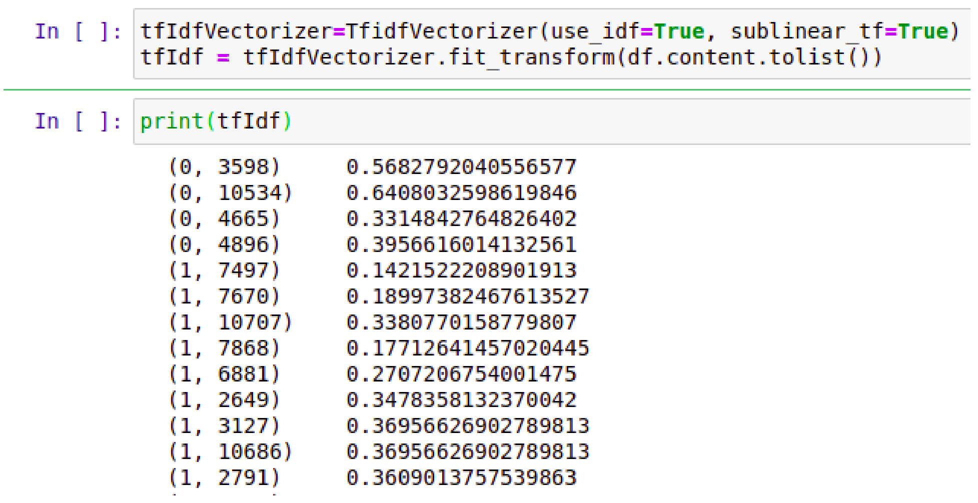
In [ ]:

```
#Stemming
porter_stemmer = PorterStemmer()
#punctuations
nltk.download('punkt')
tok_list = []
size = df.shape[0]

for i in range(size):
  word_data = df['content_without_puncs'][i]
  nltk_tokens = nltk.word_tokenize(word_data)
  final = ''
  for w in nltk_tokens:
    final = final + ' ' + porter_stemmer.stem(w)
  tok_list.append(final)

df['content_tokenize'] = tok_list
del df['content_without_puncs']
df
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[48]:

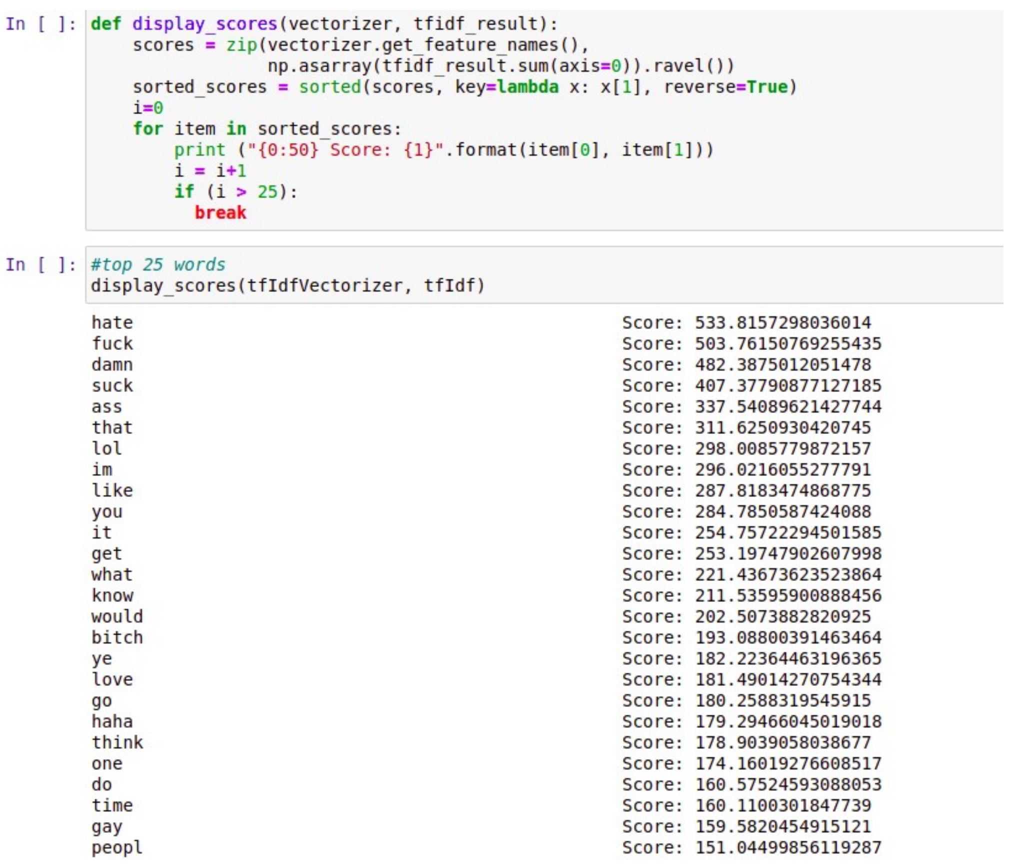
	annotation	content_tokenize
0	1	get fuck real dude
1	1	she dirti come crook rengel dem fuck corrupt ...
2	1	fuck up I could day too let hour ping later s...
3	1	dude dont finish enclos fuck shower I hate ha...
4	1	wtf talk men No men that menag that gay



```
In [ ]: tfIdfVectorizer=TfidfVectorizer(use_idf=True, sublinear_tf=True)
        tfIdf = tfIdfVectorizer.fit_transform(df.content.tolist())
```

```
In [ ]: print(tfIdf)
```

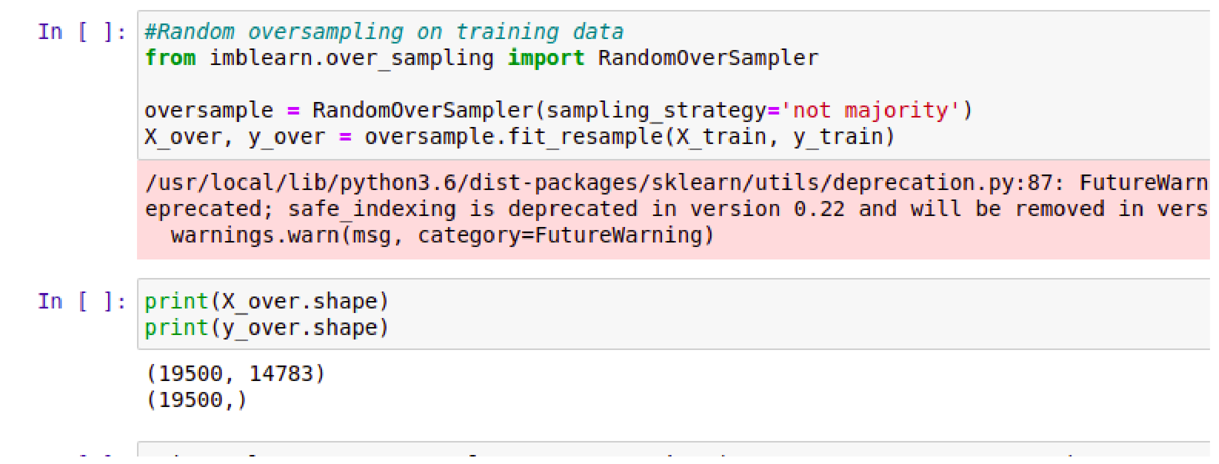
```
  (0, 3598)	0.5682792040556577
  (0, 10534)	0.6408032598619846
  (0, 4665)	0.3314842764826402
  (0, 4896)	0.3956616014132561
  (1, 7497)	0.1421522208901913
  (1, 7670)	0.18997382467613527
  (1, 10707)	0.3380770158779807
  (1, 7868)	0.17712641457020445
  (1, 6881)	0.2707206754001475
  (1, 2649)	0.3478358132370042
  (1, 3127)	0.36956626902789813
  (1, 10686)	0.36956626902789813
  (1, 2791)	0.3609013757539863
```



```
In [ ]: def display_scores(vectorizer, tfidf_result):
            scores = zip(vectorizer.get_feature_names(),
                         np.asarray(tfidf_result.sum(axis=0)).ravel())
            sorted_scores = sorted(scores, key=lambda x: x[1], reverse=True)
            i=0
            for item in sorted_scores:
                print ("{0:50} Score: {1}".format(item[0], item[1]))
                i = i+1
                if (i > 25):
                  break
```

```
In [ ]: #top 25 words
        display_scores(tfIdfVectorizer, tfIdf)
```

```
hate                                               Score: 533.8157298036014
fuck                                               Score: 503.76150769255435
damn                                               Score: 482.3875012051478
suck                                               Score: 407.37790877127185
ass                                                Score: 337.54089621427744
that                                               Score: 311.6250930420745
lol                                                Score: 298.0085779872157
im                                                 Score: 296.0216055277791
like                                               Score: 287.8183474868775
you                                                Score: 284.7850587424088
it                                                 Score: 254.75722294501585
get                                                Score: 253.19747902607998
what                                               Score: 221.43673623523864
know                                               Score: 211.53595900888456
would                                              Score: 202.5073882820925
bitch                                              Score: 193.08800391463464
ye                                                 Score: 182.22364463196365
love                                               Score: 181.49014270754344
go                                                 Score: 180.2588319545915
haha                                               Score: 179.29466045019018
think                                              Score: 178.9039058038677
one                                                Score: 174.16019276608517
do                                                 Score: 160.57524593088053
time                                               Score: 160.1100301847739
gay                                                Score: 159.5820454915121
peopl                                              Score: 151.04499856119287
```

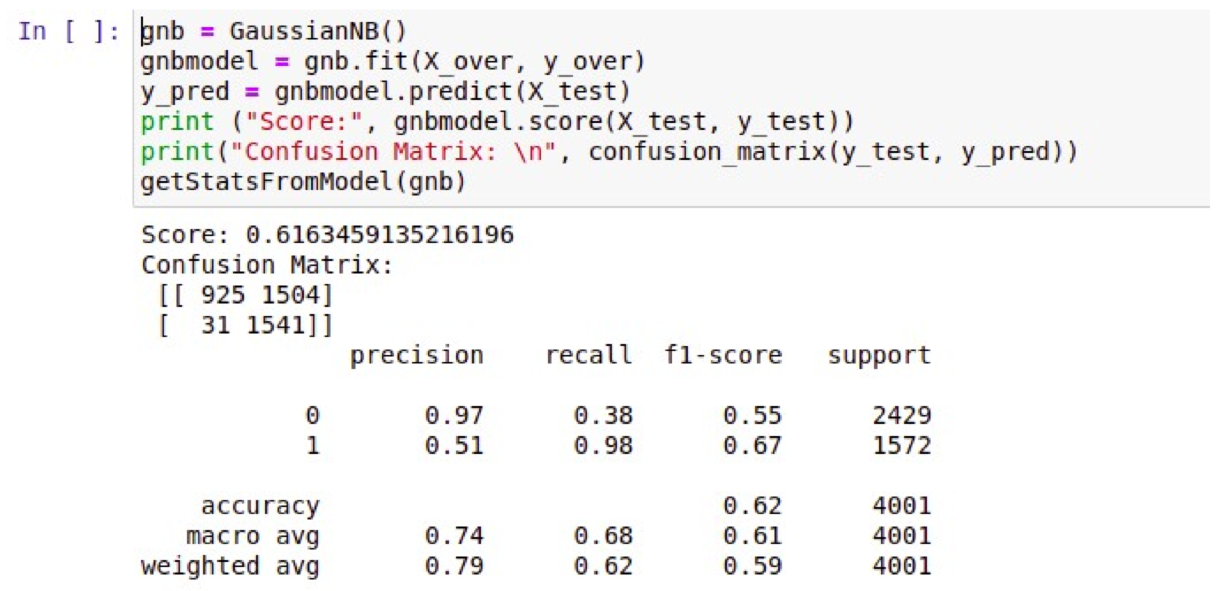


In [ ]:
```
#Random oversampling on training data
from imblearn.over_sampling import RandomOverSampler

oversample = RandomOverSampler(sampling_strategy='not majority')
X_over, y_over = oversample.fit_resample(X_train, y_train)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/utils/deprecation.py:87: FutureWarn
eprecated; safe_indexing is deprecated in version 0.22 and will be removed in vers
  warnings.warn(msg, category=FutureWarning)
```

In [ ]:
```
print(X_over.shape)
print(y_over.shape)
```

```
(19500, 14783)
(19500,)
```



In [ ]:
```
gnb = GaussianNB()
gnbmodel = gnb.fit(X_over, y_over)
y_pred = gnbmodel.predict(X_test)
print ("Score:", gnbmodel.score(X_test, y_test))
print("Confusion Matrix: \n", confusion_matrix(y_test, y_pred))
getStatsFromModel(gnb)
```

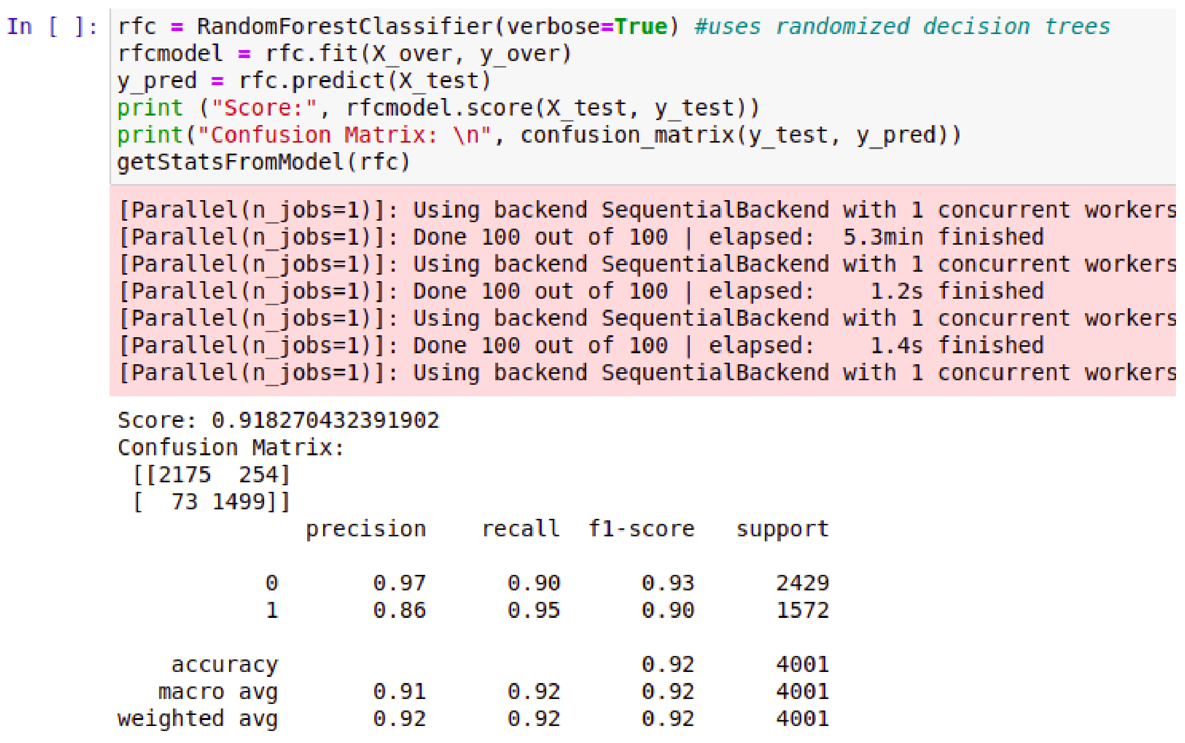
```
Score: 0.6163459135216196
Confusion Matrix:
 [[ 925 1504]
 [  31 1541]]
              precision    recall  f1-score   support

           0       0.97      0.38      0.55      2429
           1       0.51      0.98      0.67      1572

    accuracy                           0.62      4001
   macro avg       0.74      0.68      0.61      4001
weighted avg       0.79      0.62      0.59      4001
```

```
In [ ]: dtc = DecisionTreeClassifier()
        dtc.fit(X_over, y_over)
        y_pred = dtc.predict(X_test)
        print("Accuracy: ",metrics.accuracy_score(y_test, y_pred))
        print("Confusion Matrix: \n", confusion_matrix(y_test, y_pred))
        getStatsFromModel(dtc)
```

```
Accuracy:  0.8500374906273431
Confusion Matrix: 
 [[1896  533]
 [  67 1505]]
              precision    recall  f1-score   support

           0       0.97      0.78      0.86      2429
           1       0.74      0.96      0.83      1572

    accuracy                           0.85      4001
   macro avg       0.85      0.87      0.85      4001
weighted avg       0.88      0.85      0.85      4001
```

```
In [ ]: #Ensemble methods from here
        abc = AdaBoostClassifier()
        abc.fit(X_over, y_over)
        y_pred = abc.predict(X_test)
        print("Accuracy: ",metrics.accuracy_score(y_test, y_pred))
        print("Confusion Matrix: \n", confusion_matrix(y_test, y_pred))
        getStatsFromModel(abc)
```

```
Accuracy:  0.7138215446138465
Confusion Matrix: 
 [[1616  813]
 [ 332 1240]]
              precision    recall  f1-score   support

           0       0.83      0.67      0.74      2429
           1       0.60      0.79      0.68      1572

    accuracy                           0.71      4001
   macro avg       0.72      0.73      0.71      4001
weighted avg       0.74      0.71      0.72      4001
```



```
In [ ]: rfc = RandomForestClassifier(verbose=True) #uses randomized decision trees
        rfcmodel = rfc.fit(X_over, y_over)
        y_pred = rfc.predict(X_test)
        print ("Score:", rfcmodel.score(X_test, y_test))
        print("Confusion Matrix: \n", confusion_matrix(y_test, y_pred))
        getStatsFromModel(rfc)
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:  5.3min finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    1.2s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    1.4s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers
```

```
Score: 0.918270432391902
Confusion Matrix:
 [[2175  254]
 [  73 1499]]
              precision    recall  f1-score   support

           0       0.97      0.90      0.93      2429
           1       0.86      0.95      0.90      1572

    accuracy                           0.92      4001
   macro avg       0.91      0.92      0.92      4001
weighted avg       0.92      0.92      0.92      4001
```

REFERENCE

[1]Mounir, John & Nashaat, Mohamed & Ahmed, Mostafaa & Emad, Zeyad & Amer, Eslam & Mohammed, Ammar. (2019). Social Media Cyberbullying Detection using Machine Learning. International Journal of Advanced Computer Science and Applications. 10. 703-707. 10.14569/IJACSA.2019.0100587.

[2]Monirah . A , Mourad Ykhlef, (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 9, No. 9, 2018

[3]Journal, I. R. J. E. T. (2020). IRJET- A Review on Cyberbullying Detection using Machine Learning. IRJET. https://doi.org/10.1016/j.cose.2019.101710.

[4]Dewani, A., Memon, M.A. & Bhatti, S. Cyberbullying detection: advanced preprocessing techniques & deep learning architecture for Roman Urdu data. J Big Data 8, 160 (2021). https://doi.org/10.1186/s40537-021-00550-7

[5]Bayari, Reem & Bensefia, Ameur. (2021). Text Mining Techniques for Cyberbullying Detection: State of the Art. Advances in Science, Technology and Engineering Systems Journal. 6. 783-790. 10.25046/aj060187.
[6]Cyber Bullying Detection on Social Media using Machine Learning, Aditya Desai, Shashank Kalaskar, Omkar Kumbhar, Rashmi Dhumal, ITM Web Conf. 40 03038 (2021) DOI: 10.1051/itmconf/20214003038

[7]J. Atoum, "Cyberbullying Detection Through Sentiment Analysis," in 2020 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 2020 pp. 292-297.doi: 10.1109/CSCI51800.2020.00056

[8]Raj, Chahat & Agarwal, Ayush & Bharathy, Gnana & Narayan, Bhuva & Prasad, Mukesh. (2021). Cyberbullying Detection: Hybrid Models Based on Machine Learning and Natural Language Processing Techniques. Electronics. 10. 2810. 10.3390/electronics10222810.

[9]Cheng, Lu & Li, Jundong & Silva, Yasin & Hall, Deborah & Liu, Huan. (2018). XBully: Cyberbullying Detection within a Multi-Modal Context. 10.1145/3289600.3291037.

[10]Dadvar, Maral & de Jong, Franciska & Ordelman, Roeland & Trieschnigg, Dolf. (2012). Improved Cyberbullying Detection Using Gender Information. Cognitive Processing - COGN PROCESS.