Detecting Insults in Online Conversation: A Mache Learning Approach

Ohagwu C. A. & Sennaike O. A.
Department of Computer Sciences
University of Lagos
Akoka, Lagos State, Nigeria
E-mail: chinedu.ohagwu@gmail.com, osennaike@unilag.edu.ng
Phone: +2348165391233, +2348033322378

ABSTRACT

Huge volumes of data are generated daily from different daily activities and processing these data poses challenges. With increasing use of social media and online communities, online conversations are especially difficult to moderate giving room for insults, offensive language and cyber bullying. In this study we detect insults in online conversation using various machine learning algorithms. Data was collected from Twitter (twitter.com) and the data science competition portal, Kaggle (kaggle.com), pre-processed and presented to various machine learning algorithms, specifically Naïve Bayes, K-Nearest Neighbour (KNN), Logistic regression. Among other metrics, our results show that logistics regression algorithm performed best with an accuracy of 82.17%.

Keywords: Machine Learning, Text Mining, Naïve Bayes, K-Nearest Neighbour, Logistic regression.

1. BACKGROUND TO THE STUDY

The world is currently experiencing an uncontrollable and rapid growth in online discussion groups and reviews site. Examples include the rotten tomatoes' comment section, where the main aim is the sharing of personal reviews and ratings on movies and TV series. The description of these articles with their sentiments helps in the provision of concise summaries of readers thought generated by emotions and feelings; indeed, these features are also implemented in different applications like on twitter, The New York Times web page, among others. The anonymity of online communication makes it particularly prone to hostility. Unchecked data streams in online discussions make it possible for information passed from one party to another to be discrediting or defaming. This paper aims at applying machine learning algorithms in detecting insults in online conversations. Insult is an expression, statement, or sometimes behaviour which is disrespectful or scornful to the other party, this may be intentional or accidental. Some insults may be factual but at the same time pejorative.

Machine learning has been successfully implemented on data by various big companies like Google, Walmart, IBM, Facebook on different operations like predicting stock prize exchange, sentiment analysis, spam Filtering, recommender systems, among others. Machine learning can be divided into two major types, the supervised and unsupervised learning (Taiwo, 2010). Unsupervised learning algorithms find hidden patterns or intrinsic structures. It draws inferences from datasets consisting of input data without labelled responses. It finds clusters of similar inputs in the data without being explicitly informed about the classes the data points belong to.

Clusters are formed so that objects in the same cluster are very similar and objects in different clusters are very distinct. Clustering algorithm falls into two broad groups: hard clustering where each data points belongs to only one cluster and soft clustering where each data points can belong to more than one cluster. Examples of clustering algorithms include K-Means, Hierarchical, Self-organizing Maps, Fuzzy c-Means, etc. Supervised learning algorithms build a model that makes predictions based on evidence in the presence of uncertainty. A supervised learning algorithm takes a known set of input data and known responses to the data (output) and trains a model to generate reasonable predictions in response to new data. Supervised learning can be used to solve classification and regression problems. This paper focusses supervised learning algorithms, specifically the Naïve Bayes, K-Nearest Neighbour (KNN), Logistic Regression algorithms.

A naïve Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. It classifies new data based on the highest probability of its belonging to a particular class. It is best used for a small dataset containing many parameters when you need a classifier that's easy to interpret when the model will encounter scenarios that weren't in the training data, as is the case with many financial and medical applications. In machine learning, naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features (Hand & Yu, 2001).

KNN categorizes objects based on the classes of their nearest neighbours in the dataset. KNN predictions assume that objects near each other are similar. Nearest neighbour classification divides data into a test set and a training set. For each row of the test set, the K nearest (in Euclidean distance) training set objects are found, and the classification is determined by majority vote with ties broken at random. If there are ties for the Kth nearest vector, all candidates are included in the vote. It has been regarded as a lazy algorithm because the learning does not occur until the test example is given. This non-parametric machine learning algorithm learns by memorizing all data in the training sets. One major weakness is the runtime. To determine the nearest neighbour of a new point x, it must compute the distance of all m training examples.

The logistic regression machine learning algorithm is a model that can predict the probability of a binary response belonging to one class or the other. Because of its simplicity, logistic regression is commonly used as a starting point for binary classification problems. This is best used when data can be clearly separated by a single linear boundary and also as a baseline for evaluating more complex classification methods. Logistic regression belongs to the family of log linear classifiers known as the exponential or log-linear classifiers (Murphy, 2012). Like naive Bayes, this classifier works by extracting some set of weighted features from the input, taking logs, and combining them linearly (meaning that each feature is multiplied by a weight and then added up). It uses a logistic function also called the sigmoid function for classification. Input values are combined linearly using weights or coefficient values to predict an output value. A key difference from linear regression is that the output value being modelled is a binary value.

A number of approaches have been proposed to tackle the problem of the use of offensive words in online conversation. In (Mahmud et al., 2008) the authors created a set of rules to extract the semantic information of a given sentence from the general semantic structure of that sentence to separate information from abusive language. Chen et al. (2012) proposed the Lexical Syntactic Feature (LSF) architecture to detect offensive content and identify potential offensive users in social media. In (Ben Ismail and Bchir, 2015) the authors proposed an approach that automatically detect verbal offense in social network comments which relies on a local approach.

This approach adapts the fusion method to different regions of the feature space in order to classify comments from social networks as insult or not. In (Burnap and Williams, 2015) the authors used a combination of probabilistic, rule-based, and spatial-based classifiers with a voted ensemble meta-classifier to detect hateful or antagonistic tweets. In (Samghabadi et al., 2017), the authors pursued different NLP approaches to distinguish the use of swear words in a neutral way from those instances in which they are used in an insulting way.

1.1 Statement of Problem

Online conversation due to its anonymity can lead to peer to peer insults in the different platforms. Though the terms of service for social networking sites like twitter, Facebook, yahoo prohibits from posting content that is unlawful and abusive to users, posts are only partially filtered for some particular collection of offensive words. Also, while some sites like youtube and some newsgroups provide flag facility to mark content as insulting/inappropriate, they are prone to collusion and are highly misused (marking a non-insulting comment because it wasn't liked). Also, it is not possible to have a human moderator to review the comments before posting because of the increasing amount of online data. Hence, we need an automatic classifier that will detect the insulting comments accurately (Spertus, 1997).

1.2 Objective

The main objective of this study is to investigate the machine learning algorithms that will efficiently detect insults in in online conversations.

2. METHODOLOGY

The methodology is described in the following subsections.

2.1 Data Collection

The dataset was was formed from the combination of two datasets collected from kaggle.com and from twitter API. The dataset from kaggle was formed by different users where insults where being posted by participants for a cash prize. The twitter feed dataset was obtained from previous tweets and retweets of certain handles. The python library tweepy was used to extract tweets from twitter APIs, and it is also used to filter the tweets to get only the needed column (text) from the tweets. The filtered tweets were then labelled using 1 for insult and 0 for comments. The two datasets were combined for pre-processing totalling 4372.

2.1.2 Data Description

The dataset contains 4372 rows and 2 columns. The matrix contains an insult column labelled with 1 for insults and 0 form non-insults, the other column contain the actual comment in plain text.

	A	B	C	D	E	F	G	H	I	J
2617	1	rada rada shameless individual								
2618	0	lol i want to be sure that you would see the message before i block you								
2619	1	you people are just a bunch of whack idiots with no morality								
2620	0	you need cane'								
2621	1	everybody has issues so please calm down and take a sit and stop acting like a child								
2622	0	see ehn im not your boyfriend im not indebted to you in any way so calm down								
2623	1	you cant like speed darlington wtf is wrong with you								
2624	1	mojssa you dont have sense								
2625	1	the people that gave him microphone are mad								
2626	1	akiinwale very mad								
2627	0	trust her not								
2628	1	webster diamond reynolds handcuffed fuck nher daughter please dont scream because i dont want you to get s								
2629	1	looooooooooooooooooool trey and his lawyer are mad								
2630	1	what is wrong with you								
2631	1	hoe behavior								
2632	1	do you realize that you are not priority to me when im less busy ill check up on you dont feel entitled to shit								
2633	1	he is stupid								
2634	1	dammydr3zy my name is dammy oshodi i hate when people always try to be funny and ask oshodi oke or isale								
2635	1	we dont have thieves in our family								
2636	0	carry your wahala and be going								

Fig. 1: Dataset Snapshot

2.2 Data Pre-processing

Data pre-processing is a step-in data mining process. Since data gathering methods are often loosely controlled, resulting in unwanted and out of range value. Analysing data that has not been carefully screened for missing and unwanted values will lead to misleading results. So, it is ideal to carry out pre-processing before loading the datasets into the algorithm. The pre-processing steps taken on the dataset are outlined below

2.2.1. Removal of Delimiters

A delimiter is a sequence of one or more characters used to specify the boundary between separate, independent regions in plain text or other data stream. Since I was using a CSV File, which we only needed the commas, I had to remove other delimiters like [{:><%&””/; \}]. This process was done manually from the UTP-8 CSV file.

2.2.2. Tokenisation

Tokenization describes the general process of breaking down a text corpus into individual elements that serve as input for various natural language processing algorithms. Usually, tokenization is accompanied by other optional processing steps, such as the removal of stop words and punctuation characters, stemming or lemmatizing, and the construction of n-grams. But we used stop-words removal and constructing n-grams. Stop words are words that are particularly common in a text corpus and thus considered as rather un-informative (e.g., so, and, or, the). In the n-gram model, a token can be defined as a sequence of n items.

The simplest case is the so-called unigram (1-gram) where each word consists of exactly one word, letter, or symbol.

2.2.3. The Bag of Words (BOW) Model

A commonly used model in natural language processing is the so-called bag of words model. The idea behind this model really is as simple as it sounds. The bag of words model come with vectorization, where the number of different words in a text document is stored.

After being tokenized, the dataset was transformed and stored in a matrix for use by the algorithms. The pre-processed dataset was used in training the Naïve Bayes, K-Nearest Neighbour, Logistic regression models. After training, the models were tested using

2.3 Training

The three algorithms were trained with the same dataset in three experiments. The first experiment trained the transformed data without removing the stop words and without using the N-gram model, this is labelled as raw data. The second experiment removed stop words without using the N-gram model this is labelled as raw data, stop words. The third experiment removed stop words and used the N-gram model this is labelled as raw data, stop words, N-gram.

2.4 Testing

Teat data was generated from the dataset using different percentages: 20, 30, 40 and 50 percent test data for each of the experiments.

3. RESULTS

The results of the experiments are presented below. Tables 1 to 12 show the various metrics used in the different experiments. The computed metrics are precision, recall, F1 score and support. The accuracy and the confusion matrix are also computed for each experiment.

The confusion matrix is presented as follows:

[[TP FP]
 [FN TN]]

where
TN - True Negatives are correct classification of insults as insults
TP – True Positives are correct classification of comments as comments
FP – False Positives are wrong classification of comments as insults and
FN – False negatives are wrong classification of insults as comments.

Table 1: Logistic Regression with 20% test data

The Shape of Train Data: (3497,) The Shape of Test Data: (875,) precision recall f1-score support 0 0.85 0.91 0.88 629 1 0.72 0.60 0.65 246 avg / total 0.82 0.82 0.82 875 Accuracy: 0.821714285714 Confusion Matrix [[571 58] [98 148]] Time Taken: 0.862262725830078 seconds Vector: Raw Data	The Shape of Train Data: (3497,) The Shape of Test Data: (875,) precision recall f1-score support 0 0.84 0.93 0.88 629 1 0.74 0.53 0.62 246 avg / total 0.81 0.82 0.81 875 Accuracy: 0.817142857143 Confusion Matrix [[584 45] [115 131]] Time Taken: 0.6474812030792236 seconds Vector: Raw, Stop-words	The Shape of Train Data: (3497,) The Shape of Test Data: (875,) precision recall f1-score support 0 0.83 0.94 0.88 629 1 0.78 0.50 0.61 246 avg / total 0.81 0.82 0.81 875 Accuracy: 0.819428571429 Confusion Matrix [[593 36] [122 124]] Time Taken: 1.5590100288391113 seconds Vector: Raw Data, Stop-words, N-gram

Table 2: Logistic Regression with 30% test data

The Shape of Train Data: (3060,) The Shape of Test Data: (1312,) precision recall f1-score support 0 0.84 0.91 0.87 929 1 0.72 0.58 0.64 383 avg / total 0.81 0.81 0.81 1312 Accuracy: 0.8125 Confusion Matrix [[843 86] [160 223]] Time Taken: 0.7145652770996094 seconds Vector: Raw Data	The Shape of Train Data: (3060,) The Shape of Test Data: (1312,) precision recall f1-score support 0 0.82 0.92 0.87 929 1 0.73 0.51 0.60 383 avg / total 0.79 0.80 0.79 1312 Accuracy: 0.801829268293 Confusion Matrix [[857 72] [188 195]] Time Taken: 0.6457514762878418 seconds Vector: Raw, Stop-words	The Shape of Train Data: (3060,) The Shape of Test Data: (1312,) precision recall f1-score support 0 0.81 0.94 0.87 929 1 0.76 0.47 0.58 383 avg / total 0.80 0.80 0.79 1312 Accuracy: 0.801829268293 Confusion Matrix [[871 58] [202 181]] Time Taken: 1.723799705505371 seconds Vector: Raw Data, Stop-words, N-gram

Table 3: Logistic Regression with 40% test data

The Shape of Train Data: (2623,) The Shape of Test Data: (1749,) precision recall f1-score support 0 0.84 0.91 0.88 1244 1 0.73 0.59 0.65 505 avg / total 0.81 0.82 0.81 1749 Accuracy: 0.817610062893 Confusion Matrix [[1133 111] [208 297]] Time Taken: 0.7240681648254395 seconds Vector: Raw Data	The Shape of Train Data: (2623,) The Shape of Test Data: (1749,) precision recall f1-score support 0 0.82 0.94 0.87 1244 1 0.76 0.49 0.60 505 avg / total 0.80 0.81 0.79 1749 Accuracy: 0.807318467696 Confusion Matrix [[1164 80] [257 248]] Time Taken: 0.6386792659759521 seconds Vector: Raw, Stop-words	The Shape of Train Data: (2623,) The Shape of Test Data: (1749,) precision recall f1-score support 0 0.82 0.94 0.87 1244 1 0.76 0.48 0.58 505 avg / total 0.80 0.81 0.79 1749 Accuracy: 0.805031446541 Confusion Matrix [[1168 76] [265 240]] Time Taken: 1.3633265495300293 seconds Vector: Raw Data, Stop-words, N-gram

Table 4: Logistic Regression with 50% test data

The Shape of Train Data: (2186,) The Shape of Test Data: (2186,) precision recall f1-score support 0 0.83 0.91 0.87 1557 1 0.71 0.55 0.62 629 avg / total 0.80 0.81 0.80 2186 Accuracy: 0.806953339433 Confusion Matrix [[1419 138] [284 345]] Time Taken: 0.701550483703613 seconds Vector: Raw Data	The Shape of Train Data: (2186,) The Shape of Test Data: (2186,) precision recall f1-score support 0 0.81 0.93 0.87 1557 1 0.73 0.45 0.56 629 avg / total 0.79 0.79 0.78 2186 Accuracy: 0.794144556267 Confusion Matrix [[1453 104] [346 283]] Time Taken: 0.576880931854248 seconds Vector: Raw, Stop-words	The Shape of Train Data: (2186,) The Shape of Test Data: (2186,) precision recall f1-score support 0 0.80 0.94 0.86 1557 1 0.73 0.42 0.53 629 avg / total 0.78 0.79 0.77 2186 Accuracy: 0.787282708143 Confusion Matrix [[1458 99] [366 263]] Time Taken: 1.243269443511963 seconds Vector: Raw Data, Stop-words, N-gram

Table 5: Naïve Bayes with 20% test data

The Shape of Train Data: (3497,) The Shape of Test Data: (875,) precision recall f1-score support 0 0.84 0.93 0.88 629 1 0.75 0.55 0.63 246 avg / total 0.81 0.82 0.81 875 Accuracy: 0.820571428571 Confusion Matrix [[583 46] [111 135]] Time Taken: 0.5661849975585938 seconds Vector: Raw Data	The Shape of Train Data: (3497,) The Shape of Test Data: (875,) precision recall f1-score support 0 0.83 0.91 0.87 629 1 0.70 0.54 0.61 246 avg / total 0.80 0.80 0.80 875 Accuracy: 0.804571428571 Confusion Matrix [[572 57] [114 132]] Time Taken: 0.8048675060272217 seconds Vector: Raw, Stop-words	The Shape of Train Data: (3497,) The Shape of Test Data: (875,) precision recall f1-score support 0 0.81 0.94 0.87 629 1 0.76 0.44 0.56 246 avg / total 0.80 0.80 0.79 875 Accuracy: 0.803428571429 Confusion Matrix [[594 35] [137 109]] Time Taken: 1.2519135475158691 seconds Vector: Raw Data, Stop-words, N-gram

Table 6: Naïve Bayes with 30% test data

The Shape of Train Data: (3060,) The Shape of Test Data: (1312,) precision recall f1-score support 0 0.82 0.92 0.87 929 1 0.72 0.52 0.61 383 avg / total 0.79 0.80 0.79 1312 Accuracy: 0.802591463415 Confusion Matrix [[853 76] [183 200]] Time Taken: 0.6559276580810547 seconds Vector: Raw Data	The Shape of Train Data: (3060,) The Shape of Test Data: (1312,) precision recall f1-score support 0 0.82 0.91 0.86 929 1 0.70 0.52 0.60 383 avg / total 0.79 0.79 0.78 1312 Accuracy: 0.794207317073 Confusion Matrix [[842 87] [183 200]] Time Taken: 0.5372874736785889 seconds Vector: Raw, Stop-words	The Shape of Train Data: (3497,) The Shape of Test Data: (875,) precision recall f1-score support 0 0.81 0.94 0.87 629 1 0.76 0.44 0.56 246 avg / total 0.80 0.80 0.79 875 Accuracy: 0.803428571429 Confusion Matrix [[594 35] [137 109]] Time Taken: 1.2519135475158691 seconds Vector: Raw Data, Stop-words, N-gram

Table 7: Naïve Bayes with 40% test data

The Shape of Train Data: (2623,) The Shape of Test Data: (1749,) precision recall f1-score support 0 0.82 0.92 0.87 1244 1 0.72 0.50 0.59 505 avg / total 0.79 0.80 0.79 1749 Accuracy: 0.799885648942 Confusion Matrix [[1147 97] [253 252]] Time Taken: 0.5708754062652588 seconds	The Shape of Train Data: (2623,) The Shape of Test Data: (1749,) precision recall f1-score support 0 0.82 0.91 0.86 1244 1 0.69 0.50 0.58 505 avg / total 0.78 0.79 0.78 1749 Accuracy: 0.7918810749 Confusion Matrix [[1130 114] [250 255]] Time Taken: 0.550513744354248 seconds	The Shape of Train Data: (2623,) The Shape of Test Data: (1749,) precision recall f1-score support 0 0.79 0.94 0.86 1244 1 0.75 0.40 0.52 505 avg / total 0.78 0.79 0.76 1749 Accuracy: 0.78730703259 Confusion Matrix [[1175 69] [303 202]] Time Taken: 1.113908052444458 seconds
Vector: Raw Data	Vector: Raw, Stop-words	Vector: Raw Data, Stop-words, N-gram

Table 8: Naïve Bayes with 50% test data

The Shape of Train Data: (2186,) The Shape of Test Data: (2186,) precision recall f1-score support 0 0.81 0.93 0.87 1557 1 0.74 0.48 0.58 629 avg / total 0.79 0.80 0.79 2186 Accuracy: 0.801006404392 Confusion Matrix [[1452 105] [330 299]] Time Taken: 0.5344107151031494 seconds	The Shape of Train Data: (2186,) The Shape of Test Data: (2186,) precision recall f1-score support 0 0.82 0.91 0.86 1557 1 0.69 0.50 0.58 629 avg / total 0.78 0.79 0.78 2186 Accuracy: 0.792772186642 Confusion Matrix [[1417 140] [313 316]] Time Taken: 0.516124963760376 seconds	The Shape of Train Data: (2186,) The Shape of Test Data: (2186,) precision recall f1-score support 0 0.79 0.95 0.86 1557 1 0.74 0.38 0.50 629 avg / total 0.78 0.78 0.76 2186 Accuracy: 0.78362305581 Confusion Matrix [[1475 82] [391 238]] Time Taken: 1.0615675449371338 seconds
Vector: Raw Data	Vector: Raw, Stop-words	Vector: Raw Data, Stop-word, N-gram

Table 9: K-Nearest neighbour with 20% test data

The Shape of Train Data: (3497,) The Shape of Test Data: (875,) precision recall f1-score support 0 0.77 0.91 0.83 629 1 0.57 0.29 0.38 246 avg / total 0.71 0.74 0.71 875 Accuracy: 0.738285714286 Confusion Matrix [[575 54] [175 71]] Time Taken: 0.8708081245422363 seconds	The Shape of Train Data: (3497,) The Shape of Test Data: (875,) precision recall f1-score support 0 0.76 0.95 0.84 629 1 0.64 0.22 0.33 246 avg / total 0.73 0.75 0.70 875 Accuracy: 0.746285714286 Confusion Matrix [[599 30] [192 54]] Time Taken: 0.6493363380432129 seconds	The Shape of Train Data: (3497,) The Shape of Test Data: (875,) precision recall f1-score support 0 0.76 0.97 0.85 629 1 0.70 0.20 0.31 246 avg / total 0.74 0.75 0.70 875 Accuracy: 0.750857142857 Confusion Matrix [[608 21] [197 49]] Time Taken: 1.3198819160461426 seconds
Vector: Raw Data	Vector: Raw, Stop-words	Vector: Raw Data, Stop-words, N-gram

Table 10: K-Nearest neighbour with 30% test data

The Shape of Train Data: (3060,) The Shape of Test Data: (1312,) precision recall f1-score support 0 0.76 0.92 0.83 929 1 0.60 0.28 0.38 383 avg / total 0.71 0.74 0.70 1312 Accuracy: 0.735518292683 Confusion Matrix [[857 72] [275 108]] Time Taken: 0.8786990642547607 seconds Vector: Raw Data	The Shape of Train Data: (3060,) The Shape of Test Data: (1312,) precision recall f1-score support 0 0.74 0.97 0.84 929 1 0.72 0.19 0.30 383 avg / total 0.74 0.74 0.68 1312 Accuracy: 0.741615853659 Confusion Matrix [[900 29] [310 73]] Time Taken: 0.7722015380859375 seconds Vector: Raw, Stop-words	The Shape of Train Data: (3060,) The Shape of Test Data: (1312,) precision recall f1-score support 0 0.74 0.97 0.84 929 1 0.72 0.18 0.28 383 avg / total 0.73 0.74 0.68 1312 Accuracy: 0.739329268293 Confusion Matrix [[902 27] [315 68]] Time Taken: 1.3642230033874512 seconds Vector: Raw Data, Stop-words, N-gram

Table 11: K-Nearest neighbour with 40% test data

The Shape of Train Data: (2623,) The Shape of Test Data: (1749,) precision recall f1-score support 0 0.75 0.75 0.75 1244 1 0.37 0.37 0.37 505 avg / total 0.64 0.64 0.64 1749 Accuracy: 0.639222412807 Confusion Matrix [[931 313] [318 187]] Time Taken: 0.9748997688293457 seconds Vector: Raw Data	The Shape of Train Data: (2623,) The Shape of Test Data: (1749,) precision recall f1-score support 0 0.74 0.94 0.83 1244 1 0.54 0.17 0.26 505 avg / total 0.68 0.72 0.66 1749 Accuracy: 0.718696397942 Confusion Matrix [[1169 75] [417 88]] Time Taken: 0.7082443237304688 seconds Vector: Raw, Stop-words	The Shape of Train Data: (2623,) The Shape of Test Data: (1749,) precision recall f1-score support 0 0.73 0.95 0.83 1244 1 0.54 0.16 0.24 505 avg / total 0.68 0.72 0.66 1749 Accuracy: 0.717552887364 Confusion Matrix [[1176 68] [426 79]] Time Taken: 1.4261677265167236 seconds Vector: Raw Data, Stop-words, N-gram

Table 12: K-Nearest neighbour with 50% test data

The Shape of Train Data: (2186,) The Shape of Test Data: (2186,) precision recall f1-score support 0 0.76 0.88 0.81 1557 1 0.50 0.30 0.37 629 avg / total 0.68 0.71 0.69 2186 Accuracy: 0.712717291857 Confusion Matrix [[1370 187] [441 188]] Time Taken: 1.0188274383544922 seconds Vector: Raw Data	The Shape of Train Data: (2186,) The Shape of Test Data: (2186,) precision recall f1-score support 0 0.74 0.95 0.84 1557 1 0.62 0.19 0.29 629 avg / total 0.71 0.73 0.68 2186 Accuracy: 0.732845379689 Confusion Matrix [[1482 75] [509 120]] Time Taken: 0.749117374420166 seconds Vector: Raw, Stop-words	The Shape of Train Data: (2186,) The Shape of Test Data: (2186,) precision recall f1-score support 0 0.73 0.97 0.83 1557 1 0.60 0.11 0.18 629 avg / total 0.69 0.72 0.65 2186 Accuracy: 0.722781335773 Confusion Matrix [[1512 45] [561 68]] Time Taken: 1.229177713394165 seconds Vector: Raw Data, Stop-words, N-gram

Tables 13 to 15 summarises the accuracy obtained for each of the experiments for different test sizes while the following charts gives the graphical summary of the best accuracy and time for the three models.

Table 13: Accuracy for Logistic Regression

Test size (%)	Kind of Data	Accuracy (%)
20	Raw Data	**82.17**
	Raw Data + Stop-Words	81.71
	Raw Data + Stop-Words + N-Gram	81.94
30	Raw Data	81.25
	Raw Data + Stop-Words	80.19
	Raw Data + Stop-Words + N-Gram	80.18
40	Raw Data	81.76
	Raw Data + Stop-Words	80.73
	Raw Data + Stop-Words + N-Gram	80.50
50	Raw Data	80.69
	Raw Data + Stop-Words	79.41
	Raw Data + Stop-Words + N-Gram	78.73

Table 14: Accuracy for Naïve Bayes

Test size (%)	Kind of Data	Accuracy (%)
20	Raw Data	80.05
	Raw Data + Stop-Words	**80.49**
	Raw Data + Stop-Words + N-Gram	80.34
30	Raw Data	80.25
	Raw Data + Stop-Words	79.42
	Raw Data + Stop-Words + N-Gram	78.81
40	Raw Data	80.00
	Raw Data + Stop-Words	79.19
	Raw Data + Stop-Words + N-Gram	78.73
50	Raw Data	80.10
	Raw Data + Stop-Words	79.28
	Raw Data + Stop-Words + N-Gram	78.36

Table 15: Accuracy for KNN

Test size (%)	Kind of Data	Accuracy (%)
20	Raw Data	73.80
	Raw Data + Stop-Words	74.63
	Raw Data + Stop-Words + N-Gram	**75.09**
30	Raw Data	73.55
	Raw Data + Stop-Words	74.16
	Raw Data + Stop-Words + N-Gram	73.93
40	Raw Data	63.92
	Raw Data + Stop-Words	71.87
	Raw Data + Stop-Words + N-Gram	71.76
50	Raw Data	71.27
	Raw Data + Stop-Words	73.28
	Raw Data + Stop-Words + N-Gram	72.28

The charts below in figures 2 and 3 give a graphical representation of the best accuracy of the algorithms and time taken.

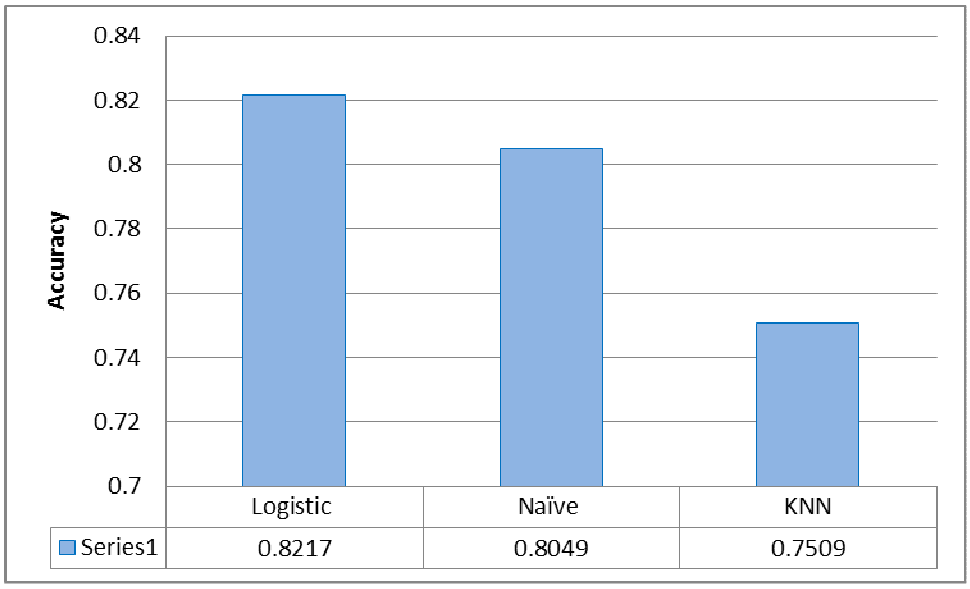

Fig. 2: Accuracy of the various algorithms

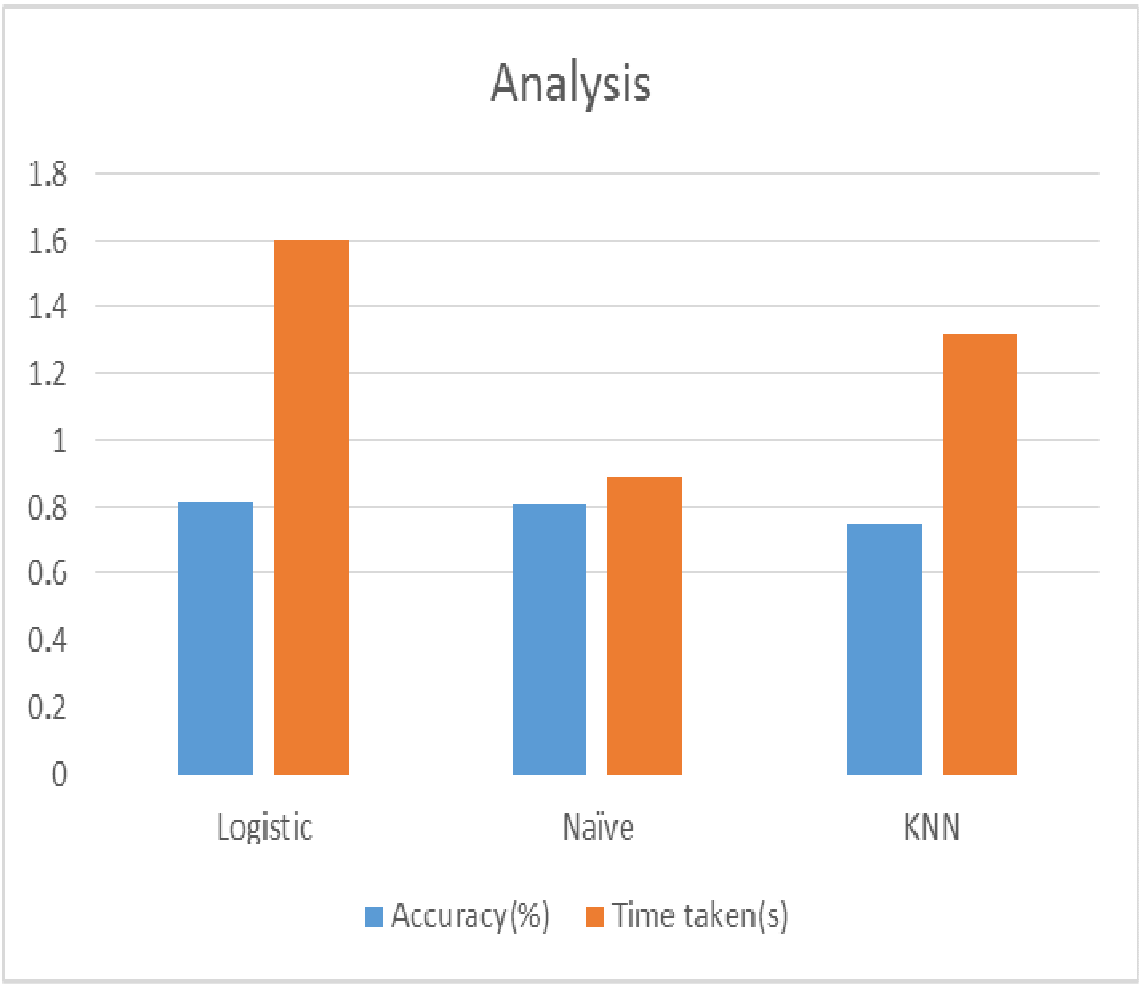

Fig. 3: Comparative Analysis of Accuracy and Time Taken

4. DISCUSSION OF FINDINGS

The logistic regression outperforms the Naïve Bayes and KNN in terms of accuracy. This is because of its discriminative mode and also the fact that it has to linearly classify data i.e. the decision boundary. The naïve Bayes was moderately accurate, and it has a better runtime than the other algorithms. The K-nearest neighbor did not perform well and this happened because it did not train itself before classification.

5. CONCLUDING REMARKS

The result of this study showed that the logistic regression model outperformed the Naïve Bayes and KNN in terms of accuracy in detecting insults. It is strongly recommended that social media and other platforms that support online conversations should implement an efficient machine learning algorithm to curb cyber bullying and reduce insults from users. This model can be extended to other areas like controlling adult content and sensitive materials online. Of particular interest is the development of a user configurable model that allows users determine the level and nature of sensitive content that is to be allowed to get to him or her. The implementation can be a web service that takes advantage of data from different sites or platforms without violating data privacy laws.

6. CONTRIBUTIONS TO KNOWLEDGE

This study lays a foundation to detecting insults in online conversations using machine learning algorithms and established the logistic model as an efficient algorithm to achieving safety in online conversations.

REFERENCES

1. Ben Ismail, M. M. and Bchir, O., (2015). Insult Detection in Social Network Comments Using Possibilistic Based Fusion Approach, R. Lee (ed.), Computer and Information Science, Studies in Computational Intelligence 566, DOI 10.1007/978-3-319-10509-3_2, Springer International Publishing Switzerland.
2. Chen, Y., Zhou, Y., Zhu, S., and Xu, H., (2012). Detecting Offensive Language in Social Media to Protect Adolescent Online Safety. In Proceedings of the 2012 ASE/IEEE International Conference on Social Computing and 2012 ASE/IEEE International Conference on Privacy, Security, Risk and Trust (SOCIALCOM-PASSAT'12). IEEE Computer Society, Washington, DC, USA, pp71-80. DOI=http://dx.doi.org/10.1109/SocialCom-PASSAT.2012.55
3. Hand, D., & Yu, K. (2001). Idiot's Bayes: Not So Stupid after All? International Statistical Review / Revue Internationale De Statistique, 69(3), 385-398. doi:10.2307/1403452
4. Mahmud, A., Ahmed, K. Z., and Khan, M. (2008). Detecting flames and insults in text, Proc. of 6th International Conference on Natural Language Processing (ICON' 08)
5. Murhy, K. P., (2002) Machine Learning: A Probabilistic Perspective, The MIT Press, Cambridge Massachusetts.
6. Samghabadi, N. S., Maharjan, S., Sprague, A., Diaz-Sprague, R., and Solorio, T., (2017). Detecting Nastiness in Social Media, Proceedings of the First Workshop on Abusive Language Online, pp 63–72, Vancouver, Canada, July 30 - August 4, 2017. Association for Computational Linguistics
7. Spertus, E., (1997). Smokey: Automatic Recognition of hostile messages. Proceedings of the fourteenth national conference on artificial intelligence and ninth conference on Innovative applications of artificial intelligence, (AAAI'97/IAAI'97). AAAI Press; pp1058-1065.
8. Taiwo, O. A. (2010). Types of Machine Learning Algorithms, New Advances in Machine Learning Yagang Zhang, IntechOpen, DOI: 10.5772/9385. Available from: https://www.intechopen.com/books/new-advances-in-machine-learning/types-of-machine-learning-algorithms
9. Burnap, P. and Williams, M. L. (2015), Cyber Hate Speech on Twitter: An Application of Machine Classification and Statistical Modeling for Policy and Decision Making. Policy & Internet, 7: 223-242. doi:10.1002/poi3.85