https://doi.org.10.3126/batuk.v10i1.62303
THE BATUK : A Peer Reviewed Journal of Interdisciplinary Studies
Vol. 10, Issue No. 1, Jan 2024, Page: 133-151
ISSN 2392-4802 eISSN 2565-4934
Received: 28 November 2023 Revised: 19 January 2024 Accepted: 24 January 2024

Importance of Data Preprocessing and Parameters Tuning for Supervised Machine Learning Models on Tweets Sentiment Analysis

Saurab Adhikari
Faculty, Nesfield International College
Email: saurab.add@gmail.com

Abstract

This paper shows the comparison of five different supervised machine learning models by showing the accuracy and classification report of these models when used for tweets sentiments analysis while showing the improvement in accuracy when data was preprocessed and parameters were tuned. The five different models that were used are: NaiveBayes, Support Vector Machine, Random Forest, Long Short-Term Memory (LSTM) and XG Boost. Total of 25000 tweets were processed, analyzed and predicted the output as positive, negative, or neutral using those models. This research would help to understand which models should be used and followed and which model would yield higher accuracy while using various approaches of data preprocessing and parameters tuning. The paper also tries to show that the standard models can still perform better and are still viable for sentiment analysis while SVM and Random Forest classifiers maybe viewed as standard learning strategies.

Keywords: machine learning, natural language processing, sentiment analysis, text analysis

Introduction

Social media is becoming the main platform to gather information about people's opinion and sentiments towards different facts and topics. Most of the people use social media such as Facebook and Twitter (lately known as X) to share their opinion. This research will show the importance of sentiments analysis by fetching Tweets from Twitter using hashtags and run sentiment analysis. Sentimental analysis has multiple applications in number of domains such as in businesses to get feedback and reviews for products by which sellers and producers can learn from user's feedback and reviews on social media.

Sentiment analysis is the process that automates mining of attitudes, opinions, views and emotions from text, speech, tweets and database sources. It is done through Natural Language Processing (NLP). Sentiment analysis means classifying opinions in the format of text into categories like "positive" or "negative" or "neutral". Sentiment analysis includes evaluation and examination of moral, emotional and sentiments states

and subjective information. It is commonly concerned with the user's opinion towards a subject; for example, surveys, reviews and polls on the web-based social networks. Sentimental analysis has multiple applications for number of domains such as in businesses to get feedback and reviews for products by which companies can check and know user's feedback and reviews on social medias.

There are various challenges related to sentiment analysis specially if the analysis is being done twitter data. The first challenge can be the circumstance under which the user has twitted something which could be difficult to figure out if that is in the negative sense or the positive. The second one could be incomplete tweets and use of abbreviations which would be also difficult to figure out. The third one could be the emoji which changes the meaning of the tweets entirely and also plays an important role in making the tweet sarcastic. The main aim of this research however was to compare five most popular machine learning models and how they predict the outcome. And also, to address some of the challenges mentioned earlier. This would help to the future researchers to understand and would also make them at ease on choosing the algorithms if they want to do research on the similar topic.

Sentiment analysis has caught attention of immense number of researchers and many different approaches have been implemented and introduced daily. Further, there are always some limitations in a research which opens the door for any new researcher to work on and contribute from his or her own part. This section explains mainly five researches, their methods and limitations. The classification of emoticons into seven categories have been one of the proposed methods to get better understanding of the emoticons. The classification can be done as strongly positive, positive, weakly positive, neutral, strongly negative, negative and weakly negative. [1]

Different researchers use different methods and algorithms but the use of Naive Bayes, Maximum Entropy, and SVM have achieved an accuracy of approximately 80 percent where n-gram and bigram model were utilized. Also, it has been found that Ensemble and hybrid-based Twitter sentiments analysis algorithms tended to perform better than supervised machine learning techniques, as they were able to achieve a classification accuracy of approximately 85 percent. [2] Machine learning methods, such as SVM and Naive Bayes have also been found to have highest accuracy and can be regarded as the baseline learning methods, while lexicon-based methods have been found very effective in some cases, which require few efforts in human-labeled document. [3] Researches have also been found where sentiment classifications have been improved on the translated data. The performance of machine learning models is generally measured by the precision, recall, and f1-score. Acceptable improvements have been recorded after

the translation and SVM have also been found to one of the best classifiers in multilingual tweets sentiments analysis. [4]

One of the researches have proposed a method using Naïve Bayes, KNN and modified k means clustering as these algorithms have been found to be more accurate than Naïve Bayes and KNN techniques individually as they have yielded an overall classification accuracy of 91 percent on the test set of 500 mobile reviews. The running time of their algorithm was O (n + V log V) for training and O (n) for testing, where n is the number of words in the documents (linear) and V the size of the reduced vocabulary. The proposed algorithm worked much faster than other machine learning algorithms like Naïve Bayes classification or Support Vector Machines which takes a long time to converge to the optimal set of weights. [5]

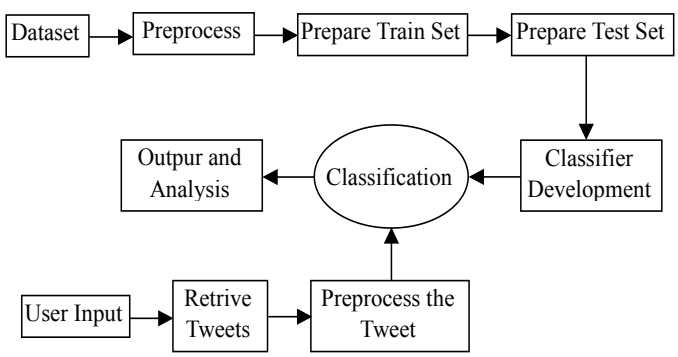
Hence, this research however considers the different types of algorithms from gradient boosting to recurrent neural network that is LSTM and instead removing or ignoring the emoticons, the emoticons are converted to plain text form using the demojize module so to train the models even better. The tweets were cleaned by removing the punctuations, special characters and stopwords while the blank tweets were not considered for the analysis.

Methods

The dataset was collected from Kaggle [6] from which 100,000 rows were used. The dataset consisted of 3 classes i.e. positive, negative and neutral and 2 columns that is text and sentiments. Five commonly used machine learning algorithms that are Naive Bayes (NB), Support Vector Machine (SVM), Random Forest (RF), Long Short-Term Memory (LSTM) and XG Boost were used for the research. Similarly, the implementation diagram of this research can be seen in Figure 1.

Figure 1

Implementation Diagram



Input: Give the input either the username or the hashtags and the number of tweets that you want to analyze the tweets.

Step 1: The models: Naive Bayes, Support Vector Machine, Random Forest, Long Short-Term Memory (LSTM) and XG Boost were developed using the predefined machine learning algorithms for the analysis.
Step 2: Dataset was divided into test and train data where 25 percent of data was used for the testing purpose.
Step 3: The data was pre-processed to remove the punctuation marks, emoticons, URLs, and stopwords.
Step 4: All those five models were trained using the collected dataset.
Step 5: The models then predicted the results for the test data as positive, negative, or neutral sentiments.
Step 6: The accuracy score and full classification score was retrieved for all the models.
Step 7: User input was taken as the twitter hashtag.
Step 8: Tweepy package was used to retrieve the tweets related to the given search word.
Step 9: The Retrieved tweets were pre-processed to remove the punctuation marks, emoticons, URLs, and stopwords.
Step 10: Then, these tweets were sent to the five machine learning models.
Step 11: The Classifiers then processed the features and then differentiated the sentiments as positive, negative and neutral sentiments.

Results

The accuracy scores achieved before preprocessing the data is presented in Table 1.

Table 1

Accuracy Scores of all the Models Before Datapreprocessing

Models	Accuracy
Random Forest	69.3
Support Vector Machine	72.83
Naïve Bayes	69.052
XG Boost	72.55
Long Short-Term Memory (LSTM)	68.74

The random forest model yielded the overall accuracy of 69.3 percent while the Naive

Bayes and LSTM models yielded the accuracy of 69.052 percent and 68.74 percent. The models with highest accuracy were XG Boost and SVM models with 72.55 percent and 72.83 percent. Similarly, for SVM model, two of the widely used kernels were used to analyse their performance and which one to follow finally. This showed that polynomial kernel performed better than linear kernel and so it was used for the SVM model.

For Naive Bayes model, two of the widely used kernels were used to analyse their performance and which one to follow finally. This showed that multinomial kernel performed better and so it was used for the Naive Bayes model. In addition, for Random Forest model, several tunings were done and after which gini criterion and n-estimators were the major parameters to achieve the overall accuracy better.

For LSTM model, several tunings were done and after which number of epochs was the major parameter to achieve the overall accuracy better. The number of epochs when increased to 25 achieved the highest accuracy. Lastly, for XG Boost model, several tunings were done and after which max-depth and n-estimators were the major parameters to achieve the overall accuracy better.

Data Preprocesing

For data cleaning, a custom function was used first which converted all the text to lower-case, removed brackets, numbers and punctuation.

A. Stop Word Removal: All the tweets under consideration were specifically in the English language. Therefore, stopwords corresponding to the English language were removed from the sentences. [7]

B. Stemming over Tweets: Stemming is the process of producing morphological variants of a root/base word. Porter stemming was used over the collected tweets. In Python, the Porter Stemming is done by using a method called PorterStemmer. [8]

C. Lemmatization: Lemmatization is an algorithmic method for finding the lemma of a word. That is, unlike stemming which may result in incorrect word removal. Lemmatization continuously decreases a word depending on its meaning. It makes a difference in returning the base or word reference shape of a word, which is known as the lemma. [9]

Support Vector Machine Model

SVM or Support Vector Machine is a linear model that is used for classification and also the regression problems. This model works well for many practical problems and is used to solve many linear and non-linear problems. SVM algorithm works by creating a line

or a hyperplane that separates the data or groups the data into different classes. [10]

- Linear Kernel:

The Linear Kernel should be used if the data is linearly separable. The linear kernel is considered to be fast and often produces irregular results.

For Linear Kernel the equation for prediction for a new input using the dot product between the input (x) and each support vector (xi) is calculated as follows:

$$f(x) = B(0) + sum(ai^* (x, xi))$$

This is an equation that involves calculating the inner products of a new input vector (x) with all support vectors in training data. The coefficients and ai (for each input) must be estimated from the training data by the learning algorithm.

Table 2

Confusion Matrix of SVM Model With Linear Kernel and Polynomial Kernel

		Actual Class					
		With Linear Kernel			With Plynomial Kernel		
		Negative	Neutral	Positive	Negative	Neutral	Positive
Predicted Class	Negative	6387	1017	1301	6402	739	896
	Neutral	686	6439	931	1144	6178	1443
	Positive	1288	914	6037	815	743	5930

The SVM model used with the linear kernel produced the accuracy score of 75.45 percent.

- Polynomial Kernel:

Then, the SVM model was passed or used with Polynomial Kernel. In general, the polynomial kernel is defined as;

$$K(X_1, X_2) = (a + X_1^T X_2)^b$$

b = degree of kernel & a = constant term. in the polynomial kernel, we simply calculate the dot product by increasing the power of the kernel.

At first, the SVM model used the linear kernel which had the accuracy score of 75.45 percent.

Then, the SVM model was passed or used with Polynomial kernel. The accuracy score of this model was 76.88 percent.

Table 3

Confusion Matrix of SVM Model With Polynomial Kernel

		Actual Class		
		Negative	Neutral	Positive
Predicted Class	Negative	6402	739	896
	Neutral	1144	6178	1443
	Positive	815	743	5930

Table 4

Classification Report of SVM Model With Polynomial Kernel

	Precision	Recall	F1-Score	Support
Negative	0.77	0.8	0.78	8037
Neutral	0.82	0.73	0.77	9475
Positive	0.72	0.79	0.75	7488
Accuracy			0.77	25000
Macro Average	0.77	0.77	0.77	25000
Weighted Average	0.77	0.77	0.77	25000

Naive Bayes Model

Naive Bayes is another supervised learning algorithm that works best for classification problems.

To make prediction on the target variable, naive bayes uses features similar to other supervised learning algorithms. The assumptions of Naive bayes is that the features are not related or dependent and there exists no correlation between the features. [11] The Naive Bayes theorem can be represented as:

$$P(y|X) = \frac{P((X|y) * P(y)}{P(X)}$$

Where,

- P(y|X): posterior probability of class (y, target) given predictor (X, attributes).
- P(y): is the prior probability of class. This is the observed probability of class out of all the observations.
- P(X|y): is the likelihood which is the probability of predictor-given class. This represents the probability of X being true, provided X is true.
- P(X): is the prior probability of predictor. This is the observed probability of predictor out of all the observations.

Gaussian Naïve Bayes

Gaussian Naive Bayes is a type of Naive Bayes which follows Gaussian normal distribution and supports continuous data. An assumption often taken while working with continuous data is that the continuous values associated with each class are distributed according to a normal (or Gaussian) distribution. The Bayesian theorem while working with statistics allows to calculate the probability that an event will occur provided that there is enough prior knowledge and information related to the specific event. [12] The Gaussian Naïve Bayes can be represented as:

$$P(X|Y) = \frac{P((Y|X)\,P(X)}{P(Y)}$$

Where,

- X and Y are two independent events, and P(Y) is not equal to zero- because we are dealing with a fraction formula, division by zero leads to no real answer.
- P(X/Y) is the probability that event X will occur if event Y is true.
- P(Y/X) is the probability that event Y will occur if event X is true.
- P(X) and P(Y) is the probability that event X and event Y will occur independently of each other.

The Gaussian Naïve Bayes model yielded the accuracy of only 37.8 percent. The classification report of the model can be shown in Table 6.

Multinomial Naive Bayes

Multinomial Naive Bayes is used for multinomially distributed data, and is one of the classic naive Bayes variants that used in text classification.

The multinomial Naive Bayes classifier is generally used for classification with discrete features (e.g., word counts) for text classification. The multinomial distribution requires feature counts in the form of integers. While in practice, tf-idf can also work that is fractional count. Parameters alpha float, default=1.0. [13]

The Multinomial Naïve Bayes accepts several parameters and some of them are:

Alpha. This accepts float values where the default value is 1.0. Consider you encounter word which is not there in your train set, then its probability of existence in a class becomes zero, which in return makes the whole probability 0, which is not good. So to avoid this situation, Laplace smoothing is used where we add alpha to numerator and the denominator field.

Fit-prior. This accepts boolean values as True or False where the default value is True. If false, a uniform prior will be used.

Class-prior. This accepts array-like values of shape (n-classes,) where the default value is None which is the prior probabilities of the classes. The priors are not adjusted according to the data if specified.

CountVectorizer. The CountVectorizer produces a matrix of token counts from a collection of text. It produces a sparse representation of the counts using scipy.sparse.csr-matrix.

n-grams. The lower and upper bounds of the n-value range for each type of word n-gram or char n-gram that needs to be extracted. All values of n such that min-n <= n <= max-n will be used. For example, an n-gram range of (1, 1) means only unigrams, (1, 2) means unigrams and bigrams, and (2, 2) means only bigrams. It is applicable only if analyzer is not callable. [14]

Several parameter tuning approaches and ideas were used to maximize the accuracy as far as possible. This also included cleaning the data before fitting into the model. Some of the major approaches and steps are described below:

At first, the accuracy of 58.78 percent was achieved when the alpha parameter was set 1.0, fit-prior was set False while class-prior was set None. The ngram range was passed as (3,3). Then, the default parameter values were passed to the model along with the n-gram range as (2,2). This improved the accuracy up to 62.47 percent. Then, the accuracy of 65.21 percent was achieved by passing the alpha parameter was set 1.0, fit-prior was set False while class-prior was set None. The ngram range was passed as (3,3). Lastly, the 'vect-ngram-range' was set as [(1, 1), (1, 2), (2, 2)],'tfidf-use-idf' was set as (True, False),'tfidf-norm' was set as ('l1', 'l2') and 'clf-alpha' was set as [1, 1e-1, 1e-2]. This yielded the accuacy of 73.032 percent.

Table 5

Confusion Matrix of Multinomial Naïve Bayes Model

		Actual Class		
		Negative	Neutral	Positive
Predicted Class	Negative	6761	1182	1578
	Neutral	588	6178	853
	Positive	1012	1010	5838

Table 6

Classification Report of Gaussian Naïve Bayes and Multinomial Naïve Bayes Model

	Gaussian Naïve Bayes				Multinominal Naïve Bayes			
	Precision	Recall	F1-Score	Support	Precision	Recall	F1-Score	Support
Negative	0.01	0.71	0.02	136	0.81	0.71	0.71	9521
Neutral	0.11	0.98	0.19	894	0.74	0.81	0.81	7619
Positive	1	0.35	0.52	23970	0.71	0.74	0.74	7860
Accuracy			0.37	25000			0.75	25000
Macro Average	0.37	0.68	0.24	25000	0.75	0.75	0.75	25000
Weighted Average	0.96	0.37	0.5	25000	0.75	0.75	0.75	25000

Random Forest Model

Random forest methods consist of multiple decision trees, where each of the trees are trained on a slightly different set of observations so the term 'forest'. A limited number of features are considered Nodes in each tree are split by considering. The average of the predictions of all individual trees are extracted and then the final predictions are made. [15]

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

This equation represents the class and probability to get the Gini of each branch on a node, deciding which of the branches is more likely to happen. Here, pi stands for the relative frequency of the class being observed within the dataset and c shows the number of classes.

The n-estimator was set as 100 and rest were left as a default. This yielded the maximum accuracy score of 74.98 percent.

Table 7

Confusion Matrix of Random Forest Model

		Actual Class		
		Negative	Neutral	Positive
Predicted Class	Negative	6648	1468	2048
	Neutral	585	6070	792
	Positive	1053	796	5540

Table 8

Classification Report of Random Forest Model

	Precision	Recall	F1-Score	Support
Negative	0.8023	0.654	0.7207	10164
Neutral	0.7283	0.815	0.7693	7447
Positive	0.6611	0.75	0.7026	7389
Accuracy			0.7303	25000
Macro Average	0.7306	0.74	0.7309	25000
Weighted Average	0.7385	0.73	0.7298	25000

Random forest methods consist of multiple decision trees, where each of the trees are trained on a slightly different set of observations so the term 'forest'. The average of the predictions of all individual trees are extracted and then the final predictions are made. [16]

The default sklearn class called 'Random Forest Classifier' is used for this. This class accepts total 19 parameters.

Several parameter tuning approaches and ideas were used to maximize the ac-curacy as far as possible. This also included cleaning the data before fitting into the model. Some of the major approaches and steps are described as follows:

n-estimators was 200, criterion means 'gini' and max-depth was set as 3. This yielded to the accuracy score of 50.24 percent. Then, some parameters were tuned while leaving some parameters as default in the following manner:

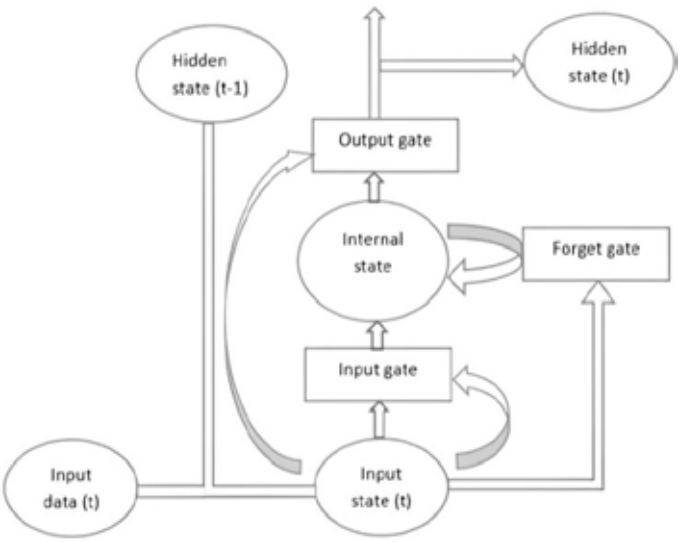
n-estimators was 100, criterion was 'gini', max-depth was None, min-samples-split was 2, min-samples-leaf was 1, min-weight-fraction-leaf was 0.0, max-features was 'auto', max-leaf-nodes was None, min-impurity-decrease was 0.0, min-impurity-split was None, bootstrap was True, oob-score was False, n-jobs was None, random-state was None, verbose was 0, warm-start was False, class-weight was None, ccp-alpha was 0.0 and max-samples was None. This yielded to the accuracy score of 75.25 percent.

Long Short-Term Memory (LSTM) Model

Long Short-Term Memory (LSTM) is a modified version of recurrent neural networks, that remembers past data in memory. This type of neural network resolves the vanishing gradient problem of RNN. They are well-suited to process, classify and predict for unknown duration the time series given time lags. Back-propagation is the method used by RNN to train the model. [17]

Figure 2

LSTM Architecture



Source. [18]

LSTMs are designed to capture long-term dependencies by default. The idea is that, in addition to learning general weights to choose which pieces of information to remember, the memory units in an LSTM learn what to remember.

There are three gates present in a LSTM network:

- Input gate — it discovers the input value that should be used to modify the memory.
- Forget gate — it discovers what details should the block discard as resulted by the sigmoid function. The gate shows the output as a number between 0 and 1 for each input in the cell state Ct−1 by looking at the previous state and the input given.
- Output gate — this gate decides the output by input and the memory of the block. The Sigmoid function decides the values to go through 0,1. The tanh function assigns weights to the values being passed by deciding their level of significance ranging from-1 to 1 and then they are multiplied by output of Sigmoid. [18]

The parameters, functions and attributes that were used to build this Sequential Model are:

The activation function was set as 'relu' which applies the rectified linear unit activation function. The dense function you can use to create a sequential model by passing a list of layers to the Sequential constructor.

Categorical cross entropy loss: If we use categorical cross entropy loss function, then we provide the ground truth as an n-dimensional vector in which all entries are 0 except the entry corresponding to the class, which is 1 (one-hot-encoding).

Sparse categorical cross entropy loss: If we use sparse categorical cross entropy then we provide the ground truth as single integer unit only rather than as an n-dimensional vector. Here the integer represents the class of the data.

Instead of the classical stochastic gradient descent procedure, Adam which is an optimization algorithm can be used to update network weights iterative based in training data. The epochs were increased up to 25 epochs which gave the overall accuracy of 72.7 percent.

Table 9

Classification Report of LSTM Model

	Precision	Recall	F1-Score	Support
Negative	0.73	0.68	0.7	8313
Neutral	0.75	0.81	0.78	8384
Positive	0.7	0.69	0.69	8303
Accuracy			0.73	25000
Macro Average	0.73	0.73	0.73	25000
Weighted Average	0.73	0.73	0.73	25000

A Sequential model is used for a plain stack of layers n which each layer has one input tensor and one output tensor exactly.

Arguments:

- inputs: A 3D tensor.
- mask: Binary tensor notifying if a given timestep should be masked.
- training: Python boolean indicating if the layer should be either in training mode or in the inference mode.
- initial-state: Initial state of tensors to be passed to the first call of the cell. [19]

At first, there were just 5 epochs used with the model. This gave the overall accuracy of 72.3 percent. Then, the epochs were increased to 15 epochs. This gave the overall accuracy of 72.5 percent. Lastly, the epochs were increased to 25 epochs. This gave the overall accuracy of 72.7 percent.

Table 10

Confusion Matrix of LSTM Model

		Actual Class		
		Negative	Neutral	Positive
Predicted Class	Negative	5668	1076	1569
	Neutral	684	6821	879
	Positive	1438	1174	5691

XG Boost (eXtreme Gradient Boosting)

XG Boost stands for "Extreme Gradient Boosting" and it is an implementation of gradient boosting trees algorithm. The XG Boost is a popular supervised machine learning model with characteristics like computation speed, parallelization, and performance. [20]

Several parameter tuning approaches and ideas were used to maximize the accuracy as far as possible where max-depth was set to 100, silent was set to False and n-estimators was set to 500 while other parameters were set to default which yielded the overall accuracy of 76.564 percent.

Several parameter tuning approaches and ideas were used to maximize the accuracy as far as possible. This also included cleaning the data before fitting into the model. Some of the major approaches and steps are described as follows:

At first, max-depth was set to 25, silent was set as False and n-estimators were set to 400 while other parameters were set to default which yielded the overall accuracy of 76.2719 percent. Then, max-depth was set to 50, silent was set to False and n-estimators was set to 500 while other parameters were set to default which yielded the overall accuracy of 76.428 percent. Then, max-depth was set to 75, silent was set to False and n-estimators was set to 500 while other parameters were set to default which yielded the overall accuracy of 76.292 percent. Finally, max-depth was set to 100, silent was set to False and n-estimators was set to 500 while other parameters were set to default which yielded the overall accuracy of 76.564 percent.

Table 11

Confusion Matrix of XG Boost Model

		Actual Class		
		Negative	Neutral	Positive
Predicted Class	Negative	6551	859	1155
	Neutral	836	6537	1061
	Positive	974	974	6053

Table 12

Classification Report of XG Boost Model

	Precision	Recall	F1-Score	Support
Negative	0.78	0.76	0.77	8565
Neutral	0.78	0.78	0.78	8434
Positive	0.73	0.76	0.74	8001
Accuracy			0.77	25000
Macro Average	0.77	0.77	0.77	25000
Weighted Average	0.77	0.77	0.77	25000

Discussion

This section presents in brief discussion about the results that were achieved and also the meaning of all the scores achieved. The scores to check the models and its accuracy were Precision, Recall, F1-Score and Accuracy.

The precision, recall and f1-score of XG Boost and Support Vector Machine seems slightly higher compared to other models. The precision shown is the average of all the precision of positive, negative and neutral values. The recall is the average of all the recall of positive, negative and neutral values. While, the f1-score shown is the average of all the f1-scores of positive, negative and neutral values.

The precision score is achieved by dividing the total number of true positives by the sum of total true positives and the false positives. The recall score was calculated by dividing the total number of true positives by the total sum of all true positives and the false

negatives. The F1 score is defined as a weighted average of the precision and recall.

Table 13

Comparative Accuracy Scores of all the Models

Models	Accuracy
Random Forest	73.096
Support Vector Machine	76.88
Naïve Bayes	73.032
XG Boost	76.654
Long Short-Term Memory (LSTM)	72.7

All the models worked well with accuracy above 72 percent. However, Support Vector Machine, XG Boost and Random Forest worked even better with better accuracy. These models can be regarded as the standard models for sentiment analysis even though there are different approaches being developed day by day.

- ROC (*Receiver Operating Characteristic*) Curve of the models:

A ROC curve is developed by plotting the true positive rate (TPR) in Y-axis against the false positive rate (FPR) in the X-axis. The true positive rate represents the proportion of those observations which are correctly predicted to be positive out of all positive observations (TP/(TP + FN)). While, the false positive rate represents the proportion of all the observations that are incorrectly predicted as positive out of all the negative observations (FP/(TN + FP)). For example, in lab test, the true positive rate people who are correctly identified to be tested positive for a particular disease. [21]

ROC curve of two of the models having the highest accuracy is presented in Figure 3.

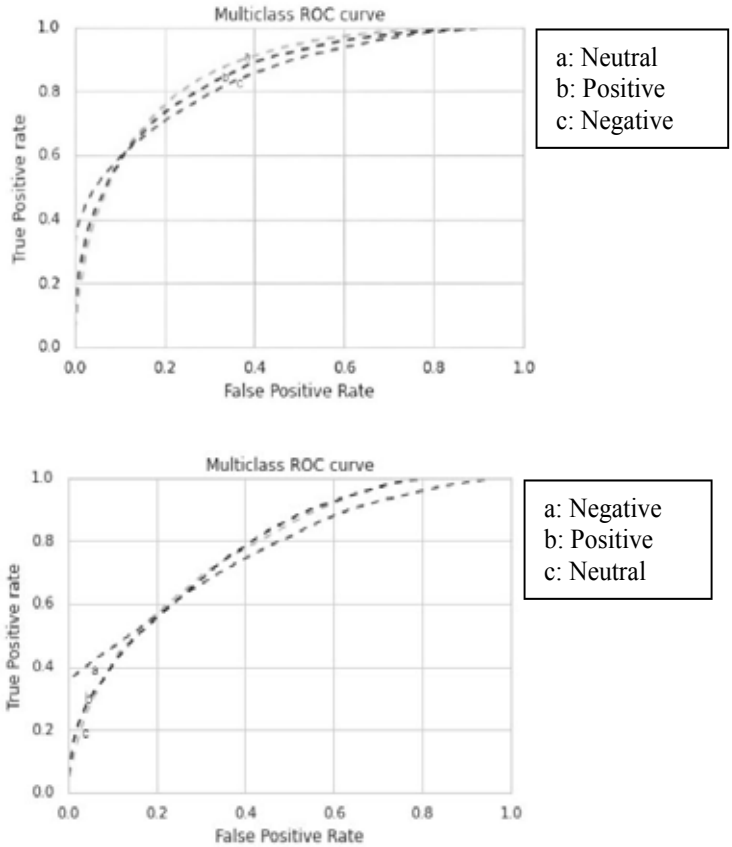
The multiclass ROC curve shows the trade-off between sensitivity (or TPR) and specificity (1 – FPR) while showing the comparison of one versus rest classes in a graphical form. Classifier is considered better if the curves are closer to the top-left corner indicate a better performance while the closer the curve comes to the 45-degree diagonal of the ROC space, the less accurate the test. As a baseline, a random classifier is expected to give points lying along the diagonal (FPR = TPR). [22]

The XG Boost models seems to be predicting the neutral sentiments more correctly than positive and negative sentiments while the SVM model seems to be predicting the negative sentiments more accurately than the positive and neutral sentiments. The altitude of the curves in the SVM model is at peak when the score rises above 0.6 until 0.8 while the altitude of the curves in the XG Boost model is at the peak when the score

goes past 0.4 and remains slightly stable until 0.6. This implies that both the models had higher True Positive rates comparatively.

Figure 3

Comparison: ROC Curve of SVM Model and XG Boost Model



As mentioned earlier, the test was conducted on 25,000 test data and all of the methods yielded the overall accuracy of around 72 percent. The XG Boost model and SVM model were the standout ones among the five considered models for the research.

Conclusion

The aim of performing the Twitter sentiments analysis was to investigate public's views poured via tweets and to build the models which would take Input as hashtag and predict the retrieved tweets as positive, negative, and neutral sentiments.

Five different machine learning algorithms for Twitter sentiments analysis methods were

used and their overall comparison has been shown. Research outcomes showed that machine learning algorithms such as SVM, LSTM, Random Forest and XG Boost produced the greatest precision, especially when multiple features were included. Classifiers such as SVM and Random Forest may be viewed as standard learning strategies while LSTM, Random Forest, XG Boost, and SVM all achieved an accuracy of approximately 75 percent.

There are many aspects left while conducting this research. Firstly, a new hybrid method can always be developed which may give higher accuracy. Secondly, there are only three classes considered for this analysis: positive, negative and neutral. So, increasing these classification classes can result in depicting the sentiments furthermore clearly and concisely. There were also some other limitations on the research which includes: no proper approach to handle incomplete tweets, only English language tweets could be classified and incapability to handle sarcastic tweets.

References

[1] Reddy, A. Brahmananda, D. N. Vasundhara and P. Subhash. "Sentiment Research on Twitter Data." *International Journal of Recent Technology and Engineering (IJRTE)* 8.2 (2019)

[2] Alsaeedi, Abdullah and Mohammad Siraj Khan. "A Study on Sentiment Analysis Techniques of Twitter Data." *International Journal of Advanced Computer Science and Applications* 10.2 (2019): 361-374.

[3] Kharde, Vishal A. and S. S. Sonawane. "Sentiment Analysis of Twitter Data: A Survey of Techniques." *International Journal of Computer Applications* 139.11 (2016): 5-15.

[4] Arun, K. and A. Srinagesh. "Multi-lingual Twitter sentiment analysis using machine learning." *International Journal of Electrical and Computer Engineering (IJECE)* 10.6 (2020): 5992-6000.

[5] Bharti, Onam and Monika Malhotra. "Sentiment Analysis on Twitter Data." *International Journal of Computer Science and Mobile Computing* 5.6 (2016): 601-609.

[6] "Sentiment140 dataset with 1.6 million tweets." https://www.kaggle.com, 2017

[7] “Stopwords in NLP” https://medium.com/@saitejaponugoti/stop-words-in-nlp-5b248dadad47

[8] “Stemming” https://towardsdatascience.com/stemming-corpus-with-nltk-7a6a6d02d3e5

[9] “Lemmatization” https://www.machinelearningplus.com/nlp/lemmatization-examples-python/

[10] "SUPPORT VECTOR MACHINES (SVM)" https://towardsdatascience.com/https-medium-com-pupalerushikesh-svm-f4b42800e989, 2018.

[11] "Naive Bayes" https://scikit-learn.org/stable/modules/naive-bayes.html

[12] “Gaussian Naive Bayes” https://iq.opengenus.org/gaussian-naive-bayes/

[13] Multinomial Naive Bayes Explained: Function, Advantages & Disadvantages, Applications in 2021 https://www.upgrad.com/blog/multinomial-naive-bayes-explained/

[14] “Multinomial NB” http://scikitlearn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html

[15] “RandomForest Classifier” https://scikitlearn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[16] "LSTM Layer" https://keras.io/api/layers/recurrent-layers/lstm/

[17] “BackPropagation” http://neuralnetworksanddeeplearning.com/chap2.html

[18] “Introduction to LSTM Units in RNN” https://www.pluralsight.com/guides/introduction-to-lstm-units-in-rnn

[19] “Sequence Models & Recurrent Neural Networks (RNNs)” https://towardsdatascience.com/sequence-models-and-recurrent-neural-networks-rnns-62cadeb4f1e1

[20] "XGBoostAlgorithm: LongMaySheReign" https://www.kdnuggets.com/2019/05/xgboost-algorithm.html 12

[21] “Classification: ROC Curve and AUC” https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

[22] “AUC ROC score and curve in multiclass classification problems” https://inblog.in/AUC-ROC-score-and-curve-in-multiclass-classification-problems-2ja4jOHb2X .