Sentiment Analysis: Understanding Social Sentiments on Twitter

Debduti Sengupta
Faculty of Science
Western University
London,Canada
dsengup2@uwo.ca

Parth Sawhney
Faculty of Science
Western University
London,Canada
psawhne3@uwo.ca

Shivanika Shah
Faculty of Science
Western University
London,Canada
sshah447@uwo.ca

***Abstract* - Social media such as Twitter have emerged as one of the primary channels of communication as well as information sharing. Tweets are an effective, concise and fast method to share information such as general news, opinions, personal updates, reviews amongst many other things. Often, the public sentiment around certain topics can be gauged by looking at the tweets on those topics or any topic that is trending.**
In this report, we are trying to assess the sentiments of tweets,either negative or positive. The dataset used is the Sentiment-140 dataset that was sourced from Kaggle. We preprocessed the data to convert the text into lower case, removed noise such as punctuation, URLS and hashtags. We then tokenized the text and lemmatized it. We use Logistic Regression, Support Vector Machine, Multinomial Naive Bayes, Adaboost and Random Forest models with TF-IDF vectorization and LSTM model with Word2Vec word embeddings. The LSTM model with 78.95% accuracy and 86.96% test AUC score is our final selection as the best performing model.

Keywords - Twitter Sentiment Analysis; Logistic Regression, Support Vector Machine, Multinomial Naive Bayes classifier, Random Forest, Adaboost classifier, Long Short Term Memory (LSTM), contraction, emoji classification, tokenization, stemming, lemmatization, stop words, count vectorizer, tf-idf vectorizer,word2vec, ROC curve, precision, recall, accuracy, python 3.

I. INTRODUCTION

The method of numerically assessing whether a text or a message is positive, negative or neutral is known as sentiment analysis. It's sometimes also called opinion mining as it involves determining a speaker's opinion or attitude.

Use cases of Sentiment Analysis

- Business: Companies use it in marketing to establish strategies, analyze customers' opinions, their response to campaigns, etc..
- Politics: It is used in politics to keep track of political opinions, to find inconsistencies between statements and government actions along with forecasting results of elections.
- Public Actions: Sentiment analysis is also used to track and analyze social events, identify potentially risky situations, and understand the mood of the people's opinion on the internet.

Twitter is a prominent social networking site where users may produce and communicate with "tweets". Individuals can use this to express their opinions or feelings regarding various topics. People have used sentiment analysis on tweets to acquire product information or perform market research. We can also improve the accuracy of our sentiment analysis forecasts using recent advances in machine learning techniques.

In this report, we use various machine learning techniques to conduct sentiment analysis on "tweets." We attempt to categorize the tweet's polarity as either positive or negative. If a tweet or text message contains both the positive and negative texts, the prevailing sentiment should be chosen as the final classification. We use a Kaggle dataset that has tweet texts categorized as positive or negative. The data includes emoticons, usernames, URLs and hashtags, all of which must be processed and translated to a simple standard format. Using the collected features, we undertake sentiment analysis using a variety of machine learning algorithms. However, depending solely on individual models did not result in high accuracy, therefore we selected the best models to create a model ensemble and recurrent neural networks. Ensembling is a type of meta learning algorithm technique in which many classifiers are

combined to improve prediction accuracy. A recurrent neural network is a form of Artificial Neural Network (ANN) utilized in Natural Language Processing (NLP) and Speech Recognition applications. An RNN model is intended to recognise data's sequential properties and then use the patterns to forecast the next situation. Finally, we present our experimental findings and results.

II. RELATED WORK

The research community is currently evaluating the analysis of feelings from Twitter, as its applications have a significant impact on how different industries operate today. The variety of speech and complex structure of data when extracted are the key challenges that this form of analysis faces.

Vinodhini and Chandrasekaran [2] evaluated the effectiveness of four classifiers in sentiment mining of online product reviews: K-Nearest Neighbors, Decision Trees, Naive Bayes, and Support Vector Machines. They created training examples from the product reviews dataset using various sampling strategies (e.g., linear sampling, bootstrap sampling, and random sampling). In terms of misclassification rate, their findings suggest that Support Vector Machine with bootstrap sampling method beats other classifiers and sampling methods. To populate the categorization input, they employed unigrams for feature space and word occurrences.

For sentiment classification, Xia et al. [3] used an ensemble framework. Various feature sets and classification techniques are combined to create an ensemble framework. To create the ensemble framework, they used two types of feature sets and three base classifiers. Part of speech information and Word-relations are used to construct two types of feature sets. The base classifiers used are Maximum Entropy, Naive Bayes, and SVMs. They improved sentiment classification accuracy by using different ensemble methods such as Fixed combination, Weighted combination, and Meta-classifier combination. Some studies have attempted to deduce public opinion about movies, news, and other topics from Twitter tweets.

Aliza Sarlan, Shuib, and Chayanit [4] conducted research on twitter data by retrieving tweets in JSON format and assigning polarity to them using a Python lexical dictionary. On the other hand, Faizan [6] examined methods for preprocessing and extracting twitter data using Python, as well as training and testing this data against a K-nearest neighbor classifier to determine the sentiments underlying tweets.

Work on sentiment analysis using the latest machine learning algorithms has been done in various different domains like politics, social media, etc. We observe that Naive Bayes gives better accuracy [4] than SVM, whereas LSTM outperforms many classifiers like Naive Bayes, etc.

III. DATA

Sentiment140 is the dataset utilized for carrying out the diverse range of experiments in this study and was taken from a public repository “Kaggle”. The dataset consists of 1.6 million tweets which were taken from the respective sources using Twitter search API. It is a well-balanced dataset containing 0.8 million positive and 0.8 million negative tweets. The tweets in this dataset are labeled as 0 indicating negative sentiment and 4 indicating positive sentiments. Manually annotating the tweets would have been a labor-intensive and time-consuming task due to the volume of tweets in this dataset. The authors of the dataset labeled the tweets by considering the emoticons’ noise for the prediction of the tweet as positive or negative. The dataset consists of six features described below:

	target	ids	date	flag	user	text
0	0	1467810369	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	_TheSpecialOne_	@switchfoot http://twitpic.com/2y1zl - Awww, t...
1	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by ...
2	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Man...
3	0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire
4	0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all....

Figure 1: Structure of the data

We can observe that the data in the text column was indeed from tweet messages posted on Twitter and their respective labels (0 or 4) were marked in the target column. As the dataset was too huge to compute, due to limited computational strength and time, we have taken a subset of this data consisting of 200k records for our modeling part. We have taken a well-balanced subset of this data as below consisting of 100k records of each label:

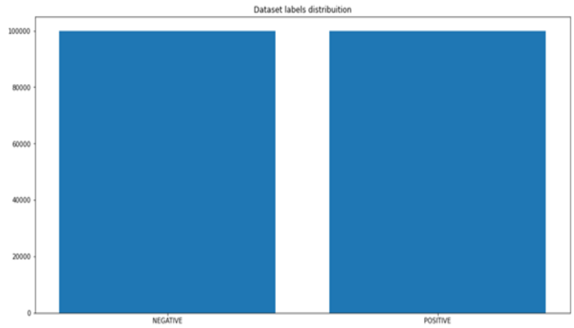


Figure 2: Dataset labels distribution

III. METHODS

1. DATA PREPROCESSING

Raw tweet data scraped from twitter is often noisy and contains data that may not necessarily hold much information regarding the sentiment of the tweet. Therefore, data cleaning of tweet texts is an essential and necessary step that needs to be taken prior to using it to create models. For our project, we have cleaned the tweet texts to accomplish the following actions:

A. URL

Hyperlinks and URLS do not convey much information regarding sentiments, they are thus removed.

B. USER MENTION

Twitter handles are prefixed with ‘@’. They are all removed..

C. HASHTAGS

Hashtags are used to mention trending topics, they are removed and replaced with just the word that follows the ‘#’ symbol..

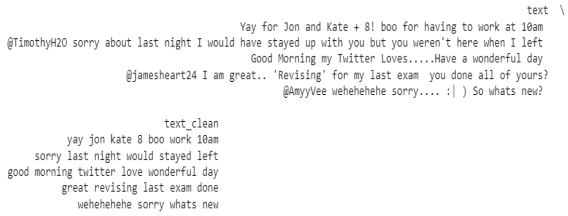
D. OTHER NOISE REMOVAL

Punctuations, repeated letters are removed.

E. STOPWORD REMOVAL

We use the ‘Stopword’ library from the NLTK package to remove English stopwords. Stopwords such as ‘The’, ‘is’, ‘are’ are commonly used words and make up a large portion of the vocabulary but do not hold much information pertaining to the sentiment of the tweet. They are thus removed.

After data cleaning, our cleaned up text looks like the following:



```
                                                                                   text  \
                                      Yay for Jon and Kate + 8! boo for having to work at 10am
@TimothyH2O sorry about last night I would have stayed up with you but you weren't here when I left
                                         Good Morning my Twitter Loves.....Have a wonderful day
                     @jamesheart24 I am great.. 'Revising' for my last exam  you done all of yours?
                                                   @AmyyVee wehehehehe sorry.... :| ) So whats new?

                      text_clean
    yay jon kate 8 boo work 10am
  sorry last night would stayed left
good morning twitter love wonderful day
    great revising last exam done
     wehehehehe sorry whats new
```

Figure 3 : Raw text vs clean text

F. LEMMATIZATION

Lemmatization uses the complete language vocabulary and does a morphological check to obtain the root word of the lemma. To do so, we need to access detailed dictionaries so that the algorithm can parse through it to derive the root word. The NLTK Wordnet library is used to lemmatize the tweets

Form	Morphological information	Lemma
reads	Third person, singular number, present tense of verb read	read
reading	Gerund/present continuous of the verb read	read

Table 1: Text words and its Lemma

G. FREQUENCY OF WORDS

We wanted to check the top 15 common words in tweets that are labeled Negative and Positive. For this, we used the cleaned up tweet text (as described in the previous sections).

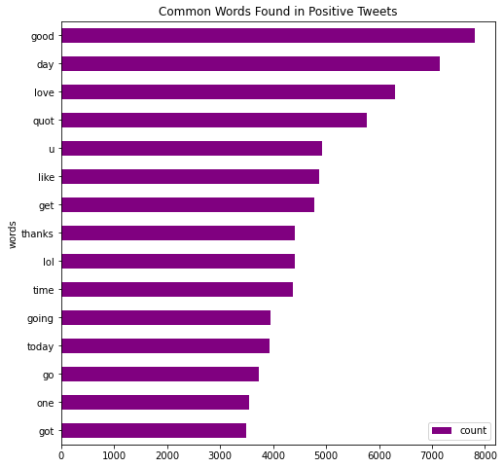


Figure 4a: Top 15 words in Positive Tweets

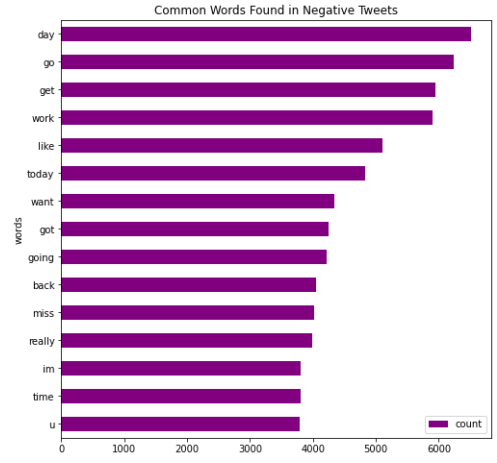


Figure 4b: Top 15 words Negative Tweets

As per expectation, the most common words do not carry any specific connotation to convey negative or positive sentiment, which is why, in the next section, we perform TF-IDF Vectorization.

H.TF-IDF VECTORIZATION

Term frequency counts how many times a term appears in a document divided by the total number of terms in that document.
TF = (Number of times term T appears in the particular document) / (number of terms in that document)
The more commonly occurring a word is, the less crucial information it contains. Therefore, the IDF of each word is the log of the ratio of the total
number of documents to the number of documents that contain the word.

IDF = log (N/n), where N stands for the total number of documents and n stands for the number of documents that contain the word.
TF-IDF is the product of the TF and IDF as calculated above.
We vectorized our corpus to calculate the TFIDF score of each word in our cleaned up tweet text.

2. MODELS

The project aims to build six different classification models, viz. Logistic Regression,Support Vector Machine,Multinomial Naive Bayes, Ada Boost, Random Forest and LSTM with Word2Vec embedding and to determine which of these performs better in terms of accuracy as well AUROC scores.

A. Logistic Regression

Logistic Regression is a regression model where the output is a categorical variable, unlike linear regression where the output is a continuous variable.

In binary Logistic Regression, the target variable can have two possible outcomes such True/False, Positive/Negative, 0/1 etc. The relationship between the response or target variable y and the predictor variables is defined using log odds. Thus, the log odds of y having a value 1, is defined as a linear equation as follows, where p is the probability of y being 1.

$$ln(\frac{p}{1-p}) = a + bx_1 + cx_2$$

We calculate the odds as follows(assuming x1 and x2 are two predictors , a is the intercept, b and c are regression coefficients of x1 and x2 respectively):

$$\frac{p}{1-p} = e^{(a+bx_1+cx_2)}$$
$$p = \frac{e^{(a+bx_1+cx_2)}}{1+e^{(a+bx_1+cx_2)}}$$
$$p = \frac{1}{1+e^{-(a+bx_1+cx_2)}}$$

Which is probability of y being 1. We decide on a threshold of decision, usually 0.5.If p is less than decision threshold, then y=0 and when p is greater than decision threshold, then y=1. Thus the equation for logistic regression is:

$$y = \frac{1}{1+e^{-(a+bx_1+cx_2)}}$$

We can generalize this equation for n number of predictors as follows:

$$y = \frac{1}{1+e^{-(\beta_0+\beta_1x_1+\beta_2x_2+\ldots+\beta_nx_n)}}$$

For our project, our target variable is the tweet sentiment. We use the LogisticRegressionCV algorithm from the ScikitLearn library for this purpose. We use a 5 fold cross validation, using 'accuracy' as our scoring metric and keep the C (parameter to control degree of regularization) at 2. The decision boundary threshold is set at 0.5, so any tweet,that has the 'probability' of having a 1 as the target (positive sentiment) will be classified as a 1 and any tweet with the probability of having a 1 as the target below 0.5 will be classified as 0.

B. Support Vector Machine

The "Support Vector Machine" (SVM) may be a supervised machine learning technique that may solve classification and regression problems. It is a non-probabilistic binary linear classifier algorithm within which each data item is plotted as a degree in n-dimensional space (where n is the number of features you have), and also the value of every feature is the value of a specific coordinate. Then we accomplish classification by locating the hyper-plane that clearly distinguishes the 2 classes. It uses the support vectors that are the coordinates of every individual observation.

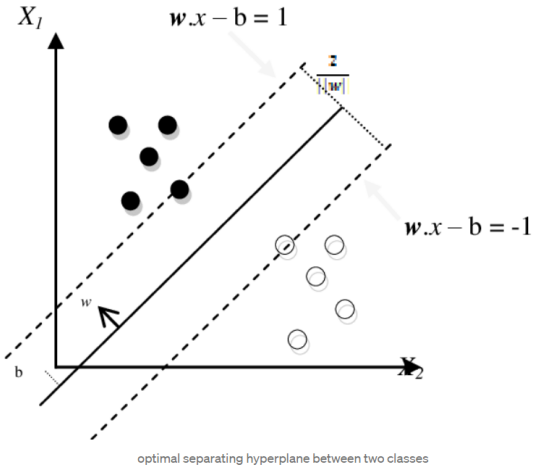


Figure 5: SVM Hyperplane

Mathematical Formulation:

For a training set of points (x_i , y_i) where x represents the feature vector and y is the class, the maximum-margin hyperplane that divides the points with $y_i = 1$ and $y_i = -1$. The hyperplane equation is -
$w \cdot x - b = 0$
We maximize the margin, denoted by γ, as follows -
$\max (w,\gamma)\ \gamma$, s.t. $\forall i, \gamma \leq y_i(w \cdot x_i + b)$
in order to separate the points well.

Kernel Trick

To classify nonlinear data by cleverly mapping the space to the next dimension might be computationally expensive: there are lots of recent dimensions, each of them possibly involving an advanced calculation. Doing this for each vector within the dataset may be plenty of labor. But using the kernel trick, SVM doesn't need the particular vectors, it actually can get by only with the dot products between them. This suggests that we are able to sidestep the expensive calculations of the new dimensions.

Kernels are the mathematical functions that take data as input and transform it into the required form as output. SVM algorithms use different types of kernel functions like linear, nonlinear, polynomial, radial basis function (RBF) and sigmoid.

Pros and Cons associated with SVM -

1. Pros:
    a. It works best when there's a definite dividing margin.

b. It works well in three-dimensional spaces.
c. When the amount of dimensions exceeds the amount of samples, this method works well.
d. It is memory efficient because it uses a subset of coaching points (called support vectors) within the decision function.

2. Cons:
a. When we have an oversized data collection, it doesn't perform well because the needed training time is longer.
b. When the information set contains more noise, like overlapping target classes, it doesn't perform well.
c. Probability estimates are produced using a fashionable five-fold cross-validation method, which isn't directly provided by SVM. It's a part of the Python scikit-learn library's related SVC algorithm.

Optimized SVM -

A mathematical model containing a variety of parameters that has got to be learned from data is said to be a Machine Learning model. There are, however, some factors called Hyperparameters that can't be learned directly. Before the particular training begins, individuals frequently choose to support intuition or trial and error. These factors demonstrate their value by enhancing the model's performance, like its complexity or learning rate. Models can include an outsized number of hyper-parameters, making determining the most effective combination of parameters an enquiry problem.

SVM also contains some hyper-parameters (such as kernel types, C, or gamma values) that may be discovered by experimenting with various combinations and seeing which of them work best. The first idea is to ascertain a grid of hyper-parameters and so just test all of their combinations (thus the name Gridsearch). GridSearchCV in Scikit-learn provides this feature. GridSearchCV uses a dictionary to specify the parameters which will be accustomed to train a model.

The three major parameters include:

1. Kernels: It takes low-dimensional input space and transforms it into a higher-dimensional space which is mainly useful in non-linear separation problems.

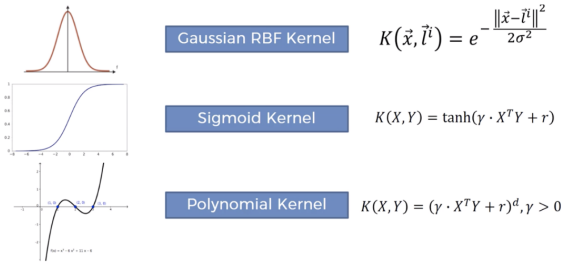


Figure 6: Different Kernels

2. C (Regularization): C is the penalty parameter, which represents misclassification or error term. It tells the SVM optimisation how much error is bearable and using which we can control the trade-off between decision boundary and the error term.

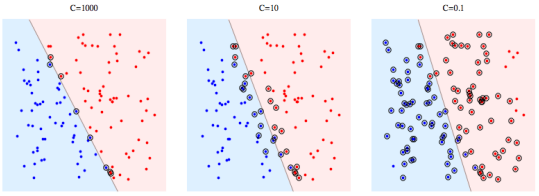


Figure 7: Classification with different Regularization Parameters

when C is high it will classify all the data points in the training set correctly increasing the chance to overfit.

3. Gamma: It defines how far it influences the calculation of plausible lines of separation.

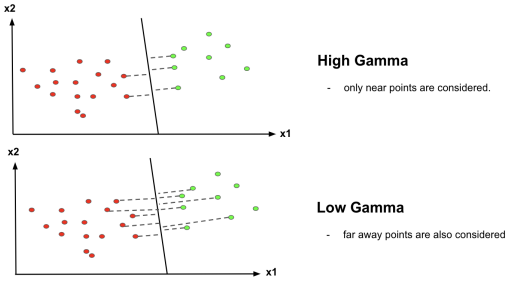


Figure 8: Effect of Gamma in Classification

Due to limited computational resources, GridSearchCV for SVM was performed with a smaller dataset and limited hyperparameter options -

Kernel - rbf
Gamma - auto
C - 0.1, 1, 10

It gave a very low accuracy which wasn't significant.

C. Multinomial Naive Bayes

Naive Bayes is a type of classification technique which is based on the Bayes theorem. Under supervised learning techniques, it is a basic yet effective algorithm for predictive modeling which is also easy to understand. When there are a lot of data points, Naive Bayes has better accuracy and speed. The Multinomial Naive Bayes model is an event-based model with features as vectors, where the sample(feature) denotes the frequency with which specific events have occurred.

For sentiment analysis applications, the multinomial Naive Bayes technique is often used as a starting point. The Naive Bayes technique works by using the joint probabilities of words and classes to find the probabilities of classes given to texts. This model calculates the probability of each tag for a given sample and outputs the tag with the highest probability. The Naive Bayes classifier is made up of several algorithms, all of which have one thing in common i.e., each feature being classified is unconnected to any other feature. As a result, the presence or absence of one attribute has no bearing on whether or not another is included.

When prior knowledge is available, the Bayes' theorem is used to figure out how likely a hypothesis is. Conditional probabilities are involved. The formula is as follows:

$$P(A|B)=\frac{P(B|A)P(A)}{P(B)}$$

For our case, we may reformulate the Bayes theorem using the naive assumption as below:

$$P(Sentiment|w_1,\ldots,w_n)=\frac{P(Sentiment)\prod_{i=1}^{n}P(w_i|Sentiment)}{P(w_1,\ldots,w_n)}$$

Probabilities are unimportant to us. All we care about is if a text has a positive or negative tone. We may completely ignore the denominator because it only scales the numerator:

$$P(Sentiment|w_1,\ldots,w_n)\propto P(Sentiment)\prod_{i=1}^{n}P(w_i|Sentiment)$$

For improving the performance of the model, we have implemented bagging with Multinomial Naive Bayes. Bagging classifiers are ensemble meta-estimators that fit base classifiers to arbitrary subsets of the original dataset then aggregate their individual projected values to reach a final prediction, either by voting or averaging. A meta-estimator can often be used to reduce the variance of an estimator by incorporating randomness into the estimator's creation mechanism and then creating an ensemble from it. Bagging reduces overfitting or variance by averaging or voting, but it also increases bias, which is countered by the reduction in variance. We have taken number of base classifiers=100 for our model as below:

```
# bagging classifier
model = BaggingClassifier(base_estimator = MultinomialNB(),
                          n_estimators = 100)
```

D. Random Forest

Ensemble classification methods are learning algorithms that construct a set of classifiers rather than a single classifier and then classify incoming data points based on the predictions of the classifiers. Random forest is a supervised machine learning technique based on ensemble learning. Ensemble learning is a type of machine learning in which multiple versions of the same algorithm are merged to create a more accurate prediction model. The random forest algorithm combines several methods of the same sort, such as numerous decision trees, to create a forest of trees, hence the name "Random Forest."

Random subspaces and "bagging" are combined in this approach. The decision tree forest approach uses numerous decision trees that are trained using data from slightly different subsets.

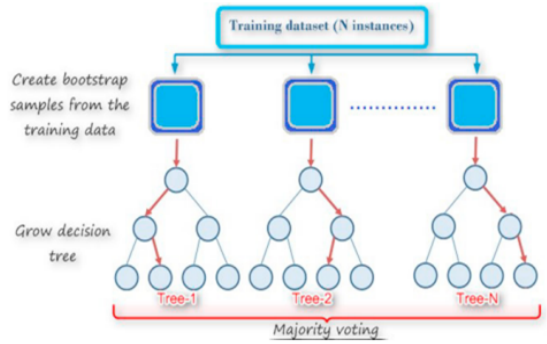


Figure 9: Picture representation of RF Algorithm

A group of tree-structured classifiers can be described as an RF classifier. It's a more complex variant of Bagging with the addition of randomization. RF splits each node using the best split among a group of predictors randomly picked at that node, rather than the best split among all variables.

The original data set is replaced to create a new training data set. Then, using random feature selection, a tree is grown. Pruning is not done on mature trees. This method gives RF a high level of precision. RF is also quick, resistant to overfitting, and allows the user to create as many trees as they wish.

The random forests algorithm is as follows:

1.Create ntree bootstrap samples from the source data.

2. Grow an unpruned classification or regression tree for each of the bootstrap samples, with the following modification- at each node, instead of choosing the best split among all predictors, randomly sample mtry of the predictors and choose the best split among those variables.

3. Aggregate the predictions of the ntree trees to predict for new data. For classification tasks, the majority vote is employed, while for regression, the average is used.

The Random Forest contains some parameters that we have fine-tuned using Grid Search. The hyper-parameters that are most important to be tuned are the no of decision trees in the forest, maximum depth of the individual trees, and minimum samples to split. After fine-tuning the hyperparameters, we have seen the accuracy of the model has increased. The best parameters selected by Grid Search was as below:

```
Best parameters are:
{'max_depth': None, 'n_estimators': 100}
```

E. Adaboost Classifier

AdaBoost (short for Adaptive Boosting) is an ensemble method of supervised learning algorithm. Ensemble learning is a learning technique in which multiple individual models, also known as base estimators, combine to create a master model. These algorithms improve the prediction power by converting several weak learners to strong learners. There are two kinds of Ensemble learning techniques- bagging and boosting. AdaBoost follows the boosting technique of learning. For boosting, multiple weak learners are trained in sequence. Each weak learner is assigned a final weight based on the accuracy of its classification. For our project, each base estimator/weak learner is a Decision tree stump, which is a Decision tree of depth 1.

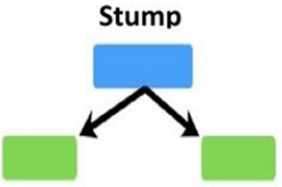


Figure 10: A decision tree stump

The goal is for each subsequent base estimator to learn from the errors of its predecessor and improve its prediction by minimizing classification error.

The main steps in AdaBoost Algorithm are Sampling,Training and Combination, as discussed below:

- Sampling step:

  Here, data samples (D_t) are selected from the dataset, where D_t is the set of samples in the iteration t. Samples are chosen based on either their Gini impurity or Entropy value, and the sample with lowest entropy gets chosen first. Initially each sample has same

weight:

$(w^1 = [w_1^1, \ldots, w_N^1], w_j^i \in [0,1], \sum_{j=1}^{N} w_j^i = 1)$, where w_j^i is weight of the j^{th} sample in the i^{th} iteration.

After each iteration, sample weights are updated as follows:

$$\alpha_t = \frac{1}{2} ln \frac{1-\epsilon_t}{\epsilon_t}$$

$$w_i^{t+1} = \frac{w_i^t}{Z_t} \begin{cases} exp(\alpha_t) & , \text{if } y_i \neq h_t(x_i) \\ exp(-\alpha_t) & , \end{cases}$$

Here alpha is the weight attached to each classifier, and calculated as follows:

Z_t is the normalization factor, calculated as

$$Z_t = \sum_{i=1}^{N} w_i^t exp(-\alpha_t y_i h_t(x_i))$$

- Training step: Here, different base estimators are trained using D_t, and the error rates (ϵ_i) for each base estimator is calculated.

$$\epsilon_t = \sum_{j=1}^{N} w_j^t l_j^t,$$

Where,

$$l_j^t = 1$$

If the estimator mis-classfies the datapoint or else it is 0.

Classifiers with higher error rates have lower weight or contribution to the final ensemble classifier. The relationship between classifier importance can be shown as follows

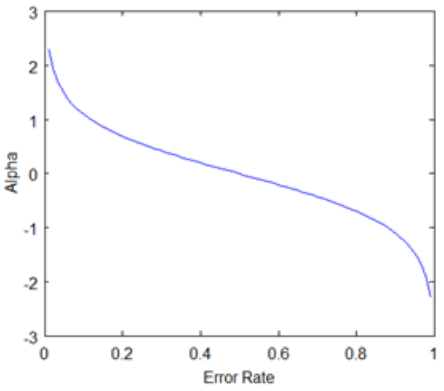


Figure 11: Relationship between alpha and Error rate

- Combination step: Here all trained models are combined to form the final model.

The final model is a weighted sum of all weak learners, weighted according to their alpha values.

$$H_{final} = sign(\sum_{t} \alpha_t h_t(x$$

And can be visualized as follows:

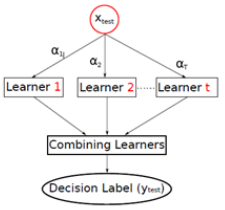


Figure 12: Adaboost combination step

HYPERPARAMETERS

We ran a grid search with 2 fold cross validation to determine the optimum value of hyperparameters, and we arrived at this combination of best hyperparameters:

```
[ ] print(loaded_model.best_params_)

    {'base_estimator__criterion': 'entropy', 'base_estimator__splitter': 'best', 'n_estimators': 200}
```

F. LSTM (Deep Neural Network)

Recurrent Neural Networks (RNNs) are a network of neuron-like nodes having a directed (one-way) connection to each other. In an RNN, the hidden state, designated by ht, acts because of the network's memory and learns contextual information that's critical for language categorization. At each step, the

output is calculated using the memory ht at time t and also the current input xt. An RNN's fundamental feature is its hidden state, which captures information's sequential reliance. In our trials, we used Long Short Term Memory (LSTM) networks, which are a sort of RNN capable of remembering information over a protracted period of your time.

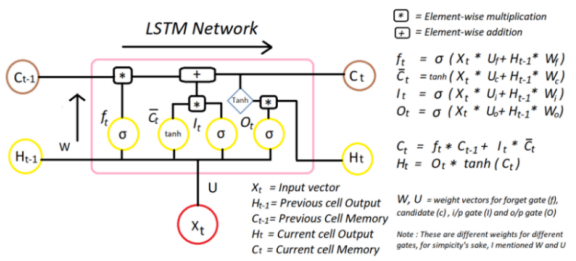


Figure 13: Architecture of LSTM

In a chain structure, the LSTM is created from four neural networks and a number of other memory blocks referred to as cells. A cell, an input gate, an output gate, and a forget gate compose a customary LSTM unit. Three gates govern the flow of data into and out of the cell, and also the cell remembers values across arbitrary time intervals. The LSTM algorithm is extremely fitted to categorizing, analyzing, and forecasting statistics of unknown duration.

The cells store information and the gates manipulate memory. There are three entrances

- Forget gate - whether to stay the knowledge from the previous timestamp or forget it
- Input gate - quantifies the importance of the new information carried by the input
- Output gate - determines the worth of the subsequent hidden state

LSTM Cycle

The LSTM cycle is divided as below:

- Forgotten information is identified from the prior step using the forget gate.
- Using input gate and tan h, new information is retrieved for updating the cell state.
- The information from the forget and input gates is then used to update the cell state.
- The output gate and the squashing operation provides useful information as output.

A dense layer receives the output of an LSTM cell after which the softmax activation function is applied to the output.

Word Embeddings

The word embedding techniques are used to represent words numerically, for example - One Hot Encoding, TF-IDF, Word2Vec, FastText. We choose the techniques according to the status, size and purpose of processing the data.

Word2Vec

Word2vec may be a collection of linked models for generating word embeddings. These are shallow two-layer neural networks that are trained to recreate word linguistic contexts. Word2vec takes an oversized corpus of text as input and outputs a vector space with several hundred dimensions, with each unique word allocated to an identical vector within the space. Word vectors are positioned in vector space in such the simplest way that words with similar contexts within the corpus are near each other.

Word2Vec's Advantages and Disadvantages

Compared to the bag of words and also the TF-IDF system, Word2Vec has significant advantages. The semantic meaning of distinct words in an exceedingly document is preserved by Word2Vec. there's no loss of context information. Another significant good thing about the Word2Vec technique is the tiny size of the embedding vector. Each dimension of the embedding vector represents a unique aspect of the word. Unlike the bag of words and TF-IDF techniques, we don't require large sparse vectors.

Model Building

A Keras sequential model is built with the linear stack of the following layers :

- An embedding layer of dimension 300 converts each word in the sentence into a fixed-length dense vector of size 300.
- Dropout layer of 50% units for regularization.
- A LSTM layer of 100 units.
- A dense layer of 1 unit and sigmoid activation outputs the probability of positive sentiments, i.e. if the label is 1.

The model is compiled with binary cross-entropy loss and adam optimizer. Since we've got a binary classification problem, binary cross-entropy loss is employed. The Adam optimizer with learning rate of 1e-2 uses stochastic gradient descent to coach deep learning models, and it compares each of the anticipated probabilities to the particular class label (0

or 1). Accuracy is employed because it is the primary performance metric.

The model summary is as below :

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, 300, 300)          41460900

 dropout_1 (Dropout)         (None, 300, 300)          0

 lstm_1 (LSTM)               (None, 100)               160400

 dense_1 (Dense)             (None, 1)                 101

=================================================================
Total params: 41,621,401
Trainable params: 160,501
Non-trainable params: 41,460,900
_________________________________________________________________
```

Features of LSTM:

- LSTMs are efficient for problems related to sequences and time series.
- The difficulty in training them is one of its disadvantages mainly as a hardware constraint, since even a simple model takes a lot of time and system resources to train.

IV. EXPERIMENTAL RESULTS

A. LOGISTIC REGRESSION

The Logistic Regression Model had a test AUC score of 75.56% and an accuracy of about 71%. The confusion matrix on test set is as follows:

True Label	Predicted Label	
	Negative	Positive
Negative	35.18%	
Positive	14.03%	

Table 2 : Confusion Matrix on Test Set for Logistic Regression

```
              precision    recall  f1-score   support

           0       0.71      0.71      0.71     29837
           1       0.71      0.72      0.72     30163

    accuracy                           0.71     60000
   macro avg       0.71      0.71      0.71     60000
weighted avg       0.71      0.71      0.71     60000


Train accuracy score:  0.8857142857142857
Test accuracy score:  0.7142333333333334

Train ROC-AUC score:  0.9590212433133498
Test ROC-AUC score:  0.7556527471531546
```

Figure 14: Accuracy , AUC-ROC for Logistic Regression

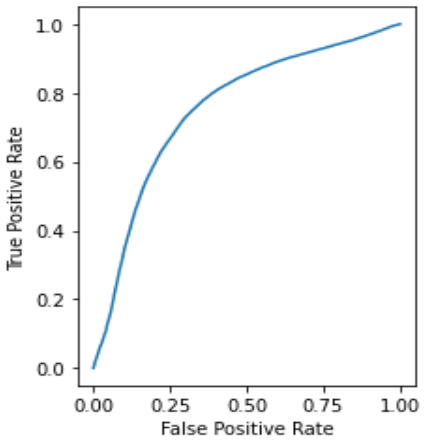


Figure 15: ROC curve for Logistic Regression

B. SVM

The experiments conducted by us showed a test AUC score of 75.2% and model accuracy of 75% for SVM (Support Vector Machine) classifier. The confusion matrix after completion of testing of classifier is given in Table 3 below:

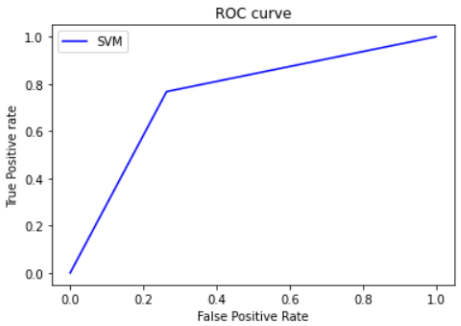
True Label	Predicted Label	
	Negative	Positive
Negative	36.62%	13.11%
Positive	11.68%	38.59%

Table 3: Confusion Matrix on Test Set for SVM

```
              precision    recall  f1-score   support

           0       0.76      0.74      0.75     29837
           1       0.75      0.77      0.76     30163

    accuracy                           0.75     60000
   macro avg       0.75      0.75      0.75     60000
weighted avg       0.75      0.75      0.75     60000
```




```
auc_score1 = roc_auc_score(y_test, y_pred2)

auc_score1
```

```
0.7520151636565369
```

Figure 16: ROC Curve and AUC score for SVM

C. MULTINOMIAL NAIVE BAYES

The Multinomial Naive Bayes Model had a test AUC score of 83.22% and an accuracy of about 74.8%. The confusion matrix on test set is as follows:

True Label	Predicted Label	
	Negative	Positive
Negative	38.27%	11.73%
Positive	13.45%	36.55%

Table 4: Confusion Matrix for MNB on test set

	precision	recall	f1-score	support
0	0.74	0.77	0.75	25000
1	0.76	0.73	0.74	25000
accuracy			0.75	50000
macro avg	0.75	0.75	0.75	50000
weighted avg	0.75	0.75	0.75	50000

```
Train accuracy score:  0.8202133333333333
Test accuracy score:  0.7482

Train ROC-AUC score:  0.8989827919111111
Test ROC-AUC score:  0.8322716

Are under Precision-Recall curve: 0.7437932437932437
Area under ROC-AUC: 0.8293274471978911
```

Figure 16: Accuracy and ROC-AUC score for MNB

D. RANDOM FOREST

The Random Forest Model had a test AUC score of 83.36% and an accuracy of about 75.61%. The confusion matrix on test set is as follows:

True Label	Predicted Label	
	Negative	Positive
Negative	37.46%	12.54%
Positive	11.84%	38.16%

Table 5: Confusion Matrix for RF on test set

	precision	recall	f1-score	support
0	0.76	0.75	0.75	25000
1	0.75	0.76	0.76	25000
accuracy			0.76	50000
macro avg	0.76	0.76	0.76	50000
weighted avg	0.76	0.76	0.76	50000

```
Train accuracy score:  0.99208
Test accuracy score:  0.75616

Train ROC-AUC score:  0.9988171013333332
Test ROC-AUC score:  0.8335705416

Are under Precision-Recall curve: 0.757845396043537
Area under ROC-AUC: 0.8271778819290335
```

Figure 17: Accuracy and ROC-AUC score for RF

E. ADABOOST

True Label	Predicted Label	
	Negative	Positive
Negative	34.54%	15.19%
Positive	15.10%	35.16%

Table 6 : Confusion Matrix for Adaboost on test set

	precision	recall	f1-score	support
0	0.70	0.69	0.70	29837
1	0.70	0.70	0.70	30163
accuracy			0.70	60000
macro avg	0.70	0.70	0.70	60000
weighted avg	0.70	0.70	0.70	60000

```
Train accuracy score:  0.9923357142857143
Test accuracy score:  0.6970166666666666

Train ROC-AUC score:  0.9998608812864805
Test ROC-AUC score:  0.7562866025319362
```

Figure 18: Accuracy and ROC-AUC for Adaboost

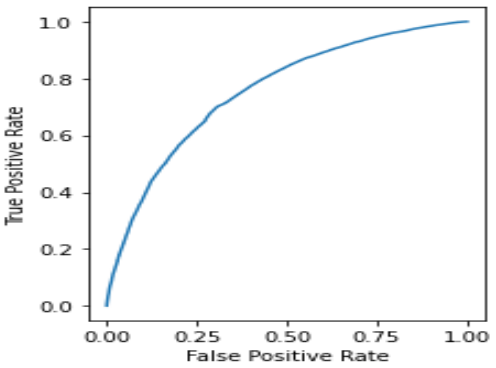


Figure 19:: ROC curve for Adaboost

F. LSTM

The experiments conducted by us showed a test AUC score of 86.96 % and model accuracy of 78.95 % for the LSTM (Long Short Term Memory) classifier. The confusion matrix after completion of testing of classifier is given in table 7 below

	Predicted Label	
True Label	Negative	Positive
Negative	79.00%	21.00%
Positive	21.00%	79.00%

Table 7: Confusion Matrix on Test Set for LSTM

▾ Classification Report

```
[ ] print(classification_report(y_test_1d, y_pred_1d))
```

```
              precision    recall  f1-score   support

    NEGATIVE       0.79      0.79      0.79     19921
    POSITIVE       0.79      0.79      0.79     20079

    accuracy                           0.79     40000
   macro avg       0.79      0.79      0.79     40000
weighted avg       0.79      0.79      0.79     40000
```

▾ Accuracy Score

```
[ ] accuracy_score(y_test_1d, y_pred_1d)
```

```
0.7895
```

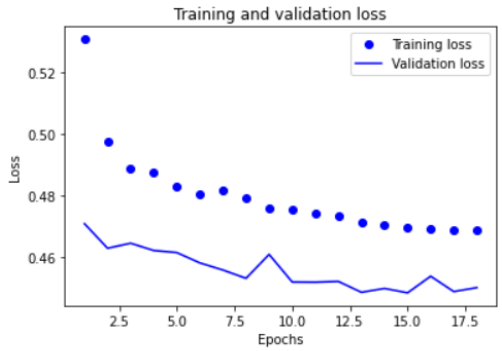


Figure 20 : Training and Validation Loss for LSTM

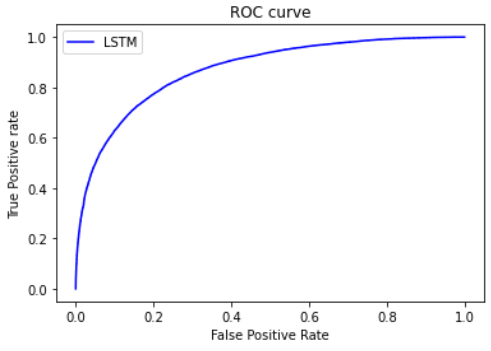


Figure 21: ROC Curve for LSTM

```
auc_score11 = roc_auc_score(y_test, y_pred2)

auc_score11
```

```
0.8696418110863575
```

V. CONCLUSIONS

In this project, we analyzed the textual twitter data in particular for the emerging field of sentiment analysis. We built models for the analysis of sentiments using Logistic Regression, SVM, Multinomial Naive Bayes, ensemble methods - Random Forest and Adaboost, and LSTM. We then performed a training and testing of this model on Twitter data set , for which we attained a maximum test AUC of **86.96 %** for LSTM. We are using Word2Vec embedding for LSTM , which retains the context of each word , and therefore is more powerful in understanding the sentiment of tweets. The model learns and retains the sequential context information and semantic similarity among words therefore resulting in higher accuracy as well as better AUC score for this model.However, LSTM is a complex and heavily resource intensive model with a large number of parameters, and takes quite long to train.

VI. FUTURE SCOPE

In the near future we aim to perform further analysis in order to increase the accuracy of the models and promote easy parallelizing of training data by using GloVE word embedding technique and different advanced deep learning methods. We also aim to handle sarcasms better by using a dataset where sarcasm tweets are defined as “/s” in texts and pre-processed accordingly.

VII. REFERENCES

1) https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/#
2) Vinodhini, G., & Chandrasekaran, R. M, "Performance Evaluation of Machine Learning Classifiers in Sentiment Mining," International Journal of Computer Trends and Technology (IJCTT), vol. 4, no. 6, 2013.

3) R. Xia, C. Zong, and S. Li, “Ensemble of feature sets and classification algorithms for sentiment classification,” Information Sciences: an International Journal, vol. 181, no. 6, pp. 1138–1152, 2011.
4) “Aliza sarlan, chayanit nadam, shuib basri ,” twitter sentiment analysis 2014 international conference on information technology and multimedia (ICIMU), november 18 – 20, 2014, putrajaya, malaysia 978-1-4799-5423-0/14/$31.00 ©2014 ieee 212”
5) “Mandava geetha bhargava , duvvada rajeswara rao , ” sentiment analysis on social media data using R” , international journal of engineering & technology, 7 (2.31) (2018) 80-84”
6) Faizan, “Twitter Sentiment Analysis”, International Journal of Innovative Science and Research Technology ISSN No:-2456-2165, Volume 4, Issue 2, February – 2019
7) https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/
8) https://data-flair.training/blogs/svm-kernel-functions/
9) https://www.vebuso.com/2020/03/svm-hyperparameter-tuning-using-gridsearchcv/
10) https://www.analyticsvidhya.com/blog/2022/03/an-overview-on-long-short-term-memory-lstm/
11) https://www.analyticsvidhya.com/blog/2022/01/sentiment-analysis-with-lstm/
12) https://www.analyticsvidhya.com/blog/2021/10/emotion-detection-using-bidirectional-lstm-and-word2vec/
13) https://stackabuse.com/implementing-word2vec-with-gensim-library-in-python/
14) https://towardsdatascience.com/word-embedding-techniques-word2vec-and-tf-idf-explained-c5d02e34d08
15) https://www.researchgate.net/publication/323119678_AdaBoost_classifier_an_overview
16) https://techvidvan.com/tutorials/r-logistic-regression/#:~:text=Logistic%20regression%20is%20a%20regression%20model%20where%20the,of%20%E2%80%9Cyes%E2%80%9D%20as%201%20and%20%E2%80%9Cno%E2%80%9D%20as%200.

VIII.APPENDIX

1) Kaggle Dataset https://www.kaggle.com/code/paoloripamonti/twitter-sentiment-analysis
2) Support Vector Machine Code https://colab.research.google.com/drive/1PSDOIXzM57qzdTL68Mqc0h4YymHwFwcW?usp=sharing
3) Long Short Term Memory Code -https://colab.research.google.com/drive/190TuYi1P9FGNklniKu3FzWcV1xbsPxSq?usp=sharing
4) Multinomial Naive Bayes and Random Forest https://colab.research.google.com/drive/1q7eoVjEsIa-f_SyAj4brRObo1wQgBbjW?usp=sharing
5) Logistic Regression and Adaboost https://colab.research.google.com/drive/1LNZEJOZeeBKhwwrvpe6IVLyUeR164QPd?usp=sharing