A Project Report on

Twitter Sentiment Analysis Using Machine Learning and NLP.

Submitted in partial fulfilment of the

Requirements for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

Information Technology

by

ADARSH CHOUDHARY	1811002
HANZALA SOHRAB	1811013

Under the supervision

Of

Mr. Dinesh Kumar Prabhakar

DEPARTMENT OF INFORMATION TECHNOLOGY

B.I.T. SINDRI, DHANBAD, JHARKHAND INDIA, PIN-828123

B.I.T. SINDRI

(DEPARTMENT OF SCIENCE & TECHNOLOGY, GOVT. OF JHARKHAND, RANCHI)

P.O. SINDRI INSTITUTE, DHANBAD-828123

DECLARATION

This is to certify that the dissertation bound herewith is an authentic record of synopsis work on "**Twitter Sentiment Analysis Using Machine Learning and NLP**" carried out by the following students under our guidance and supervision of partial fulfilment of the requirement for the award of B.Tech degree in Information Technology (session: 2018 - 2022) of B.I.T. Sindri, Dhanbad under the affiliation of Jharkhand University of Technology, Jharkhand.

STUDENTS	ROLL NO.
ADARSH CHOUDHARY	1811002
HANZALA SOHRAB	1811013

Mr. Dinesh Kumar Prabhakar	Prof. (Dr) S C Dutta
Assistant Professor	Head of Department,
Information Technology	Information Technology
(Signature of Supervisor)	(Signature of the HOD,IT)

B.I.T. SINDRI

CERTIFICATE OF APPROVAL

The forgoing thesis entitled " **Twitter Sentiment Analysis Using Machine Learning and NLP"** is here by approved as a creditable study of research topic carried out and has been presented in satisfactory manner to warrant its acceptance as prerequisite to the degree for which it has been submitted.

It is understood that by this approval, the undersigned do not necessarily endorse any conclusion drawn or opinion expressed there in, but approve the thesis for the purpose for which it is submitted.

Prof. (Dr) S C Dutta
Head of Department
Dept. of Information Technology
B.I.T. Sindri, Dhanbad

(External Examiner)

ACKNOWLEDGEMENT

It gives us immense pleasure to express our heartiest and sincere gratitude, indebtedness to our supervisor and mentor, Mr. Dinesh Kumar Prabhakar, Assistant Professor, Department of Information Technology, B.I.T. Sindri for his cooperation and tireless efforts in executing this project. His firm determination always boosted confidence in us. It was his constant hard work that helped us in giving this project a successful outcome.

We are extremely grateful to and would like to add our deep sense of gratitude to, Prof. (Dr) S C Dutta, Head of the Department, Information Technology, B.I.T. Sindri for his suggestions, cooperation and constant encouragement that helped us to grow personally and professionally.

Our heartfelt gratitude and appreciation is also to the administration, the workers and the lab assistants at department of Information Technology who were always there in need and support.

Date: 10.03.2022

ADARSH CHOUDHARY

HANZALA SOHRAB

ABSTRACT

Social media has created a new way for individuals to express their thoughts and opinions. This medium is used by an estimated 2.95 billion people worldwide, generating a massive platform for ideas to be shared. Sentiment analysis, or opinion analysis, is the process of retrieving textual information and discerning which emotions are exhibited by the author.

This project will perform sentiment analysis on real tweets harnessed from Twitter. The social networking service provides programmers with access to their data through its APIs (application programming interfaces). The primary aim is to provide a method for analysing sentiment scores in noisy twitter streams. This is done by classifying individual tweets as either negative, neutral, or positive. If a large enough collection of tweets is analysed, their collective sentiment score can then be used within a confidence range to state how the user pool feels towards the specific topic.

TABLE OF CONTENTS

1. INTRODUCTION

Twitter is a good source of information for individuals' opinions. Twitter receives about 500 million tweets a day, where people share comments regarding a wide range of topics. Many consumers take to Twitter to give their opinion on current events, including real-time affairs. By performing sentiment analysis on these tweets, one can determine the polarity and inclination of a population towards specific topics, items, or entities. On Twitter, users all over the world can express their opinion on any topic in a matter of seconds.

One of the earliest and most common forms of sentiment analysis is to conduct a bag-of-words analysis. This is when individual words are used as features, and individual tweets are the observations.

Bravo-Marques, Felipe, et al. (2015) combined this technique with a lower-dimension semantic vector to generate an opinion lexicon specifically oriented for Twitter posts. This is not always enough, though; many tweets include pictures, videos, GIFs, or other types of media that inherently add to the intended sentiment of the tweet.

Wang, Min, et al. (2014) modified the typical bag of text words to include a bag of image words. By using a cross-media bag of words model, they were able to improve the accuracy of standard text-only models by 4%. Others have tried to implement much more complex models, such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs).

A semi-automated method for creating sentiment dictionaries in several languages was suggested by Steinberger et al., (2012) and it yielded high-level sentiment dictionaries for two languages and automatic translation into a third language.

Ji et al., (2010) proposed a sentiment mining and retrieval system which mines useful knowledge from product reviews. Furthermore, the sentiment orientation and comparison between positive and negative evaluation were presented visually in the system. Outcomes of experiments on a real-world dataset have shown the system is both feasible and effective.

This project uses data of twitter to train and analyse the model.

1.1 Types Sentiment:

Basically, there are three types of sentiments — “positive”, “negative” and “neutral” along with more intense emotions like angry, happy and sad or interest or not interested etc. Further you can find here more refined sentiments used to analysis the sentiments of the people in different scenarios.

Emotion detection

This type of sentiment analysis aims to detect emotions, like happiness, frustration, anger, sadness, and so on. Many emotion detection systems use lexicons (i.e. lists of words and the emotions they convey) or complex machine learning algorithms.

One of the downsides of using lexicons is that people express emotions in different ways. Some words that typically express anger, like bad or kill (e.g. your product is so bad or your customer support is killing me) might also express happiness (e.g. this is bad ass or you are killing it).

Why Is Sentiment Analysis Important?

Sentiment analysis is extremely important because it helps businesses quickly understand the overall opinions of their customers. By automatically sorting the sentiment behind reviews, social media conversations, and more, you can make faster and more accurate decisions.

It's estimated that 90% of the world's data is unstructured, in other words it's unorganized. Huge volumes of unstructured business data are created every day: emails, support tickets, chats, social media conversations, surveys, articles, documents, etc). But it's hard to analyze for sentiment in a timely and efficient manner.

1.2 PROBLEM STATEMENT

We aim to create a machine learning model that is able to accurately classify the sentiment of a tweet. "Accuracy" for this model will be defined not only by its ability to correctly classify individual tweets, but also its ability to correctly classify large quantities of tweets to create an aggregated sentiment score.

Fig-1.1 Tweets indicating two emotions

Though the accuracy of the model on individual tweets will be limited by the subjective nature of this application, we believe it is reasonable to achieve very accurate aggregate scores. Once a model has been created, trained, and validated, we intend to use it on real-time and historical tweets to collect aggregated sentiment scores for varying topics. These can be used to compare real-time responses and attitudes towards different subjects

2. PROPOSED SOLUTION

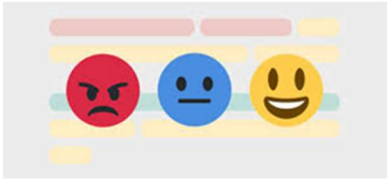


Fig-2.1 Three general emotions Negative, Neutral, Positive

The main goal of this project is to build and train a model that can detect the sentiment of a tweet. For simplicity, this model limit its scope to label all tweets into one of these three categories:

- Negative
- Neutral
- Positive

Though there are many more specific sentiments that would need to be learned in order to obtain a clearer picture of a tweet's intentions, we will limit our model to learning these three classifications. However, a perfect model is nearly impossible to create. The reason is because sentiment is highly subjective. Roughly 20% of tweets would cause a debate amongst humans as to which category they fall into. To illustrate this, consider the following two tweets:

- I love the world!
- I hate the world.

These two tweets clearly fall into the positive and negative, respectively, sentiment categories. However, consider this tweet:

- I am indifferent about the world.

This tweet lies somewhere in between the previous two. Yet, rather than labeling it as "neutral", one could argue that this in fact elicits a negative sentiment, as it is sad for someone to feel indifferent about the world. In any case, this classification is much less trivial to assign.

Fig-2.2 non-trivial tweet indicating mix emotion:

Another non-trivial tweet is

- The S&P 500 was down 300 points on Thursday.

This would come s bad news for investors, but happy news for short-sellers. But broadly, this tweet doesn't seem to have any emotion, but rather only delivers a fact. Lastly, consider one more tweet:

- I love candy, but it has too much sugar in it.

This person begins by saying something positive, but then ends it negatively. Overall, it is unclear whether this is a net-positive or net-negative tweet. It is also not as bland as the third example tweet; this sentence exhibits both positive and negative sentiments in one. The point of these examples is to show that sentiment analysis is not an exact science, and that this problem has an upper-bound. Even if a model had a 100% accuracy, humans would disagree with it about 20% of the time. Furthermore, not all sentiments are as easy to distinguish. A "neutral" label is much more subjective and poorly-defined than a "positive" or "negative" label.

3. PROJECT STRUCTURE

3.1 Data collection

To analyse any sentiment analysis, one needs large amounts of sentiment to test it on. These sentiments need to be sorted into three response; One of positive, negative and neutral. The data was provided from various sources like github user Tharindu Munasinge.This dataset contains 997 tweets that have a pre-labelled sentiment: 0 = negative ,2 = neutral ,4 = positive We have manually prepared the dataset to train the machine. The second dataset we will use in our training set comes from the twitter sentiment corpus created by Sanders Analytics. This dataset consists of 3424 relevant hand-classified tweets. Irrelevant observations in this dataset include Spanish content. For this project, we will restrict our model to only classify English tweets..

3.2 Methodology

First, we will use CountVectorizer to create a dictionary where the keys are every word in the training set and the values are the frequencies of each word.

bow = CountVectorizer(lowercase=True, strip_accents='ascii', stop_words='english’)

Next, we will convert the tweet text data (which is a 1D array) to an matrix, where n is the number of tweets in the training data and w is the number of words in the vocabulary dictionary. Each value in the matrix represents the number of times a given word appears in a given tweet. This will result in a very sparse dataset.

bow_matrix = bow.transform(X)

After this, we will use TfidfTransformer to calculate the term-frequency times inverse document-frequency (tf-idf) value for each word in the training set.

- term-frequency is the number of times the word appears in a tweet
- inverse document-frequency is the number of texts that contain the word

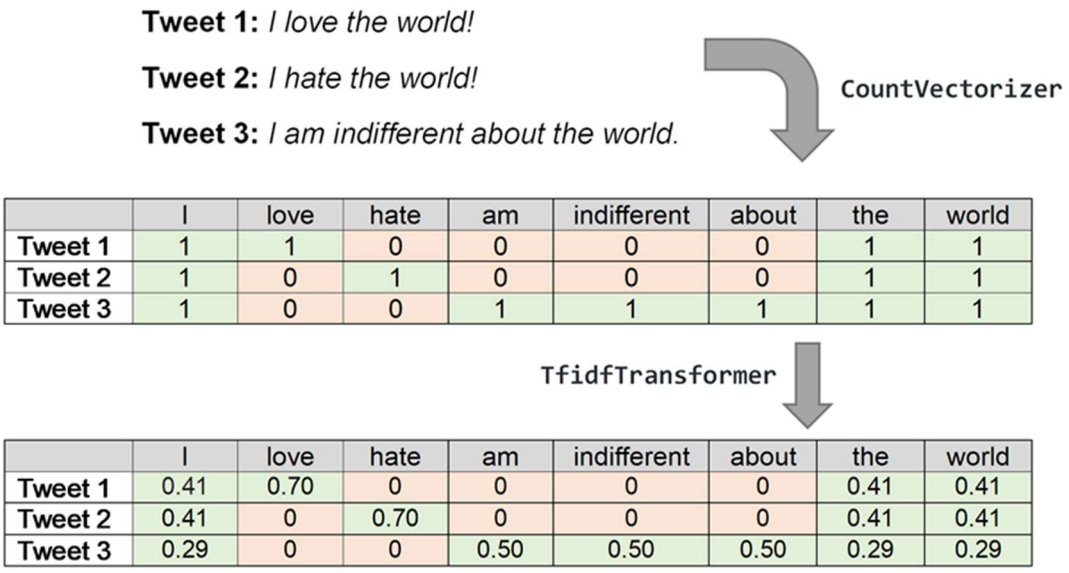
Hence, a word's tf-idf score increases proportionally to the number of times it appears in the text, and is offset by the total number of tweets that contain the word. This is used to determine the importance of a word in a given tweet. The more times the word appears in the tweet, the more important it must be to the sentiment of the tweet. However, if many tweets contain this word, it must not be as important in differentiating between sentiments.

The fit() method will learn the idf vector, which is the total number of tweets that contain each word. The transform() method will compute the tf vector and calculate the tf-idf matrix as.

```
tfidf_transformer = TfidfTransformer()
tfdif.fit(bow_matrix)
messages_tfidf = tfidf_transformer.transform(bow_matrix)
```

3.3 OVERALL METHODOLOGY

The following figure gives a visual representation of how these modules manipulate the data to prepare it for a classical estimator.



	I	love	hate	am	indifferent	about	the	world
Tweet 1	1	1	0	0	0	0	1	1
Tweet 2	1	0	1	0	0	0	1	1
Tweet 3	1	0	0	1	1	1	1	1

	I	love	hate	am	indifferent	about	the	world
Tweet 1	0.41	0.70	0	0	0	0	0.41	0.41
Tweet 2	0.41	0	0.70	0	0	0	0.41	0.41
Tweet 3	0.29	0	0	0.50	0.50	0.50	0.29	0.29

Figure 3.3.1: The above figure shows how CountVectorizer and TfidfTransformer convert a collection of words into a bag-of-words tf-idf matrix.

Finally, we will use MultinomialNB to train the tf-idf vectors with a Naive-Bayes classifier. This will be the final model that is used to classify tweet sentiment.

```
model = MultinomialNB().fit(messages_tfidf)
```

3.4 STREAMLINED METHODOLOGY

The TfidfVectorizer is a module which is equivalent to CountVectorizer followed by TfidfTransformer. This will convert raw text into a tf-idf matrix. Rather than have two separate objects, we can combine them into one.

This entire process can be further streamlined by using the Pipeline object. This object will conduct a series of fits and transforms. On different estimators. By setting up our Pipeline to first fit a TfidfVectorizer to our training data and then fit a MultinomialNB to the result, we have fully replicated our process in only one object. This will become extremely helpful when tuning our model.

```
pipeline = Pipeline([ ('tfidf', TfidfVectorizer(lowercase=True, strip_accents='ascii',

                                stop_words='english')), ('classifier', MultinomialNB()), ])
```

The first element of each tuple passed into the Pipeline is the name of the estimator. The second element of each tuple is the estimator itself, instantiated with any non-changing arguments.

3.5 CROSS VALIDATION

The benefit of using the Pipeline approach is that it allows us to apply cross validation on both of our estimators at once. Using GridSearchCV, we are able to create models that cycle through a given set of hyperparameters.

The parameters dictionary contains the set of hyperparameters to loop through. In our case, the hyperparameters of our model are the arguments that are passed into each estimator. The parameters dictionary is created using the following format:

```
parameters = {estimatorName__argumentName : [list_of_hyperparameters]}
```

```
parameters = { 'tfidf__ngram_range': [(1, 1), (1, 2)],   'tfidf__norm' : ['l1', 'l2'],

                'classifier__fit_prior' : [True, False]   ,   'classifier__alpha' : np.arange(0.1, 1.5, 16)

              }
```

We can then find and compute the best hyperparameters for our model by cycling through these arguments and using 10-fold cross validation. Using the optimal hyperparameters, we can then train our final model on the full training set.

```
grid = GridSearchCV(pipeline, cv=10, param_grid=parameters, verbose=1)

grid.fit(X, y)
```

3.6 TUNING AND TESTING

One last improvement we can make is to define a CLFSwitcher function that will allow us to use different classifier methods in our Pipeline process. This object needs to extend the fit(), predict(), predict_proba(), and score() methods of the classifier to be used. To implement this change, we need to make slight adjustments to our Pipeline process and parameters list.

```
pipeline = Pipeline([ ('tfidf', TfidfVectorizer(lowercase=True, strip_accents='ascii', stop_words='english')), ('clf', CLFSwitcher()), # Replace MultinomialNB to CLFSwitcher])
```

This is very helpful, because it allows us to not only pass in different estimators as a hyperparameter, but also because each estimator has unique hyperparameters of its own. The hyperparameters of a Naive Bayes classifier are different from those of a Support Vector classifier. With this wrapper class, we can pass in multiple estimators, as well as their unique parameters, to GridSearchCV when tuning our model. In our parameters list, we create multiple dictionaries -- one for each classifier. For a classifiers, the parameters list would look like

```
parameters = [
    {
     'clf__estimator': [MultinomialNB()],
    'tfidf__ngram_range': [(1, 1), (1, 2)],
    'tfidf__norm' : ['l1', 'l2'],
    'clf__estimator__fit_prior' : [True, False],
    'clf__estimator__alpha' : np.arange(0.1, 1.5, 16)
    },
    {'clf__estimator': [SGDClassifier(random_state=637, n_jobs=-1)],
    'tfidf__ngram_range': [(1, 1), (1, 2)],
    'tfidf__norm' : ['l1', 'l2'],
    'clf__estimator__loss': ['hinge', 'log', 'modified_huber'],
    'clf__estimator__penalty': ['l1', 'l2', 'elasticnet']
```

},

For this project, we will be cross validating with the following classifiers:

- Support Vector Machines
- Naive Bayes
- K-Nearest Neighbor
- Random Forests
- Logistic Regression

By cross validating among each of these classifiers (as well as cross validating among the hyperparameters of each estimator) we will be able to achieve the best model possible for our training data.

4. Some of the Basic concepts

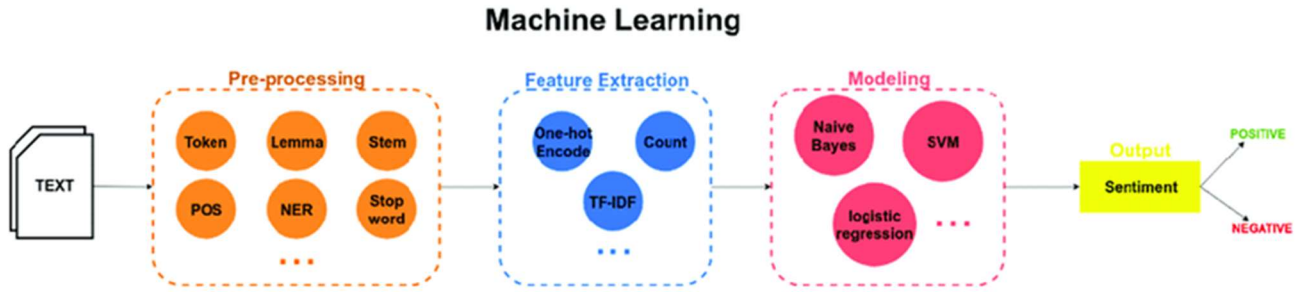


Fig-4.1: Flow diagram of model

4.1 Introduction to Sentiment Analysis

Sentiment analysis is the process of using natural language processing, text analysis, and statistics to analyse customer sentiment. The best businesses understand the sentiment of their customers—what people are saying, how they're saying it, and what they mean. Customer sentiment can be found in tweets, comments, reviews, or other places where people mention your brand. Sentiment Analysis is the domain of understanding these emotions with software, and it's a must-understand for developers and business leaders in a modern workplace.
As with many other fields, advances in deep learning have brought sentiment analysis into the foreground of cutting-edge algorithms. Today we use natural language processing, statistics, and text analysis to extract, and identify the sentiment of words into positive, negative, or neutral categories.

4.2 TERMINOLOGY

4.2.1 Tokenization

Tokenization is, generally, an early step in the NLP process, a step which splits longer strings of text into smaller pieces, or tokens. Larger chunks of text can be tokenized into sentences, sentences can be tokenized into words, etc. Further processing is generally performed after a piece of text has been appropriately tokenized.

4.2.2 Normalization

Before further processing, text needs to be normalized. Normalization generally refers to a series of related tasks meant to put all text on a level playing field: converting all text to the same case (upper or lower), removing punctuation, expanding contractions, converting numbers to their word equivalents, and so on. Normalization puts all words on equal footing, and allows processing to proceed uniformly.

4.2.3 Stemming

Stemming is the process of eliminating affixes (suffixed, prefixes, infixes, circumfixes) from a word in order to obtain a word stem.

running → run

4.2.4 Lemmatization

Lemmatization is related to stemming, differing in that lemmatization is able to capture canonical forms based on a word's lemma.

For example, stemming the word "better" would fail to return its citation form (another word for lemma); however, lemmatization would result in the following:

better → good

It should be easy to see why the implementation of a stemmer would be the less difficult feat of the two.

4.2.5 Corpus

In linguistics and NLP, corpus (literally Latin for body) refers to a collection of texts. Such collections may be formed of a single language of texts, or can span multiple languages -- there are numerous reasons for which multilingual corpora (the plural of corpus) may be useful. Corpora may also consist of themed texts (historical, Biblical, etc.). Corpora are generally solely used for statistical linguistic analysis and hypothesis testing.

4.2.6 Stop Words

Stop words are those words which are filtered out before further processing of text, since these words contribute little to overall meaning, given that they are generally the most common words in a language. For instance, "the," "and," and "a," while all required words in a particular passage, don't generally contribute greatly to one's understanding of content. As a simple example, the following pangram is just as legible if the stop words are removed:
The quick brown fox jumps over the lazy dog.

4.2.7 Parts-of-speech (POS) Tagging

POS tagging consists of assigning a category tag to the tokenized parts of a sentence. The most popular POS tagging would be identifying words as nouns, verbs, adjectives, etc.

4.2.8 Bag of Words

Bag of words is a particular representation model used to simplify the contents of a selection of text. The bag of words model omits grammar and word order, but is interested in the number of occurrences of words within the text. The ultimate representation of the text selection is that of a bag of words (bag referring to the set theory concept of multisets, which differ from simple sets).

Actual storage mechanisms for the bag of words representation can vary, but the following is a simple example using a dictionary for intuitiveness. Sample text:

"Well, well, well," said John. "There, there," said James. "There, there."

The resulting bag of words representation as a dictionary:

{

'well': 3,

'said':2,

'john': 1,

'there': 4,

'james': 1

}

4.2.9 N-Grams

N-grams are continuous sequences of words or symbols or tokens in a document. In technical terms, they can be defined as the neighbouring sequences of items in a document. They come into play when we deal with text data in NLP(Natural Language Processing) tasks.

4.2.10 Term Frequency (TF)

It measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear much more times in long documents than shorter

ones. Thus, the term frequency is often divided by the document length (aka. the total number of terms in the document) as a way of normalization:

$$\mathrm{TF}(t) = \frac{Number\ of\ times\ term\ t\ appears\ in\ a\ document}{Total\ number\ of\ terms\ in\ the\ document} \qquad \text{----------(4.1)}$$

4.2.11 Inverse Document Frequency(IDF)

It measures how important a term is. While computing TF, all terms are considered equally important. However, it is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. Thus we need to weigh down the frequent terms while scale up the rare ones, by computing the following: [SEP]

$$\mathrm{IDF(t)} = \log \frac{(Total\ number\ of\ document)}{(Number\ of\ documents\ wit\quad term\ t\ in\ it)} \qquad \text{----------(4.2)}$$

4.2.12 Tf-Idf

Tf-idf stands for term frequency-inverse document frequency, and the tf-idf weight is a weight often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus.

$$\mathrm{Wt,d} = \mathrm{TFt,d} \log \frac{(N)}{(DFt)} \qquad \text{----------(4.3)}$$

4.3 MACHINE LEARNING IN SENTIMENT ANALYSIS

Sentiment analysis is a machine learning tool that analyses texts for polarity, from positive to negative. By training machine learning tools with examples of emotions in text, machines automatically learn how to detect sentiment without human.

To put it simply, machine learning allows computers to learn new tasks without being expressly programmed to perform them. Sentiment analysis models can be trained to read beyond mere definitions, to understand things like, context, sarcasm, and misapplied words For example:

“Super user-friendly interface. Yeah right. An engineering degree would be helpful.”

Out of context, the words ‘super user-friendly’ and ‘helpful’ could be read as positive, but this is clearly a negative comment. Using sentiment analysis, computers can automatically process text data and understand it just as a human would, saving hundreds of employee hours.

Imagine using machine learning to process customer service tickets, categorize them in order of urgency, and automatically route them to the correct department or employee. Or, to analyse thousands of product reviews and social media posts to gauge brand sentiment.

There are three types of machine learning:-

Supervised Learning:

“The outcome or output for the given input is known before itself” and the machine must be able to map or assign the given input to the output. Multiple images of a cat, dog, orange, apple etc. here the images are labelled. It is fed into the machine for training and the machine must identify the same. Just like a human child is shown a cat and told so, when it sees a completely different cat among others still identifies it as a cat, the same method is employed here.

Unsupervised Learning:

“The outcome or output for the given inputs is unknown”, here input data is given and the model is run on it. The image or the input given are grouped together here and insights on the inputs can be found here (which is the most of the real-world data available).

Reinforced Learning:

The machine is exposed to an environment where it gets trained by trial and error method, here it is trained to make a much specific decision. The machine learns from past experience and tries to capture the best possible knowledge to make accurate decisions based on the feedback received.

The models used in e-mail classification are divided into following sections:

4.3.1 Naive Bayes

Bayesian Classification model, proposed by Thomas Bayes (1702 - 1761), is based on statistical and probabilistic method of learning. It is a type of supervised learning algorithm. Supervised learning is defined as the task of inferring a function from supervised training data [3]. A supervised training algorithm analyses the training data and comes up with the classifier function and the regression function. We are going to deal with classification here. The inferred classification function performs the task of predicting the correct output value for a given input

value. The Bayesian classifier assumes a probabilistic underlying model and determines probabilities of the outcomes. It can solve diagnostic as well as predictive problems. Naïve Bayes is based on conditional probabilities. Conditional probability is the probability of the occurrence of an event, given that another event has occurred. For implementing Bayesian Classification for e-mail spam filtering, I used the Bayes theorem. The Bayes theorem in terms of spam and ham emails can be expressed as:

$\Pr(S|W) = \Pr(W|S)\ \Pr(S)\ |\ ((\Pr(W|s)\ \Pr(S) + \Pr(W|H)\ \Pr(H))$ ----------(4.4)

Where Pr(S|W) is the probability that a message is spam knowing the presence of a given word in it.

Pr(W|S) is the probability that a certain word appears in spam messages.

Pr(S) is the overall probability of a message being a spam message.

Pr(W|H) is the probability that a certain word appears in ham messages.

Pr(H) is the overall probability that a message is a ham message.

[4].

Why use Naïve Bayes?

NB needs works well even with less sample data

NB is highly scalable. It scales linearly with the number of predictors and data points.

NB can be used for both binary and multi-class classification problems and handles continuous and discrete data

NB is not sensitive to irrelevant features.

NB is very simple, easy to implement and fast because essentially, you're just doing a bunch of counts.

4.3.2 Logistic Regression

Logistic regression analysis studies the association between a categorical dependent variable and a set of independent (explanatory) variables. The name logistic regression is used when the dependent variable has only two values, such as 0 and 1 or Yes and No. The name multinomial logistic regression is usually reserved for the case when the dependent variable has three or more unique values, such as Married, Single, Divorced, or Widowed. Although the type of data used for the dependent variable is different from that of multiple regression, the practical use of the procedure is similar. Logistic regression competes with discriminant analysis as a method for analysing categorical-response variables. This program computes binary logistic regression and multinomial logistic regression on both numeric and categorical independent variables. It reports on the regression equation as well as the goodness of fit, odds ratios, confidence limits,

likelihood, and deviance. It performs a comprehensive residual analysis including diagnostic residual reports and plots [5]. It can perform an independent variable subset selection search, looking for the best regression model with the fewest independent variables. It provides confidence intervals on predicted values, and provides ROC curves to help determine the best cut-off point for classification. It allows you to validate your results by automatically classifying rows that are not used during the analysis.

Threshold value is taken as 0.5.

If value is greater than threshold value, then output is 1.

If value is lesser than threshold value, then output is 0.

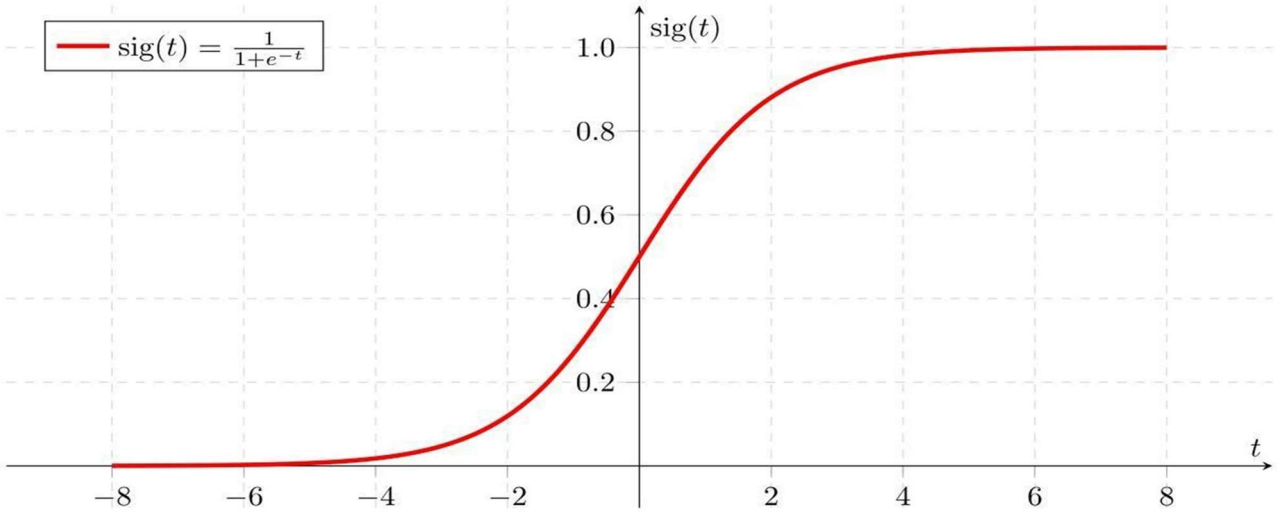


Fig-4.2 Sigmoid function

If ‘Z’ goes to infinity, Y(predicted) will become 1 and if ‘Z’ goes to negative infinity, Y(predicted) will become 0.Output = 0 or 1

4.3.3 Support vector machines

"Support Vector Machine" (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes very well.
SVM delivers high accuracy results because it uses an optimization procedure. SVM builds a classifier by searching for a separating hyperplane (optimal hyperplane) which is optimal and maximises the margin that separates the categories (in our case spam and ham). Thus, SVM has the advantage of robustness in general and effectiveness when the number of dimensions is greater than the number of samples. Unlike Naive Bayes, SVM is a non-probabilistic algorithm.

It produces a hyper plane on a high dimensional space or a set of hyper planes on an infinite dimension space. Classification of data points is done around these hyper planes. For two classes with linearly separable data, there may be many such separators that can separate the two classes of data. Intuitively, it seems like a decision boundary that is drawn in the middle of the void between the two classes seems to be the best positioning of the decision boundary.

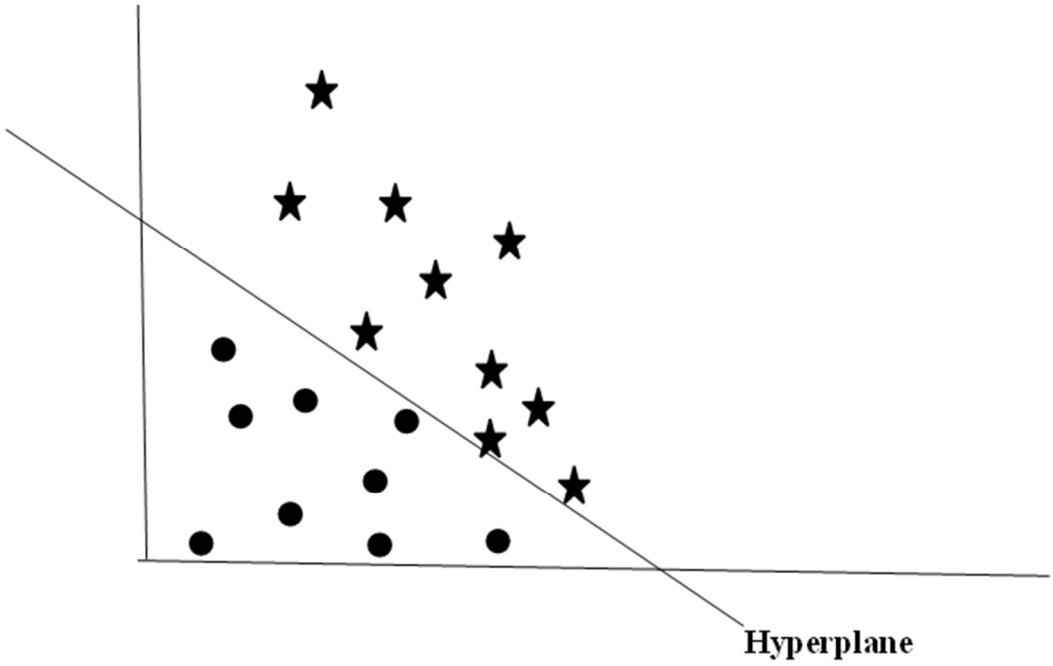


Figure-4.3 Support Vector Machine

4.3.4 Random Forest

Random Forest is a supervised learning algorithm. it creates a forest and makes it somehow random. The "forest" it builds, is an ensemble of Decision Trees, one big advantage of random forest is, that it can be used for both classification and regression problems, which form the majority of current machine learning systems. Random Forest has nearly the same hyper parameters as a decision tree.
Random Forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model. Therefore, in Random Forest, only a random subset of the features is taken into consideration by the algorithm for splitting a node.

One of the big problems in machine learning is overfitting, but most of the time this won't happen that easy to a random forest classifier. That's because if there are enough trees in the forest, the classifier won't overfit the model.

The main limitation of Random Forest is that a large number of trees can make the algorithm to slow and ineffective for real-time predictions. In general, these algorithms are fast to train, but quite slow to create predictions once they are trained. A more accurate prediction requires more trees, which results in a slower model. In most real-world applications the random forest algorithm is fast enough, but there can certainly be situations where run-time performance is important and other approaches would be preferred.

4.3.5 K Nearest Neighbour (KNN)

k Nearest Neighbours algorithm, also known as KNN, is a non-parametric method used for pattern classification. It is an instance-based learning algorithm. The function is simply approximated locally. All computation is deferred until classification. The neighbours for the data points are taken as a set of objects whose values are known. This is basically the training phase of the algorithm. In the classification phase of the algorithm, for each point in the testing set, we consider k neighbours where k is predefined by the programmer. We then find the most frequent class values in the training set. We can find the neighbours using many methods such as linear search, space partitioning, locality sensitive hashing etc.

For implementing the algorithm, I have used the Euclidean distance between the attributes of the instances. The squared Euclidean distance between the point p = (p1, p2, ..., pn) and the point q = (q1, q2, ..., qn) is the sum of the squares of the differences between the components: $Dist2(p,q) = \Sigma i\ (pi - qi)^2$. The Euclidean distance is then the square root of Dist2(p,q)[10].

While Euclidean distance is a good measure for finding all the neighbours of a given instance, it has a major drawback that it cannot work for very large data sets as the computational time is very high. I am going to demonstrate this by implementing it on the 5574 instances that I have in the dataset.

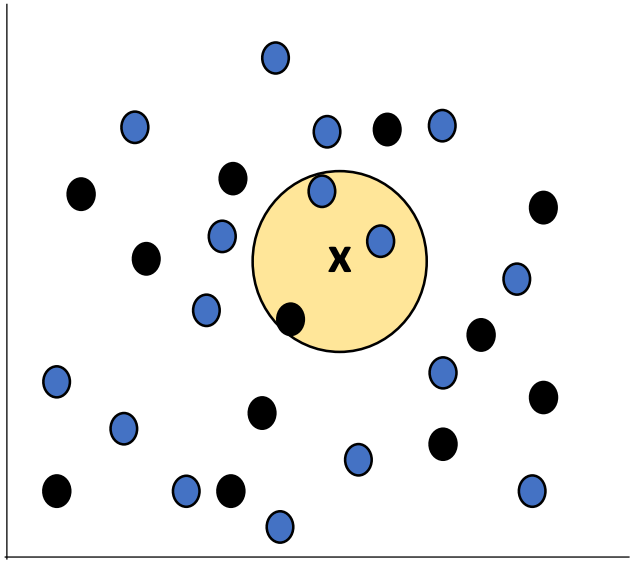


Fig-4.4 Plot depicting 3 nearest neighbours of x

The above figure depicts the general idea of the k Nearest Neighbours algorithms. There are two classes, blue and black. We have a data point x that needs to be classified. Considering that the value of k is taken as 3, we first find the distance of x from all of the given points. All the distances are then sorted in ascending order and the first 3 distances are considered. We are thus considering the 3 points that are nearest to point x. We then check the maximum frequency of points within the neighbourhood of x. As we can see, there are 1 black points and 2 blue points. So the maximum frequency is 2 and it belongs to blue. X is therefore classified as a blue point.

Following are the metrics we used to evaluate the performance of ML techniques:

True Positive(TP):

Interpretation: You predicted positive and it's true.

You predicted that a woman is pregnant, and she actually is.

True Negative(TN):

Interpretation: You predicted negative and it is true.

You predicted that a woman is not pregnant, and she actually is not.

False Positive(FP):

Interpretation: You predicted positive and it's false.

You predicted that a woman is pregnant but she actually is not.

False Negative(FN):
Interpretation: You predicted negative and it's false.

You predicted that a woman is not pregnant but she actually is.

4.3.6 Confusion Matrix

It is a performance measurement for machine learning classification problem where output can be two or more classes. It is a table with 4 different combinations of predicted and actual values.

N = 165	Predicted: No	Predicted: Yes	
Actual: No	TN = 50	FP = 10	60
Actual: Yes	FN = 5	TP = 100	105
	55	110	

Fig-4.5 Confusion Matrix

Precision

Precision refers to the closeness of two or more measurements to each other. In Machine Learning, precision is the fraction of relevant instances among the retrieved instances.

Precision = $\frac{TP}{TP+}$ ----------(4.7)

(Where TP = True Positive, TN = True Negative, FP = False Positive, FN = False Negative).

Accuracy

Accuracy refers to the closeness of a measured value to a standard or known value.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad \text{----------(4.8)}$$

Recall

Recall is how many of the true positives were recalled (found), i.e., how many of the correct hits were also found.

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

F-Score

F-scores are a statistical method for determining accuracy accounting for both precision and recall. It is essentially the harmonic mean of precision and recall.

$$\text{F1 Score} = \frac{2*(Recall*Precission)}{Recall+Precis} \quad \text{----------(4.9)}$$

Support

Support is the number of actual occurrences of the class in the specified dataset. Imbalanced support in the training data may indicate structural weaknesses in the reported scores of the classifier and could indicate the need for stratified sampling or rebalancing. Support doesn't change between models but instead diagnoses the evaluation process.

Support = TN+FP or FN+TP ----------(4.10)

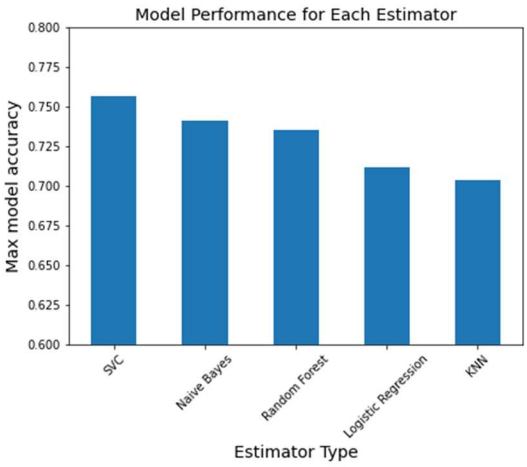
Error rate

Error rate (ERR) is calculated as the number of all incorrect predictions divided by the total number of the dataset. The best error rate is 0.0, whereas the worst is 1.0.

$$\text{ERR} = \frac{FP+FN}{TP+TN+FP+FN} \quad \text{----------(4.11)}$$

5. Results and Discussion

5.1 COMPARISON OF PERFORMANCE OF THE MODELS



Out[26]:

Estimator	score	ngram_range	tfidf__norm	computing_time	estimator_obj
SVC	0.756851	(1, 2)	l2	19.901534	SVC(kernel='linear', probability=True, random_...
Naive Bayes	0.741470	(1, 2)	l1	0.344977	MultinomialNB()
KNN	0.704060	(1, 2)	l2	0.448500	KNeighborsClassifier()
Random Forest	0.735143	(1, 1)	l2	3.246425	RandomForestClassifier(n_jobs=-1, random_state...
Logistic Regression	0.711621	(1, 1)	l2	1.197857	LogisticRegression(n_jobs=-1, random_state=637)

Figure 5.1: The above bar chart and table displays the cross validation accuracy of the best model for each estimator.

5.1.1 Support vector machine

As seen from the above graph, the SVC model had the best performance. It is important to note that models were compared by their accuracy on predicting individual tweets. A model's ability to predict aggregated sentiment scores were not considered when selecting hyperparameters. As seen in Figure 5.1, the Support Vector Classifier(SVC), with an accuracy of 75.69%.However, this model took over 8 times as long to train (Figure 3). When training a single model, this is only a couple of seconds, but there were many different models created during cross validation when tuning the hyperparameters, making this difference much more noticeable

5.1.2 Naive Bayes

For this problem, Naive Bayes algorithm worked well as expected.

High values suggests that the model is very good at correctly classifying sentiments of tweets. The Naive Bayes estimator produced the second best results, with a cross validation accuracy of 74.14%. Unlike the SVC, this model computed roughly 20% quicker than the SVC model. Naïve Bayes model estimated and computed quicker than all the model which is seen in fig5.2

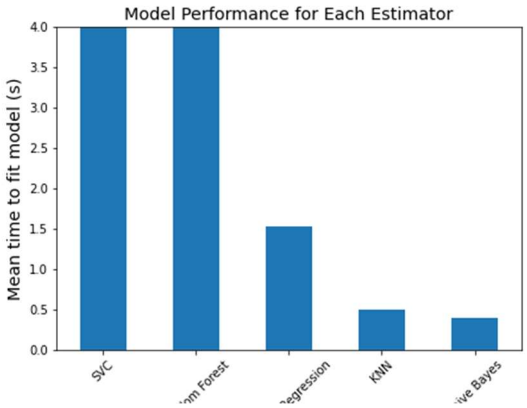


Figure 5.2: This bar chart displays the computing time for the best model of each estimator, as defined in Figure 5.1.

5.1.3 Logistic Regression

Accuracy value is quite low as compared to Naïve Bayes , Random forest and SVC model. however, it is faster in terms of computing time better than SVC model .As seen in Figure 5.1, the Logistic Regression, with an accuracy of 71.11%

5.1.4 Random Forest

This model shows a better result among the others and this gives the same result like Naïve Bayes model. The Random Forest model was the most computationally expensive model, yet fell in the middle of the pack in terms of accuracy with 73.50%.

5.1.5 K Nearest Neighbour (KNN)

KNN classifier fell short with accuracy of about 70% and is the second most quicker in terms of computing overall as seen in fig 5.2 and hence was also the worst performer among all the model .

5.2 VALIDATION

We can examine the accuracy of the model more closely by plotting the receiver operating characteristic (ROC) curve for each sentiment. Curves closer to the top-left corner of the plane signal that a model can correctly classify most of the tweets of a certain sentiment, without falsely classifying tweets of other sentiments as such. The area under the curve (AUC) is a numerical quantity used to measure ROC curves. As visible in the Figure 4, the ROC curves of each sentiment hug the top-left corner quite nicely. While not all curves follow the exact same path, their AUC values are quite similar; this implies the model is good at predicting each sentiment and is not over-trained to recognize one of them and under-trained in recognizing another. Here The ROC curve implies model is relatively good at predicting all three sentiments equally

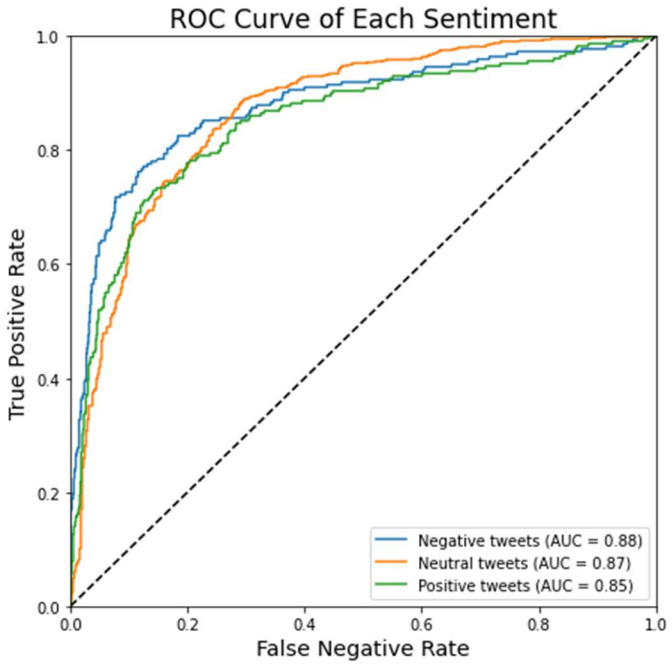


Figure 5.4: The ROC curve plots the true positive rate against the false negative rate at various thresholds for each sentiment

5.3 SENTIMENT PREDICTION

We used this model on our three benchmark tweets to examine the behaviour of the model when classifying tweets. Table 2 contains the model's sentiment predictions.

Tweet	Negative probability	Neutral probability	Positive probability
I love the world	0%	0%	100%
I hate the world	100%	0%	0%
I am indifferent about the world	16.7%	81.1%	2.3%

Table 5.1: This table shows the conditional probabilities of each tweet being assigned a specific sentiment. The sentiment with the highest probability is chosen as the classification by the model.

The first and second tweets are objective. The model is able to predict with extreme confidence that the tweets are positive and negative, respectively. As we mentioned previously, however, the third tweet is much more subjective. While it lies directly in between the first two, it may also carry a slightly more negative tone. We see this in our model too, as it predicts neutral as the most likely sentiment, but also notes that there is a decent chance the tweet is negative . These were hand-picked tweets that are used only as a proof-of-concept and a benchmark that any good model should be able to pass.

5.4 SENTIMENT SCORES

Moving forward, we can now measure this model's ability to predict aggregate sentiment scores on large collections of tweets. We took 1,000 bootstrapped samples from our test set and computed the aggregate sentiment score of each. The aggregate score is simply the average of each sentiment score in the collection of tweets, normalized to a -1 to 1 scale. A score of -1 means all tweets are negative; a score of 0 means either all tweets are neutral or there are an equal number of negative and positive tweets; a score of 1 means that all tweets are positive.

These aggregate scores can be compared to the known aggregate scores of each bootstrapped sample, as the individual tweets have known sentiments. Because these bootstrapped samples come from our test set, this can provide a reliable measure of the uncertainty in the model's aggregate score predictions. These tweets were never seen by the model during training and tuning, and thus the model has no bias towards them. As seen in Figure 5, the aggregate score residuals were well behaved, with a roughly normal distribution cantered at 0.004, which is close enough to zero for our purposes. The standard deviation was 0.0247, which means that the model's aggregate sentiment scores can be accepted with 95% confidence at ± 0.05 the predicted value. Note that the scale for aggregate scores is -1 to 1, meaning that this interval is equivalent to ± 2.5%

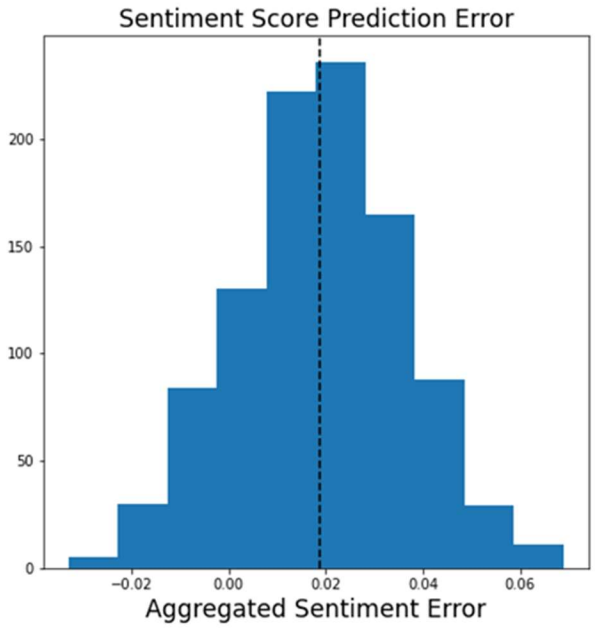


Figure 5: This histogram shows the distribution of the residuals for the model's aggregate sentiment score of 1,000 bootstrapped test samples

5.5 Application of Project

Search term	Agg. Score (excluding retweets)	Agg. Score (including retweets)
(Donald) Trump	1 %	6 %
(Joe) Biden	-3 %	9 %
(Nancy) Pelosi	- 43 %	- 23 %
Mitch McConnell	2 %	1 %
(Barack) Obama	19 %	28 %
Republicans	- 4 %	4 %
Democrats	- 3 %	0 %

Table 4: This table shows the aggregate sentiment scores for various high ranking government officials. Scores are given on a scale from -100% to $+100\%$

An interesting application of this model was to compare the aggregate sentiment scores of key politicians in the United States government. Table 4 highlights the scores of ranking members in each political party. This search first points out a peculiar phenomenon. For each term, roughly 2,000 tweets were collected. The first 1,000 tweets are all unique; i.e., all retweets were excluded. The second 1,000 tweets did contain retweets and were thus not all unique. It is noteworthy that for nearly all search terms, the aggregated sentiment score was higher for the collection of tweets that contained retweets. This would suggest that Twitter users tend to retweet positive opinions more so than negative ones.

By comparing the actual scores, we can see that the two presumptive Presidential nominees Donald Trump and Joe Biden have relatively similar numbers. House Speaker Nancy Pelosi (D-CA) is very unpopular on Twitter, with one of the lowest scores seen in any topic discussed in this paper. However, she benefited strongly from positive retweets. Senate Majority Leader Mitch McConnell is much less polarizing, with scores almost exactly neutral. Users are tweeting more positively about former President Barack Obama than any other political member. On a broad note, we also compared the terms “Republicans” and “Democrats”. The only takeaway is that Republicans received a larger boost from positive retweets than Democrats.

6. CONCLUSIONS

Sentiment analysis, or opinion analysis, is the process of retrieving textual information and discerning which emotions are exhibited by the author. This type of analysis is used in many ways, including:

- determining consumers’ perception of a product, service, brand or marketing campaign;
- analyzing a company’s brand recognition on any social networking sites;
- examining citizen’s opinions on policy changes, government officials, campaigns, etc.

In our study, we were able to achieve an accuracy of 75.68% on individual tweet classifications, and a 95% confidence interval of ± 0.05 on aggregated sentiment score predictions. The best machine learning method was found to be Support Vector Machine. This model was then used on live tweets relating to various subject matters to extract real-time user sentiment

7. REFERENCES

[1]. Bravo-Marquez, Felipe, et al. “From Unlabelled Tweets to Twitter-Specific Opinion Words.” Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval -SIGIR '15 , 2015, doi: 10.1145/2766462.2767770. Conference on Image Processing, Vol. 29, Iss. 1, pp. 63-92

[2].NLP:(KrishNaik-Youtube) - https://www.youtube.com/watch?v=fM4qTMfCoak&list=PLZoTAELRMXVMdJ5sqbCK2LiM0HhQVWNzm

[3]. Wikipedia- Machine Learning & Natural Language Processing Annual Hawaii International Conference on Transaction on Spam

[4] Wang, Min, et al. “Microblog Sentiment Analysis Based on Cross-Media Bag-of-Words Model.”Proceedings of International Conference on Internet Multimedia Computing and Service - ICIMCS '14,2014, doi:10.1145/2632856.2632912.

[5]. Severyn, Aliaksei, and Alessandro Moschitti. “Twitter Sentiment Analysis with Deep Convolutional Neural Networks.” Proceedings of the 38th International ACM SIGIR Conference on Research andDevelopment in Information Retrieval - SIGIR '15, 2015, doi:10.1145/2766462.2767830

[6]. Steinberger, Josef, et al. “Creating Sentiment Dictionaries via Triangulation.” Decision Support Systems,vol. 53, no. 4, 2012, pp. 689–694., doi:10.1016/j.dss.2012.05.029.

[8]. Yuan, Ye and You Zhou. “Twitter Sentiment Analysis with Recursive Neural Networks.” (2015).