TWITTER SENTIMENT ANALYSIS SYSTEM

Naresh Ponthangi
University of Massachusetts Lowell
Lowell Massachusetts
saidurganaresh_ponthangi@student.uml.edu

Vishista Reddy Pandiri
University of Massachusetts Lowell
Lowell Massachusetts
vishistareddy_pandiri@student.uml.edu

Sasi kiran reddy kalam
University of Massachusetts Lowell
Lowell Massachusetts
sasikiranreddy_kalam@student.uml.edu

Abstract— **Twitter is a social networking website where users post their opinions known as "tweets". This acts as a medium for individuals to express their thoughts or feelings about different subjects. Many consumers and marketers have done sentiment analysis on such tweets to gather insights into products or to conduct market analysis. In this project, we will conduct sentiment analysis on "tweets" using different machine learning and deep learning algorithms. We attempt to classify the sentiment of the tweet where it is either positive or negative. If the tweet has both positive and negative sentiments, the more dominant sentiment shall be picked as the final label.**

We use the Twitter sentiment dataset, the data provided comes with positive or negative emotions, usernames and hashtags which are required to be processed and converted into a standard form. We also need to extract useful features from the text such as unigrams, bigrams which is a form of representation of a "tweet". We use various machine learning and deep learning algorithms to conduct sentiment analysis using the extracted features and we combine different classifiers in order to improve the prediction accuracy. Finally, we present our experimental results and findings at the end.

All the work has been done on Jupyter Notebook using Python as the programming language. **Keywords—Tweets, Classification, Information Gain, Gini Index, Naïve Bayes, KNN, Random Forest, CNN, Confusion Matrix, classification report.**

I. Introduction

In today era twitter is the major source of information and individuals around the globe express their feelings or views i.e. sentiments which are known as tweets. So what we have done is, we have taken a dataset and done an analysis on tweets and a unique attribute having values 0 and 1(0 for negative sentiment and 1 for positive sentiment). So whenever someone posts a tweet, our model will find out in which category will that post fit in. A new use for sentiment analysis is to forecast the outcomes of popular political elections and surveys. One such study was carried out in Germany by Tumasjan et al. to forecast the results of federal elections, and it found that Twitter is a good indicator of offline opinion.

Dataset: The dataset contains 1,600,000 tweets extracted using the twitter api. The tweets have been annotated (0 = negative, 1 = positive) and they can be used to detect sentiment. The data is in the form of a comma-separated values files with tweets and their corresponding sentiments. The training dataset is a csv file of tweet_id, sentiment, tweet where the tweet_id is a unique integer identifying the tweet, sentiment is either 1 (positive) or 0 (negative), and tweet is the text enclosed in "".

It contains the following 6 fields:

1. target: the polarity of the tweet (*0* = negative, *1* = positive)
2. ids: The id of the tweet
3. date: date of the tweet
4. flag: The query (*lyx*). If there is no query, then this value is NO_QUERY.
5. user: the user that tweeted (*markzuck*)
6. text: text of the tweet (*abc is cool*)

Classification: Classification is a task in which we try to find a model that describes and distinguishes data classes and concepts. Classification is the problem to identify in which set of categories new observations belongs.
Below are listed various classification algorithms:
1.Decision Tree Classifier
2.Naïve Bayes Classifier
3.KNN (k- nearest neighbours) Classifier
4.Random Forest Classifier
5.CNN
There are various methods available for the evaluation of classification models such as Confusion Matrix, Precision and Recall, ROC Curve etc.

The advantages of Classification is it is cost efficient, helps in predicting risks etc. and the disadvantages of Classification is accuracy problem i.e. there has to be an accurate model to get the best result.

II. Problem Definition

Twitter Sentiment Analysis is a natural language processing (NLP) task that involves analyzing and classifying tweets or text data based on the sentiment they express. The main objective of this task is to determine whether a tweet expresses a positive, negative, or neutral sentiment.
The problem definition of Twitter Sentiment Analysis is to develop a machine learning or deep learning model that can accurately analyze and classify a large volume of tweets based on their sentiment. This model should be able to handle the challenges of processing unstructured text data, including variations in language, grammar, slang, and context. The applications of Twitter Sentiment Analysis are wide-ranging and can be used in various fields, including marketing, politics, finance, and customer service. For

example, businesses can use Twitter Sentiment Analysis to understand customer feedback and improve their products and services, while politicians can use it to gauge public opinion and track voter sentiment.

III. Methodology

a) Decision Tree

A decision tree is a tree-like graph with nodes, what happens in it is that it represents the place where we pick an attribute and ask a question; over to which branch should we proceed here

- Edges represent the answers to the question;
- Leaves represent the actual output or class label.

A general algorithm for a decision tree follow these steps:

- Pick the best attribute/feature. The best attribute is one which best splits or separates the data.
- Ask the relevant question.
- Follow the answer path.
- Go to step 1 until you arrive to the answer.

How do we decide how to split a tree at a particular node?

The dataset can be split using entropy and information gain.

1.Information Gain

Entropy is the measure of impurity or uncertainty in a bunch. Entropy controls how a Decision Tree decides to split the data. It effects how a Decision Tree draws its boundaries.

Information gain is calculated by comparing the entropy of the dataset before and after a transformation. So we use this way to train a Decision Tree, the best splitting of a dataset can be evaluated by optimising Information Gain.

$$Entropy = \sum_{i=1}^{C} -p_i * \log_2(p_i)$$

$$Gain(S, A) = Entropy(S) - \sum_{v \epsilon Values(A)} \frac{|S_v|}{|S|}.Entropy(S_v)$$

2.Gini Index

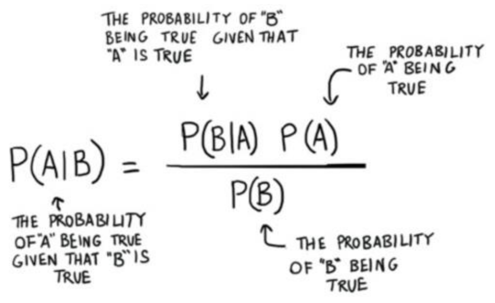
Gini index or Gini impurity measures the degree or probability of a particular variable being wrongly classified when it is randomly chosen. But what is actually meant by ‘impurity’? If all the elements belong to a single class, then it can be called pure. The degree of Gini index varies between 0 and 1, where 0 denotes that all elements belong to a certain class or if there exists only one class, and 1 denotes that the elements are randomly distributed across various classes. A Gini Index of 0.5 denotes equally distributed elements into some classes.

Formula for Gini Index:

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

b) Naïve Bayes algorithm

A Naive Bayes classifier is a probabilistic machine learning model that’s used for classification task. The crux of the classifier is based on the Bayes theorem. A Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. Yes, it is really Naive!


THE PROBABILITY OF "B" BEING TRUE GIVEN THAT "A" IS TRUE
THE PROBABILITY OF "A" BEING TRUE
P(A|B) = P(B|A) P(A) / P(B)
THE PROBABILITY OF "A" BEING TRUE GIVEN THAT "B" IS TRUE
THE PROBABILITY OF "B" BEING TRUE

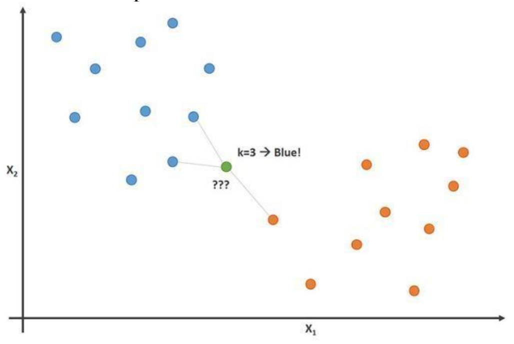
The algorithm first creates a frequency table (similar to prior probability) of all classes and then creates a likelihood table. Then, finally, it calculates the posterior probability. Naive Bayesian models are easy to build and particularly useful for small & medium sized data sets Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

c) K Nearest Neighbours Algorithm

KNN is a non-parametric, lazy learning algorithm. Its purpose is to use a database in which the data points are separated into several classes to predict the classification of a new sample point.

Working of KNN:

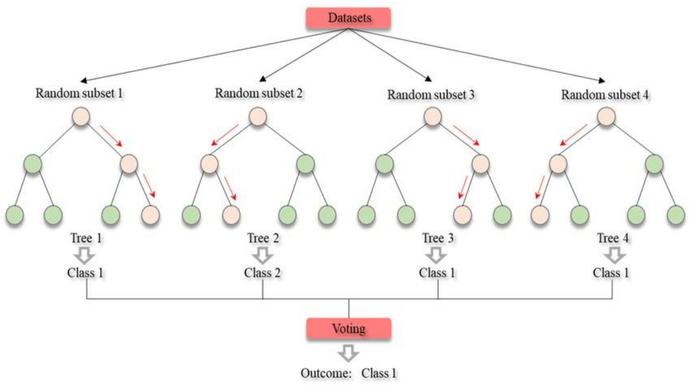
As we know that kennel is based on the nearest neighbours so in this algorithm a positive integer k a is given along with the sample we select the k closest entries in a database for a new sample to be classified the most common classification of these entries is simply the classification for the new sample.



But why do we call KNN Algorithm lazy that is because it does no training at all when you provide the training data. At the time of training all it does is it stores the complete data set and only performs the calculation after providing with the testing data set.

d) Random Forest Algorithm

"Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Let us suppose we have a data set which contains a variety of subsets. And when the random forest classifier is applied on this data set a number of decision trees work on a given random subset and after the training each decision tree predicts a given result and according to it if a new data point appears the random forest tree gives us the final decision on the basis of majority of the outcomes.



e) Convolution neural network (CNN)

CNNs can also be used for text sentiment analysis. The CNN architecture for text classification typically consists of an embedding layer, one or more convolutional layers, a max-pooling layer, and one or more fully connected layers.

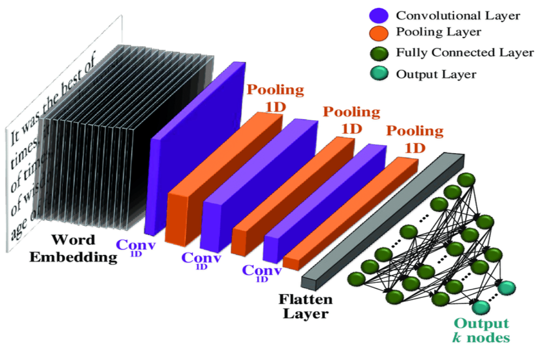
The first step is to convert the text data into numerical form. This is done using an embedding layer, which maps each word in the text to a high-dimensional vector. The embedding layer learns the representation of each word in the context of the other words in the sentence.

The output of the embedding layer is then passed through one or more convolutional layers, where each filter scans over a fixed-length window of the text. This window size is chosen based on the expected length of the text sequences.

The filters in the convolutional layers extract local features from the text, such as n-grams and other patterns. The output of each filter is a feature map, which is passed through an activation function such as ReLU to introduce non-linearity.

The max-pooling layer reduces the dimensionality of the feature maps by selecting the maximum value from each feature map. This helps to capture the most important features in each feature map and reduce the number of parameters.

The output of the max-pooling layer is flattened and fed to one or more fully connected layers, which perform the final classification. The final output layer uses a softmax activation function to produce a probability distribution over the different sentiment classes.



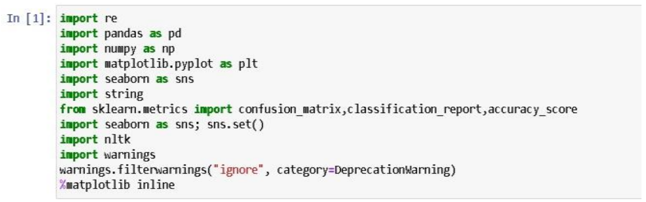
During training, the CNN algorithm learns the optimal set of filters and parameters through backpropagation and gradient descent. The model's performance is evaluated using a loss function such as cross-entropy, and the parameters are adjusted to minimize the loss.

CNNs for text sentiment analysis have shown promising results in various applications, such as movie reviews, product reviews, and social media posts.

IV. Data Importing and Preprocessing: -

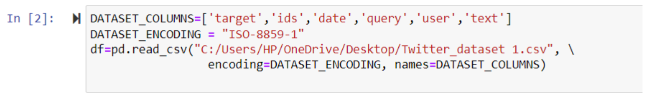
1. Importing Dataset : -

Firstly, we import the necessary modules for our project such as re (Regular Expression), pandas for DataFrame, numpy for handling array, matplotlib and seaborn for plotting and visualization of dataset.



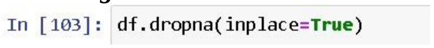
```
In [1]: import re
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        import string
        from sklearn.metrics import confusion_matrix,classification_report,accuracy_score
        import seaborn as sns; sns.set()
        import nltk
        import warnings
        warnings.filterwarnings("ignore", category=DeprecationWarning)
        %matplotlib inline
```

Load the dataset into pandas Dataframe using read_csv function.



```
In [2]:  DATASET_COLUMNS=['target','ids','date','query','user','text']
         DATASET_ENCODING = "ISO-8859-1"
         df=pd.read_csv("C:/Users/HP/OneDrive/Desktop/Twitter_dataset 1.csv", \
                        encoding=DATASET_ENCODING, names=DATASET_COLUMNS)
```

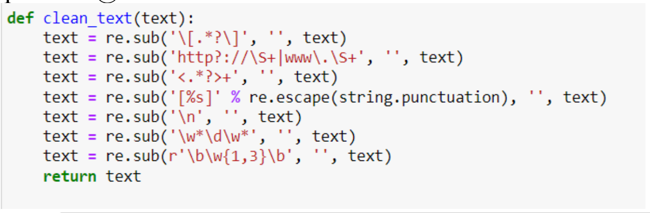
2. Removing Null Values:-



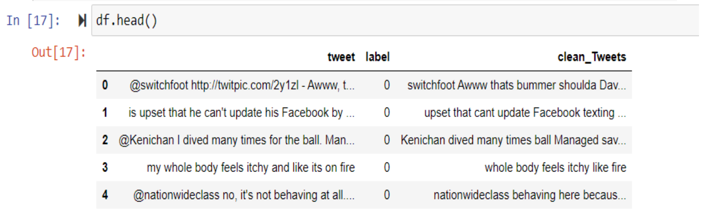
```
In [103]: df.dropna(inplace=True)
```

3. Removing Twitter Handles: -

Given below is a user-defined function to remove unwanted text patterns from the tweets. It takes two arguments, one is the original string of text and the other is the pattern of text that we want to remove from the string. The function returns the same input string but without the given pattern. We will use this function to remove the pattern '@user' from all the tweets in our data



```
def clean_text(text):
    text = re.sub('\[.*?\]', '', text)
    text = re.sub('http?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\n', '', text)
    text = re.sub('\w*\d\w*', '', text)
    text = re.sub(r'\b\w{1,3}\b', '', text)
    return text
```



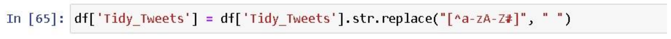
```
In [17]:  df.head()

Out[17]:
```

	tweet	label	clean_Tweets
0	@switchfoot http://twitpic.com/2y1zl - Awww, t...	0	switchfoot Awww thats bummer shoulda Dav...
1	is upset that he can't update his Facebook by ...	0	upset that cant update Facebook texting ...
2	@Kenichan I dived many times for the ball. Man...	0	Kenichan dived many times ball Managed sav...
3	my whole body feels itchy and like its on fire	0	whole body feels itchy like fire
4	@nationwideclass no, it's not behaving at all....	0	nationwideclass behaving here becaus...

4. Removing Punctuations, Numbers, and Special Characters: -

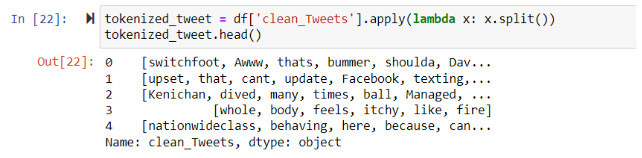
Punctuations, numbers and special characters do not help much. It is better to remove them from the text just as we removed the twitter handles. Here we will replace everything except characters and hashtags with spaces**.**



```
In [65]: df['Tidy_Tweets'] = df['Tidy_Tweets'].str.replace("[^a-zA-Z#]", " ")
```

5. Tokenization *:-*

Now we will tokenize all the cleaned tweets in our dataset. Tokens are individual terms or words, and tokenization is the process of splitting a string of text into tokens.

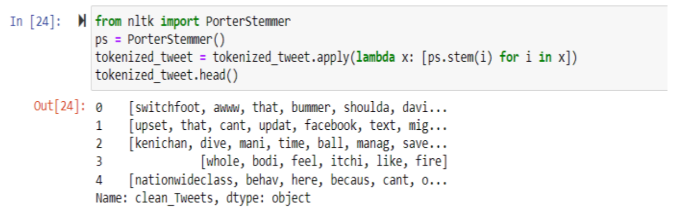


```
In [22]:  tokenized_tweet = df['clean_Tweets'].apply(lambda x: x.split())
          tokenized_tweet.head()

Out[22]: 0    [switchfoot, Awww, thats, bummer, shoulda, Dav...
         1    [upset, that, cant, update, Facebook, texting,...
         2    [Kenichan, dived, many, times, ball, Managed, ...
         3             [whole, body, feels, itchy, like, fire]
         4    [nationwideclass, behaving, here, because, can...
         Name: clean_Tweets, dtype: object
```

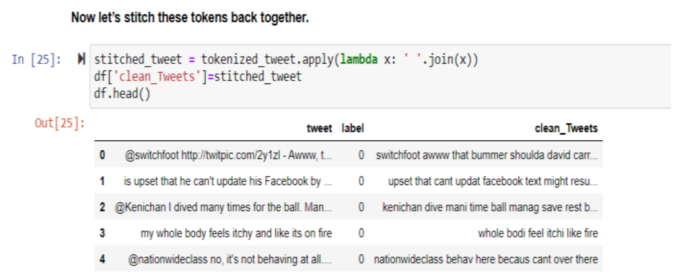
6.Stemming: -

Stemming is a rule-based process of stripping the suffixes (“ing”, “ly”, “es”, “s” etc) from a word. For example– “play”, “player”, “played”, “plays” and “playing” are the different variations of the word“play”



```
In [24]:  from nltk import PorterStemmer
          ps = PorterStemmer()
          tokenized_tweet = tokenized_tweet.apply(lambda x: [ps.stem(i) for i in x])
          tokenized_tweet.head()

Out[24]: 0    [switchfoot, awww, that, bummer, shoulda, davi...
         1    [upset, that, cant, updat, facebook, text, mig...
         2    [kenichan, dive, mani, time, ball, manag, save...
         3             [whole, bodi, feel, itchi, like, fire]
         4    [nationwideclass, behav, here, becaus, cant, o...
         Name: clean_Tweets, dtype: object
```



Now let's stitch these tokens back together.

```
In [25]:  stitched_tweet = tokenized_tweet.apply(lambda x: ' '.join(x))
          df['clean_Tweets']=stitched_tweet
          df.head()

Out[25]:
```

	tweet	label	clean_Tweets
0	@switchfoot http://twitpic.com/2y1zl - Awww, t...	0	switchfoot awww that bummer shoulda david carr...
1	is upset that he can't update his Facebook by ...	0	upset that cant updat facebook text might resu...
2	@Kenichan I dived many times for the ball. Man...	0	kenichan dive mani time ball manag save rest b...
3	my whole body feels itchy and like its on fire	0	whole bodi feel itchi like fire
4	@nationwideclass no, it's not behaving at all....	0	nationwideclass behav here becaus cant over there

V. Extracting Features from cleaned Tweets: -

1. Bag-of-Words Features: -

Bag of Words is a method to extract features from text documents. These features can be used for training machine learning algorithms. It creates a vocabulary of all the unique words occurring in all the documents in the training set.

Consider a corpus (a collection of texts) called C of D documents {d1,d2…..dD} and N unique tokens extracted out of the corpus C. The N tokens (words) will form a list, and the size of the bag-of-words matrix M will be given by D X N. Each row in the matrix M contains the frequency of tokens in document D(i).

For example, if you have 2 documents

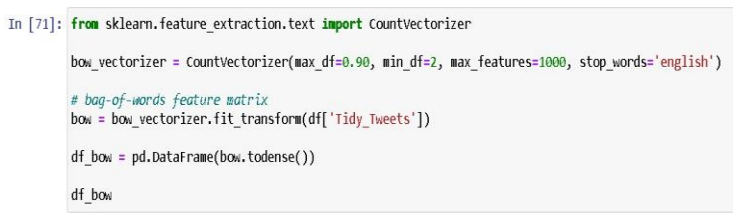
- D1: He is a lazy boy. She is also lazy.
- D2: Smith is a lazy person.

First, it creates a vocabulary using unique words from all the documents.

[‘He’ , ’She’ , ’lazy’ , 'boy’ , 'Smith’ , ’person’]

- Here, D=2, N=6
- The matrix M of size 2 X 6 will be represented as

	He	She	lazy	boy	Smith	person
D1	1	1	2	1	0	0
D2	0	0	1	0	1	1



```
In [71]:  from sklearn.feature_extraction.text import CountVectorizer

          bow_vectorizer = CountVectorizer(max_df=0.90, min_df=2, max_features=1000, stop_words='english')

          # bag-of-words feature matrix
          bow = bow_vectorizer.fit_transform(df['Tidy_Tweets'])

          df_bow = pd.DataFrame(bow.todense())

          df_bow
```

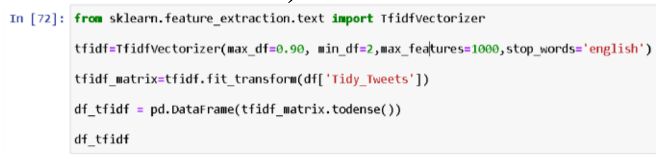
2. TF-IDF Features:-

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. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus.

Typically, the tf-idf weight is composed by two terms: the first computes the normalized Term Frequency (TF), aka. the number of times a word appears in a document, divided by the total number of words in that document; the second term is the Inverse Document Frequency (IDF), computed as the logarithm of the number of the documents in the corpus divided by the number of documents where the specific term appears.

TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document).

IDF(t) = log_e(Total number of documents / Number of documents with term t in it).



```
In [72]:  from sklearn.feature_extraction.text import TfidfVectorizer

          tfidf=TfidfVectorizer(max_df=0.90, min_df=2,max_features=1000,stop_words='english')

          tfidf_matrix=tfidf.fit_transform(df['Tidy_Tweets'])

          df_tfidf = pd.DataFrame(tfidf_matrix.todense())

          df_tfidf
```

Now let's move to the part where we apply all the classification techniques one-by-one using the two the features from the two different training sets namely:

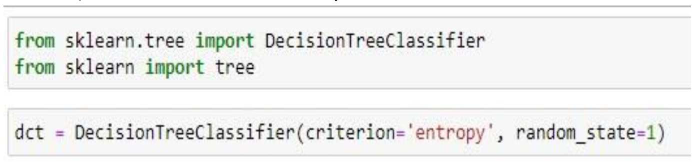
1.Bag of words model

2. TF-IDF model

VI. Experiments/Results

1.Decision Tree Classifier

a)Decision Tree From Information Gain



```
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
```

```
dct = DecisionTreeClassifier(criterion='entropy', random_state=1)
```

The “criterion” parameter specifies the function used to measure the quality of a split. In this case, 'entropy' is chosen as the criterion. Entropy is a measure of the impurity or disorder of a set of samples. The entropy criterion is used to maximize the information gain of a split, which is the reduction in entropy achieved by partitioning the data based on a given attribute.

The “random_state” parameter is used to ensure the reproducibility of the results. It is set to a fixed value of 1 in this case. By fixing the random seed, the same sequence of random numbers will be generated each time the code is run, which makes the results deterministic and reproducible.

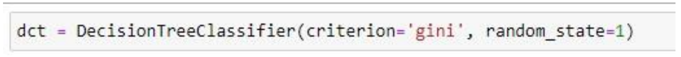
i. For Bag-of-Words

A decision tree classifier(ct) on the training data (x_train_bow and y_train_bow) which is then used to predict the labels for the validation data (x_valid_bow). accuracy_score and confusion_matrix functions from the scikit-learn library are used for evaluating classification models. The accuracy_score() function is used to compute the accuracy score of the predicted labels y_pred against the true labels y_valid_bow. It takes two arguments:
y_true: True labels of the validation data.
y_pred: Predicted labels of the validation data.
It returns the accuracy score which is defined as the fraction of correct predictions out of the total number of predictions made. The accuracy score and confusion matrix are then printed to evaluate the performance of the model on the validation data. The accuracy score is a measure of how well the model is able to correctly classify the validation data. The confusion matrix provides more detailed information on the performance of the model by showing the number of true positives, false positives, true negatives, and false negatives. The accuracy obtained using bag-of-words for decision tree is 66.57%.

ii. For TFIDF

A machine learning model using the dct (stands for decision tree classifier) algorithm is being trained using the training data x_train_tfidf and their corresponding labels y_train_tfidf. The fit() method of the dct object is used to train the model. Once the model is trained, it is used to make predictions on the validation dataset x_valid_tfidf, using the predict() method. The predicted labels are stored in the y_pred variable. The accuracy of the model's predictions is then evaluated using the accuracy_score() function from the sklearn.metrics module. The accuracy obtained using TFIDF is 66.61%.

b) Decision Tree using Gini Index



```
dct = DecisionTreeClassifier(criterion='gini', random_state=1)
```

In this case, 'gini' is chosen as the criterion. Gini impurity is another measure of the impurity or disorder of a set of samples. The gini criterion is used to maximize the Gini impurity reduction of a split, which is the reduction in the probability of misclassifying a randomly chosen sample from a set of samples.

i. For Bag-of-Words

A decision tree clasifier on training data is used to predict the labels for validation/test data. And the accuracy measure is 66.27%.

ii. For TFIDF

The same is repeated to check the accuracy between the output labels of a predict function and actual labels. And the accuracy measure is 66.69%.

2.Naïve Bayes Classifier

Gaussian Naive Bayes is a popular algorithm used for classification tasks, especially when the features are continuous (i.e., real-valued). We import the Gaussian Naive Bayes classifier class from the naive_bayes module of the scikit-learn library. "gnb = GaussianNB()" creates an instance of the GaussianNB class and assigns it to the variable "gnb". This object can be used to train a Gaussian Naive Bayes classifier on a training dataset, and then use it to make predictions on new data. While predicting, the accuracy obtained using BOW is 66.34% and the accuracy obtained using TFIDF is 51.44%.

3.KNN (k- nearest neighbours) Classifier

The k-NN algorithm is a type of supervised machine learning algorithm used for classification and regression problems. It works by identifying the k-nearest data points in the training set for a given test data point and classifying the test point based on the majority class among its k-nearest neighbors. Trains the KNN classifier on the Bag-of-Words representation of the training data x_train_bow (which is a sparse matrix) converted to a dense array using .toarray(), with the corresponding labels y_train_bow. The accuracy score using Bag-of-words is 0.659600, which means that the model correctly classified approximately 66% of the samples. And the accuracy using TFIDF is 63.02%.

4.Random Forest Classifier

Random Forest classifier performs classification on a Bag of Words (BoW) feature representation of the training data and obtained an accuracy of 70.47% and the accuracy obtained using TFIDF is 70.92%. This algorithm works much better than Decision tree, Naïve Bayes Classifier and KNN.

5.CNN

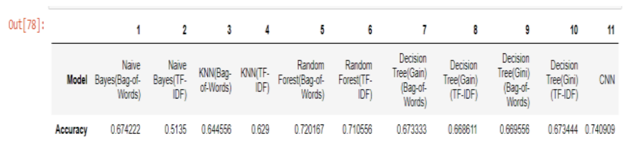
The “model.evaluate” method is used to evaluate the performance of a trained machine learning model on a validation dataset (valid_ds), which is passed to the method as a batched dataset using valid_ds.batch(128) with a batch size of 128. model.evaluate(valid_ds.batch(128) ,verbose=2), where the verbose parameter is set to 2, which means that the evaluation progress will be displayed on the console as the evaluation is running.
The method returns the evaluation results, which are the test loss and test accuracy. These results are assigned to the variables test_loss and test_acc, respectively.
Finally, the test accuracy provides a measure of how well the model is able to generalize to new data and can be used to assess the overall performance of the model on the validation dataset. The accuracy obtained using CNN algorithm is 75.88%.

VII. CONCLUSION

After conducting a sentiment analysis of Twitter data using various machine learning algorithms, including Decision tree, Naive Bayes, KNN, CNN, and Random Forest, it was found that the CNN algorithm performed the best with the highest accuracy compared to the other models.
CNNs are a type of deep learning algorithm that are particularly well-suited for image recognition, but they have also shown to be highly effective in natural language processing tasks such as sentiment analysis. The reason behind this is that CNNs can capture the complex relationships and patterns within text data through its layers of convolutions.
Furthermore, it is worth noting that the other algorithms, such as Naive Bayes, Decision tree, KNN, and Random Forest, also performed well in sentiment analysis, but were outperformed by CNN. Each of these algorithms has its own strengths and weaknesses, and choosing the best algorithm for a particular task will depend on various factors, including the nature of the data and the problem at hand.

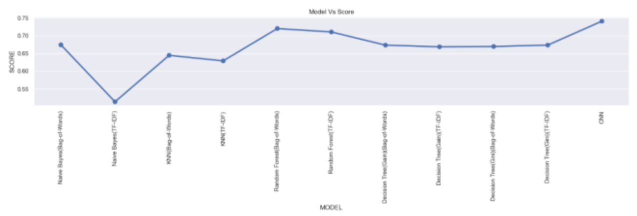
In conclusion, the sentiment analysis of Twitter data revealed that CNN is a powerful algorithm with highest accuracy of 75.88% in analyzing sentiment in text data. However, further research and testing are required to determine the optimal algorithm for specific tasks and datasets.

VIII. *Figures and Tables*



Out[78]:

	1	2	3	4	5	6	7	8	9	10	11
Model	Naive Bayes(Bag-of-Words)	Naive Bayes(TF-IDF)	KNN(Bag-of-Words)	KNN(TF-IDF)	Random Forest(Bag-of-Words)	Random Forest(TF-IDF)	Decision Tree(Gain) (Bag-of-Words)	Decision Tree(Gain) (TF-IDF)	Decision Tree(Gini) (Bag-of-Words)	Decision Tree(Gini) (TF-IDF)	CNN
Accuracy	0.674222	0.5135	0.644556	0.629	0.720167	0.710556	0.673333	0.668611	0.669556	0.673444	0.740909



A table is plot to display the accuracies of all the algorithms used. And it is observed that CNN has the most advantage in getting the accurate results.

IX. *Future scope*

The future scope of Twitter sentiment analysis is promising, as social media platforms like Twitter continue to grow in popularity and usage. Here are some potential directions for the future of Twitter sentiment analysis:

Improved accuracy: One of the biggest areas of focus for the future of Twitter sentiment analysis is improving its accuracy. This can be achieved by developing more sophisticated algorithms that can better distinguish between positive, negative, and neutral tweets, as well as by incorporating more contextual information.

Real-time analysis: Another area of future development is real-time sentiment analysis. This involves analyzing tweets as they are posted, which can provide valuable insights into how people are feeling about a particular topic or event in real-time. Real-time analysis can also help identify emerging trends and issues before they become mainstream.

Multilingual analysis: As Twitter continues to expand globally, multilingual sentiment analysis will become increasingly important. This involves analyzing tweets in multiple languages to gain insights into how people are feeling about a particular topic or issue across different regions and cultures.

Integration with other technologies: Twitter sentiment analysis can also be integrated with other technologies such as natural language processing (NLP), machine learning (ML), and artificial intelligence (AI) to improve accuracy and efficiency.

Industry-specific applications: Finally, there is potential for Twitter sentiment analysis to be applied to specific industries such as marketing, politics, and finance. For example, marketers can use sentiment analysis to gauge consumer opinions about a particular product or service, while political campaigns can use it to monitor public opinion on key issues.

X. *References*

[1] Go, A., Bhayani, R., & Huang, L. (2009). Twitter sentiment classification using distant supervision. Technical report, Stanford University.

[2] Pak, A., & Paroubek, P. (2010). Twitter as a corpus for sentiment analysis and opinion mining. In LREc (Vol. 10, pp. 1320-1326).

[3] Agarwal, A., Xie, B., Vovsha, I., Rambow, O., & Passonneau, R. (2011). Sentiment analysis of Twitter data. In Proceedings of the Workshop on Languages in Social Media (LSM 2011) (pp. 30-38).

[4] Bifet, A., Frank, E., Holmes, G., & Pfahringer, B. (2010). Sentiment knowledge discovery in Twitter streaming data. In Proceedings of the 13th International Conference on Discovery Science (pp. 1-15).

[5] Mohammad, S. M., & Turney, P. D. (2013). Crowdsourcing a word–emotion association lexicon. Computational Intelligence, 29(3), 436-465.

[6] Saif, H., He, Y., Alani, H., & Zhou, L. (2016). A semantic web-based approach for sentiment analysis in Twitter. Journal of Web Semantics, 37, 1-18.

[7] Zhang, W., & Skiena, S. (2010). Twitter sentiment analysis: Capturing sentiment from dynamic social media. In Proceedings of the 4th International AAAI Conference on Weblogs and Social Media (pp. 583-586).

[8] Zhou, G., Zhang, J., Su, J., & Tan, C. L. (2016). A C-LSTM neural network for text classification. In Proceedings of the 26th International Conference on Computational Linguistics (pp. 1-12).

[9] Y. Liu, F. Wei, S. Li, H. Ji, "Empowering Tweet Sentiment Analysis with Contextualized Representation Learning", ACL 2019.

[10] Agarwal, A., Xie, B., Vovsha, I., Rambow, O., & Passonneau, R. (2011). Sentiment analysis of twitter data. In Proceedings of the Workshop on Languages in Social Media (pp. 30-38).

[11] Go, A., Bhayani, R., & Huang, L. (2009). Twitter sentiment classification using distant supervision. CS224N Project Report, Stanford, 1(12), 2009.

[12] Mohammad, S. M., & Turney, P. D. (2013). Crowdsourcing a word–emotion association lexicon. Computational Intelligence, 29(3), 436-465.

[13] Liu, B. (2012). Sentiment analysis and opinion mining. Synthesis Lectures on Human Language Technologies, 5(1), 1-167.

[14] Kouloumpis, E., Wilson, T., & Moore, J. D. (2011). Twitter sentiment analysis: The good the bad and the OMG!. Icwsm, 11(538-541), 164.

[15] Wang, H., Can, D., Kazemzadeh, A., Bar, F., Narayanan, S., & Mihalcea, R. (2012). A system for real-time twitter sentiment analysis of 2012 us presidential election cycle. Proceedings of the ACL 2012 System Demonstrations, 115-120.

[16] Pak, A., & Paroubek, P. (2011). Cross-language opinion analysis: An experiment on multi-lingual and cross-domain sentiment analysis. Proceedings of the 2011 Workshop on Cross-Cultural Sentiment Analysis: CrossCultural Sentiment Analysis in Social Media, 61-69.

Contribution:

Feature Engineering: Sasi Kiran Reddy

Data Pre-Processing: Vishista Sai Durga Naresh

Feature Extraction: Sasi Kiran Reddy

KNN: Sasi Kiran Reddy

Naïve Bayes, Decision Tree: Vishista Reddy

Random Forest , CNN: Sai Durga Naresh