Text Classification-Sentiment Analysis of Tweets

Member	Raena Rahimi Bafrani	raenar@vt.edu

Introduction

The increasing popularity of social media provided the opportunity to people to share their thoughts on the web. Twitter is one of the most popular microblogging services on which users create their opinions about different topics in the form of messages that are called 'tweets'. Tweets are often useful in understanding the opinions of the people. However, there are so many tweets that it can be hard for businesses to prioritize which to respond first. In order to compute the customer perception and provide marketing strategies, we need to develop an automated machine learning model. Sentiment analysis has become a solution to tackle this problem; in Twitter, it is the interpretation and classification of emotions of people's opinions on a particular subject using text analysis techniques. In general, sentiment includes classifying statements as positive, negative or neutral.

Our goal is to carefully inspect the people's thoughts on Twitter using sentiment analysis tools such as natural language processing or NLP which plays a bridging role with the machine learning to understand humans in their Natural Language.

Project Problem Statement

The problem we are trying to solve through this project is to analyze user emotions and opinions expressed in tweets. Twitter allows companies, organizations and businesses to engage with the customers personally, to understand their audience and what they say about their brand and as a result they can discover significant trends in industry. In the past, analyzing tweets was a manual and very time-consuming process as it required groups of people to go through messages and label them according to their emotions. Here, we will try to implement a Twitter sentiment analysis model that classifies positive and negative tweets in text format, to overcome the challenges of identifying the emotions of the tweets. Our goal is to understand the role of a machine learning model that identifies the tweets with negative sentiment from positive in Twitter social media platform. This method enables companies to monitor customer sentiment in many different cases.

Data Set

The data set we are using is the Sentiment-140[1]. This data contains a labels data of 1.6 Million Tweets that have been extracted using the Twitter Search API.

The tweets have been annotated in target class (0 = negative, 4= positive) and they can be used to detect sentiment. It contains the following 6 fields:

target: the polarity of the tweet (0 = negative, 2 = neutral, 4 = positive)
ids: The id of the tweet (2087)
date: the date of the tweet (Sat May 16 23:58:44 UTC 2009)
flag: The query (lyx). If there is no query, then this value is NO_QUERY.
user: the user that tweeted (robotickilldozr)

1 Available here: http://help.sentiment140.com/for-students

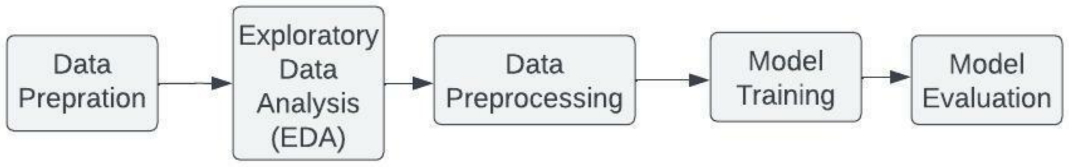
text: the text of the tweet (Lyx is cool)

Project Structure

The various steps involved in this project:

- Import Necessary Dependencies
- Read and Load the Dataset
- Exploratory Data Analysis
- Data Visualization of Target Variables
- Data Preprocessing
- Splitting our data into Train and Test Subset
- Transforming Dataset using TF-IDF Vectorizer
- Function for Model Evaluation
- Model Building
- Conclusion

Machine Learning Pipeline



EAD and Data Visualization

Before visualizing data, we need to look for missing values, duplicate rows and/or columns so that we can discard them or perform some imputation on them. Fortunately, there are no missing values and there is no class imbalance in the dataset. In order to train our Machine Learning models in time, we will pick one eighth of data (200K) samples from the dataset.

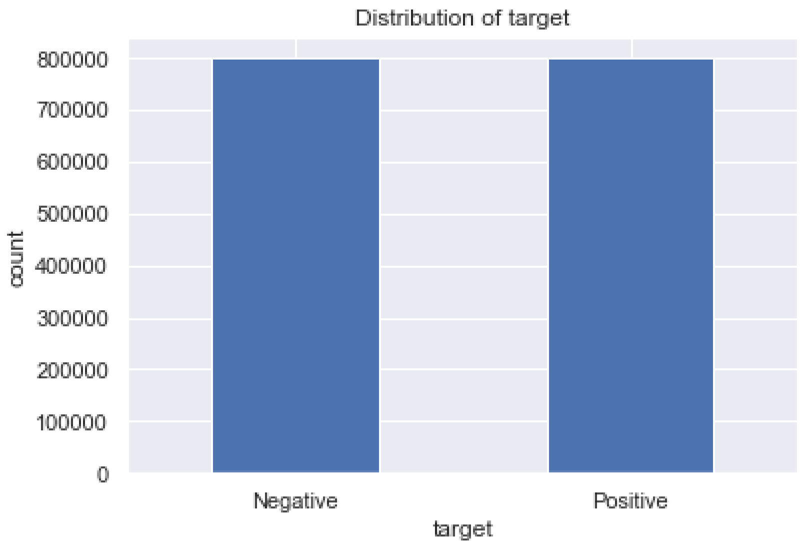


Figure1

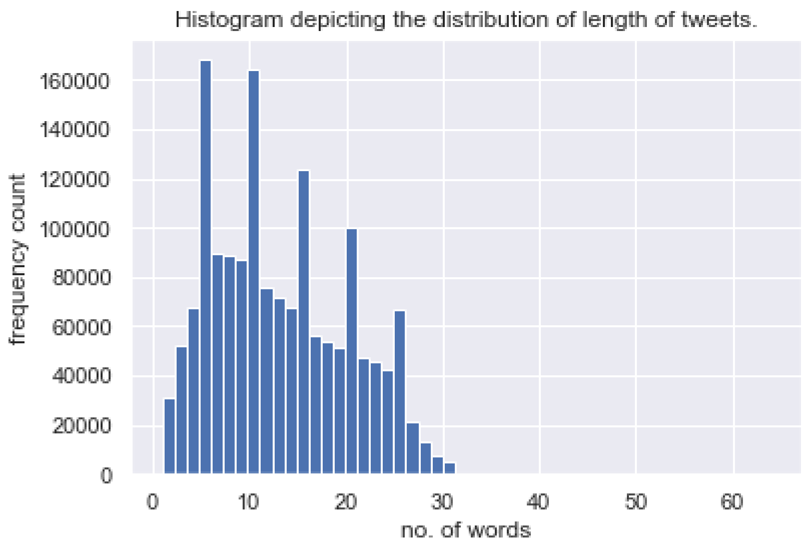


Figure2

According to the distribution of Figure(1), we can see that data is equally distributed in the 'target' class, meaning it is a very good dataset without any skewness. From Figure2 we can see that most of the tweets are under 30 words.

Preprocessing steps

Preprocessing is not a simple process but it is essential, as text data often contain redundant and/or repetitive words, other user mentions, hyperlink texts, emoticons and punctuations. This is especially true in Twitter sentiment analysis, so processing our text data is the first step towards our project.

- **Lowering case**: Each text is converted to lowercase.

In (Sample from dataset) >> 'Omg i had fun @ Club Ice 2nite wit my twin &amp; Kei. &amp; i saw a lot of ppl i kno! Time 4 sleep!'

Out >> 'omg i had fun @ club ice 2nite wit my twin &amp; kei. &amp; i saw a lot of ppl i kno! time 4 sleep!'

- **Removing Stopwords**: Stopwords are the English words which does not add much meaning to a sentence. They can safely be ignored without sacrificing the meaning of the sentence.

In (Sample from dataset) >> '@FashionGrail I was gonna do the same pic but here is the dress I picked as a cheaper version http://tinyurl.com/nhoj8h... yours is cuter'

Out >> '@FashionGrail gonna pic but dress picked cheaper version http://tinyurl.com/nhoj8h... cuter'

- **Replacing URLs**: Links are replaced by "URL".

In (Sample from dataset) >> '@FashionGrail I was gonna do the same pic but here is the dress I picked as a cheaper version http://tinyurl.com/nhoj8h... yours is cuter '

Out >> '@FashionGrail I was gonna do the same pic but here is the dress I picked as a cheaper version URL yours is cuter '

- **Replacing Usernames**: Replace @Usernames with blank " ".

In (Sample from dataset) >> '@FashionGrail I was gonna do the same pic but here is the dress I picked as a cheaper version http://tinyurl.com/nhoj8h... yours is cuter '

Out >> ' I was gonna do the same pic but here is the dress I picked as a cheaper version http://tinyurl.com/nhoj8h... yours is cuter '

- **Removing punctuations**: Replacing characters except Digits and Alphabets with a space.

In (Sample from dataset) >> 'On a lighter note.....Me+mara+carlo+'nell+carly=demetri martin july 25th in downtown. Its on baby!'
Out >> 'On a lighter noteMemaracarlonellcarlydemetri martin july 25th in downtown Its on baby '

- **Removing repeating characters**: 3 or more consecutive letters are replaced by 2 letters.

In (Sample from dataset) >> 'doneeee wheeee hahaaaaaaaa so tired and sleepy peter u suck not coming to my bday!'
Out >> 'donee whee hahaa so tired and sleepy peter u suck not coming to my bday!'

- **Removing numeric values**: Replace numbers with NUM

In (Sample from dataset) >> 'DANG want some pho n too bad there isn't a 24 hour place like in Cali Guess I have to go to the FOB club n eat it there...'
Out >> 'DANG want some pho n too bad there isn't a NUM hour place like in Cali Guess I have to go to the FOB club n eat it there...'

- **Stemming**: removing and replacing suffixes from a token to obtain the root or base form of the word (the stem for the words, 'satisfied', 'satisfaction', and 'satisfying' is 'satisfy' and all of these imply the same feeling()
- **Lemmatizing**: Lemmatization is the process of reducing inflectional endings only and to return the base and dictionary form of a word (the lem for the words runs, running, ran is run)

Text Feature Engineering

- **Word Embedding**: tried the **TF-IDF Vectorizer** method to transform the data. This is a very common algorithm to transform text into a meaningful representation of numbers which is used to fit a machine algorithm for prediction. The maximum number of features parameter enables using only the 'n' most frequent words as features instead of all the words. In this project we consider 50,000 to be passed for this parameter.

(Note: For the ease of this phase of the project we only use one fourth of the whole data to train our machine learning algorithm)

Methods and Models

After completing the model architecture, we need to train the model on the dataset. Generally, it is a good idea to try different machine learning algorithms to find the highest accuracy score. To reach this goal, we are creating 5 different types of model for our sentiment analysis problem:

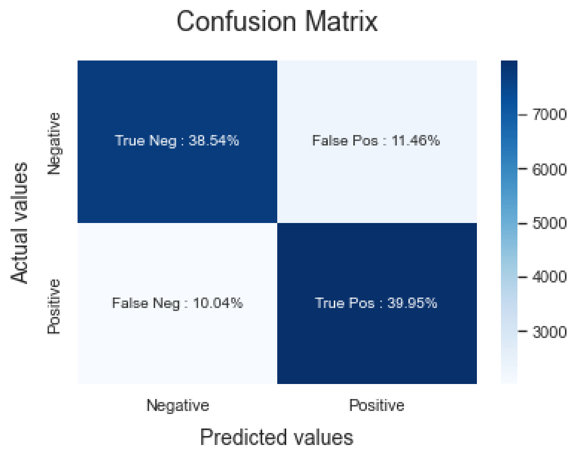
- Bernoulli Naive Bayes (BernoulliNB)
- Linear Support Vector Machine (LinearSVM)
- Logistic Regression (LR)
- Random Forest Classification
- Naive Bayes classifier for Multinomial models (MultinomialNB)

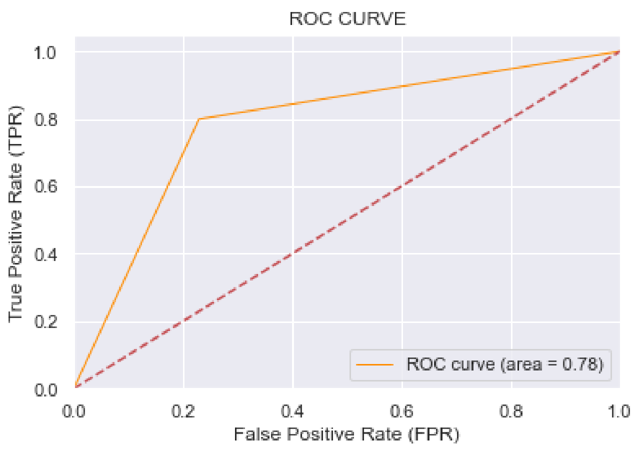
The idea behind choosing these models is that we want to try all the classifiers on the dataset ranging from simple ones to complex models and then try to find out the one which gives the best performance among them.
Our dataset has an equal number of Positive and Negative Predictions (it is not skewed). Therefore, we are choosing Accuracy as our evaluation metric to get an understanding of how our model is performing on both classification types.

Bernoulli Naïve Bayes
Works well for many text classification problems

```python
BNBmodel = BernoulliNB(alpha = 2)
BNBmodel.fit(X_train, y_train)

y_pred1 = BNBmodel.predict(X_test)

print(classification_report(y_test, y_pred1))
```

```
              precision    recall  f1-score   support

           0       0.79      0.77      0.78     10000
           1       0.78      0.80      0.79     10000

    accuracy                           0.78     20000
   macro avg       0.79      0.78      0.78     20000
weighted avg       0.79      0.78      0.78     20000
```

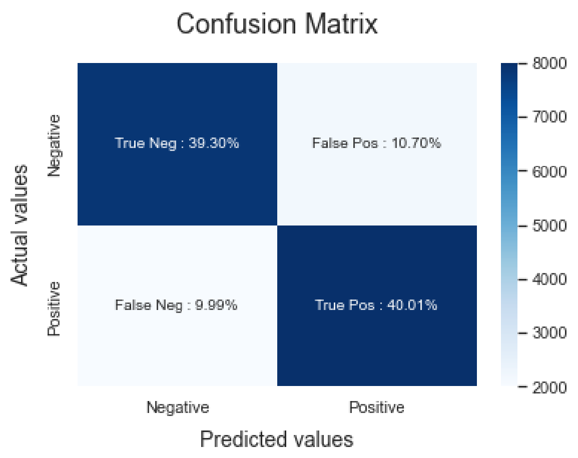


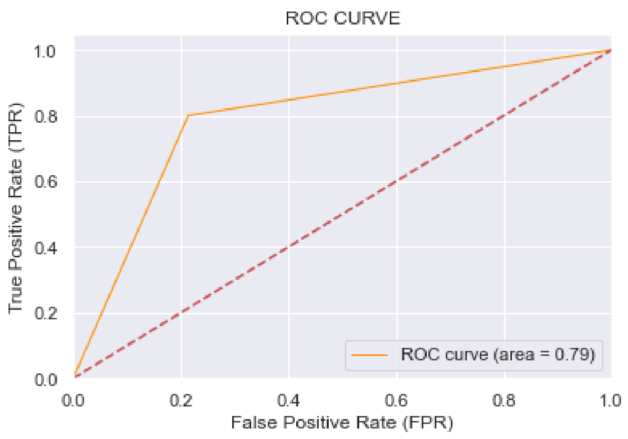


Linear-Support Vector Machine (SVM)
Works well for text classification

```python
SVCmodel = LinearSVC()
SVCmodel.fit(X_train, y_train)

y_pred3 = SVCmodel.predict(X_test)

print(classification_report(y_test, y_pred3))
```

```
              precision    recall  f1-score   support

           0       0.80      0.79      0.79     10000
           1       0.79      0.80      0.79     10000

    accuracy                           0.79     20000
   macro avg       0.79      0.79      0.79     20000
weighted avg       0.79      0.79      0.79     20000
```

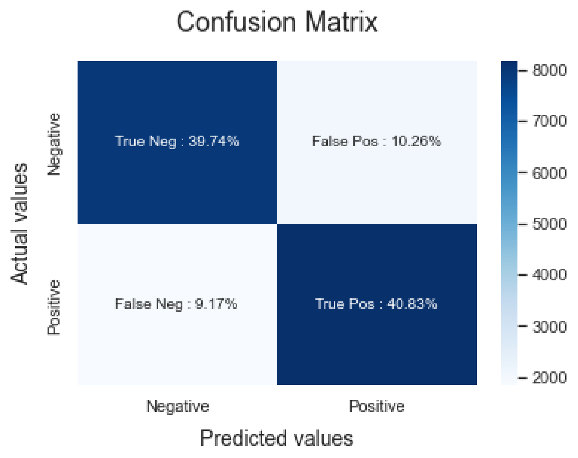


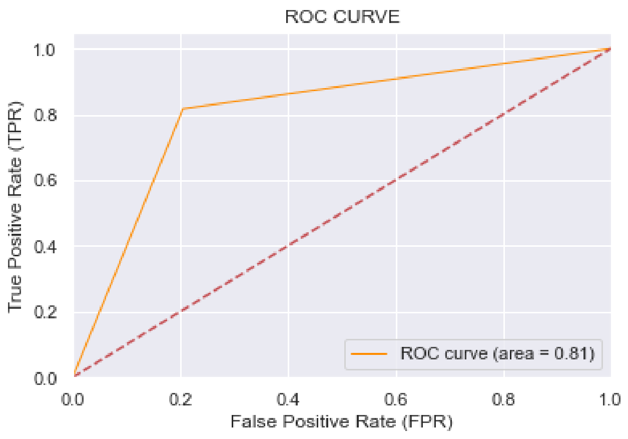


Logistic Regression
Simple algorithm to solve the binary classification problem

```python
LRmodel = LogisticRegression(C = 2, max_iter = 1000, n_jobs=-1)
LRmodel.fit(X_train, y_train)

y_pred2 = LRmodel.predict(X_test)

print(classification_report(y_test, y_pred2))
```

```
              precision    recall  f1-score   support

           0       0.81      0.79      0.80     10000
           1       0.80      0.82      0.81     10000

    accuracy                           0.81     20000
   macro avg       0.81      0.81      0.81     20000
weighted avg       0.81      0.81      0.81     20000
```



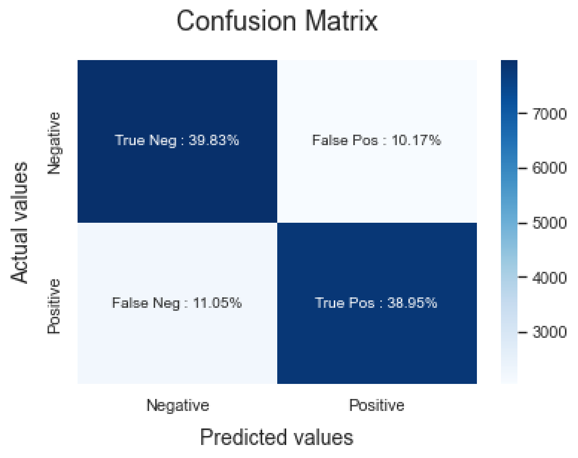


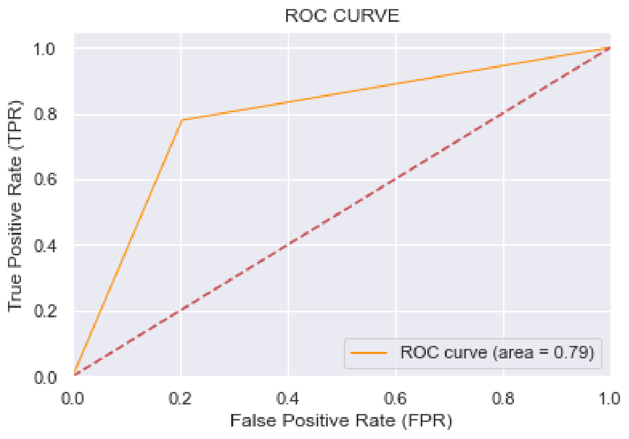
Naive Bayes Classifier for Multinomial Models

Suitable for classification with discrete features (e.g., word counts for text classification). The multinomial distribution normally requires integer feature counts. However, in practice, fractional counts such as tf-idf may also work.

```
mnb = MultinomialNB()

mnb.fit(X_train, y_train)

y_pred4 = mnb.predict(X_test)
print(classification_report(y_test, y_pred4))
```

```
              precision    recall  f1-score   support

           0       0.78      0.80      0.79     10000
           1       0.79      0.78      0.79     10000

    accuracy                           0.79     20000
   macro avg       0.79      0.79      0.79     20000
weighted avg       0.79      0.79      0.79     20000
```





Random Forest

Suitable for dealing with the high dimensional noisy data in text classification. An RF model comprises a set of decision trees each of which is trained using random subsets of features.

```
randomforest = RandomForestClassifier(random_state=42,n_jobs=-1,n_estimators=200)
```

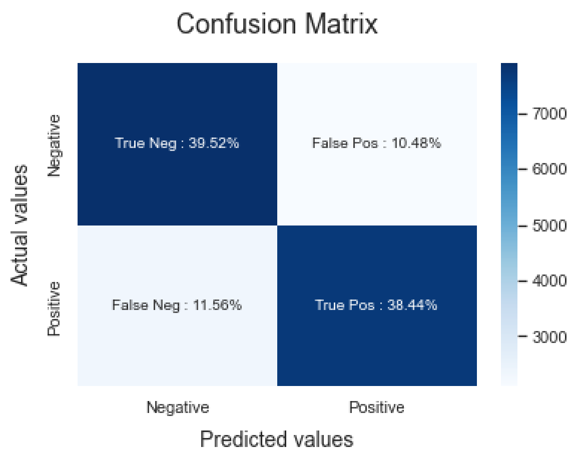
```
randomforest.fit(X_train, y_train)
```

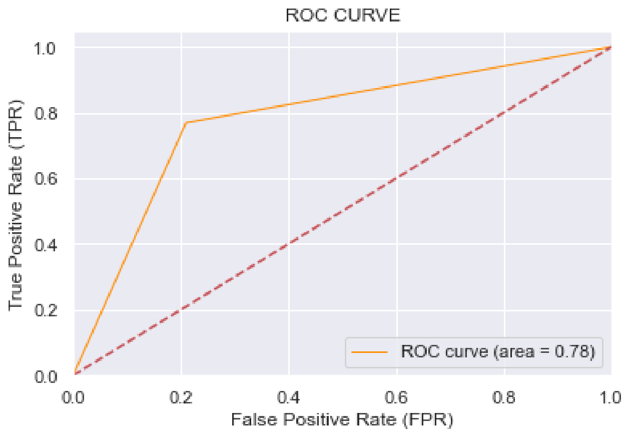
```
RandomForestClassifier(n_estimators=200, n_jobs=-1, random_state=42)
```

```
y_pred5 = randomforest.predict(X_test)
```

```
print(classification_report(y_test, y_pred5))
```

```
              precision    recall  f1-score   support

           0       0.77      0.79      0.78     10000
           1       0.79      0.77      0.78     10000

    accuracy                           0.78     20000
   macro avg       0.78      0.78      0.78     20000
weighted avg       0.78      0.78      0.78     20000
```





Results
Looking at the accuracy and the confusion matrix we can say that Logistic Regression is performing better compared to the other models shown above. After training the model we then apply the evaluation measures to check how the model is performing. Accordingly, we use the following evaluation parameters to check the performance of the models respectively:

- Accuracy Score
- Confusion Matrix with Plot
- ROC-AUC Curve

It achieves nearly 80% accuracy while classifying the sentiment of a tweet.
We will be also performing 3-fold cross-validation along with Grid Search to determine the best set of parameters on BernoulliNB, Random Forest, Logistic Regression and Linear-SVM.

Model	**Accuracy**	**Precision**		**Recall**		**F1-score**	
		Class 0	**Class 1**	**Class 0**	**Class 1**	**Class 0**	**Class 1**
Bernoulli Naïve Bayes	78%	0.79	0.78	0.77	**0.8**	0.78	0.79
Linear SVM	79%	0.8	0.79	0.79	**0.8**	0.79	0.79
Logistic Regression	**81%**	**0.81**	**0.8**	0.79	**0.82**	**0.8**	**0.81**
MultinomialNB	79%	0.78	0.79	**0.8**	0.78	0.79	0.79
Random Forest	78%	0.77	0.79	0.79	0.77	0.78	0.78

Based on the evaluation table we observe:
Accuracy:
Highest: Logistic Regression (accuracy = 81%)
Lowest accuracy Bernoulli Naïve Bayes and Random Forest (accuracy = 78%)
Precision:

(a) For class 0: Lowest: Random Forest (accuracy = 0.77)
Highest: Logistic Regression (accuracy = 0.81)
(b) For class 1: Lowest: Bernoulli Naive Bayes (accuracy = 0.78)
Highest: Logistic Regression (accuracy = 0.80)

Recall:
(a) For class 0: Lowest: Bernoulli Naive Bayes (accuracy = 0.77)
Highest: MultinomialNB (accuracy = 0.80)
(b) For class 1: Lowest: Random Forest (accuracy = 0.77)
Highest:Logistic Regression (accuracy = 0.82)

F1-score:
(a) For class 0: Lowest: Bernoulli Naive Bayes, Random Forest (accuracy = 0.78)
Highest: Logistic Regression (accuracy = 0.80)
(b) For class 1: Lowest: Random Forest (accuracy = 0.78)
Highest: Logistic Regression (accuracy = 0.81)

AUC Score:
Lowest: Bernoulli Naive Bayes, Random Forest (ROC-AUC = 0.78)
Highest: Logistic Regression (ROC-AUC = 0.81)

We, therefore, conclude that the Logistic Regression is performing better compared to the other models on this dataset. In our problem statement, Logistic Regression is following the principle of Occam's Razor which defines that for a particular problem statement if the data has no assumption, then the simplest model works the best. Since our dataset does not have any assumptions and Logistic Regression is a simple model, therefore the concept holds true for the above-mentioned dataset.

Test Model
80% accuracy is good enough considering the baseline human accuracy is also pretty low in these tasks. We take some sample tweets from our test dataset to test our hypothesis.

	Tweet	**Sentiment(predict)**
1	@stellargirl I loooooooovvvvvveee my Kindle2. ...	Negative
2	Reading my kindle2... Love it... Lee childs i...	Positive
3	Ok, first assesment of the #kindle2 ...it fuck...	Positive
4	@kenburbary You'll love your Kindle2. I've had...	Positive
5	@mikefish Fair enough. But i have the Kindle2...	Positive

Sentiment Analysis Challenges
When it comes to sentiment analysis challenges, there are quite a few things that we struggle with in order to obtain sentiment analysis accuracy. Sentiment or emotion analysis can be difficult in natural language processing simply because machines have to be trained to analyze and understand emotions as a human brain does. Some few points:
Polarity: Words such as “love” and “hate” are high on positive (+1) and negative (-1) scores in polarity. These are easy to understand. But there are in-between conjugations of words such as

“not so bad” that can mean “average” and hence lie in mid-polarity (-75). Sometimes phrases like these get left out, which dilutes the sentiment score. We need to get familiar with different Sentiment analysis tools that can easily figure out these mid-polar phrases and words in order to give a holistic view of a comment.

Sarcasm: People use irony and sarcasm in casual conversations and memes on social media. The act of expressing negative sentiment using backhanded compliments can make it difficult for sentiment analysis tools to detect the true context of what the response is actually implying. This can often result in a higher volume of “positive” feedback that is actually negative.

Emojis: The problem with social media content that is text-based, like Twitter, is that they are inundated with emojis. NLP tasks are trained to be language specific. While they can extract text from even images, emojis are a language in itself. Most emotion analysis solutions treat emojis like special characters that are removed from the data during the process of sentiment mining. But doing so means that companies will not receive holistic insights from the data. We need to train the models to learn the correlation between words and different emojis.

Negations: Negations, given by words such as not, never, cannot, were not, etc. can confuse the ML model. For example, a machine algorithm needs to understand that a phrase that says, “I can’t not go to my class reunion”, means that the person intends to go to the class reunion.

Comparative sentences: Comparative sentences can be tricky because they may not always give an opinion. Much of it has to be deduced. For example, when somebody writes, “the Galaxy S20 is larger than the Apple iphone12”, the sentence does not mention any negative or positive emotion but rather states a relative ordering in terms of the size of the two phones. Sentiment analysis accuracy can be achieved in this case when a sentiment model can compare the extent to which an entity has one property to a greater or lesser extent than another property.

Conclusion

In this project, we learned various text processing and a word embedding technique, and implemented a Twitter sentiment analysis classification model on processed data.

Due to time constraints, it was not possible to use different Vectorization methods such as Bag of Words (BoW), Word2Vec, etc. Deep Learning models like RNN, LSTM, BERT etc. unfortunately, however it is planned to improve the accuracy beyond 80%.

That said, concepts and techniques learned in this project can be applied to a variety of natural language processing problems. Other thanTwitter sentiment analysis, similar techniques can be used for spam/fraud detection, building chatbots, text summarization and language translation models.

Resources

- https://towardsdatascience.com/social-media-sentiment-analysis-part-ii-bcacca5aaa39
- https://www.analyticsvidhya.com/blog/2020/04/beginners-guide-exploratory-data-analysis-text-data/
- http://karpathy.github.io/2015/05/21/rnn-effectiveness/
- https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
- https://www.kaggle.com/code/viznrvn/optimal-parameters-for-svc-using-gridsearch/notebook
- https://www.sciencedirect.com/science/article/pii/S1877050913001385
- https://towardsdatascience.com/machine-learning-nlp-text-classification-using-scikit-learn-python-and-nltk-c52b92a7c73a