INTRODUCTION TO DATA SCIENCE AND ANALYTICS

Project Delivery #7:
Final Report

Project Title:
Social Media Sentiment Analysis

Social Media Sentiment Analysis

Arda Bayram
Computer Engineering
Marmara University
Istanbul, Turkey
arda.bayram1998@gmail.com

Göksel Tokur
Computer Engineering
Marmara University
Istanbul, Turkey
gokseltokur04@gmail.com

Gülnihal Erdem
Industrial Engineering
Marmara University
Istanbul, Turkey
gulnihalerdem@gmail.com

İrem Seçmen
Electrical and Electronics Engineering
Marmara University
Istanbul, Turkey
iremsecmenn@gmail.com

***Abstract* - The generation of millions of tweets every day has made tweets an important source for understanding opinions of a specific group of people. By using different machine learning algorithms, sentiment analysis serves this purpose. In this project, a sentiment analysis was applied to a large dataset that consists of 1,600,000 tweets extracted using Twitter API. We applied pre-processing, descriptive and predictive analysis to the data. The tweets are classified using 6 models generated from several widely used machine learning algorithms such as LSTM, CNN and Naive Bayes. The best performing model for the dataset is determined after the mentioned analyses are completed.**

```` ``` ````

Keywords - sentiment analysis, NLTK, embedding, corpus

I. INTRODUCTION

Sentiment analysis, a sub-field of Natural Language Processing, is one of the most popular topics and research fields in data science. We will be working on social media sentiment analysis. We aim to be able to classify tweets, reviews and comments from social media as positive, negative or neutral.

The most important point of our project is data mining to collect a large amount of data from several sources. For this purpose, we found open source datasets such as Sentiment140 [1] and many others. After all the searching we decided to use the Sentiment140.

Most of the open-source datasets that we found on the internet are properly labeled and structured. Data collected by ourselves needs to be properly labeled. Then, we will go through the cleaning, preprocessing and separation of test and training data steps.

We searched for some tools for our project and found some popular and powerful open-source NLP frameworks in Python. We will probably use the Natural Language Toolkit (NLTK) [2]. It comes with all the pieces you need to get started on sentiment analysis.

II. RELATED WORK

One of the works on social media sentiment analysis (tweets, specifically) is [3]. They analyze the emotions behind the tweets by using distant supervision. The training data of their work includes emoticons such as :) and :(as noisy label and use the help of these emoticons to identify the true emotion behind the tweets.

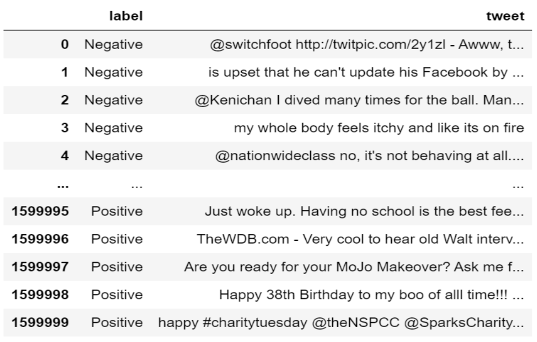
Another work on Twitter sentiment analysis is [4]. They use a combined lexicon-based and learning-based methods. They apply a chi-square test to the opinionated indicators that they have extracted, which helps them to identify the polarities of more tweets. In [5] tweets are analyzed using both target-dependent and context-aware. They use graph-based optimization by checking the related tweets. They also include syntactic features to distinguish the different targets within the tweet text.
````

There are several methods that we used in this project. One of them is the GloVe embedding method that helps to get the vector representation of the words. As expressed in [6], by using GloVe, the learning happens through finding the related words.

We applied three main methods for the classification/regression part of the project which are LSTM, CNN and Naive Bayes. These methods are selected due to their widespread use among sentiment analysis. Some works that use these methods are [3], [6], [7], [8], [9], [10]. One of these methods is LSTM. As stated in [7], LSTM is able to catch the long-term semantic dependence between the word sequences and the words whereas CNN, which is another method used for the classification, catches the close semantic relations in the text. The Naive Bayes method is based on Bayes' Theorem. It is a convenient method for the categorization of the text data as mentioned in [3].

III. APPROACH

First of all, we have to apply preprocesses to our dataset to avoid unexpected results. Our dataset [1] contains 1,600,000 tweets extracted using the Twitter API. The tweets have been classified from 0 (negative) to 4 (positive). The dataset contains 6 fields which are target as integer, ids as integer, date as date, flag as string, user as string and text as string. But we don't need to use all of these fields. For our purpose, we eliminated 4 fields which don't serve our purpose. After this elimination, our dataset has only two fields, which are label and tweet. The updated dataset is shown in Figure 1.



	label	tweet
0	Negative	@switchfoot http://twitpic.com/2y1zl - Awww, t...
1	Negative	is upset that he can't update his Facebook by ...
2	Negative	@Kenichan I dived many times for the ball. Man...
3	Negative	my whole body feels itchy and like its on fire
4	Negative	@nationwideclass no, it's not behaving at all....
...	...	...
1599995	Positive	Just woke up. Having no school is the best fee...
1599996	Positive	TheWDB.com - Very cool to hear old Walt interv...
1599997	Positive	Are you ready for your MoJo Makeover? Ask me f...
1599998	Positive	Happy 38th Birthday to my boo of alll time!!! ...
1599999	Positive	happy #charitytuesday @theNSPCC @SparksCharity...

Figure 1. The Sentiment140 dataset after preprocess

Finally, we removed the missing values from all dataset, and our dataset distribution after all transformations is shown in Figure 2.

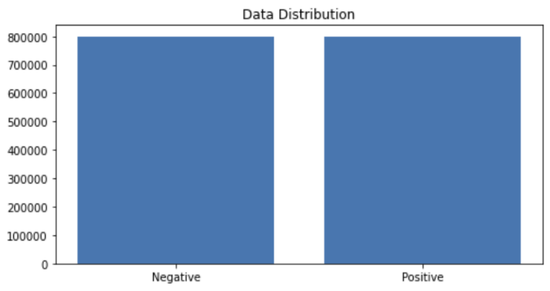


Figure 2. Data distribution of preprocessed dataset

We can train the embedding ourselves. However, that approach can take a long time to train. So, we use transfer learning techniques, and we use GloVe: Global Vectors for Word Representation.

The Global Vectors for Word Representation, or GloVe, algorithm is an extension to the word2vec method for efficiently learning word vectors, developed by Pennington, et al. at Stanford. It is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space. We downloaded the GloVe. Then, we initialize an embedding index that has 400000 word vectors, and an embedding matrix. We chose to use LSTM, CNN and Multinomial Naive Bayes algorithms for classification/regression.

IV. EXPERIMENT SETUP

First of all, we apply preprocessing to data and analysis in detail. In the preprocessing part, we applied data reduction and cleared stop words and punctuations from all instances. Then, we analyzed the data in terms of letter frequencies, distribution of the letters relative to the expected frequency of English language with chi-square test, word frequencies and their maximum, minimum and standard deviation. Then, we have analyzed the most common words in 2 classes that are positive and negative instances. Lastly, we used feature extraction methods, bag-of-words, and word embedding. Bag-of-words with TF-IDF is a common and simple way of feature extraction. We have created and analyzed correlation of words in corpus with this way.

For classification/regression experiments, the test set percentage is set to be 20%. 6 different models that are applied are CNN Model-1 with 1024 batch size, CNN Model-2 with 512 batch size, LSTM Model-1 with 1024 batch size, LSTM Model-2 with 512 batch size, Multinomial Naive Bayes Model-1 with Count vectorizer and Multinomial Naive Bayes Model-2 with TF-IDF vectorizer. We have chosen precision, recall, f1-score, AUC and ROC to evaluate our models.

V. EXPERIMENTAL RESULTS AND DISCUSSION

A. Preprocessing

In the preprocessing part of the project, we mainly have analyzed the data in 2 terms, which are letter and word.

Firstly, By counting the letters of the instances, we have analyzed frequency and relative frequency of the letters of the whole dataset. Then, we applied the chi square test to see whether the distribution of the letters in data is the same as what we expect from English texts.

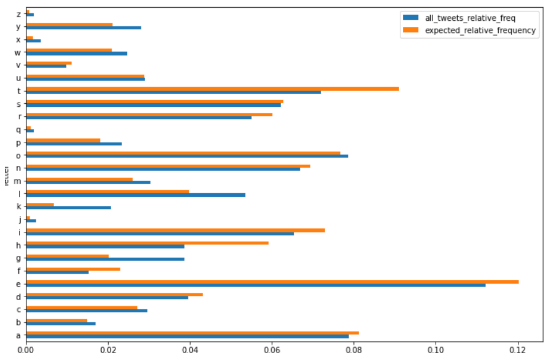
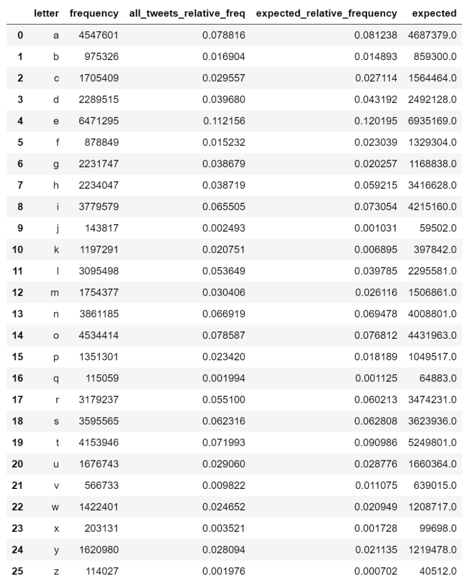


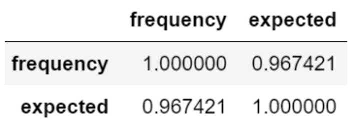
Figure 3. Letter frequencies of each 26 characters in English Alphabet.



	letter	frequency	all_tweets_relative_freq	expected_relative_frequency	expected
0	a	4547601	0.078816	0.081238	4687379.0
1	b	975326	0.016904	0.014893	859300.0
2	c	1705409	0.029557	0.027114	1564464.0
3	d	2289515	0.039680	0.043192	2492128.0
4	e	6471295	0.112156	0.120195	6935169.0
5	f	878849	0.015232	0.023039	1329304.0
6	g	2231747	0.038679	0.020257	1168838.0
7	h	2234047	0.038719	0.059215	3416628.0
8	i	3779579	0.065505	0.073054	4215160.0
9	j	143817	0.002493	0.001031	59502.0
10	k	1197291	0.020751	0.006895	397842.0
11	l	3095498	0.053649	0.039785	2295581.0
12	m	1754377	0.030406	0.026116	1506861.0
13	n	3861185	0.066919	0.069478	4008801.0
14	o	4534414	0.078587	0.076812	4431963.0
15	p	1351301	0.023420	0.018189	1049517.0
16	q	115059	0.001994	0.001125	64883.0
17	r	3179237	0.055100	0.060213	3474231.0
18	s	3595565	0.062316	0.062808	3623936.0
19	t	4153946	0.071993	0.090986	5249801.0
20	u	1676743	0.029060	0.028776	1660364.0
21	v	566733	0.009822	0.011075	639015.0
22	w	1422401	0.024652	0.020949	1208717.0
23	x	203131	0.003521	0.001728	99698.0
24	y	1620980	0.028094	0.021135	1219478.0
25	z	114027	0.001976	0.000702	40512.0

Figure 4. Letter frequency of the dataset, relative frequencies of the dataset, expected relative frequency according to the English language and expected character length according to the English language.

Then, we got the p-value (p) as 0 which implies that the letter frequency does not follow the same distribution with what we see in English tests, although the Pearson correlation is too high (~96.7%) as shown in Figure 6.



	frequency	expected
frequency	1.000000	0.967421
expected	0.967421	1.000000

Figure 5. Correlation.

We counted the number of characters for each tweet and analyzed the data frame according to maximum number of characters, minimum number of characters, mean of the number of characters column and its standard deviation. Our longest tweet is 189 characters long, the shortest tweet is 1 character long and mean of all tweets' character length 42.78. The standard deviation of all tweet character length is 24.16.

Secondly, we counted the number of words for each tweet and analyzed the data frame according to maximum number of words, minimum number of words, mean of the number of words column and its standard deviation. Our longest tweet is 50 words long, the shortest tweet is 1 word long and the mean of all tweets' word length is 7.24. The standard deviation of all tweet character length is 4.03.

Also, we have analyzed the most common words in 2 classes that are positive and negative instances.

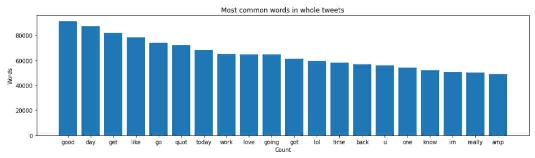


Figure 6. Most common words in our dataset.

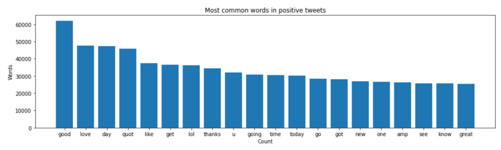


Figure 7. Distribution of most common words in positive tweets in our dataset.

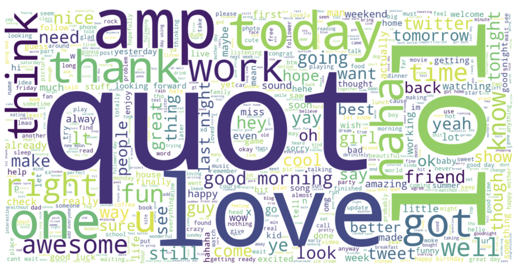


Figure 8. Most common words in positive tweets in our dataset.

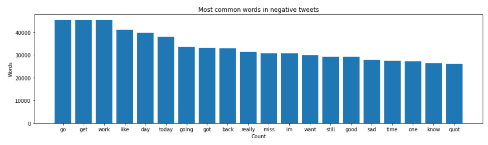


Figure 9. Distribution of most common words in negative tweets in our dataset.

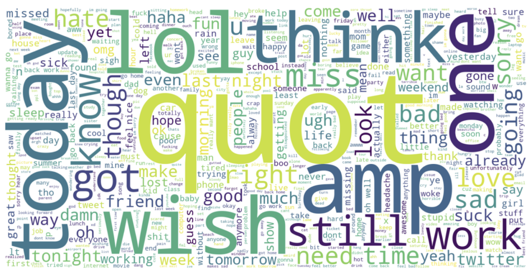


Figure 10. Most common words in negative tweets in our dataset.

We used feature extraction methods, bag-of-words, and word embedding. Bag of words with TF-IDF is a common and simple way of feature extraction. Bag-of-Words is a representation model of text data and TF-IDF is a calculation method to score the importance of words in a document.

After applying bag-of-words with TF-IDF, we create the scatter plot according to these results.

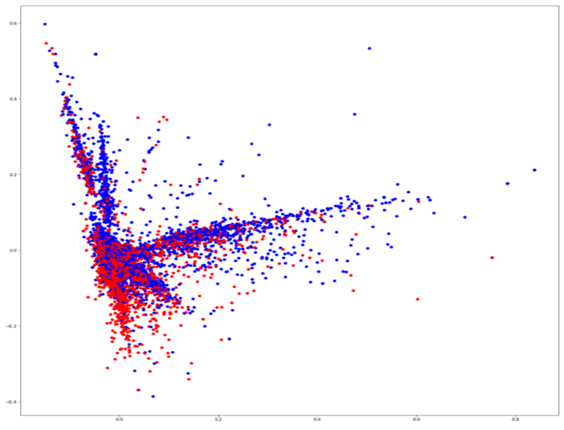


Figure 11. Scatter plot that shows correlation of words in the corpus: red indicates negatives, blue indicates positives.

B. Predictive Analysis

At the beginning, our dataset had 6 features which were target, id, date, query, user and text. We chose two of them for our purposes which are target and text. We can see that the entropy decreases significantly after this transformation.

Information gain
First entropy of dataset = 41.082
Entropy after preprocess = 14.733

For classification/regression experiments, the test set percentage is set to be 20%. 6 different models that are applied are CNN Model-1, CNN Model-2, LSTM Model-1, LSTM Model-2, Naive Bayes Model-1 and Naive Bayes Model-2.

CNN Model - 1 :
Conv1D = 64
Dense = 512
Dense = 512
1024 batch size

CNN Model - 2 :
Conv1D = 31
Dense = 256
Dense = 256
512 batch size

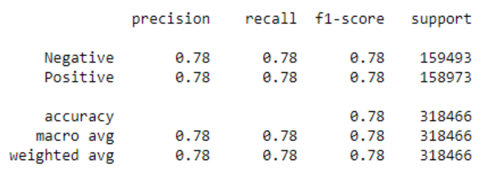
LSTM Model - 1 :
1024 Batch size

LSTM Model - 2 :
512 Batch size

Multinomial Naive Bayes Model - 1 :
Count Vectorizer

Multinomial Naive Bayes Model - 2 :
TF-IDF Vectorizer

Precision, recall, f1 score and accuracy of the models are shown below.

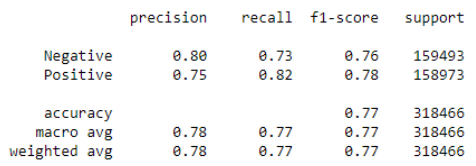


```
              precision    recall  f1-score   support

    Negative       0.78      0.78      0.78    159493
    Positive       0.78      0.78      0.78    158973

    accuracy                           0.78    318466
   macro avg       0.78      0.78      0.78    318466
weighted avg       0.78      0.78      0.78    318466
```

Figure 12. Precision, recall, f1 score and accuracy of the CNN Model-1

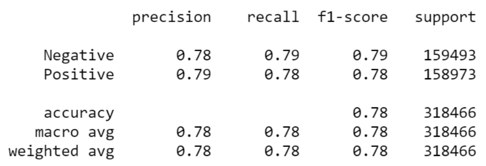


```
              precision    recall  f1-score   support

    Negative       0.80      0.73      0.76    159493
    Positive       0.75      0.82      0.78    158973

    accuracy                           0.77    318466
   macro avg       0.78      0.77      0.77    318466
weighted avg       0.78      0.77      0.77    318466
```

Figure 13. Precision, recall, f1 score and accuracy of the CNN Model-2

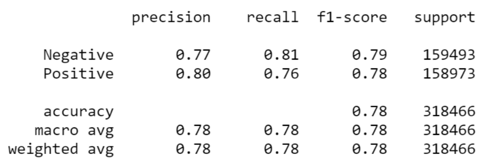


```
              precision    recall  f1-score   support

    Negative       0.78      0.79      0.79    159493
    Positive       0.79      0.78      0.78    158973

    accuracy                           0.78    318466
   macro avg       0.78      0.78      0.78    318466
weighted avg       0.78      0.78      0.78    318466
```

Figure 14. Precision, recall, f1 score and accuracy of the LSTM Model-1

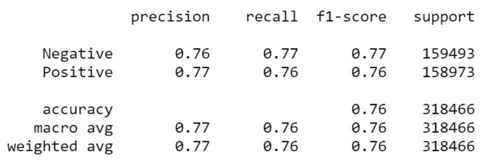


```
              precision    recall  f1-score   support

    Negative       0.77      0.81      0.79    159493
    Positive       0.80      0.76      0.78    158973

    accuracy                           0.78    318466
   macro avg       0.78      0.78      0.78    318466
weighted avg       0.78      0.78      0.78    318466
```

Figure 15. Precision, recall, f1 score and accuracy of the LSTM Model-2

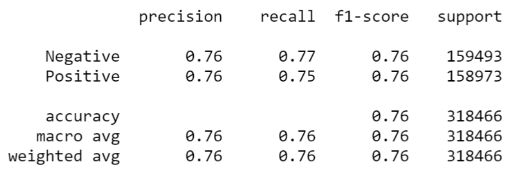


```
              precision    recall  f1-score   support

    Negative       0.76      0.77      0.77    159493
    Positive       0.77      0.76      0.76    158973

    accuracy                           0.76    318466
   macro avg       0.77      0.76      0.76    318466
weighted avg       0.77      0.76      0.76    318466
```

Figure 16. Precision, recall, f1 score and accuracy of the Multinomial Naive Bayes Model - 1 (CountVectorizer)



```
              precision    recall  f1-score   support

    Negative       0.76      0.77      0.76    159493
    Positive       0.76      0.75      0.76    158973

    accuracy                           0.76    318466
   macro avg       0.76      0.76      0.76    318466
weighted avg       0.76      0.76      0.76    318466
```

Figure 17. Precision, recall, f1 score and accuracy of the Multinomial Naive Bayes Model - 2 (TF-IDF Vectorizer)

After determining the evaluation metrics, ROC curves of the models are formed. Also, AUC values are calculated and shown at the bottom of each graph.

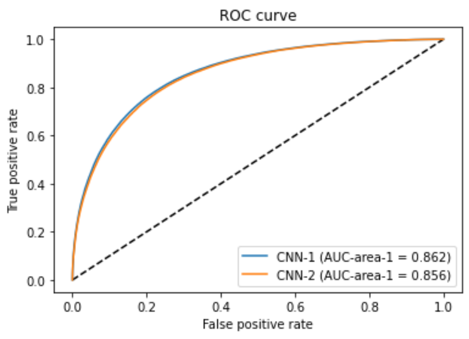


Figure 18. ROC Curve of CNN Model-1 and CNN Model-2

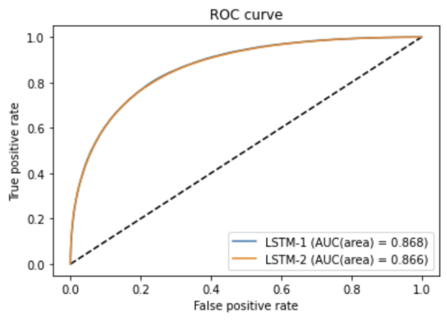


Figure 19. ROC Curve of LSTM Model-1 and LSTM Model-2.

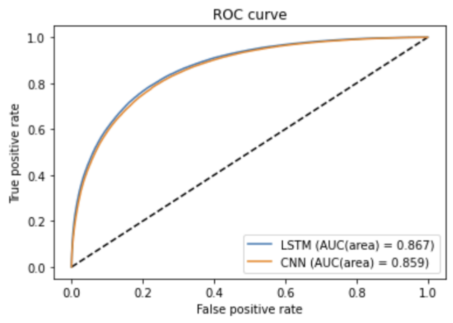


Figure 20. ROC Curve of best LSTM model and best CNN model.

Confusion matrices of the 6 model used to train the data, including the best performing model LSTM-1, are as follows:

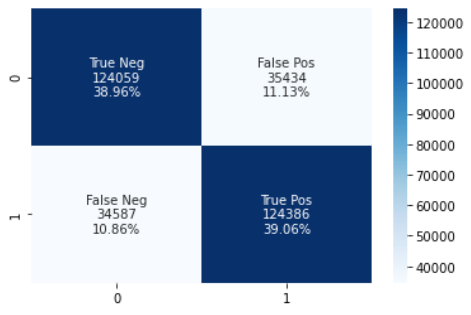


Figure 21. Confusion Matrix of CNN Model - 1.

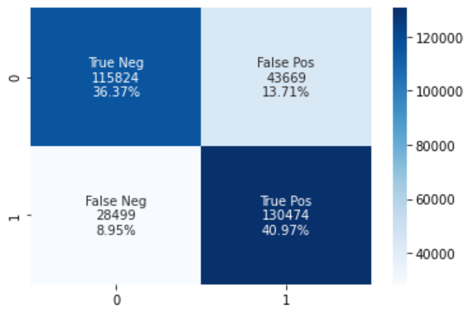


Figure 22. Confusion Matrix of CNN Model - 2.

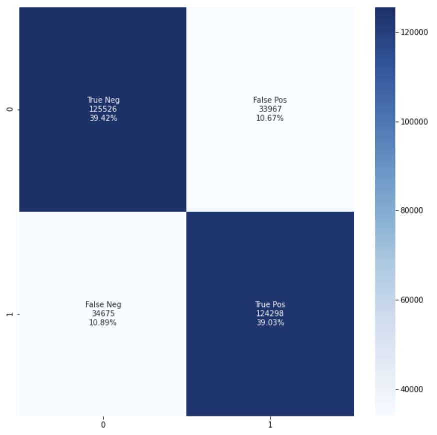


Figure 23. Confusion Matrix of LSTM Model - 1.

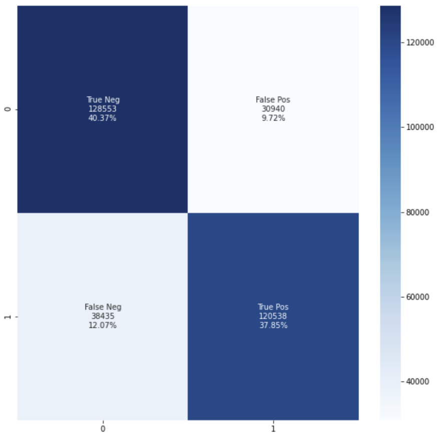


Figure 24. Confusion Matrix of LSTM Model - 2.

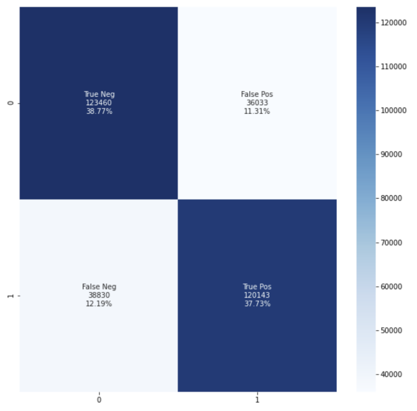


Figure 25. Confusion Matrix of Multinomial Naive Bayes with Count Vectorizer.

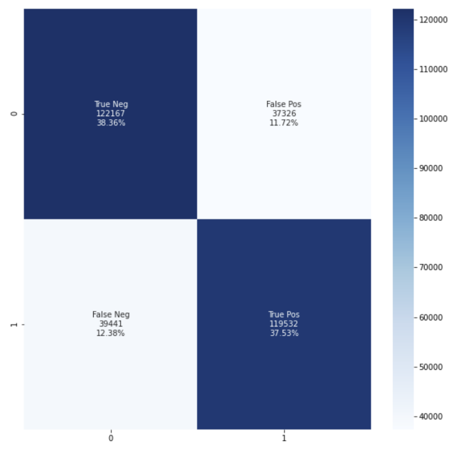
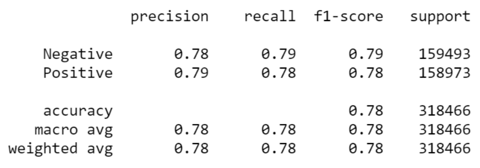


Figure 26. Confusion Matrix of Multinomial Naive Bayes with TF-IDF Vectorizer.

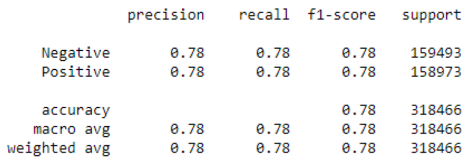
According to Accuracy, P, R, F1, AUC, our best performing model is LSTM model 1 with 1024 batch size and 0.789 accuracy and the closest competitor to LSTM model 1 is CNN model 1 with accuracy 0.781. Multinomial Naive Bayes with tf-idf is the worst performing algorithm among them, with accuracy 0.758.


```
              precision    recall  f1-score   support

    Negative       0.78      0.79      0.79    159493
    Positive       0.79      0.78      0.78    158973

    accuracy                           0.78    318466
   macro avg       0.78      0.78      0.78    318466
weighted avg       0.78      0.78      0.78    318466
```

Figure 27. Accuracy, P, R and F1 of LSTM Model-1.


```
              precision    recall  f1-score   support

    Negative       0.78      0.78      0.78    159493
    Positive       0.78      0.78      0.78    158973

    accuracy                           0.78    318466
   macro avg       0.78      0.78      0.78    318466
weighted avg       0.78      0.78      0.78    318466
```

Figure 28. Accuracy, P, R and F1 of CNN Model-1.

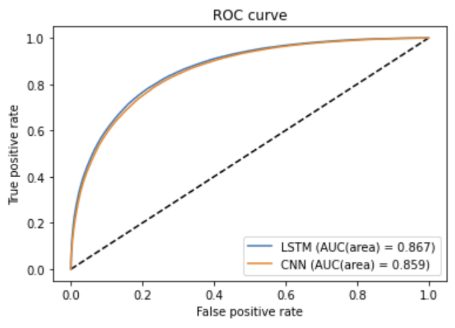


Figure 29. ROC - AUC Analysis of Best Performing Models.

VI. CONCLUSION

Our raw dataset has unnecessary features for our purpose. Its first entropy value was 41.08. Then we dropped the unnecessary columns, deleted the empty valued rows, and we have obtained an entropy value of 14.73. After this preprocess, we can easily see that there is an important change in entropy values.

After all six experiments, we can see that different LSTM and CNN give us very close accuracy ratios after training. Although there are really low differences, LSTM Model-1 has the best result and Naive Bayes models performed slightly worse.

Naive Bayes models have the best training time durations. It has very good speed compared to LSTM and CNN models. LSTM model-1, LSTM model-2 and CNN model-1 have close training times as each epoch takes 10 to 13 minutes for these models. Although changing the batch size in LSTM did not give an effective result difference, CNN model-2 has a better training time like 7 to 8 minutes for each epoch. Also, its accuracy is really close to the others.

LSTM model-1 has 78.9% accuracy rate with 1024 batch size and LSTM model-2 has 78.6% accuracy rate with 512 batch size. CNN model-1 has 78.2% accuracy rate with 1024 batch size and CNN model-2 has 77.2% accuracy rate with 512 batch size. Both algorithms have better training times with 512 batch size, are better than their 1024 batch sized models and their accuracy rates are really close. As a result of these, we can say that LSTM and CNN models with 1024 batch size are better for accuracy rate. But, models with 512 batch size have close accuracy rates within better training times.

For accuracy rates of Naive Bayes models there is a small difference like 1.5%. As a result of that, we can say that Naive Bayes with the CountVectorizer method gives better results than Naive Bayes with the TF-IDF method.

REFERENCES

[1]Sentiment140, http://help.sentiment140.com/home

[2] Natural Language Toolkit, https://www.nltk.org/

[3] Go, A., Bhayani, R. & Huang, L. (2009). Twitter Sentiment Classification using Distant Supervision. Processing, 1--6.

[4] Zhang, L., Ghosh, R., Dekhil, M., Hsu, M., & Liu, B. (n.d.). Combining lexicon-based and learning-based methods for twitter sentiment analysis. Retrieved June 20, 2021, from Hpl.hp.com website: https://www.hpl.hp.com/techreports/2011/HPL-2011-89.pdf

[5] Jiang, L., Yu, M., Zhou, M., Liu, X., & Zhao, T. (2011). Target-dependent Twitter sentiment classification. Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1, 151–160. USA: Association for Computational Linguistics.

[6]Abid, F., Li, C., & Alam, M. (2020). Multi-source social media data sentiment analysis using bidirectional recurrent convolutional neural networks. Computer Communications, 157, 102–115.

[7] Venkatesh, Hegde, S. U., A, Z., & Nagaraju. (2021). Hybrid CNN-LSTM model with GloVe word vector for sentiment analysis on football specific tweets. 2021 International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT), 1–8. IEEE.

[8] Swathi Lakshmi, B., Sini Raj, P., & Raj Vikram, R. (n.d.). Sentiment analysis using deep learning technique CNN with KMeans. Retrieved June 20, 2021, from Acadpubl.eu website: https://acadpubl.eu/jsi/2017-114-7-ICPCIT-2017/articles/11/6.pdf

[9]Truong, Q.-T., & Lauw, H. W. (2017). Visual sentiment analysis for review images with item-oriented and user-oriented CNN. Proceedings of the 2017 ACM on Multimedia Conference - MM '17. New York, New York, USA: ACM Press.

[10] Hermanto, D. T., Ziaurrahman, M., Bianto, M. A., & Setyanto, A. (2018). Twitter social media sentiment analysis in tourist destinations using algorithms naive Bayes classifier. Journal of Physics. Conference Series, 1140, 012037.