Emotion Detection from Text

TEXT ANALYSIS AND NLP PROJECT

Mesgari, Nastaran

CONSTRUCTOR UNIVERSITY PROFESSORS : **Meckel, Matthias. Wilhelm, Adalbert F.X**

CONTENTS

1 EXECUTIVE SUMMARY

In this project, I pursued several objectives:

I examined the text to determine whether the sentiment it conveys is positive, negative, or neutral. We investigated the best algorithm for identifying sentiment within this dataset. Using ChatGPT, I created a dataset with emotional elements present in the text. I categorized the information into positive, negative, and neutral labels in equal proportions and generated a balanced dataset. Then, I utilized the algorithm obtained in the first stage to evaluate them, observing how much the output is influenced by the fact that the text was generated by a machine. Considering the methods I had observed on Kaggle for sentiment analysis, I examined my data with different scenarios and algorithms. I compared the results with those obtained by myself.

In the first step, the texts in the comment dataset were preprocessed, which included text cleaning, tokenization, lemmatization, etc. Then the dataset was split into training and testing datasets, and they were both transformed into numerical vectors using Word2Vec word embedding algorithm. Then, several classification algorithms such as logistic regression, decision tree, and LSTM were trained on the training dataset and were tested on the test dataset. Regarding the evaluation metrics, the classifiers' performance was relatively poor.

In the next object, ChatGPT language model was used to generate texts, and the texts were labeled by Sentiments. Then, the dataset was cleaned using the same preprocessing steps and was transformed into numerical vectors using the word2vec model which was trained on the tweet_Comments dataset. Finally, the same classifier algorithms were used to predict the Sentiment on the dataset. In this part, once the classifiers which were trained on the dataset were used and again the ChatGPT dataset was divided into train and test datasets and the classifiers were trained using the training dataset and the used to predict on the test dataset.

The classifiers trained on tweet-comments dataset had better performance than those trained on ChatGPT dataset.

What was found from this analysis is that various methods can be employed to discern sentiments from text. However, in this approach, the best-performing algorithm is related to LDA. Contrary to the assumption that sentiment detection might be more discernible when generated by artificial intelligence, here, we observed a lower accuracy with the LDA (from 52 to 48 in accuracy for ChatGPT data set) algorithm, while the algorithm derived from real user comments yielded better results.

Furthermore, the use of TF-IDF for feature engineering facilitated swift algorithm evaluation, underscoring the efficiency of this approach in handling textual data. However, it's essential to note the significant time investment required for implementing techniques like Word2Vec, indicating the importance of balancing computational resources with performance gains. and this step between 3 algorithms by using TF-IDF the SVD is better than the MVP and MNB, it is about 59 in accuracy. And the same as LDA by using word2vec model.

Overall, the project sheds light on the diverse landscape of sentiment analysis techniques, emphasizing the need for continuous exploration and adaptation to effectively extract insights from textual data in various contexts.

2 INTRODUCTION

In this project, the primary objective was to explore various methods for sentiment analysis on textual data. Sentiment analysis plays a crucial role in understanding public opinion, customer feedback, and user sentiments across various platforms. With the advent of advanced natural language processing (NLP) techniques and machine learning algorithms, extracting sentiment from text has become increasingly feasible and valuable.

The project encompasses several key components:

Text Sentiment Analysis: The initial focus was on examining textual data to determine whether it conveys positive, negative, or neutral sentiments. Various algorithms and approaches were explored to identify the most effective method for sentiment analysis within the dataset.

Dataset Creation with Emotional Elements: Utilizing ChatGPT, a dataset was generated containing text with emotional elements. This dataset was carefully labeled with sentiments categorized as positive, negative, or neutral in equal proportions, ensuring a balanced representation of emotional states.

Algorithm Evaluation: The sentiment analysis algorithm obtained in the first stage was applied to evaluate the dataset. Notably, this evaluation included an investigation into how the output is influenced by the fact that the text was generated by a machine, thereby assessing the algorithm's robustness and generalizability.

Comparative Analysis: Drawing insights from methods observed on platforms like Kaggle, the project involved a comparative analysis of different scenarios and algorithms for sentiment analysis. The results obtained from these analyses were compared with those derived internally.

The project involved rigorous preprocessing of textual data, including tasks such as text cleaning, tokenization, lemmatization, and numerical vectorization using advanced techniques like Word2Vec word embedding. Classification algorithms, including logistic regression, decision trees, and Long Short-Term Memory (LSTM) networks, were trained and evaluated on the transformed datasets.

Details about the dataset:

- This public domain dataset is collected from data

-world platform, shared by @crowdflower, a data enrichment, mining and crowdsourcing company based in the US.

- The dataset consists of 40000 records of tweets labelled with 13 different sentiments, followed by Tweet ID number.

- All data types are strings, except for the 'tweet ID' column, which is an integer.

- There is a class imbalance of <21.32%. Tweets primarily convey neutral and negative sentiments.
- Contains 172 duplicate rows (tweets) but categorized with different sentiment labels.

- Word lengths vary, ranging from a minimum of 1 word to a maximum of 16 words.

- The data exhibits some degree of disorder and lack of cohesion. Various linguistic patterns are used to express sentiment. - Some information contains only special characters or hyperlinks.

- Contains informal and colloquial terms. For instance, “peeps” is a friendly term for “People”. - Contains self-made terms, slang or misspellings such as “Humpalow”.

2.0.1 IMPORTING LIBRARIES.

Code

	tweet_id	sentiment	author	content
0	1956967341	empty	xoshayzers	@tiffanylue i know i was listenin to bad habi...
1	1956967666	sadness	wannamama	Layin n bed with a headache ughhhh...waitin o...
2	1956967696	sadness	coolfunky	Funeral ceremony...gloomy friday...
3	1956967789	enthusiasm	czareaquino	wants to hang out with friends SOON!
4	1956968416	neutral	xkilljoyx	@dannycastillo We want to trade with someone w...
5	1956968477	worry	xxxPEACHESxxx	Re-pinging @ghostridah14: why didn't you go to...
6	1956968487	sadness	ShansBee	I should be sleep, but im not! thinking about ...
7	1956968636	worry	mcsleazy	Hmmm. http://www.djhero.com/ is down
8	1956969035	sadness	nic0lepaula	@charviray Charlene my love. I miss you
9	1956969172	sadness	Ingenue_Em	@kelcouch I'm sorry at least it's Friday?

2.0.2 DATA EXPLORATION

It involves exploring the characteristics of a dataset to gain insights and inform further analysis.

Code

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40000 entries, 0 to 39999
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   tweet_id   40000 non-null  int64 
 1   sentiment  40000 non-null  object
 2   author     40000 non-null  object
 3   content    40000 non-null  object
```

dtypes: int64(1), object(3)
memory usage: 1.2+ MB

we do not have any missing value.

Code

Text (0.5, 1.0, 'Missing value in the dataset')

check the unique value in the sentiment column. We have 13 categories for the sentiment column.

Code

['empty' 'sadness' 'enthusiasm' 'neutral' 'worry' 'surprise' 'love' 'fun'
'hate' 'happiness' 'boredom' 'relief' 'anger']

checking the frequency distribution of sentiment for finding the balancing

Code

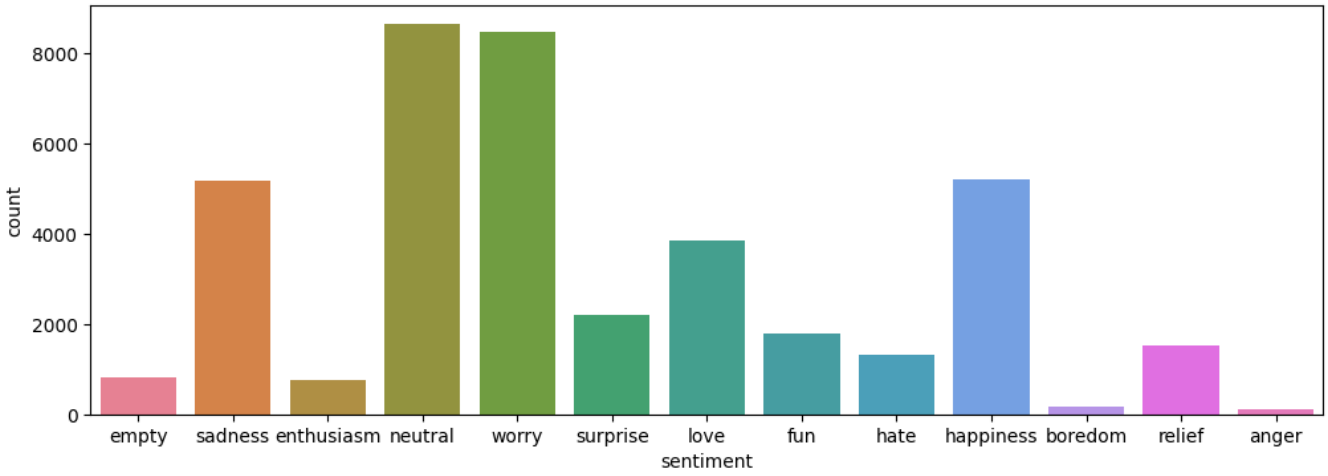
sentiment	Counts	Percentage
neutral	8638	21.5950
worry	8459	21.1475
happiness	5209	13.0225
sadness	5165	12.9125

love 3842 9.6050
surprise 2187 5.4675
fun 1776 4.4400
relief 1526 3.8150
hate 1323 3.3075
empty 827 2.0675
enthusiasm 759 1.8975
boredom 179 0.4475
anger 110 0.2750

total values: 13

Regarding the below count plot based on the sentiment, it can be seen that the dataset is imbalanced. And tweets are more about negative and neutral.

Code



for balancing the data I selected 3 groups, Positive, Negative and Neutral. Surprise sentiment was selected as a neutral because it is not shown is it positive or negative and when I do that the prevalence ≤ 10% and it is considered for misbalancing data. I created a new column, and the name is label for mapping the sentiment.

Code

Using this code to obtain the lengths of words in texts can help us gain a better understanding of textual data. Information such as the average length of texts can indicate to us how much the average number of words or the average length of texts is, which can be useful in decisions related to text processing and analysis. For example, if the average length of texts is very large or very small, there may be a need to adjust model parameters or use different methods in text processing. Additionally, this information can be used to determine more specific needs such as allocating computational resources or memory for text processing.

In addition, I check out the average length of the Comments are approximately 73.5 words.

Code

```
The average length of the texts is 73.40555.
```

3 TEXT PRE-PROCESSING

The text pre-processing is a very critical step because better results can only be achieved with good quality of data. Since the tweets texts are unstructured or in other words, they are raw and very noisy, they require cleaning. The main objective of this step is to remove noisy and inconsistent texts. Comments that carry very little weighting in text context, for example numbers, special character, punctuations, hashtags, extra blank space, etc. need to be removed. The following steps are taken for the text cleaning.

3.0.1 REMOVING SPECIAL CHARACTERS AND LOWERCASING.

Unstructured texts such as Comments may contain numbers, special characters, punctuations, hashtags, extra blank space, etc. We need to remove them from the texts. Therefore, I This function performs the following steps:

removing any word which starts with @.

removing all non-alphabetic characters in the string with spaces.

changing all the letters into lower case.

replacing more than one space with one space and removing any leading and trailing space.

filtering out any punctuation marks from the text.

3.0.2 TOKENIZATION

Tokenization involves dividing a text or document into smaller elements known as tokens, aiming to transform continuous text into distinct units for computer processing. It's an essential process in natural language processing and text analysis, facilitating various algorithms and methods that rely on tokenized data. In my approach, I utilized the wordpunct_tokenize() function from NLTK library, which efficiently separates text into individual tokens or words by identifying non-alphanumeric characters like punctuation marks or special symbols.

3.0.3 REMOVING NUMBERS

Eliminating numbers from text during text analysis can be crucial, especially considering the context and objectives of the analysis. Given that the project involves comments and tweets, with a focus on language patterns, numbers might not contribute significantly to the analysis. Thus, excluding numbers can enhance the consistency of the text. Following tokenization, I filtered out any tokens identified as digits.

3.0.4 REMOVING STOP-WORDS

Stop words, such as “the”, “a”, “an”, “and”, “but”, “in”, “on”, etc., are highly common in language and typically add little substantive meaning. Conversely, non-stop words, encompassing nouns, verbs, adjectives, adverbs, and other significant parts of speech, carry more weight in conveying meaning. In text analysis, filtering out stop words can enhance the relevance and clarity of the analysis. To achieve this, I compared each word in the Comments texts with NLTK’s pre-defined English stop words list, retaining only the non-stop words for further analysis.

3.0.5 ENGLISH LANGUAGE DETECTION

Ensuring that the texts are in English is vital for achieving accurate and effective text analysis. Since numerous NLP techniques and tools are tailored to specific languages, particularly English, analyzing English text enables the utilization of language-specific resources and models. To accomplish this, I implemented a filter on the tokenized texts, verifying whether each token belongs to the set of English words sourced from the nltk.corpus.words module.

3.0.6 LEMMATIZING THE WORDS.

Lemmatization simplifies words to their base or root form, known as the “lemma,” aiming to unify inflectional and sometimes derivational variations of a word. This process reduces the total number of unique words in a text, aiding in text analysis by standardizing vocabulary and mitigating data sparsity. For this project, I employed NLTK’s WordNetLemmatizer as the lemmatization tool, which utilizes morphological analysis and part-of-speech tagging to identify the base form of a word. Considering the part-of-speech is crucial, as the lemma may vary based on whether the word functions as a noun, verb, adjective, or adverb.

4 CLEANING THE DATASET

Code

I found the numerical column for removal because it wasn’t necessary in the process of the text analytics.

Code

```
Num of numerical variables: 1


The numerical variables are: ['tweet_id']
```

Code

in this step we remove the duplicate rows, it be shown 13 rows.

Code

```
Total duplicated rows: 13
```

Code

	sentiment	author	content	label
366	worry	jmil1733	I feel so deflated. No more doggy.	negative
521	worry	Chassidy7	Somebody please save the polar bears!	negative
1026	neutral	benmfowler	I'm at work	neutral
3684	sadness	nnurse	@dublins98dave me too! I am down 400 euro	negative
4363	worry	becca4656	is upset, I left my phone at home again	negative
...	...	...	...	...
39859	love	Miamarie33	Happy Mothers Day	positive
39898	love	xoxodominique	happy mothers day!	positive
39913	happiness	wailanik	happy mother's day!	positive
39915	love	ennahdii	happy mother's day everyone	positive
39945	love	chronophasia	Happy Mother's Day to all the moms out there!	positive

173 rows × 4 columns

in this step I reindex the data set

Code

after remove the duplicate rows we have 39827 rows with three columns

Code

(39827, 4)

Code

After creating a “label” column to examine the balance of frequencies related to the “label” column, I investigated it.

Code

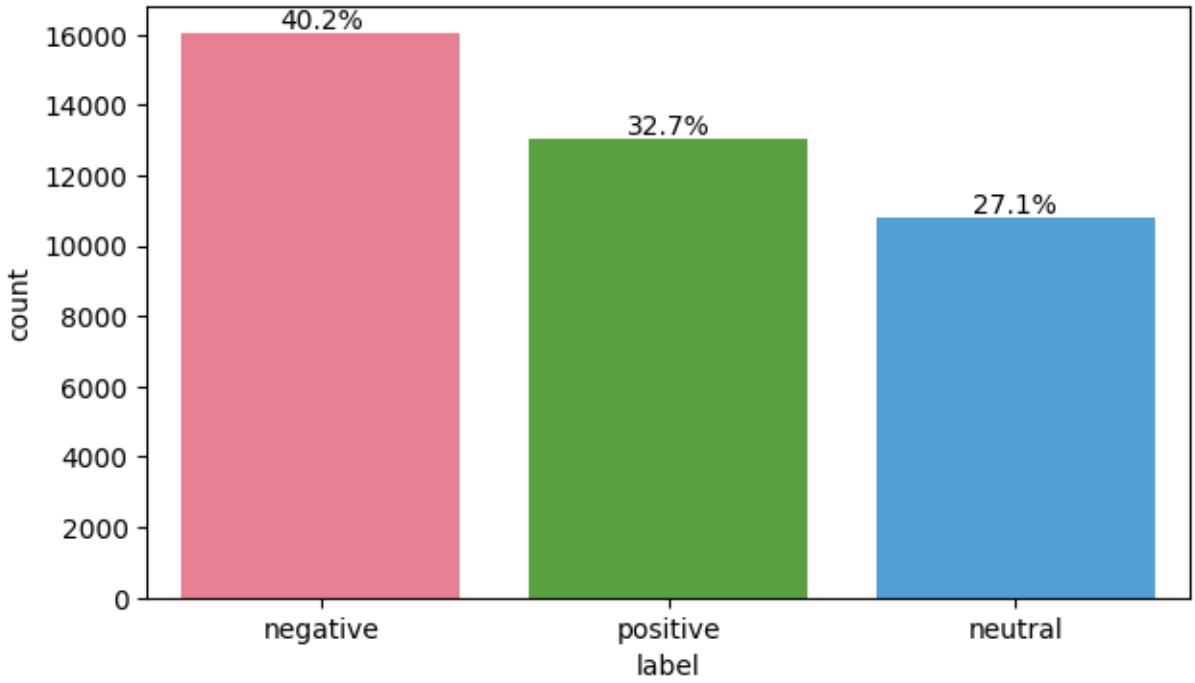
```
       Counts  Percentage
label
negative   16024    40.0600
positive   13024    32.5600
neutral    10779    26.9475

total values: 3
```

Code

['negative' 'positive' 'neutral']

As depicted in the graph, the frequency of negative instances is the highest, while positive instances are the lowest. However, since this difference is less than 10 percent, we consider it as balanced.

Code



for the statistical method, I mapped the label to an integer, and I dropped the label column because it is not necessary.

Code

	sentiment	author	content	label_num
0	empty	xoshayzers	@tiffanylue i know i was listenin to bad habi...	0
1	sadness	wannamama	Layin n bed with a headache ughhhh...waitin o...	0
2	sadness	coolfunky	Funeral ceremony...gloomy friday...	0
3	enthusiasm	czareaquino	wants to hang out with friends SOON!	1
4	neutral	xkilljoyx	@dannycastillo We want to trade with someone w...	2

I check the columns to be sure the column in content has label in label_num and we dont have missing data.and both of them have 39827 rows.

Code

39827

39827

Code

Index(['sentiment', 'author', 'content', 'label_num'], dtype='object')

To increase the process of cleaning the data I divide the data set into 2 parts.

Code

```
      sentiment      author  \
0         empty   xoshayzers
1       sadness    wannamama
2       sadness    coolfunky
3     enthusiasm  czareaquino
4       neutral    xkilljoyx
...         ...         ...
19995     worry      AP2005
19996      love   rissastory
19997    neutral    kridrules
19998  happiness      isabuu
19999  happiness    gAllethOo

                                 content  label_num
0     @tiffanylue i know  i was listenin to bad habi...          0
1     Layin n bed with a headache  ughhhh...waitin o...          0
2                Funeral ceremony...gloomy friday...          0
3                wants to hang out with friends SOON!          1
4     @dannycastillo We want to trade with someone w...          2
...                                        ...       ...
19995  I just want this semester to be over! Only a w...          0
19996  listening to i can't wait - akon feat t pain ....          1
19997        Cleaning my room and listening to britney.          2
19998  @aruky Yes, this NBA song is great!!!  Got an ...          1
19999   NEW LOGO! for all the Web, Cheak it!...gAllethOo          1

[20000 rows x 4 columns]
```

This code provides a function called “lemmatize” which is responsible for converting words into their common root forms (lemmatization). It utilizes the “WordNetLemmatizer” from the NLTK library to perform lemmatization.

The function first determines the Part-of-Speech (POS) tags for each input word. Then, it uses a dictionary to convert these tags into the required format by the “WordNetLemmatizer”. Subsequently, for each word, the common root is obtained using the “lemmatize” function from the “WordNetLemmatizer” and stored in a list. Finally, the list containing the common roots is returned as the output of the function.

Having common root forms for words in the text helps us improve text processing and analysis by reducing ambiguities, enhancing semantic understanding, reducing data size, and facilitating text preprocessing tasks, ultimately leading to better performance in various text-related tasks such as classification or named entity recognition.

Code

This code provides a function called “text_cleaner” that preprocesses text for use in other text processing tasks. The function employs several preprocessing steps to clean the input text and make it more usable for various text processing tasks:

Remove words starting with ‘@’. Replace non-alphabetic and non-numeric characters with spaces. Convert all letters to lowercase. Remove extra spaces. Tokenize the text. Remove digits. Remove words not found in the dictionary or identified as stop words. Lemmatize the words. Convert the list of tokens back to text. These steps collectively aim to refine the text data for further analysis and processing tasks.

Code

Now I use the function for the content column and create new column , that name is clean_content.this section time_consuming is about 10 minutes.

Code

```
<bound method DataFrame.info of       sentiment       author  \
0         empty    xoshayzers
1       sadness     wannamama
2       sadness     coolfunky
3    enthusiasm    czareaquino
4       neutral     xkilljoyx
...         ...          ...
39822    neutral  showMe_Heaven
39823       love      drapeaux
39824       love      JenniRox
39825  happiness      ipdaman1
39826       love    Alpharalpha
```

```
                                content  label_num  \
0     @tiffanylue i know  i was listenin to bad habi...         0
1     Layin n bed with a headache  ughhhh...waitin o...         0
2                Funeral ceremony...gloomy friday...         0
3              wants to hang out with friends SOON!         1
4     @dannycastillo We want to trade with someone w...         2
...                                    ...       ...
39822                          @JohnLloydTaylor         2
39823                Happy Mothers Day  All my love         1
39824  Happy Mother's Day to all the mommies out ther...         1
39825  @niariley WASSUP BEAUTIFUL!!! FOLLOW ME!!  PEE...         1
39826  @mopedronin bullet train from tokyo    the gf ...         1


                              clean content
0                           know bad habit part
1                             n bed headache call
2                         funeral ceremony gloomy
3                                        soon
4                          want trade someone one
...                                        ...
39822
39823                              happy day love
39824      happy mother day woman man long someone day
39825  beautiful follow peep new hit single wat u video
39826          bullet train visit japan since vacation
```

[39827 rows x 5 columns]>

4.0.1 INVESTIGATING STOP WORDS IN TEXT

for better finding the dataset I check the stop word and count it

Code

```
     sentiment       author  \
0        empty     xoshayzers
1      sadness      wannamama
3     enthusiasm    czareaquino
4      neutral      xkilljoyx
5        worry  xxxPEACHESxxx
```

```
...          ...          ...
39819    neutral    _Alectrona_
39820    neutral     bushidosan
39823       love       drapeaux
39824       love       JenniRox
39826       love     Alpharalpha

                                content  label_num  \
0     @tiffanylue i know  i was listenin to bad habi...         0
1     Layin n bed with a headache  ughhhh...waitin o...         0
3                wants to hang out with friends SOON!         1
4     @dannycastillo We want to trade with someone w...         2
5     Re-pinging @ghostridah14: why didn't you go to...         0
...                                     ...       ...
39819  @jasimmo Ooo showing of your French skills!! l...         2
39820  @sendsome2me haha, yeah. Twitter has many uses...         2
39823                  Happy Mothers Day  All my love         1
39824  Happy Mother's Day to all the mommies out ther...         1
39826  @mopedronin bullet train from tokyo    the gf ...         1

                           clean_content  \
0                        know bad habit part
1                        n bed headache call
3                                    soon
4                      want trade someone one
5                                  go like
...                                    ...
39819         show good lovely weather outside u
39820               yeah twitter many know care
39823                       happy day love
39824  happy mother day woman man long someone day
39826      bullet train visit japan since vacation

                                tokenized  stop_words
0                        [know, bad, habit, part]          6
1                        [n, bed, headache, call]          4
3                                    [soon]          3
4                      [want, trade, someone, one]          6
```

5 [go, like] 5
... ...
39819 [show, good, lovely, weather, outside, u] 6
39820 [yeah, twitter, many, know, care] 11
39823 [happy, day, love] 1
39824 [happy, mother, day, woman, man, long, someone... 12
39826 [bullet, train, visit, japan, since, vacation] 6

[36279 rows x 7 columns]

After text pre-processing, I removed these NaN or duplicate values from dataset. The shape of the dataset reduces into 39827.

Code

The number of NaN-values in the pre-processed dataset is: 0

5 DATA VISUALIZATION

5.0.1 COMPARING THE TEXT LENGTH WITH BOXPLOT AND HISTOGRAM.

In this phase of the project, I examined comment lengths across different label categories by introducing the content length variable to the dataset. Upon analyzing the statistical information of this column, it became apparent that some comments exceed 120 words, although their frequency is relatively low compared to the dataset size. To enhance visualization clarity in the box plot and histogram representations of the content length variable, I opted to filter out comments containing more than 120 words. However, despite this preprocessing step, neither the box plot nor the histogram revealed any significant differences among the label categories.

Code

	sentiment	author	content	label_num	clean_content	tokenized	content_length	tokenized_length	clean_content_length
0	empty	xoshayzers	@tiffanylue i know i was listenin to bad habi...	0	know bad habit part	[know, bad, habit, part]	92	4	19
1	sadness	wannamama	Layin n bed with a headache ughhhh...waitin o...	0	n bed headache call	[n, bed, headache, call]		4	19
2	sadness	coolfunky	Funeral ceremony...gloomy friday...	0	funeral ceremony gloomy	[funeral, ceremony, gloomy]	35	3	23
3	enthusiasm	czareaquino	wants to hang out with friends SOON!	1	soon	[soon]	36	1	4

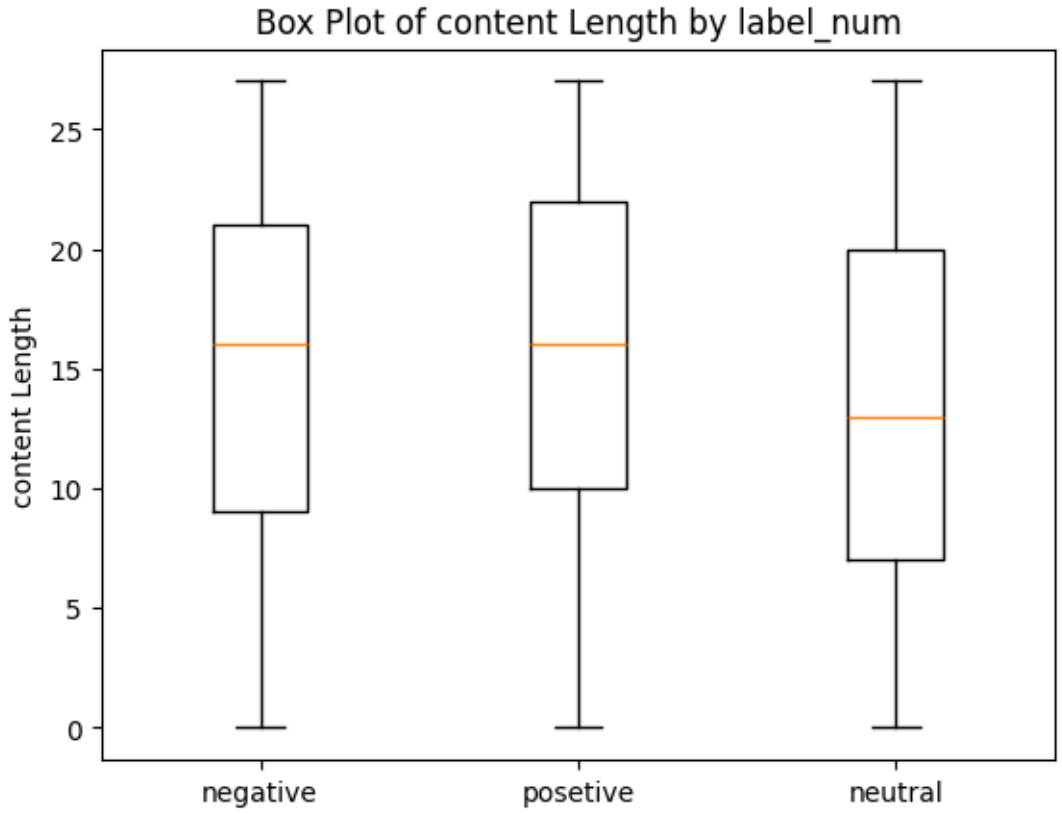
	sentiment	author	content	label_num	clean_content	tokenized	content_length	tokenized_length	clean_content_length
4	neutral	xkilljoyx	@dannycastillo We want to trade with someone w...	2	want trade someone one	[want, trade, someone, one]	86	4	22

Code

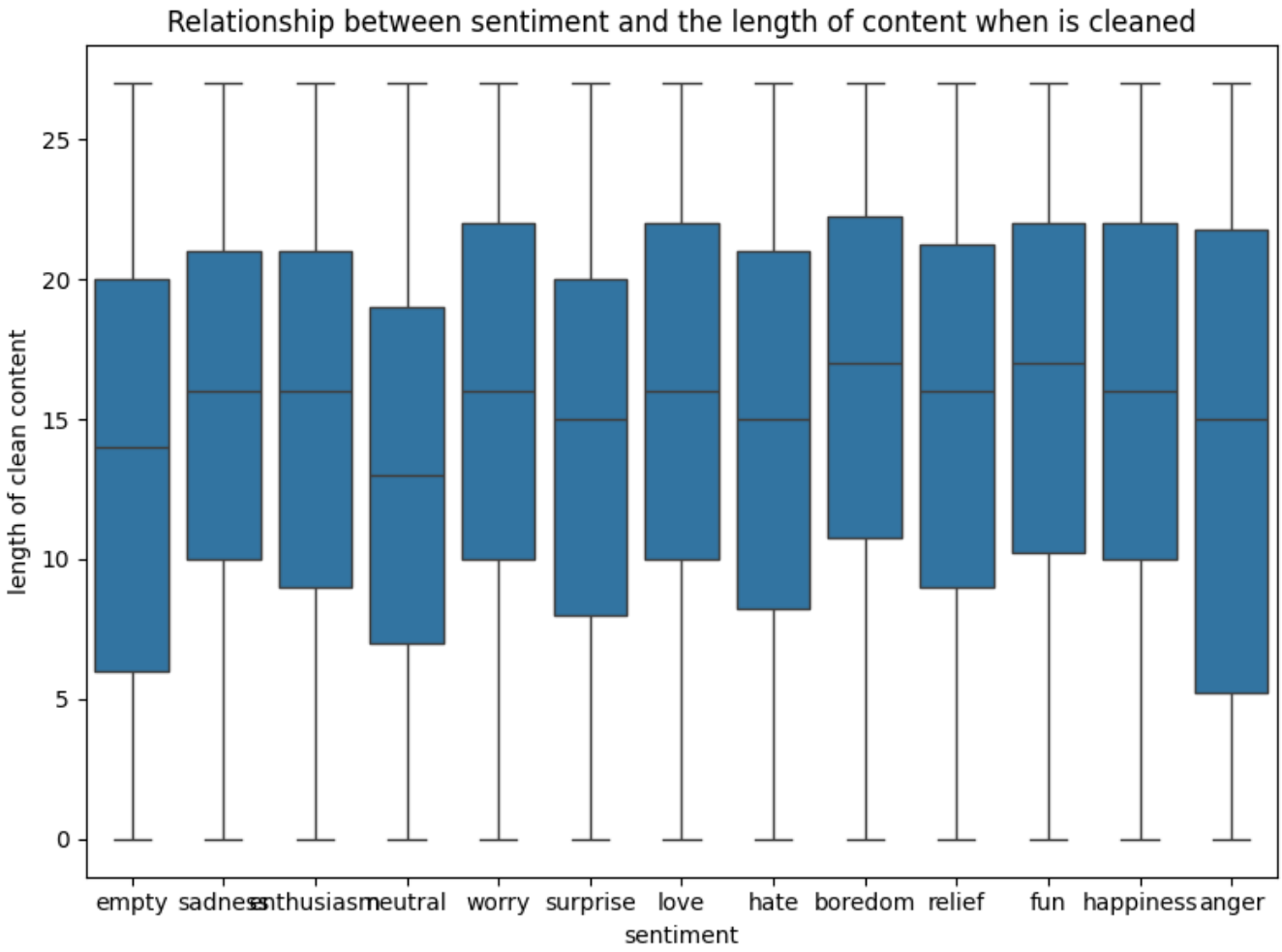
```
(39827, 9)
```

The below boxplots show that the length of the texts in all sentiment have almost the same distribution.

Code



Code

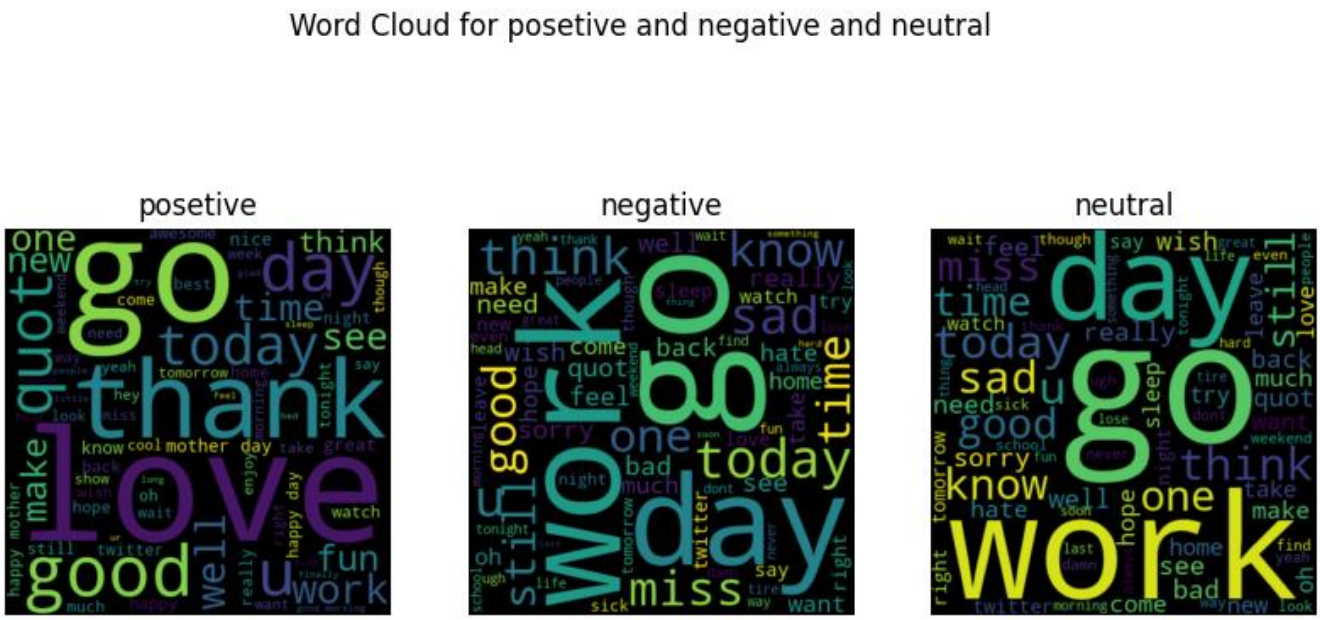


when the label is negative most sentiment is worry and sadness, and when it is positive most sentiment is happiness and love and when label is neutral the most sentiment is neutral and a little surprise in these categories we selected and labeled

5.0.2 WORD CLOUD

The word clouds for each label category (positive, negative, neutral) visually showcase the most prevalent words within their respective comments. Utilizing varying font sizes, the word clouds emphasize the frequency of each word, enabling quick recognition of prominent patterns or trends. Interestingly, negative comments exhibit recurrent words like “go” and “work,” indicating prevalent sentiments, while positive comments lack similarly prominent terms, suggesting a more diverse vocabulary in expressing positivity.

Code



5.1 DROPPING THE UNNECESSARY COLUMNS

The objective of this project is to train a classification model on the comments text and the sentiment label. So other columns (‘content’, ‘clean_content’, ‘content_length’, ‘tokenized_length’, ‘clean_content_length’) are removed from the dataset and it would be ready for next steps.

5.1.1 TRAINING THE WORD2VEC MODEL

I utilized Word2Vec technique to convert textual data into numerical vectors for training machine learning models. Word2Vec is a neural network-based approach that represents each word in a text as a high-dimensional vector, capturing semantic and syntactic relationships between words. This method is crucial for natural language processing tasks like text classification. I configured the Word2Vec model with the following hyperparameters:

- **Vector Size (vector_size)**: The dimensionality of the embedding vector for each word, which determines the number of dimensions in the continuous vector space used to represent words.

- **Window Size (window)**: The maximum distance between the current word and the predicted word within a sentence, influencing the scope of context words considered during prediction.

- **Minimum Word Count (min_count)**: The threshold counts of a word required for it to be included in the vocabulary. Words occurring less frequently than min_count are ignored during model training to filter out noise.

- **Algorithm (sg)**: This parameter specifies the Word2Vec algorithm used, where "skip-gram" (sg=1) predicts context words given a target word, and "continuous bag of words (CBOW)" predicts the target word given the context words.

Following the model configuration, I trained it using the entire text dataset and transformed the comments dataset into numeric vectors.

After training the Word2Vec model on all the comments, I proceeded to transform the dataset into numerical vectors. Initially, I divided the dataset into training and testing sets, reserving 20% of the data for testing purposes. Subsequently, for both the training and testing sets, I created separate files to store the word vectors.

Next, I computed the average word vector for each tokenized text. In each iteration, equivalent to the number of rows in the dataset, I filtered out any tokens do not present in the vocabulary of the pre-trained Word2Vec model (w2v_model) using list comprehension. Then, I calculated the meaning of the word vectors for the remaining tokens along the rows, resulting in a single vector of length 100. This resulting NumPy array was converted to a list. In cases where the result was not a list, indicating no tokens in the row matched the Word2Vec model's vocabulary, a line of zeros was written to the file instead.

Subsequently, I added the header row to the CSV file. The header row contained numbers ranging from 0 to 99, corresponding to the indices of elements in each word vector.

Code

```
Train Dataset shape:  (31861, 2)
Test Dataset shape:  (7966, 2)
```

```
invalid value encountered in scalar divide.
```

In general, each line in this CSV file contains the sentence class label and then the numeric vector corresponding to that sentence.

Code

	label_num	0	1	2	3	4	5	6	7	8
0	0	-0.044937	0.084955	-0.026313	-0.012374	0.017773	-0.247082	-0.043022	0.414152	-0.168104
1	2	-0.019493	0.043809	-0.327925	-0.117586	0.201473	-0.269427	-0.121567	0.650061	-0.130141
2	0	0.065516	0.139563	-0.072931	0.180659	-0.110970	-0.331901	0.257717	0.284429	-0.272371
3	1	-0.041268	0.049705	0.013300	-0.021870	-0.016283	-0.260949	-0.028070	0.343669	-0.158223
4	1	-0.025596	0.061945	-0.162854	-0.070937	0.041979	-0.274175	-0.072541	0.510699	-0.099315

5 rows × 101 columns

Code

	label_num	0	1	2	3	4	5	6	7	8
0	1	-0.012972	0.045507	-0.143181	-0.004083	0.023248	-0.240824	-0.009820	0.404257	-0.199855
1	1	-0.253301	-0.143929	0.029894	0.200798	-0.114317	-0.437478	-0.130571	0.393769	-0.076973
2	2	0.030080	0.098437	0.046356	-0.014123	0.067651	-0.246343	-0.031931	0.369123	-0.166014
3	0	-0.014718	0.106900	0.059657	-0.002311	0.052736	-0.247671	-0.062669	0.405076	-0.157858
4	1	0.036882	0.138428	-0.055992	-0.101299	0.097715	-0.255150	-0.017321	0.502747	-0.177074

5 rows × 101 columns

6 TRAINING THE CLASSIFICATION MODEL

In previous steps, the text dataset was cleaned and transformed into numerical vectors and now it can be used as the input of the classification algorithms. The **label_num** column is the label. In this part of the project, different classification algorithms such as **Logistic Regression, Linear Discriminant Analysis (LDA), Decision Tree, K-Nearest Neighbor, Random Forest Classifier** and **Gaussian** were trained on the training dataset and their performance were evaluated on the testing dataset.

In addition, the LSTM model was used for classification. LSTM stands for Long Short-Term Memory, and it is a type of artificial neural network architecture that is commonly used in natural language processing (NLP), speech recognition, and other sequence modeling tasks. I used the Keras library to define my LSTM model. The model has an LSTM layer with 100 neurons, followed by a dense output layer with three neurons (one for each class). As the input vectors have shape (100,) (i.e., 100 features), I reshaped them to have shape (100, 1) to match the input shape of the LSTM layer. The labels are multi (0 or 1 and 2). I trained the model on the training data and used the testing data as validation set during training. After training, I evaluated the model on the test dataset.

The table provides classification performance metrics for various models: LSTM, Logistic Regression (LR), Linear Discriminant Analysis (LDA), K-Nearest Neighbors (KNN), Decision Tree Classifier (DTC), Random Forest Classifier (RFC), and Naive Bayes (NB).

- **Accuracy**: Overall correctness of the model's predictions.
- **Precision**: Proportion of correctly predicted positive cases out of all cases predicted as positive.
- **Recall**: Proportion of correctly predicted positive cases out of all actual positive cases.
- **F1 Score**: Harmonic means of precision and recall, indicating balance between precision and recall.
- **ROC AUC**: Area under the Receiver Operating Characteristic curve, measuring the model's ability to distinguish between positive and negative classes.

LDA achieved the highest accuracy, precision, and F1 score among the models, indicating better overall performance. However, with most models scoring at or below 0.52 accuracy, the classifiers' performance on this dataset is generally poor. Several factors, including the complexity and diversity of

the comments texts and limitations of the chosen classification algorithms, may have contributed to the subpar results.

Model	LSTM	LR	LDA	KNN	DTC	RFC	NB
Accuracy	0.48	0.51	0.52	0.47	0.42	0.51	0.46
Precision	—-	0.50	0.51	0.45	0.42	0.49	0.46
Recall	0.48	0.47	0.48	0.44	0.42	0.49	0.43
F1 Score	0.48	0.46	0.47	0.43	0.42	0.49	0.42
ROC AUC	0.64	0.67	0.67	0.62	0.56	0.68	0.62

This section is too time consuming, and it is done about 1 hour.

Code

Epoch 1/10

996/996 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 216s 214ms/step - accuracy: 0.4218 - loss: 1.0751 - val_accuracy: 0.4360 - val_loss: 1.0624

Epoch 2/10

996/996 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 236s 237ms/step - accuracy: 0.4486 - loss: 1.0566 - val_accuracy: 0.4577 - val_loss: 1.0489

Epoch 3/10

996/996 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 216s 216ms/step - accuracy: 0.4516 - loss: 1.0556 - val_accuracy: 0.4597 - val_loss: 1.0574

Epoch 4/10

996/996 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 269s 270ms/step - accuracy: 0.4593 - loss: 1.0499 - val_accuracy: 0.4574 - val_loss: 1.0486

Epoch 5/10

996/996 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 228s 229ms/step - accuracy: 0.4651 - loss: 1.0464 - val_accuracy: 0.4485 - val_loss: 1.0618

Epoch 6/10

996/996 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 241s 242ms/step - accuracy: 0.4660 - loss: 1.0478 - val_accuracy: 0.4708 - val_loss: 1.0383

Epoch 7/10

996/996 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 257s 258ms/step - accuracy: 0.4686 - loss: 1.0363 - val_accuracy: 0.4725 - val_loss: 1.0333

Epoch 8/10

996/996 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 240s 241ms/step - accuracy: 0.4792 - loss: 1.0309 - val_accuracy: 0.4702 - val_loss: 1.0348

Epoch 9/10

996/996 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 244s 245ms/step - accuracy: 0.4770 - loss: 1.0307 - val_accuracy: 0.4785 - val_loss: 1.0307

Epoch 10/10

996/996 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 220s 221ms/step - accuracy: 0.4870 - loss: 1.0239 - val_accuracy: 0.4761 - val_loss: 1.0293

249/249 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24s 92ms/step

Confusion Matrix:

[[1820 909 450]

[830 1491 327]

[928 729 482]]

Accuracy: 0.4761486316846598

F1 Score: 0.46274330101865285

Recall: 0.4761486316846598

ROC AUC: 0.6437952562561863

Code

Classification Algorithms on Data

Classifier: LR

Accuracy: 0.51

Precision: 0.49

Recall: 0.47

F1 Score: 0.46

ROC AUC:0.67

Classifier: LDA

Accuracy: 0.51

Precision: 0.49

Recall: 0.47

F1 Score: 0.46

ROC AUC:0.67

Classifier: KNN

Accuracy: 0.47

Precision: 0.45

Recall: 0.44

F1 Score: 0.43

ROC AUC:0.61

Classifier: DTC
Accuracy: 0.42
Precision: 0.41
Recall: 0.41
F1 Score: 0.41
ROC AUC:0.55

Classifier: RF
Accuracy: 0.51
Precision: 0.50
Recall: 0.49
F1 Score: 0.50
ROC AUC:0.68

Classifier: NB
Accuracy: 0.46
Precision: 0.47
Recall: 0.43
F1 Score: 0.42
ROC AUC:0.61

7 CHATGPT GENERATED TEXTS.

In the second part of the project, I want to evaluate if the texts generated by ChatGPT can be classified based on emotions and answer the question if the use of artificial intelligence gives us have a better answer in the obtained model. Therefore, the first step of this part was the production of texts considering the emotion tag.

7.0.1 CREATING THE DATASET.

I asked ChatGPT to generate 200 sentences based on the sentiments I have, which were derived from the 13 types mentioned earlier. I requested the sentences to be balanced equally across all sentiments. Then, I saved the generated data in an Excel file. While I could have used the API command to accomplish this task and leverage the engine provided by ChatGPT, it required payment. Hence, I opted to manually request this task from ChatGPT.

After reading the ChatGPT dataset, I cleaned texts using the same methods which were used for cleaning the tweet dataset. Then I removed the unnecessary columns and plotted the word cloud for each label.

Code

	sentiment	comment	label
0	sadness	Feeling blue today, everything seems gloomy.	negative

	sentiment	comment	label
1	happiness	Grinning from ear to ear, life is beautiful!	positive
2	worry	Anxiety kicking in, can't shake off these thou...	negative
3	neutral	Feeling neither high nor low, just going with ...	neutral
4	love	Heart bursting with affection, surrounded by w...	positive

Code

	sentiment	comment	label	clean_content
0	sadness	Feeling blue today, everything seems gloomy.	negative	feel blue today everything gloomy
1	happiness	Grinning from ear to ear, life is beautiful!	positive	grin ear ear life beautiful
2	worry	Anxiety kicking in, can't shake off these thou...	negative	anxiety kick shake
3	neutral	Feeling neither high nor low, just going with ...	neutral	feel neither high low go flow
4	love	Heart bursting with affection, surrounded by w...	positive	heart affection surround warmth

Code

The number of NaN-values in the pre-processed dataset is: 0

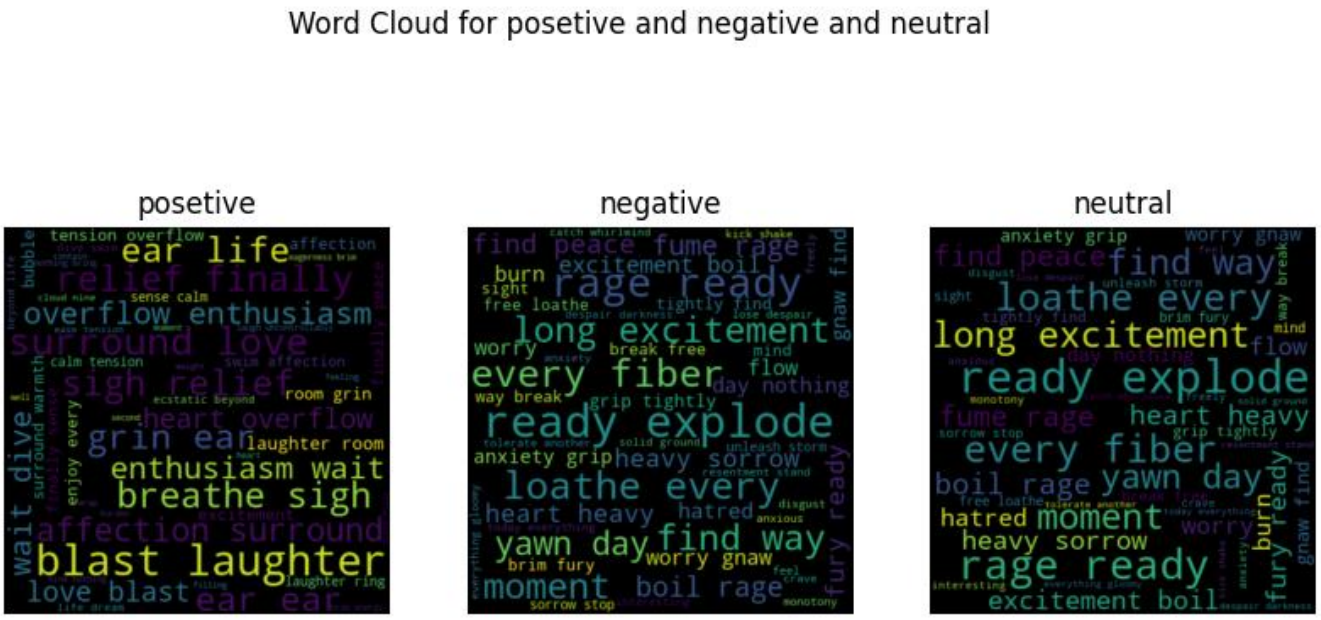
Code

	sentiment	comment	clean_content	label_num
0	sadness	Feeling blue today, everything seems gloomy.	feel blue today everything gloomy	0
1	happiness	Grinning from ear to ear, life is beautiful!	grin ear ear life beautiful	1
2	worry	Anxiety kicking in, can't shake off these thou...	anxiety kick shake	0
3	neutral	Feeling neither high nor low, just going with ...	feel neither high low go flow	2
4	love	Heart bursting with affection, surrounded by w...	heart affection surround warmth	1

in the excel file was generated by ChatGPT for ChatGPT dataset I try to have an equal data for each sentiment and the whole of the data set is the similar shape with the original data set in tweet comments.

in the excel was generated by ChatGPT I try to have an equal row for each sentiment and the whole data was same the original data in tweet comments.

Code



7.0.2 CLASSIFICATION ALGORITHM ON CHATGPT DATASET

In the following, the label_num column was transformed into 0, 1 and 2 then used the word2vec model which was trained on the tweet comment dataset to convert the ChatGPT texts into numerical vectors. Finally, the classification models were evaluated on the ChatGPT texts. In the evaluation part, firstly I tested the same classification models trained on the dataset on the ChatGPT data. Then, I divided the ChatGPT texts into train and test dataset and fitted the classification models on training data and evaluated their prediction on the training dataset.

The classification Metris for both approaches are presented in the following tables.

Classifiers on ChatGPT dataset, trained by tweet comment Dataset:

Model	LSTM	LR	LDA	KNN	DTC	RFC	NB
Accuracy	0.49	0.36	0.38	0.43	0.40	0.38	0.31
Precision	0.15	0.38	0.47	0.36	0.40	0.38	0.31
Recall	0.31	0.31	0.39	0.35	0.41	0.37	0.23
F1 Score	0.20	0.30	0.41	0.35	0.38	0.26	0.25
ROC AUC	0.66	0.67	0.67	0.62	0.55	0.64	0.62

Classifiers on ChatGPT dataset, trained by ChatGPT Dataset:

Model	LSTM	LR	LDA	KNN	DTC	RFC	NB
Accuracy	0.37	0.51	0.51	0.46	0.42	0.51	0.46
Precision	0.15	0.49	0.49	0.45	0.41	0.50	0.46
Recall	0.20	0.47	0.47	0.44	0.41	0.50	0.43
F1 Score	0.37	0.45	0.46	0.43	0.41	0.50	0.42
ROC AUC	0.61	0.67	0.67	0.62	0.55	0.68	0.62

The accuracy scores for all classifiers were around 40%, which means that they were not much better than random guessing. The precision scores were also low for most classifiers, indicating that the models were not very good at correctly predicting the sentiments.

The recall scores were particularly low for the classifiers trained on the ChatGPT dataset, while they were higher for the classifiers trained on the tweet comments dataset.

The ROC AUC scores were around 0.5, which is the same as random guessing, except for Naive Bayes algorithm trained on ChatGPT dataset, which was 62%.

Overall, the low classification metrics indicate that the models were not very effective in identifying the sentiment based on the generated texts by ChatGPT.

Code

	sentiment	comment	clean_content	label_num	tokenized
0	sadness	Feeling blue today, everything seems gloomy.	feel blue today everything gloomy	0	[feel, blue, today, everything, gloomy]
1	happiness	Grinning from ear to ear, life is beautiful!	grin ear ear life beautiful	1	[grin, ear, ear, life, beautiful]
2	worry	Anxiety kicking in, can't shake off these thou...	anxiety kick shake	0	[anxiety, kick, shake]
3	neutral	Feeling neither high nor low, just going with ...	feel neither high low go flow	2	[feel, neither, high, low, go, flow]
4	love	Heart bursting with affection, surrounded by w...	heart affection surround warmth	1	[heart, affection, surround, warmth]

Code

	sentiment	comment	clean_content	label_num	tokenized
0	sadness	Feeling blue today, everything seems gloomy.	feel blue today everything gloomy	0	[feel, blue, today, everything, gloomy]
1	happiness	Grinning from ear to ear, life is beautiful!	grin ear ear life beautiful	1	[grin, ear, ear, life, beautiful]
2	worry	Anxiety kicking in, can't shake off these thou...	anxiety kick shake	0	[anxiety, kick, shake]
3	neutral	Feeling neither high nor low, just going with ...	feel neither high low go flow	2	[feel, neither, high, low, go, flow]
4	love	Heart bursting with affection, surrounded by w...	heart affection surround warmth	1	[heart, affection, surround, warmth]

Code

	label_num	tokenized
0	0	[feel, blue, today, everything, gloomy]
1	1	[grin, ear, ear, life, beautiful]
2	0	[anxiety, kick, shake]
3	2	[feel, neither, high, low, go, flow]
4	1	[heart, affection, surround, warmth]

8 KAGGLE

Purpose of this step:

To compare the performance of three classification algorithms

Algorithms and parameters used: - Multinomial Naive Bayes (MNB), Param: default - Multilayer Perceptron (MLP), Param: 100,100, a=0.01 - Support Vector Machine (SVM), Param: default

Results (accuracy):

MNB: 0.533

MLP: 0.53

SVM: 0.59

Best-performing algorithm and why:

Support Vector Machine.

Reason: SVM demonstrated the highest Accuracy, Precision and Recall rates. It also exhibited robustness to noise and changes in input data, making it the ideal choice for the this particular dataset. Additionally, it does not require significant computational resources, making it an approachable option.

- TF-IDF as Feature Engineering method

Rationale: TF-IDF exhibited higher accuracy compared to GloVe. Accuracy rates dropped by up to 20% after embedding text using GloVe.

Code

	sentiment	author	content	label_num
0	empty	xoshayzers	@tiffanylue i know i was listenin to bad habi...	0
1	sadness	wannamama	Layin n bed with a headache ughhhh...waitin o...	0
2	sadness	coolfunky	Funeral ceremony...gloomy friday...	0
3	enthusiasm	czareaquino	wants to hang out with friends SOON!	1
4	neutral	xkilljoyx	@dannycastillo We want to trade with someone w...	2

Exploring Cleaned Data & Investigating Stop words in Text

Code

```
     sentiment       author \
0        empty    xoshayzers
1      sadness     wannamama
3     enthusiasm   czareaquino
4      neutral     xkilljoyx
5        worry  xxxPEACHESxxx
...        ...          ...
39819    neutral   _Alectrona_
39820    neutral    bushidosan
39823       love     drapeaux
39824       love     JenniRox
39826       love   Alpharalpha

                                content  label_num \
0     @tiffanylue i know  i was listenin to bad habi...         0
```

```
1     Layin n bed with a headache  ughhhh...waitin o...        0
3                wants to hang out with friends SOON!         1
4     @dannycastillo We want to trade with someone w...        2
5     Re-pinging @ghostridah14: why didn't you go to...        0
...                                      ...       ...
39819  @jasimmo Ooo showing of your French skills!! l...        2
39820  @sendsome2me haha, yeah. Twitter has many uses...        2
39823                  Happy Mothers Day  All my love         1
39824  Happy Mother's Day to all the mommies out ther...        1
39826  @mopedronin bullet train from tokyo    the gf ...        1


      stop_words
0            6
1            4
3            3
4            6
5            5
...        ...
39819        6
39820       11
39823        1
39824       12
39826        6

[36279 rows x 5 columns]
```

8.1 TOKENIZATION

Code

```
<class 'list'>
39827
tiffanylue know wa listenin to bad habit earlier and started freakin at his part
layin bed with headache ughhhh waitin on your call
funeral ceremony gloomy friday
want to hang out with friend soon
dannycastillo we want to trade with someone who ha houston ticket but no one will
re pinging ghostridah14 why didn you go to prom bc my bf didn like my friend
i should be sleep but im not thinking about an old friend who want but he married now damn amp he want me 2 scandalous
hmmm http www djhero com is down
```

charviray charlene my love miss you

kelcouch m sorry at least it friday

cant fall asleep

choked on her retainer

ugh have to beat this stupid song to get to the next rude

brodyjenner if watch the hill in london will realise what tourture it is because were week and week late just watch itonlinelol

got the news

8.2 VECTORIZING USING TF-IDF

Vectorizing text data using TF-IDF (Term Frequency-Inverse Document Frequency) is a common and effective method in natural language processing (NLP) to convert text into numerical features for machine learning models.

8.3 EVALUATION MNB

To evaluate a Multinomial Naive Bayes (MNB) classifier, focus on interpreting the performance metrics produced after training the classifier and making predictions on the test set.

Code

Classification Report (Cross-Validation):

	precision	recall	f1-score	support
0	0.49	0.92	0.64	12834
1	0.63	0.45	0.53	10395
2	0.53	0.02	0.05	8632
accuracy			0.53	31861
macro avg	0.55	0.47	0.41	31861
weighted avg	0.55	0.53	0.44	31861

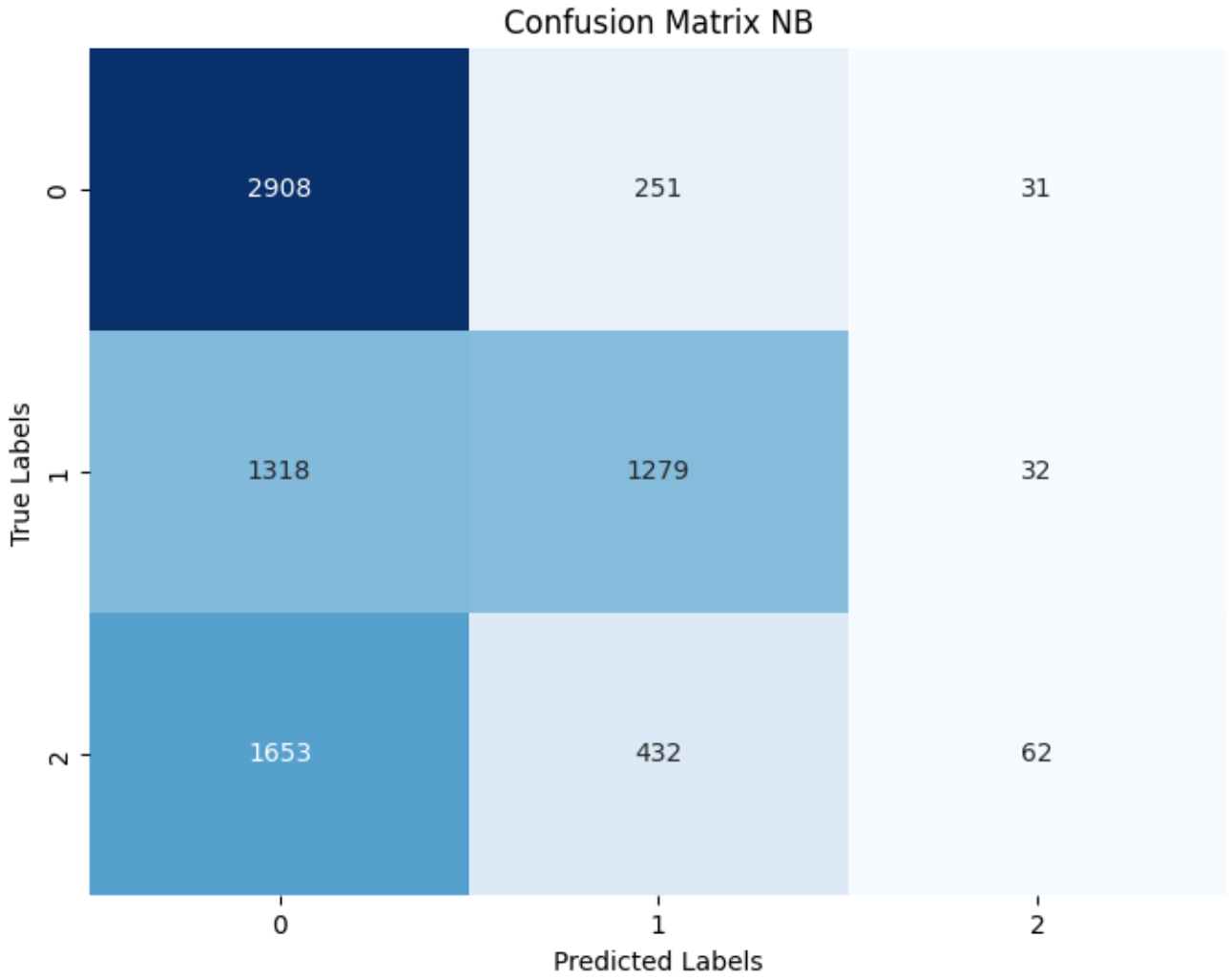
Accuracy (Cross-Validation): 0.5254386240231004

Classification Report (Test Data):

	precision	recall	f1-score	support
0	0.49	0.91	0.64	3190
1	0.65	0.49	0.56	2629
2	0.50	0.03	0.05	2147

accuracy			0.53	7966
macro avg	0.55	0.48	0.42	7966
weighted avg	0.55	0.53	0.46	7966

Accuracy (Test Data): 0.5333919156414763



Interpretation
Class 0:

High recall (0.91-0.92) but moderate precision (0.49). This indicates the model is good at identifying most instances of Class 0, but also has a high number of false positives, leading to lower precision.
Class 1:

Moderate precision (0.63-0.65) and recall (0.45-0.49). This shows the model has a balanced but moderate performance for this class. Class 2:

Precision around 0.50 but very low recall (0.02-0.03). This indicates the model is poor at identifying Class 2, missing most instances (high false negatives). Overall Accuracy:

Both cross-validation (52.54%) and test data (53.34%) accuracy are similar, indicating consistency but not high performance. Macro and Weighted Averages:

The macro averages (precision, recall, F1-score) indicate the model's overall balanced performance is low, especially in recall and F1-score, reflecting the poor performance on Class 2. Weighted averages are slightly better than macro averages because they account for the support of each class, showing that performance is somewhat skewed by the larger number of instances in Class 0 and Class 1. Conclusion The model shows reasonable performance for Class 0 and Class 1 but struggles significantly with Class 2. This is reflected in the low recall and F1-score for Class 2. The accuracy (~53%) suggests the model is slightly better than random guessing (assuming a balanced dataset).

8.4 EVALUATION MLP

we evaluate MLP and find the following data :

Code

Training interrupted by user.
Accuracy: 0.5289
Precision: 0.5233
Recall: 0.5289
F1-score: 0.5247

Training interrupted by user.

Accuracy: 0.5289

Precision: 0.5233

Recall: 0.5289

F1-score: 0.5247

8.5 EVALUATION SVM

For SVM We check the result

Code

Classification Report:

	precision	recall	f1-score	support
0	0.60	0.75	0.67	3190

1	0.65	0.59	0.62	2629
2	0.48	0.35	0.41	2147
accuracy			0.59	7966
macro avg	0.58	0.56	0.56	7966
weighted avg	0.58	0.59	0.58	7966

Accuracy: 0.5906351995982927

Here's a breakdown of what each metric and value means:

Class-Level Metrics Precision: This measures how many of the predicted positive instances for a class were actually correct. It's the ratio of true positives to the sum of true and false positives.

Class 0: 0.60 Class 1: 0.65 Class 2: 0.48 Recall: This measures how many of the actual positive instances for a class were correctly predicted. It's the ratio of true positives to the sum of true positives and false negatives.

Class 0: 0.75 Class 1: 0.59 Class 2: 0.35 F1-Score: This is the harmonic mean of precision and recall, providing a single metric that balances both concerns.

Class 0: 0.67 Class 1: 0.62 Class 2: 0.41 Support: This indicates the number of actual instances for each class in the test set.

Class 0: 3190 Class 1: 2629 Class 2: 2147 Overall Metrics Accuracy: This is the proportion of correctly predicted instances over the total number of instances.

Accuracy: 0.5906 (or 59.06%) Macro Average: This is the average of the precision, recall, and F1-score for all classes, treating each class equally.

Precision: 0.58 Recall: 0.56 F1-Score: 0.56 Weighted Average: This is the average of the precision, recall, and F1-score for all classes, weighted by the number of instances in each class (support).

Precision: 0.58 Recall: 0.59 F1-Score: 0.58 Interpretation Class 0 has relatively high precision (0.60) and recall (0.75), meaning the model is good at identifying this class and has fewer false negatives.

Class 1 has moderate precision (0.65) and lower recall (0.59), indicating the model is less reliable in identifying all instances of this class.

Class 2 has the lowest performance, with precision (0.48) and recall (0.35), indicating the model struggles significantly with this class.

Accuracy of 59.06% indicates that 59.06% of the total predictions were correct. T

The macro average suggests that, on average, the model's performance across all classes is somewhat balanced but not high.

The weighted average accounts for the support of each class, giving more weight to the performance on classes with more instances.

9 CONCLUSION

In this project, we evaluate sentiment recognition algorithms to determine their effectiveness. Firstly, we assess various algorithms to identify the most suitable one for this task. Subsequently, we analyze a dataset, generated by ChatGPT, to ascertain if the chosen algorithm's performance is affected. Additionally, we compare this algorithm with others sourced from Kaggle, aiming to enhance classification accuracy.

Our approach involves training word2vec and classification algorithms on a dataset comprising tweet comments. We preprocess the data, split it into training and testing sets, and convert it into numerical vectors using Word2Vec word embedding. We then train and test several classification algorithms on this dataset.

Following this, we utilize ChatGPT to generate texts and repeat the preprocessing steps to prepare the dataset. We apply the same classifiers to predict sentiment on this data. Surprisingly, the overall performance of the classifiers on both datasets is poor, with ChatGPT-generated data yielding worse results. This suggests that classification algorithms struggle to recognize texts generated by ChatGPT, even when assigned specific sentiments.

Furthermore, we explore additional classification algorithms available on Kaggle to improve sentiment recognition. While Support Vector Machine (SVM) shows promising accuracy, its implementation is notably time-consuming.

Apart from the insignificant effect of sentiment on the text structure, the poor performance could be due to various factors, such as the complexity and variety of the comment texts, as well as the limitations of the chosen classification algorithms. There are several steps that can be done in future to improve the performance as such applying alternative feature extraction techniques and word embedding models like GloVe or fastText, or investigate the use of contextualized embeddings like BERT, which may better capture the nuances in the texts. In addition, more sophisticated deep learning architectures such as recurrent neural networks (RNNs), attention mechanisms, or transformer models which are specifically designed for text classification tasks can be used. However, sentiment classification based on text can be a challenging task as people may use language differently and stereotypes may not always hold true.

10 REFERENCES

https://github.com/Mesgarin/NLPPJ

https://data.world/crowdflower/sentiment-analysis-in-text

https://kaggle.com/(search about NLP)

https://www.researchgate.net/publication/366484479_Sentiment_Analysis_of_Online_Lectures_Tweets_using_Naive_Bayes_Classifier?_sg=LDTFlOMrZxtn3gs6FCOs2Esue7ZyAXnfxRi0JIU91B_0Jl0JkXjsVIx4raE_i43Lp7Kwd9GwW95xUlM&_tp=eyJjb250ZXh0Ijp7ImZpcnN0UGFnZSI6InB1YmxpY2F0aW9uIiwicGFnZSI6Il9kaXJlY3QifX0

https://www.researchgate.net/publication/358917798_Sentiment_Analysis_for_COVID_Vaccinations_Using_Twitter_Text_Clustering_of_Positive_and_Negative_Sentiments?_sg=MqIWi_mkxEtX-P7HCvEG96u1rCbVdgorwgkpc-3H9qFoQNxF1TVLlW6-GUiw9VEf090DxJbDU4kxHCY&_tp=eyJjb250ZXh0Ijp7ImZpcnN0UGFnZSI6InB1YmxpY2F0aW9uIiwicGFnZSI6Il9kaXJlY3QifX0

ChatGPT was used mainly to generate texts and helped in solving codes error.