ChatGPT Sentiment Analysis

Huangan Pan, Yiyuan Li

Abstract

Twitter is one of the largest online social media platforms in the world. As of 2023, Twitter has approximately 450 million monthly active users, and nearly 6,000 tweets were posted on the platform every second during 2022. Twitter users are mainly concentrated in the 25-34 age group, accounting for almost 38.5% of its total users. Thus, tweets on this platform are highly helpful for the collection of sentiment data and the building of language models.

Given the popularity of ChatGPT, an AI chatbot created by OpenAI, there has been a significant amount of discussion about it in the public domain. To gain insights into people's sentiments towards ChatGPT, this article aims to conduct sentiment analysis and build natural language processing models. By analyzing what people think about ChatGPT, we can identify areas of improvement and monitor public opinion towards it over time. This analysis will be valuable not only for computer scientists working on ChatGPT but also for businesses and individuals who use AI chatbots in their operations. By leveraging public opinion data, we can improve the design and functionality of AI chatbots to better meet user needs and preferences.

1 Dataset

We used a dataset called ChatGPT sentiment analysis from Kaggle, which was from the public domain and uploaded by user CHARUNI SA in 2023. This dataset contains 3 columns, Serial number, tweets (text), and labels (sentiment), respectively, and there are 219,293 units of analysis

	text	labels
0	ChatGPT Optimizing Language Models for Dialogue	neutral
1	Try talking with ChatGPT our new AI system whi...	good
2	ChatGPT Optimizing Language Models for Dialogue	neutral
3	THRILLED to share that ChatGPT our new model o...	good
4	As of 2 minutes ago OpenAI released their new ...	bad

Figure 1

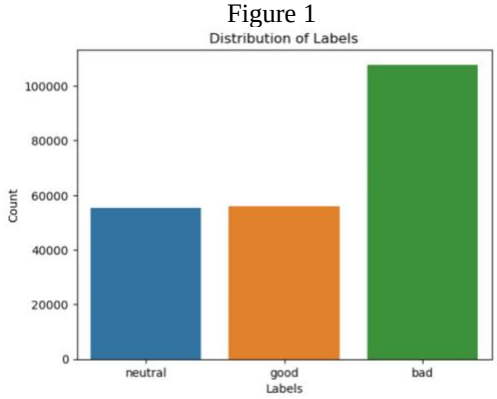

Figure 2

(rows) in this dataset. Figure 1 showed the first 5 rows in this dataset.

The data in this dataset is mainly collected from one recent month on Twitter, which is related to users' comments and opinions about ChatGPT. Column 'tweets' represents the tweets without preprocessing whereas column 'labels' represents the overall sentiment of that tweets. There are three categories under the column 'labels', 'good', 'neutral', and 'bad', respectively. Based on the preliminary descriptive analysis of this dataset, 49.16% of the tweets have a bad sentiment, 25.54% of the tweets have a good sentiment, and 25.30% have a neutral sentiment. Details are shown in Figure 2.

Furthermore, since the tweets contain detailed URLs, which are useless for the analysis, we conducted a certain level of data cleaning to remove the null value, some detailed words such as

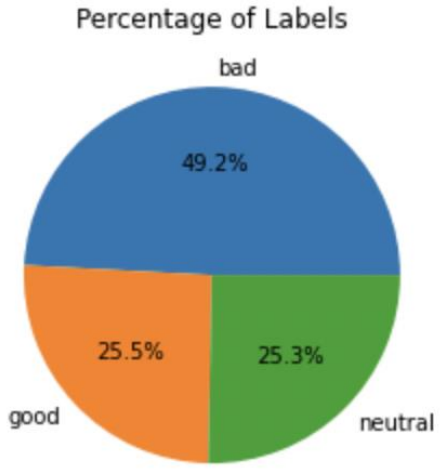

Figure 2

Figure 3: Overall

Figure 3: Good

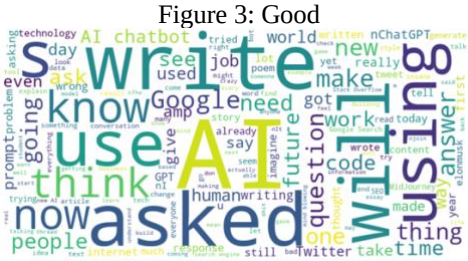

Figure 3: Bad

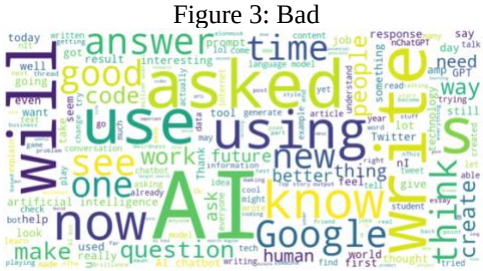

Figure 3: Neutral

'ChatGPT, chatgpt, OpenAI, etc.', and the URL after each tweet.

We also generated word clouds for all three categories, 'good', 'bad', and 'neutral', respectively, along with the overall word cloud to further discover the most frequent keywords in these three categories and explore the difference between these three categories. As shown in Figure 3, the overall most frequent keywords are AI, write, asking, using, know, etc. As for the 'good' category, the most frequent keywords are AI, asked, use, good, help, etc.; similarly, the most frequent keywords under the 'bad' category are AI, write, will, asked, using, etc. and under the 'neutral' category are AI, use, new, now, Google, etc. Overall, the four word clouds are very similar to each other with only very subtle differences. Tweets about ChatGPT on Twitter are often related to what AI can do now and that is also why there are many verbs in these word clouds.

2 Literature Study

Sentiment analysis, also referred to as emotion AI and opinion mining, is a technique used in natural language processing (NLP) to determine the emotional tone of a document. This is a common method used by businesses to identify and group opinions regarding a certain good, service, or idea. Machine Learning (ML), AI, computational linguistics, and data mining are used in sentiment analysis to mine text for sentiment and subjective information, such as whether the text is conveying positive, negative, or neutral sentiments.

In terms of exploring public sentiment towards ChatGPT, many researchers have conducted such studies based on generic domains, including social media platforms, online streaming platforms, and news platforms.

(Mubin UI Haque et al., 2022) also used Twitter data from early ChatGPT users to identify the topics and perform a qualitative sentiment analysis of each topic. Based on the results of the analysis, only a small percentage of users expressed concerns about problems like the potential misuse of ChatGPT, especially regarding topics like the Impact on educational aspects. The majority of early adopters expressed overwhelmingly positive sentiments related to topics like Disruptions to software development, Entertainment and exercising creativity.

(Dr.Öğr.Üyesi Ceren Çubukcu CERASİ et al. 2023) collected comments from YouTube regarding ChatGPT and wanted to perform a

sentiment analysis based on these comments. The attitudes of YouTube users about ChatGPT were examined in this study by using Long Short Term Memory (LSTM) classifier. They also applied several methods for identifying the polarity of comments, including User Sentiment Detection and Event Classification of YouTube comments. Nonetheless, many difficulties arise when analyzing users' sentiments on ChatGPT.

(Umar Bukar et al. 2023) conducted data mining from LinkedIn about ChatGPT in order to perform in-depth text analysis regarding the opinion of ChatGPT. Specifically, they used VOSviewer and found that plagiarism, references, citations, manuscripts, papers, and literature reviews are significant areas that involve the greatest concerns. These findings show that ChatGPT has the ability to both advance academic processes and increase academic misconduct. Therefore, it is still necessary to define the moral and proper restrictions of Large language model usage in order to better assist scientific writing.

3 Approaches

This dataset mainly contains 3 columns, and we want to discover the sentiments of each tweet and develop a reliable and accurate classification model to help predict the sentiment of given tweets. Thus, it is necessary for us to clean and preprocess the dataset in order to better fit the models we want to build. In this project, we processed the tweet data using a number of methods and created a new dataset with the useless information and data removed.

3.1 Missing Values

We downloaded the data from the Kaggle website and we could not make sure whether the dataset was clean or not, we decided to check if there was any null or missing value in this dataset. After the screening, as shown in Figure 4, the dataset was clean and there was no missing value or null value existed in this dataset.

3.2 Text Preprocessing

Before we conduct text vectorization and build NLP models, we need to preprocess the tweets in a standard way to make the texts format better for the future process. Specifically, we tokenized all the texts and removed all the punctuation, special characters, and English stop words, and we also transformed all uppercase into lowercase. Then, we lemmatized words to their root format. See Figure 5.

```
Missing values:
 Unnamed: 0    0
tweets         0
labels         0
dtype: int64
Summary statistics:
            Unnamed: 0
count   219294.000000
mean    109646.500000
std      63304.869303
min          0.000000
25%      54823.250000
50%     109646.500000
75%     164469.750000
max     219293.000000
```

Figure 4: Missing Values

```
In [25]: df.tweets.apply(pre_processing_by_nltk)
Out[25]: 0         [chatgpt, :, optim, languag, model, dialogu, h...
         1         [tri, talk, chatgpt, ,, new, ai, system, optim...
         2         [chatgpt, :, optim, languag, model, dialogu, h...
         3         [thrill, share, chatgpt, ,, new, model, optim,...
         4         [2, minut, ago, ,, @, openai, releas, new, cha...
                                    ...
         219289    [softwar, project, tri, replic, chatgpt, http,...
         219290    [ask, #, chatgpt, write, #, nye, joke, seo, de...
         219291              [chatgpt, disassembl, onli, dissembl]
         219292    [2023, predict, #, chatgpt, ., noth, realli, s...
         219293    [chatgpt, ,, neat, stuff, http, :, //t.co/qjju...
         Name: tweets, Length: 219294, dtype: object
```

Figure 5: Text Preprocessing

3.3 Text Vectorization

After text preprocessing, we need to convert the text data into a normalized numerical form, which is also referred to as vectors, so that they can be easily used to build models. In this project, we mainly used three types of text representation methods, which are Bag of Words (BoW), Term Frequency-Inverse Document Frequency (TF-IDF), and Word Embeddings such as Word2Vec. We will discuss the vectorization methods in detail in the experiment section of this article.

3.4 Model Training

Based on the above text vectorization methods, we mainly used one classification model for this project in order to construct models to predict the sentiment of tweets relevant to ChatGPT. The model we used is logistic regression. Logistic Regression is a statistical model that is often used for predictive analysis and classification. It estimates the likelihood of an event occurring. Multinomial logistic regression is a type of logistic

regression classification model that can deal with a dependent variable that has three or more possible categories but no specific order. In this project, we will use multinomial logistic regression since we have 3 different categories in sentiments but no specific order.

Besides, we also used the LDA (Linear Discriminant Analysis) algorithm based on the BoW vectorization. The LDA algorithm is a classification method that is widely used in the field of machine learning for predicting the class of a given sample based on a set of input features. It is a probabilistic model that estimates the probability of a sample belonging to a particular class. LDA is a type of linear classifier that works by projecting the data onto a lower-dimensional space. The goal of LDA is to find a linear combination of features that can separate the classes in the best possible way. Since we have three different categories in sentiments with no specific order, we used LDA with the multinomial option. This allows us to perform classification on a categorical response variable with more than two categories without assuming any particular order among them.

3.5 Model Evaluation

The goal of this project is to build reliable and accurate models to help computer scientists monitor the sentiment of the public toward ChatGPT and help them make decisions in response to public sentiment. Thus, after training the model, we have to evaluate the performance of the models we made and compare them to find out the model that has the best performance.

There are many model evaluation metrics we can use, such as accuracy, precision, recall, micro F1 score, macro F1 score, and Area Under the Receiver Operating Characteristic Curve (AUC-ROC).

Accuracy measures the percentage of true positives and true negatives in the dataset. Precision is a metric that focuses on the correct prediction of the positive class while recall is a metric that focuses on identifying true positives in the dataset. F1 scores consider both precision and recall and thus provide a more balanced way to measure the performance of models. AUC-ROC is a measurement for classification problems and ROC is a probability curve and AUC is the degree of separability. Usually, the higher the scores, the higher the performance. In this project, we mainly used F1 scores as our measurement metrics since they are more balanced in measuring our models.

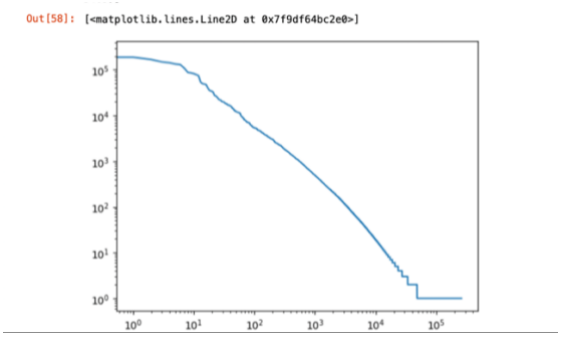

Figure 6: Zip’f Law

4 Experiment

4.1 Zip'f Law

We tested Zipf's law based on a corpus of text data. We first imported the necessary libraries, including defaultdict from the collections module, re for regular expressions, and matplotlib for plotting. It then created a corpus variable by joining all the tweets from a data frame, removes all non-alphanumeric characters from the corpus using regular expressions, and converts all words to lowercase.

Next, we code to create a defaultdict object called “freq”, which will be used to count the frequency of each word in the corpus. It then iterates over each word in the corpus, updating the count in the freq dictionary for each word. The resulting freq dictionary is sorted in descending order based on the frequency of each word, and the top 100 words are printed.

Finally, we plot the frequency distribution of the words using a log-log plot. The x-axis represents the word's rank in the frequency table, and the y-axis represents its frequency. The resulting plot can be used to visualize how well Zipf's law holds for the given corpus of text data.

The results show the top 100 words in the corpus, along with their frequencies. The most common word in the corpus is 'chatgpt', which appears 229,140 times, followed by 't', 'co', 'https', and 'the'.

4.2 Binary Vectorizer

First, we decided to use Binary-valued vector processing to convert textual data into binary vectors. This process is an important step of text

feature engineering. The function if_exist_vocab takes a document and a vocabulary as inputs and returns a binary-valued vector that represents the presence or absence of each word in the vocabulary in the input document. The resulting binary vector can then be used as input to machine learning models such as logistic regression for classification.

The first step in the preprocessing pipeline is tokenization, which involves splitting the input document into individual words or tokens. We use the pre_processing_by_nltk function for tokenization. After tokenization, we iterate over each token in the document and check whether it is present in the vocabulary. If the token is not in the vocabulary, it is replaced with a special token <UNK> to indicate that it is an unknown word. Then, the binary vector has a 1 for each word in the vocabulary that is present in the document and a 0 for each word that is not present.

The resulting binary vector can be used as input to machine learning models such as logistic regression for classification. We use the Logistic Regression model to train on the binary vectors using the fit method, and its performance is evaluated using the AUC-ROC score and the micro and macro F1 scores.

The value of AUC-ROC is 0.9305, which indicates that the classifier is quite good at distinguishing between the different categories. The value of Micro F-1 ranges from 0 to 1, with a higher value indicating better performance. In this case, the value of Micro F-1 is 0.8371, which indicates that the classifier is quite accurate in predicting both the positive and negative classes. The value of Macro F-1 is 0.8139, which indicates that the classifier is quite good at predicting each class separately.

```
AUROC: 0.9305568524710721
Micro F-1: 0.8371143892929616
Macro F-1: 0.8139846485241463
```

Figure 7: Logistic Regression

```
Accuracy: 0.8050571148452997
F1 Score: 0.7761177621328971
Recall: 0.7648387871576275
```

Figure 7: LDA

We also implemented the LDA algorithm to predict the target variable for a given set of features. We first created an instance of the LDA model and then fitted the model on the training data using the fit() method. After training, the model is used to predict the target variable for the test data using the predict() method. The predicted values are stored in y_pred.

The accuracy of the model is found to be 0.805, which indicates that the model correctly predicted 80.5% of the test samples. The F1 score is found to be 0.776, which suggests that the model has a good balance between precision and recall. The Recall score is found to be 0.765, which indicates that the model has a good ability to detect positive samples.

4.3 Frequency Vectorizer

We used tf() function takes a document (tweet) and a vocabulary as inputs. It first tokenized the document using a pre-processing function from the Natural Language Toolkit (NLTK) library. It then replaced any tokens not in the vocabulary with the <UNK> token (representing unknown words). Finally, it created a frequency vector of length equal to the size of the vocabulary, where each entry corresponds to the frequency of the corresponding word in the document.

The X list is initialized as an empty list, and then filled with the frequency vectors for each tweet in the dataset using the tf() function. The train_test_split() function from the scikit-learn library is used to split the dataset into training and testing sets, with 20% of the data being used for testing. The logistic regression model is trained on the training set using LogisticRegression().fit(), and its performance is evaluated on the testing set using the area under the ROC curve (roc_auc_score()) and the micro and macro F1 scores (f1_score()).

The resulting frequency vectors have a length equal to the size of the vocabulary, which is specified by the user. In this case, it appears that the vocabulary contains 400 words. Therefore, each frequency vector represents the frequency of each of the 400 words in the corresponding tweet.

The value of the AUC-ROC score is 0.9209, which indicates that the model has a good ability to distinguish between the classes. The Micro F-1 score of 0.8188 indicates that the model has a reasonably good performance on the test set. The Macro F-1 score of 0.7924 indicates that the

model's performance is slightly worse when considering each class separately.

```
AUROC: 0.9209178083570294
Micro F-1: 0.8188057183246312
Macro F-1: 0.7924243262751421
```

Figure 8: Logistic Regression

We also implemented the LDA model on a text classification problem using the frequency vectorizer. The frequency vectorizer converts the text data into numerical representation based on the frequency of words present in the text. The LDA model is then trained on the training data and used to predict the labels of the test data.

The accuracy of the LDA model trained on frequency vectorized data was 0.805, indicating that the model correctly classified 80.5% of the test data. The F1 score of the model was 0.776, which is a weighted average of precision and recall, and a measure of the model's overall performance. The recall score, which measures the proportion of actual positives that are correctly identified by the model, was 0.765.

```
Accuracy: 0.8050571148452997
F1 Score: 0.7761177621328971
Recall: 0.7648387871576275
```

Figure 8: LDA

4.4 TF-IDF Vectorizer

We also import the TfidfVectorizer from the scikit-learn library to transform text data into a numerical form for machine learning algorithms. The vectorizer is configured to remove accents, convert all text to lowercase, and tokenize the text using a pre-processing function defined in a separate module (pre_processing_by_nltk). The vectorizer is set to use inverse document frequency (IDF) weighting, which is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The normalization parameter is set to l2, which normalizes the vector by the Euclidean norm. Smooth_idf parameter is set to True which prevents division by zero error in case a token is not present in a document. Next, we train a Logistic Regression model on the vectorized features using the fit() method, with the input X_train and y_train data.

The performance of the trained model is evaluated on the test data, with the AUC-ROC, Micro F-1 score, and Macro F-1 score. The model achieved an AUC-ROC of 0.93, indicating a good performance. The Micro and Macro F-1 scores are 0.81 and 0.79 respectively, which means the model is fairly good at predicting the correct classes on the test data.

The AUC-ROC score is 0.932, which suggests that the classifier is quite good at distinguishing between the classes. The micro F-1 score is 0.817, which suggests that the classifier performs reasonably well on the test data. The macro F-1 score in this case is 0.787, which suggests that the classifier is good at identifying some classes but not so good at identifying others.

```
AUROC: 0.9322940479212146
Micro F-1: 0.8176657014523815
Macro F-1: 0.7869465361483954
```

Figure 9: Logistic regression

We also used the LDA model that is fitted to the training data using the tf-idf vectorizer. The accuracy of the model is 0.810, which indicates that the model can classify text data with high accuracy. The F1 score of the model is 0.777, which indicates that the model can achieve a balance between precision and recall. The recall score of the model is 0.769, which indicates that the model can identify a high proportion of true positives.

4.5 Word2Vec

We applied the powerful Word2Vec model to produce word embeddings for text classification. Firstly, we split the data from our CSV file into training and testing sets and preprocessed it by removing stopwords and punctuation with the preprocess_df() function. Next, we used the word_tokenize() function from the NLTK library to tokenize the preprocessed training data.

To build a vocabulary from the tokenized training data, we employed the build_vocab() function. This function calculated the frequency of each word in the training data and mapped each word to an index. Using the get_embeddings() function, we then generated word embeddings from the training data, which leveraged the Word2Vec algorithm to learn high-quality word embeddings.

Furthermore, we utilized the word embeddings to represent each document in the training and testing data as a vector. We calculated the vectors by taking the average of the embeddings of the words in the document.

To train the model, we employed the widely-used LogisticRegression() function from the scikit-learn library and optimized the hyperparameters of the model using grid search. We evaluated the accuracy of the model on the testing data.

By utilizing the Word2Vec algorithm, we successfully generated high-quality word embeddings and trained a Logistic Regression model that achieved an accuracy of 79% on the testing data. This demonstrates the effectiveness of Word2Vec for text classification tasks and provides a promising avenue for future research in the field.

The precision, recall, and F1-score metrics are reported for three classes: 'bad', 'good', and 'neutral'. The support column represents the number of instances for each class. The overall accuracy of the model is reported as 0.79, which means that it correctly predicted the class labels for around 79% of the instances. However, it is important to look at the individual class metrics to get a better understanding of the model's performance.

Looking at the precision scores, the model performs best for the 'bad' class with a precision score of 0.85, meaning that 85% of instances predicted as 'bad' were 'bad'. The precision score for the 'good' class is 0.73, and for the 'neutral' class is 0.72.

The recall score, which represents the proportion of true positives correctly identified, is highest for the 'bad' class with a score of 0.91. The recall score for the 'good' class is 0.81, and for the 'neutral' class is 0.53.

The F1-score, which is the harmonic mean of precision and recall, is also highest for the 'bad' class with a score of 0.88. The F1-score for the 'good' class is 0.77, and for the 'neutral' class is 0.61.

The model seems to perform well for the 'bad' and 'good' classes, with F1-scores of 0.88 and 0.77 respectively. However, the model performs relatively poorly for the 'neutral' class with an F1-score of 0.61. The lower recall score for the 'neutral' class indicates that the model is having difficulty correctly identifying instances of this class. It is also worth noting that the weighted average F1-score is 0.78, which suggests the model performs well across all classes when considering the class distribution. In conclusion, the output suggests that the model trained using word2vec embeddings performs well for the 'bad' and 'good' classes but has difficulty correctly identifying instances of the 'neutral' class.

	precision	recall	f1-score	support
bad	0.85	0.91	0.88	107796
good	0.73	0.81	0.77	56011
neutral	0.72	0.53	0.61	55487
accuracy			0.79	219294
macro avg	0.77	0.75	0.75	219294
weighted avg	0.79	0.79	0.78	219294

Figure 10

5 Conclusions

This article discussed the analysis of the sentiment of people towards ChatGPT using sentiment analysis and natural language processing models. The analysis is based on a dataset collected from Twitter, containing 219,293 tweets related to ChatGPT. The report also included testing Zipf's law based on a corpus of text data and used BoW and Word Embeddings to convert the words into vectors. We mainly used logistic regression and LDA to build our classification models, and from the seven models we built, we found that the logistic regression model using binary vectorizer is the most accurate based on the F1 score, see table 11.

Features	**Model**	**F1 Score**
Binary Vectorizer	LR	0.81
Binary Vectorizer	LDA	0.78
Frequency Vectorizer	LR	0.79
Frequency Vectorizer	LDA	0.78
TF-IDF Vectorizer	LR	0.79
TF-IDF Vectorizer	LDA	0.77
Word2Vec Embedding	LR	0.78

Table 11

There are still some limitations to our report. Although we tried different approaches used for text feature engineering, including binary vectorization, frequency vectorization, TF-IDF vectorization, and Word2Vec, there are still several different approaches to text feature engineering and different models may worth a try. SVMs, decision trees, and neural networks are all popular models for text classification tasks, and it may be worthwhile to try them out to see if they perform better than the models we have already explored. Additionally, there may be other feature engineering techniques that we don’t have enough

time to perform to improve performance. It is important to continue experimenting and iterating until we find the best possible model for our specific use case.

References

Daniel Ruby. 2023. 58+ Twitter Statistics For Marketers In 2023 (Users & Trends). https://www.demandsage.com/twitter-statistics/#:~:text=Twitter%20has%20around%20450%20million,users%2C%20with%2079.6%20million%20users

Mubin Ul Haque, Isuru Dharmadasa, et al. 2022. Exploring Sentiments of ChatGPT Early Adopters using Twitter Data. https://arxiv.org/pdf/2212.05856.pdf

Ceren Çubukçu Çerasi and Yavuz selim Balcioglu. 2023. SENTIMENT ANALYSIS ON YOUTUBE: FOR CHATGPT. https://www.researchgate.net/profile/Yavuz-Balcioglu/publication/368850748_SENTIMENT_ANALYSIS_ON_YOUTUBE_FOR_CHATGPT/links/63fdfcbe0d98a97717c5ac80/SENTIMENT-ANALYSIS-ON-YOUTUBE-FOR-CHATGPT.pdf

Umar Bukar, Md Shohel Sayeed, et al. 2023. Text Analysis of Chatgpt as a Tool for Academic Progress or Exploitation. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4381394

IBM. What is logistic regression? Accessed March 18th. https://www.ibm.com/topics/logistic-regression#:~:text=Resources-,What%20is%20logistic%20regression%3F,given%20dataset%20of%20independent%20variables.

Sarang Narkhede. 2018. Understanding AUC - ROC Curve. https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5#:~:text=the%20multiclass%20model%3F-,What%20is%20the%20AUC%20%2D%20ROC%20Curve%3F,capable%20of%20distinguishing%20between%20classes.