IMDB Review Usefulness Prediction using Sentiment Analysis and Machine Learning

Author: Jichen Jiang

Abstract

The aim of this project is to classify movie reviews on IMDB as useful or not by using sentiment analysis techniques and machine learning algorithms. The reviews are scraped for the top 250 movies on IMDB, and useful information such as upvote and downvote numbers, movie genres, and ratings for each movie are extracted. The upvote ratio is used to determine the usefulness of a review, and reviews with upvote ratios above the mean are considered useful. The Latent Dirichlet Allocation technique is used to cluster the reviews and provide latent topics. The NLTK part-of-speech tagging technique is used to obtain counts of crucial vocabulary types, and a numerical data frame related to the reviews is created. Tf-idf is used as the second feature extraction technique, and the results are used as features for the classification process. Finally, Convolutional Neural Networks (CNNs), are used as the final model to analyze the text data from the movie reviews to obtain the optimal results among all the classification techniques used. GitHub Link: `https://github.com/aturret/CS583-project`

I. Introduction

As a common filter for moviegoers, movie rating websites are widely used by movie fans. There exist many movie rating websites. For English speakers, IMDB is the most accessed movie rating website. Ratings could be crucial for the majority of people to make a decision, and reviews could be spoilers for people before watching the movie. However, reviews could be essential for moviegoers to make an insightful decision by considering their emotions and personal interests regarding whether to watch the movie or not.

For this project, we plan to utilize IMDB as a resource base for scraping movie reviews and other essential attributes. Specifically, we aim to extract useful information such as upvote and downvote numbers, movie genres, and ratings for each movie. This data can provide valuable insights into user opinions about movies and help with our sentiment analysis and classification process.

The main goal of this project is to use sentiment analysis techniques and machine learning algorithms to classify the reviews as helpful or not. To determine the usefulness of a review, we calculate its upvote ratio, which is the number of upvotes divided by the total number of votes. To classify a review as useful, we set a threshold based on the mean upvote ratio of all reviews. Any review with an upvote ratio above the mean is considered useful.

We scraped all of the reviews for the top 250 movies on IMDB(by June 2023). To closely analyze the sentiment factors in the reviews, feature extraction is a crucial routine to obtain information from the text data. We used the Latent Dirichlet Allocation technique to cluster the reviews and provide latent topics for the reviews.

Next, we counted the number of different types of words in the reviews, using the NLTK part-of-speech tagging technique to obtain counts of crucial vocabulary types such as nouns, adjectives, verbs, and adverbs. We then created a numerical data frame related to the reviews with manually extracted features. Additionally, we used tf-idf as our second feature extraction technique, and the results of tf-idf will be used as features for the classification process.

Finally, we moved on to using Convolutional Neural Networks (CNNs) for our analysis. CNNs are proven to be an effective and widely used technique for analyzing text data. Therefore, we decided to use CNNs as our final model to analyze the text data from the movie reviews in order to obtain the optimal results among all the classification techniques we used.

II. Literature Review

In Anindya and Panagiotis' paper[1],a consumer-oriented ranking that assesses the expected helpfulness of reviews, and a manufacturer-oriented ranking that evaluates their expected impact on sales. To develop these mechanisms, the authors combine econometric analysis with text mining techniques, specifically subjectivity analysis. Subjectivity analysis is a form of sentiment analysis that identifies the subjective nature of text and categorizes it as positive, negative, or neutral. The authors demonstrate the value of subjectivity analysis in providing insights into the helpfulness and impact of product reviews. By analyzing the subjectivity of reviews, they were able to identify those reviews that were most likely to be helpful to consumers and those that were most likely to influence sales.

In Vivek Singh's paper[2], author focuses on aspect-level sentiment analysis of movie reviews using a domain-specific feature-based heuristic. The authors propose an aspect-oriented scheme that analyzes textual reviews of a movie and assigns a sentiment label to each aspect. They use a SentiWordNet based scheme with two different linguistic feature selections and n-gram feature extraction. They compare their scheme with Alchemy API and document-level sentiment analysis. The results show that their approach produces a more

accurate and focused sentiment profile than document-level sentiment analysis.

In Samuel Onalaja, Eric Romero, and Bosang Yun's paper[3], their study focuses on comparing different classification models used for aspect-based sentiment analysis of movie reviews. The study uses five different machine and deep learning algorithms, including Logistic Regression, Naive Bayes, Support Vector Machine, and Recurrent Neural Network Long-Short-Term Memory, to predict binary sentiment of movie reviews based on genre and aspect-specific driving factors. The aspect words are aggregated from lexicon-based, supervised, and unsupervised learning methods, and their impact on the sentiment classification is investigated based on the accuracy of each model. The study found that assigning higher driving factors to certain aspects and genres results in higher accuracy of the sentiment prediction models used in the research. Overall, this study contributes to the understanding of the effectiveness of different machine learning algorithms for aspect-based sentiment analysis of movie reviews.

In H. M. Keerthi Kumar, B. S. Harish, and H. K. Darshan's paper[4], their study focuses on sentiment analysis of movie reviews on IMDb using hybrid features obtained by concatenating machine learning features with lexicon features. The study aims to distinguish between positive and negative reviews and improve the accuracy of sentiment classification. The researchers compared the performance of SVM, Naïve Bayes, KNN, and Maximum Entropy classifiers on the proposed model. The results showed that the hybrid feature model outperformed the individual machine learning and lexicon-based models, indicating the importance of capturing the context of the reviews in sentiment classification. The study highlights the significance of using hybrid features for sentiment analysis and improving the accuracy of classification.

According to Abinash Tripathy, Ankit Agrawal, and Santanu Kuma's paper[5], authors formed a study that compares the performance of Naive Bayes (NB) and Support Vector Machine (SVM) classification algorithms for sentiment analysis. The study uses a labeled dataset based on the polarity movie dataset and compares the results with existing literature. Sentiment analysis is described as a prominent branch of natural language processing that deals with text classification to determine the intention of the author, whether it is positive or negative. The focus of this paper is on the comparison of the two classification algorithms for binary classification of sentiment analysis.

Last but not least, Mais Yasen and Sara Tedmori's paper[6], the research employs various methods such as tokenization, stemming, feature selection, and classification to label reviews as positive or negative. The model is evaluated and compared using eight different classifiers and five evaluation metrics on a real-world dataset. The results reveal that the Random Forest classifier outperforms the other classifiers, while the Ripper Rule Learning classifier performed the worst. Overall, the review demonstrates the importance of sentiment analysis and the effectiveness of using different techniques to improve its accuracy.

III. Method

The research question posed in this study is: "How can we predict the helpfulness of a movie review using machine learning models and sentiment analysis?" The anticipated contribution of this research is the development of a robust and efficient routine to predict the helpfulness of a movie review with optimal results. Therefore, to predict the helpfulness, we can extract the feature of the review text and label the dataset by its helpfulness and then training our model by currently available deep learning models.

IV. Data

In this section, we will discuss the methods employed in this project, including how we collected the data, conducted preprocessing, and performed basic NLP analysis. For access to all of our code, including the scraping script, preprocessing scripts, and analysis scripts, please refer to the files that we have uploaded to Kaggle: https://www.kaggle.com/datasets/aturret/imdb-top250-reviews/

A. Data Collection

The dataset was obtained from IMDb.com. This study utilizes data from the famous IMDb Top 250 Movies. We have collected the metadata and all user reviews of these movies. Table.1 displays the information on the metadata that we have scraped, while Table.2 presents the information on the reviews that we have gathered.

Column	Meaning	Data Type	Example
movie_id	The IMDB id of the movie	String	tt0468569
movie_title	The title of the movie	String	Dark Knight
movie_rating	The average rating of the movie	Float	9.0
movie_rating_number	The number of ratings of the movie	Int	2687759
movie_genres	The genres of the movie	List	['Action','Crime', 'Drama']
movie_year	The year of the movie	Int	2008
movie_content_rating	The content rating of the movie	String	PG-13
movie_duration	The duration of the movie	String	2h 32m
review_number	The number of reviews of the movie	Int	8430

Table 1: Information of Movie Metadata

The primary data utilized in this project is the review dataset, which was scraped on Mar 16, 2023, resulting in a total of 340195 reviews. Due to the diverse nature of movie reviews, we will create a sample from the scraped data and perform analysis and modeling based on it. When we refer to the divergent nature of reviews, we mean that the sentimental factors in reviews do not have consistent meanings for each genre or even for each movie. As a result, making predictions based on a generalized text dataset would not be advisable.

Scraping

To obtain our primary data set, which is the review data set, we utilized two web scraping tools: Beautifulsoup4 and Selenium. We scraped a total of 340195 reviews by March 16, 2023, and saved the data in the CSV format. This approach allowed us to store the data locally and import it into a Pandas data frame conveniently. By having the data in a structured format, we can perform various exploratory data analysis techniques and prepare the data for modeling. We have the data set with structures shown in Table 2.

B. Preprocessing

In this section, we will perform some basic text preprocessing techniques, such as tokenization, lemmatization, and stemming, to generate new attributes for our data frame. Additionally, we will utilize various sentiment analysis techniques, which will enable us to extract valuable insights from our data. To carry out these tasks, we will utilize the Natural Language Toolkit (NLTK) library, which provides several functions and modules for NLP tasks. By applying these techniques, we can transform the raw text data into a more structured and meaningful form, which will facilitate the machine learning models' training and prediction processes. Furthermore, we will create a new target value called "helpful," which is the primary value that our model aims to predict and classify our text data into.

Column	Meaning	Data Type	Example
movie_id	The IMDB id of the movie	String	tt0050083
movie_title	The title of the movie	String	12 Angry Men
review_id	The average rating of the movie	String	rw3666418
review_author	The username of the reviewer	String	mark.waltz
review_title	The title of the review	String	One of the great theatrical examples of what makes for superb drama.
review_date	The date of the review	String	20 March 2017
review_rating	The rating of the review for the movie. If the user didn't give a rating, leave it as None	Int	10
review_text	The duration of the movie	String	Theater at its best is practically impossible to get down on film correctly. When Hollywood gets it right, they create a work of art. In this case, ...
review_upvote	The number of upvotes for "if this review is helpful"	Int	42
review_total	The total number of votes for "if this review is helpful"	Int	46

Table 2: Raw Data Information of Movie review

Creating Sample Data Set

Considering the nature of movies, the sentimental factors of movie reviews can exhibit divergence. For instance, a review containing the word "terrified" in a romantic movie may have a higher probability of being classified as negative, whereas the same review in a thriller or horror movie could be classified as positive. In this project, we aim to minimize such biases as much as possible. Therefore, we created a sample data set for our analysis and modeling. To obtain a sufficient number of instances, we were fortunate enough to find that the movie "Joker" has the largest number of reviews and is relatively recent, which helps eliminate some sentimental factors affected by time.

General Cleaning

Before performing any type of data analysis, we need a clean dataset. Therefore, we will drop the"review_id", "review_author", "review_date", and "movie_id" attributes since we are only analyzing the text data set under the movie "Joker". We will also count the number of null values in the data set. We found that the null values only exist in the "rating" attribute, which are "int64" data type entries. Thus, we will fill the "NAN" values with the mean of all ratings. Additionally, we want to create a "helpful" label for our prediction, which involves division. Therefore, we will remove instances with a total_vote of 0.

Removing Stop Words

To move forward, we need to preprocess the text attribute of each review. Our first step is to remove the stop words present in the text. Stop words are commonly used words in a language that do not contribute to the meaning or context of the text. Eliminating them can decrease data dimensionality, leading to faster and more efficient analysis. The Natural Language Toolkit (nltk) library provides a comprehensive stopword list that we can use to filter out stop words after tokenizing the text. This step is crucial in several natural language processing tasks, such as sentiment analysis, text classification, and topic modeling. By eliminating irrelevant words, we can focus on the more meaningful ones and improve analysis accuracy and efficiency.

Tokenization

Tokenization allows us to identify individual words or groups of words, such as phrases or named entities, which can be used as features in a model. By analyzing the frequency and distribution of these features across a large corpus of text, we can gain insights into patterns and trends in the language that may be useful for predicting or classifying new text data.

In our project, we will use tokenization to create a sequential representation of the text data in our movie review dataset. This will allow us to apply various natural language processing techniques, such as lemmatization, stemming, and sentiment analysis, to extract meaningful features from the text and train predictive models to classify the reviews based on their helpfulness.

Normalization

In order to transform words with different surface forms in the tokenized text into a more uniform representation, we will use two techniques: lemmatization and stemming. During our modelling, we will compare the impact and performance of each normalization process. For lemmatization, we will use the WorldNetLemmatizer, and for stemming, we will use the SnowballStemmer. This step is important for many natural language processing tasks, as it helps to reduce the dimensionality of the data and improve analysis accuracy and efficiency. (Example shown below)

```
Text Original: the movie affects you in a way that makes it physically painful to experience but in a good way
Text After Stemed: movi affect way make physic pain experi good way
Text After Lemmerization: movie affect way make physically painful experience good way
```

Figure 1: Normalization Result

Label Creation

To create the helpful label, we will first calculate the helpfulness for each reviews as

$$\text{helpfulness} = \frac{\text{upvotes}}{\text{totalvotes}}$$

, and we would like to look at the distribution of the each quartiles to determine the satisfied labels for target label helpful. We can observe that there are no outliers, and the median is around 0.5, lying between the first

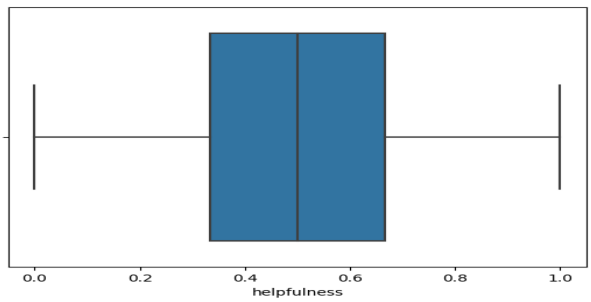


Figure 2: Boxplot for Helpfulness

and third quartiles. Therefore, we can use the mean helpfulness value as a threshold to identify helpful reviews. If the helpfulness score of a review is greater than the mean, it will be labeled as helpful is 1, and 0 otherwise. As result we obtain 6,666 reviews are not helpful and 4,401 reviews are helpful.

C. Exploratory Data Analysis on Meta Data

In this section, our aim is to perform an analysis of the data that we have scraped from the IMDB website. To begin with, we will conduct a correlation analysis based on the metadata that we have gathered from IMDB, which includes several general information about the movies as well as its numerical attributes. The primary goal of this analysis is to examine the relationships between various attributes and identify any potential patterns or trends that might exist in the data. This correlation analysis will help us gain a better understanding of the data and its characteristics, which will subsequently aid us in building our predictive models.

Correlation Analysis

We conducted a correlation analysis for the numeric data of the IMDb top 250 movies metadata. The results are presented in Table 5. The correlation analysis table displays the relationships among four variables: movie_rating, movie_rating_number, movie_year, and review_number.

It is worth noting that a strong positive correlation was observed between movie_rating_number and review_number (0.727), which suggests that movies with a higher number of ratings also tend to have more reviews. Additionally, movie_rating exhibited a moderate positive correlation with both movie_rating_number (0.602) and review_number (0.524), indicating that movies with higher ratings tend to have more ratings and reviews.

Regarding the relationship between movie_year and the other columns, movie_year had a weaker positive correlation with movie_rating_number (0.427) and review_number (0.359). This implies that more recent movies may have slightly more ratings and reviews. However, the correlation between movie_year and movie_rating was negligible (0.029), indicating that there is no meaningful relationship between the movie's release year and its rating.

	movie_rating	rating_number	movie_year	review_number
movie_rating	1.000000	0.602066	0.029436	0.524194
movie_rating_number	0.602066	1.000000	0.427648	0.727157
movie_year	0.029436	0.427648	1.000000	0.359601
review_number	0.524194	0.727157	0.359601	1.000000

Table 3: Correlation Analysis for Numeric Data of Metadata

Type of Docs Counting

First, we iterate through each review in the dataset and tokenize the text. Then, we use NLTK's pos_tag function to tag each token with its corresponding part of speech. We then count the number of tokens that are tagged as nouns, adjectives, and adverbs and store these counts as separate attributes for each review (as equation (1)). Additionally, we calculate the length of each review in terms of the number of characters and store this as another attribute. Finally, we use the NLTK library to count the number of sentences contained in each review and append this count as the last numerical attribute. This process allows us to extract more information from the text data and potentially improve the performance of our predictive models.

$$\begin{cases} \text{nouns} = \text{NN} + \text{NNS} + \text{NNP} + \text{NNPS} \\ \text{adjs} = \text{JJ} + \text{JJR} + \text{JJS} \\ \text{advs} = \text{RB} + \text{RBR} + \text{RBS} \end{cases} \tag{1}$$

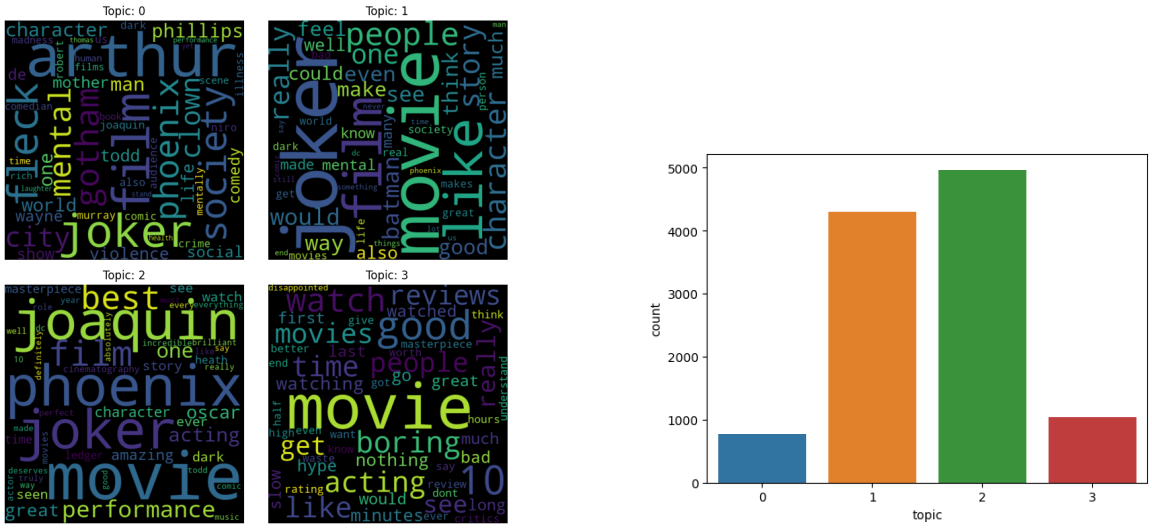
Thus, we obtain five numerical attributes for each reviews appended to the sample data set.

Topic Clustering using Latent Dirichlet Allocation

To extract more hidden features from the text, we use Latent Dirichlet Allocation (LDA) as a clustering tool. To identify the optimal number of topics, we implemented LDA with varying numbers of topics from 3 to 6 and evaluated them based on their top words and perplexity. We found that LDA with 4 topics had the best performance, and we labeled each review with its corresponding topic number.

LDA is a topic modeling technique that helps to identify the underlying topics in a set of documents. It works by representing each document as a mixture of topics, where a topic is a distribution over words. The optimal number of topics depends on the dataset and the research question, and there are various methods to estimate it, such as coherence score, perplexity, and visual inspection. In our case, we manually inspected the top words and perplexity to select the best number of topics. By labeling each review with its corresponding topic number, we can incorporate this additional feature into our analysis and improve the performance of our models. (Word Cloud Map shown in Figure 2.1a).

To analysis close and interpret each topic, we could name topic 0 as 'social', topic 1 as 'movie', topic 2 as 'actor' and topic 3 as 'negative'. And then we could see the distribution of reviews respect to topics as shown in Figure 2.1b.



(a) Topic Word Cloud

(b) Topic Distribution

Figure 3: Plots for LDA

Correlation Analysis on Numerical Features

We will analyze the numerical features obtained from the previous procedures. We observed that the topics obtained by the LDA method have low correlation with other features. Additionally, we noticed that the count numbers between the vocabulary and sentence are highly correlated, as expected.

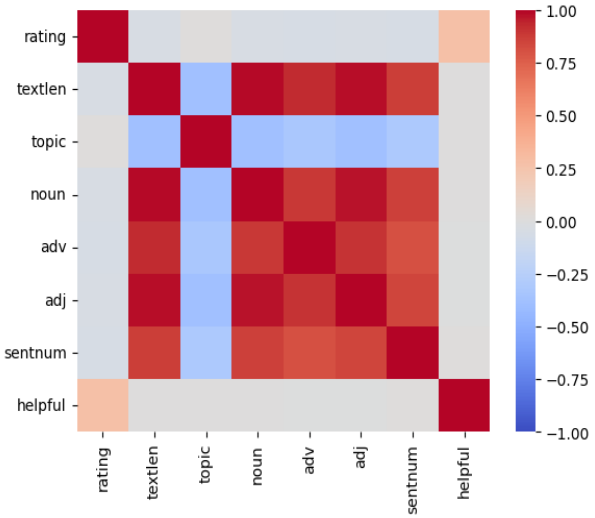


Figure 4: HeatMap for Numerical Features Generated

TF-IDF Vectorization

Term Frequency-Inverse Document Frequency (TF-IDF) is based on the Bag of Words (BoW) model, which provides insights into the relevance of words in a document. The importance of a word in the text is crucial in information retrieval. To convert the text data into a numerical vector representation of the terms, we adopt the TF-IDF vectorization algorithm, which works as follows:

$$\text{tf}(t,d) = \frac{count(t \in d)}{len(d)}, \quad \text{idf}(t,d) = log(\frac{|D|}{df(t)}), \quad \forall d \in D$$

$$\text{tf-idf}(t,d) = \text{tf}(t,d) \times \text{idf}(t,d)$$

And the following screen shot from our code shows the transformation by tf-idf:

```
Text After Tf-idf on Stemed Text:
  (0, 9520)     0.21991215830783573
  (0, 7850)     0.36377535527961674
  (0, 16524)    0.3690216011160821
  (0, 17245)    0.4231474051059488
  (0, 13549)    0.2171446029998366
  (0, 24730)    0.48717605999409014
  (0, 1039)     0.45325534448821775
  (0, 14756)    0.11892336032482567
```

Figure 5: TF-IDF Vectorized for Figure 1 Example

V. Tools

We use my laptop to run the models on my GPU, which is Nvidia 3070Ti. Here's the Python packages we are using for the project: We used nltk to tokenize our dataset. We used sklearn to apply SVM and logistics regression. For CNN classification part, we used tensorflow.

VI. Experiments

In this section, we will put into practice the machine learning models that we have studied over the course of this semester and assess their efficacy using two disparate feature extraction techniques that we explored in Section IV.D. Our primary objective is to determine the most efficient feature extraction method in conjunction with the classification model.

Our initial model will be logistic regression, which will serve as a benchmark for the other classification models we employ. We will only consider classification models that demonstrate higher accuracy rates than the baseline model while using diverse feature extraction techniques. We will utilize several well-established classification models, such as Naive Bayes, Support Vector Machines, Random Forest Modeling, and Convolutional Neural Networks, to determine the optimal method.

Each of these models has its own set of advantages and disadvantages, and they have been found to be effective in various situations. For instance, Naive Bayes is a probabilistic algorithm that is suitable for multi-class classification tasks. Support Vector Machines (SVMs) are particularly beneficial for handling high-dimensional datasets. Random Forests are well-suited for dealing with noisy or incomplete data, while Convolutional Neural Networks (CNNs) excel at extracting features from high-dimensional data such as text.

By comparing the performance of these classification models with the two feature extraction techniques, we aim to identify the most effective combination of model and feature extraction method that can be employed for our specific problem domain. This will help us to make better-informed decisions and enhance the overall performance of our machine learning models.

General Setting

Before proceeding with our modeling, we will split the data into training and testing sets. To accomplish this, we will use the test-train split tool from the Scikit-Learn package. Specifically, we will set the training set as 80% of the total data and the test set as 20% of the total data. We will also set the random state to 42 to ensure that the same split is obtained each time the code is run. Additionally, we will utilize hyperparameter tuning techniques for each model to identify the optimal parameters for each dataset. We use the GridSearchCV with the cross-validation number set to 10. The results of the hyperparameter tuning will be presented in the following section.

A. Logistic Regression

The logistic regression model is often considered the baseline model for classification problems. In this project, we will utilize the binomial logistic regression model since the target value is binary. The main idea behind logistic regression is to minimize the cost function as follows:

$$J(\theta) = -\frac{1}{m}\sum[y^{(i)}\log h_\theta(x(i)) + (1-y^{(i)})\log(1-h_\theta(x(i))]$$

via.Numerical Feature Data

With hyperparameter tuning technique, we find the best parameters for logistic regression model are 'C' = 1 and 'penalty' = 'l2'. And we obtained the classification report as shown:

	precision	recall	f1-score	support
0	0.60	0.92	0.73	1315
1	0.46	0.09	0.16	899
accuracy			0.59	2214
macro avg	0.53	0.51	0.44	2214
weighted avg	0.54	0.59	0.50	2214

Figure 6: Classification Report For LogReg + Num Features

via. TF-IDF Feature Extraction

With hyperparameter tuning technique, we find the best parameters for logistic regression model are 'C' = 0.01 and 'penalty' = 'l2'. And we obtained the classification report as shown:

	precision	recall	f1-score	support
0	0.59	1.00	0.75	1315
1	0.00	0.00	0.00	899
accuracy			0.59	2214
macro avg	0.30	0.50	0.37	2214
weighted avg	0.35	0.59	0.44	2214

Figure 7: Classification Report For LogReg + TFIDF Features

B. Naive Bayes Classification

The Naive Bayes classifier is a popular machine learning model that uses probability to classify instances. Its principle is based on the Bayes theorem, where the posterior probability of classes is calculated given the input features. In our case, we are dealing with a binary target value, where the class variable indicates whether something is helpful or not. For each instance, we aim to determine the class with the highest probability. And the calculation process as below:

$$\mathbb{P}[B_i \mid (A_1, A_2, \ldots, A_n)] = \frac{\mathbb{P}[(A_1, A_2, \ldots, A_n) \mid B_i]\mathbb{P}[B_i]}{\mathbb{P}[(A_1, A_2, \ldots, A_n)]}$$

via. Numerical Feature Data

With hyperparameter tuning technique, we find the best parameters for the Multinomial Naive Bayes model are 'alpha' = 10, 'class_prior' = None, 'fit_prior' = True. And we obtained the classification report as shown:

	precision	recall	f1-score	support
0	0.67	0.54	0.60	1315
1	0.48	0.61	0.54	899
accuracy			0.57	2214
macro avg	0.57	0.58	0.57	2214
weighted avg	0.59	0.57	0.57	2214

Figure 8: Classification Report For NB + Num Features

via. TF-IDF Feature Extraction

With hyperparameter tuning technique, we find the best parameters for the Multinomial Naive Bayes model are 'alpha' = 1, 'class_prior' = None, 'fit_prior' = True. And we obtained the classification report as shown:

	precision	recall	f1-score	support
0	0.59	1.00	0.74	1315
1	0.43	0.00	0.01	899
accuracy			0.59	2214
macro avg	0.51	0.50	0.38	2214
weighted avg	0.53	0.59	0.44	2214

Figure 9: Classification Report For NB + TFIDF Features

B. Random Forest Classification

The random forest algorithm is known for its ability to reduce overfitting and increase accuracy by combining the predictions of multiple decision trees. Each decision tree in the forest is trained on a random subset of the data and a random subset of the features, which helps to reduce the variance in the predictions.

During the training process, the algorithm grows each decision tree in the forest by recursively partitioning the data based on the features. At each node, the algorithm selects the feature that results in the best split, based on some impurity measure, such as Gini index or entropy. The process continues until the tree reaches a maximum depth or a minimum number of samples at each leaf.

via.Numerical Feature Data

With hyperparameter tuning technique, we find the best parameters for the Random Forest Classification Model are 'n_estimators' = 100, 'max_depth' = None, 'min_samples_split' = 5, 'min_samples_leaf' = 5.And we obtained the classification report as shown:

	precision	recall	f1-score	support
0	0.63	0.75	0.68	1315
1	0.48	0.34	0.40	899
accuracy			0.59	2214
macro avg	0.56	0.55	0.54	2214
weighted avg	0.57	0.59	0.57	2214

Figure 10: Classification Report For RF + Num Features

via.TF-IDF Feature Extraction

With hyperparameter tuning technique, we find the best parameters for the Random Forest Classification Model are 'min_samples_split' = 5, 'min_samples_leaf' = 5.And we obtained the classification report as shown:

	precision	recall	f1-score	support
0	0.60	0.98	0.74	1315
1	0.61	0.04	0.08	899
accuracy			0.60	2214
macro avg	0.61	0.51	0.41	2214
weighted avg	0.61	0.60	0.47	2214

Figure 11: Classification Report For RF + TFIDF Features

C. Supported Vector Machine Classification

The main aim of the Support Vector Machine (SVM) algorithm is to identify a hyperplane in an N-dimensional space, where N refers to the number of features, that can effectively classify the given data points. While attempting to separate the two classes of data points, there may be several potential hyperplanes to

choose from. The goal of SVM is to locate a plane that has the largest margin, or the maximum distance between the data points of both classes. This maximum margin is advantageous as it provides more support and allows future data points to be classified with greater certainty.In this project, we will use linear supported vector classifiers.

via.Numerical Feature Data

With hyperparameter tuning technique, we find the best parameters for the Supported Vector Machine Classification are 'C'= 1 and 'penalty' = 'l2'.And we obtained the classification report as shown:

	precision	recall	f1-score	support
0	0.60	0.97	0.74	1315
1	0.48	0.05	0.08	899
accuracy			0.59	2214
macro avg	0.54	0.51	0.41	2214
weighted avg	0.55	0.59	0.47	2214

Figure 12: Classification Report For LinearSVM + Num Features

via.TF-IDF Feature Extraction

With hyperparameter tuning technique, we find the best parameters for the Supported Vector Machine Classification are 'C'= 0.01 and 'penalty' = 'l2'.And we obtained the classification report as shown:

	precision	recall	f1-score	support
0	0.59	0.99	0.74	1315
1	0.18	0.00	0.00	899
accuracy			0.59	2214
macro avg	0.39	0.50	0.37	2214
weighted avg	0.43	0.59	0.44	2214

Figure 13: Classification Report For LinearSVM + TFIDF Features

D. Convolution Neural Network

Convolutional Neural Networks (CNNs) are a type of deep learning model that is widely used for various computer vision applications. The core of the CNN architecture is its convolutional layer, which applies a set of filters to the input image to extract meaningful features. However, the power of the CNN architecture is not limited to image processing. In recent years, researchers have successfully adapted CNNs for natural language processing (NLP) tasks by treating the input text as a two-dimensional matrix of word embeddings.

In this project, we design our convolution neural network as shown in Figure 14. Following the embedding layer, there is a convolutional layer with 128 filters. Each filter has a kernel size of 5, allowing the model to pick up patterns spanning 5 words at a time. The activation function used here is ReLU (Rectified Linear Unit), which introduces non-linearity to the model.

We obtain training accuracy: 0.6049 and texting accuracy: 0.5939.

E. Recurrent Neural Network

A recurrent neural network (RNN) is one of the two broad types of artificial neural network, characterized by direction of the flow of information between its layers.

We have those layers for the model settings: Embedding Layer: The RNN model also starts with an embedding layer similar to the CNN model, transforming input data into dense vectors. Simple RNN Layer: A single RNN layer with 128 units is used. The RNN layer processes the sequence of words in the input data by maintaining an internal state that captures information about the words it has seen so far. Output Layer: The final layer is a dense layer with a sigmoid activation function, suitable for binary classification. We obtain Training Accuracy: 0.6130 Testing Accuracy: 0.5962.

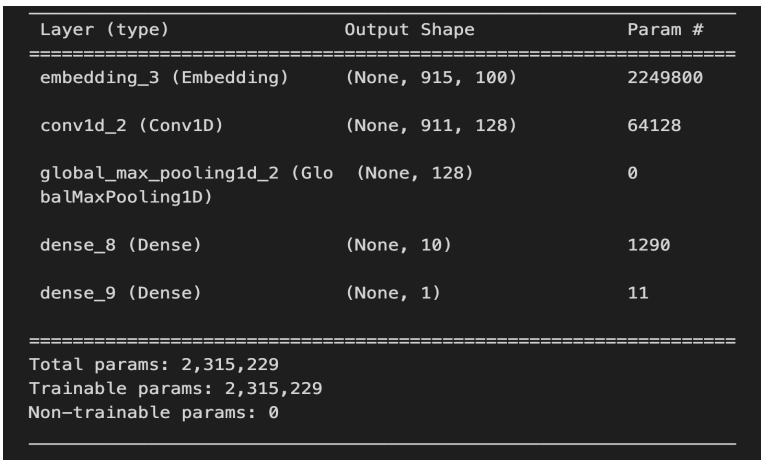


Figure 14: Summary For Convolution Neural Network

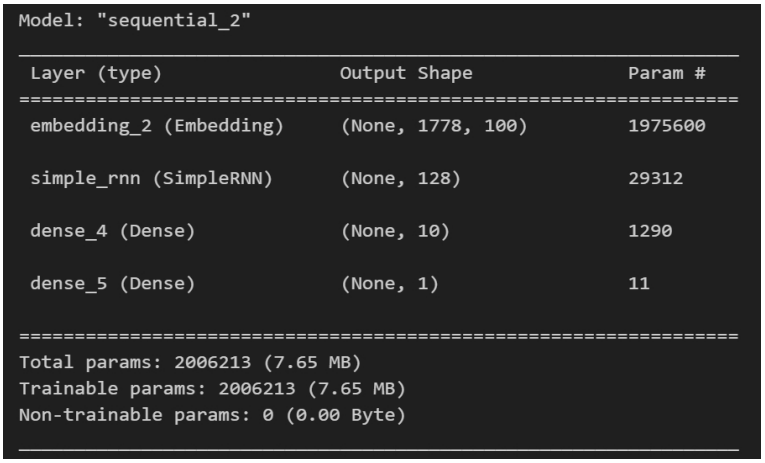


Figure 15: Summary For Recurrent Neural Network

F. Long Short-term Memory

Long short-term memory (LSTM) network is a recurrent neural network (RNN), aimed to deal with the vanishing gradient problem present in traditional RNNs. Its relative insensitivity to gap length is its advantage over other RNNs, hidden Markov models and other sequence learning methods.

For this model, we set the model as this: Embedding Layer: Like the previous models, the LSTM model begins with an embedding layer. LSTM Layer: It features an LSTM layer with 128 units. LSTMs are a special kind of RNN capable of learning long-term dependencies. They are particularly effective for sequence prediction problems because they can remember information for long periods, which is a key advantage over simple RNNs. Output Layer: The model concludes with a dense layer with a sigmoid activation function for binary classification.

We obtain Training Accuracy: 0.6130 Testing Accuracy: 0.5962.

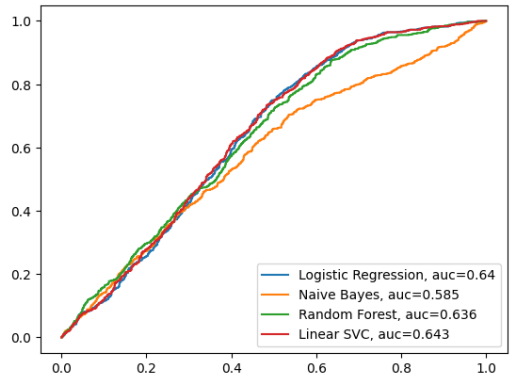
VII. Conclusion

In Section VI, we fitted 4 classification models combined with 2 feature extraction techniques and were able to generate 8 classification tables that provide us with a comprehensive view of our models' performance. We then visualized our results on two ROC Curve plots, allowing us to generalize our findings and gain a better understanding of the models' performance across different thresholds.

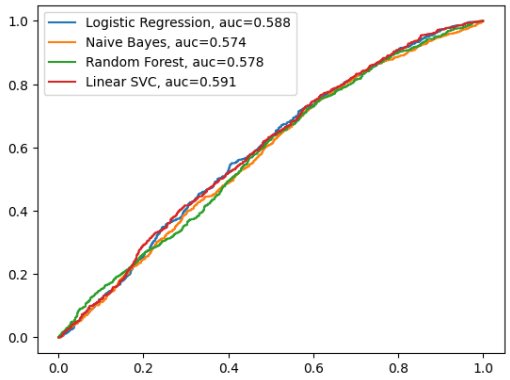
Upon analyzing the AUC scores, we observed that the Linear SVC model combined with manually extracted features achieved the highest score of 0.643. Meanwhile, the Random Forest Classifier combined with TFIDF feature extraction demonstrated the highest accuracy score of 0.60, according to the classification table.

The result is almost the same as the result of RNN. This might be a result of overfitting. In future research. We might be able to improve this part by simply the models and change the hyperparameters. However, we noticed that the prediction scores were relatively lower in class 1, indicating that our models performed better in predicting non-helpful reviews. To investigate further, we may need to consider several factors, including the

quality and quantity of data, the choice of features, and the classification algorithms' suitability for the task. It is possible that our models were biased towards not helpful reviews due to imbalanced data, and more data in class 1 may help improve the models' performance. Additionally, we may need to explore alternative feature extraction techniques and classification algorithms that are better suited for this task.



(a) ROC Curve for Models on Numerical Features



(b) ROC Curve for Models on TFIDF Features

Figure 16: Plots for ROC Curves

References

[1] Anindya Ghose and Panagiotis G. Ipeirotis. "Designing Novel Review Ranking Systems: Predicting the Usefulness and Impact of Reviews". In: *Proceedings of the Ninth International Conference on Electronic Commerce*. ICEC '07. Minneapolis, MN, USA: Association for Computing Machinery, 2007, pp. 303–310. ISBN: 9781595937001. DOI: 10.1145/1282100.1282158. URL: https://doi.org/10.1145/1282100.1282158.

[2] Vivek Singh et al. "Sentiment analysis of movie reviews: A new feature-based heuristic for aspect-level sentiment classification". In: Mar. 2013, pp. 712–717. ISBN: 978-1-4673-5089-1. DOI: 10.1109/iMac4s.2013.6526500.

[3] Samuel Onalaja, Eric Romero, and Bosang Yun. "Aspect-based Sentiment Analysis of Movie Reviews". In: *SMU Data Science Review* 5.3 (2021), Article 10. URL: https://scholar.smu.edu/datasciencereview/vol5/iss3/10.

[4] H. M. Keerthi Kumar, B. S. Harish, and H. K. Darshan. "Sentiment Analysis on IMDb Movie Reviews Using Hybrid Feature Extraction Method". In: *Int. J. Interact. Multim. Artif. Intell.* 5 (2019), pp. 109–114.

[5] Abinash Tripathy, Ankit Agrawal, and Santanu Kumar Rath. "Classification of Sentimental Reviews Using Machine Learning Techniques". In: *Procedia Computer Science* 57 (2015). 3rd International Conference on Recent Trends in Computing 2015 (ICRTC-2015), pp. 821–829. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2015.07.523. URL: https://www.sciencedirect.com/science/article/pii/S1877050915020529.

[6] Mais Yasen and Sara Tedmori. "Movies Reviews Sentiment Analysis and Classification". In: *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*. 2019, pp. 860–865. DOI: 10.1109/JEEIT.2019.8717422.