

Natural Language Processing (NLP) for Amazon Food Reviews

Project Report

Natural Language Processing (NLP) for Amazon Food Reviews

Project objectives

In this technologically advanced world, businesses are expanding their market online. E-commerce is taking advantage of this opportunity by providing customers with products in the comfort of their own home. Online reviews are important for customers for a better understanding of the product and their purchasing decisions. Reviews can also help retailers to improve their service or product. This project aims to use NLP techniques to build prediction models to predict how helpful new reviews will be or gain insight into the features that influence helpfulness. To better understand the variety of NLP techniques, we plan to test and apply multiple NLP techniques for this assignment.

Data Exploration

Data Dictionary

Field Name	Data Type	Description
Id	int64	Id number for user
ProductId	object	Unique identifier for the product
UserId	object	Unique identifier for the user
ProfileName	object	Profile name of the user
HelpfulnessNumerator	int64	Number of users who found the review helpful
HelpfulnessDenominator	int64	Number of users who indicated whether they found the review helpful or not
Score	int64	Rating between 1 and 5
Time	int64	Timestamp for the review
Summary	object	Brief summary of the review
Text	object	Text of the review

Reason to choose the dataset

We have chosen this dataset, as the dataset contains an enriched amounts of text data (568454) with no missing values or typographical errors. This allowed the team to focus on gathering insights from the data and exploring more interesting modelling techniques, rather than spending significant time collecting data or dealing with the legalities of web-scraping or API usage. Furthermore, it allowed for a wide variety of NLP techniques can be considered, in terms of pre-processing, language analysis and building models.

Finally, the predicted outcomes would provide potential insights into Amazon foods industry and providing recommendations. By focusing on the text contents of the reviews as input into models, users can benefit from seeing predictions on the helpfulness of reviews before posting. Meanwhile companies like Amazon can bring more visibility to helpful reviews that are too new to gather many human votes, from new users that have yet to build a significant track record, or for products that are niche. This potentially provides another avenue for overcoming the cold-start problem.

Data EDA

DataFrame

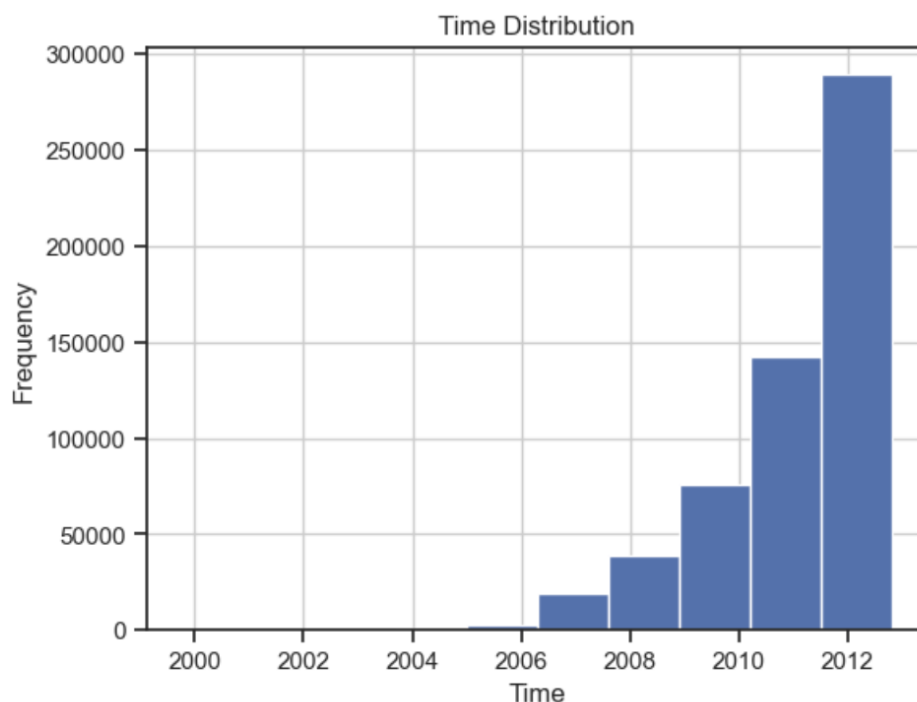
There were 568454 rows \times 10 columns in the Amazon Fine Food Reviews dataset, each containing id, product id, user id, profile name, helpfulness numerator, helpfulness denominator, score, time, summary and review text. There were no missing values or typographical errors in the dataset. We mainly performed data EDA to analyse any correlations between variables, frequency of words and cleaned texts using various techniques, including stopwords, lemmatization, tokenisation, vectorisation and punctuation. It was decided data that we will be using most in for this analysis is 'HelpfulnessNumerator', 'HelpfulnessDenominator' and 'Text'. Target variable is 'Helpfulness' which is calculated as the 'HelpfulnessNumerator' divided by 'HelpfulnessDenominator' to provide a fairer comparison of reviews with varying amounts of votes. As a result of removing records that do not have a value for 'Helpfulness', the usable data set is reduced to 298k reviews.

	Id	Productid	Userid	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	1303862400	Good Quality Dog Food	I have bought several of the Vitality canned d...
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0	1	1346976000	Not as Advertised	Product arrived labeled as Jumbo Salted Peanut...
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1	4	1219017600	"Delight" says it all	This is a confection that has been around a fe...
3	4	B000UA0QIQ	A395BORC6FGVXV	Karl	3	3	2	1307923200	Cough Medicine	If you are looking for the secret ingredient i...
4	5	B006K2Z27K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	0	0	5	1350777600	Great taffy	Great taffy at a great price. There was a wid...
5	6	B006K2Z27K	ADT0SRK1MG0EU	Twoapennything	0	0	4	1342051200	Nice Taffy	I got a wild hair for taffy and ordered this f...
6	7	B006K2Z27K	A1SP2KVFXXRUI	David C. Sullivan	0	0	5	1340150400	Great! Just as good as the expensive brands!	This saltwater taffy had great flavors and was...
7	8	B006K2Z27K	A3JRGQVEQN31IQ	Pamela G. Williams	0	0	5	1336003200	Wonderful, tasty taffy	This taffy is so good. It is very soft and ch...
8	9	B000E7L2R4	A1MZY09TZK0BBI	R. James	1	1	5	1322006400	Yay Barley	Right now I'm mostly just sprouting this so my...
9	10	B00171APVA	A21BT40VZCCYT4	Carol A. Reed	0	0	5	1351209600	Healthy Dog Food	This is a very healthy dog food. Good for thei...

Figure 1. Preview Amazon Fine Food Reviews Dataset

Time Distribution

The reviews ranged from 1999-2012. There are more records in 2012 compared to previous years. When building models it should be considered that predictions are centred on newer reviews from 2012.



Positive WordCloud

Sentiment analysis was performed on the dataset and classified the reviews into positive, negative and neutral classes. The reviews were filtered from the dataframe to generate the wordclouds. In the positive wordcloud, words in the positive reviews can be observed are 'really', 'good' and 'recommend' which means most reviews in the dataset express a positive sentiment about the product.



Some words in the negative review can be observed are 'bad', 'little' and 'problem' which means most reviews in the dataset express a negative sentiment about product. There were some words such as 'good', 'product', 'taste', 'flavor' and 'coffee', which are also present in the negative wordcloud despite being positive words. This is probably used in a negative context such as 'not good' or 'bad taste'.

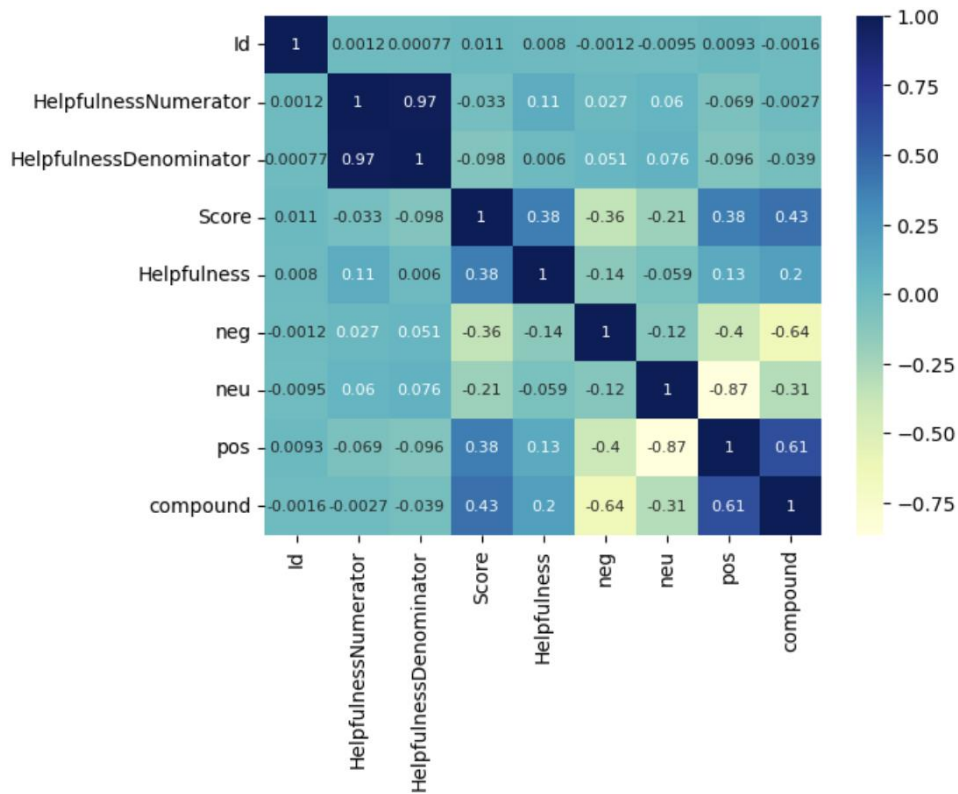


Figure 5. Correlation Heatmap

Helpfulness rating was a continuous variable and linear regression model was fitted into the dataset. However, with further investigation into the data, we have found that the helpfulness was better suited to a classification model to predict probability of helpful or not helpful. Since there were more counts of positive Helpfulness reviews which lead to an imbalanced dataset, classification models were chosen as it was more suitable with handling this issue.

Most of the reviews with helpfulness ratings have a helpfulness rating of 1, and there are many reviews with less than ten votes, so the reviews with a helpfulness rating of 1 and less than ten votes may not reflect the opinions of general users in a fair and universal manner.

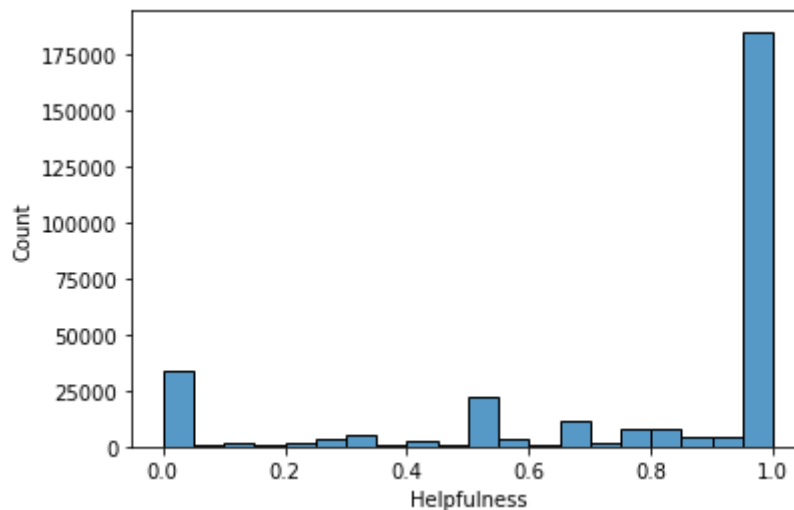


Figure 6. Distribution of Helpfulness Score

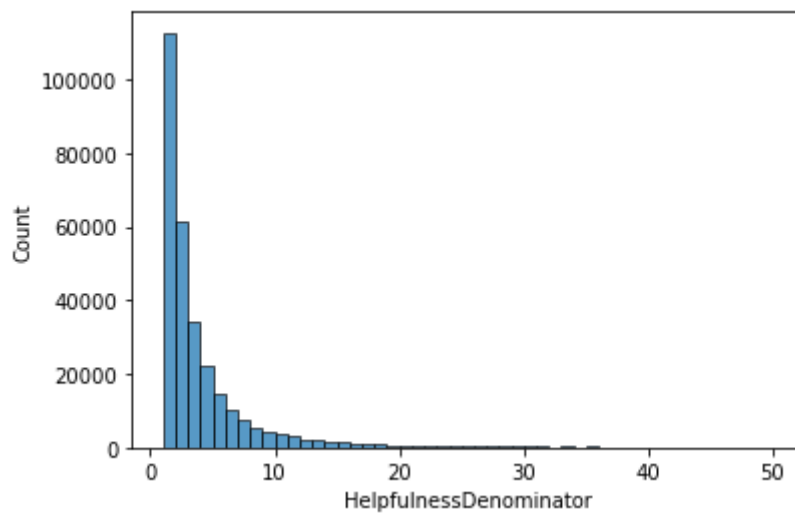


Figure 7. Distribution of HelpfulnessDenominator

To ensure our model only flags reviews that are considered helpful by most users, we have amended the definition of "helpful review" to mark reviews that have been rated at least 10 times and have a helpfulness score greater than 0.8. The red box in the figure below shows out target variable '1' as Helpful and '0' being not helpful.

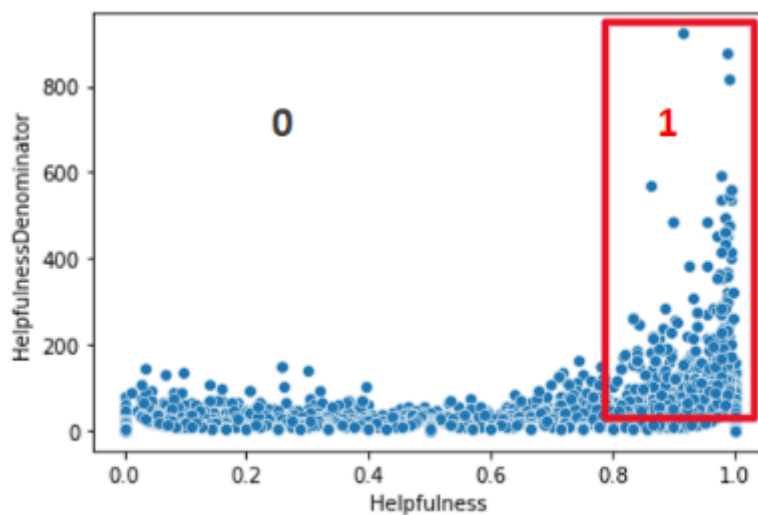


Figure 8. Scatterplot for Helpfulness vs HelpfulnessDenominator

Data Cleaning

The following text data preprocessing steps were performed using The Natural Language Toolkit (NLTK) python package.

Lowercasing:

NLP models are case sensitive. Word such as 'Great' and 'great' are the same however treated differently when they are not converted to lowercase and represented as two different words in the model.

Stopwords removal:

Stop-words are the most common words in the English language such as 'I' this' and 'in' which do not add much value to the meaning to the corpus. NLTK provides a list of commonly agreed stopwords. They are removed to decrease dataset size and focus on meaningful words.

Lemmatization:

NLTK Lemmatization is a text processing technique that breaks a word down to its root meaning to identify similarities. It considers the context and converts text to its base form. For example: 'Caring' would return 'Care'. `nltk.corpus.wordnet` was used because lemmatization requires speech of tag value of the word in the sentence.

Tokenisation:

NLTK provides a function that splits strings into tokens based on whitespace and punctuations. The output of the word tokenisation is converted to data frame for better text understanding for machine learning applications.

Special Characters:

Tokens that did not add deeper meaning in the text were filtered out such as punctuation (e.g. '...', '#') and website links. This was done by creating a list of tokens to filter and iterating over all the tokens to keep tokens that add value to the text.

Time:

Since the variable 'Time' in the original dataset is represented as the form of a Unix timestamp, it was converted into the Unix form Year-Month-Day. This column was later dropped as it was not valuable in our model analysis.

NLP Techniques

Topic Modelling

Topic Modelling was performed by leveraging the BERT (Bidirectional Encoder Representations from Transformers) language model developed by Google (Devlin, et al 2018). The advantage of a pretrained model is that it is developed on much larger datasets and with a lot more compute power than we can access. However, given that it is a general-purpose model, it may not necessarily help with predictions for our more niche use cases.

The Python package BERTopic employs the following steps to assist with topic modelling: Embeddings via a sentence-transformer, dimensionality reduction using UMAP (Uniform Manifold Approximation and Projection), hierarchical clustering with HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise), term relevancy with c-TF-IDF, and finally Keyword extraction via KeyBERT.

This process was able to identify 550 topics amongst the dataset, as well as measure the distance based on similarity between these topics as seen below.



Figure 9. Distance Map, sized by frequency

A significant portion (approximately 38.6%) of the dataset could not find a topic with high confidence as indicated a Topic of “-1” in Figure 10 which ranks topics by frequency across reviews. These reviews had middling helpfulness scores, with the highest probability keywords being phrases like taste and good, indicating their lack of detailed descriptions are a likely cause for the lack of usefulness.

Topic	Helpfulness	Count	Name
-1	0.650674	9650	-1_taste_product_sugar_good
0	0.701365	586	0_oil_olive_hair_coconut
1	0.718182	220	1_charsiu_scotch_zevia_coke
2	0.849515	206	2_plant_tree_grow_kit
3	0.331606	193	3_science_percent_hungry_deliberately
4	0.811518	191	4_noodle_soup_add_sauce
5	0.754839	155	5_cookie_kettle_chip_mother
6	0.809917	121	6_grinder_fillet_norte_amoretti
7	0.551724	116	7_box_seal_greenie_cardboard
8	0.732759	116	8_tea_green_celestial_iced
9	0.536364	110	9_film_movie_jim_alec
10	0.740000	100	10_organic_canola_flour_granola

Figure 10. Topics ranked by Frequency

When ranking by helpfulness scores, amongst the keywords for highly helpful reviews are a mix of descriptive adjectives (e.g., “palatable”, “wasteful”, “anxious”) and non-salient product features (e.g., “mass-produced”, “palette”) in Figure 11. By contrast, in the previous Figure 10 there are frequent product features mentioned but these are more obvious terms that are unlikely to give a reader more information than a simple product description (e.g., “plant”, “noodle”, “cookie”, etc.).

Topic	Helpfulness	Count	Name
157	1.0	27	157_crackerscookie_dismember_children_backpack
160	1.0	26	160_palette_palatable_track_yum
167	1.0	26	167_believeri_anecdotal_wasteful_midwife
174	1.0	26	174_exhausted_anxious_flash_sleep
178	1.0	26	178_massproduced_highprotein_nitrite_lowfat
192	1.0	25	192_flashessage_relieve_wow_possible
185	1.0	25	185_dr_chosenettle_growthi_teaaivita
184	1.0	25	184_midwife_labor_twoat_teabut
204	1.0	24	204_marine_desiccant_afghanistan_snack
247	1.0	21	247_pink_salt_creamysalty_dishalso

Figure 11. Topics ranked by Helpfulness (high)

Amongst the least helpful reviews were much more exaggerated language such as “roadkill”, “godawful”, “idiot” as seen in Figure 12. This is a likely indication that extreme polarity in reviews is a deterrent to usefulness. This could be either because the reviewers seem biased, or this language does not convey any descriptive details about the product. While the polarity in these topics is skewed towards the negative, it is worth noting that extreme positivity did not show up in the most useful topics.

Topic	Helpfulness	Count	Name
340	0.0	17	340_meatnot_unnamed_roadkill_euthanize
348	0.0	16	348_terriyaki_jack_subscriber_links
365	0.0	15	365_godawful_slap_appetite_edible
392	0.0	15	392_michaels_ton_represent_sale
407	0.0	14	407_delicately_nearby_munch_buttery
443	0.0	13	443_wasabi_appealing_glow_overwhelming
456	0.0	13	456_idiot_cancel_ignore_request
485	0.0	12	485_sweeten_splenda_sample_soda
512	0.0	11	512_srewe_covering_ring_safe
536	0.0	11	536_indulge_reward_jim_training

Figure 12. Topics Ranked by Helpfulness (low)

The distance map from BERTopic allows for topics to be clustered based on their similarity as seen on the left side of Figure 13 below. The chart on the right by contrast shows the average helpfulness of these topics. While there does not appear to be a correlation between various clusters and the helpfulness score, the right-side chart does show that certain topics are more likely to be helpful

than others. As such, individual topic labels are likely to be a more suitable input into predictive models than the topic clusters.

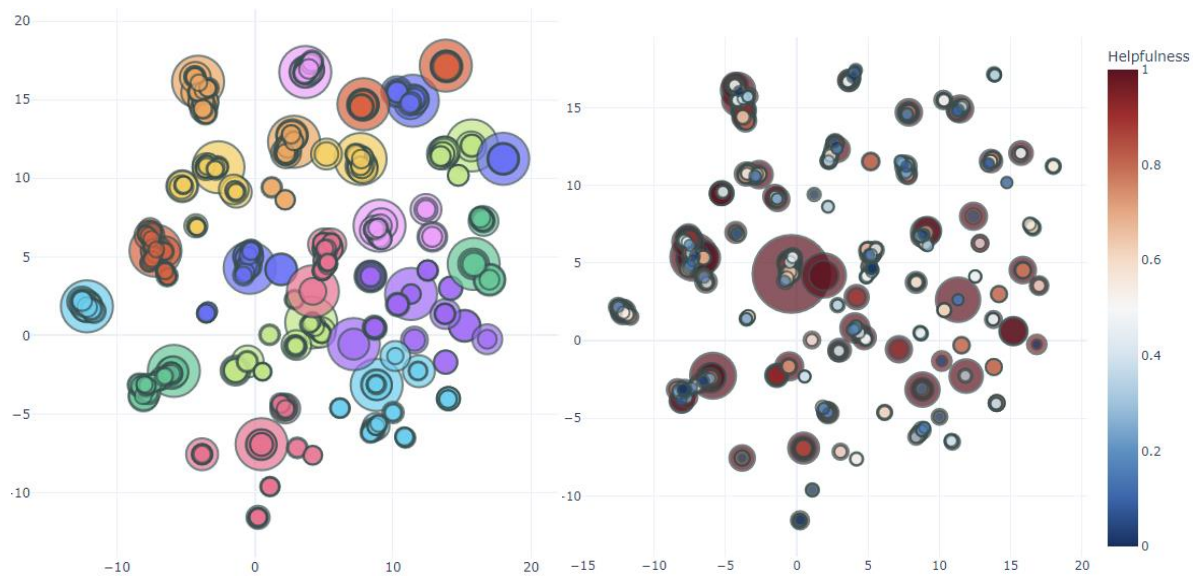


Figure 13. Left: Topic Clustering, Right: Topic Helpfulness

K-mean Clustering

Clustering was performed via K-mean clustering, which is an unsupervised clustering algorithm and allows to group data points with similar patterns into several clusters. Before clustering, texts were first cleaned using 'TfidfVectorizer' and then cleaned words were fit into K-means, splitting into three clusters. New column 'Cluster_label' was also created to sort three different clusters and then split into three different columns representing each cluster. First cluster was mainly associated with tea businesses. Second cluster was mainly related to reviews about coffee businesses. Lastly, third cluster has the most data, which were associated with reviews about food businesses.

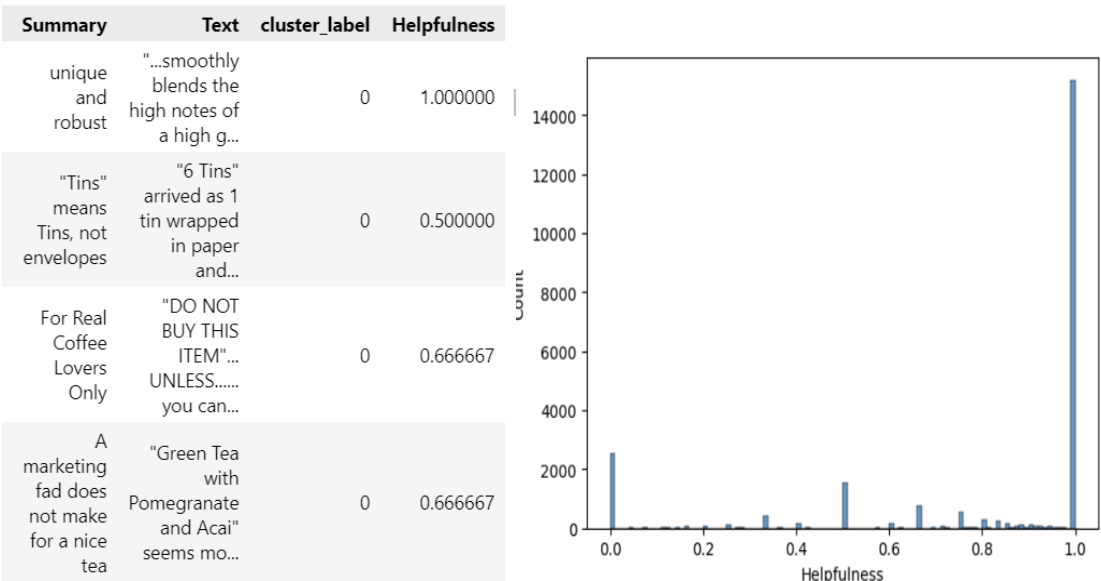


Figure 14. Cluster 0 and its Distribution of Helpfulness

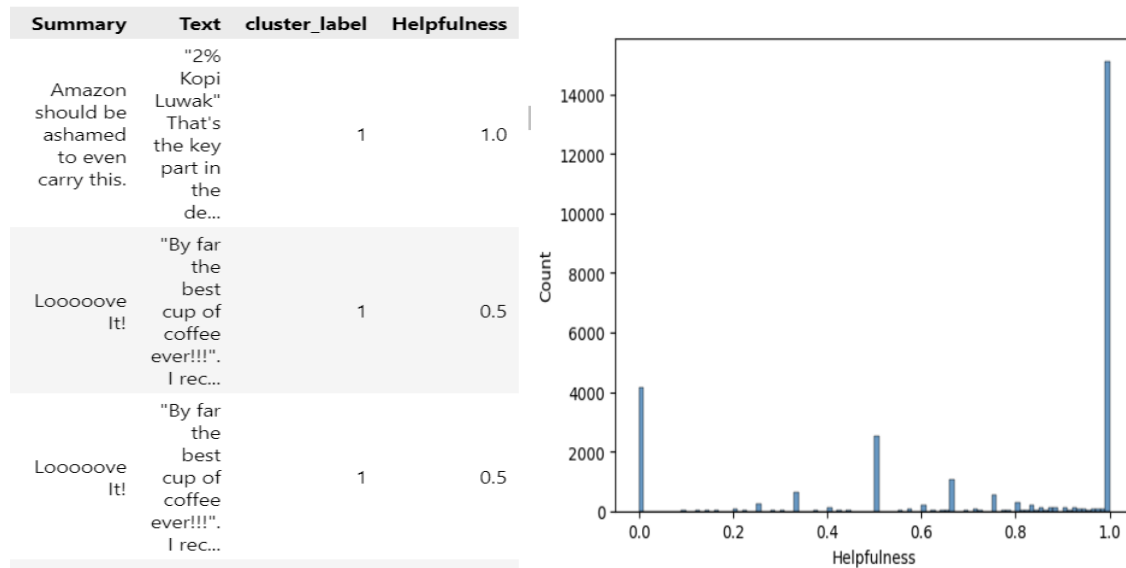


Figure 15. Cluster 1 and its Distribution of Helpfulness

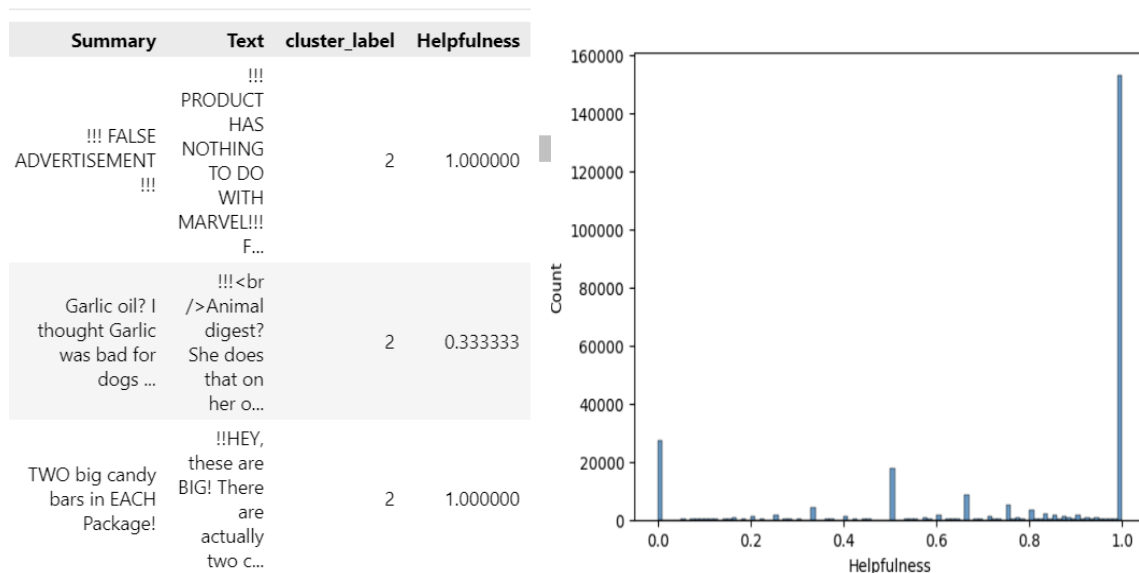


Figure 16. Cluster 2 and its Distribution of Helpfulness

Subsequently, we have tried to only include reviews, which have high helpfulness scores in each cluster. Therefore, we filter the dataset further to only include the reviews that have helpfulness scores between 0.8 and 1.0 and helpfulness denominator scores higher than 10.0 and removed the rest of reviews with relatively poor helpfulness scores.

	index	Id	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	cluster_label	Helpfulness	kmeans
count	1049.000000	1049.000000	1049.000000	1049.000000	1049.000000	1.049000e+03	1049.0	1049.000000	1049.0
mean	288638.937083	288639.937083	18.709247	19.741659	4.453765	1.233807e+09	0.0	0.943885	0.0
std	166293.140317	166293.140317	28.841835	29.412650	1.094751	5.838618e+07	0.0	0.059640	0.0
min	804.000000	805.000000	8.000000	10.000000	1.000000	1.038010e+09	0.0	0.800000	0.0
25%	128596.000000	128597.000000	10.000000	11.000000	4.000000	1.179187e+09	0.0	0.909091	0.0
50%	299499.000000	299500.000000	13.000000	14.000000	5.000000	1.234656e+09	0.0	0.947368	0.0
75%	444655.000000	444656.000000	18.000000	20.000000	5.000000	1.282954e+09	0.0	1.000000	0.0
max	566650.000000	566651.000000	580.000000	593.000000	5.000000	1.350605e+09	0.0	1.000000	0.0

Figure 17. Statistic Summary of Helpful Reviews about Tea Businesses

	index	Id	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	cluster_label	Helpfulness	kmeans
count	1488.000000	1488.000000	1488.000000	1488.000000	1488.000000	1.488000e+03	1488.0	1488.000000	1488.0
mean	297415.430108	297416.430108	39.102823	40.690188	4.389113	1.260747e+09	1.0	0.946598	1.0
std	165690.099648	165690.099648	74.554518	75.546670	1.157910	5.581520e+07	0.0	0.057833	0.0
min	2380.000000	2381.000000	8.000000	10.000000	1.000000	1.036109e+09	1.0	0.800000	1.0
25%	148444.750000	148445.750000	12.000000	12.000000	4.000000	1.232431e+09	1.0	0.910714	1.0
50%	293141.500000	293142.500000	17.000000	19.000000	5.000000	1.270426e+09	1.0	0.964286	1.0
75%	453389.500000	453390.500000	33.000000	35.000000	5.000000	1.302134e+09	1.0	1.000000	1.0
max	568124.000000	568125.000000	559.000000	562.000000	5.000000	1.347408e+09	1.0	1.000000	1.0

Figure 18. Statistic Summary of Helpful Reviews about Coffee Businesses

	index	Id	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	cluster_label	Helpfulness	kmeans
count	13795.000000	13795.000000	13795.000000	13795.000000	13795.000000	1.379500e+04	13795.0	13795.000000	13795.0
mean	285672.972309	285673.972309	22.284161	23.680029	4.088003	1.248126e+09	2.0	0.941656	2.0
std	161711.039962	161711.039962	30.787878	32.566613	1.452879	5.687737e+07	0.0	0.062654	0.0
min	32.000000	33.000000	8.000000	10.000000	1.000000	9.482400e+08	2.0	0.800000	2.0
25%	152678.500000	152679.500000	11.000000	11.000000	4.000000	1.206662e+09	2.0	0.900000	2.0
50%	285494.000000	285495.000000	14.000000	15.000000	5.000000	1.256602e+09	2.0	0.953846	2.0
75%	424475.500000	424476.500000	22.000000	23.000000	5.000000	1.294099e+09	2.0	1.000000	2.0
max	568436.000000	568437.000000	866.000000	923.000000	5.000000	1.349827e+09	2.0	1.000000	2.0

Figure 19. Statistic Summary of Helpful Reviews about Food Businesses

Sentiment analysis

Sentiment analysis, also known as opinion mining, has been increasingly being used to describe emotions in a text. It uses computational means to classify positive, neutral and negative attitudes to a subject. In e-commerce, sentiment analysis is being used to analyse product reviews to investigate topics such as consumer rating of products and services (e.g., food, books and movies).

Sentiment analysis was performed using Valence Aware Dictionary for Sentiment Reasoning (VADER), an NLTK module that provides sentiment scores based on words used created by C.J Hutto and Eric Gilbert. It is a general purpose sentiment analysis tool that is popularly used in many studies.

VADER sentiment analysis relies on a dictionary which maps lexical features to emotion intensities called sentiment scores. The dictionary was constructed by the creators of module and enlisted a number of human raters and averages their ratings on each word. This relies on the concept on human wisdom and collective opinion.

The VADER module produces four sentiment measurements from each word grading, as seen in the figure below. The model was graded has three measures 'neg', 'neu' and 'pos' and the content falls into each classification.

Scoring

The fourth measurement is compound score. The compound score is computed by summing the valence scores of each word in the lexicon, adjusted according to the rules and normalized to be between -1 (negative) and 1 (positive).

It has been decided by researchers and authors (Hutto et. Al 2014) that the typical threshold values are:

1. Positive sentiment: compound score ≥ 0.05
2. Neutral sentiment: compound score ≥ -0.05 and compound score < 0.05
3. Negative sentiment: compound score ≤ -0.05

id	Productid	Userid	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Text	text_token	text_string	text_string_fdist	text_string_lem	Helpfulness	polarity	neg	neu	pos	compound
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	2011-04-27	i have bought several of the vitality canned d...	[bought, several, vitality, canned, dog, food, ...]	bought several vitality canned dog food produc...	bought several vitality canned dog food produc...	1.0	('neg': 0.0, 'neu': 0.503, 'pos': 0.497, 'comp...	0.000	0.503	0.497	0.9413
1	2	B00813GRG4	A1D87F6ZCVE5NK	dil pa	0	0	1	2012-09-07	product arrived labeled as jumbo salted peanut...	[product, arrived, labeled, jumbo, salted, pea...	product arrived labeled jumbo salted peanuts p...	product arrived labeled jumbo salted peanuts p...	NaN	('neg': 0.129, 'neu': 0.762, 'pos': 0.11, 'com...	0.129	0.762	0.110	-0.1027
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1	4	2008-08-18	this is a confection that has been around a fe...	[confection, around, centuries, light, pillowy...	confection around centuries light pillowy citr...	confection around centuries light pillowy citr...	1.0	('neg': 0.171, 'neu': 0.544, 'pos': 0.285, 'co...	0.171	0.544	0.285	0.8073

Figure 20. Summary of VADER sentiment measurements

Most reviews have positive sentiment, indicating majority of users likely post reviews when they happy about their product.

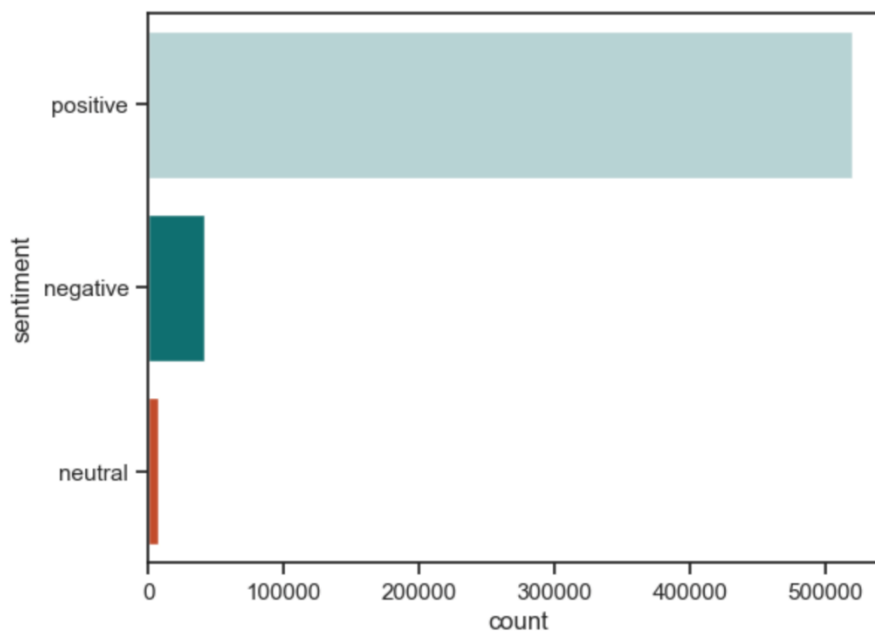


Figure 21. Visualisation sentiment analysis

Feature Engineering

Count of Sentences and Words:

The number of sentences and words in a review can indicate the complexity of the review. Reviews with more sentences and a higher word count are considered more complex and contain more information. Nltk package is used to tokenize the review into words and sentences, then a loop is built to count the words and sentences for each review. By including this feature in the model, the model can account for the complexity of the review.

Polarity and Subjectivity:

Polarity and Subjectivity are generated for each review using the TextBlob package. Polarity identifies whether a review is positive, negative or neutral, while subjectivity indicates whether it is based on personal opinion or objective. By including both features, the model can predict based on both sentiment and subjectivity of the review.

Count of punctuation:

Periods and commas can affect the readability and complexity of a review, while exclamation points and question marks may indicate strong emotions. Adding these features would enable the model to consider the readability and emotional aspects of the text.

Readability:

The Textstatistic package is used to calculate the readability score for each review. We use a Flesching reading ease score ranging from 0 to 100 to assess how easy the text is to read. The higher the score, the easier the text is to read. A helpful review should be relatively easy to understand, and we include this feature in the model because we believe it might impact the review's usefulness. (Pamela O. 2021)

Count of Named Entities:

Using Named Entity Recognition (NER), the process of identifying text which mentions specific items such as names of persons, locations, companies, products, percentages, monetary values, etc. These can provide useful information or context to reviews that may complement word frequencies. Even with the removal of stop words, commonly appearing words may not always convey the most information to readers.

Topic Labels:

In a similar vein to NER, topics are a way of capturing important information that is not necessarily the highest occurring words. By contrast, while NER may detect interesting phrases within a review, topic modelling seeks to condense the overall text into keywords and can therefore provide different predictive power to NER. As seen in the previous topic modelling section, the average helpfulness scores can vary quite significantly between different topics.

Cluster Labels:

Cluster labels, that consists of three different clusters representing tea, coffee and foods in each cluster, were created using K-mean clustering technique. Then, we cleaned data in each cluster, by only including reviews that have 'Helpfulness' scores between 0.8 and 1.0. Three clusters were merged, in order to be used as one feature of models.

Embeddings/Vectorisation

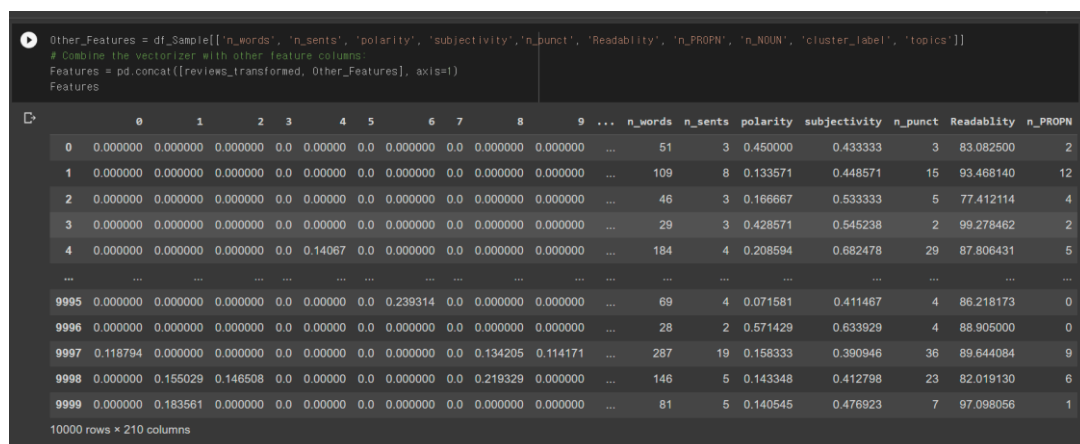
Word embeddings are a numerical representation of a word which encodes its meaning and context within a vectorised space. Words with similar meanings or usage will have similar vector representations. For the supervised classification and regression models, text is transformed into numerical features using 'TfidfVectorizer' technique from the Sklearn package. We chose this Vectorizer because it assigns weights to words not only based on their frequency in documents, but also on their importance across multiple documents.

For the Deep learning model, the Python package spaCy was used to create the word embeddings. This allowed us to test models on both uncleaned and cleaned data to compare results. The purpose for testing uncleaned data was to determine if spaCy's ability to identify sentences, punctuation, parts of speech, named entities, etc. provided any useful information for our models. Ultimately, the deep learning model had superior performance when using the cleaned data, employing the steps laid out in the prior Data Cleaning section such as lemmatisation, stop words removal, etc.

Modelling

Supervised Learning – Regression

Two different regression models are tested in the experiments, including linear regression and KNN regressor with regularization to obtain predicted models of 'Helpfulness' of reviews. As mentioned, we used 'Helpfulness' as a target variable and 10 numeric features were constructed through feature engineering and used as predictors of 'Helpfulness' (Shown in Figure 22).



```
Other_Features = df_Sample[['n_words', 'n_sents', 'polarity', 'subjectivity', 'n_punct', 'Readability', 'n_PROPN', 'n_NOUN', 'cluster_label', 'topics']]
# Combine the vectorizer with other feature columns
Features = pd.concat([reviews_transformed, Other_Features], axis=1)
Features
```

	0	1	2	3	4	5	6	7	8	9	...	n_words	n_sents	polarity	subjectivity	n_punct	Readability	n_PROPN
0	0.000000	0.000000	0.000000	0.0	0.000000	0.0	0.000000	0.0	0.000000	0.000000	...	51	3	0.450000	0.433333	3	83.082500	2
1	0.000000	0.000000	0.000000	0.0	0.000000	0.0	0.000000	0.0	0.000000	0.000000	...	109	8	0.133571	0.448571	15	93.468140	12
2	0.000000	0.000000	0.000000	0.0	0.000000	0.0	0.000000	0.0	0.000000	0.000000	...	46	3	0.166667	0.533333	5	77.412114	4
3	0.000000	0.000000	0.000000	0.0	0.000000	0.0	0.000000	0.0	0.000000	0.000000	...	29	3	0.428571	0.545238	2	99.278462	2
4	0.000000	0.000000	0.000000	0.0	0.14067	0.0	0.000000	0.0	0.000000	0.000000	...	184	4	0.208594	0.682478	29	87.806431	5
...
9995	0.000000	0.000000	0.000000	0.0	0.000000	0.0	0.239314	0.0	0.000000	0.000000	...	69	4	0.071581	0.411467	4	86.218173	0
9996	0.000000	0.000000	0.000000	0.0	0.000000	0.0	0.000000	0.0	0.000000	0.000000	...	28	2	0.571429	0.633929	4	88.905000	0
9997	0.118794	0.000000	0.000000	0.0	0.000000	0.0	0.134205	0.114171	287	19	0.158333	0.390946	36	89.644084	9
9998	0.000000	0.155029	0.146508	0.0	0.000000	0.0	0.000000	0.0	0.219329	0.000000	...	146	5	0.143348	0.412798	23	82.019130	6
9999	0.000000	0.183561	0.000000	0.0	0.000000	0.0	0.000000	0.0	0.000000	0.000000	...	81	5	0.140545	0.476923	7	97.098056	1

10000 rows x 210 columns

Figure 22. Numeric Features Used as Predictors

Features that were used as predictors include number of tokenized words, number of words that were sent, polarity, subjectivity, number of punctuations, readability, number of pronouns, number of nouns, clustered label and topic modelling.

Linear Regression

Linear regression model is used to predict reviews that are more likely to be helpful. To evaluate the performance of model, R squared score, mean squared error and mean absolute error have been tested as performance metrics. Accordingly, the model showed very poor R squared score (= 0.038), indicating that numeric features we established were not relevant to the 'Helpfulness' of reviews.

Considering that a massive amounts texts in reviews are extremely difficult to predict, it is explainable that the model has a very poor R squared.

```
[ ] r2_score(y_test, mm_linear_preds)

0.03849295179962864

[ ] print(mm_linear_train.score(multi_scaler_testing, mm_linear_preds))

1.0

from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error

multivariate_mae = mean_absolute_error(y_test, mm_linear_preds)
multivariate_mse = mean_squared_error(y_test, mm_linear_preds, multioutput = 'uniform_average')
multivariate_rmse = mean_squared_error(y_test, mm_linear_preds, squared=False)

print("MAE:", multivariate_mae)
print("MSE:", multivariate_mse)
print("RMSE:", multivariate_rmse)

MAE: 0.2864675318845052
MSE: 0.11971367852366367
RMSE: 0.3459966452491464
```

Figure 23. R Squared, Mean Squared Error and Mean Absolute Error for Linear Regression

On the other hand, mean squared error and mean absolute error were relatively great (MSE = 0.12, MAE = 0.29), indicating that the values of predicted model are close to actual values.

	Actual Value (Best Features)	Predicted Value (Best Features)	Difference (Best Features)
6052	1.000000	0.735582	0.264418
1396	0.000000	0.634916	0.634916
8752	1.000000	0.690931	0.309069
355	0.833333	0.795764	0.037569
7665	1.000000	0.744319	0.255681
...
9426	1.000000	0.940137	0.059863
5918	1.000000	0.824106	0.175894
4612	1.000000	0.747524	0.252476
8415	1.000000	0.787709	0.212291
2321	0.500000	0.751823	0.251823

2000 rows × 3 columns

Figure 24. Difference Between Actual Value and Predicted Value

KNN Regressor Model (n = 100, Metric = 'manhattan')

'KNeighbourRegressor' was used as another regression model with 100 neighbors and 'manhattan' metric. Similar to Linear Regression, this model also showed a low R squared (=0.0334) but great mean squared and mean absolute error (MSE = 0.12, MAE = 0.28).

```

knn_mae = mean_absolute_error(y_test, knn_preds)
knn_mse = mean_squared_error(y_test, knn_preds, multioutput = 'uniform_average')
knn_rmse = mean_squared_error(y_test, knn_preds, squared=False)

print("MAE:", knn_mae)
print("MSE:", knn_mse)
print("RMSE:", knn_rmse)

```

MAE: 0.28040961040624973
MSE: 0.12034618167205654
RMSE: 0.34690947172433406

Figure 25. R squared, Mean Squared Error and Mean Absolute Error for KNN Regression

	Actual Value (Best Features)	Predicted Value (Best Features)	Difference (Best Features)
6052	1.000000	0.752147	0.247853
1396	0.000000	0.735501	0.735501
8752	1.000000	0.673423	0.326577
355	0.833333	0.813369	0.019964
7665	1.000000	0.753451	0.246549
...
9426	1.000000	0.701439	0.298561
5918	1.000000	0.877083	0.122917
4612	1.000000	0.799826	0.200174
8415	1.000000	0.876749	0.123251
2321	0.500000	0.876619	0.376619

2000 rows × 3 columns

Figure 26. Difference Between Actual Value and Predicted Value

Visualization

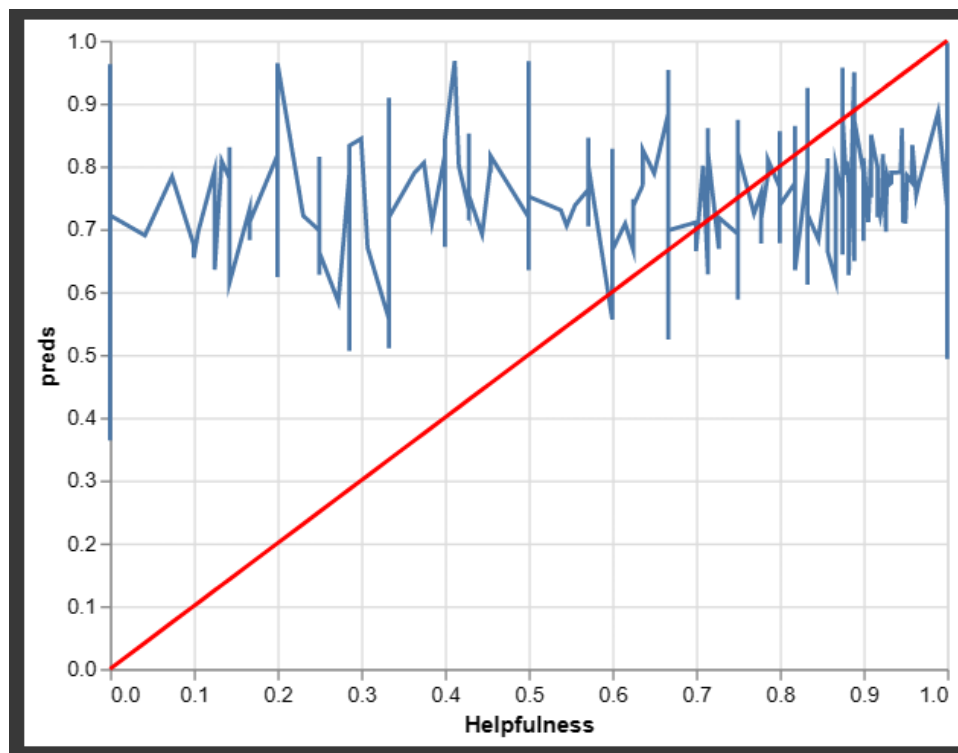


Figure 27. Multivariate Linear Regression Model

As shown in Figure 27, the model mostly predicted high scopes of helpfulness scores, regardless of actual scores of helpfulness. Given that the machine learning tends to better predict the majority of data (High Helpfulness Scores) than minority of data (Poor Helpfulness Scores) and the distribution of helpfulness scores is imbalanced with having most scores of 1.0, the model is very unlikely to predict low helpfulness scores (< 0.6).

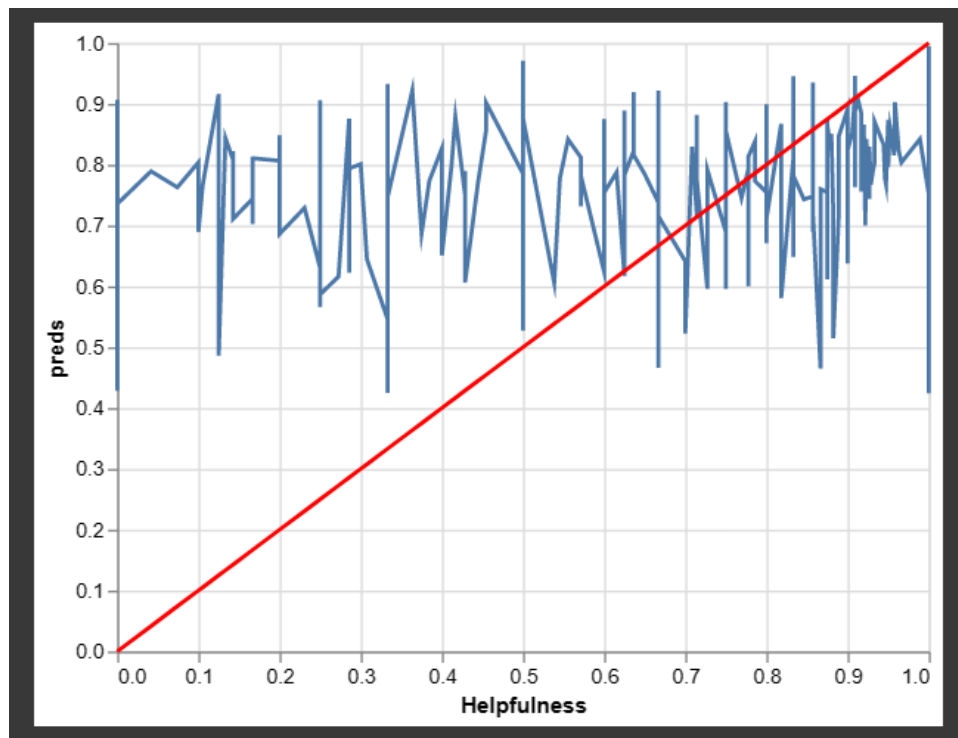


Figure 28. KNN Regression Model

KNN model has a wider scope of helpfulness scores compared to linear regression model. However, it is not still accurate model to predict a low range of Helpfulness score.

Evaluation

We aim to predict 'Helpfulness' of reviews as a target variable. The range of 'Helpfulness' is between 0 to 1.0 and multiple numeric features, which were created by reviews, were used as predictors. The results showed very poor R squared and hence it can be concluded that regression models are not suitable for this dataset. Rather, classification and neural deep learning models are suggested and better to be used as predicted models of data.

Supervised Learning – Classification

Overview:

Several types of supervised classification models are compared, including Logistic Regression, Decision Trees, Support Vector Machines (SVM), and XGBoost, to determine the best supervised classification model to compare with the deep learning model.

Process:

A resampling approach is used to under-sampling the 0 class by selecting samples from the 0 class at random to match the number of records for the 1 class since the labels of the dataset are highly imbalanced. Making labels more balanced prevents biased results and inaccurate predictions.

Before:

Text	
Helpfulness Label	
0	282070
1	16332

Figure 29. Count of reviews by class before resampling

After:

Text	
Helpfulness Label	
0	16332
1	16332

Figure 30. Count of reviews by class after resampling

Then, the Features columns and Label columns are specified, and the dataset is split into 70% train and 30% test. Models of supervised classification, including Logistic Regression, Decision Trees, Support Vector Machines (SVM), and XGBoost, have been tested, and performance statistics such as Accuracy, F1-Score, Precision, Recall, and AUC have been generated for each model to compare them and decide which one performs the best for further improvement. A comparison of the performance results of the tested models can be found below.

Model	Accuracy	Weighted Avg F1-score	Precision	Recall	AUC
LR	64.80 %	0.6470	0.6697	0.5946	0.7028
DT	66.94 %	0.6693	0.6680	0.6837	0.6695
SVM	63.63 %	0.6346	0.6623	0.5676	0.6835
XGB	70.26 %	0.7025	0.7093	0.6942	0.7686

Figure 31. Classification Model Performance Metrics

According to above model performance stats, XGBoost has better overall performance, higher precision and recall, which means it has a better chance of correctly identifying helpful reviews. Additionally, the XGBoost model has an AUC of 0.76, indicating that it can distinguish between helpful and unhelpful reviews well.

By using GridSearch, the XGBoost model is improved further by finding the best combination of `n_estimators`, `max_depth`, and `learning_rate` parameters. GridSearch found that higher value of number of trees and depth of tree and more iterations resulted in better performance. The results also indicate that our model is quite complex and could potentially be improved by increasing iterations, adding depth, and reducing the learning rate in the future. However, a higher number of estimators would require more computing power, and a greater number of depths might lead to overfitting.

Result

After using the best hyperparameter, the XGBoost Model was able to achieve an accuracy of 72.24% with an AUC of 0.80.

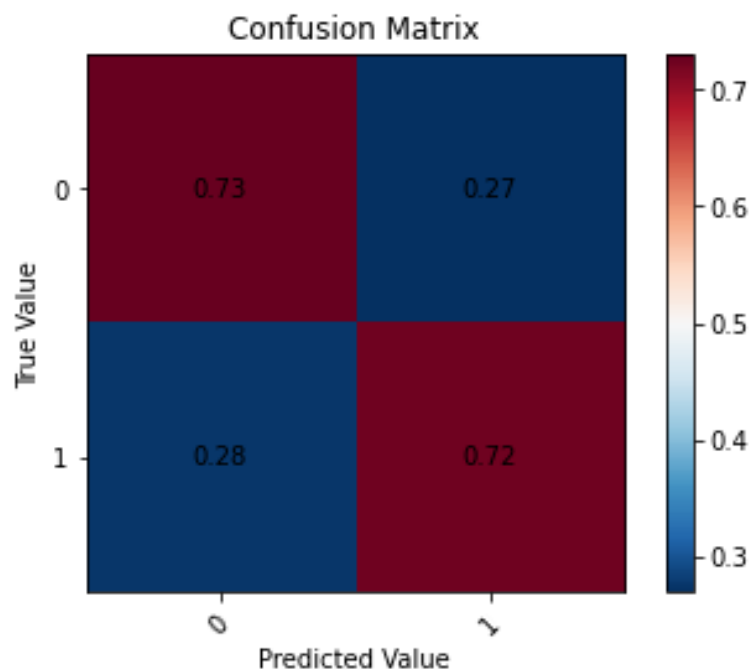


Figure 32. Enhanced XGBoost Model Confusion Matrix

```
Accuracy = 72.71 %  
Weighted Avg F1-score = 0.7271  
precision = 0.7351  
recall = 0.7167  
auc = 0.8024
```

Figure 33. Enhanced XGBoost Model Performance Metrics

Gradient Boost model XGBoost focuses on speed, flexibility, and model performance. It's usually considered the first choice in many scenarios for regression and classification models and has won a lot of Kaggle competitions. It used the boosting mechanism in which each tree corrects the error of the previous one and used a gradient descent algorithm to minimise the error. (David, 2021) As a result, the performance is quite impressive when compared to traditional classification methods.

Deep Learning

Overview

Recurrent Neural Networks (RNN) were the focus of this research given their strength in processing sequential data. Sequence is likely to be of high importance in our data, given there were no obvious trends based on keywords during the topic modelling and clustering efforts. Given the large size of our dataset and its narrow focus in subject matter, a new neural net was created rather than relying on pre-trained models.

Specifically, Long-Short Term Memory (LSTM) and Gated Recurrent Units (GRU) architectures were compared as they have advantages over traditional RNNs which make them suitable for NLP. While traditional RNNs models use information from the previous layer, LSTM and GRU uses the entire sequence which increases their capacity to contextualise information. Performance improvements are particularly significant for large vocabulary problems (Sak et al., 2014).

Both methods also attempt to solve the vanishing gradient problem common to deep learning. The gradients, or values which update weights in a network, become extremely small over time and thus do not contribute to learning.

Process

Initial tests were performed on word embeddings extracted from uncleaned text data. This was done to establish a benchmark, to be later compared with cleaned text data as well as with stacked models containing the additional variables from the feature engineering section. Due to limited compute power, initial tests were done on subsets of our dataset.

LSTM and GRU were found to have comparable results, as such GRU was preferred for scaled up experiments due to the lower computation expense. This is consistent with the prior research literature (Cho et al., 2014). Additionally, learning would plateau earlier for LSTM models in our initial experiments (measured by the stagnation in validation loss).

Bidirectional functionality was also incorporated to improve performance. This is the process by which text is encoded forward and backwards, allowing words at the end of text have stronger influence in long sequences and improving detection of context by considering past and future dependencies (Huang et al., 2015). The drawback of its inclusion is that the computational expense increases significantly to process sequences both ways.

Two layers GRU layers followed by two dense layers were found to be the most optimal for performance, while the variation of the number of neurons and learning rates had little impact. An Adam (Adaptive Moment Estimation) optimiser was chosen since as it is more computationally efficient compared to other optimisers, whose purpose is to adjust a model's learning parameters, while also requiring less parameter tuning and providing better performance (Kingma & Ba, 2014).

Dropout, the process of randomly excluding nodes within the network, was added to improve generalisability (Srivastava et al., 2014). Deeper networks in general are prone to overfit, and exclusions ensures nearby nodes do not adjust their weights at the same time and become correlated. Pooling layers, the process of down-sampling a feature map borrowed from Convolutional Neural Network (CNN) architectures, were also tested as a form of regularisation but were not included in the final model as they did not improve performance.

Class weights were also tested within networks, where different weights are provided to positive and negative classes to help adjust for imbalanced datasets. However, the addition decreased performance of the final model. It is likely that our choice to remove records from the dataset which had only small amounts of helpfulness votes have improved the balance enough to not bias our learners. The inclusion of dropout layers in the final model is also likely to help avoid overfitting the majority class and help generalise the model to out-of-sample data.

Finally, once improvements could no longer be made within the RNN architecture training on text data only, a hybrid network was created to further boost performance. The variables in the earlier feature engineering section were fed into a separate neural network. This new network was then concatenated with the Bidirectional-GRU network, before being fed into another fully connected layer for further processing before the final classification. This process is mapped out in the Figure 34 below.

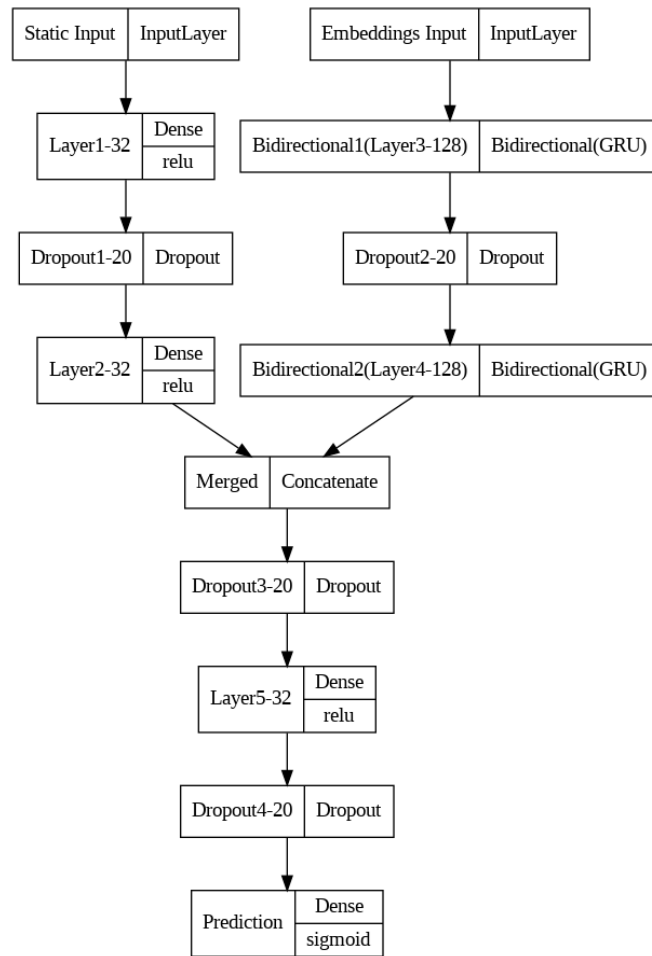


Figure 34. Final Architecture

Results

The final model from the above process was able to achieve an F1 score of 0.85 against a hold-out test dataset. The hybrid model was able to improve upon RNN's by boosting the ability to identify the true negative class.

F1 score was the key metric due to the imbalance of the dataset, as it balances the precision and recall of the model. Precision measures the quality of positive predictions, whether positive predictions are actually positive, while recall measures the how many positive instances within the data have been correctly identified. Simple models would generally have a very high recall, but poor precision.

The summary below also shows strong performance across all metrics. The large gap between the metrics for deep learning compared to the previous classifiers justifies the use of the more complex model.

Model	Accuracy	Weighted Avg F1-score	Precision	Recall	AUC
Bi-GRU	74.06 %	0.8154	0.7637	0.8746	0.7908
Hybrid	79.32 %	0.8504	0.8082	0.8971	0.8374

Figure 35. Deep Learning Performance Metrics

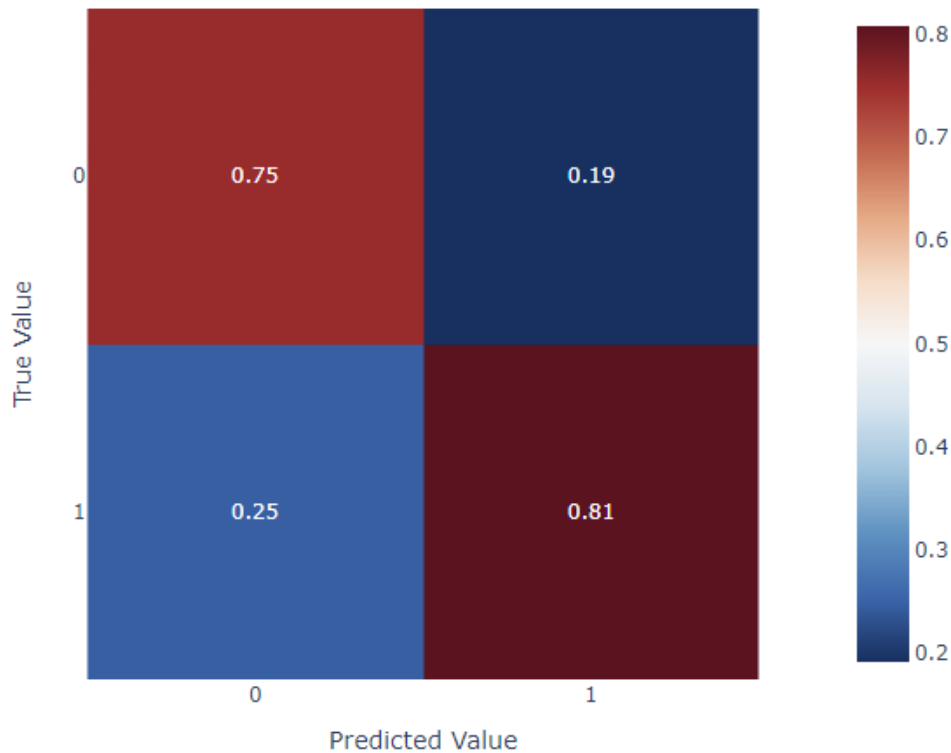


Figure 36. Hybrid Model Confusion Matrix

Ethical Considerations

The major ethical concern with our dataset is the presence of user and product information. These can be used to model a user's past behaviours, particularly with the inclusion of timestamps to further increase the risk of user surveillance or create unintended biases for those users. Even when only encoded user ids are included, multiple researchers have shown it is possible to de-anonymise these by cross-referencing other publicly available datasets. Anonymised users from the well-publicised Netflix Prize dataset, could be de-anonymised by using IMDB to discover preferences and sensitive information (Narayanan & Shmatikov, 2006), or public Amazon data to find users' full names and shopping habits (Archie et al., 2018).

User consent is also a concern as users cannot foresee all the possible outcomes of their data being collected. While users writing these reviews would be aware the content is freely available to the public viewing the Amazon website, they may not necessarily consent to data scientists modelling their past behaviours to reasons unknown to them. Machine Learning can potentially make unfair associations with an individual, for example if a user only reviews certain product or price ranges it might make inferences into their lifestyle, socio-economic status or background.

To protect privacy, the models in this project do not use user identifiers, products or timestamps. Although these are highly likely to be useful in feature engineering to improve accuracy of predictive models (e.g., usefulness of past reviews, number of past reviews, tenure of user, etc.), the aim of this project is to create a strong predictive model based only on the review text and not contain bias towards any users or products.

Findings and outcomes

Performance comparison of models

The hybrid deep learning model outperformed the classification models on all metrics, with a relatively high F1 score of 0.85 compared to approximately 0.73 for the enhanced XGBoost model. These levels indicate a strong model that has a good balance of precision and recall.

The XGBoost model had a higher precision rate than recall, which is indicative of a more conservative approach by not labelling as many helpful reviews to avoid false positives (i.e., promoting an unhelpful review). The deep learning model was able to improve both precision and recall, hence increasing the amount of helpful review labels without mislabelling the unhelpful.

Top features contribute to the classification model prediction

Feature importance is calculated to understand how each feature impacts the classification model, and the top 30 features are shown below. For this model, the word count stands out as the most important feature. To gain a deeper understanding of top features, further analysis is needed.

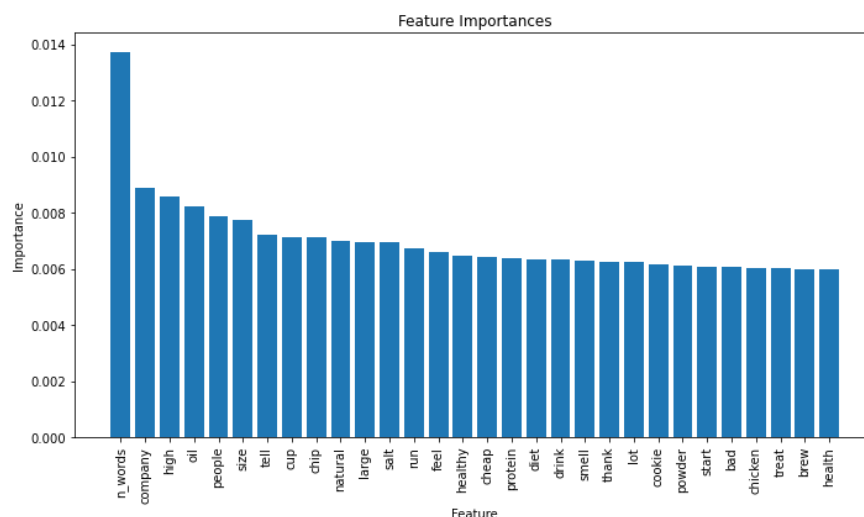


Figure 37. Top features for Enhanced XGBoost Model

Valued Added

A classification model was built to predict Helpfulness for a review. Based on both existing votes for helpfulness and predicted helpfulness, review websites can push the best reviews to the top. Currently, most reviews are predicted based on existing votes; new reviews are harder to predict. As shown in Figure 2 from our EDA earlier, the number of reviews grows exponentially over time, therefore increasing the need to predict how useful a review is before it can receive a lot of votes.

For a company such as Amazon, the depth of their product range is enormous and continuously expanding. This type of predictive model would also benefit new products or users get visibility without the need to wait and hope for enough votes to compete with older reviews. Additionally,

niche products or low-volume, high-margin products would benefit where magnitude of votes will consistently lag those of high-volume products.

Finally, this can be used as a tool for those who write reviews as the inputs into the model are all text based rather than product or user meta-data. If the reviewer's motivation is to inform potential buyers, or provide feedback to sellers, being able to gauge how useful a draft review is before posting can help them refine their writing immediately instead of waiting for validation through votes to adjust their writing style on the next review.

Given the high accuracy of the model on hold out datasets, we have delivered an extremely useful predictive tool to address all the above opportunities. As we have not optimised by product or user data, this model should do well in generalising to other products. A company such as Amazon would be able to leverage its resources to greatly increase the size of the training data and compute power which our project lacked, allowing for further fine-tuning of our model to boost performance even further.

Challenges faced and solutions

Compute power for deep learning – GPU

Deep Learning models required significant computing power to run, which limited the time to experiment with a wide variety of architectures and hyperparameter tuning. Inclusion of Bidirectional models, requiring input flows in forward and backward, further compounded the problem. This required the use Google Colab Pro TPU's with tensorflow, otherwise models could run more than 10hrs or time out frequently. One alternative of using pre-trained models comes with a lot of ethical baggage. Additionally, while these models may have impressive performance for general applications, it is not necessarily clear how well they would suit our specific needs without fine-tuning.

Slow running speed

Since the dataset has around 298k reviews, pre-processing steps such as lemmatization using spacy took a long time. We originally planned to include some additional features like count of proper nouns and nouns for the classification, but the code only worked on a sampler dataset of 10000 records, and failed to generate after 6 hours on the full dataset. As a result, we have to give up on using these features.

Errors occur when generating the Readability Feature

Apart from the slowness of the generation speed, the code returns errors several times when hyperlinks are included in the text or if the review contains fewer than 10 words or less than 2 sentences. To solve this problem, Exceptions are embedded in the loop to bypass ValueError and ZeroDivisionError, and those exceptions are assigned an average readability score of 70.

Imbalanced label

In the dataset, the helpfulness ratings are highly imbalanced with most reviews having helpfulness ratings of 1. Only 5% of reviews are classified as class 1 after we modify the definition of the label to mark reviews that have been rated 10 times and score more than 0.8 as "helpful reviews". To minimize the impact of imbalanced labels, we tried several approaches, including resampling the 0 class for supervised classification model and adjusting the class weights parameters for deep learning models.

Merging Cleaned Data

After stages of pre-processing, cleaning and establishing features, we tried to merge data and features each team member constructed separately, but the ways we cleaned data were slightly different and the number of data did not match when we merged data so we reset the number of data as 10000 for training regression models and 298k records are used for Classification and Deep learning models.

Limitations and future steps

General purpose sentiment analysers such as VADER was able to produce satisfactory classifications of sentiment polarity for returning product reviews, however we must also need to consider its validity and the study context when using this in other domains. When using this tool in specialised domains such as healthcare, it is shown that their performance varied to a great extent across health datasets (He 2019). To increase the validity of sentiment analysers, future work is needed to customize them for specific context sentiment analysis tools for analysing online data.

Even with compelling results from our classification experiments, measuring the feature importance of the most accurate model using deep learning is difficult. The use of the hybrid model, as well as relying on preconfigured cloud environments like Google Colab, meant that methods to extract feature importance such as Shapley values could not be executed. We attempted to circumvent this issue by supplying feature importance insights from our other classifier experiments such as Boosting. Another potential idea to improve explainability would be to calculate Shapley values using the deep learning model trained on only word embeddings, as its accuracy is much closer to our final model. This would require more in-depth data engineering skills to create an environment to handle both model training and evaluation code.

Project Progress Timeline with Milestones Achieved

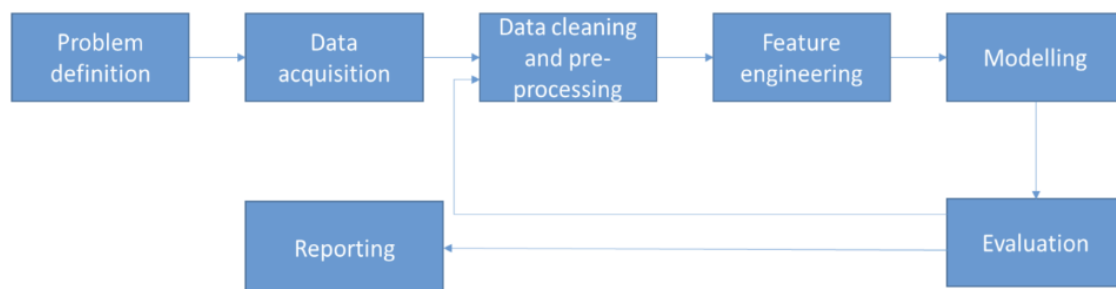


Figure 38. NLP Pipeline

References

- Archie, M., Gershon, S., Katcoff, A., & Zeng, A. (2018). Who's watching? de-anonymization of netflix reviews using amazon reviews.
- Cho, K., Van Merriënboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.
- Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- He, L. and Zheng, K. (2019). How do General-Purpose Sentiment Analyzers Perform when Applied to Health-Related Online Social Media Data? *Studies in health technology and informatics*, [online] 264, pp.1208–1212. doi:<https://doi.org/10.3233/SHTI190418>.
- Huang, Z., Xu, W., & Yu, K. (2015). Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*.
- Hutto, C. and Gilbert, E. (2014). VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text. Proceedings of the International AAAI Conference on Web and Social Media, [online] 8(1). Available at: <https://ojs.aaai.org/index.php/ICWSM/article/view/14550> [Accessed 14 May 2023].
- Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Narayanan, A., & Shmatikov, V. (2006). How to break anonymity of the netflix prize dataset. *arXiv preprint cs/0610105*.
- Nltk.org. (2019). *nltk.sentiment.vader — NLTK 3.4.5 documentation*. [online] Available at: <https://www.nltk.org/modules/nltk/sentiment/vader.html>.
- Sak, H., Senior, A., & Beaufays, F. (2014). Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *arXiv preprint arXiv:1402.1128*.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., (2014). Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* 15, 1 (January 2014), 1929–1958.
- David.M. (2021). *XGBoost: A Complete Guide to Fine-Tune and Optimize your Model - Towards Data Science*. [online] Available at: <https://towardsdatascience.com/xgboost-fine-tune-and-optimize-your-model-23d996fab663>
- Pamela O. (2021). Flesch Reading Ease Test: What Readability Means for SEO? - Flexisource IT. [online] Available at: [Flesch Reading Ease Test: What Readability Means for SEO? - Flexisource \(flexisourceit.com.au\)](https://flexisourceit.com.au)

Appendix

Link to group code:

<https://github.com/annaly114141/AT2-Group-35>

Individual Contribution and Reflection

- Data description
 - Data Dictionary - Simon
 - Reason to choose the dataset – Anna, Simon
 - Data EDA - Ana
 - Data Cleaning - Everyone
- NLP methods and techniques (including models, justification for usage, trials conducted, and ethical implications)
 - Topic Modelling - Martin
 - Clustering - Simon
 - Sentiment Analysis - Anna
 - Features Engineering – Everyone
 - Embedding/Vectorization – Everyone
 - Supervised Learning - Regression - Simon
 - Supervised Learning - Classification – Michelle
 - Deep Learning - Martin
- Findings – insights from the data, model and evaluation– Everyone
 - Performance Comparison – Michelle and Martin
 - Project delivery outcomes and value added - Everyone
- Challenges faced and solutions - Everyone
- Limitations and future steps - Everyone
- Project progress timeline with milestones achieved

Reflection

At first stage, we did EDA, data cleaning, analysis, and training models separately and planned to compare and merge the results. Although we have had two meetings weekly and checked our progress every meeting, we were only able to report our progress but a bit difficult to be interacted with each other's codes and analysis. At last, we all came up with different outcomes and just picked one that showed better results. I mainly did regression analysis for my parts, but it was later confirmed that regression models did not fit into the dataset. I believe if our team started with

assigning more specific roles and parts from the beginning, we could have done NLP analysis and training different models more smoothly and supported each other. Nevertheless, our team works were successful we gave each other many valuable feedbacks during the meetings and each team member's dedication resulted in good quality of group assignment.