Università degli Studi di Milano-Bicocca
Corso di Laurea Magistrale in Data Science

Text Classification and Summarization on Amazon Fine Food Reviews Dataset

FRANCESCA CORVINO ENRICO MANNARINO CHRISTIAN PERSICO
898058 850859 829558

ACADEMIC YEAR 2023/2024

Contents

Introduction **1**

1 Data Exploration **2**

2 Text Processing **5**

2.1 Normalization 5

2.2 Language detection and correction 6

2.3 Decontractions 6

2.4 Tokenization and stopwords removal 6

2.5 Lemmatization 7

2.6 Final result 7

3 Text Classification **8**

3.1 Class imbalance 8

3.2 Text representation 10

3.3 Classification metrics 10

3.4 Logistic model 11

3.4.1 BOW 11

3.4.2 TF-IDF 13

3.5 Random Forest 14

3.5.1 BOW 14

3.5.2 TF-IDF 16

3.6 Support Vector Machine 16

3.6.1 BOW 16

3.6.2 TF-IDF 17

3.7 Model selection and improvements 18

4 Text Summarization **20**

4.1 Extractive summarization 20

4.1.1 LSA 20

4.1.2 Bert Extractive Summarizer 21

4.1.3 Automatic Evaluation 21

4.2 Abstractive summarization 22

4.2.1 BART 22

4.3 Human evaluation 23

Conclusions **25**

Introduction

In today's fast-paced world of online shopping, customer reviews can play a significant role in influencing purchasing decisions. Platforms like Amazon have made it easier than ever for customers to share their feedback, helping other shoppers make informed choices. In this context we find our dataset consisting of fine food reviews from Amazon shared over a period of more than 10 years. The project aims to accomplish two primary tasks, using text classification and text summarization. In particular, through review classification, we aim to automatically separate positive feedbacks from the negative ones, giving businesses transparent insights into customer perceptions. On the other hand, text summarization is vital in condensing lengthy reviews into informative summaries, ensuring that critical opinions and informations are preserved. This method simplifies the review analysis process for both sellers and potential buyers, facilitating the comprehension of the overall attitude toward a product. By doing so, we are able to transform unstructured reviews into structured, actionable data.

1. Data Exploration

The *Amazon Fine Food Reviews* dataset contains 568,454 reviews about 74,258 products of fine foods, contributed by 256,059 unique users. In the dataset, there are 10 variables:

- **ID** (int64): a unique identifier for each review in the dataset
- **ProductId** (object): the unique identifier of the reviewed product
- **UserId** (object): the unique identifier of the user who wrote the review
- **ProfileName** (object): the profile name of the reviewing user
- **HelpfulnessNumerator** (int64): the number of votes indicating a review's helpfulness
- **HelpfulnessDenominator** (int64): the total number of votes a review has received for helpfulness
- **Time** (int64): the time when the review was posted in UNIX format
- **Summary** (object): a brief summary of the evaluation provided by the user
- **Text** (object): the full text of the review detailing the user's experience with the product
- **Score** (int64): a rating between 1 and 5

Figure 1.1: Data display

After inspecting the data, we found 16 missing values for the *ProfileName* and 27 for the *Summary* feature, and replaced them with an empty string. We then converted the Unix timestamps in the *Time* column to a datetime format and saw that the reviews span from 8th October 1999 to 26th October 2012. This operation allowed us to create a histogram, visualizing annual trends that indicated the changes in the frequency of reviews over the years (Figure 1.2):

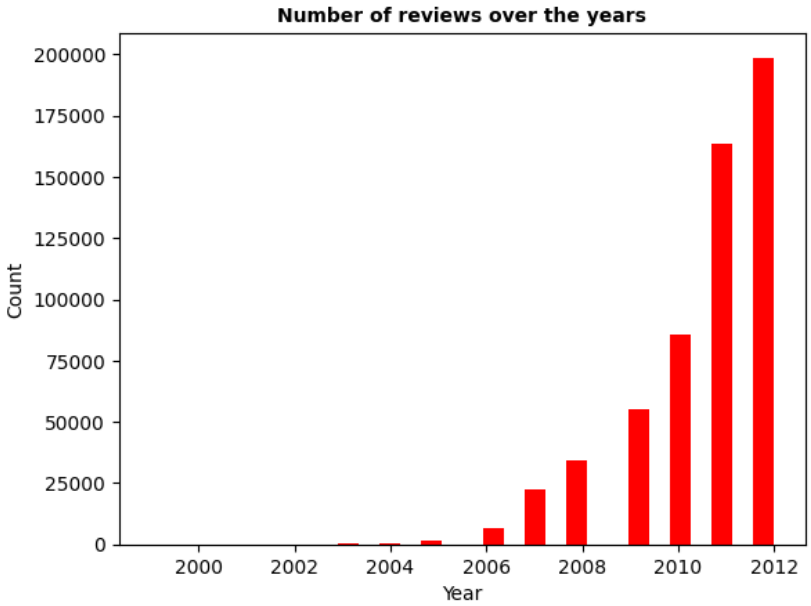

Figure 1.2: Distribution of the reviews over the years

The histogram showed a steady increase in reviews from year to year, with a significant

acceleration since the mid-2000s. This growth can be attributed to the development of e-commerce and the increasing popularity of Amazon as a platform for food purchases. So, we took an additional step to examine the distribution of review scores as shown in Figure 1.3:

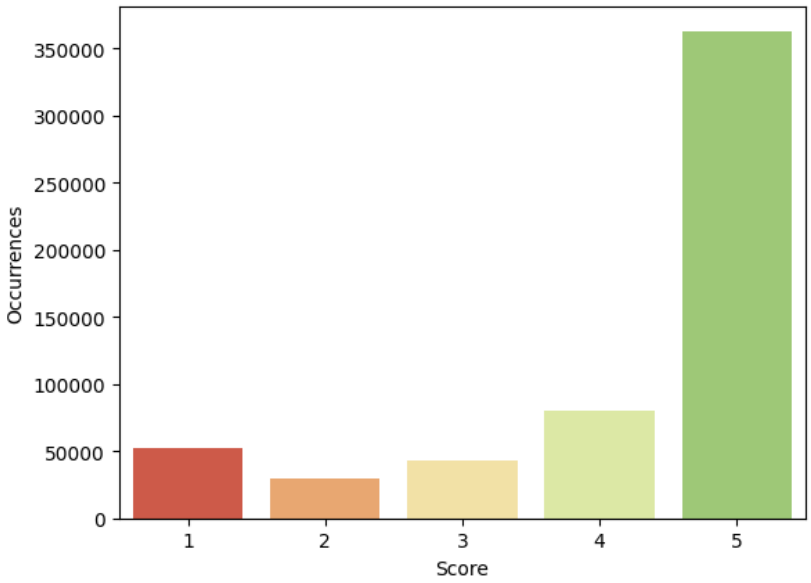

Figure 1.3: Distribution of review scores

The analysis revealed that the dataset is significantly imbalanced, as most reviews have a 5-star rating. This indicates that the data is positively skewed, with customers more likely to leave higher ratings - as shown in Figure 1.3. Next, we focused on detecting duplicate rows in the dataset considering two key columns - the *Score* and *Text* fields - the most reliable indicators of a review's uniqueness and relevance. The 'Score' reflects the reviewer's overall product assessment, while the 'Text' column contains their detailed opinions and experiences. Our approach to identifying duplicate entries involved searching for rows with the same values in these two fields because it is doubtful for two independent reviewers to express their opinions in the exact words and assign the same score. By removing these instances, we get a final dataset composed of 393,675 rows - 174,779 less than the original one - now ready to be fed into the processing step.

2. Text Processing

Text processing involves cleaning and transforming unstructured text data preparing it for analysis.

2.1 Normalization

The processing phase started with the creation of a text normalization function that consolidates several key operations to clean and standardize the texts:

- **Lowercase Conversion**: transforms all text to lowercase to prevent the exact words from being counted as distinct tokens in different cases.
- **Link/HTML Removal**: strips out any embedded HTML content and hyperlinks that are irrelevant for text analysis and could potentially skew the results;
- **Emoji Removal**: although emojis can convey sentiment, they are removed to maintain textual consistency and to focus the analysis on linguistic content. We also defined a set of ASCII emojis to replace that kind of emojis as well;
- **Accent Removal**: normalizes characters with accents to their simplest form to ensure uniformity across the dataset, as accents can lead to the same word being treated as different tokens. To do so, we used NFKD, which is a Unicode normalization form that systematically decomposes characters into their base components and separate diacritics - for instance, it transforms "é" into "e" plus the distinct acute mark "´";
- **Punctuation Removal**: clears away all punctuation marks, as these can interfere with many text mining tools and do not typically contribute to the meaning of words in the analysis;
- **Whitespace Normalization**: trims extra spaces, tabs, and newline characters to clean up the text and avoid the misinterpretation of word boundaries;

2.2 Language detection and correction

In order to address the issue of linguistic diversity in our dataset, the first step was to detect the language of each review using language detection algorithms to ensure accuracy and consistency in our analysis. For those reviews that were not in English (169), we used the Google Translator API to automatically translate them into English. After translation, we further refined the text by normalizing it and then converted all text to lowercase because we noticed that after translating, the algorithm capitalized the first letter.

Example of a review: "Ste es el mejor aceite que he usado desde que cocino comida asiatica el lijero y no se quema facilmente cuando pones comida a temperatura muy alta y tiene un olor que despierta tu apetito inmediatamentellego perfectamente empacado y felicito a los distribuidores por tener este producto de primera calidad" becomes "This is the best oil I have used since I cooked asian food and it does not burn easily when you put food at a very high temperature and it has a smell that awakens your appetite immediately. It arrived perfectly packaged and i congratulate the distributors for having this top quality product".

2.3 Decontractions

As part of our processing workflow, we encountered a common issue with textual contractions often present in customer reviews. To address this problem and improve the clarity and consistency of our dataset, we designed a function called *decontractions* that expands contractions to their full form, replacing contracted words like "can't" with their expanded equivalents like "can not". To ensure uniformity, we also modified "cannot" to "can not". By using a regular expression substitution method, we were able to transform both common contractions and specific cases like "cannot" uniformly.

Example: "I dont" and "I wont" becomes "I do not" and "I will not".

2.4 Tokenization and stopwords removal

We also managed stop words, which are words that have minimal lexical content and are commonly removed during text processing. First, we started with a standard English stop

words list provided by the *nltk* library and made some strategic adjustments to customize this list for our project. For instance, we removed the word "not" from the list to preserve negations, which are crucial for sentiment analysis. On the other hand, we added words like "would", "product", and "Amazon" to the list because they frequently appeared in our dataset without any contextual significance.

So, to filter out stop words we implemented a custom function, which uses the WordPunctTokenizer to tokenize the text into individual words and iteratively removes any tokens found in our customized stop words list. This process is essential for reducing noise in the dataset and focusing on words that significantly contribute to the analytical outcome.

2.5 Lemmatization

Lemmatization is a technique that normalizes text by reducing words to their base or dictionary form, called lemma. Unlike stemming, which simply removes word endings, lemmatization considers the context and morphology of words, resulting in more accurate and meaningful results. For example, "running" becomes “run", and "better" is converted to "good". We incorporated this approach into our processing pipeline using the WordNetLemmatizer from the Natural Language Toolkit (NLTK) ensuring that variations of a word are all analyzed as the same token, and so focusing on the essence of the word rather than its form.

2.6 Final result

Lastly, we created a new column *Y* that represents our target variable for the classification task and is derived from the *Score* variable. We assigned a value of 1 to reviews with a *Score* greater than 3 (i.e., scores of 4 and 5), indicating positive reviews. Conversely, reviews with a *Score* of 3 or lower were assigned a value of 0, denoting them as negative. This binary classification approach simplifies the model’s task by categorizing reviews into two distinct classes: positive and negative.

3. Text Classification

In binary classification, algorithms are trained to analyze text and make predictions, categorizing each piece of text into one of two predefined categories based on its content. This approach is particularly effective in applications like review analysis, where determining whether a review is favorable or unfavorable is essential.

3.1 Class imbalance

The first thing we did was to split the dataset into a training set containing 70% of the records and a test set composed of the other 30%. Inspecting the data, we noticed that the training set was not balanced from the perspective of the target variable. In other words, the number of records belonging to the positive and negative classes was not equal. Specifically, there were 86,856 records belonging to the negative class and 306,814 records belonging to the positive class (Figure 3.1). This imbalance could potentially lead to inaccurate results and mislead our model's outcome.

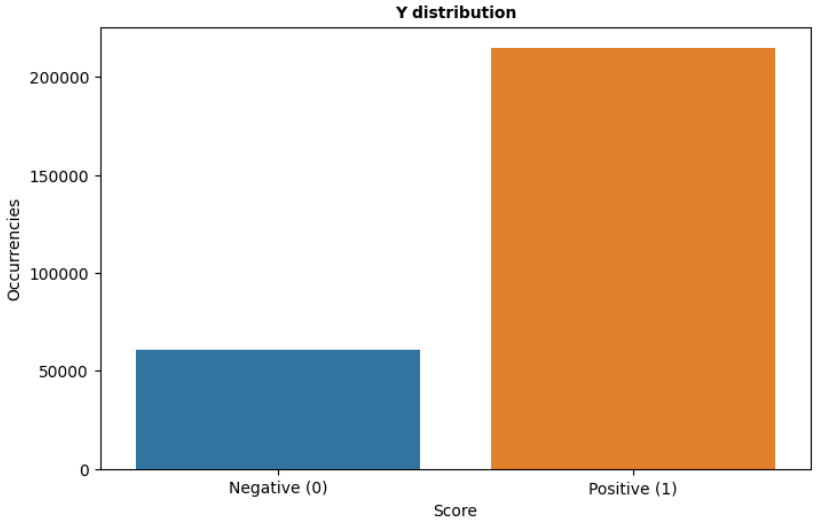

Figure 3.1: Class imbalance

To mitigate this problem, we implemented an undersampling strategy reducing the size of the most numerous class to equalize it with the least represented one, thus creating a more

balanced dataset.

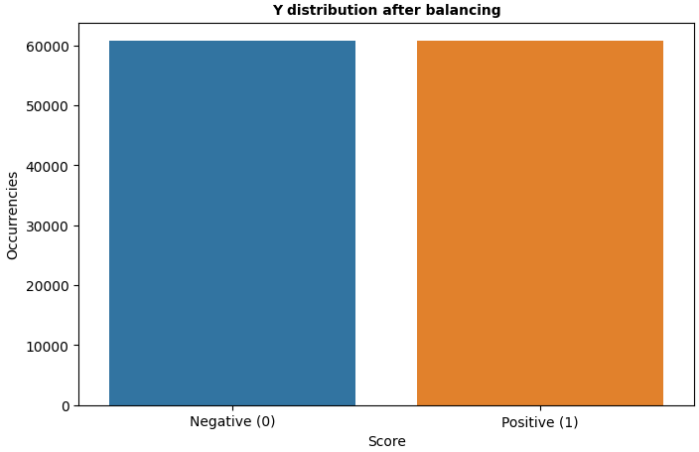

Figure 3.2: Distribution after balancing

We then examined the distribution of review lengths to understand our dataset better (Figure 3.2). The length of reviews can reveal the amount of information they contain, affecting the accuracy of classification models.

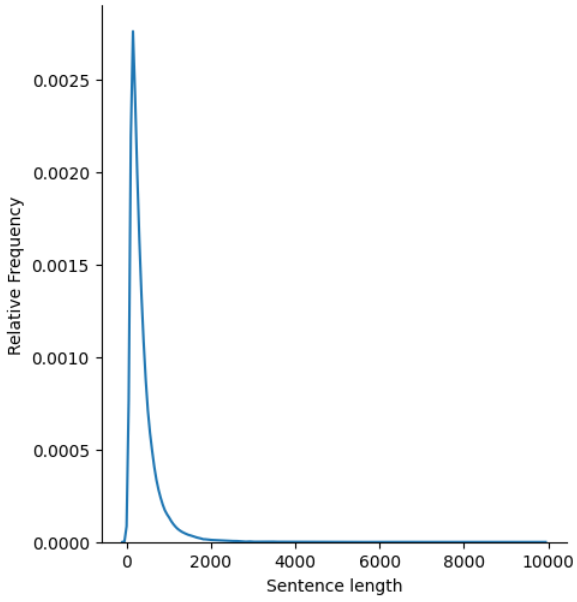

Figure 3.3: Distribution length reviews

According to the plot, shorter reviews are more common, with a sharp decline as the length increases.

3.2 Text representation

In text mining, one of the crucial steps is converting unprocessed text into a structured format that can be effectively processed by machine learning algorithms. This process is known as text representation. For the models that we are planning to use in our project, we have employed two commonly used techniques for text representation: Bag of Words (BOW) and Term Frequency-Inverse Document Frequency (TF-IDF).

- **The Bag of Words (BOW)** approach is a simple but effective method used in natural language processing. Essentially, it involves creating a "bag" of all the words that appear in a given text corpus, without taking into account their order or context. Each unique word represents a feature, and the text is subsequently represented as a vector, which indicates the presence or frequency of each word within the text.

- **Term Frequency-Inverse Document Frequency (TF-IDF)** is an extension of the BOW concept. While BOW considers only the frequency of a word in a document, TF-IDF takes into account the rarity of a word in all documents in the corpus. This approach assigns more weight to unique terms that appear only in a single document, which may indicate their higher importance and less weight to common words that appear across many documents.

In both cases, we set a threshold of 0.001 to determine the minimum document frequency for words to be included in the analysis. Words that appear in fewer documents than this threshold are excluded. For the BOW representation, the minimum value of zero indicates that some words do not appear in certain reviews, while the maximum value of 65 represents the most frequent term's occurrence across all reviews. In the TF-IDF method, we got a minimum value of 0 and a maximum value of 1.00, indicating that the word has the highest unique relevance in that document compared to others in the corpus.

3.3 Classification metrics

During the evaluation phase, we used standard metrics to measure the effectiveness of our models. These metrics are essential for understanding how well the predictions align with

our labels. The classification report provides an overall view of the performance, including metrics such as precision, recall, and F1-score for both classes. These scores give us a measure of the number of the model's positive predictions that are correct, the number of actual positive instances that are correctly identified, and a balance between precision and recall. For predictions on the test set, we used a confusion matrix to visually display the relationship between observed labels and those predicted by the model. This intuitive graphic representation highlights the number of true positives, true negatives, false positives, and false negatives, offering an immediate picture of the model's predictive accuracy. Moreover, we analyzed specific examples that the model incorrectly classified as false positives and false negatives helping us to pinpoint areas where the model can be improved and provides insights into possible reasons for errors. Lastly, we examined the most informative features, i.e., the words that carry the most weight in the model's classification decision.

3.4 Logistic model

Logistic regression is a statistical model that is commonly used for binary classification tasks. In our context, this model predicts the probability of a given input belonging to one of two distinct classes, such as 'Positive' or 'Negative'. The final decision of the model is based on a threshold (in our case, we used the default value which was set to 0.5): if the computed probability exceeds this threshold, the input is classified into one class, otherwise into the other.

3.4.1 BOW

After applying the logistic regression model in conjunction with the BOW text representation technique, we analyzed the performance of both the training and test sets (Figure 3.14):

	precision	recall	f1-score	support
Negative	0.85	0.86	0.86	60137
Positive	0.86	0.85	0.86	61461
accuracy			0.86	121598
macro avg	0.86	0.86	0.86	121598
weighted avg	0.86	0.86	0.86	121598

(a) Training set

	precision	recall	f1-score	support
Negative	0.84	0.61	0.71	35880
Positive	0.85	0.95	0.90	82221
accuracy			0.85	118101
macro avg	0.84	0.78	0.80	118101
weighted avg	0.85	0.85	0.84	118101

(b) Test set

Figure 3.4: BOW Logistic model metrics

During the training process, the model displayed good levels of precision, recall, and F1-score, with scores of roughly 0.86 for both positive and negative reviews. This suggests a well-balanced proficiency in accurately identifying both types of reviews. Although the precision remained consistent when applied to the test set, there was a variation in recall, especially for the 'Negative' class. This implies that while the model is still proficient in correctly identifying positive reviews (as shown by the higher recall score for the 'Positive' class), it is not as effective in identifying negative reviews in the data that it has yet to see during training. The overall accuracy of 0.85 remained robust, indicating that most predictions were correct. However, the model's performance varied significantly between the two classes, highlighting the importance of considering other evaluation metrics. We referred to the ROC curve for this purpose, which presented an Area Under the Curve (AUC) score - an accurate and single performance measure across all classification thresholds - of 0.91, indicating a high discrimination level between the positive and negative classes. After building our model, we proceeded to analyze its most predictive features. This allowed us to determine which words were the strongest indicators of positive and negative reviews. We found that words with high positive or negative coefficients were strongly associated with their respective sentiments. The results of our analysis are as follows:

-2.4510	worst	2.3237	skeptical
-2.2647	disappointment	1.9476	pleasantly
-2.1912	ripoff	1.7612	yum
-2.1439	mediocre	1.6491	hook
-1.9453	unacceptable	1.3967	highly
-1.9196	concept	1.3641	delicious
-1.7585	useless	1.3343	versatile
-1.7023	terrible	1.3184	hesitate
-1.5686	inedible	1.2655	fragrant
-1.5541	deceptive	1.2596	perfect
-1.5305	undrinkable	1.2495	excellent
-1.5194	ugh	1.2238	importantly
-1.4991	disgust	1.1979	yummy
-1.4795	awful	1.1953	whim
-1.4692	unfortunately	1.1855	fabulous
-1.4663	nope	1.1825	awesome
-1.4627	rancid	1.1815	beat
-1.4343	hop	1.1480	terrific
-1.4298	cancel	1.1406	addict
-1.3805	tasteless	1.1176	worry

Figure 3.5: Most informative features

3.4.2 TF-IDF

We then applied the TF-IDF representation and obtained the following results for the training and test sets:

	precision	recall	f1-score	support
Negative	0.86	0.86	0.86	61223
Positive	0.85	0.86	0.86	60375
accuracy			0.86	121598
macro avg	0.86	0.86	0.86	121598
weighted avg	0.86	0.86	0.86	121598

(a) Training set

	precision	recall	f1-score	support
Negative	0.86	0.61	0.71	36812
Positive	0.84	0.95	0.89	81289
accuracy			0.84	118101
macro avg	0.85	0.78	0.80	118101
weighted avg	0.85	0.84	0.84	118101

(b) Test set

Figure 3.6: TF-IDF Logistic model metrics

The results obtained using the TF-IDF method are similar to those observed with the Bag of Words approach. The model shows good outcomes on the training set but, when tested on the unseen data in the test set, there is a slight drop in performance metrics. Despite this decrease, the results are still reliable, suggesting that the model has good generalization capabilities. The AUC value is also slightly increased to 0.92.

The most informative features are now the following:

-8.6857	worst	8.1036	delicious
-8.0966	not	7.9041	great
-7.2186	disappoint	7.4417	perfect
-6.7003	disappointment	7.3915	best
-6.6353	unfortunately	7.0188	highly
-6.4977	terrible	6.7160	love
-6.1690	ok	6.3081	excellent
-6.1156	hop	5.7912	wonderful
-6.0587	awful	5.5137	pleasantly
-5.7617	horrible	5.3446	amaze
-5.4833	disgust	5.1715	hook
-5.4289	bland	4.9276	awesome
-5.4025	return	4.9044	favorite
-5.2803	okay	4.8225	yummy
-5.0640	stale	4.7905	glad
-4.7008	lack	4.6186	skeptical
-4.6185	weak	4.4855	worry
-4.6100	however	4.4284	happy
-4.4084	refund	4.3152	yum
-4.3705	concept	4.1509	addict

Figure 3.7: Most informative features

3.5 Random Forest

Random Forest is a popular ensemble learning technique that is often used in text mining for classification purposes. This method works by generating multiple decision trees during the training phase and then selecting the class that is most frequently predicted by these individual trees.

3.5.1 BOW

When employing the Random Forest classifier in conjunction with the Bag of Words (BOW) approach for text representation, we observed the following performance metrics on the training and test sets:

	precision	recall	f1-score	support
Negative	0.79	0.80	0.79	60277
Positive	0.80	0.79	0.79	61321
accuracy			0.79	121598
macro avg	0.79	0.79	0.79	121598
weighted avg	0.79	0.79	0.79	121598

(a) Training set

	precision	recall	f1-score	support
Negative	0.78	0.50	0.61	40463
Positive	0.78	0.93	0.85	77638
accuracy			0.78	118101
macro avg	0.78	0.72	0.73	118101
weighted avg	0.78	0.78	0.77	118101

(b) Test set

Figure 3.8: BOW Random Forest metrics

The Random Forest model performed well on the training set, achieving balanced precision, recall, and F1-score of approximately 0.80 for both the 'Negative' and 'Positive' classes. However, on the test set, while the overall accuracy remained stable at 0.78, there was a noticeable difference in recall between the two classes - higher for the 'Positive' one - indicating a tendency to predict positive outcomes more accurately than negative ones. The F1-scores also reflect this difference, with a higher score for positive reviews. The AUC value also decreased to 0.86. Then the feature importance algorithm has computed the top ten most important features and provided the result:

```
Top 10 Feature Importance:

not: 0.09051019653468385
love: 0.05258313065098667
great: 0.05217696615727457
bad: 0.034820663626306474
best: 0.03133535235785007
disappoint: 0.028501298838598907
perfect: 0.02653736523091155
delicious: 0.0251775373976598
ok: 0.01916613791625092
think: 0.016940613107366883
```

Figure 3.9: Feature importance

Words like "not" and "love" have the highest importance scores, suggesting they are key indicators of sentiment in the reviews.

3.5.2 TF-IDF

We then repeated the same steps with the TF-IDF text representation:

	precision	recall	f1-score	support
Negative	0.79	0.79	0.79	61511
Positive	0.78	0.79	0.79	60087
accuracy			0.79	121598
macro avg	0.79	0.79	0.79	121598
weighted avg	0.79	0.79	0.79	121598

(a) Training set

	precision	recall	f1-score	support
Negative	0.79	0.49	0.61	41856
Positive	0.77	0.93	0.84	76245
accuracy			0.77	118101
macro avg	0.78	0.71	0.72	118101
weighted avg	0.78	0.77	0.76	118101

(b) Test set

Figure 3.10: TF-IDF Random Forest metrics

The model performed consistently on the training set, with metrics such as precision, recall, and F1-score all at 0.79. However, there appears to be a similar discrepancy in the values of the test metrics as in previous cases. We have obtained the same value of 0.86 for the AUC measure as with the BOW text representation technique and also quite the same words in feature importance.

3.6 Support Vector Machine

The Support Vector Machine (SVM) is a supervised machine learning algorithm that helps in classifying data into different classes by finding the optimal hyperplane. SVM is particularly effective in high-dimensional spaces, such as those created by textual data.

3.6.1 BOW

Our SVM model with Bag of Words (BOW) showed suboptimal performance, indicating the need for further analysis and model refinement:

	precision	recall	f1-score	support
Negative	0.29	0.52	0.37	34311
Positive	0.73	0.51	0.60	87287
accuracy			0.51	121598
macro avg	0.51	0.51	0.48	121598
weighted avg	0.60	0.51	0.53	121598

(a) Training set

	precision	recall	f1-score	support
Negative	0.28	0.22	0.24	33512
Positive	0.72	0.78	0.75	84589
accuracy			0.62	118101
macro avg	0.50	0.50	0.50	118101
weighted avg	0.59	0.62	0.60	118101

(b) Test set

Figure 3.11: BOW Support Vector Machine metrics

The model performed with a precision of 0.29 for the 'Negative' class and 0.73 for the 'Positive' class on the training set. The F1-scores, which balance precision and recall, were relatively low for the 'Negative' class at 0.37. This suggests that the model had difficulty in correctly identifying negative reviews. The performance on the test set showed similar challenges, with precision and recall for the 'Negative' class at 0.28 and 0.22, respectively, and an F1-score of 0.24, which confirms the above. These unsatisfactory performances translate into informative words for the model that have no overtly positive or negative meaning.

3.6.2 TF-IDF

Implementing the SVM model with TF-IDF text representation yielded a little better results on our training and test sets:

	precision	recall	f1-score	support
Negative	0.55	0.60	0.57	55112
Positive	0.64	0.58	0.61	66486
accuracy			0.59	121598
macro avg	0.59	0.59	0.59	121598
weighted avg	0.60	0.59	0.59	121598

(a) Training set

	precision	recall	f1-score	support
Negative	0.55	0.30	0.39	47815
Positive	0.64	0.83	0.72	70286
accuracy			0.62	118101
macro avg	0.59	0.57	0.55	118101
weighted avg	0.60	0.62	0.59	118101

(b) Test set

Figure 3.12: TF-IDF Support Vector Machine metrics

We see that the results are slightly better than the previous case but still worse than the ones we get with the other models.

3.7 Model selection and improvements

We prioritized using the AUC as our decisive metric, taking into account the trade-off between a true positive rate and a false positive rate. After evaluating the models based on AUC value, the logistic one with the TF-IDF text representation technique performed the best (whose forecasts are shown in the matrix in Figure 3.13).

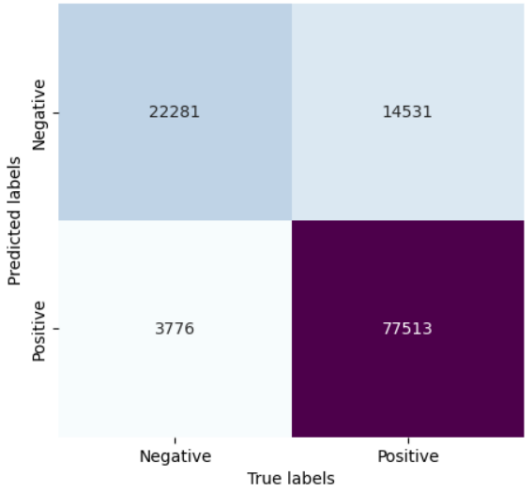

Figure 3.13: Confusion matrix of TF-IDF Logistic model

Next, we aimed to improve our model's accuracy searching for the most common words that appeared in misclassified reviews to identify any pattern or similarities in predicition errors. By recognizing these aspects, we could refine our text representation or adjust the model to enhance its predictive accuracy.

We realized that word semantic can change depending on context, like in "absolutely not a good chocolate" - potentially expected as a positive review. To improve our feature set, we

Table 3.1: Most frequent tokens in misclassified reviews

not	29658
like	11888
taste	10130
get	7406
good	7371

have decided to integrate n-grams, which let the model consider n contiguous tokens from a given sample of text. We employed the TF-IDF Vectorizer with a specified n-gram range to capture both unigrams and bigrams in our texts. After training with this updated set of features, we observed a significant improvement in our model's performance metrics.

	precision	recall	f1-score	support
Negative	0.87	0.64	0.74	35048
Positive	0.86	0.96	0.91	83053
accuracy			0.86	118101
macro avg	0.87	0.80	0.82	118101
weighted avg	0.87	0.86	0.86	118101

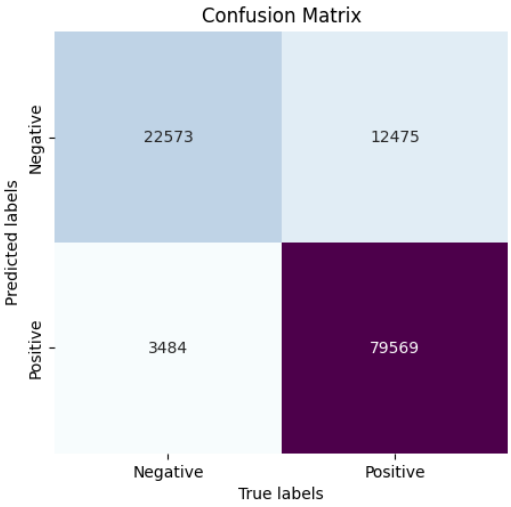

Figure 3.14: Uni-Bigrams Logistic model metrics on test set

The precision scores for the 'Negative' and 'Positive' classes were 0.87 and 0.86, respectively. The recall for the 'Negative' class increased to 0.64, and the 'Positive' class maintained a high recall of 0.96. These findings suggest that the model has become better at identifying negative reviews while still performing strongly in identifying positive ones. The model achieved an average accuracy of 0.86 on the test set and obtained the highest AUC value achieved thus far of 0.94.

4. Text Summarization

The process of automatically summarizing texts is an essential tool in many fields, including journalism, financial analysis or advertising. These professionals need to obtain and check a large amount of information in a short period of time, so they require a method that can facilitate this task. There are two types of procedures for performing this operation:

- **Extractive summarization** that consists of the selection from the text of one or more sentences, which best represent the concept in the written document;
- **Abstractive summarization** which is the method of generating the summary of the document in a completely new way.

Clearly, the former procedure is faster while the latter is more computationally expensive. This will be reflected in our work, from all the initial reviews, only 1000 will undergo the first procedure and only 10 of them will be the input of the more advanced technique.

4.1 Extractive summarization

4.1.1 LSA

Latent Semantic Analysis is an unsupervised technique that derives the implicit representaion of a document semantic from the co-occurrence of words, effectively extracting a summary from the textual input given. To achieve this, it represents the input using a matrix and the TF-IDF method for the weights. This matrix is decomposed as follows:

$$A = U\Sigma V^T$$

- U is an n (words) by m (topics) matrix of real numbers;
- Σ is diagonal m (topics) by m (topics) matrix and the single entry of the matrix corresponds to the weight of the "topic";
- V is an m (sentences) by m (topics) matrix.

So, becomes evident how this decomposition enables the selection of sentences, that best capture the representation of the concept in the document. In Python is available the *sumy* package, for extracting summary from HTML pages or plain texts using LSA.

Note: the mentioned package already incorporates text processing modules, so the summary function was supplied with the original texts without any pre-processing applied.

4.1.2 Bert Extractive Summarizer

In this second example of extractive summarizations, we implemented Bert Extractive Summarizer, from the pyhton library *bert-extractive-summarizer*.

The algorithm functions through a three-step process:

1. Embed the sentences representation of sentences as vectors of real numbers, enabling comparisons of sentence similarity;

2. Run a clustering algorithm over the newly created matrix;

3. Find the sentences that are closest to the cluster's centroids.

We expect the fidelity in the summary generated by the function of this library to be higher because it also uses coreference techniques (expressions in a given text that refer to the same entity), from the *neuralcoref* library, to resolve words in summaries that need more context.

Note: also this package already incorporates text processing modules, so the summary function was supplied with the original texts without any pre-processing applied.

4.1.3 Automatic Evaluation

Human factor is not negligible when we want to perform an automatic, and effective, inspection of the predictions. To carry out this control, summaries made by the model are compared with those made by professionals and Rouge is the score used.

Note: it's computed using the appropriate function located in the *rouge* package by Paul Tardy and not *rouge-score* package made by Google.

Results:

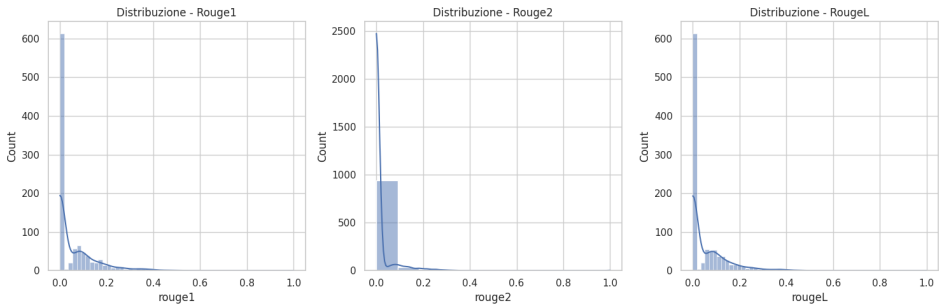

Figure 4.1: Rouge Score on LSA predictions

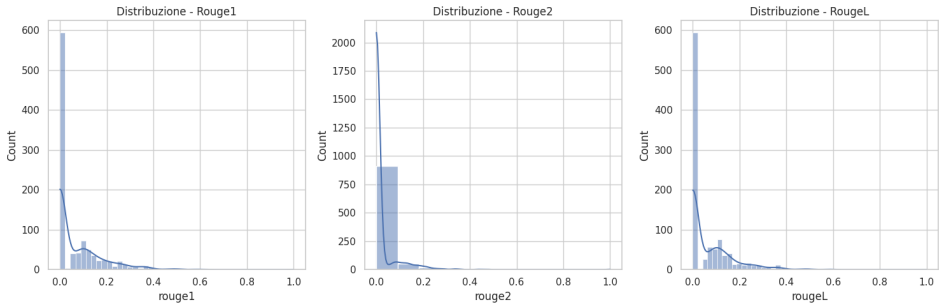

Figure 4.2: Rouge Score on Bert predictions

As it's shown both in Figure 4.1 and in Figure 4.2, performance are poor for all the three metrics used. A manual inspection of the errors reveals that the summary box is not used properly by the users. In most cases they wrote only the name of the product or few words unrelated to those within the actual review. Moreover, even if the summary box was used correctly, a human user doesn't repeat words within the summary and the corpus. On this premises, it's not possible for extractive systems to perform well on metrics based on co-occurring n-grams.

4.2 Abstractive summarization

4.2.1 BART

In relation to the abstractive summarization, we implemented a model designed by Facebook, pre-trained on the CNN/DailyMail Dataset, that's an English-language collection of over 312,000 news articles from CNN and the Daily Mail journals. To be more precise, BART is a transformer seq2seq with bidirectional encoder and autoregressive decoder, trained by

corrupting the input with an arbitrary noising function and, only afterwards, learning to reconstruct the original text.

Note: in order to obtain summaries as faithful as possible to the original, we imposed on the model to limit the size of the output. As we'll see in Section 4.3, this element will cause partial sentences because it reaches the maximum number of usable tokens.

4.3 Human evaluation

In this section we present explanatory examples of the elements mentioned so far.

Figure 4.3: Examples of summaries

In Figure 4.3.a and 4.3.b is shown that BART model, when severely limited in the output dimension, sometimes interrupts sentences in unusual positions. In fact, in both cases, the generated summary well captures the semantics of the original text, but the phrases ends with a preposition. Figure 4.3.c reveals the misuse of the summary box by the user. This make the comparison with the reference, and a faithful computation of the rouge score impossible.

Lastly, in Figure 4.3.d, it's shown that the two extractive methodologies output different results because of their different architectures.

Conclusions

Starting from the binary classification problem in the case of Amazon fine food reviews, we considered the score and the text properly processed to be suitable for machine learning models. The one that performed best on the test set, according to the AUC metric, was the Logistic classifier in combination with a TF-IDF representation, which was later extended by taking into account also unigrams and bigrams - this was because most of the misclassified reviews contained the word 'not' which could lead to semantic comprehension problems. After this upgrade, the model was positively affected, rising to a higher level than its predecessors. Ideally, if we had more computing power at our disposal, we could have considered more advanced models on which we could do hyperparameter tuning and more precise evaluation by means of cross-validation. In this sense, it would have been useful to be able to select only those reviews whose content remains consistent with the associated score. Secondly, the text summarization procedure was carried out using *Summary* as a reference variable to control the results - which were in any case unsatisfactory. In fact, the extractive summarization procedure yields similar results between the LSA methods and the Bert summarizer, although not optimal as the text of the reviews is often not easily traceable to the reference summary. Finally, using the pre-trained BART model for abstractive summarization, we achieved acceptable results in extracting small synopses. However, the outcomes are constrained both from the high computational demands, preventing us from fine-tuning the model and by the imposed length limit on the output, resulting in some phrases being truncated.