AMAZON FINE FOOD REVIEWS: TEXT CLASSIFICATION AND TOPIC MODELING

Francesco Paolo Luigi Caruso 906416, Samir Doghmi 897358, Davide Prati 845926

Abstract
This project utilizes Text Mining techniques on the Kaggle Amazon Fine Food Reviews dataset, emphasizing two main tasks: classification based on review scores and topic modeling. Employing diverse models for comparative analysis, the objective is to extract valuable insights into customer experiences. The analysis explores into patterns within the dataset, offering an understanding of sentiments and preferences. Through these tasks, the study aims to contribute meaningful perspectives for a comprehensive understanding of the customer journey.

Contents

Introduction

Description of the dataset

The dataset we examined for our project consists of a total of 568,454 reviews on Amazon, collected between October 1999 and October 2012, and related to 74,258 food products.
The dataset, obtained from Kaggle, contains indeed 568,454 instances, for which we have 10 columns, each representing a different feature:

- **Id**: an integer value that serves the purpose of identifying the review
- **ProductID**: alphanumeric code which represents an unique identifier for the product
- **UserID**: alphanumeric code which identifies the user
- **ProfileName**: user-chosen profile name, displayed alongside the review and visible to other users
- **HelpfulnessNumerator**: an integer representing the number of users who found the review useful
- **HelpfulnessDenominator**: an integer representing the number of users who have expressed an opinion, either positive or not, on the review
- **Score**: an integer showing the rating associated with the review, ranging from 1 (the worst) to 5 (the best).
- **Time**: the date (day-month-year) on which the review was written
- **Summary**: a brief summary that anticipates the content of the review
- **Text**: the whole text of the review

Objective

The project's objective was to apply Text Mining techniques to the Kaggle Fine Food Reviews dataset. After phases of data cleaning, data exploration and text preprocessing there follows a phase of then text representation. Subsequently, the two main selected tasks are approached: classification based on the score and topic modeling, using different models in both cases for comparisons. The analysis aims to uncover valuable insights from the dataset, particularly through classification and topic modeling, contributing to a better understanding of the customer experience.

Structure of the report

The report is organized as follows:

1. **Data cleaning and preprocessing**: we addressed issues with duplicate, inconsistent, and missing data, retaining only the relevant information.
2. **Data exploration**: in this section, we present some graphs and visualizations that provide an overview of the construction and characteristics of the dataframe
3. **Text preprocessing**: we handled contractions, applied normalization, removed stop words and corrected the spelling. Then we used lemming and stemming.
4. **Sampling**: we made two samplings, one with five classes (multiclass) and one with two classes (binary).
5. **Text classification**: we addressed the first task: classifying reviews initially based on their scores and then in a binary manner between negative and positive ones. We will present various models that we will compare with each other.
6. **Topic modeling**: we used the Latent Semantic Analysis (LSA) and Latent Dirichlet Allocation (LDA) algorithms as well as the BERTopic model. The comparison across these approaches is conducted using the metrics U_mass and C_v coherence. To improve interpretation, we represented the results visually through various figures.

1. Data cleaning and preprocessing

Before conducting any data analysis, we performed preliminary cleaning steps. Specifically, we proceeded following the steps outlined below:

1. **Handling duplicated rows**: in order to obtain a dataset with unique reviews and make the reviews more reliable, we decided to remove all rows that are duplicates based on the attributes ['UserId', 'Time', 'ProfileName', 'Text']. The removal process resulted in a reduction in size from 568,454 to 393,577 records, removing approximately 30.75% of the original rows.
2. **Handling inconsistent rows**: we did a consistency check on the columns regarding helpfulness score. If its numerator (i.e. amount of users who found the review helpful) was greater than its denominator (the amount of users who indicated whether they found the review helpful or not) the row was considered not consistent so we proceeded by dropping it. By doing this, two additional rows are eliminated.
3. **Keeping most recent reviews**: we kept only the most recent review for each combination ['UserId', 'ProfileName', 'Text'], discarding the others.
4. **Handling missing values**: we got a very small percentage of null values only in ProfileName column (0.0028%) and Summary column (0.0008%), but, since we will focus on the Text Column, we didn't need to make any change based on the missing values.
5. **Feature selection / considering only attributes of interest**: finally, we dropped the columns unuseful for our objective, which are: 'ProductId', 'UserId', 'Time', 'ProfileName', 'Id', 'HelpfulnessNumerator', 'HelpfulnessDenominator' and 'Summary'.

2. Data exploration

Once the entire data cleaning and preprocessing process is completed, we can proceed with a brief exploratory phase where we visualize some significant features of our now clean and processed dataframe. Specifically, we will focus on the features that are clearly most relevant for our analysis, which are Text and Score.
Let's start by reporting the number of reviews present in the dataframe for each distinct Score. It's worth noting that reviews with the maximum score are the clear majority, surpassing all other scores combined. It is correct to assume that the proportions would have been substantially the same even before data processing, as there is no evidence of a trend in the removal of instances related to a specific score or a certain subset of scores rather than others.

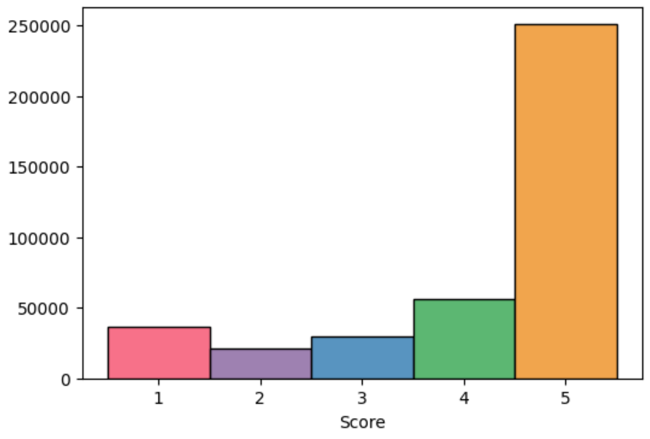


Figure 1. *Amount of reviews for each score*

Another interesting analysis involves the length of reviews in terms of the number of words, categorized according to their associated scores.
To develop a visualization highlighting this characteristic, we decided to create 5 violin plots, one for each score, illustrating the distribution of review lengths. To achieve this, we excluded clearly anomalous cases of reviews that were outliers in terms of the number of words, simply removing reviews with a word count greater than the corresponding 99.9th percentile for this analysis.

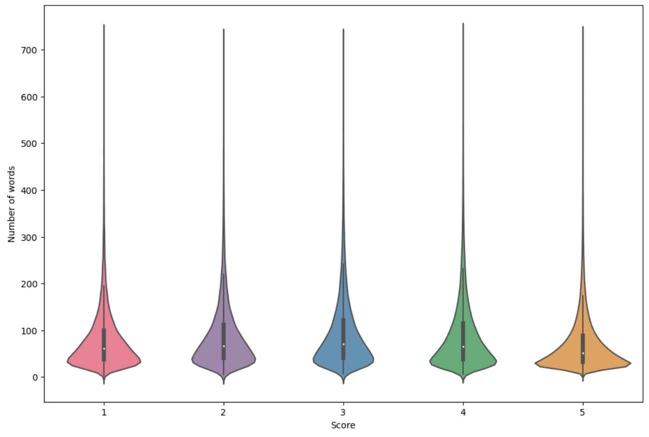


Figure 2. *Distribution of review lengths (up to the 99.9th percentile) for each score*

As we can observe, the distribution is not identical for each score. While all are clearly skewed towards shorter lengths, the distribution for Score 1 (the lowest) and especially Score 5 (the highest) is notably more compressed than the others. This likely occurs because users tend to be more concise when strongly disapproving or wholeheartedly endorsing a product, while they may prefer to provide more extensive reasoning for opinions that fall in intermediate ranges.

3. Text preprocessing

The text preprocessing phase consisted of the following steps:

3.1 Text normalization and tokenization

In order

- Convert text to lowercase
- Remove HTML tags
- Remove numerical digits
- Remove URLs and links
- Use of contractions.fix (from contractions package) to handle contractions
- Expansion of contractions of the words “USA” and “Walmart”
- Remove special characters and punctuations, leaving only alphabetic characters
- Removes leading and trailing white space

We applied the normalization to the column of the dataset with the text and stored the result in a new column.
Also we used NLTK tokenization in order to create the tokens.

3.2 Stop-words removal

We used NLTK package [2] for removing the stop words and we added the words ’would’, ’amazon’, ’product’ due to their high frequency, while excluding the word ’not’ from the list of stop words since we need to distinguish negative from positive reviews.

3.3 Spelling corrections of words

We adjusted the spelling of some words that we believed would improve the information content of the reviews during the later stages.
Given that English words typically have a maximum of two repeated letters, the objective was to correct words with more than three consecutive identical letters.
After identifying words with more than three repeated consecutive characters, a function was employed to reduce occurrences to two. Then it was checked whether the reduced words were valid English words present in the NLTK English dictionary. If a correct word was found, the original value was replaced with the correct word.
For the rest, in order to avoid removing commonly occurring words, we examined their frequency: if it was greater than 15, a manual correction was made using the most probable term, given that the number of distinct words was not very high. All other words were eliminated as they were deemed less relevant.

3.4 Stemmatization and lemmatization

In this phase, two new columns were created within the dataset: one with text to which lemmatization was applied to the words, and another with text to which stemming was applied to the words.

Lemmatization is the reduction of words to their base form (lemma). That transformation is made possible through the use of WordNet [3], an English lexical database.
Stemming allows a word to be brought back to its root. There is a risk of losing information about the meaning of words, while it offers a gain in terms of memory. The specific stemming technique employed is the Porter Stemmer [4].

It was chosen to proceed further only with the text to which lemmatization has been applied. This decision was based on the consideration that lemmatization is a milder approach, providing greater accuracy and comprehensiveness.

3.5 WordClouds

After all the text preprocessing, we decided to proceed with a presentation of some WordClouds to gain an overview of the main themes within the dataset.
First of all, we present a general WordCloud that refers to the whole preprocessed dataset:

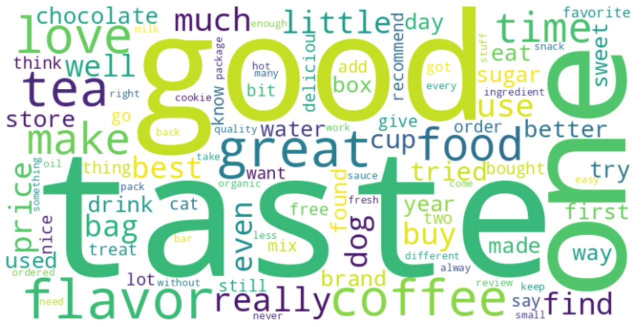


Figure 3. *WordCloud of the whole cleaned and preprocessed dataset*

We can also focus on the most frequent words in negative and positive reviews, i.e. the ones with score 1 and the ones with score 5, highlighting the differences between them.

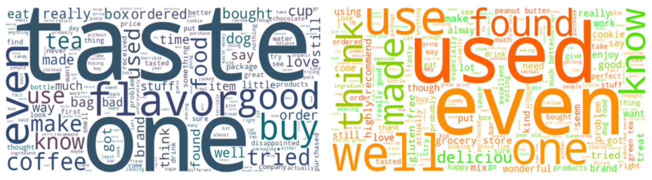


Figure 4. *WordCloud of score 1 vs WordCloud of score 5*

4. Sampling

Before working with models on the dataset, observing its descriptive statistics, it becomes apparent that there is a highly imbalanced distribution of scores: in particular, concerning the maximum score, it appears in significantly higher numbers than the others (even more than 10 times the occurrences of score 2). Therefore, to have an unbiased dataset, sampling was performed in two different ways.

4.1 Multiclass sampling

We applied a downsampling approach while keeping the original 5 classes. The sampling cardinality was set equal to the number of rows with a score of 2 (which is the score with the smallest amount of data), therefore equal to 20 796.
The result is a dataset which consists of 103 980 words, distributed in an exactly equal manner for each score.

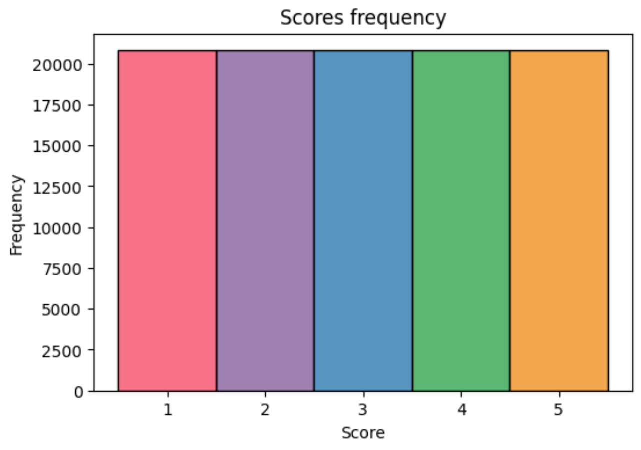


Figure 5. *Dataset downsampled while retaining the 5 classes*

4.2 Binary sampling

We employed a downsampling method considering two classes, creating a binary classification where scores 1 and 2 are labeled as 'negative', and scores 4 and 5 are labeled as 'positive', while records with score 3 is not considered. The sampling cardinality for each score is set equal to the number of rows with a score of 2, which is 20 796. Therefore, the number of records in each class is 41 592.
In this way, the final dataset has a number of records equal to 83 184.

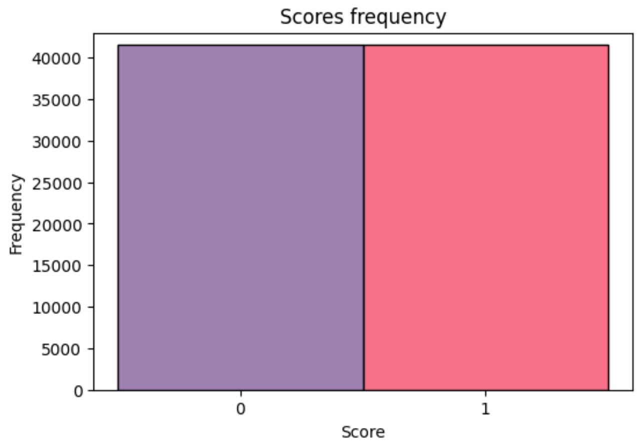


Figure 6. *Dataset downsampled considering a binary classification problem*

5. Text classification

Text classification, the first of the two tasks we aim to develop, is defined as the activity of predicting to which class, within a predefined and limited set, various instances of the dataframe under consideration belong.
Clearly, the membership of an object to a particular class cannot be determined with absolute certainty, and for this reason, parameters will come into play to evaluate the "correctness"

of our predictions.
Therefore, for each of the various machine learning models we will use to classify our reviews, we will proceed to evaluate their effectiveness using the following parameters: accuracy, precision, recall, and F_1 measure, in addition to the classic confusion matrix.
To provide a brief overview of these concepts, given the confusion matrix

		PREDICTION	
		-1	+1
ACTUAL CLASS	-1	TN	FP
	+1	FN	TP

Figure 7. Confusion Matrix

where TP and TN stand for the instances correctly classified, respectively as belonging to positive and negative class, while FP and FN indicate the amount of instances wrongly classified as positive and negative, we can define accuracy, precision, recall and F_1 measure as

$$\begin{aligned} \text{accuracy} &= \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \\ \text{recall} &= \text{r} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \\ \text{precision} &= \text{p} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \\ \text{F}_1 &= \frac{2 \cdot \text{r} \cdot \text{p}}{\text{r} + \text{p}}. \end{aligned}$$

As for the classifications we will perform, we first proceed with a multiclass classification using the first sampling conducted in section 4, where we essentially associate each review with its original score. Subsequently, we move on to a binary classification, this time utilizing the second sampling, where we classify the reviews as negative (0) and positive (1).
The first classification follows the schema $f_1 : D \to C$, where $C = \{1,2,3,4,5\}$, while the second has the same schema ($f_2 : D \to C$), but with $C = \{0,1\}$.

5.1 Text representation and dataset splitting into train and test for classification

Although from this point onward, we will work separately regarding multiclass and binary classification with the two different samplings, the preliminary dataset preparation is common to both. Therefore, we will present it in this section.

To prepare the dataset for compatibility with the machine learning models we will later apply, it is necessary to transform the text into a numerical form. This process will be carried out using two alternative and distinct approaches: bag of words (BOW) and term frequency–inverse document frequency (TF-IDF).
As for the bag of words approach, this strategy involves counting the occurrences of each individual word in each preprocessed document. Subsequently, these counts are placed in a large document-term matrix, where the rows represent the documents and the columns represent the words. This matrix will inevitably be excessively sparse, given that many entries will be zero due to the absence of certain words in specific documents.
The second approach, TF-IDF, is based on two concepts: again, the term frequency ($\text{tf}_{\text{d,t}}$) of a term t in a document d, which is the count of the times t appears in d, and the inverse document frequency (idf) of t, defined as the logarithm of the ratio between the total number of documents N and the number of documents containing t, denoted as df_{t}. Based on these two concepts, we then create a new matrix that associates documents with words. In this case, the weight of each word in a document is defined as the product of its term frequency and the corresponding inverse document frequency. However, even with this more complex and elaborate approach, the result will still be a sparse matrix.

Therefore, in both cases, to address the presence of sparse matrices with a large number of features (one for each word in the dataset), we need to apply dimensionality reduction techniques before using the models. These techniques, while compromising information quality to some extent, allow for more efficient data processing. In particular, we employ a truncated singular value decomposition (SVD).
Clearly, the goal of this approach is to strike a good balance between the number of features, which we want to keep as low as possible, and the quality of the data. It is crucial not to reduce the dimensions excessively. We can measure the quality through a parameter called explained variance, which quantifies the fraction of variance in the original data captured by the first k principal components retained. The higher the explained variance, the greater the amount of information retained during dimensionality reduction.
By operating with k=400 in the case of the bag of words approach, we achieve an explained variance of 0.657. It's worth noting that if we were to use the same number of features in the TF-IDF approach, we would obtain a much lower explained variance. Therefore, we increase the features to k=1000 in the TF-IDF approach, resulting in an explained variance of 0.569. Clearly, with a higher k, we could approach the variance value obtained in the bag of words approach, but the computational cost of training models would significantly increase. Nonetheless, future developments in this project could certainly take this factor into consideration.

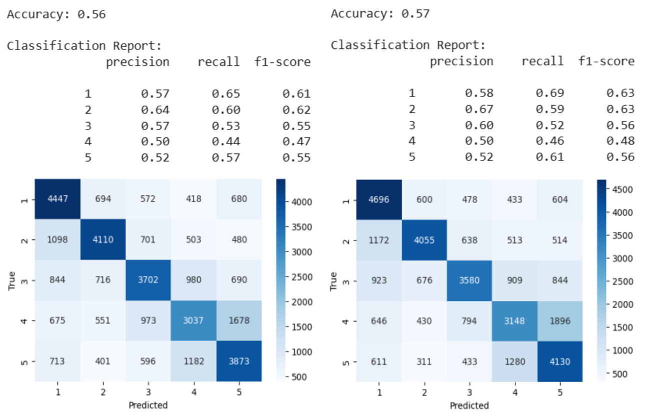
Finally, the division between the training and test sets is performed randomly using the conventional percentage ratios of $2/3 - 1/3$ for both the text processed with the bag of words and that which has undergone the TF-IDF approach.

5.2 Multiclass classification

For multiclass classification, we have chosen to use a tree-based model, which is Random Forest, a probabilistic model, which we chose to be logistic regression, and a boosting model, XGBoost, all taken from the scikit-learn library, with the exception of XGBoost.
All these models, as in the case of binary classification, will be trained on the text that has undergone lemmatization, and not on the one where stemming has been applied for the reasons explained in section 3.

5.2.1 Random Forest

A Random Forest is a machine learning model based on an ensemble of decision trees. During training, multiple decision trees are created on random subsets of the training data, and the results of these different decision trees are then combined to obtain the best possible prediction. It is designed to significantly reduce overfitting.



```
Accuracy: 0.56

Classification Report:
              precision    recall  f1-score

           1       0.57      0.65      0.61
           2       0.64      0.60      0.62
           3       0.57      0.53      0.55
           4       0.50      0.44      0.47
           5       0.52      0.57      0.55
```

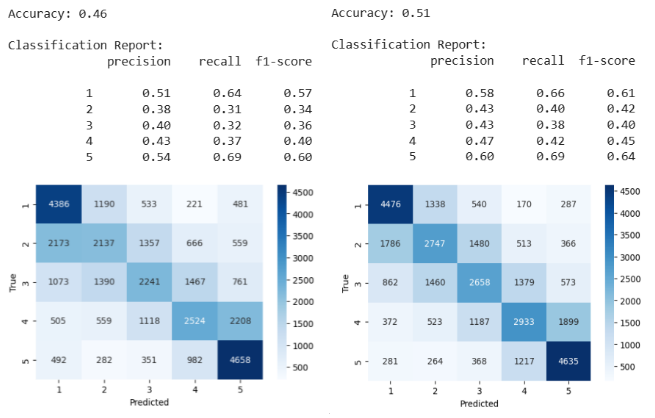
```
Accuracy: 0.57

Classification Report:
              precision    recall  f1-score

           1       0.58      0.69      0.63
           2       0.67      0.59      0.63
           3       0.60      0.52      0.56
           4       0.50      0.46      0.48
           5       0.52      0.61      0.56
```

Figure 8. *Random Forest results, for BOW and TF-IDF respectively*

As we can observe, the accuracy shows a slight improvement when transitioning from text represented with the BOW approach to that represented with the TF-IDF one. This will be a trend that we will consistently observe in every model, whether applied for multiclass or binary classification.

5.2.2 Logistic Regression

The logistic regression model is a probabilistic model, primarily designed for binary classification but extendable and adaptable for multiclass classification tasks. In this project, we will present it in a multiclass context, and indeed, we will see how the results will prove to be considerably worse compared to its ideal binary classification version.
Logistic regression's performance tends to degrade in multiclass scenarios, where other models like those designed specifically for multiclass classification will be more suitable in such cases.



```
Accuracy: 0.46

Classification Report:
              precision    recall  f1-score

           1       0.51      0.64      0.57
           2       0.38      0.31      0.34
           3       0.40      0.32      0.36
           4       0.43      0.37      0.40
           5       0.54      0.69      0.60
```

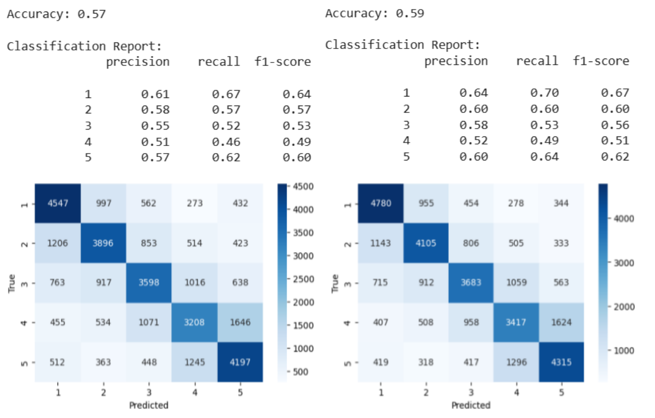
```
Accuracy: 0.51

Classification Report:
              precision    recall  f1-score

           1       0.58      0.66      0.61
           2       0.43      0.40      0.42
           3       0.43      0.38      0.40
           4       0.47      0.42      0.45
           5       0.60      0.69      0.64
```

Figure 9. *Logistic regression results, for BOW and TF-IDF respectively*

In this case we can observe that there is an increase in accuracy, although it still remains considerably lower compared to the previous case. From the reports and confusion matrices, it is noticeable that the results drop significantly, especially for the intermediate classes, namely, class 2, 3, and 4. This suggests that the logistic regression model faces challenges in accurately distinguishing between multiple classes, particularly those that are not at the extremes. It underscores the importance of considering models specifically designed for multiclass classification tasks.

5.2.3 XGBoost

XGBoost, which stands for "Extreme Gradient Boosting," is a model belonging to the category of gradient boosting algorithms. It is based on the boosting approach, which combines a series of weak models (typically shallow decision trees) to create a more robust and predictive model. XGBoost is designed to be highly scalable and, like logistic regression, is intended for binary classification. We will observe that the results are significantly better when the model is applied to the dataset sampled with binary approach.



```
Accuracy: 0.57

Classification Report:
              precision    recall  f1-score

           1       0.61      0.67      0.64
           2       0.58      0.57      0.57
           3       0.55      0.52      0.53
           4       0.51      0.46      0.49
           5       0.57      0.62      0.60
```

```
Accuracy: 0.59

Classification Report:
              precision    recall  f1-score

           1       0.64      0.70      0.67
           2       0.60      0.60      0.60
           3       0.58      0.53      0.56
           4       0.52      0.49      0.51
           5       0.60      0.64      0.62
```

Figure 10. *XGBoost results, for BOW and TF-IDF respectively*

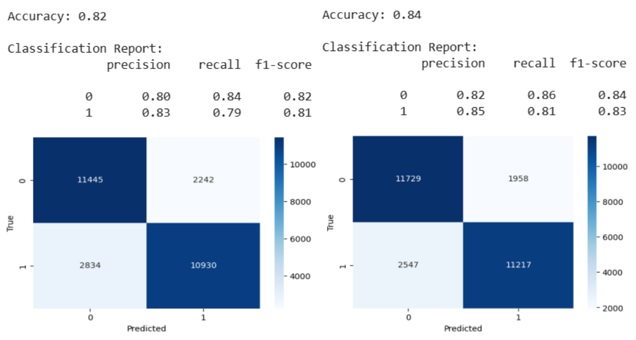
We can observe that the accuracy once again shows a slight

improvement when transitioning from the Bag of Words approach to the TF-IDF approach, similar to the situation with all the other models.
Additionally, this algorithm proves to be the best, in terms of accuracy, among those presented so far.

5.3 Binary classification

Let's now focus on binary classification. In this case, as mentioned earlier, we will work again with lemmatized text and use the same methods employed for multiclass classification, adding just one more - the Support Vector Machine (SVM), a linear method.
SVM works finding an optimal hyperplane that separates instances of the two classes in the most effective way possible. The optimal hyperplane is identified by maximizing the margin between the training instances of the two classes. The margin is defined as the distance between the hyperplane and the closest point of each class, known as support vectors. The choice of the optimal hyperplane is based on maximizing this margin.
SVM has not been implemented in the multiclass case because, although it also has a multiclass extension, even if its original scope regards binary classification, its effectiveness in this context is generally lower compared to logistic regression or XGBoost. Additionally, SVM involves significantly higher computational costs than all the other models we implemented, so it was preferred to apply it only in its ideal binary classification context.

5.3.1 Random Forest



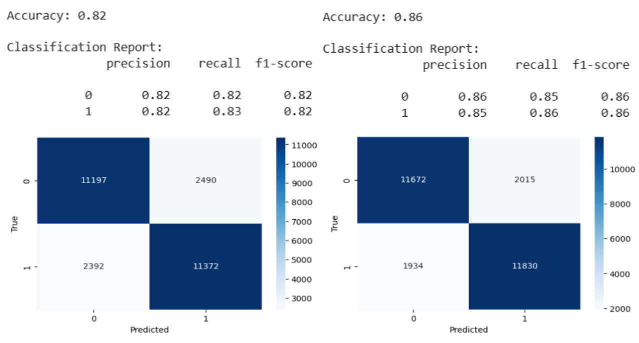
```
Accuracy: 0.82

Classification Report:
              precision    recall  f1-score

           0       0.80      0.84      0.82
           1       0.83      0.79      0.81
```

```
Accuracy: 0.84

Classification Report:
              precision    recall  f1-score

           0       0.82      0.86      0.84
           1       0.85      0.81      0.83
```

Figure 11. *Random Forest results, for BOW and TF-IDF respectively*

As we can see, in the binary classification case, the Random Forest model achieves significantly better accuracy results compared to the multiclass scenario. Once again, there is an improvement when using the TF-IDF approach compared to the Bag of Words.
These consistently high results will be observed similarly for all models.

5.3.2 Logistic Regression



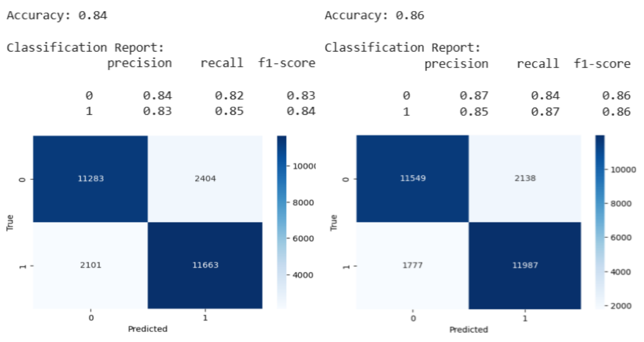
```
Accuracy: 0.82

Classification Report:
              precision    recall  f1-score

           0       0.82      0.82      0.82
           1       0.82      0.83      0.82
```

```
Accuracy: 0.86

Classification Report:
              precision    recall  f1-score

           0       0.86      0.85      0.86
           1       0.85      0.86      0.86
```

Figure 12. *Logistic regression results, for BOW and TF-IDF respectively*

In the case of logistic regression, the improvement is even more evident, considering that the results in binary classification are similar to those obtained with the Random Forest model, but the results in multiclass classification were significantly worse. All of that is clearly consistent with the fact that logistic regression is originally designed to work with only two classes.
It's worth noting that, while in the case of Random Forest the True Negatives (TN) were predicted slightly better, in this case, the ones predicted slightly better are the True Positives (TP).

5.3.3 XGBoost



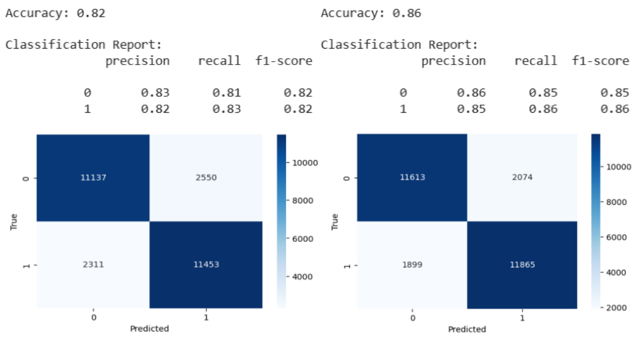
```
Accuracy: 0.84

Classification Report:
              precision    recall  f1-score

           0       0.84      0.82      0.83
           1       0.83      0.85      0.84
```

```
Accuracy: 0.86

Classification Report:
              precision    recall  f1-score

           0       0.87      0.84      0.86
           1       0.85      0.87      0.86
```

Figure 13. *XGBoost results, for BOW and TF-IDF respectively*

The XGBoost model exhibits the best results achieved with the Bag of Words approach among all the models we trained. Nevertheless, the results with TF-IDF are still superior. In this case, similar to logistic regression, the True Positives are predicted slightly better.

5.3.4 SVM



```
Accuracy: 0.82

Classification Report:
              precision    recall  f1-score

           0       0.83      0.81      0.82
           1       0.82      0.83      0.82
```

```
Accuracy: 0.86

Classification Report:
              precision    recall  f1-score

           0       0.86      0.85      0.85
           1       0.85      0.86      0.86
```

Figure 14. *SVM results, for BOW and TF-IDF respectively*

Regarding SVM, in this case, we do not have a direct comparison with the multiclass model. However, we can still observe that the accuracies are very high and improve when transitioning from BOW to TF-IDF. Similarly, the True Positives (TP) are predicted slightly better than the True Negatives (TN).

In general, we can conclude that there is no clear preference in the choice of the model in the case of binary classification, while this preference is more pronounced and evident in the case of multiclass classification.

6. Topic modeling

Topic modelling is an unsupervised machine learning technique that aims at automatically extract word groups that best characterize a set of documents. It provides topics, which are interpreted as a set of words that makes sense together.
The fundamental concept involves clustering words into distinct groups, where:

- Each word in the cluster is likely to occur "more" (have a probability of occurrence) for the given topic;
- Different topics have their respective clusters of words along with corresponding probabilities;
- Different topics may share some words and a document can have more than one topic associated with it.

In this study, the multiclass dataset was used as the input for the Latent Semantic Analysis (LSA) and Latent Dirichlet Allocation (LDA) algorithms, both implemented using the gensim library.Additionally, we incorporated a pre-trained transformer-based model known as BERTopic into our analysis.
The determination of the number of topics was assessed through the metrics `U_mass` and `C_v` coherence for both LDA and LSA. The `U_mass` measures the degree of coherence of the topics by assessing the semantic similarity between words within the same topic: a lower value (close to zero) indicates greater coherence. While `C_v` measures the coherence of arguments by assessing the pairwise similarity between the main terms of an argument. Thus, a higher value indicates more interpretable topics.
In addition to quantitative metrics, our assessment of model performance incorporated qualitative measures, such as human judgment. Given that these metrics did not have a strong correlation with human interpretability [5], we improved our analysis by visualising the topic models using Worldclouds and different visual figures. This made it much easier to examine in detail the top terms most closely associated with each topic, providing a deeper understanding of the content and context of the topics.

6.1 Text representation for LDA and LSA models

To represent the text, the Bag of Word approach was used starting with the construction of bigrams and trigrams to capture more intricate relationships and structures among words in a text than individual words. Bigrams were defined as pairs of words that occurred at least 50 times in the corpus, with a normalized pointwise mutual information (npmi) score of at least 30%. Trigrams, on the other hand, were composed of words that appeared at least 20 times with an npmi score of 0.3%. Once identified, they were added to the corpus, as we wanted to retain the words 'baby' and 'shower' and the bigram 'baby_shower'. A total of 5055 bigrams and 319 trigrams were identified. Then, for the implementation of LDA and LSA algorithms we created a dictionary and and filtered it by excluding words that appear in less than 10 documents or in more than 0.3% of the total number of documents. As a result, a dictionary of 15181 words was obtained.
Thus, in constructing the LDA model, we employed a bag-of-words corpus transformation, following the example of gensim [6]. This is consistent with the observation that LDA works optimally when a raw count of word frequencies is presented. In contrast, for LSA, we opted for the TF-IDF matrix [7].

6.2 LDA

Latent Dirichlet allocation (LDA) is a generative statistical model that allows us to discover hidden topics from a collection of documents. Each document is assumed to contain a mixture of these hidden topics, and each topic is assumed to generate words according to its own probability distribution. The goal is to identify a set of topics that are most likely to generate the observed document data.
In a standard LDA model, there are a few key parameters to consider: the document-topic density (α) and topic-word density (β) parameters. When set to 'auto', the model automatically learns the best values for the hyperparameters as it is trained on more and more data, while the number of topics was determined iteratively by testing models with different topic numbers and comparing the resulting coherence metrics.

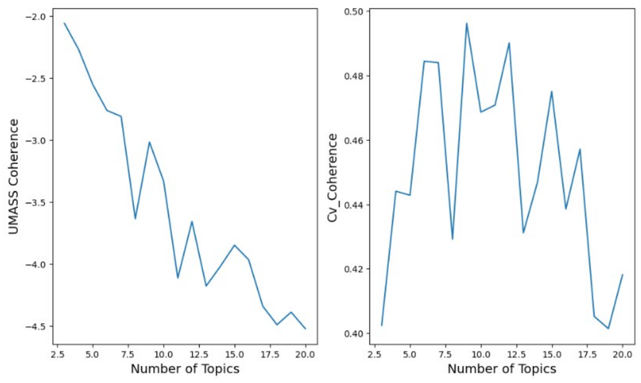


Figure 15. *Comparison of Cv and U_mass Coherence trough different LDA model topics*

Considering the quantitative metrics, a trade-off between the two values can be observed with the 9-topic model, where a Cv_Coherence of 0.50 and a U_mass Coherence of -3.02 are achieved.
To visualise the result, we generated WordCloud based on the word weights for each topic. The weights were computed using the γ parameter, set to 0.6, which is used to balance two factors: the probability of a term within a topic against its overall probability in the corpus and the overall probability of the term in the corpus in the pyLDAvis library

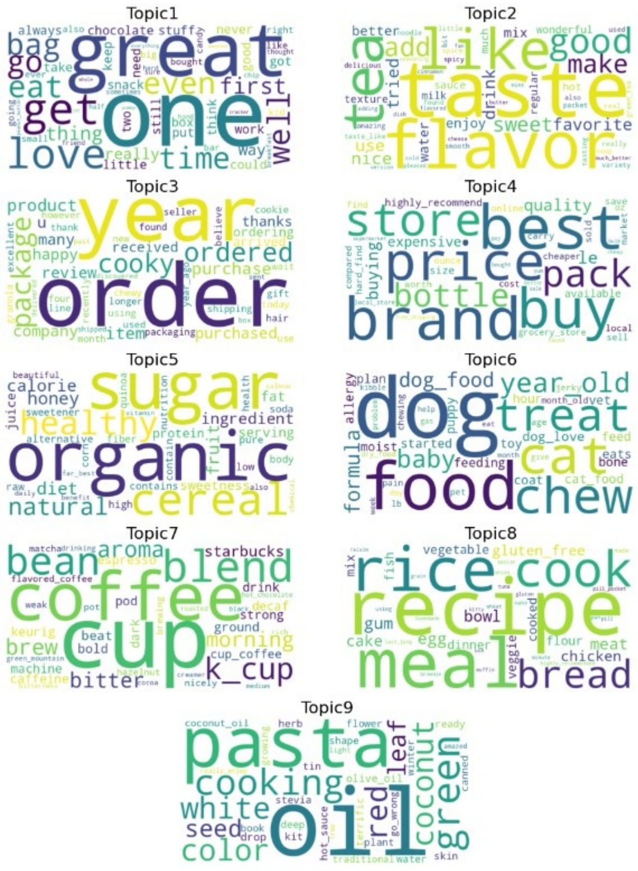


Figure 16. *Wordcloud of topics identified by the LDA model*

Following, we have labeled the word clouds according to our understanding, with two topics remaining uninterpretable:

1. Topic1: **Uninterpretable**: great, one, love, bag, time, good.
2. Topic2: **Uninterpretable**: taste, flavour, use, drink, tea, add.
3. Topic3: **Packaging**: order,package, product, purchase, item.
4. Topic4: **Product characteristics**: price, buy, brand, store, highly_recommend, available, cost.
5. Topic5: **Healthy**: sugar, organic, cereal, natural, calorie, low, alternative.
6. Topic6: **Animal care**:food, dog, treat, cat, dog_food, dog_love.
7. Topic7: **Coffee**: coffee, cup, k_cup, morning, bitter.
8. Topic8: **Meal**: recipe, meal, rice, bread, chicken, meat, bowl.
9. Topic9: **Sauces, Oils and Herbs**: oil, green, water, seed, red.

6.3 LSA

Latent Semantic Analysis (LSA) is a statistical model of word usage that permits comparisons of semantic similarity between pieces of textual information.The primary assumption of LSA is that there is an underlying, or 'latent', structure in the pattern of word usage in documents and that statistical techniques can be used to estimate this latent structure.
Therefore, LSA utilizes Singular Value Decomposition (SVD) to transform the document-terminal matrix derived from the TF-IDF frequency computations into a low-dimensional space to discover latent topics. The result of SVD is a k-dimensional vector space containing a vector for each term and each document.The location of term vectors reflects the correlations in their usage across documents, while the location of document vectors reflects correlations in the terms used in the documents according to their semantic similarity.
In our work, as with the Lsa model of the gensim library, a truncated SVD is applied, while the number of arguments must be specified a priori. We proceeded step by step, creating models with different numbers of topics and testing their performance with two measurements.

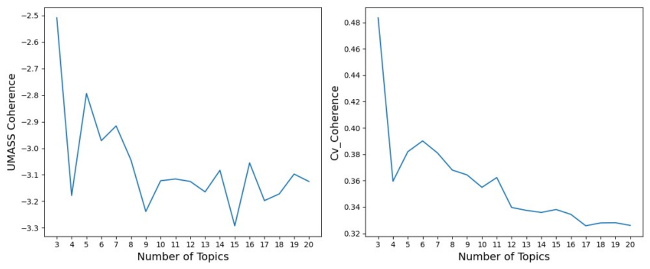


Figure 17. *Comparison of Cv and U_mass Coherence trough different LSA model topics*

After reviewing the graphs, we chose the 6-topic model, which showed a Cv_Coherence of 0.39 and a U_mass Coherence of -2.97.

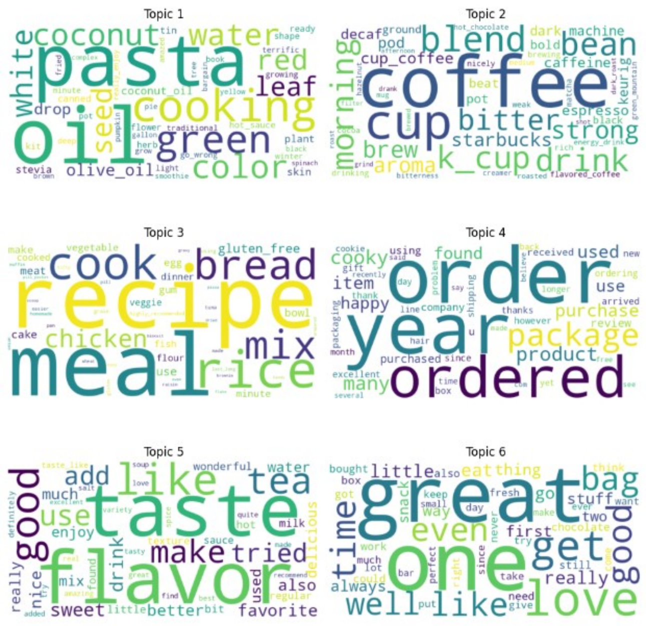


Figure 18. *Wordcloud of topics identified by the LSA model*

By looking at the figure, we labelled the word clouds based on our own comprehension, while two topics remained undecidable:

1. Topic1: **Sauces, Oils and Herbs**: oil, green, water, seed,red.
2. Topic2: **Coffee**: coffee, cup, k_cup, morning, bitter.
3. Topic3: **Meal**: recipe, meal, rice, bread, chicken, meat, bowl.
4. Topic4: **Packaging**: order,package, product, purchase, item.
5. Topic5: **Uninterpretable**: taste, flavour, use, drink, tea, add.
6. Topic6: **Uninterpretable**: great, one, love, bag, time, good.

Nevertheless, LSA model exhibited lower performance compared to LDA model.

6.4 BERTopic

BERTopic[8], developed in 2020 by Grootendorst, is a algorithm designed to generate document embedding with pre-trained transformer-based language models. Then, it proceeds to cluster these embeddings and finally generates topic representations through topic representations with the class-based TF-IDF procedure.
The process of generating topic representations with BERTopic involves three key steps. Firstly, each document is converted to its embedding representation using a pre-trained language model—specifically, in this instance, the default embedding model of all-MiniLM-L6-v2 is employed. Then, the dimensionality of the resulting embeddings is reduced to optimize the clustering process through UMAP method. The resulting reduced embeddings are then subjected to clustering using HDBSCAN. This prevents unrelated documents to be assigned to any cluster and is expected to improve topic representations. Lastly, TF-IDF procedure models the importance of words in clusters instead of individual documents.This allows us to generate topic-word distributions for each cluster of documents.

After training the model with our corpus, we identified 29 different topics. We generated a similarity matrix using cosine similarity to visualise the results. It was observed that some topics show associations, such as the topics bag_wine_box and mashroom_garlic_tomato, which have a similarity score of 0.76.

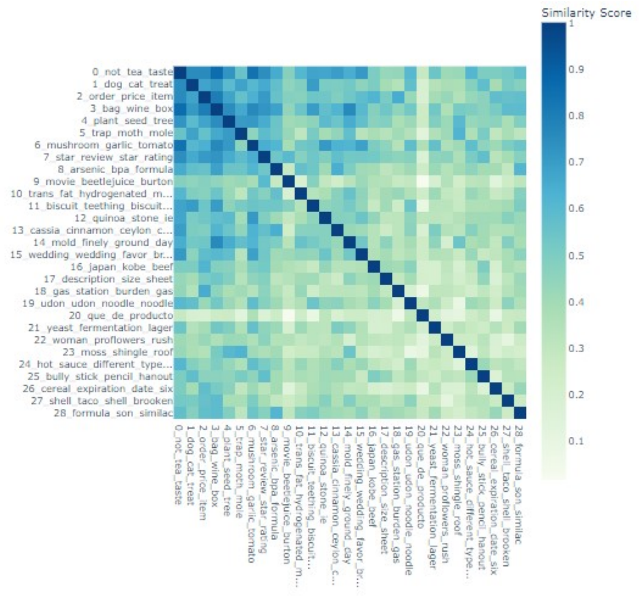


Figure 19. *Similarity Matrix*

Consequently, we decided to reduce the number of topics to 7, choosing this value randomly, resulting in a maximum similarity score of 0.53. The coherence measure Cv_coherence and U_mass coherence for the 7-topic model were 0.61 and -0.43, respectively.

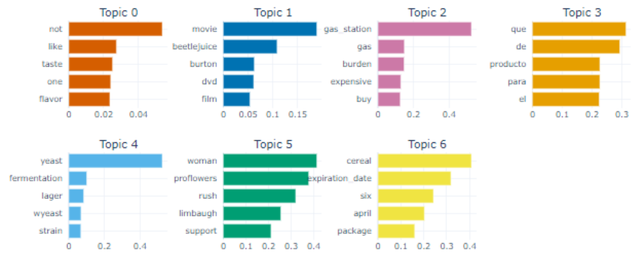


Figure 20. *Topic Word Score*

The figure above shows the most important words for each topic. In particular, the topics identified appear more granular and context-specific than those identified by other models. This improvement is attributed to the use of the embeddings generated by BERT, which capture more contextual information than the statistical co-occurrence matrix of bag-of-words (BoW) and TF-IDF representations.
We then tried to assign tags to each topic, resulting in the following potential labels:

1. Topic0: **Uninterpretable**: not, like, taste, one, flavor, good, tea, dog, get, food
2. Topic1: **Movies**: movie, beetlejuice, burton, dvd, film, ghost.
3. Topic2: **Fuel**: gas_station, gas, burden, expensive, buy, struggling.
4. Topic3: **General Terms in Spanish**: que, de, producto, para, el, muy.
5. Topic4: **Fermentation**: yeast, fermentation, lager, wyeast, strain, malt, cider.
6. Topic5: **Uninterpretable**: woman, proflowers, rush, limbaugh,upport, show, flower.
7. Topic6: **Cereal Packaging and Expiry Information**: cereal, expiration_date, six, april, package, teared, box, mesa, sunrise.

Conclusions and future developments

Regarding classification, a clear conclusion is that the TF-IDF approach outperforms the Bag of Words (BOW) approach in terms of accuracy in every case, making it the preferable choice. This result becomes even more significant considering that the explained variance obtained with the TF-IDF approach is lower than that of BOW.
Focusing on the models, in the multiclass case there are quite distinct differences in results, with XGBoost outperforming the others, as expected, given that the other models are not optimized for multiclass classification.
However, in the binary case, the models essentially perform similarly, or at least, there is no model with significantly better results than the others. This aligns with the “no free lunch principle”, which asserts that no machine learning algorithm can consistently outperform others in all contexts.
Future developments could certainly involve increasing the number of selected features with Singular Value Decomposition (SVD) to enhance explained variance. Additionally, training the models on data represented with BOW incorporating bigrams or trigrams could be explored.

Regarding topic modeling, we observed that BerTopic identifies more detailed and context specific topics by using the embeddings generated by Bert. This approach has led to achieving the highest cv_coherence, but not the in case of U_mass coherence. Therefore, we can examine different embeddings representations for further development, as they can capture more information than bag-of-words and Tf-Idf representations.
When evaluating the optimal number of topics, as there is no definitive state-of-the-art metric, we can explore different alternatives and fine-tune the hyper-parameters of each model to optimize the results.

References

[1] G. Pasi and M. Viviani. Text Mining and search course lecture notes and slides, a.a. 2023/2024.

[2] NLTK:
https://www.nltk.org/

[3] WordNet:
https://wordnet.princeton.edu/

[4] Porter Stemmer:
https://www.nltk.org/_modules/nltk/stem/porter.html

[5] Ismail Harrando, Pasquale Lisena and Raphael Troncy.Apples to Apples: A Systematic Evaluation of Topic Models:
https://aclanthology.org/2021.ranlp-1.55.pdf

[6] LDA gensim:
https://radimrehurek.com/gensim/auto_examples/tutorials/run_lda.html

[7] LSA gensim:
https://radimrehurek.com/gensim/auto_examples/core/run_topics_and_transformations.html

[8] BerTopic:
https://maartengr.github.io/BERTopic/index.html