Amazon Fine Food Reviews: Classification and Clustering

Gianluca Cavallaro (matr. 826049)[1], Remo Marconzini (matr. 883256)[2]

Abstract
The project aims to analyze the *Amazon Fine Food Reviews* dataset using different Text Mining techniques. Initially, an exploratory analysis of the data is performed, followed by some preprocessing activities. Then, different classification and clustering models are implemented, to classify reviews starting from their text and grouping similar reviews.

Keywords
Text Mining – Text Representation – Text Classification – Text Clustering

[1,2] *Dipartimento di Informatica, Sistemistica e Comunicazione, Università degli studi di Milano-Bicocca, Milano, Italia*

Contents

1. Introduction

Online shopping is an increasingly common practice, especially after the pandemic, which has led to the use of the Internet for everyday activities. When making an online purchase, one of the most important aspects is the reviews left by previous

customers. A new customer, according to the indications left by those who have already bought a product, can be guided during the purchase, identifying the product that best suits him. The leader in the sector is undoubtedly Amazon, the online shopping giant that has extended its offer to a huge range of products and sectors over the years.
In this project, a subset of these products is analyzed, specifically those falling under the *Amazon Fine Foods* group. The dataset[1] contains 568,454 reviews covering the period between October 1999 and October 2012. The information contained in the dataset concerns 256,059 unique users and 74,258 different products. The dataset contains 10 features, including:

- **Id**: unique identifier for the review;
- **ProductId**: unique identifier for the product;
- **UserId**: unique identifier for the user;
- **ProfileName**: user name;
- **HelpfulnessNumerator**: number of users who found the review helpful;
- **HelpfulnessDenominator**: number of users who indicated whether they found the review helpful or not;
- **Score**: rating between 1 and 5;
- **Time**: timestamp for the review;
- **Summary**: brief summary of the review;
- **Text**: text of the review.

2. Research questions

The project aims to answer the following questions:

1. Can a review be classified as *good* or *bad* from its text?
2. Is it possible to predict the user's rating starting from the text of the review?
3. Is it possible to group similar reviews?

3. Data Cleaning

First, the dataset has been adjusted to make it ready for the following analysis. The following operations are performed:

- **Consistency check of the values of the attributes HelpfulnessNumerator and HelpfulnessDenominator**: by definition, the value of **HelpfulnessNumerator** must necessarily be less than or equal to that of **HelpfulnessDenominator**. For each of the reviews contained in the dataset, this property is verified. A number of 2 reviews do not respect the previous inequality. Therefore, they are eliminated;
- **Missing values**: for each of the attributes of the dataset, missing values are checked. The only attributes to have missing values are **ProfileName** and **Summary**. As these are not essential attributes for future analysis, corresponding records are not deleted;
- **Removing duplicates**: during the analysis of the records with missing **Summary** it emerges as such rows are, most likely, duplicates. From a deeper analysis emerges that 174,521 reviews are duplicates of some other review. In fact, they are performed by the same user, at the same time and with the same comment (same values for the attributes **UserId**, textbfTime and textbfText) of some other review in the dataset. We proceed, therefore, with the removal of duplicates.

As a result of previous operations, about 30% of the initial records are removed. The dataset currently has 393,931 reviews.

4. Data Exploration

Below, an in-depth analysis of the dataset is proposed. Given the purpose of the project, the main focus was placed on the attributes **Score** and **Text**.
First, the distribution of reviews was analysed according to the respective value of **Score**:

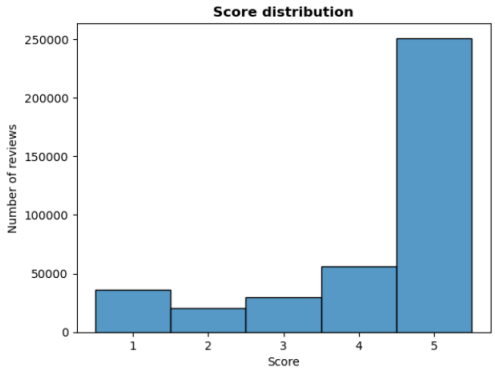

Figure 1. Score distribution

It is possible to notice that most reviews (250,961, 63.71% of the total) have a score of 5. Among others, the class with the lowest number of reviews is 2 (20,802 reviews, 5.3% of the total).
Considering the variable **Text**, instead, we show the distribution of the number of words by value of the variable **Score**:

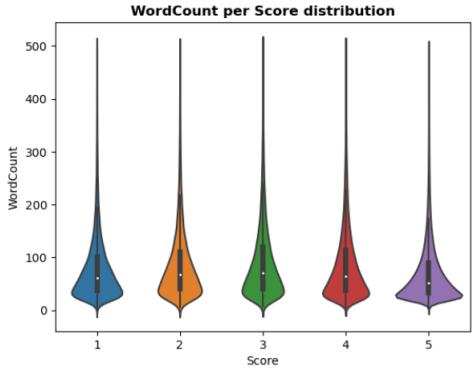

Figure 2. Word count per score value

Only reviews with less than 500 words are considered, to generate a more readable graph. From figure 2 you can see that class 5 has a lower median than the others, which shows that satisfied clients tend not to write long reviews.
Next, the distribution of the number of reviews versus attributes **Time** and **HelpfulnessRatio** is analyzed. **HelpfulnessRatio** was generated by dividing **HelpfulnessNumerator** by **HelpfulnessDenominator**:

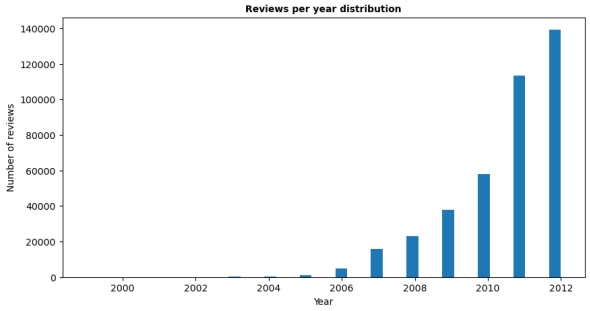

Figure 3. Number of reviews per year

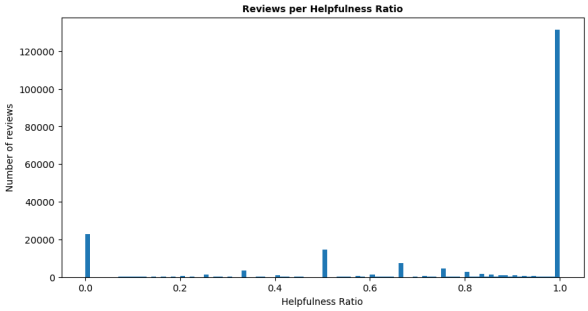

Figure 4. Reviews distribution per HelpfulnessRatio

From figure 3 we can see that the number of reviews has grown from year to year, a sign of the ever-increasing spread of online

commerce, while from figure 4 we can notice that the greatest number of reviews is considered useful by all users who have evaluated them. Interestingly, the second highest number is that of reviews that all users who evaluated them found useless.
As last thing, in figure 5 we show a WordCloud made considering all the reviews of the dataset, so as to show the most recurring words:

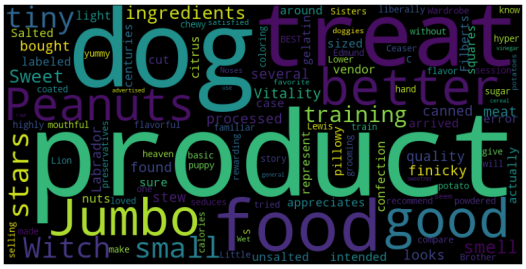

Figure 5. WordCloud for **Text**

5. Preprocessing

In order to carry out the proposed analyses, it is necessary to properly process the text of the reviews, contained in the attribute **Text**. Specifically, the following steps are performed:

- **Normalization**: the process of transforming text into a single canonical form. This phase includes the following operations:
  - Conversion to lowercase letters;
  - Removing alphanumeric characters;
  - Removing emoji, URLs and HTML tags;
  - Removal of duplicate characters and spelling correction;
  - Removal of punctuation and special characters;
- **Tokenization**: an operation that consist in separating a piece of text into smaller units called tokens. Tokens can be either words, characters, or subwords. In our case, tokens are simply words. In fact, the presence of n-grams was not considered;
- **Stop-words removal**: an operation that removes the words that occur commonly across all the documents in the corpus. Typically, articles and pronouns are generally classified as stop words;
- **Lemmatization**: the process of grouping together the inflected forms of a word so they can be analysed as a single item, identified by the word's lemma. Although it is more expensive than stemming, lemmatization is preferred because it generates lemmas based on context and lexical rules. Also, in general, it is a more accurate operation.

6. Text Representation

In order to apply the classification and clustering algorithms, useful to answer the proposed research questions, it is necessary to transform the reviews from unstructured text to a representation that makes them mathematically computable. Among the possible representations, Bag-Of-Words (BOW), TF-IDF and Word2Vec are selected. The same approach and analysis on text representations are applied for both binary and multi-class classification. The results for multi-class classification will be shown later.

6.1 Bag-Of-Words (BOW)

Bag-Of-Words model is used to preprocess the text by converting it into a bag of words, which keeps a count of the total occurrences of most frequently used words.
Following the application of the model, a dimensionality reduction technique was applied in order to reduce the high number of features of the obtained vectors. Specifically, a Singular Value Decomposition (SVD) has been applied. It is important to determine the number of components to use, in order to obtain an optimal trade-off between computational efficiency and

preserving the most important features of the data[2]. One strategy for selecting the number of components is to use cumulative variance analysis. The method of cumulative variance allows to choose the number of components based on a desired explained variance proportion. In our case, we chose to consider a number of components that explain 85% of the total variance of the dataset.

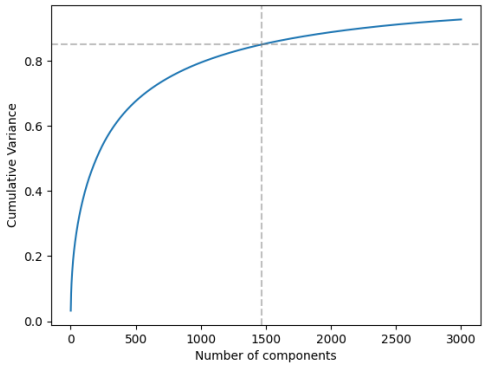

Figure 6. Cumulative variance analysis BOW - Binary Dataset

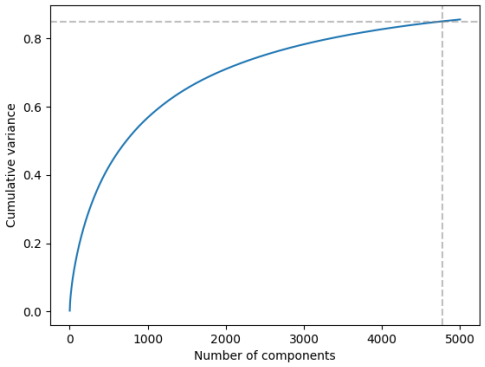

Figure 7. Cumulative variance analysis TFIDF - Binary Dataset

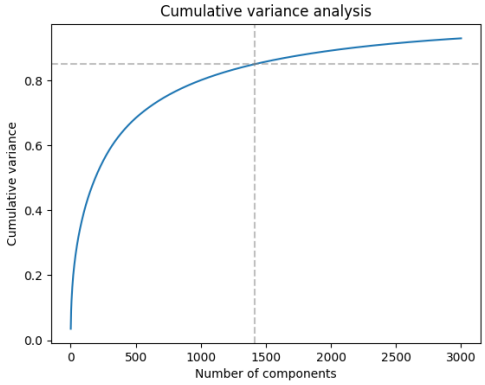

Figure 8. Cumulative variance analysis BOW - Multiclass Dataset

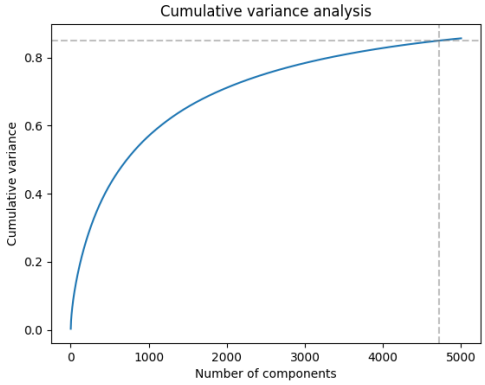

Figure 9. Cumulative variance analysis TFIDF - Multiclass Dataset

Based on figures 6 and 8 it was decided to select 1466 for the binary classification task and 1413 as the number of components for the multi-class classification task.

6.2 TF-IDF

TF-IDF stands for **term frequency-inverse document frequency** and it is a measure, used in Information Retrieval, that can quantify the importance or relevance of string representations in a document amongst a collection of documents.
Here again, the SVD is implemented for the reduction of dimensionality. As with the previous representation (BOW), in this case we also used cumulative variance analysis to choose an optimal number of components.
Based on figure 7 and 9 it was decided to select 4776 for the binary classification task and 4719 as the number of components for the multi-class classification task.

6.3 Word2Vec

Word2Vec is one of the most common techniques in Natural Language Processing for creating word embeddings.
In our case, to create representative vector for the reviews, it is necessary to make two steps. First, the vectors corresponding to all the words in the corpus are generated. Second, for each review, a vector is generated averaging the vectors corresponding to all the words in the review [3].
Word embeddings are generated considering the following parameters:

- **vector_size** = 300;

- **window** = 8;
- **min_count** =2.

7. Classification

The objective of this section is to answer the following two questions:

- Can a review be classified as *good* or *bad* from its text?
- Is it possible to predict the user's rating starting from the text of the review?

To find an answer, different Machine Learning models are implemented on top of each of the proposed text representation, in order to identify the most reliable one[4].
Before going into detail, however, it is necessary to specify an important aspect. As shown in figure 1, classes are heavily unbalanced, with most reviews belonging to class 5. Models trained on this dataset would achieve decent levels of accuracy simply by assigning all reviews to class 5. To avoid this, a downsampling of the dataset is performed. We randomly sampled 20,802 reviews for each class (where 20,802 is the number of reviews belonging to the least populated class).
Moreover, the dataset for the binary classification task was then created, merging classes 1 and 2 as the '*negative*' class and classes 4,5 as the '*positive*' class, thus excluding the intermediate class 3. Here again, it was necessary to balance the dataset, as can be seen from figure 10.
The two resulting datasets have:

- **Binary score dataset**: 114216 reviews
- **Multiclass score dataset**: 104010 reviews

After downsampling, the dataset is divided into training and test sets with a ratio of 70%-30%. The training set is used to train the models and the test set is used to evaluate the performance of the models. This ensures that the models are able to generalize well and that the results are not overly optimistic.

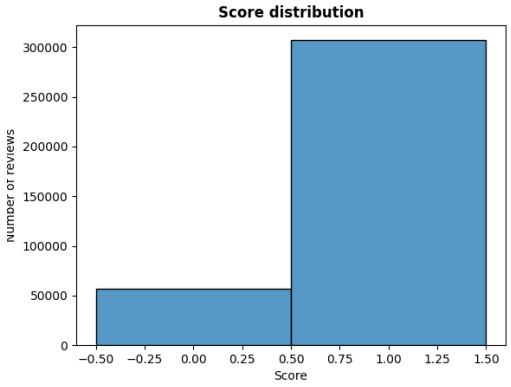

Figure 10. Binary score distribution

7.1 Algorithm Implementation

In this section, we present the algorithms that were implemented for our binary and multiclass classification task. We used four different algorithms: Logistic Regression, XGBoost, K-NN, and Random Forest.
These classifiers were estimated using all three text representations: BOW, TF-IDF, and W2V[3]. Additionally, we performed fine-tuning with cross-validation on each model to improve performance[5]. The optimization algorithm used was Random Search, a method that randomly selects a set of parameters from a predefined distribution and evaluates their performance, as opposed to exhaustively trying every possible combination.

- **Logistic regression** is a type of linear model that is commonly used for classification problems. It estimates the probability of a certain class and makes a prediction based on a threshold.
  For this model, the following parameters have been optimized:

  - **solver**, determines the algorithm used to find the optimal set of parameters that minimize the loss function;
  - **penalty**, used to prevent over-fitting by adding a penalty term to the loss function;
  - **c-values**, denotes the inverse of regularization strength. Smaller values specify stronger regularization.

- **XGBoost** is an optimized implementation of gradient boosting that uses decision trees as weak learners. It is known for its speed and performance on large datasets. It belongs to the family of Ensemble Learning methods.
  In this case, the optimization of hyperparameters was only possible for the BOW representation, because the optimization requires a lot of computational power. For the other representations, an optimal configuration was identified for the task at hand.
  For the BOW representation, the following parameters have been optimized:
  - **max_depth**, the maximum depth of a tree in the model;
  - **eta**, the learning rate of the algorithm;
  - **colsample_bytree**, the subsample ratio of columns when constructing each tree;
  - **sampling_method**, method to sample instances for the model.

- **K-NN**, or k-Nearest Neighbors, is a simple yet powerful algorithm that classifies a point based on the majority class of its k nearest neighbors. It belongs to the family of instance-based learning algorithms.
  For this model, the only optimized parameter is the number of neighbors (**n_neighbors**) which is a hyperparameter used to determine the number of closest points to consider when making a prediction.

- **Random Forest** is an ensemble method that combines multiple decision trees to make predictions. It is known for its ability to reduce overfitting and improve accuracy. It also belongs to the family of Ensemble Learning methods.
  For this model, the following parameters have been optimized:
  - **n_estimators**, which is the number of trees in the forest
  - **max_features**, which is the number of features to consider when looking for the best split in the decision tree

All the models presented have been implemented and optimized using the *sklearn* library [6].

7.1.1 Performance evaluation

In order to evaluate the most suitable classifier for our tasks, it is necessary to be able to make comparisons between the performance of different algorithms. For this purpose, we will mainly focus on **accuracy**, which measures the proportion of correctly classified instances over the total number of instances. Additionally, for binary classification, we will also present the **AUC-ROC curve**. The AUC-ROC curve is a performance measurement for the classification problems at various threshold settings. ROC is a probability curve and AUC represents the degree or measure of separability. It tells how much the model is capable of distinguishing between classes.
For the multi-class classification, we will also analyze the **confusion matrix**, which will allow to have a better understanding of how the classifier is performing on different classes. These evaluations will enable us to compare the performance of different algorithms and determine the most appropriate classifier for our task.

7.2 Analysis Of The Results - Binary Classification

In this section, we aim to address the first research question presented earlier: *Can a review be classified as good or bad from its text?*
The results obtained with the various classifiers are then presented for each text representation considered (table 1).

Model	**Bag-Of-Word**		**TF-IDF**		**Word2Vec**	
	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC
Logistic Regression	**0.87**	0.94	**0.88**	0.95	**0.86**	0.93
K-NN	0.55	0.58	0.64	0.70	0.82	0.90
Random Forest	0.74	0.83	0.79	0.87	0.85	0.92
XG-Boost	0.81	0.89	0.83	0.91	**0.86**	0.93

Table 1. Performance Comparison of Different Models and Text Representations

Based on the results presented in the table, it appears that the Logistic Regression and XG-Boost models performed the best in terms of accuracy and AUC across all text representations. This suggests that these models might be suitable choices for our classification task. However, it is worth noting that the number of components used in each representation may have an effect on the performance of each classifier. For example, the K-NN model performed poorly when using the Bag-of-Word representation with 1466 features, but performed better when using the TF-IDF representation with 4719 features, considering that in any case the TFIDF representation is more informative than BOW. If only the best performance is considered, it might be more appropriate to use Logistic Regression or XG-Boost for our task. However, it is also important to consider the text representation used.
From the results, it can be seen that the TF-IDF representation performed slightly better than the other representations, but it considers almost four times as many features as Bag-of-Word.
Additionally, it's worth noting that the AUC values for the Word2Vec representation are relatively high, which suggests that the model is able to accurately distinguish between positive and negative examples. This could be due to the fact that Word2Vec captures semantic information, allowing the model to better understand the meaning of the text, thus leading to better performance.
In conclusion, considering performance, model complexity, and text representation, it could be recommended to use Logistic Regression with either the TF-IDF or the W2V representation for the classification task.

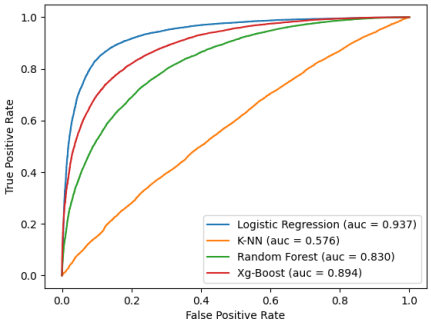

Figure 11. ROC curve Bag-Of-Word

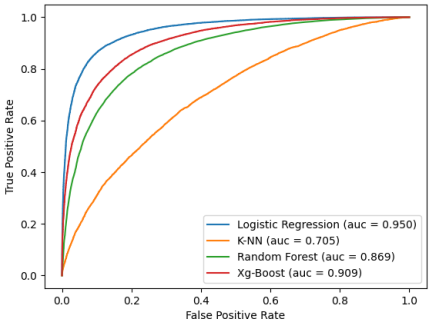

Figure 12. ROC curve TF-IDF

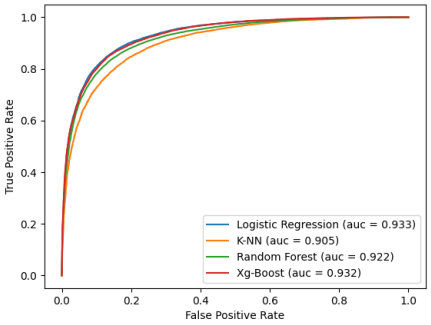

Figure 13. ROC curve Word2Vec

7.3 Analysis Of The Results - Multi-class Classification

In this section, we aim to address the second research question presented earlier: *Is it possible to predict the user's rating starting from the text of the review?*
The results obtained with the various classifiers are then presented for each text representation considered (table 2).

From the results, it can be seen that the overall accuracy of the models is generally lower compared to the binary classification task: this is expected as it's more challenging to classify into multiple classes than two. The Logistic Regression model had the highest accuracy when using the Word2Vec representation, with an accuracy of 0.47. XG-Boost had the highest accuracy when using the TF-IDF representation, with an accuracy of 0.43, followed by Random Forest using Bag-of-Word representation with

Model	**Bag-Of-Word** Accuracy	**TF-IDF** Accuracy	**Word2Vec** Accuracy
Logistic Regression	0.22	0.31	**0.47**
K-NN	0.22	0.22	0.38
Random Forest	**0.28**	0.29	0.45
XG-Boost	0.27	**0.43**	0.45

Table 2. Performance Comparison of Different Models and Text Representations

an accuracy of 0.28. K-NN performed poorly with Bag-of-Word and TF-IDF, but better with Word2Vec with an accuracy of 0.38. However, its performance is still below that of the other models.
It also can be seen that the Word2Vec representation performed significantly better than the other representations. This suggests that the Word2Vec representation is able to capture semantic information in the text data, allowing the model to better understand the meaning of the text and making more accurate predictions. Additionally, Word2Vec is a dense vector representation and does not have the sparsity problem of BOW and TF-IDF representations. This could also contribute to the better performance of the model when using the Word2Vec representation.
Analysing the confusion matrices, which show the number of times each predicted class was assigned to each true class, we

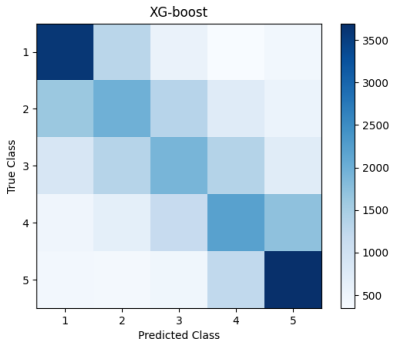

Figure 14. Confusion matrix - XG-Boost

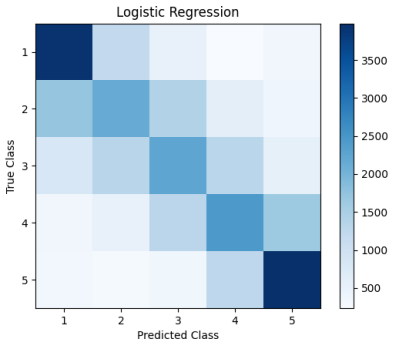

Figure 15. Confusion matric - Logistic Regression

can see that the models performed well in classifying the extreme classes (1 and 5) with high number of true positives, however it struggled with classifying the intermediate classes (2,3, and 4) with a high number of false positives and false negatives. This could be due to the fact that reviews with intermediate ratings may contain mixed sentiments and can be harder to classify. It's worth noting that models tend to perform better on polarized classes, where the sentiment is more clear cut, as opposed to intermediate classes where the sentiment may be more nuanced.

8. Clustering

The objective of this section is to answer the question: Is it possible to group similar reviews?
To find an answer, two different clustering algorithms are implemented: **K-Means** and **Agglomerative clustering**. Before proceeding with the implementation of the algorithms, the same number of reviews are randomly sampled from each of the 5 modes of the variable **Score**, with a procedure similar to that described in section 7.

8.1 Determination of the optimal number of clusters

The proposed clustering algorithms require the number of clusters as an input parameter. The goal is to identify the optimal number of clusters, in order to group in the same cluster reviews as similar as possible and making the differences between clusters as clear as possible.
Among the possible metrics, the optimal number of clusters is determined through **Silhouette coefficients**, which is a suitable metric for both K-Means and Agglomerative clustering. The Silhouette coefficient for a given i point is defined as:

$$S(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

where $b(i)$ is the smallest average distance of point i to all points in any other cluster and $a(i)$ is the average distance of i from all other points in its cluster. The Silhouette Coefficient for the dataset is the average of the Silhouette Coefficient of individual

points.
The Silhouette Coefficient tells us if individual points are correctly assigned to their clusters. Silhouette coefficients near +1 indicate that the sample is far away from the neighboring clusters. A value of 0 indicates that the sample is on or very close to the decision boundary between two neighboring clusters. Negative values indicate that those samples might have been assigned to the wrong cluster.
For each of the proposed text representation, different values are tested for the optimal number of clusters, selecting the optimal one through the analysis of the Silhouette coefficients. Below, the example of the BOW representation, for both K-Means algorithm and Agglomerative Clustering:

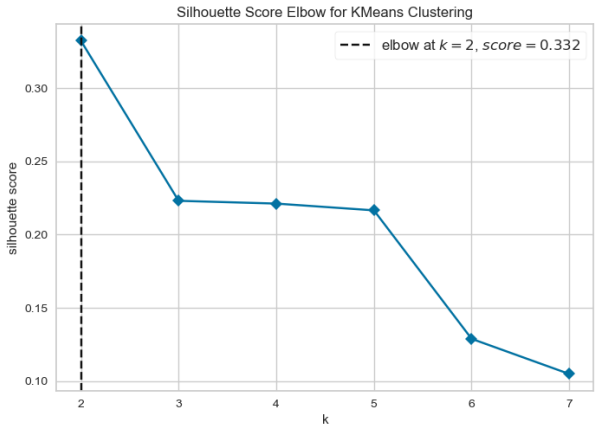

Figure 16. K-Means

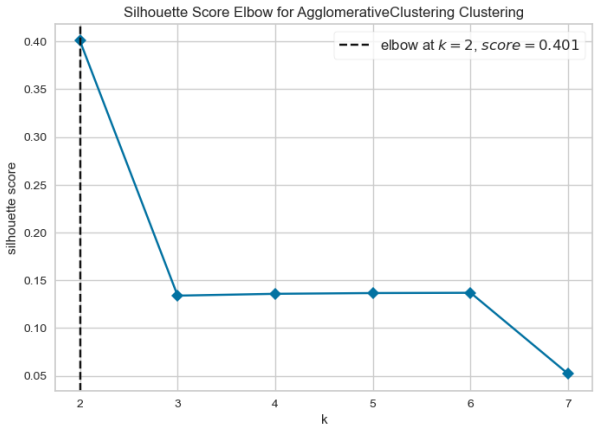

Figure 17. Agglomerative Clustering

In both cases the optimal number of clusters is 2, the one that leads to the higher Silhouette coefficient.

8.2 K-Means

K-Means is one of the simplest and popular unsupervised machine learning algorithms. The K-Means algorithm aims to divide observations into a number of non-overlapping clusters. The first step of the algorithm is to randomly select the centroids that define the clusters, assigning each observation to the nearest cluster centroid. After this, it proceeds in an iterative way recalculating the centroids, repeating the iteration until a certain stop criterion occurs (reaching a certain number of iterations or if the centroids are not modified for a certain number of consecutive iterations).
According to 8.1, different values for the optimal number of clusters are tested for each of the considered text representations. Below are the results.

8.2.1 Results

Based on the Silhouette coefficients, for BOW and Word2Vec representation the optimal number of clusters turns out to be 2, while for TF-IDF representation turns out to be 3 (Appendix, figure 18 and 20). In addition to Silhouette coefficients, we calculate the **Davies-Bouldin index** as an additional metric for clustering validation in the absence of ground truth. The **Davies-Bouldin index** is defined as the average similarity measure of each cluster with its most similar cluster, where similarity is the ratio of within-cluster distances to between-cluster distances. Thus, clusters which are farther apart and less dispersed will result in a better score. The minimum score is zero, with lower values indicating better clustering.

Text representation	Silhouette	Davies-Bouldin
BOW	0.336	5.376
TF-IDF	0.013	6.257
W2V	0.086	3.035

Table 3. Validation metrics

Both the Silhouette coefficients, very close to 0, and the Davies-Bouldin indexes, far from 0, indicate that the clusters are not well separated. For a qualitative analysis of the results, the 5 most frequent words for each cluster are also reported in table 4 (in Appendix there are also WordClouds).
As you can see, it is difficult to find a common concept for all reviews grouped in the same cluster.

Text representation					
BOW - cluster 1	taste	like	product	good	one
BOW - cluster 2	like	taste	one	coffee	product
TF-IDF - cluster 1	tea	taste	like	flavour	good
TF-IDF - cluster 2	coffee	like	taste	cup	flavour
TF-IDF - cluster 3	like	taste	product	good	one
W2V - cluster 1	taste	like	coffee	flavour	good
W2V - cluster 2	product	one	like	food	would

Table 4. Top-5 most frequent words per cluster

8.3 Agglomerative Clustering

Agglomerative Clustering is a type of hierarchical clustering algorithm that uses a bottom-up approach. In particular, initially each observation starts in its own cluster. Then, the clusters are progressively merged until you get to have a single cluster that contains all the observations.
Agglomerative clustering is a much more computationally expensive method, so only 10,000 observations have been used. As before, different values for the number of clusters are evaluated for each text representation. Below are the results.

8.3.1 Results

For all text representations the optimal value for the number of clusters is 2 (Appendix, figure 19 and 21). Below are the results for the obtained models:

Text representation	Silhouette	Davies-Bouldin
BOW	0.401	5.642
TF-IDF	0.002	8.022
W2V	0.178	1.825

Table 5. Validation metrics

The results are very similar to those obtained with K-Means, with slight improvements in the case of Word2Vec. The most common words for clusters are also reported (in (Appendix there are also WordClouds):

Text representation					
BOW - cluster 1	taste	like	product	good	one
BOW - cluster 2	like	coffee	taste	one	food
TF-IDF - cluster 1	like	taste	product	good	one
TF-IDF - cluster 2	coffee	like	taste	cup	flavour
W2V - cluster 1	taste	like	coffee	flavour	good
W2V - cluster 2	product	like	one	food	would

Table 6. Top-5 most frequent words per cluster

Even the most frequent words are similar to those reported for K-Means clusters. Thus, it is still difficult to identify a characteristic concept for the various clusters.

9. Conclusion

- For the **binary classification** task, the Logistic Regression and XG-Boost models performed the best in terms of accuracy and AUC across all text representations. Using the TF-IDF or Word2Vec representation with these models may yield the best results.
- For the **multiclass classification** task, it can be seen that Word2Vec representation performed better than the other representations. This could be due to the fact that Word2Vec captures semantic information, allowing the model to better understand the meaning of the text, thus leading to better performance. Overall, it is recommended to use the Word2Vec representation in combination with the Logistic Regression model for the multiclass classification task.

- **Clustering** has not produced satisfactory results. For both clustering algorithms metrics are bad, despite slight improvements in agglomerative clustering. This leads to the conclusion that there are probably no well-defined clusters in the available data.
  This may be due to several reasons, such as the presence of a large number of semantically useless words. In addition, it should be noted that the application of the proposed text representation on short texts (as the reviews) tend to generate sparse vectors, difficult to manage for the proposed clustering algorithms.
  In order to achieve more satisfactory results, an ad-hoc preprocessing could be implemented (for example, with a rigorous application of the Zipf's law, implementing cut-off values such as to keep only semantically relevant words).

Bibliography

[1] *Stanford Network Analysis Project. Amazon Fine Food Reviews*. URL: https://www.kaggle.com/datasets/snap/amazon-fine-food-reviews.

[2] Chen H-H Cheng C-H. "Sentimental text mining based on an additional features method for text classification". In: (2019). DOI: https://doi.org/10.1371/journal.pone.0217591.

[3] Dilip Valeti. *Classification using Word2vec*. URL: https://medium.com/@dilip.voleti/classification-using-word2vec-b1d79d375381.

[4] G. Pasi and M. Viviani. *Text Mining and search course lecture notes and slides*. 2022-23.

[5] *Tune Hyperparameters for Classification Machine Learning Algorithms*. URL: https://machinelearningmastery.com/hyperparameters-for-classification-machine-learning-algorithms/.

[6] *Scikit-learn*. URL: https://scikit-learn.org/stable/index.html.

10. Appendix

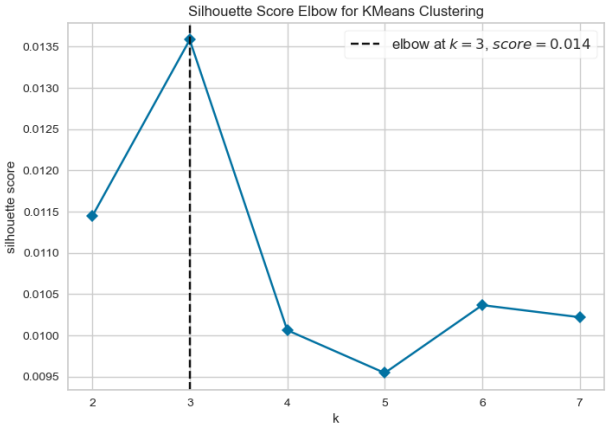

Figure 18. TF-IDF K-Means

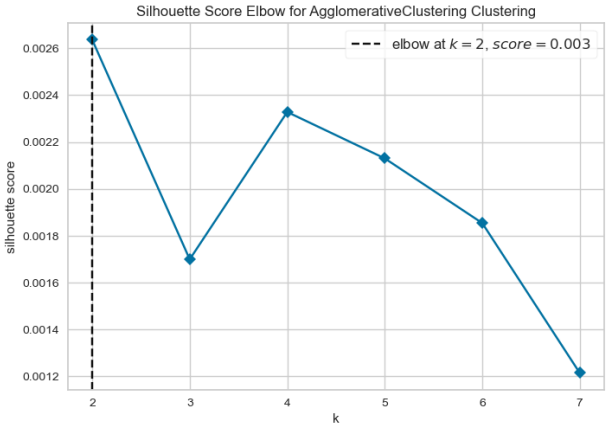

Figure 19. TF-IDF Agglomerative Clustering

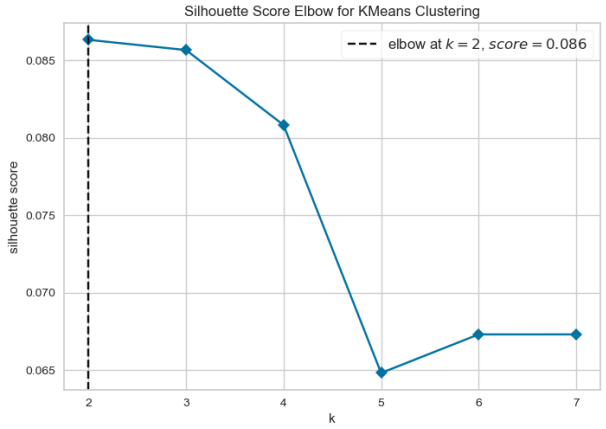

Figure 20. W2V K-Means

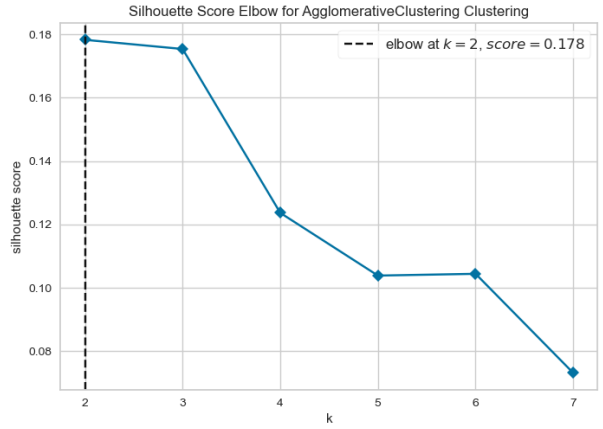

Figure 21. W2V Agglomerative Clustering

Figure 22. BOW - K-Means - Cluster 1

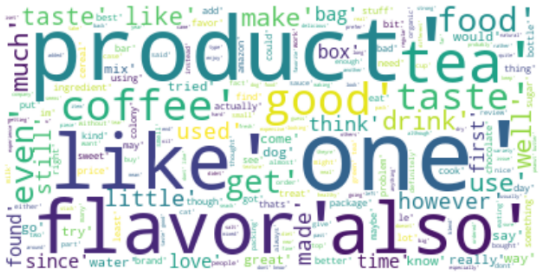

Figure 23. BOW - K-Means - Cluster 2

Figure 24. BOW - Agglomerative - Cluster 1

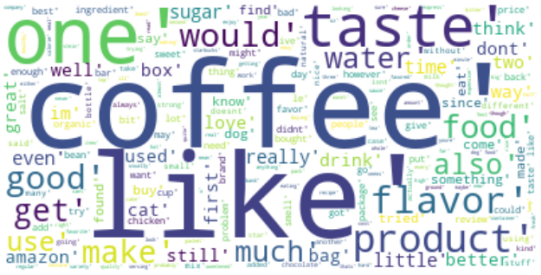

Figure 25. BOW - Agglomerative - Cluster 2

Figure 26. TF-IDF - K-Means - Cl. 1

Figure 27. TF-IDF - K-Means - Cl. 2

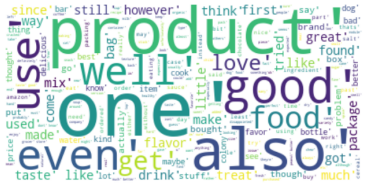

Figure 28. TF-IDF - K-Means - Cl. 3

Figure 29. TF-IDF - Agglomerative - Cluster 1

Figure 30. TF-IDF - Agglomerative - Cluster 2

Figure 31. W2V - K-Means - Cluster 1

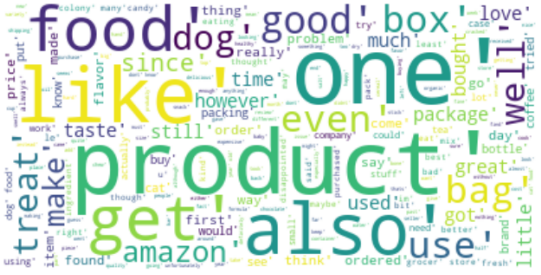

Figure 32. W2V - K-Means - Cluster 2

Figure 33. W2V - Agglomerative - Cluster 1

Figure 34. W2V - Agglomerative - Cluster 2