# From Reviews to Refinement: Driving Product Excellence with Advanced Sentiment Analysis

## List of team members & tasks

Tal Ramon:

- Conducted NLP research and outlined next steps & best practices
- Created Github page

Dan Finel:

- Initial EDA including preprocessing
- Modeling and additional visualizations

Nick Mishkin:

- Initial EDA
- Product review analysis and data cleaning

# Section 1.0

*1.1: Project name*

Our project is called *Sentiment Synthesizer*.

*1.2: Description*

The goal of our project is to create a tool for sellers of products to get insights on the reviews they have of their products. We want to tell them what customers like from their products and what they do not like.

At this stage, our project is in an intermediate phase, having successfully established a baseline model and conducted exploratory data analysis (EDA). We're now at a crucial turning point, evaluating possible methodologies such as SMOTE to address the imbalance in our target variable, and considering the implementation of feature selection techniques like chi-square to optimize our Bag of Words (BoW) inputs. We're also deliberating on the best practices for NLP preprocessing to enhance the quality of our data. Our ambition to progress into deep learning models, including CNNs and RNNs, signifies our commitment to adopting sophisticated approaches for sentiment analysis. Moreover, our initiative to develop a feature-sentiment dictionary for business applications highlights our strategic direction towards converting analytical insights into actionable intelligence for product improvement. This phase of our project is characterized by a concerted effort to refine model performance and accuracy while ensuring that our technical advancements align closely with tangible business outcomes, showcasing our forward-thinking approach to leveraging NLP for enhancing product development and customer satisfaction.

*1.2: Business Use*

From a business perspective, our platform offers valuable insights into customer satisfaction and product performance. By leveraging natural language processing (NLP) for sentiment analysis, companies can prioritize product development efforts, enhance customer experience, and strategically address feedback to improve product ratings and sales. This approach not only helps in refining product offerings but also in building a positive brand image by responding to customer needs and preferences.

# Section 2.0

*2.1: Data Exploration*

- Our data includes 568,454 product reviews and 10 features:
  - 'Id', 'ProductId', 'UserId', 'ProfileName', 'HelpfulnessNumerator', 'HelpfulnessDenominator', 'Score', 'Time', 'Summary', 'Text'

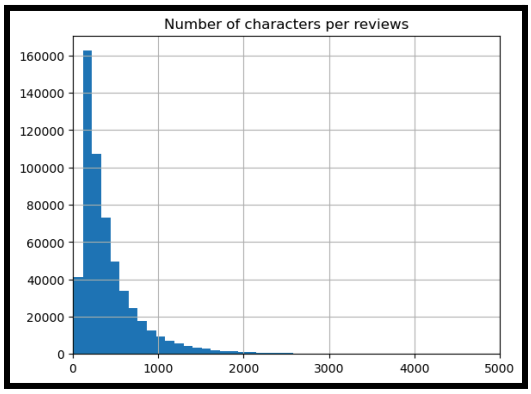
```
In [5]: df.columns

Out[5]: Index(['Id', 'ProductId', 'UserId', 'ProfileName', 'HelpfulnessNumerator',
               'HelpfulnessDenominator', 'Score', 'Time', 'Summary', 'Text'],
              dtype='object')
```

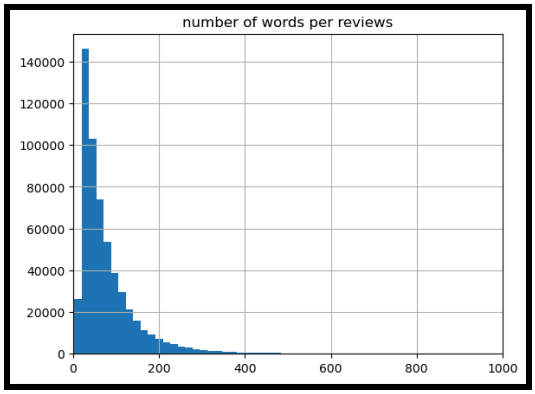
- We keep only the features ‘ProductId’,’Score’ and ‘Text’. ‘Score’ is our target variable. We use ‘UserId’ just to remove duplicates.

- Out of these 568,454 product reviews, 58,040 have duplicate reviews.
- Out of the 58,040 duplicates, 2,122 have the same User Id.
- Additionally, 14,188 duplicates have over 100 words but have different User ID's.
- After feature selection, there are no null values.

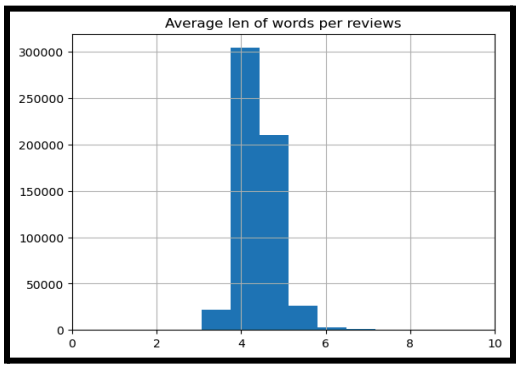
```
In [12]: df.isnull().sum()

Out[12]: ProductId    0
         Text         0
         Score        0
         dtype: int64
```

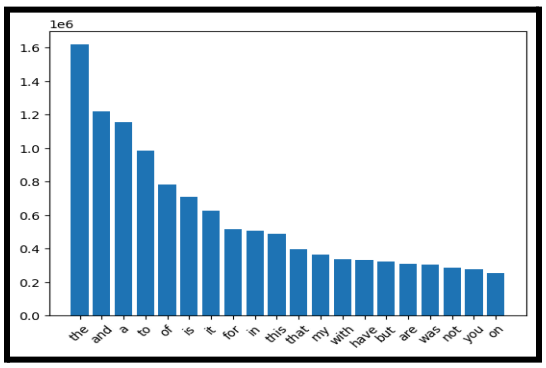
- We have 43,842 products in our dataset with more than 2 reviews.
- We have 17,310 products in our dataset with more than 5 reviews.
- We have 9,618 products in our dataset with more than 10 reviews.
- We have 862 products in our dataset with more than 100 reviews.
- Most of our reviews have less than 1,000 characters. The majority of them have less than 100 characters. Diving deeper, mo

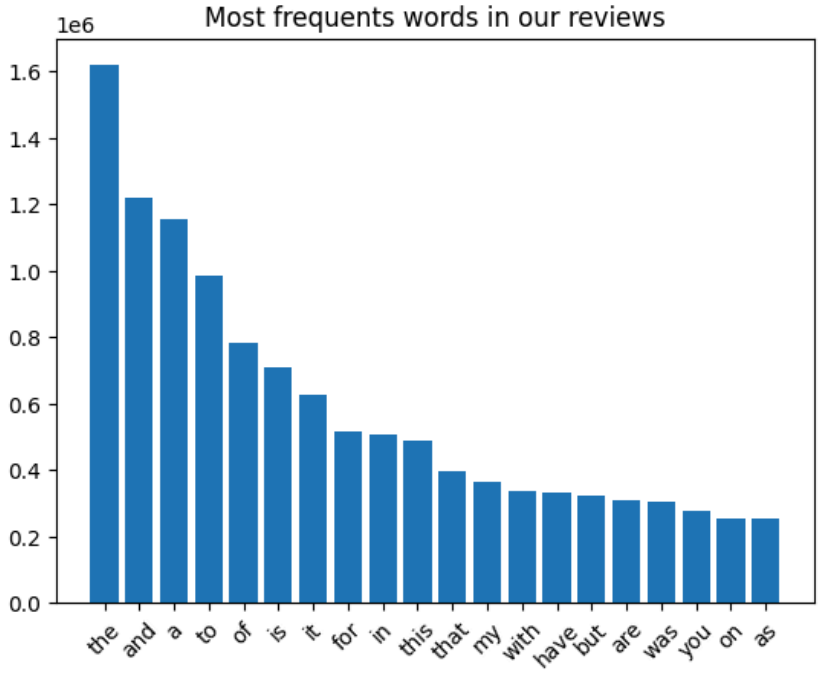



- Most of the reviews have a length of words between 3 and 6. This can be explained by the 'stop words' which are reducing the mean.



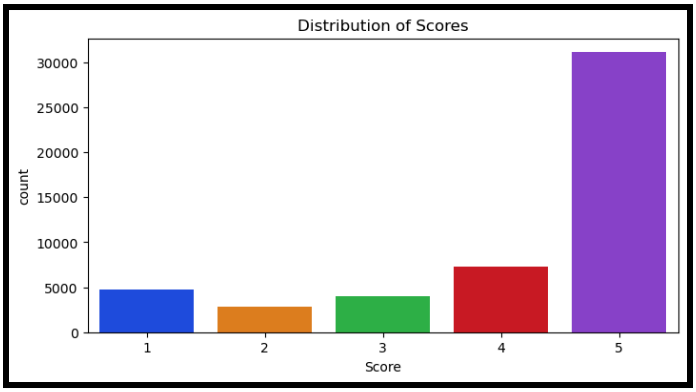





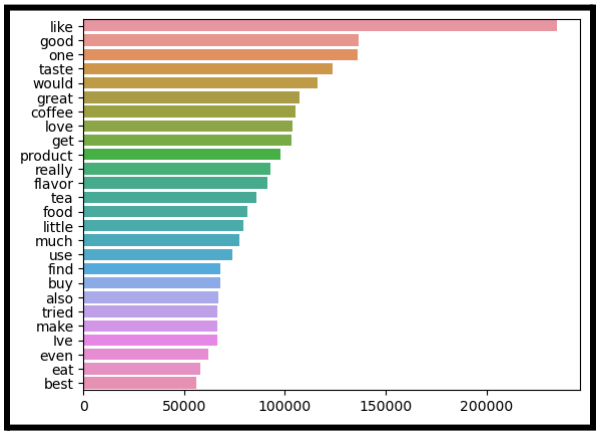

- We see how often stop words are appearing in our dataset.

*2.2 Target Analysis*

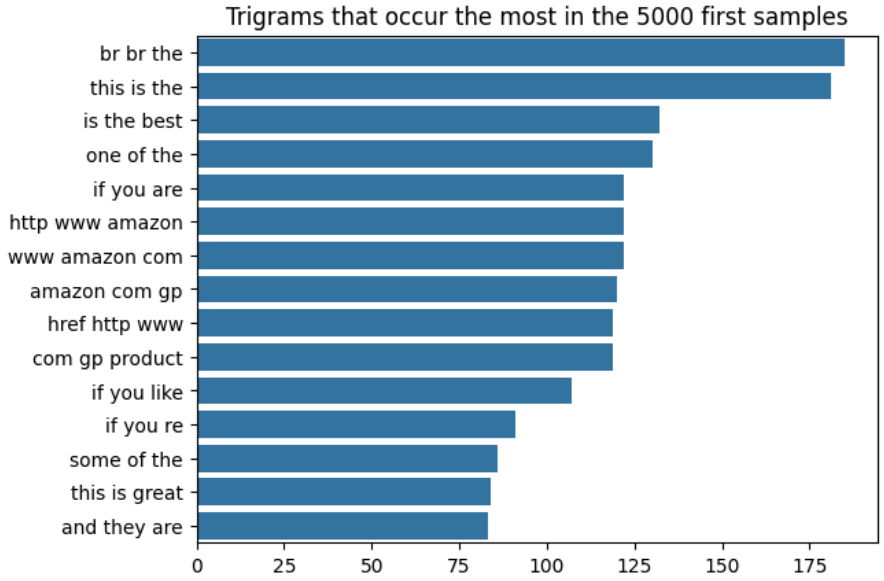
- 64% of reviews have a score of 5, 14% a score of 4, 9% a score of 1, 8% a score of 3, and 5% a score of 2.



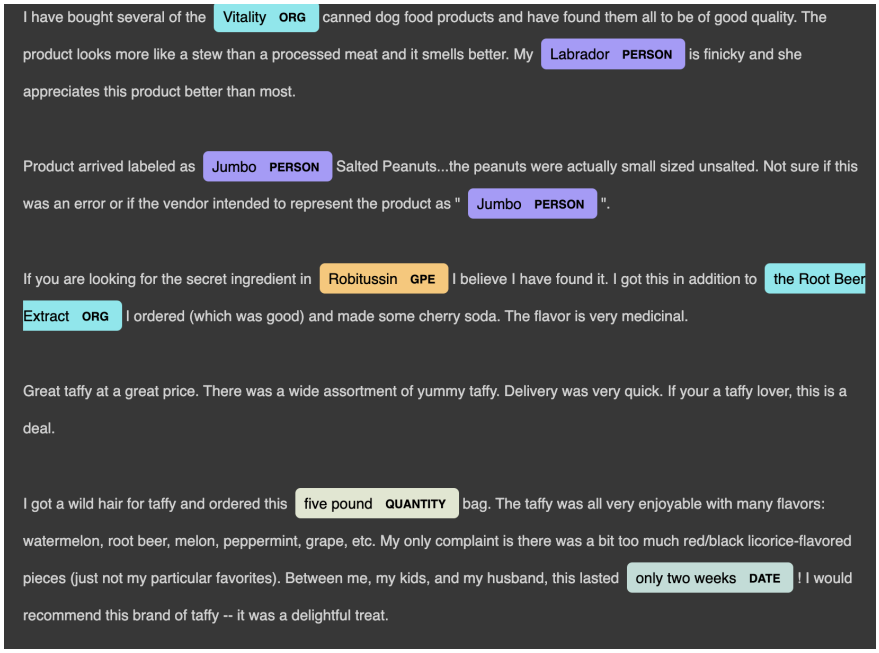
- On the above graph, we see that we have a lot of positive words due to the fact that our dataset is imbalanced with a lot of good reviews with words like ‘good’,’great’,’love’.



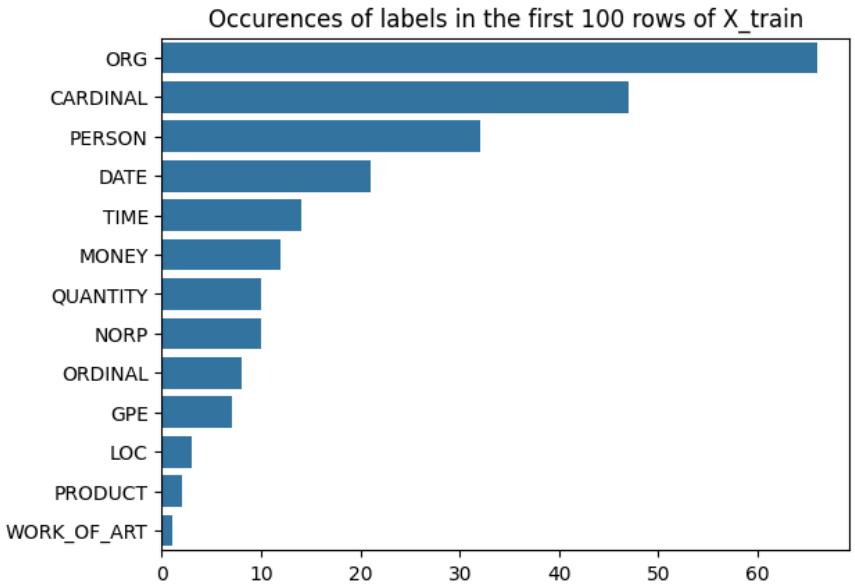

- On the above graph, we see the trigrams that occur the most in the first 5000 samples. We have a lot of reference to amazon and also to the link of the product. We also have a lot of positive review patterns such as ‘if you like’, ‘is the best’, ‘this is great’. We will have to do some data cleaning.




- To learn the dataset deeply, we split our data into train, validation and test set using the GroupShuffleSplit from sklearn.model_selection. We split the data according to the feature ‘ProductId’.
- Afterwards, we only worked with the train set. We first used the library **‘spacy’** to see if we have words that refer to a person, an organization, a country, a date,...
  This is what we got for the first 5 samples of the train set :

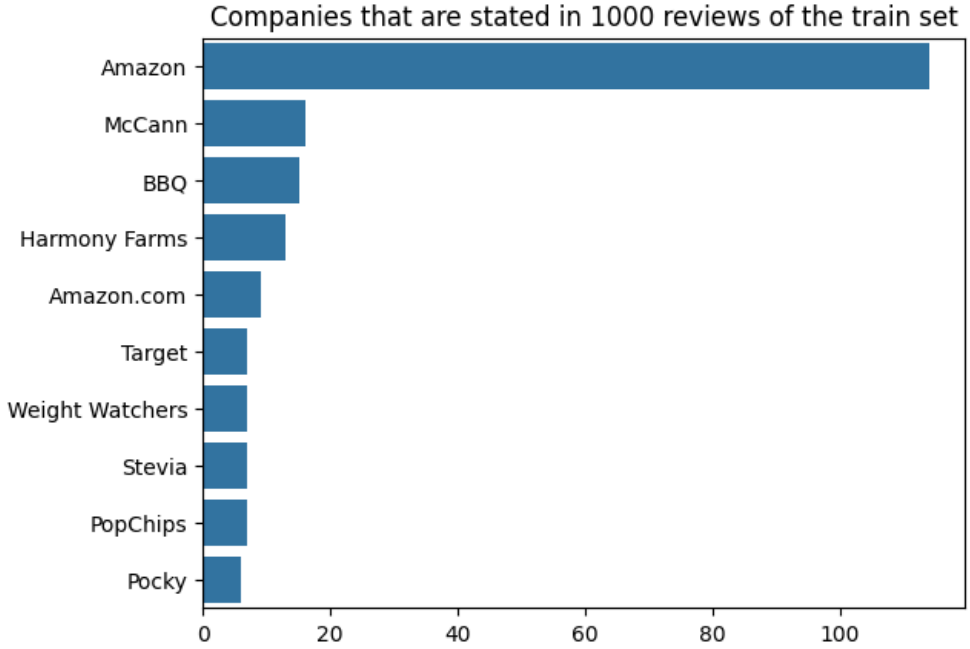


This technique is very visual and helps us understand the data better.
Then, we plotted the occurrences of the labels we got from the spacy library for the first 100 rows of X_train :

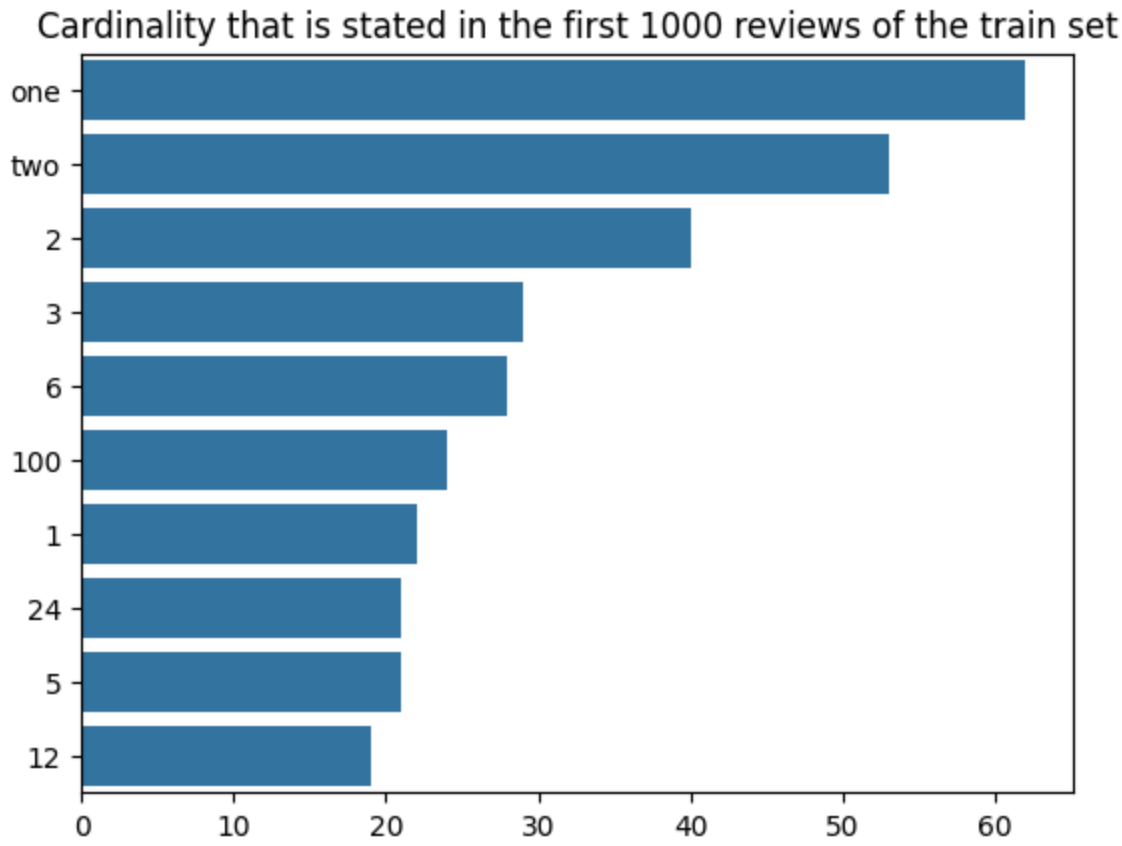



We see that we have a lot of companies that are stated in the reviews, probably corresponding to the company that is selling the product. We also have a lot of references to numbers and persons.

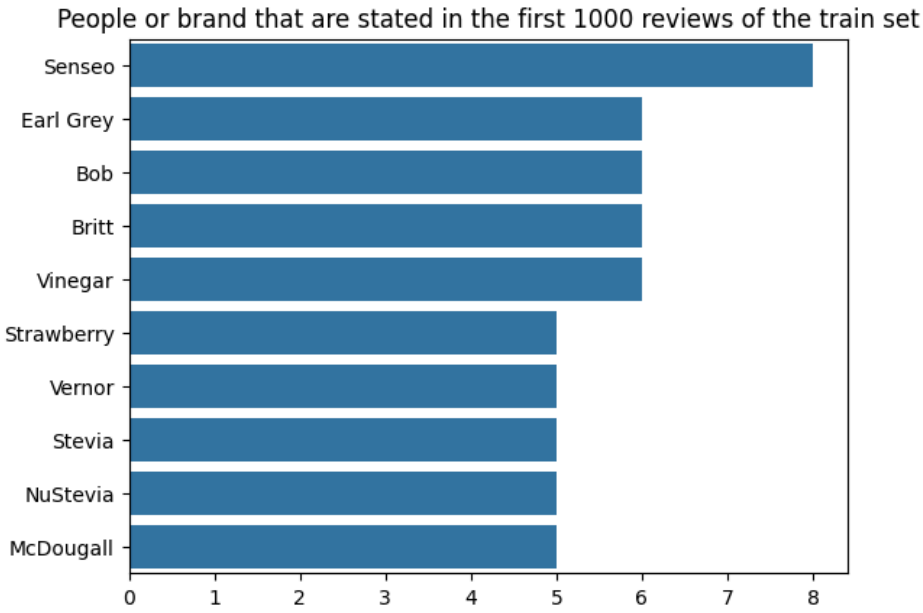
Let's go deeply into it :




With no surprise, Amazon is highly stated in the reviews.




We can observe that some numbers are recurrent on the reviews.




We see that some brands are recurrent, probably linked to the brand of the product. ‘Senseo’ being a coffee brand, we can imagine that we have a lot of reviews from coffee

products. ‘Earl Grey’ being a tea brand, we can imagine that we have a lot of reviews from tea products.

- To clean our data, we implemented a function to :
  - Remove contractions
  - Tokenized each word with word_tokenize from **nltk.tokenize**
  - Lowered all the words
  - Removed the words of length lower than 2
  - Removed punctuation
  - Removed digits
  - Removed stop words. We added some stop words to the one given by nltk.corpus such as ‘amazon’,’like’,’product’,’flavor’,’product’,... as they were omnipresent on the reviews. We will go deeper on those stop words in the next step to keep some of them and remove some more.
  - Lemmatized words using WordNetLemmatizer from **nltk.stem**

- We added all those words to a list of lists named corpus. Here are the most occuring words of the 50k first samples of X_train :

```
[('not', 58599),
 ('tea', 16058),
 ('great', 14363),
 ('love', 14001),
 ('coffee', 13129),
 ('dog', 10612),
 ('really', 8993),
 ('much', 8388),
 ('little', 7412),
 ('drink', 7248),
 ('price', 7076),
 ('buy', 6828),
 ('treat', 6722),
 ('even', 6573),
 ('best', 6537),
 ('well', 6444),
 ('better', 6239),
 ('store', 5821),
 ('chocolate', 5754),
 ('sugar', 5643)]
```

# Section 3.0

- Our project is done in two parts : one part is supervised with the prediction of the score of a review and the other part is unsupervised with finding insights on each review.
    - For the supervised part , as we want to give insights to sellers about the reviews of their product, low scores may be more helpful for them than the higher ones. Thus, we need to capture low scores in priority. Then, recall seems to be the most adapted metric to deal with.
    - For the unsupervised part, we can use the perplexity and the coherence score.

- For the supervised part, we have chosen 3 differents baseline models :
    - LogisticRegression
    - KNN
    - DummyClassifier (with all possibles strategies)

  To run those models, we have to create a bag of words from the initial reviews and then get a sparse matrix where features are each word and rows correspond to each review.
  We end up with a sparse matrix of shape (50000,43332).
  We have 50000 rows as the computation is too heavy for more samples. To get a first result, we thought that 50k samples would be enough to get first insights on our dataset.
  We need to remove some features by proceeding to feature selection. For our baseline model, we just remove features with less than 100 non zeros values. We end up having 2051 features which seems more reasonable. We may need to do further feature selection in the future to obtain more accurate results and take more into consideration each word.

  Table of result for the supervised baseline models :

| | Precision_1 | Precision_2 | Precision_3 | Precision_4 | Precision_5 | Recall_1 | Recall_2 | Recall_3 | Recall_4 | Recall_5 | accuracy |
|---|---|---|---|---|---|---|---|---|---|---|---|
| LogisticRegression | **0.6** | **0.29** | **0.31** | **0.39** | **0.78** | **0.55** | **0.21** | **0.2** | **0.2** | **0.92** | **0.69** |
| KNN | 0.31 | 0.2 | 0.2 | 0.24 | 0.7 | 0.26 | 0.11 | 0.1 | 0.08 | 0.89 | 0.61 |
| DummyClassifier (strategy = 'most_frequent') | 0.0 | 0.0 | 0.0 | 0.0 | 0.63 | 0.0 | 0.0 | 0.0 | 0.0 | 0.77 | 0.63 |
| DummyClassifier (strategy = 'prior') | 0.0 | 0.0 | 0.0 | 0.0 | 0.63 | 0.0 | 0.0 | 0.0 | 0.0 | 0.77 | 0.63 |
| DummyClassifier (strategy = 'stratified') | 0.09 | 0.05 | 0.06 | 0.15 | 0.62 | 0.08 | 0.05 | 0.07 | 0.14 | 0.63 | 0.43 |
| DummyClassifier (strategy = 'uniform') | 0.09 | 0.05 | 0.08 | 0.13 | 0.62 | 0.18 | 0.19 | 0.19 | 0.19 | 0.2 | 0.2 |

- For the unsupervised part, we implemented a Lda model from the **gensim** library. To do it, we created a bag of words based on the corpus list we got during preprocessing.

```
import gensim
dic=gensim.corpora.Dictionary(corpus) # assign an
bow_corpus = [dic.doc2bow(doc) for doc in corpus]
```

```
lda_model = gensim.models.LdaMulticore(bow_corpus,
                                       num_topics = 5,
                                       id2word = dic,
                                       passes = 5,
                                       workers = 2) # We
# LDA models are used to get the main topics of our c
# For each topic, we get the top words and the probab
# based on all the possible words of this topic
lda_model.show_topics()
```

We got those topics :

```
[(0,
  '0.048*"tea" + 0.035*"not" + 0.022*"drink" + 0.010*"sugar" + 0.010*"water" + 0.009*"green" + 0.006*"really" + 0.006*"sweet" + 0.006*"great" +
0.006*"energy"'),
 (1,
  '0.065*"coffee" + 0.032*"not" + 0.017*"cup" + 0.009*"great" + 0.007*"strong" + 0.006*"love" + 0.006*"cat" + 0.006*"roast" + 0.006*"kcups" +
0.005*"price"'),
 (2,
  '0.034*"not" + 0.029*"dog" + 0.018*"treat" + 0.015*"love" + 0.007*"great" + 0.006*"cat" + 0.006*"day" + 0.005*"give" + 0.005*"little" + 0.005*"year"'),
 (3,
  '0.039*"not" + 0.009*"price" + 0.009*"chocolate" + 0.009*"store" + 0.007*"buy" + 0.007*"great" + 0.007*"order" + 0.006*"salt" + 0.005*"chip" +
0.005*"much"'),
 (4,
  '0.029*"not" + 0.012*"great" + 0.009*"love" + 0.009*"snack" + 0.007*"cereal" + 0.006*"really" + 0.006*"bar" + 0.006*"little" + 0.006*"sweet" +
0.006*"rice"')]
```

We get 5 main topics :

- One seems to be about dog food
- One seems to be about tea and drinks
- One seems to be about food with chocolate
- One seems to be about coffee with a chocolate flavor maybe
- One seems to be about snack food such as chips

We see that many positive words are very frequent in the dataset such as ‘delicious’,’love’,’good’,... we also have many words from fooding products such as ‘snack’,’chocolate’,’salt’,’sweet’.

Table of result for the unsupervised model :

| | perplexity | coherence score |
|---|---|---|
| LDA Model | -7.7 | 0.43 |

We see that our perplexity is very low but the coherence score is not that bad.

# Section 4.0

*4.1: Questions*

- Do we need to use Resampling - SMOTE for the imbalanced target? The baseline model identified moderately class 1- accuracy of 0.61.
- Do we need to add a method for feature selection? Like chi2 from sklearn? If yes, can we use it based on the BoW?
- Regarding the NLP preprocessing: we have a step of removing all words that have a length <=2 characters ('TX', 'by',...). Is it the best practice? Or just to remove only 1 character of a word (for example: b d w…)?

*4.2: Next steps*

- Use deep learning models: CNN, RNN.
- Create a dictionary with the relevant 'features' per product and its sentiment for the business use case- to show the seller on Amazon what features in the product he/she needs to improve- based on the Negative Sentiment, class 1-2.