Natural Language Processing

Ignacio Martínez Martin, Pablo Valle Nieto, Pedro Rodríguez Viñuales

January 2022

Contents

1 Introduction

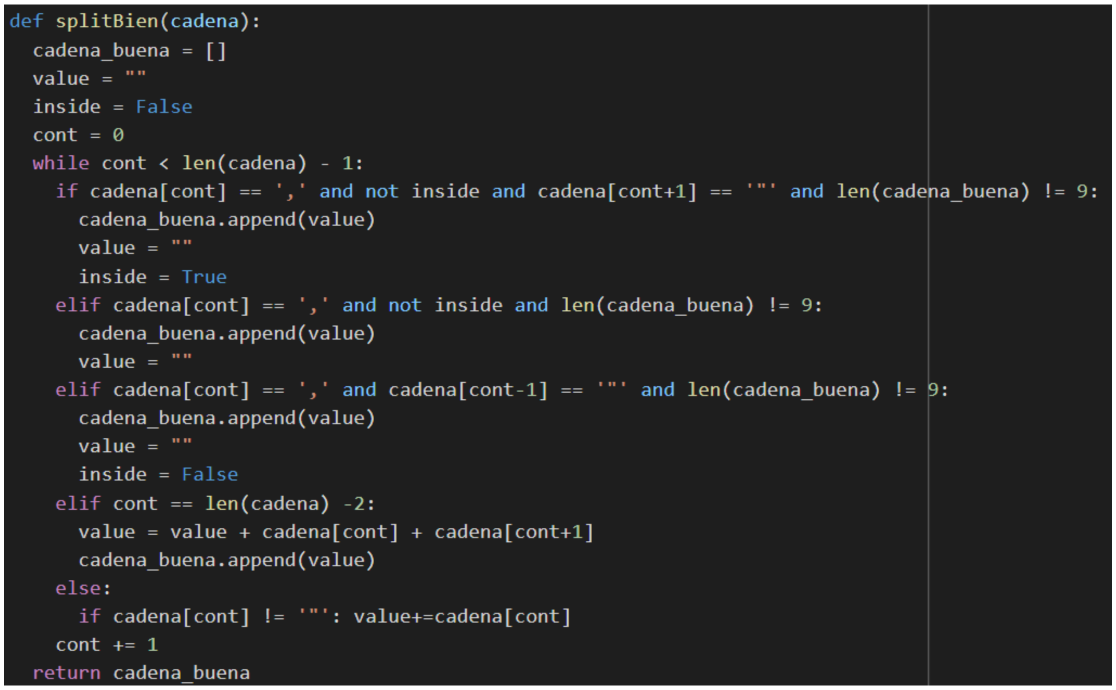
The aim of this project is to analyze a collection about comments on purchases about drinkable and edible products. The collection is stored in the file “products.csv” made up of the following the fields:

1. **Id**: number
2. **ProductId**: String
3. **UserID**: String
4. **ProfileName**: String
5. **HelpfulnessNumerator**: int
6. **HelpfulnessDenominator**: int
7. **Score**: int in the range [1,5]
8. **Time**: int
9. **Summary**: String
10. **Text**: String

2 Preprocessing

In order to achieve the objective of this project, we need to carry out a series of steps that include obtaining the correct information from the collection and optimizing in order to achieve the best results.

To obtain the dataset information, it was not enough to write *”read_csv”* so we divided the text into lines and passed the following method to each line to divide the information as a list and be able to pass it to a collection.



```
def splitBien(cadena):
  cadena_buena = []
  value = ""
  inside = False
  cont = 0
  while cont < len(cadena) - 1:
    if cadena[cont] == ',' and not inside and cadena[cont+1] == '"' and len(cadena_buena) != 9:
      cadena_buena.append(value)
      value = ""
      inside = True
    elif cadena[cont] == ',' and not inside and len(cadena_buena) != 9:
      cadena_buena.append(value)
      value = ""
    elif cadena[cont] == ',' and cadena[cont-1] == '"' and len(cadena_buena) != 9:
      cadena_buena.append(value)
      value = ""
      inside = False
    elif cont == len(cadena) -2:
      value = value + cadena[cont] + cadena[cont+1]
      cadena_buena.append(value)
    else:
      if cadena[cont] != '"': value+=cadena[cont]
    cont += 1
  return cadena_buena
```

Figure 1: Method to read the file

Once the collection is obtained correctly, we must clean the "Text" column. To carry out this cleaning, special characters, lemmatize all terms and remove all capital letters.

2.1 Special characters

First of all, we are going to remove the punctuation marks and the useless characters.

We have considered all the English punctuation marks. So, in order to remove them, an algorithm has been developed and you can find it in the below section.

2.2 All capital letters

Once we have deleted all the punctuation marks we can proceed by removing all the capital letters that exists in the data of the data-set. As we are using Python3 we can easily turn all the text from capital letter to lower case by using a simple function.

So, in order to remove them from all the reviews, an algorithm has been developed and you can find it in the below section.

2.3 Lemmatize all terms

As we have successfully removed all the punctuation marks and capital letter we can now lemmatize all the words that are part of the different sentences that we have into the data-set.

Lemmatisation is the algorithmic process of determining the lemma of a word based on its intended meaning. So, we have to do this because we do not have any profit by using the conjugated words so by removing them we simplify the data.

In order to remove them from all the reviews, an algorithm has been developed and you can find it in the below section.

3 Vectorization

Term Frequency - Inverse Document Frequency) is a statistical data that shows the degree of frequency of a word in a specialized, non-generic, corpus of documents.

For example, if I want to search Google for the following phrase What is Neoplatonism Google, without using tf-idf, will probably find something similar to:

What: very common word, billions of matches is: very common word, billions of matches Neoplatonism: very rare word: only thousands of coincidences.

What TF—IDF does is recognize that the term Neoplatonism is a very rare word, so the search will focus on those documents that contain that rare word (hence the inverse document frequency). Focusing on the rare word, will give me the results I'm looking for, even if it's only a few hundred, discarding those that use much more common words from the search.

To carry out this process, we have vectorized all the opinions of the collection to show the degree of frequency of a word in each opinion.

We have seen ourselves in the delivery of, when carrying out this process, to use only 2500 rows of the collection due to using errors in RAM.

4 Feature selection

The SelectKBest method selects the features according to the k highest score. By changing the 'score_func' parameter we can apply the method for both classification and regression data. Selecting best features is important process when we prepare a large dataset for training. It helps us to eliminate less important part of the data and reduce a training time.

In our context, the parameter "k" will be approximately 30 per cent of the features of the collection.

Before moving on to the next point we will classify the opinions according to the field "score" (1, 2, 3, 4 or 5 stars).

5 Classification algorithm

The problem consists of predicting the score of a given opinion, which is found within the collection.

For this step we will partition the collection labeled: 30% test and 70% training. Once this is done, we will define the SVM model and train it with the training data (70%).

After that, we use the test data to make the prediction. In order to better visualize the prediction we have used a confusion matrix and we have printed a classification_report on the screen in order to measure our prediction.

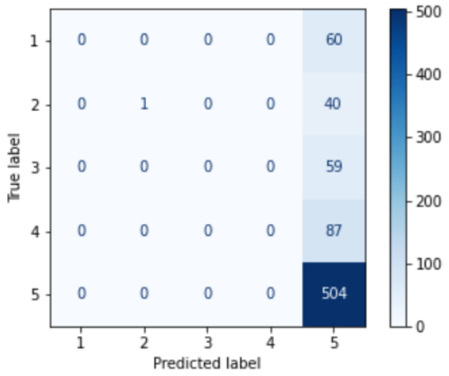
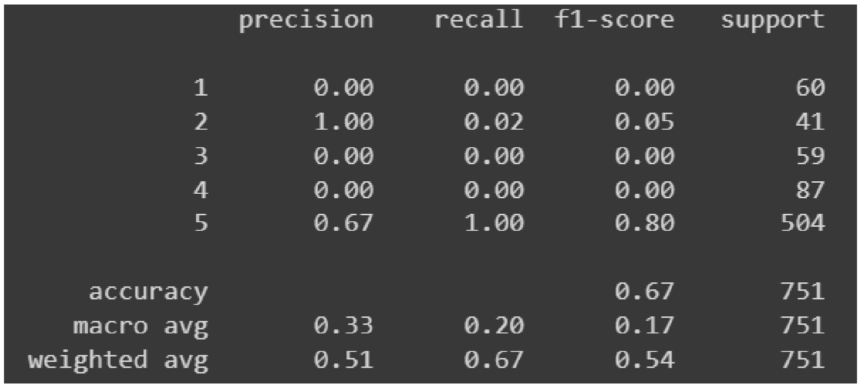


Figure 2: Confusion matrix



```
              precision    recall  f1-score   support

           1       0.00      0.00      0.00        60
           2       1.00      0.02      0.05        41
           3       0.00      0.00      0.00        59
           4       0.00      0.00      0.00        87
           5       0.67      1.00      0.80       504

    accuracy                           0.67       751
   macro avg       0.33      0.20      0.17       751
weighted avg       0.51      0.67      0.54       751
```

Figure 3: Classification report