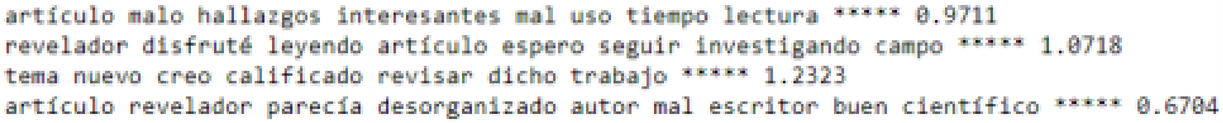
Travis Browning
University of Wolverhampton
Machine Learning, L7N008
Dr. Burcu Can
Assignment 2 Report
17 June 2021

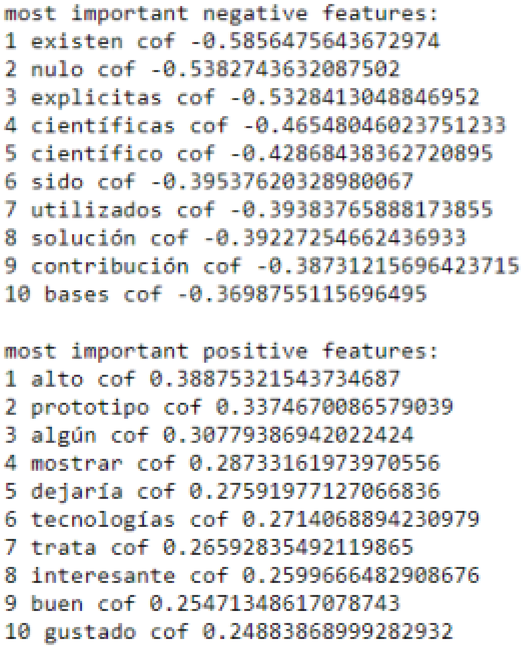
Although task 1 of the assignment was to build the linear regression model, in order to build this model, text would first need to be preprocessed. I first tried to use json.dumps() to filter the data, but even when opening the document with encoding, accented characters did not display correctly. ‘Artículo would display as ‘Art\\u00edculo’. Treating the item like a dictionary, I was able to extract from every paper the text and review score by using a for loop. Once texts were extracted they were lowercase and split by whitespace. To remove punctuation str.maketrans() and regular expressions were used. From here 18 English reviews were removed, reviews were filtered as lists of words for Spanish stopwords, and tokens were stemmed. I chose stemming over lemmatizing for this exercise, as I struggled to find an effective Spanish lemmatizer. Each review was saved as a tuple containing an integer, representing the review score, and a list of stemmed tokens.

I used scikit-learn CountVectorizer to vectorize the tokens. Each tuple mentioned above was split, with the review scores turning into a NumPy array, and the list of tokens being fed into another list that was fit to CountVectorizer. A bag of words representation was formed as an input by transforming the lists of tokens into a SciPy CSR matrix. Training data was split from test data by partitioning the final 32 entries, roughly 10% of the data. This split could be enhanced by shuffling the data, to deter overfitting. 5986 coefficients were fed into the model. In observing the mean squared error and coefficient of determination, models were compared after having been trained on bag of words models with and without english reviews removed, as well as with and without stemming. After every setting was given one evaluation, the model trained on unstemmed data including English reviews had the lowest mean squared error. The evaluations were not encouraging, with none of the models producing a positive R squared score. This could be addressed in the future by augmenting the amount of data available to the model. New reviews were generated, each shorter than average review length in the training data. This list of new reviews was similarly preprocessed, vectorized using CountVectorizer, and fed into LinearRegression for predictions. Pictured below are the outputs, featuring text of ratings and review score separated by asterisks:



```
artículo malo hallazgos interesantes mal uso tiempo lectura ***** 0.9711
revelador disfruté leyendo artículo espero seguir investigando campo ***** 1.0718
tema nuevo creo calificado revisar dicho trabajo ***** 1.2323
artículo revelador parecía desorganizado autor mal escritor buen científico ***** 0.6704
```

Approaching task 2, I had planned on creating an equation to plot use of tokens in different distributional environments for interpretation. Investigating the functions of the LinearRegression and CountVectorizer, I realized it would be possible to zip a list of feature names with coefficient weights. After lambda sorting this list by values of coefficient weights, I was able to print the top 10 and bottom 10 entries of this list:



```
most important negative features:
1 existen cof -0.5856475643672974
2 nulo cof -0.5382743632087502
3 explicitas cof -0.5328413048846952
4 científicas cof -0.46548046023751233
5 científico cof -0.42868438362720895
6 sido cof -0.39537620328980067
7 utilizados cof -0.39383765888173855
8 solución cof -0.39227254662436933
9 contribución cof -0.38731215696423715
10 bases cof -0.3698755115696495

most important positive features:
1 alto cof 0.38875321543734687
2 prototipo cof 0.3374670086579039
3 algún cof 0.30779386942022424
4 mostrar cof 0.28733161973970556
5 dejaría cof 0.27591977127066836
6 tecnologías cof 0.2714068894230979
7 trata cof 0.26592835492119865
8 interesante cof 0.2599666482908676
9 buen cof 0.25471348617078743
10 gustado cof 0.24883868999282932
```