CAPSTONE PROJECT - NLP APPLICATIONS
SENTIMENT ANALYSIS

Dataset: Consumer Reviews of Amazon Products

Objective:

Predict the sentiment of product reviews by:
- giving its sentiment label
- Giving its polarity score

Compare the similarity of two product reviews

1. Description of the Dataset used

This dataset contains consumer reviews for Amazon products like Kindle, Fire TV Stick, amazon marketplace and more. The dataset includes basic product information such as ratings, reviews in text format, the source of the reviews, Id and much more. It has 28332 rows and 24 columns.
For our analysis we required only one column named ‘reviews.text’ as the other columns will not be useful to us in this context.
The dataset is available on kaggle with url Consumer Reviews of Amazon Products (kaggle.com) .

2. Details of the Preprocessing Steps

I used Jupyter Notebook for my scripts, to prepare the text for analysis here are the steps I have followed:

a. Import libraries such as pandas and spacy
b. Import the dataset
    - I first downloaded the dataset from Kaggel and saved it in my PC.
    - Load the dataset in the editor
c. Explore the dataset by:
    - Printing the first 5 rows and all the columns to check the kind of data we have
    - Checking the size (total rows and columns)
    - Checking missing values
d. Select the column that we need for our analysis
    - Select our column ‘reviews.text’ and remove all missing values

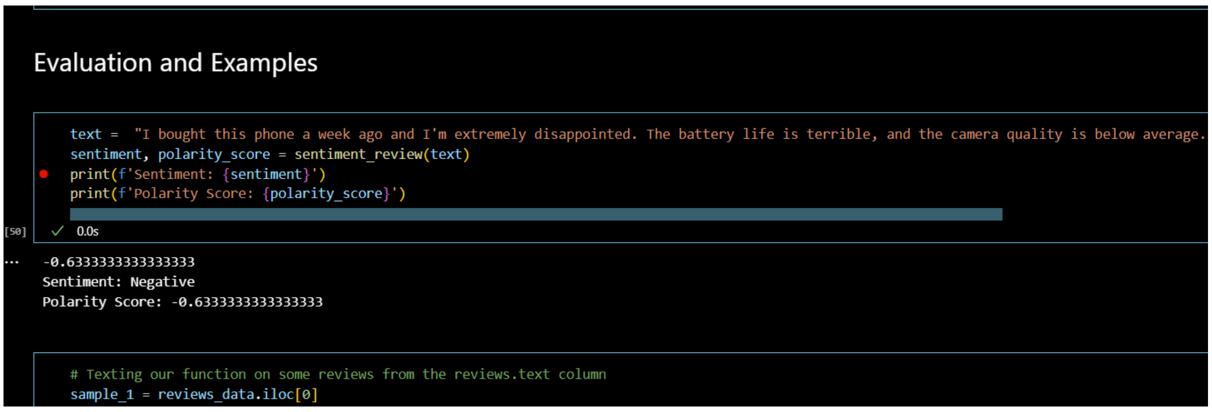
- Check the new lengths
- Check sample data in our column

e. The NLP preprocessing
- Load the spacy medium language model ‘en_core_web_md’
- Define a function to do the preprocessing: it takes a review as input, passes it through the language model, cleans it and returns a string that is tokenized, lemmatized with stop words and punctuation removed.

After these steps the reviews are now ready for analysis.

3. Evaluation of Results

Below is a snippet (*Figure 1*) taken from my model. It's a negative review taken online that I used for testing. I know it’s negative and I want to find out if the model can detect that. It shows that the polarity is -0.6333333 and sentiment is Negative, which confirms the nature of the review.



Evaluation and Examples

```
text =  "I bought this phone a week ago and I'm extremely disappointed. The battery life is terrible, and the camera quality is below average.
sentiment, polarity_score = sentiment_review(text)
print(f'Sentiment: {sentiment}')
print(f'Polarity Score: {polarity_score}')
```

[50] ✓ 0.0s

```
-0.6333333333333333
Sentiment: Negative
Polarity Score: -0.6333333333333333
```

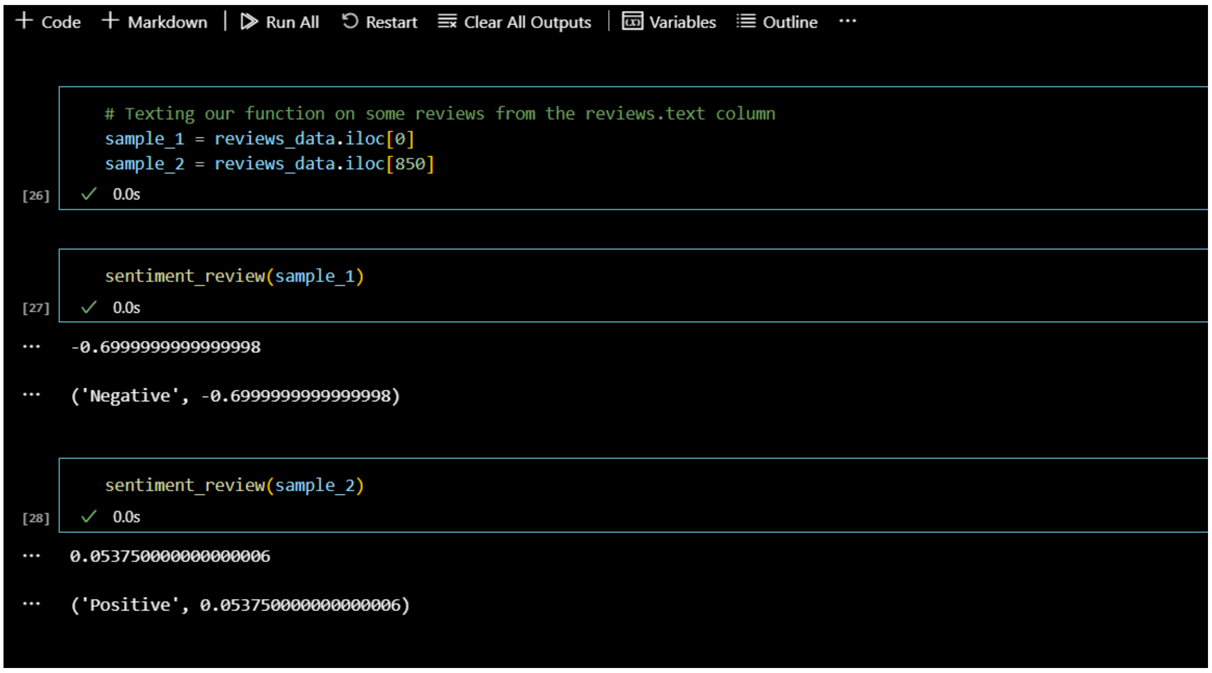
```
# Texting our function on some reviews from the reviews.text column
sample_1 = reviews_data.iloc[0]
```

Figure 1

4. Insights into the Model’s Strengths and Limitations

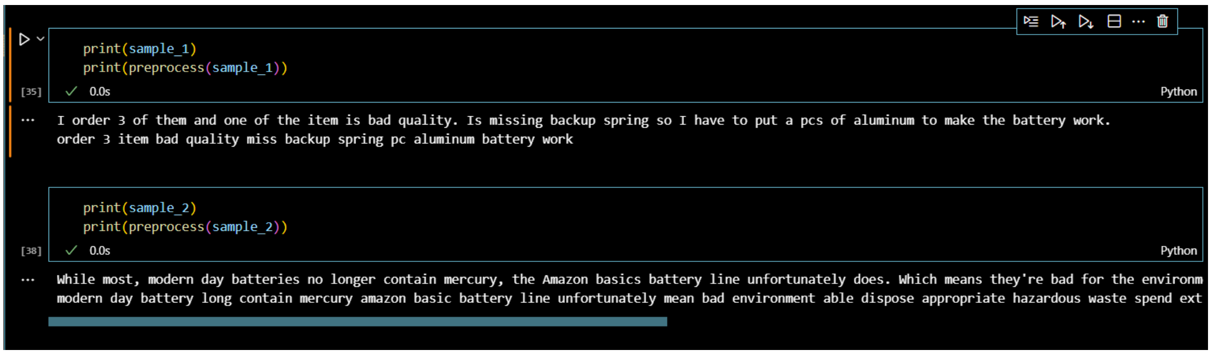
This project has highlighted a lot of interesting results. I have tested the program on a few reviews and the results confirmed the nature of the reviews when I printed them out to check. But some of the results were quite interesting to interpret such as the snippets below (*Figure 2, 3 and 4*).

The 2 reviews selected are both negative reviews but the program gave the second one (Sample_2) a positive score, although nearing a neutral review at ‘0.05375’ while the first one has a negative score of -0.6999999.
I believe this kind of limitation could be explained by the fact that the language model checks each word and decides if it’s negative or positive and subsequently a positive word in a negative sentence could impact the score.



```
# Texting our function on some reviews from the reviews.text column
sample_1 = reviews_data.iloc[0]
sample_2 = reviews_data.iloc[850]
```
```
[26] ✓ 0.0s
```
```
sentiment_review(sample_1)
```
```
[27] ✓ 0.0s
-0.6999999999999998
('Negative', -0.6999999999999998)
```
```
sentiment_review(sample_2)
```
```
[28] ✓ 0.0s
0.053750000000000006
('Positive', 0.053750000000000006)
```

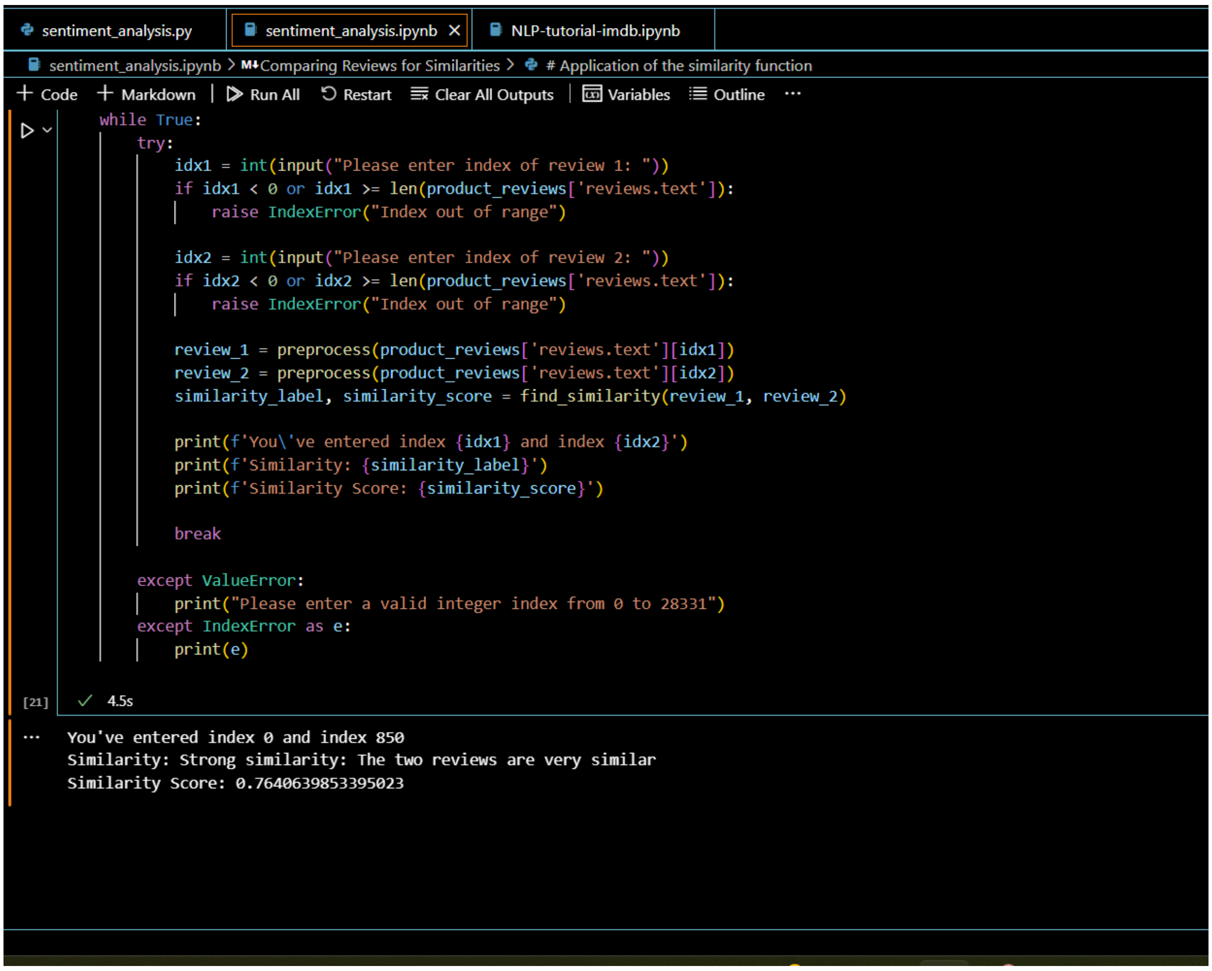
Figure 2



```
print(sample_1)
print(preprocess(sample_1))
```
```
[35] ✓ 0.0s
I order 3 of them and one of the item is bad quality. Is missing backup spring so I have to put a pcs of aluminum to make the battery work.
order 3 item bad quality miss backup spring pc aluminum battery work
```
```
print(sample_2)
print(preprocess(sample_2))
```
```
[38] ✓ 0.0s
While most, modern day batteries no longer contain mercury, the Amazon basics battery line unfortunately does. Which means they're bad for the environm
modern day battery long contain mercury amazon basic battery line unfortunately mean bad environment able dispose appropriate hazardous waste spend ext
```

Figure 3

I used the same 2 reviews from our dataset to compare them using the similarity() function and the result was exciting (see *Figure 4*). We got a similarity score of 0.76406 which means that they're very similar and when printed out we can see that they're similar indeed.


sentiment_analysis.py
sentiment_analysis.ipynb
NLP-tutorial-imdb.ipynb
sentiment_analysis.ipynb > M↓Comparing Reviews for Similarities > # Application of the similarity function
+ Code + Markdown | Run All Restart Clear All Outputs | Variables Outline ···
[21] ✓ 4.5s

```python
while True:
    try:
        idx1 = int(input("Please enter index of review 1: "))
        if idx1 < 0 or idx1 >= len(product_reviews['reviews.text']):
            raise IndexError("Index out of range")

        idx2 = int(input("Please enter index of review 2: "))
        if idx2 < 0 or idx2 >= len(product_reviews['reviews.text']):
            raise IndexError("Index out of range")

        review_1 = preprocess(product_reviews['reviews.text'][idx1])
        review_2 = preprocess(product_reviews['reviews.text'][idx2])
        similarity_label, similarity_score = find_similarity(review_1, review_2)

        print(f'You\'ve entered index {idx1} and index {idx2}')
        print(f'Similarity: {similarity_label}')
        print(f'Similarity Score: {similarity_score}')

        break

    except ValueError:
        print("Please enter a valid integer index from 0 to 28331")
    except IndexError as e:
        print(e)
```

```
You've entered index 0 and index 850
Similarity: Strong similarity: The two reviews are very similar
Similarity Score: 0.7640639853395023
```

Figure 4