1.Data Collection

Name: **amazon_uk_shoes_products_dataset_2021_12.csv**

We collected the dataset from a third party website which is data.world in CSV file format. It has 11 columns but we will work on 2 of them, review_text and review_rating. The dataset contains 6844 record. review_rating column has five classes from 1 to 5.

2.Data Preparation

2.1. First

- Installing PySpark and its related packages in Colab notebook.
- Reading the data using the appropriate functions and parameters as coded in the notebook.
- After printing the schema of the data, we figured out two features that interest us the most in our sentiment analysis problem which are review_text and review_rating.

2.2. Second

- Time to clean 6844 data entries.
- Fortunately no duplicated rows have been found, but unfortunately some **nulls** have been found in either
- review_text or review_rating so we dropped their rows after applying a filter using **filter()**.

```
#drop null values at review_text and review_rating
df = df.filter(df.review_text.isNotNull()).drop()
df = df.filter(df.review_rating.isNotNull()).drop()
```

- We arranged the data ascending according to review_rating

```
#order the data ascending according to review_rating
df = df.orderBy(df.review_rating.asc())
```

- Now, we have 6807 data entries
- Another issue is the review_rating column has some **nonnumeric values**!

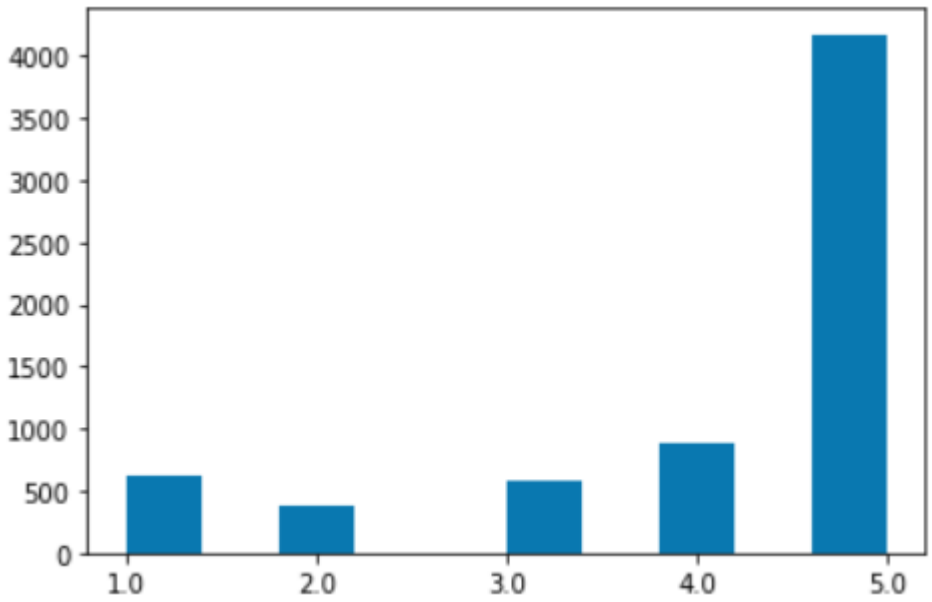
```
d! ...|                    3.0|
est...|                    5.0|
er ...| maybe even a 5. ...|
 la...|                    5.0|
s r...|                    3.0|
------+-----------------------+
```

- we exclusive that text rows from our dataframe by selecting only the rows with numeric values ranging from 1 to 5.

```
| (df.review_rating == 4) | (df.review_rating == 5)]
```

```
#removing noisy data at review_rating
df = df[(df.review_rating == 1) | (df.review_rating == 2) | (df.review_rating == 3)
```

- We have 5 classes, this is the distribution of these 5 classes on the dataset.



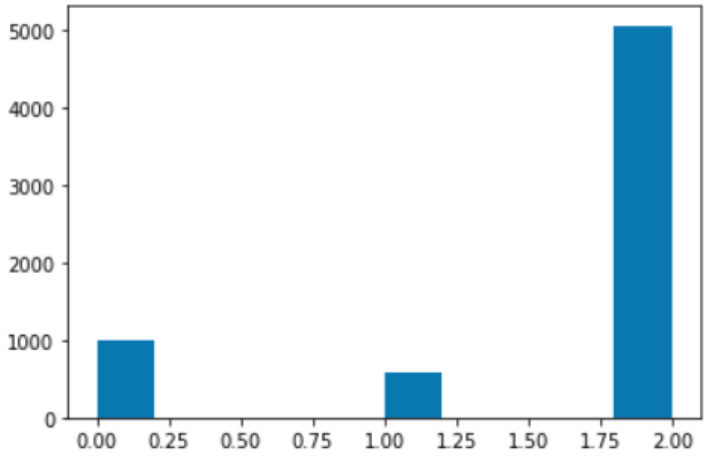
- After that, we re-divide the column into three categories 0, 1, 2 as interpretation to negative, neutral and positive respectively and cast the value to be integer.

```
#encoded the review data into zeros and ones
#zero indicates to negative
#one indicates to neutral
#two indicates to positive
df =df.select('review_text','review_rating').replace(
    ['1.0','2.0','3.0','4.0', '5.0'],['0.0','0.0','1.0','2.0', '2.0'])
```

- We have to casting the data from string to integers so we can do analysis on it

```
#casting the review rating data from string to integer
df = df.withColumn("review_rating",df.review_rating.cast('int'))
```

- After We encoded the data to 3 classes 0,1,2



3.Data Analysis

- Splitting the data into training and testing with 70% and 30% respectively.
- Performing some neutral language processing (NLP) over the review_text column.
    - Tokenization: that means converting the text into words
    - Removing the stop words: like “a”, “the”, and the other words which don't have weight in our classification process.
    - Converting the resulting words into a vector of size 100
- Applying Logistic Regression (LR) as a supervised learning model

- Evaluating the model by measuring the accuracy, precision, recall and F1 Score.

4.Model Evaluation

- We evaluated the model using classification metrics, accuracy, precision, recall and f1-score. We got these metrics results on both training and testing data.

4.1. On training data:

```
accuracy: 0.7736
precision: 0.6987
recall: 0.7736
F1 score: 0.7079
```

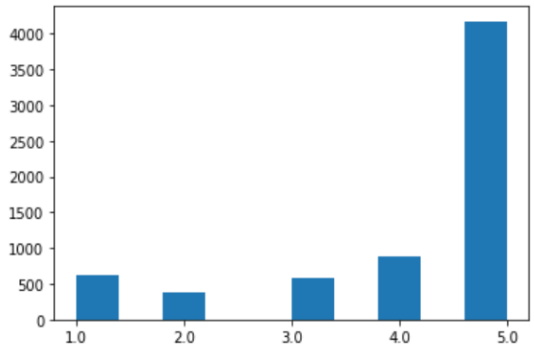
4.2. On testing data:

```
accuracy: 0.7547
precision: 0.6498
recall: 0.7547
F1 score: 0.6808
```

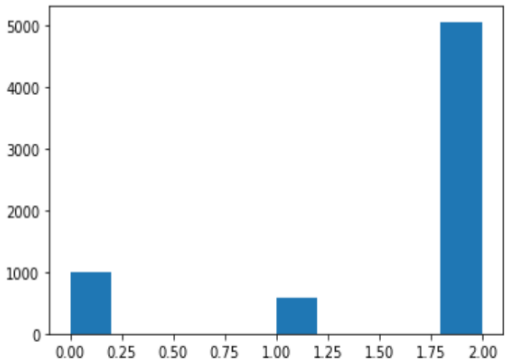
5. Data Visualization

- We applied the **histogram** in the data preparation phase to see the weight of every rate when the categories were 5 and after dividing them to 3 only.

```
# Before
#plot review_rating
plt.hist(df.toPandas()['review_rating'])
plt.show()
```



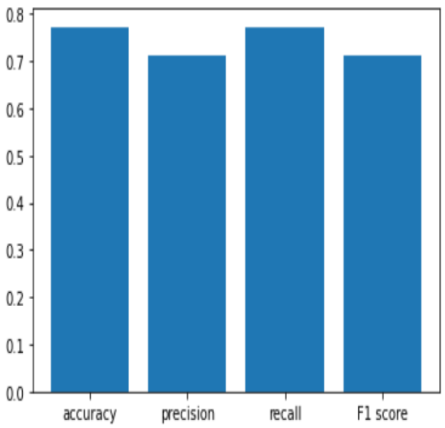

```
# After
#plot the data
plt.hist(df.toPandas()['review_rating'])
plt.show()
```



- We used the **bar chart** in the evaluation phase to visualize the percentages of our different measures and see how far they differ from each other both in the training and testing.

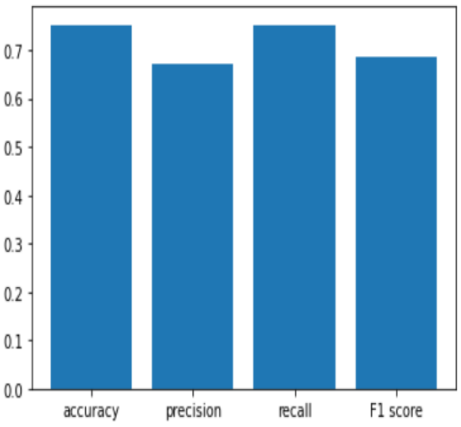
```
# Training
#plot train metrics
list_train = {'accuracy':accuracy,'precision':precision,
x= list(list_train.keys())
y= list(list_train.values())
plt.bar(x,y)
```

```
<BarContainer object of 4 artists>
```




```
# Testing (Evalution)
#plot test metrics
list_train = {'accuracy':accuracy,'precision':precision,
x= list(list_train.keys())
y= list(list_train.values())
plt.bar(x,y)
```

```
<BarContainer object of 4 artists>
```



6.Findings and Discussion

- According to train and test evaluation metrics results, it is obvious that there is no underfitting or overfitting, but the results still not good enough because of the imbalanced dataset. Class 5 has the majority of records in the dataset.

6.1. First Trial

- We mapped the 5 classes into 2 classes only and did a binary classification.
    - Classes 1,2,3 represented the negative class
    - Classes 4,5 represented the positive class

- The accuracy increased but the algorithm was biased towards positive class because it has class 5 (which has the majority of the data).

6.2.Second Trial

- we used 1,2,3,4 as negative class and only 5 as positive,But it was a trivial solution because 4 can not be a negative class

6.3.Third Trial

- we used 3 classes:
    - 1,2 negative
    - 3 neutral
    - 4,5 positive
- And this was **the best outcome** we have, by work as a multiclass classification.