28.1 Dataset Overview: Amazon Fine Food Reviews (EDA)

Important Links

Data Source

https://www.kaggle.com/snap/amazon-fine-food-reviews

Blog on EDA

https://nycdatascience.com/blog/student-works/amazon-fine-foods-visualization/

Ipython Notebook

https://colab.research.google.com/drive/1FCQfzaYb-yyDIuF5RKTFa0VPnsYJZyR8?usp=drive_open

Note

In order to download the dataset from Kaggle, you first have to login to Kaggle and then download it.

Dataset Description

The Amazon Fine Food Reviews dataset consists of reviews of fine foods from Amazon.

Number of reviews: 568,454

Number of users: 256,059

Number of products: 74,258

Timespan: Oct 1999 - Oct 2012

Number of Attributes/Columns in data: 10

Attribute Information

1. Id - Unique Identifier given to each review
2. ProductId - Unique Identifier for the product
3. UserId - Unique Identifier for the user
4. ProfileName - Customer/User's name
5. HelpfulnessNumerator - number of users who found the review helpful
6. HelpfulnessDenominator - number of users who indicated whether they found the review helpful or not
7. Score - rating between 1 and 5
8. Time - timestamp at which the review was posted on the website
9. Summary - brief summary of the review
10. Text - text of the review

Problem Objective

We have to determine if the given review is positive or negative.

How to categorize a given review as positive/negative?

Categorization of the reviews is done on the basis of the 'Score' column values. If a given review has a score value of 4 (or) 5, then it is considered as a Positive review, and if the score value is 1 (or) 2, then it is considered as a Negative Review.

A score value of 3, is considered to be a neutral review, but for the time being, we shall ignore the neutral reviews because, considering the neutral reviews will pose our problem as a multi-class classification. But as we are looking to pose our problem as a binary classification problem, we consider only the positive and the negative reviews.

Loading the Dataset

The dataset is given in two forms.

1) CSV File format (with the name Reviews.csv)
2) SQLite Database

The dataset is already present in the SQLite database, so that it can be directly loaded into a pandas dataframe, and from there we can perform analysis and visualizations on the data easily.

But in case, if you do not want to perform any analysis or visualizations on the data, but just want to have a look at the data, then instead of loading the data into a pandas dataframe every time, you can directly look into the 'Reviews.csv' file.

Below is the line of code that was discussed at the timestamp 18:40, which establishes the connection with the database.

```
con = sqlite3.connect('database.sqlite')
```

Note: If the 'database.sqlite' file is present in the same folder where your ipython notebook is present, then you can pass only the file name as a value to the sqlite3.connect(). If not, then you have to specify the entire path as well.

Below is the line of code that was discussed at the timestamp 19:15, which loads the dataset into a pandas dataframe.

```
filtered_data = pd.read_sql_query(""" SELECT * FROM Reviews WHERE Score != 3 LIMIT 5000""", con)
```

Note: Here as we are loading the dataset from the database, we are using the method pandas.read_sql_query() and are passing the SQL query and the sqlite connection object. If we had to load the dataset from a CSV file, then we would have used pandas.read_csv() with the path to the CSV file as an argument.

The data in our database is stored in the table 'Reviews', and we are selecting only those rows whose score value is not equal to 3.

The below code snippet was discussed starting from the timestamp 20:33 and here we are relabelling the scores 4 and 5 as 'Positive', and 1 and 2 as 'Negative'.

```
# Give reviews with Score>3 a positive rating, and reviews with a score<3 a negative rating.
def partition(x):
    if x < 3:
        return 0
    return 1

#changing reviews with score less than 3 to be positive and vice-versa
actualScore = filtered_data['Score']
positiveNegative = actualScore.map(partition)
filtered_data['Score'] = positiveNegative
print("Number of data points in our data", filtered_data.shape)
filtered_data.head(3)
```

We are first loading the values of the 'Score' column into a series variable 'actualScore', and then using the map() function, we apply relabelling the values. The code for relabelling is defined in the function partition().