Libraries

```
In [1]: import pandas as pd
        import numpy as np
        from bs4 import BeautifulSoup
        import requests
        import time
```

Reading in Data

Product 1: Texas Instruments TI-83 Plus Graphing Calculator

```
In [2]: # initialzing df
        eBay_1 = []
        eBay_1 = pd.DataFrame(eBay_1)

        # initilazing lists
        stars = []
        content = []
        titles = []
```

```
In [3]: # Getting page base and making list of links to the reviews

        page_base = 'https://www.ebay.com/urw/Texas-Instruments-TI-83-Plus-Graphing-Calculator/p
        total_pages = 208
        links = []

        for link in range(1,total_pages):
            links.append(page_base+str(link))
```

Initially, we pulled the links directly through the html and looped through, but the initial page only provided the first 7 pages of reviews. This was the easiest way around this issue.

```
In [4]: # Filling lists with data from Web Scrape

        for page in links:

            r = requests.get(page)
            #time.sleep(1)
            soup = BeautifulSoup(r.text, 'html.parser')

            for star in soup.find_all('div', class_='ebay-review-section-l'):
                stars.append(star.find('span','star-rating')['aria-label'])

            for review_content in soup.find_all('p', class_ = 'review-item-content rvw-wrap-spac
                content.append(review_content.text)

            for review_title in soup.find_all('h3', class_ = 'review-item-title rvw-nowrap-space
                titles.append(review_title.text)
```

```
In [5]: # Putting list into Data Frame

        eBay_1['Review Title'] = titles
        #eBay['Review Content'] = content
        eBay_1['Stars'] = stars
        eBay_1['Item'] = 'Texas Instruments TI-83 Plus Graphing Calculator'
```

A user must include a title in their review, but content is not needed. This creates a different amount of content items than titles and stars. This results in us having to use titles as our source of text, as there is no way to know which reviews have content in their review or not.

Product 2: Apple AirPods Pro Left Airpod OEM Left Side Airpods Pro Only

In [6]:
```
# initialzing df
eBay_2 = []
eBay_2 = pd.DataFrame(eBay_2)

# initilazing lists
stars = []
titles = []
```

In [7]:
```
# Getting page base and making list of links to the reviews

page_base = 'https://www.ebay.com/urw/Apple-AirPods-Pro-Wireless-In-Ear-Headsets-White/p
total_pages = 230
links = []

for link in range(1,total_pages):
    links.append(page_base+str(link))
```

In [8]:
```
# Filling lists with data from Web Scrape

for page in links:

    r = requests.get(page)
    #time.sleep(1)
    soup = BeautifulSoup(r.text, 'html.parser')

    for star in soup.find_all('div', class_='ebay-review-section-l'):
        stars.append(star.find('span','star-rating')['aria-label'])

    for review_title in soup.find_all('h3', class_ = 'review-item-title rvw-nowrap-space
        titles.append(review_title.text)
```

In [9]:
```
# Putting list into Data Frame

eBay_2['Review Title'] = titles
eBay_2['Stars'] = stars
eBay_2['Item'] = 'Apple AirPods Pro Left Airpod OEM Left Side Airpods Pro Only'
```

Product 3: Sony PS5 Console w/ Blu-Ray Disc

In [10]:
```
# initialzing df
eBay_3 = []
eBay_3 = pd.DataFrame(eBay_3)

# initilazing lists
stars = []
titles = []
```

In [11]:
```
# Getting page base and making list of links to the reviews

page_base = 'https://www.ebay.com/urw/Sony-PS5-Blu-Ray-Edition-Console-White/product-rev
total_pages = 194
links = []
```

```
for link in range(1,total_pages):
    links.append(page_base+str(link))
```

In [12]:
```
# Filling lists with data from Web Scrape

for page in links:

    r = requests.get(page)
    # time.sleep(1)
    soup = BeautifulSoup(r.text, 'html.parser')

    for star in soup.find_all('div', class_='ebay-review-section-l'):
        stars.append(star.find('span','star-rating')['aria-label'])

    for review_title in soup.find_all('h3', class_ = 'review-item-title rvw-nowrap-space
        titles.append(review_title.text)
```

In [13]:
```
# Putting list into Data Frame

eBay_3['Review Title'] = titles
eBay_3['Stars'] = stars
eBay_3['Item'] = 'Sony PS5 Console w/ Blu-Ray Disc'
```

Product 4: Super Bright 90000LM LED Tactical Flashlight Zoomable With Rechargeable Battery

In [14]:
```
# initialzing df
eBay_4 = []
eBay_4 = pd.DataFrame(eBay_4)

# initilazing lists
stars = []
titles = []
```

In [15]:
```
# Getting page base and making list of links to the reviews

page_base = 'https://www.ebay.com/urw/Garberiel-AF20171221-8000-Lumens-Tactical-Flashlig
total_pages = 41
links = []

for link in range(1,total_pages):
    links.append(page_base+str(link))
```

In [16]:
```
# Filling lists with data from Web Scrape

for page in links:

    r = requests.get(page)
    #time.sleep(1)
    soup = BeautifulSoup(r.text, 'html.parser')

    for star in soup.find_all('div', class_='ebay-review-section-l'):
        stars.append(star.find('span','star-rating')['aria-label'])

    for review_title in soup.find_all('h3', class_ = 'review-item-title rvw-nowrap-space
        titles.append(review_title.text)
```

In [17]:
```
# Putting list into Data Frame

eBay_4['Review Title'] = titles
```

```
eBay_4['Stars'] = stars
eBay_4['Item'] = 'Super Bright 90000LM LED Tactical Flashlight Zoomable With Rechargeabl
```

Product 5: Canon PIXMA MG2520 All-In-One Inkjet Printer

In [18]:
```
# initialzing df
eBay_5 = []
eBay_5 = pd.DataFrame(eBay_5)

# initilazing lists
stars = []
titles = []
```

In [19]:
```
# Getting page base and making list of links to the reviews

page_base = 'https://www.ebay.com/urw/Canon-PIXMA-MG2520-All-In-One-Inkjet-Printer/produ
total_pages = 42
links = []

for link in range(1,total_pages):
    links.append(page_base+str(link))
```

In [20]:
```
# Filling lists with data from Web Scrape

for page in links:

    r = requests.get(page)
    # time.sleep(1)
    soup = BeautifulSoup(r.text, 'html.parser')

    for star in soup.find_all('div', class_='ebay-review-section-l'):
        stars.append(star.find('span','star-rating')['aria-label'])

    for review_title in soup.find_all('h3', class_ = 'review-item-title rvw-nowrap-space
        titles.append(review_title.text)
```

In [21]:
```
# Putting list into Data Frame

eBay_5['Review Title'] = titles
eBay_5['Stars'] = stars
eBay_5['Item'] = 'Canon PIXMA MG2520 All-In-One Inkjet Printer'
```

In [22]:
```
# Combining data to get one data frame

eBay = pd.concat([eBay_1, eBay_2, eBay_3, eBay_4, eBay_5], ignore_index=True)
eBay.reset_index(drop = True, inplace = True)
```

In [23]:
```
# Binning reviews for predictive modeling

is_positive = []

for stars in eBay['Stars']:
    if stars == '5 stars' or stars == '4 stars':
        is_positive.append(1)
    else:
        is_positive.append(0)

eBay['Is_Positive?'] = is_positive
```

In [24]:
```
# What the final dataframe looks like

eBay
```

Out[24]:

	Review Title	Stars	Item	Is_Positive?
0	Nice item- as we used to say: "Works fine, las...	5 stars	Texas Instruments TI-83 Plus Graphing Calculator	1
1	Cheap	3 stars	Texas Instruments TI-83 Plus Graphing Calculator	0
2	Texas Instruments TI-83 Plus Graphic Calculator	4 stars	Texas Instruments TI-83 Plus Graphing Calculator	1
3	TI-83	4 stars	Texas Instruments TI-83 Plus Graphing Calculator	1
4	Handy calculator, solid and functional	4 stars	Texas Instruments TI-83 Plus Graphing Calculator	1
...	...	...	...	...
7098	is so so	5 stars	Canon PIXMA MG2520 All-In-One Inkjet Printer	1
7099	Nice printer great price	5 stars	Canon PIXMA MG2520 All-In-One Inkjet Printer	1
7100	like it	5 stars	Canon PIXMA MG2520 All-In-One Inkjet Printer	1
7101	PERFECT	5 stars	Canon PIXMA MG2520 All-In-One Inkjet Printer	1
7102	Excelente	5 stars	Canon PIXMA MG2520 All-In-One Inkjet Printer	1

7103 rows × 4 columns

Calculating Descriptive Statistics on the Data

Importing Useful Functions

In [26]:

```python
from string import punctuation
from nltk.corpus import stopwords
import re
import emoji

# Some punctuation variations
punctuation = set(punctuation) # speeds up comparison
tw_punct = punctuation - {"#"}

# Stopwords
sw = stopwords.words("english")

# Two useful regex
whitespace_pattern = re.compile(r"\s+")
hashtag_pattern = re.compile(r"^#[0-9a-zA-Z]+")

# full set of emojis
all_language_emojis = set()

for country in emoji.UNICODE_EMOJI :
    for em in emoji.UNICODE_EMOJI[country] :
        all_language_emojis.add(em)
```

In [25]:
```
#function to calculate descriptive stats

def descriptive_stats(tokens, num_tokens = 5, verbose=True) :
    """
        Given a list of tokens, print number of tokens, number of unique tokens,
        number of characters, lexical diversity (https://en.wikipedia.org/wiki/Lexical_d
        and num_tokens most common tokens. Return a list with the number of tokens, numb
        of unique tokens, lexical diversity, and number of characters.

    """

    # Fill in the correct values here.
    tokes = tokens.split()
    num_tokens = sum(map(len, (s.split() for s in tokes)))
    num_unique_tokens = len(set(w.lower() for w in tokes))
    lexical_diversity = num_unique_tokens / num_tokens
    num_characters = sum(list(map(len, tokes)))

    if verbose :
        print(f"There are {num_tokens} tokens in the data.")
        print(f"There are {num_unique_tokens} unique tokens in the data.")
        print(f"There are {num_characters} characters in the data.")
        print(f"The lexical diversity is {lexical_diversity:.3f} in the data.")

        # print the five most common tokens

    return([num_tokens, num_unique_tokens,
            lexical_diversity,
            num_characters])
```

In [27]:
```
#other useful functions from previous modules
def is_emoji(s):
    return(s in all_language_emojis)

def contains_emoji(s):

    s = str(s)
    emojis = [ch for ch in s if is_emoji(ch)]

    return(len(emojis) > 0)


def remove_stop(tokens) :
    # modify this function to remove stopwords
    tokens_wo_sw = []
    for w in tokens:
        if w.lower() not in sw:
            tokens_wo_sw.append(w)
    return(tokens_wo_sw)

def remove_punctuation(text, punct_set=tw_punct):
    for ele in text:
        if ele in punct_set:
            text = text.replace(ele, "")
    return(text)

def tokenize(text) :
    """ Splitting on whitespace rather than the book's tokenize function. That
        function will drop tokens like '#hashtag' or '2A', which we need for Twitter. ""

    # modify this function to return tokens
    text = text.split()
    return(text)

def prepare(text, pipeline) :
    tokens = str(text)

    for transform in pipeline :
        tokens = transform(tokens)

    return(tokens)
```

Clean and Tokenize the Data

In [33]:
```
my_pipeline = [str.lower, remove_punctuation, tokenize, remove_stop]
```

In [42]:
```
clean_eBay = []

for review in range(len(eBay)):
    temp = prepare(eBay['Review Title'][review], my_pipeline)
    clean_eBay.append(temp)
```

Run Descriptive_Stats on our Dataset

In [45]:
```
descriptive_stats(str(clean_eBay), verbose=True)
```

```
There are 16880 tokens in the data.
There are 3480 unique tokens in the data.
There are 160628 characters in the data.
The lexical diversity is 0.206 in the data.
```

Out[45]:
```
[16880, 3480, 0.20616113744075829, 160628]
```

In [274…
```
# Reading DataFrame into CSV

eBay.to_csv('C:/Users/mzazu/OneDrive/Documents/USD papers/509/eBay.csv', index=False)
```