In [1]:
```
# Import necessary libraries

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt



#Read the data

df = pd.read_csv(r"C:\Users\Vaish\Downloads\archive (1)\Reviews.csv", nrows=500)



# Look at the top 5 rows of the data

df.head(3)
```

Out[1]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessD
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 10 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   Id                      500 non-null    int64 
 1   ProductId               500 non-null    object
 2   UserId                  500 non-null    object
 3   ProfileName             500 non-null    object
 4   HelpfulnessNumerator    500 non-null    int64 
 5   HelpfulnessDenominator  500 non-null    int64 
 6   Score                   500 non-null    int64 
 7   Time                    500 non-null    int64 
 8   Summary                 500 non-null    object
 9   Text                    500 non-null    object
dtypes: int64(5), object(5)
memory usage: 39.2+ KB
```

In [5]:
```
#summary of reviews
df.Summary.head()
```

Out[5]:
```
0    Good Quality Dog Food
1        Not as Advertised
2    "Delight" says it all
3           Cough Medicine
4              Great taffy
Name: Summary, dtype: object
```

In [7]:
```
df.Text.head()
```

Out[7]:
```
0    I have bought several of the Vitality canned d...
1    Product arrived labeled as Jumbo Salted Peanut...
2    This is a confection that has been around a fe...
3    If you are looking for the secret ingredient i...
4    Great taffy at a great price.  There was a wid...
Name: Text, dtype: object
```

In [9]:
```
#!pip install textblob
#!python -m textblob.download_corpora
```

In [11]:
```
# Import libraries
import pandas as pd
from nltk.corpus import stopwords
from textblob import TextBlob, Word

# Sample DataFrame
#df = pd.DataFrame({'Text': ["This is an exmple sentence with some erors."]})

# Lower casing and removing punctuations
df['Text'] = df['Text'].apply(lambda x: " ".join(x.lower() for x in x.split()))
df['Text'] = df['Text'].str.replace(r'[^\w\s]', ' ', regex=True)

# Removal of stop words
stop = set(stopwords.words('english'))
df['Text'] = df['Text'].apply(lambda x: " ".join(x for x in x.split() if x not in s
```

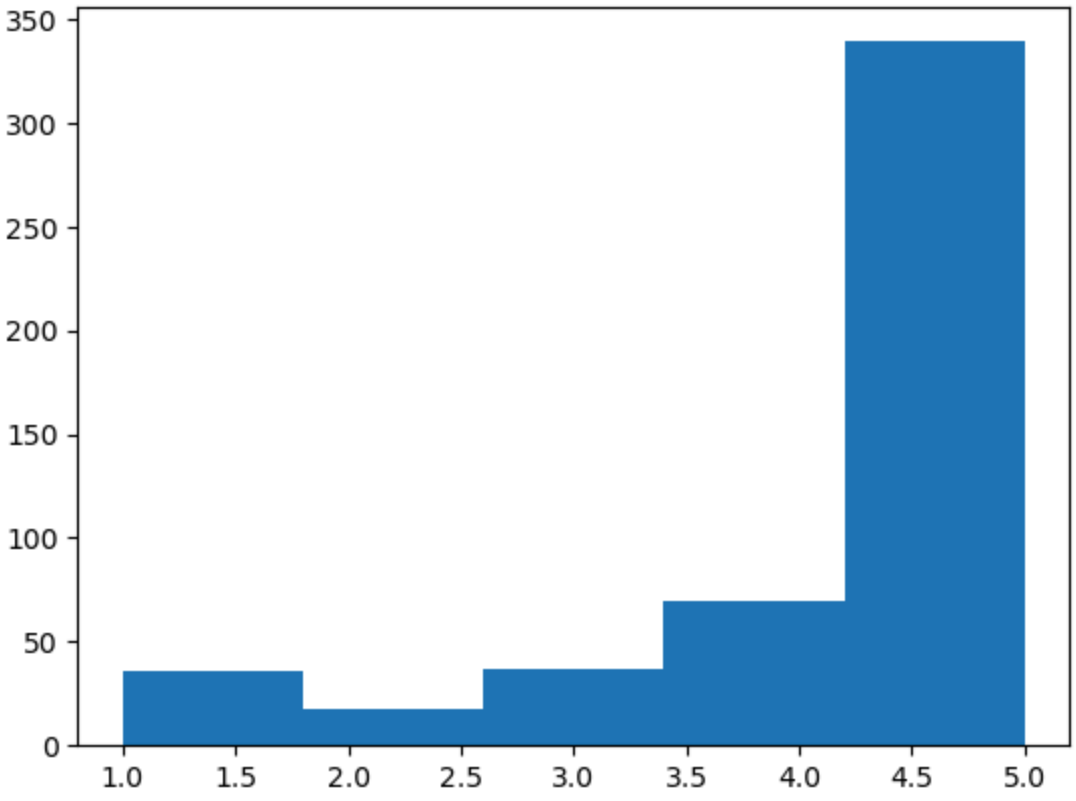
```
# Spelling correction
df['Text'] = df['Text'].apply(lambda x: str(TextBlob(x).correct()))

# Lemmatization
df['Text'] = df['Text'].apply(lambda x: " ".join([Word(word).lemmatize() for word i

# Display first few rows
print(df.Text.head())
```

```
0    bought several vitality canned dog food produc...
1    product arrived labelled lumbo halted peanut p...
2    connection around century light pillow city ge...
3    looking secret ingredient robitussin believe f...
4    great staff great price wide assortment mummy ...
Name: Text, dtype: object
```

In [13]:
```
# Create a new data frame "reviews" to perform exploratory data analysis upon that
reviews = df
# Dropping null values
reviews.dropna(inplace=True)
# The histogram reveals this dataset is highly unbalanced towards high rating.
reviews.Score.hist(bins=5,grid=False)
plt.show()
print(reviews.groupby('Score').count().Id)
```




```
Score
1     36
2     18
3     37
4     70
5    339
Name: Id, dtype: int64
```

In [15]:
```
score_1 = reviews[reviews['Score'] == 1].sample(n=18)

score_2 = reviews[reviews['Score'] == 2].sample(n=18)

score_3 = reviews[reviews['Score'] == 3].sample(n=18)

score_4 = reviews[reviews['Score'] == 4].sample(n=18)

score_5 = reviews[reviews['Score'] == 5].sample(n=18)
```

In [17]:
```
# Here we recreate a 'balanced' dataset.

reviews_sample = pd.concat([score_1,score_2,score_3,score_4,score_5],axis=0)

reviews_sample.reset_index(drop=True,inplace=True)



# Printing count by 'Score' to check dataset is now balanced.

print(reviews_sample.groupby('Score').count().Id)
```

```
Score
1    18
2    18
3    18
4    18
5    18
Name: Id, dtype: int64
```

In [23]:
```
from wordcloud import WordCloud
```

In [25]:
```
# Create wordclouds

wordcloud_negative = WordCloud(background_color='white').generate(negative_reviews_

wordcloud_positive = WordCloud(background_color='white').generate(positive_reviews_

# Plot

fig = plt.figure(figsize=(10,10))

ax1 = fig.add_subplot(211)

ax1.imshow(wordcloud_negative,interpolation='bilinear')

ax1.axis("off")

ax1.set_title('Reviews with Negative Scores',fontsize=20)

ax2 = fig.add_subplot(212)

ax2.imshow(wordcloud_positive,interpolation='bilinear')

ax2.axis("off")
```

```
ax2.set_title('Reviews with Positive Scores',fontsize=20)

plt.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[25], line 3
      1 # Create wordclouds
----> 3 wordcloud_negative = WordCloud(background_color='white').generate(negative_r
eviews_str)
      5 wordcloud_positive = WordCloud(background_color='white').generate(positive_r
eviews_str)
      7 # Plot

NameError: name 'negative_reviews_str' is not defined
```

In [ ]:
```
!pip install vaderSentiment
```

In [37]:
```
import seaborn as sns

from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

plt.style.use('fivethirtyeight')

# Function for getting the sentiment

cp = sns.color_palette()



analyzer = SentimentIntensityAnalyzer()



# Generating sentiment for all the sentence present in the dataset

emptyline=[]

for row in df['Text']:

    vs=analyzer.polarity_scores(row)

    emptyline.append(vs)
```

In [29]:
```
# Creating new dataframe with sentiments

df_sentiments=pd.DataFrame(emptyline)

df_sentiments.head()
```

Out[29]:

	neg	neu	pos	compound
0	0.000	0.503	0.497	0.9413
1	0.258	0.644	0.099	-0.5719
2	0.134	0.602	0.264	0.7880
3	0.000	0.854	0.146	0.4404
4	0.000	0.455	0.545	0.9186

In [31]:
```
# Merging the sentiments back to reviews dataframe

df_c = pd.concat([df.reset_index(drop=True), df_sentiments], axis=1)

df_c.head(3)
```

Out[31]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessD
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	

In [43]:
```
result=df_c['Sentiment'].value_counts()

print(result)

result.plot(kind='bar', rot=30)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File ~\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3805, in Index.get_loc(self, key)
   3804 try:
-> 3805     return self._engine.get_loc(casted_key)
   3806 except KeyError as err:

File index.pyx:167, in pandas._libs.index.IndexEngine.get_loc()

File index.pyx:196, in pandas._libs.index.IndexEngine.get_loc()

File pandas\\_libs\\hashtable_class_helper.pxi:7081, in pandas._libs.hashtable.PyObjectHashTable.get_item()

File pandas\\_libs\\hashtable_class_helper.pxi:7089, in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'Sentiment'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Cell In[43], line 1
----> 1 result=df_c['Sentiment'].value_counts()
      3 print(result)
      5 result.plot(kind='bar', rot=30)

File ~\anaconda3\Lib\site-packages\pandas\core\frame.py:4102, in DataFrame.__getitem__(self, key)
   4100 if self.columns.nlevels > 1:
   4101     return self._getitem_multilevel(key)
-> 4102 indexer = self.columns.get_loc(key)
   4103 if is_integer(indexer):
   4104     indexer = [indexer]

File ~\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3812, in Index.get_loc(self, key)
   3807     if isinstance(casted_key, slice) or (
   3808         isinstance(casted_key, abc.Iterable)
   3809         and any(isinstance(x, slice) for x in casted_key)
   3810     ):
   3811         raise InvalidIndexError(key)
-> 3812     raise KeyError(key) from err
   3813 except TypeError:
   3814     # If we have a listlike key, _check_indexing_error will raise
   3815     #  InvalidIndexError. Otherwise we fall through and re-raise
   3816     #  the TypeError.
   3817     self._check_indexing_error(key)

KeyError: 'Sentiment'
```

In [ ]: