Data Science

Data Mining Techniques

Text Mining

Task 3:

Text Mining

Scenario:

Imagine you are a data analyst at a leading online retail company. Your company receives a vast amount of customer feedback through various channels like emails, reviews, and social media. Your task is to analyse the text data from these feedback sources to identify patterns, extract helpful knowledge, and support decision-making processes to improve customer satisfaction.

Data Pre-processing

i. Collect and pre-process a dataset of any customer feedback text (minimum 1000 feedback). You can take any data set of your choice.

Note: downloaded and used the below dataset from kaggle.com website.

Data source: https://www.kaggle.com/datasets/harshalhonde/walmart-reviews-dataset

In order to prepare unstructured text data for analysis, text preprocessing—a crucial stage in natural language processing (NLP)—involves cleansing and transforming the data. Tokenization, stemming, lemmatization, stop-word elimination, and part-of-speech tagging are all included.

Using the Walmart_reviews_data.csv file and checking the no of rows and columns.

```
In [14]: import pandas as pd

In [15]: df = pd.read_csv('D:\Walmart_reviews_data.csv')

In [16]: print(df.shape)

(300, 6)
```

Finding data types of each column.

```
In [17]: print(df.dtypes)

name           object
location       object
Date           object
Rating          int64
Review         object
Image_Links    object
dtype: object
```

Checking if there are any null values available or not.

```
In [18]: print(df.isnull().sum())

name           0
location       0
Date           0
Rating         0
Review         0
Image_Links    0
dtype: int64
```

Removing unwanted columns.

```
In [19]: df.drop(['Image_Links'], axis = 1, inplace = True)

In [20]: print(df.shape)

(300, 5)
```

Checking more information about source data.

```
In [21]: print(df.info)
```

```
<bound method DataFrame.info of          name             location                      Date  Rating  \
0          A.          Ankeny, IA  Reviewed Sept. 13, 2023       1
1       DONNA         Phoenix, AZ   Reviewed Sept. 2, 2023       1
2        Jill     Baton Rouge, LA   Reviewed Aug. 28, 2023       1
3     Sukanya          Maumee, OH   Reviewed Aug. 25, 2023       1
4     Tiffany      Laurinburg, NC   Reviewed Aug. 18, 2023       1
..        ...                 ...                      ...     ...
295   Melissa       Texas City, TX   Reviewed Aug. 15, 2023       1
296     David  West Palm Beach, FL    Reviewed Aug. 8, 2023       1
297       Jay         Torrance, CA   Reviewed July 29, 2023       1
298   Theresa          Chester, PA   Reviewed July 27, 2023       1
299     Jenna       Waterville, OH   Reviewed July 21, 2023       1

                                                Review
0    The customer service is very bad. I bought a B...
1    I have attempted to put this review on Walmart...
2    I have been a Walmart plus member for years no...
3    I refused a living room set that they sent one...
4    Beware!! Practices discriminatory/preferential...
..                                                 ...
295  I have spent hours on the phone with no resolu...
296  Reading the previous complaints, I can only co...
297  While it may be difficult to not shop at Walma...
298  I updated my phone number and I was trying to ...
299  I bought a 75 inch tv online using Walmart. Ri...

[300 rows x 5 columns]>
```

Viewing last 5 rows,

```
In [22]: print(df.tail())
```

```
        name             location                     Date  Rating  \
295  Melissa        Texas City, TX   Reviewed Aug. 15, 2023       1
296    David   West Palm Beach, FL    Reviewed Aug. 8, 2023       1
297      Jay          Torrance, CA   Reviewed July 29, 2023       1
298  Theresa           Chester, PA   Reviewed July 27, 2023       1
299    Jenna        Waterville, OH   Reviewed July 21, 2023       1

                                                Review
295  I have spent hours on the phone with no resolu...
296  Reading the previous complaints, I can only co...
297  While it may be difficult to not shop at Walma...
298  I updated my phone number and I was trying to ...
299  I bought a 75 inch tv online using Walmart. Ri...
```

Converting capital letters into lower case and rechecking last 5 rows,

```
In [23]: df['Review'] = df['Review'].str.lower()
```

```
In [24]: print(df.tail())
```

```
        name             location                      Date  Rating  \
295  Melissa        Texas City, TX  Reviewed Aug. 15, 2023       1
296    David  West Palm Beach, FL   Reviewed Aug. 8, 2023       1
297      Jay         Torrance, CA  Reviewed July 29, 2023       1
298  Theresa          Chester, PA  Reviewed July 27, 2023       1
299    Jenna       Waterville, OH  Reviewed July 21, 2023       1

                                                Review
295  i have spent hours on the phone with no resolu...
296  reading the previous complaints, i can only co...
297  while it may be difficult to not shop at walma...
298  i updated my phone number and i was trying to ...
299  i bought a 75 inch tv online using walmart. ri...
```

Removing punctuations from the "Review" column data and rechecking the last 5 rows of the same column.

```
In [25]: import string
         df['Review'] = df['Review'].str.translate(str.maketrans('','', string.punctuation))
```

```
In [26]: print(df.tail())
```

```
        name             location                      Date  Rating  \
295  Melissa        Texas City, TX  Reviewed Aug. 15, 2023       1
296    David  West Palm Beach, FL   Reviewed Aug. 8, 2023       1
297      Jay         Torrance, CA  Reviewed July 29, 2023       1
298  Theresa          Chester, PA  Reviewed July 27, 2023       1
299    Jenna       Waterville, OH  Reviewed July 21, 2023       1

                                                Review
295  i have spent hours on the phone with no resolu...
296  reading the previous complaints i can only con...
297  while it may be difficult to not shop at walma...
298  i updated my phone number and i was trying to ...
299  i bought a 75 inch tv online using walmart rig...
```

ii. Implement text cleaning, tokenization, and normalization techniques to prepare the data for analysis.

Text cleaning, tokenization, and normalization

Notes:

The most effective Natural Language Processing methods for Text Data Processing.

1. Tokenization
2. Stemming and Lemmatization
3. Stop Words Removal
4. TF-IDF
5. Keyword Extraction
6. Word Embedding
7. Sentiment Analysis
8. Topic Modeling
9. Text Summarization
10. Named Entity Recognition Sentences

Applying Tokenization: generating tokens or small chunks of words or sentences.

```
In [33]: from nltk.corpus import stopwords
         from nltk.tokenize import sent_tokenize

         example_sent = str(df['Review'])

         stop_words = set(stopwords.words('english'))

         sent_tokens = sent_tokenize(example_sent)

         filtered_paras = [w for w in sent_tokens if not w.lower() in stop_words]

         filtered_paras = []

         for w in word_tokens:
             if w not in stop_words:
                 filtered_paras.append(w)

         print(sent_tokens)
         print("")
         print(filtered_paras)
```

```
['0      customer service bad bought breville barista p...\n1      attempted review walmartcom indicates opted pu...\n2
walmart plus member years played left shock er...\n3      refused living room set sent week ahead schedu...\n4      beware
practices discriminatorypreferential or...\n                                    ...                               \n295    spent hours p
hone resolution sunday im paying ...\n296    reading previous complaints concur walmart mem...\n297    difficult shop walm
art obviously competitive p...\n298    updated phone number trying log account update...\n299    bought 75 inch tv online
walmart right buying ...\nName: Review, Length: 300, dtype: object']

['0', 'customer', 'service', 'bad', 'bought', 'br', '...', '1', 'attempted', 'put', 'review', 'walmart', '...', '2', 'walm
art', 'plus', 'member', 'years', '...', '3', 'refused', 'living', 'room', 'set', 'sent', 'one', '...', '4', 'beware', 'pra
ctices', 'discriminatorypreferential', '...', '...', '295', 'spent', 'hours', 'phone', 'resolu', '...', '296', 'reading',
'previous', 'complaints', 'con', '...', '297', 'may', 'difficult', 'shop', 'walma', '...', '298', 'updated', 'phone', 'num
ber', 'trying', '...', '299', 'bought', '75', 'inch', 'tv', 'online', 'using', 'walmart', 'rig', '...', 'Name', ':', 'Revi
ew', ',', 'Length', ':', '300', ',', 'dtype', ':', 'object']
```

Pre-processing is the process of transforming data so that a computer can comprehend it. The removal of unnecessary data is a common pre-processing method. Stop words are words that are considered unimportant or data in natural language processing.

Converting words into word_tokens and finding out stop words.

In [18]:

```python
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

example_sent = str(df['Review'])

stop_words = set(stopwords.words('english'))

word_tokens = word_tokenize(example_sent)
# converts the words in word_tokens to lower case and then checks whether
#they are present in stop_words or not
filtered_sentence = [w for w in word_tokens if not w.lower() in stop_words]
#with no lower case conversion
filtered_sentence = []

for w in word_tokens:
    if w not in stop_words:
        filtered_sentence.append(w)

print(word_tokens)
print("")
print(filtered_sentence)
```

```
['0', 'the', 'customer', 'service', 'is', 'very', 'bad', 'i', 'bought', 'a', 'br', '...', '1', 'i', 'have', 'attempted', '
to', 'put', 'this', 'review', 'on', 'walmart', '...', '2', 'i', 'have', 'been', 'a', 'walmart', 'plus', 'member', 'for', '
years', 'no', '...', '3', 'i', 'refused', 'a', 'living', 'room', 'set', 'that', 'they', 'sent', 'one', '...', '4', 'beware
', 'practices', 'discriminatorypreferential', 'or', '...', '...', '295', 'i', 'have', 'spent', 'hours', 'on', 'the', 'phon
e', 'with', 'no', 'resolu', '...', '296', 'reading', 'the', 'previous', 'complaints', 'i', 'can', 'only', 'con', '...', '2
97', 'while', 'it', 'may', 'be', 'difficult', 'to', 'not', 'shop', 'at', 'walma', '...', '298', 'i', 'updated', 'my', 'pho
ne', 'number', 'and', 'i', 'was', 'trying', 'to', '...', '299', 'i', 'bought', 'a', '75', 'inch', 'tv', 'online', 'using',
'walmart', 'rig', '...', 'Name', ':', 'Review', ',', 'Length', ':', '300', ',', 'dtype', ':', 'object']

['0', 'customer', 'service', 'bad', 'bought', 'br', '...', '1', 'attempted', 'put', 'review', 'walmart', '...', '2', 'walm
art', 'plus', 'member', 'years', '...', '3', 'refused', 'living', 'room', 'set', 'sent', 'one', '...', '4', 'beware', 'pra
ctices', 'discriminatorypreferential', '...', '...', '295', 'spent', 'hours', 'phone', 'resolu', '...', '296', 'reading',
'previous', 'complaints', 'con', '...', '297', 'may', 'difficult', 'shop', 'walma', '...', '298', 'updated', 'phone', 'num
ber', 'trying', '...', '299', 'bought', '75', 'inch', 'tv', 'online', 'using', 'walmart', 'rig', '...', 'Name', ':', 'Revi
ew', ',', 'Length', ':', '300', ',', 'dtype', ':', 'object']
```

Stop Words: A stop word is a term that search engines are designed to ignore while indexing entries for searching and retrieving them as a result of a search query. Examples of such words include "the," "a," "an," and "in." These terms shouldn't be using up important processing time or taking up space in our database.

Words are filtered out in many NLP and information retrieval applications, which can lower the dimensionality of the data and improve the effectiveness and efficiency of the algorithms. For instance, eliminating stopwords from a document can assist a text classification algorithm in concentrating on the most significant and pertinent words and designating the content appropriately for a label or category.

Popular English stopwords, such as:

- articles (a, an, the)
- conjunctions (and, but, or)

- prepositions (in, on, at)
- pronouns (he, she, it, they)
- auxiliary verbs (is, are, was, were)

It is crucial to do this first cleaning before turning text input into a bag of words for NLP modeling. Stopwords can, however, occasionally have an impact on the context or meaning of the text as well as the effectiveness of natural language processing and information retrieval algorithms.

Libraries for the removal of stop words:

1. NLTK stop words
2. SpaCy stop words
3. Gensim stop words

1. NLTK.

Finding the NLTK library's list of stop words in the English language.

```
In [27]: import nltk
         from nltk.corpus import stopwords

         nltk.download('stopwords')
         print(stopwords.words('english'))
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours
', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its',
'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'the
se', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did
', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with
', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'd
own', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'wh
y', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own
', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd
', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", '
hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'ne
edn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn',
"wouldn't"]
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\ravi\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
In [44]: nltk.download('stopwords')
         stopw_nltk = stopwords.words('english')
         print(len(stopw_nltk))

179
```

There are 179 English language stop words under the NLTK library.

Using NLTK library's stop words removing technique on the "Review" data column.

```
In [30]: import nltk
         nltk.download('stopwords')
         from nltk.corpus import stopwords

         stop_words = stopwords.words('english')
         df['Review'] = df['Review'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop_words)]))

[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\ravi\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!

In [31]: print (df['Review'])

0      customer service bad bought breville barista p...
1      attempted put review walmartcom indicates opte...
2      walmart plus member years well played really l...
3      refused living room set sent one week ahead sc...
4      beware practices discriminatorypreferential or...
                             ...
295    spent hours phone resolution since sunday im d...
296    reading previous complaints concur walmart mem...
297    may difficult shop walmart due obviously compe...
298    updated phone number trying log account update...
299    bought 75 inch tv online using walmart right b...
Name: Review, Length: 300, dtype: object
```

2. **spaCy Library: using spaCy NLP library and removing unnecessary words or data from the "Review" column.**

Finding out the no of stop words this library has.

```
In [45]: import spacy

         en = spacy.load('en_core_web_lg')
         stopw_spacy = en.Defaults.stop_words

         print(stopw_spacy)
```

```
{'then', "'d", 'me', 'six', 'last', 'forty', 'through', 'less', 'five', 'do', 'seemed', 'back', 'per', 'after', 'became',
'never', 'serious', 'using', 'hereby', 'such', 'an', 'had', 'one', 'nor', 'toward', 'with', "'s", 'someone', 'to', 'next',
'amongst', 'perhaps', 'was', 'namely', 'all', "'m", 'still', 'eleven', 'are', 'whether', 'becomes', 'out', 'before', 'alwa
ys', 'might', 'therefore', 'whereby', 'please', 'you', 'anyhow', 'nothing', 'although', 'also', 'show', 'thus', 'two', 'to
wards', 'hereupon', 'wherever', 'them', 'has', 'it', 'into', 'none', 'hers', 'nevertheless', 'those', 'regarding', 'anywhe
re', 'seems', 'too', 'eight', 'whither', 'its', 'very', 'name', 'four', 'until', 'rather', 'will', 'any', 'thereby', 'twel
ve', 'unless', 'beside', 'part', 'bottom', 'something', 'afterwards', 'become', 'over', 'if', 'ourselves', 'and', 'latter
', 'along', 'formerly', 'quite', 'ten', 'among', 'against', 'together', 'many', 'somehow', 'everywhere', 'about', 'three',
'whose', 'same', '’m', "'ve", 'nobody', 'other', 'on', 'even', 'take', 'own', 'every', 'these', 'seeming', 'as', 'give', '
top', 'whereupon', 'could', 'others', 'side', 'whom', 'neither', 'up', 'upon', 'around', 'well', 'alone', 'is', 'whole', '
anyway', 'by', 'itself', 'moreover', 'thence', 'myself', 'been', 'whereas', 'since', 'either', 'ever', 'they', '‘ll', 'eac
h', 'beforehand', 'say', '’s', "'ll", 'how', 'whenever', 'now', 'whatever', 'where', 'though', 'ours', 'hundred', 'themsel
ves', 'yours', 'elsewhere', 'be', '’re', 'move', 'twenty', '‘ve', 'more', 'have', 'down', 'just', 'besides', 'several', 't
his', 'being', 'much', 'whoever', 'full', 'fifty', 'or', 'we', 'who', 'front', 'the', 'not', 'while', 'us', 'i', 'meanwhil
e', 'can', 'yet', 'hence', '‘re', 'herself', '’ve', 'mine', 'because', 'latterly', 'noone', 'am', 'few', 'keep', 'did', 'a
gain', 'n’t', 'your', 'yourself', 'when', 'thereafter', 'ca', 'amount', 'within', "'re", 'indeed', 'used', 'first', 'nowhe
re', 'beyond', 'therein', 'third', 'various', 'what', 'would', 'anyone', 'cannot', '‘s', 'only', 'but', 'made', 'under', '
doing', 'his', 'why', 'in', 'him', 'almost', 'nine', 'sometime', 'that', 'often', 'yourselves', 'between', 'there', 'where
in', 'via', 'done', 'our', 'mostly', 'at', 'he', 'else', 'fifteen', 'her', 'here', 'throughout', 'whence', 'further', 'eno
ugh', 'a', 'their', 'should', 'sometimes', 'otherwise', 'above', 'due', 'hereafter', 'already', 'onto', '‘d', 'must', 'beh
ind', 'thru', 'most', 'during', 'were', 'thereupon', 'least', 'once', 'make', 'put', 'call', 'no', 'really', 'she', 'somew
here', 'n‘t', 'both', 'from', '’ll', 'whereafter', "n't", 'does', 'herein', 'go', 'anything', 'some', 'which', 'sixty', 'f
or', 're', 'another', '’d', 'former', 'except', 'so', 'may', 'without', 'himself', 'everything', 'below', 'seem', 'than',
'‘m', 'across', 'becoming', 'empty', 'my', 'however', 'get', 'of', 'everyone', 'off', 'see'}
```

```
In [46]: print(len(stopw_spacy))
```

```
326
```

It is evident that the spaCy library has a greater number of stop words than NLTK, which implies that with the spaCy library, input data can be cleaned more thoroughly than with NLTK.

Application of spaCy library to clean "Review" column's data.

```
In [49]: df['Review'] = df['Review'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stopw_spacy)]))
```

```
In [50]: print (df['Review'])
```

```
0      customer service bad bought breville barista p...
1      attempted review walmartcom indicates opted pu...
2      walmart plus member years played left shock er...
3      refused living room set sent week ahead schedu...
4      beware practices discriminatorypreferential or...
                             ...
295    spent hours phone resolution sunday im paying ...
296    reading previous complaints concur walmart mem...
297    difficult shop walmart obviously competitive p...
298    updated phone number trying log account update...
299    bought 75 inch tv online walmart right buying ...
Name: Review, Length: 300, dtype: object
```

3. **Gensim Library:**

Unlike most other machine learning software packages, Gensim is built to handle massive text collections with incremental online algorithms and data streaming.

Gensim has a similar stop word count as spaCy.

```
In [28]: import gensim
         from gensim.parsing.preprocessing import remove_stopwords, STOPWORDS
         print(STOPWORDS)
```

```
frozenset({'regarding', 'very', 'over', 'also', 'her', 'much', 'well', 'your', 'first', 'latterly', 'several', 'less', 'el
se', 'again', 'itself', 'hence', 'mine', 'con', 'someone', 'except', 'already', 'meanwhile', 'keep', 'fifty', 'nine', 'kg
', 'find', 'my', 'us', 'why', 'onto', 'give', 'forty', 'various', 'whatever', 'whom', 'herein', 'here', 'nobody', 'might',
'former', 'every', 'perhaps', 'from', 'whether', 'i', 'yet', 'had', 'against', 'below', 'whole', 'un', 'found', 'between',
'interest', 'a', 'five', 'describe', 'part', 'computer', 'see', 'would', 'without', 'only', 'doing', 'wherever', 'neither
', 'along', 'then', 'should', 'hasnt', 'although', 'done', 'yourself', 'anyway', 'bottom', 'thereby', 'something', 'ie', '
into', 'next', 'what', 'please', 'together', 'via', 'thus', 'could', 'somewhere', 'any', 'around', 'km', 'one', 'may', 'af
terwards', 'ever', 'herself', 'take', 'because', 'as', 'always', 'become', 'name', 'alone', 'ltd', 'there', 'if', 'not', '
empty', 'mill', 'them', 'many', 'we', 'now', 'ours', 'nevertheless', 'really', 'during', 'most', 'detail', 'co', 'whereas
', 'across', 'does', 'call', 'have', 'thick', 'towards', 'among', 'front', 'whenever', 'get', 'through', 'go', 'was', 'unt
il', 'everywhere', 'eg', 'became', 'either', 'down', 'others', 'is', 'own', 'beyond', 'nor', 'whence', 'don', 'than', 'som
ehow', 'themselves', 'will', 'serious', 'never', 'such', 'though', 'has', 'full', 'say', 'you', 'seemed', 'moreover', 'six
ty', 'whoever', 'out', 'quite', 'after', 'by', 'the', 're', 'whither', 'anyhow', 'per', 'both', 'nowhere', 'about', 'noone
', 'etc', 'upon', 'being', 'throughout', 'to', 'who', 'these', 'thereupon', 'too', 'using', 'whereby', 'side', 'becomes',
'elsewhere', 'top', 'once', 'its', 'on', 'so', 'their', 'behind', 'since', 'bill', 'am', 'another', 'just', 'otherwise', '
other', 'doesn', 'put', 'didn', 'no', 'me', 'yours', 'and', 'beside', 'even', 'been', 'whereupon', 'however', 'almost', 'm
ove', 'more', 'an', 'were', 'back', 'rather', 'himself', 'above', 'make', 'whose', 'enough', 'his', 'anyone', 'each', 'how
', 'all', 'ourselves', 'thin', 'third', 'eleven', 'he', 'namely', 'before', 'that', 'couldnt', 'under', 'hereby', 'amount
', 'seems', 'four', 'hereupon', 'still', 'must', 'hereafter', 'last', 'beforehand', 'off', 'when', 'besides', 'anything',
'or', 'everyone', 'fifteen', 'our', 'used', 'which', 'anywhere', 'six', 'thereafter', 'twenty', 'while', 'two', 'but', 'si
ncere', 'same', 'seem', 'sometimes', 'of', 'latter', 'hundred', 'therein', 'often', 'system', 'hers', 'in', 'indeed', 'unl
ess', 'seeming', 'mostly', 'thence', 'three', 'eight', 'this', 'therefore', 'fill', 'cannot', 'everything', 'at', 'inc', '
can', 'show', 'due', 'for', 'myself', 'ten', 'few', 'him', 'are', 'twelve', 'those', 'be', 'further', 'they', 'none', 'bec
oming', 'formerly', 'some', 'it', 'yourselves', 'sometime', 'de', 'whereafter', 'did', 'amoungst', 'made', 'nothing', 'cry
', 'toward', 'wherein', 'she', 'where', 'least', 'fire', 'up', 'do', 'within', 'with', 'amongst', 'cant', 'thru'})
```

```
In [29]: print(len(STOPWORDS))
```

```
337
```

Application of Gensim library to remove stop words from the “Review” column’s data.

```
In [32]: new_review = remove_stopwords(str(df['Review']))
         print(new_review)
```

```
0 customer service bad bought breville barista p... 1 attempted review walmartcom indicates opted pu... 2 walmart plus mem
ber years played left shock er... 3 refused living room set sent week ahead schedu... 4 beware practices discriminatorypre
ferential or... ... 295 spent hours phone resolution sunday im paying ... 296 reading previous complaints concur walmart m
em... 297 difficult shop walmart obviously competitive p... 298 updated phone number trying log account update... 299 boug
ht 75 inch tv online walmart right buying ... Name: Review, Length: 300, dtype: object
```

Stemming: Reducing a word to its base or root form is known as stemming.

Stemming algorithms: Porter stemmer and Snowball stemmer.

Porter stemmer: Method for taking English words' suffixes away.

Application to “Review” dataset:

```
In [80]: from nltk.stem import PorterStemmer

         word_Pstemmer = PorterStemmer()

         # Split the sentences to lists of words.
         df['Review'] = df['Review'].str.split()

         df['stemmed_Review'] = df['Review'].apply(lambda x: [word_Pstemmer.stem(y) for y in x]) # Stem every word.

         df
```

Out[80]:

	name	location	Date	Rating	Review	stemmed_Review
0	A.	Ankeny, IA	Reviewed Sept. 13, 2023	1	[customer, service, bad, bought, breville, bar...	[custom, servic, bad, bought, brevil, barista,...
1	DONNA	Phoenix, AZ	Reviewed Sept. 2, 2023	1	[attempted, review, walmartcom, indicates, opt...	[attempt, review, walmartcom, indic, opt, purc...
2	Jill	Baton Rouge, LA	Reviewed Aug. 28, 2023	1	[walmart, plus, member, years, played, left, s...	[walmart, plu, member, year, play, left, shock...
3	Sukanya	Maumee, OH	Reviewed Aug. 25, 2023	1	[refused, living, room, set, sent, week, ahead...	[refus, live, room, set, sent, week, ahead, sc...
4	Tiffany	Laurinburg, NC	Reviewed Aug. 18, 2023	1	[beware, practices, discriminatorypreferential...	[bewar, practic, discriminatorypreferenti, ord...
...	...	...	...	...	...	...
295	Melissa	Texas City, TX	Reviewed Aug. 15, 2023	1	[spent, hours, phone, resolution, sunday, im, ...	[spent, hour, phone, resolut, sunday, im, pay,...
296	David	West Palm Beach, FL	Reviewed Aug. 8, 2023	1	[reading, previous, complaints, concur, walmar...	[read, previou, complaint, concur, walmart, me...
297	Jay	Torrance, CA	Reviewed July 29, 2023	1	[difficult, shop, walmart, obviously, competit...	[difficult, shop, walmart, obvious, competit, ...
298	Theresa	Chester, PA	Reviewed July 27, 2023	1	[updated, phone, number, trying, log, account,...	[updat, phone, number, tri, log, account, upda...
299	Jenna	Waterville, OH	Reviewed July 21, 2023	1	[bought, 75, inch, tv, online, walmart, right,...	[bought, 75, inch, tv, onlin, walmart, right, ...

300 rows × 6 columns

Snowball stemmer: Reducing a word to its basic word or stem such that words of the same kind fall under a common stem is known as snowball stemming.

Unit Test:

```
In [13]: from nltk.stem import SnowballStemmer
         snowball = SnowballStemmer(language='english')
         words = ['generous','generate','generously','generation','authentication','stimulation','precisely','validate',
                  'validation']
         for word in words:
             print(word,"--->",snowball.stem(word))

generous ---> generous
generate ---> generat
generously ---> generous
generation ---> generat
authentication ---> authent
stimulation ---> stimul
precisely ---> precis
validate ---> valid
validation ---> valid
```

Find out the languages that Snowball Stemming supports..

```
In [28]: from nltk.stem.snowball import SnowballStemmer

         print(" ".join(SnowballStemmer.languages))
```

```
arabic danish dutch english finnish french german hungarian italian norwegian porter portuguese romanian russian spanish s
wedish
```

Choosing whether to not to stem stopwords when creating a new instance of a subclass appropriate to a language.

```
In [29]: sb_stemmer = SnowballStemmer("english")
```

```
In [30]: sb_stemmer_1 = SnowballStemmer("english", ignore_stopwords=True)
```

```
In [32]: print(sb_stemmer.stem("having"))
         print(sb_stemmer_1.stem("having"))
```

```
have
having
```

Application:

Converting the “Review” data list into a string.

```
In [35]: all_feedbacks_together = ' '.join(df['Review'])
         print(all_feedbacks_together)
```

```
gas vehicle not only did it come up that they attempted to deliver it but they dont even reschedule or even try to accommo
date i wasted hours trying to resolve this reading the previous complaints i can only concur with walmart membership servi
ce in general it’s obvious that customer service representatives are located overseas and speak broken english for starter
s why they are trained for niceties in the hopes to quell the poor services it serves no purpose if the simple question wh
y my promised delivery twice in one week did not appear insult to injury i contacted the local walmart where the delivery
was coming from 12 miles away the operator told me there was no store manager that i can speak to and referred to the onli
ne customer service representative  while it may be difficult to not shop at walmart due to their obviously very competiti
ve prices versus their competitors the reason they are able to do this is because they cut corners everywhere else whether
it’s the terrible conditions for their employees or the overseas labor they use as their supposed customer service departm
ent that don’t speak english and respond only with automated scripted answers or the lack of any type of sales support at
all once the initial purchase is completed until we as consumers stop giving them our billions of dollars each and every y
ear you can expect to continue to receive the same 1 star treatment as evidenced by the thousands of reviews posted here a
nd elsewhere i updated my phone number and i was trying to log into my account to update the information on my walmart acc
ount it showed that a otp was sent to my old number which i dont have any longer i called support and spoke with 2 reps an
d a supervisor they told me that since i do not have access to the old number i would need to create a new account the cat
ch is that i cannot use my email address i would have to create another account with a different email address when i told
them i dont have another email address they said there is nothing that they could do basically they are saying they no lon
ger want my business also since my credit card information is on my account that i cannot access if my account gets hacked
walmart will be responsible for the loss i will never shop walmart again i bought a 75 inch tv online using walmart right
```

Dividing the sentences into a word list.

```
In [39]: # Split the sentences to lists of words.
         all_feedbacks_together_to_words = all_feedbacks_together.split()
         print(all_feedbacks_together_to_words)
```

```
['the', 'customer', 'service', 'is', 'very', 'bad', 'i', 'bought', 'a', 'breville', 'barista', 'pro', 'espresso', 'machine
', 'for', '468', 'which', 'is', 'too', 'cheap', 'because', 'the', 'machines', 'price', 'is', '850', 'the', 'thirdparty', '
seller', 'was', 'a', 'scammer', 'he', 'puts', 'low', 'prices', 'to', 'attract', 'buyers', 'he', 'sent', 'me', 'a', 'tracki
ng', 'number', 'to', 'the', 'same', 'city', 'as', 'my', 'address', 'but', 'to', 'a', 'different', 'address', 'and', 'sent
', 'a', 'small', 'box', 'that', 'fit', 'a', 'mailbox', 'the', 'machine', 'is', 'about', '30', 'pounds', 'i', 'called', 'th
e', 'usps', 'and', 'they', 'told', 'me', 'that', 'the', 'tracking', 'number', 'was', 'not', 'to', 'my', 'address', 'so', '
walmart', 'charged', 'me', '468', 'and', 'later', 'they', 'removed', 'that', 'seller', 'from', 'their', 'site', 'due', 'to
', 'quality', 'issues', 'that', 'means', 'he', 'is', 'a', 'scammer', 'i', 'have', 'attempted', 'to', 'put', 'this', 'revie
w', 'on', 'walmartcom', 'but', 'it', 'indicates', 'i', 'have', 'been', 'opted', 'out', 'i', 'purchased', 'binax', 'covid',
'tests', 'online', 'to', 'pick', 'up', 'at', 'store', 'good', 'product', 'product', 'is', 'five', 'stars', 'online', 'shop
ping', 'experience', 'is', 'one', 'star', 'online', 'order', 'sunday', '82723', 'with', 'store', 'pickup', 'on', 'monday',
'82823', 'around', '2pm', 'i', 'am', 'reviewing', 'to', 'caution', 'other', 'online', 'shoppers', 'to', 'always', 'scroll
', 'down', 'before', 'adding', 'any', 'item', 'to', 'their', 'walmart', 'cart', 'because', 'the', 'notice', 'if', 'somethi
ng', 'is', 'not', 'returnable', 'isnt', 'visible', 'unless', 'you', 'do', 'i', 'picked', 'up', 'my', 'order', 'on', 'the',
'way', 'out', 'of', 'town', 'and', 'only', 'later', 'saw', 'that', 'the', 'expiration', 'dates', 'were', 'all', 'close', '
enough', 'that', 'i', 'never', 'would', 'have', 'purchased', 'the', 'same', 'amount', 'of', 'tests', 'had', 'i', 'been', '
shopping', 'instore', 'so', 'i', 'clicked', 'start', 'a', 'return', 'on', 'my', 'receipt', 'and', 'only', 'then', 'found',
'out', 'i', 'had', 'purchased', 'something', 'nonrefundable', 'i', 'have', 'been', 'a', 'walmart', 'plus', 'member', 'for
', 'years', 'now', 'and', 'well', 'how', 'this', 'all', 'played', 'out', 'has', 'really', 'left', 'me', 'in', 'shock', 'du
```

Applying SnowBall stemmer.

```
In [48]: new_list = []

         for word in all_feedbacks_together_to_words:
             new_list.append(word)

             result = word

             print(result)

         print(sb_stemmer.stem(new_list))
```

```
the
customer
service
is
very
bad
i
bought
a
breville
barista
pro
espresso
machine
for
468
which
is
too
```

Below the “All_feedbacks_together_to_words_2” is the “Review” column’s data

```
In [49]: all_feedbacks_together_to_words_2 = all_feedbacks_together.split()
         print(all_feedbacks_together_to_words_2)
```

```
['the', 'customer', 'service', 'is', 'very', 'bad', 'i', 'bought', 'a', 'breville', 'barista', 'pro', 'espresso', 'machine
', 'for', '468', 'which', 'is', 'too', 'cheap', 'because', 'the', 'machines', 'price', 'is', '850', 'the', 'thirdparty', '
seller', 'was', 'a', 'scammer', 'he', 'puts', 'low', 'prices', 'to', 'attract', 'buyers', 'he', 'sent', 'me', 'a', 'tracki
ng', 'number', 'to', 'the', 'same', 'city', 'as', 'my', 'address', 'but', 'to', 'a', 'different', 'address', 'and', 'sent
', 'a', 'small', 'box', 'that', 'fit', 'a', 'mailbox', 'the', 'machine', 'is', 'about', '30', 'pounds', 'i', 'called', 'th
e', 'usps', 'and', 'they', 'told', 'me', 'that', 'the', 'tracking', 'number', 'was', 'not', 'to', 'my', 'address', 'so', '
walmart', 'charged', 'me', '468', 'and', 'later', 'they', 'removed', 'that', 'seller', 'from', 'their', 'site', 'due', 'to
', 'quality', 'issues', 'that', 'means', 'he', 'is', 'a', 'scammer', 'i', 'have', 'attempted', 'to', 'put', 'this', 'revie
w', 'on', 'walmartcom', 'but', 'it', 'indicates', 'i', 'have', 'been', 'opted', 'out', 'i', 'purchased', 'binax', 'covid',
'tests', 'online', 'to', 'pick', 'up', 'at', 'store', 'good', 'product', 'product', 'is', 'five', 'stars', 'online', 'shop
ping', 'experience', 'is', 'one', 'star', 'online', 'order', 'sunday', '82723', 'with', 'store', 'pickup', 'on', 'monday',
'82823', 'around', '2pm', 'i', 'am', 'reviewing', 'to', 'caution', 'other', 'online', 'shoppers', 'to', 'always', 'scroll
', 'down', 'before', 'adding', 'any', 'item', 'to', 'their', 'walmart', 'cart', 'because', 'the', 'notice', 'if', 'somethi
ng', 'is', 'not', 'returnable', 'isnt', 'visible', 'unless', 'you', 'do', 'i', 'picked', 'up', 'my', 'order', 'on', 'the',
'way', 'out', 'of', 'town', 'and', 'only', 'later', 'saw', 'that', 'the', 'expiration', 'dates', 'were', 'all', 'close', '
enough', 'that', 'i', 'never', 'would', 'have', 'purchased', 'the', 'same', 'amount', 'of', 'tests', 'had', 'i', 'been', '
shopping', 'instore', 'so', 'i', 'clicked', 'start', 'a', 'return', 'on', 'my', 'receipt', 'and', 'only', 'then', 'found',
'out', 'i', 'had', 'purchased', 'something', 'nonrefundable', 'i', 'have', 'been', 'a', 'walmart', 'plus', 'member', 'for
', 'years', 'now', 'and', 'well', 'how', 'this', 'all', 'played', 'out', 'has', 'really', 'left', 'me', 'in', 'shock', 'du
```

Regex stemmer: By using regular expressions, morphological affixes are identified.

Unit Test 1:

```
In [11]: from nltk.stem import RegexpStemmer

         re_stemmer = RegexpStemmer('ing')

         words = ['meeting','meets','catching','pushes','pushing','hiding']
         for word in words:
             print(word,"--->",re_stemmer.stem(word))
```

```
meeting ---> meet
meets ---> meets
catching ---> catch
pushes ---> pushes
pushing ---> push
hiding ---> hid
```

Unit Test 2:

```
In [19]: re_stemmer = RegexpStemmer('ing$|s$|e$|able$', min=4)
         words = ['mass','was','bee','computer','advisable','eating','shouting','streaming','notifies']
         for word in words:
             print(word,"--->",re_stemmer.stem(word))
```

```
mass ---> mas
was ---> was
bee ---> bee
computer ---> computer
advisable ---> advis
eating ---> eat
shouting ---> shout
streaming ---> stream
notifies ---> notifie
```

Application to the "Review" column's data.

```
In [51]: from nltk.stem import RegexpStemmer

         re_stemmer = RegexpStemmer('ing')


         new_list2 = []

         for word in all_feedbacks_together_to_words_2:
             new_list.append(word)

             result = word

             print(result)

         print(re_stemmer.stem(new_list))
```

```
the
customer
service
is
very
bad
i
bought
a
breville
barista
pro
espresso
machine
for
468
which
is
too
```

Lancaster stemmer: Although Lancaster Stemmer is simple to use, it often produces results with excessive stemming.

Unit Test: stemming the list of words.

```
In [9]: from nltk.stem import LancasterStemmer
        l_stammer = LancasterStemmer()

        words = ['meeting','meets','met','pushes','pushing']
        for word in words:
            print(word,"--->",l_stammer.stem(word))

meeting ---> meet
meets ---> meet
met ---> met
pushes ---> push
pushing ---> push
```

Applying with the "Review" dataset:

```
In [6]: from nltk.stem import LancasterStemmer
        l_stammer = LancasterStemmer()

        new_list = []

        for word in all_feedbacks_together_to_words:
            new_list.append(word)

            result = word

            print(result)

        print(l_stammer.stem(new_list))
```

```
the
customer
service
is
very
bad
i
bought
a
breville
barista
pro
espresso
machine
for
468
which
is
too
```

Comparing Porter vs Snowball vs Lancaster vs Regex Stemming in NLTK.

Unit Test:

```
In [32]: from nltk.stem import PorterStemmer, SnowballStemmer, LancasterStemmer, RegexpStemmer

         porter_stemmer = PorterStemmer()

         lancaster_stemmer = LancasterStemmer()

         snowball_stemmer = SnowballStemmer(language='english')

         regexp_stemmer = RegexpStemmer('ing$|s$|able$|b$|ship$', min=6)

         word_list = ["horrible", "friendship", "superb", "disgusting"]

         print("{0:20}{1:20}{2:20}{3:30}{4:40}".format("Word","Porter Stemmer","Snowball Stemmer","Lancaster Stemmer",
                                                        'Regexp Stemmer'))

         for word in word_list:
             print("{0:20}{1:20}{2:20}{3:30}{4:40}".format(word,porter_stemmer.stem(word),
                                                            snowball_stemmer.stem(word),
                                                            lancaster_stemmer.stem(word),
                                                            regexp_stemmer.stem(word)))
```

```
Word                Porter Stemmer      Snowball Stemmer    Lancaster Stemmer             Regexp Stemmer
horrible            horribl             horribl             horr                          horrible
friendship          friendship          friendship          friend                        friend
superb              superb              superb              superb                        super
disgusting          disgust             disgust             disgust                       disgust
```

Lemmatization: This is an option to stemming. This method looks at the word's meaning, whereas the stemming technique only looks at the word's form. Users can use the NLTK, spaCy, and Gensim packages for lemmatization.

1. NLTK Lemmatizer: applying WordNetLemmatizer

Unit Test:

```
In [10]: import nltk
         from nltk.stem import WordNetLemmatizer

         # Define a text string
         sample_text = "words books eating"

         # Tokenizing / individual words
         tokens = nltk.word_tokenize(sample_text)

         # WordNetLemmatizer object / instance
         WN_lemmatizer = WordNetLemmatizer()

         # Lemmatizing
         for token in tokens:
           la = WN_lemmatizer.lemmatize(token)
           print(token, "-->", la)

         words --> word
         books --> book
         eating --> eating
```

Application: when applying NLTK Lemmatizer with the "Review" dataset,

```
In [7]: all_feedbacks_together = ' '.join(df['Review'])
        print(all_feedbacks_together)

        # Split the sentences to lists of words.
        all_feedbacks_together_to_words = all_feedbacks_together.split()
        print(all_feedbacks_together_to_words)

        the customer service is very bad i bought a breville barista pro espresso machine for 468 which is too cheap because the m
        achines price is 850 the thirdparty seller was a scammer he puts low prices to attract buyers he sent me a tracking number
        to the same city as my address but to a different address and sent a small box that fit a mailbox the machine is about 30
        pounds i called the usps and they told me that the tracking number was not to my address so walmart charged me 468 and lat
        er they removed that seller from their site due to quality issues that means he is a scammer  i have attempted to put this
        review on walmartcom but it indicates i have been opted out i purchased binax covid tests online to pick up at store good
        product product is five stars online shopping experience is one star online order sunday 82723 with store pickup on monday
        82823 around 2pm i am reviewing to caution other online shoppers to always scroll down before adding any item to their wal
        mart cart because the notice if something is not returnable isnt visible unless you do i picked up my order on the way out
        of town and only later saw that the expiration dates were all close enough that i never would have purchased the same amou
        nt of tests had i been shopping instore so i clicked start a return on my receipt and only then found out i had purchased
        something nonrefundable i have been a walmart plus member for years now and well how this all played out has really left m
        e in shock due to their errors and failure to fulfill my orders they have closed my acct due to me refusing to pay for ite
        ms i didnt receive before i moved to my new place i rarely had any issue with the walmart delivery but once i moved it see
        med like every order there was more and more items missing  i refused a living room set that they sent one week ahead of s
        chedule i refused it while there was still one week left to the delivery date they brought it one week ahead of schedule a
        nd then charged me a 155 shipping return and restocking fee saying it was the buyers fault it took walmart 3 weeks and num
        erous phone calls to finally give me a refund of 665 out of my 1038 return their customer service said i need to call thei
        r angi living room assembly service separately to get my assembly refund of 194 this was inspite of the fact that angie se
```

In [13]:
```
all_feedbacks_together = ' '.join(df['Review'])
tokens_2 = nltk.word_tokenize(all_feedbacks_together)

WN_lemmatizer_2 = WordNetLemmatizer()

for token in tokens_2:
  la_2 = WN_lemmatizer_2.lemmatize(token)
  print(token, "-->", la_2)
```

```
pro --> pro
espresso --> espresso
machine --> machine
for --> for
468 --> 468
which --> which
is --> is
too --> too
cheap --> cheap
because --> because
the --> the
machines --> machine
price --> price
is --> is
850 --> 850
the --> the
thirdparty --> thirdparty
seller --> seller
was --> wa
a --> a
```

2. SpaCy Lemmatizer

Unit Test:

In [26]:
```
import spacy

# Load the English language model in spaCy
nlp_spacy = spacy.load('en_core_web_lg')

# Define a text string
text_sample = "This is a sample text and this contains some words"

# Create a Doc object
doc = nlp_spacy(text_sample)

# Lemmatize each token and print the result
for token in doc:
  le = token.lemma_
  print(token.text, "-->", le)
```

```
This --> this
is --> be
a --> a
sample --> sample
text --> text
and --> and
this --> this
contains --> contain
some --> some
words --> word
```

Application:

```
In [27]: nlp_spacy = spacy.load('en_core_web_lg')

         text_2 = ' '.join(df['Review'])

         doc = nlp_spacy(text_2)

         for token in doc:
           le = token.lemma_
           print(token.text, "-->", le)
```

```
bad --> bad
i --> I
bought --> buy
a --> a
breville --> breville
barista --> barista
pro --> pro
espresso --> espresso
machine --> machine
for --> for
468 --> 468
which --> which
is --> be
too --> too
cheap --> cheap
because --> because
the --> the
machines --> machine
price --> price
is --> be
```

By default, this only takes into account nouns, verbs, adjectives, and adverbs (all other lemmas are eliminated).

Limitations in Lemmatization: its computational difficulty, the requirement for a large vocabulary, and the morphological analysis of the words.

Alternatives to lemmatization:

Stemming: taking a word's suffixes and prefixes away.

Synonym mapping: swapping a predetermined synonym or group of synonyms for each word.

Dimensionality reduction: reducing the number of dimensions in the text data by applying mathematical approaches like non-negative matrix factorization (NMF) and singular value decomposition (SVD).

Part of Speech Tagging (POS) with Stop words using NLTK in python

The function of the POS tagger is to provide sub-sentential units or tokens (words and symbols, such as punctuation) with linguistic (primarily grammatical) information.

list of the tags, and their meaning:

CC coordinating conjunction

CD cardinal digit

DT determiner

EX existential there (like: “there is” … think of it like “there exists”)

FW foreign word

IN preposition/subordinating conjunction

JJ adjective – ‘big’

JJR adjective, comparative – ‘bigger’

JJS adjective, superlative – ‘biggest’

LS list marker 1)

MD modal – could, will

NN noun, singular ‘- desk’

NNS noun plural – ‘desks’

NNP proper noun, singular – ‘Harrison’

NNPS proper noun, plural – ‘Americans’

PDT predeterminer – ‘all the kids’

POS possessive ending parent’s

PRP personal pronoun – I, he, she

PRP$ possessive pronoun – my, his, hers

RB adverb – very, silently,

RBR adverb, comparative – better

RBS adverb, superlative – best

RP particle – give up

TO – to go 'to' the store.

UH interjection – errrrrrrrm

VB verb, base form – take

VBD verb, past tense – took

VBG verb, gerund/present participle – taking

VBN verb, past participle – taken

VBP verb, sing. present, non-3d – take

VBZ verb, 3rd person sing. present – takes

WDT wh-determiner – which

WP wh-pronoun – who, what

WP$ possessive wh-pronoun, eg- whose

WRB wh-adverb, eg- where, when

Unit Test:

```
In [4]: import nltk
        from nltk.corpus import stopwords
        from nltk.tokenize import word_tokenize, sent_tokenize
        stop_words = set(stopwords.words('english'))

        feedback = "The customer service is very bad. I bought a Breville Barista Pro espresso machine for 468$ "
        "which is too cheap because the machine's price is 850$. The third-party seller was a scammer."


        tokenized = sent_tokenize(feedback)
        for i in tokenized:

            # Word tokenizers is used to find the words, punctuation
            wordsList = nltk.word_tokenize(i)

            # removing stop words
            wordsList = [w for w in wordsList if not w in stop_words]

            #  Using a Tagger. Which is part-of-speech tagger or POS-tagger
            tagged = nltk.pos_tag(wordsList)

            print(tagged)

        [('The', 'DT'), ('customer', 'NN'), ('service', 'NN'), ('bad', 'JJ'), ('.', '.')]
        [('I', 'PRP'), ('bought', 'VBD'), ('Breville', 'NNP'), ('Barista', 'NNP'), ('Pro', 'NNP'), ('espresso', 'FW'), ('machine',
        'NN'), ('468', 'CD'), ('$', '$')]
```

Application with the feedback data.

Getting feedback string.

```
In [11]: all_feedbacks_together_reviews = ' '.join(df['Review'])
         print(all_feedbacks_together_reviews)
```

```
the customer service is very bad. i bought a breville barista pro espresso machine for 468$ which is too cheap because the
machine's price is 850$. the third-party seller was a scammer. he puts low prices to attract buyers. he sent me a tracking
number to the same city as my address but to a different address and sent a small box that fit a mailbox. the machine is a
bout 30 pounds. i called the usps and they told me that the tracking number was not to my address. so walmart charged me 4
68$ and later they removed that seller from their site due to quality issues (that means he is a scammer).  i have attempt
ed to put this review on walmart.com but it indicates i have been 'opted out'. i purchased binax covid tests online to pic
k up at store. good product. product is five stars, online shopping experience is one star. online order sunday 8/27/23 wi
th store pickup on monday 8/28/23 around 2pm. i am reviewing to caution other online shoppers to always scroll down before
adding any item to their walmart cart because the notice if something is not returnable isn't visible unless you do. i pic
ked up my order on the way out of town, and only later saw that the expiration dates were all close enough that i never wo
uld have purchased the same amount of tests had i been shopping in-store. so i clicked 'start a return' on my receipt and
only then found out i had purchased something non-refundable. i have been a walmart plus member for years now and well how
this all played out has really left me in shock. due to their errors and failure to fulfill my orders they have closed my
acct due to me refusing to pay for items i didn't receive. before i moved to my new place i rarely had any issue with the
walmart delivery but once i moved it seemed like every order there was more and more items missing.  i refused a living ro
om set that they sent one week ahead of schedule. i refused it while there was still one week left to the delivery date. t
hey brought it one week ahead of schedule and then charged me a $155 'shipping return and restocking fee' saying it was th
e buyer's fault. it took walmart 3 weeks and numerous phone calls to finally give me a refund of $665 out of my $1038 retu
rn. their customer service said i need to call their angi living room assembly service separately to get my assembly refun
```

Applying POS.

```
In [12]:
tokenized = sent_tokenize(all_feedbacks_together_reviews)
for i in tokenized:

    # Word tokenizers is used to find the words, punctuation
    wordsList = nltk.word_tokenize(i)

    # removing stop words
    wordsList = [w for w in wordsList if not w in stop_words]

    #  Using a Tagger. Which is part-of-speech tagger or POS-tagger
    tagged = nltk.pos_tag(wordsList)

    print(tagged)
```

```
[('customer', 'NN'), ('service', 'NN'), ('bad', 'JJ'), ('.', '.')]
[('bought', 'VBN'), ('breville', 'NN'), ('barista', 'NN'), ('pro', 'JJ'), ('espresso', 'FW'), ('machine', 'NN'), ('468', '
CD'), ('$', '$'), ('cheap', 'JJ'), ('machine', 'NN'), ("'s", 'POS'), ('price', 'NN'), ('850', 'CD'), ('$', '$'), ('.',
'.')]
[('third-party', 'JJ'), ('seller', 'NN'), ('scammer', 'NN'), ('.', '.')]
[('puts', 'NNS'), ('low', 'JJ'), ('prices', 'NNS'), ('attract', 'JJ'), ('buyers', 'NNS'), ('.', '.')]
[('sent', 'NN'), ('tracking', 'VBG'), ('number', 'NN'), ('city', 'NN'), ('address', 'RB'), ('different', 'JJ'), ('address
', 'NN'), ('sent', 'VBD'), ('small', 'JJ'), ('box', 'NN'), ('fit', 'NN'), ('mailbox', 'NN'), ('.', '.')]
[('machine', 'NN'), ('30', 'CD'), ('pounds', 'NNS'), ('.', '.')]
[('called', 'VBN'), ('usps', 'JJ'), ('told', 'VBD'), ('tracking', 'VBG'), ('number', 'NN'), ('address', 'NN'), ('.', '.')]
[('walmart', 'NN'), ('charged', 'VBD'), ('468', 'CD'), ('$', '$'), ('later', 'RB'), ('removed', 'VBN'), ('seller', 'NN'),
('site', 'NN'), ('due', 'JJ'), ('quality', 'NN'), ('issues', 'NNS'), ('(', '('), ('means', 'NNS'), ('scammer', 'NN'),
(')', ')'), ('.', '.')]
[('attempted', 'VBN'), ('put', 'VBD'), ('review', 'NN'), ('walmart.com', 'NN'), ('indicates', 'VBZ'), ("'opted", 'VBN'),
("'", 'POS'), ('.', '.')]
[('purchased', 'VBN'), ('binax', 'NN'), ('covid', 'NN'), ('tests', 'NNS'), ('online', 'VBP'), ('pick', 'JJ'), ('store', 'N
N'), ('.', '.')]
[('good', 'JJ'), ('product', 'NN'), ('.', '.')]
[('product', 'NN'), ('five', 'CD'), ('stars', 'NNS'), (',', ','), ('online', 'JJ'), ('shopping', 'NN'), ('experience', 'NN
```

Named entity recognition (NER) / entity chunking or entity extraction: finding items in a text body that fall into predetermined categories.

NER categories: names of individuals, organizations, locations, expressions of times, quantities, medical codes, monetary values and percentages.

Unit Test 1: Extracting Named Entities

```
In [9]: NER = spacy.load("en_core_web_sm")

        review_text="I refused a living room set that they sent one week ahead of schedule. I refused it while there was "
        "still one week left to the delivery date. They brought it one week ahead of schedule and then charged "
        "me a $155 'shipping return and restocking fee' saying it was the "

        text1= NER(review_text)

        for word in text1.ents:
            print(word.text,word.label_)
```

```
one week DATE
one week DATE
one week DATE
155 MONEY
```

Finding the type of a Named Entity.

```
In [4]: spacy.explain("GPE")
Out[4]: 'Countries, cities, states'
```

Applying the NE extraction with the Feedback / Review dataset.

```
In [12]: reviews= NER(' '.join(df['Review']))

         for word in reviews.ents:
             print(word.text,word.label_)
```

```
one CARDINAL
sunday 8/27/23 DATE
monday DATE
around 2pm TIME
years DATE
one week DATE
one week DATE
one week DATE
155 MONEY
3 weeks DATE
665 MONEY
1038 MONEY
194 MONEY
the day DATE
first ORDINAL
hours TIME
sunday DATE
1 CARDINAL
between 2-6 CARDINAL
8/13/23 DATE
```

```
In [14]: displacy.render(NER(' '.join(df['Review'])),style="ent",jupyter=True)
```

CARDINAL on 8/13/23 DATE . i get a call from them at 8 CARDINAL telling me the van ran out of battery. ok, then use a gas vehicle. not only did it come up that they "attempted" to deliver it, but they don't even reschedule or even try to accommodate. i wasted hours TIME trying to resolve this. reading the previous complaints, i can only concur with walmart+ membership service in general. it's obvious that customer service representatives are located overseas and speak broken english LANGUAGE for starters. why? they are trained for niceties, in the hopes to quell the poor services, it serves no purpose if the simple question why my promised delivery twice in one week DATE did not appear. insult to injury, i contacted the local walmart where the delivery was coming from (1.2 miles QUANTITY away) the operator told me there was no store manager that i can speak to, and referred to the online customer service representative. while it may be difficult to not shop at walmart due to their obviously very competitive prices versus their competitors, the reason they are able to do this is because they cut corners everywhere else. whether it's the terrible conditions for their employees, or the overseas labor they use as their supposed customer service department, that don't speak english LANGUAGE and respond only with automated scripted answers, or the lack of any type of sales support at all once the initial purchase is completed, until we as consumers stop giving them our billions of dollars MONEY each

Chunking: combining words to form meaningful phrases, following the POS tagging.

Unit Test

```
In [18]: import nltk
         from nltk import ne_chunk #importing chunk library from nltk
         from nltk.tokenize import word_tokenize

         text_from_bbc_2 = "The US is conducting unarmed UAV flights over Gaza, as well as providing advice and assistance to support

         token = word_tokenize(text_from_bbc_2) #tokenizing

         tags = nltk.pos_tag(token) #tagging

         chunk = ne_chunk(tags)

         chunk
```

```
Out[18]: Tree('S', [('The', 'DT'), Tree('ORGANIZATION', [('US', 'NNP')]), ('is', 'VBZ'), ('conducting', 'VBG'), ('unarmed', 'JJ'),
Tree('ORGANIZATION', [('UAV', 'NNP')]), ('flights', 'NNS'), ('over', 'IN'), Tree('LOCATION', [('Gaza', 'NNP')]), (',',
','), ('as', 'RB'), ('well', 'RB'), ('as', 'IN'), ('providing', 'VBG'), ('advice', 'NN'), ('and', 'CC'), ('assistance', 'N
N'), ('to', 'TO'), ('support', 'VB'), ('our', 'PRP$'), Tree('GPE', [('Israeli', 'JJ')]), ('partner', 'NN'), ('as', 'IN'),
('they', 'PRP'), ('work', 'VBP'), ('on', 'IN'), ('their', 'PRP$'), ('hostage', 'NN'), ('recovery', 'NN'), ('efforts', 'NNS
'), (',', ','), ('the', 'DT'), Tree('ORGANIZATION', [('Pentagon', 'NNP')]), ("'s", 'POS'), ('statement', 'NN'), ('on', 'IN
'), ('Friday', 'NNP'), ('said.The', 'POS'), ('confirmation', 'NN'), ('comes', 'VBZ'), ('after', 'IN'), ('reporters', 'NNS
'), ('spotted', 'VBD'), ('MQ-9', 'JJ'), ('Reapers', 'NNS'), (',', ','), ('usually', 'RB'), ('operated', 'VBN'), ('by', 'IN
'), Tree('GPE', [('American', 'JJ')]), ('special', 'JJ'), ('forces', 'NNS'), (',', ','), ('circling', 'VBG'), Tree('PERSON
', [('Gaza', 'NNP')]), ('on', 'IN'), Tree('GPE', [('Flightradar24', 'NNP')]), (',', ','), ('a', 'DT'), ('publicly', 'RB'),
('available', 'JJ'), ('flight-tracking', 'JJ'), ('website', 'NN'), ('.', '.')])
```

Text Mining Techniques and Algorithm Application

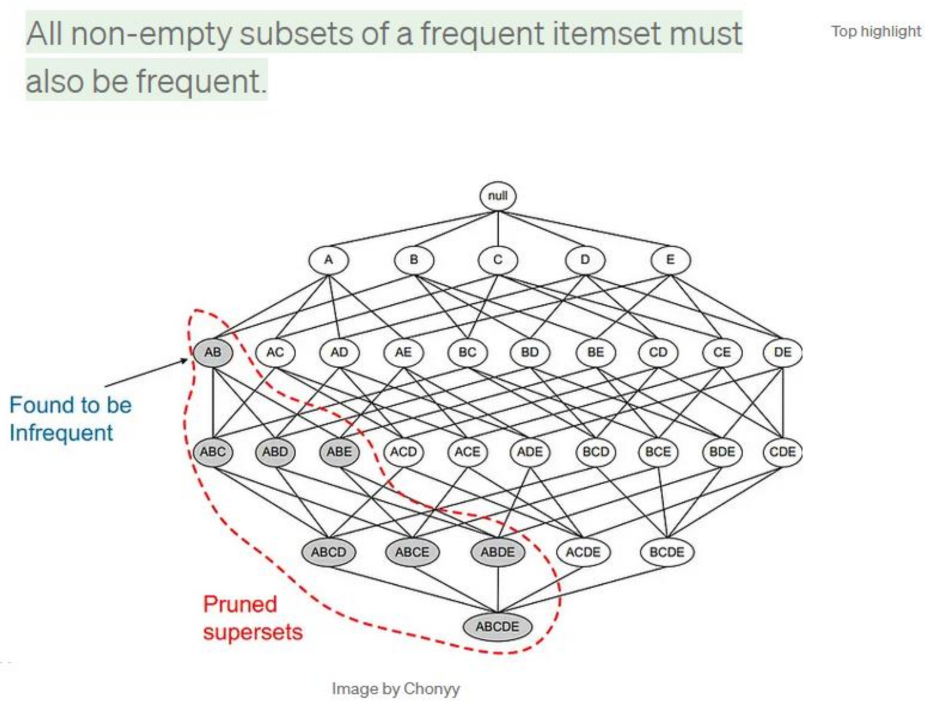
iii. Apply at least two different text mining techniques (e.g., clustering, classification, or association rule mining - Apriori) to identify exciting patterns in the data.

Text Mining Techniques and Algorithm Application

Association rule mining: a method to find the underlying relationships between different items.

Association rule mining methods: Apriori

Apriori: It takes a bottom-up strategy. The method began by going over each and every item in the itemset list. Next, self-joining is used to generate the candidates. One item at a time, the method lengthen the itemsets. Every level involves the subset test, and itemsets containing uncommon subsets are pruned. Until no more successful itemsets can be obtained from the data, the Apriori method repeat the procedure.



Pattern Evaluation and Analysis: Metrics for evaluating patterns are essential for determining the significance and applicability of patterns found by data mining. These metrics guide the selection of important trends and aid in measuring the efficacy of the mining process.

Apriori algorithm's concepts

1. Support
2. Confidence

3. Lift

1.**Support**: A proportion of transactions with an itemset.

$$\text{Support}(X) = \frac{\text{Number of transactions containing } X}{\text{Total number of transactions}}$$

2.**Confidence**: The chance that if item X is purchased, item Y will be purchased as well.

$$\text{Confidence}(X \rightarrow Y) = \frac{\text{Number of transactions containing } X \text{ and } Y}{\text{Number of transactions containing } X}$$

3.**Lift (X -> Y):** the rise in Y's sales ratio in response to X's sale.

$$\text{Lift}(X \rightarrow Y) = \frac{(\text{Confidence}(X \rightarrow Y))}{\text{Support}(Y))}$$

Lift = 1 :- Products X and Y are not related in any way.

Lift > 1 :- It is more likely that products X and Y will be purchased together.

Lift < 1 :- It's rare that two products will be purchased together.

Because there are so many possible possibilities, this procedure can be extremely tedious hence it is recommended to carry out the following steps in order to expedite the process:

1. Maintain a minimum level for confidence and support: the elements that have a minimum value for co-occurrence with other items (like confidence) and a specific default existence (like support) are the only ones for which we are looking for rules.
2. Remove any subsets with support values greater than the minimal threshold.
3. Choose every rule from the subsets whose confidence value is greater than the minimal threshold.
4. Sort the rules in Lift's descending order.

Data source: Market_Basket_Optimisation.csv

https://www.kaggle.com/datasets/sindraanthony9985/marketing-data-for-a-supermarket-in-united-states/data

Viewing the number of rows and columns.

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt


        dataset = pd.read_csv('D:\Market_Basket_Optimisation.csv')
        dataset.shape

Out[1]: (7500, 20)
```

Viewing the first and the last 5 rows.

```
In [2]: dataset.head()
Out[2]:
```

	shrimp	almonds	avocado	vegetables mix	green grapes	whole weat flour	yams	cottage cheese	energy drink	tomato juice	low fat yogurt	green tea	honey	salad	mineral water	salmon	antioxydant juice	froz smooth
0	burgers	meatballs	eggs	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
1	chutney	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
2	turkey	avocado	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
3	mineral water	milk	energy bar	whole wheat rice	green tea	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
4	low fat yogurt	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na

```
In [3]: dataset.tail()
Out[3]:
```

	shrimp	almonds	avocado	vegetables mix	green grapes	whole weat flour	yams	cottage cheese	energy drink	tomato juice	low fat yogurt	green tea	honey	salad	mineral water	salmon	antioxydan juice
7495	butter	light mayo	fresh bread	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7496	burgers	frozen vegetables	eggs	french fries	magazines	green tea	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7497	chicken	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7498	escalope	green tea	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7499	eggs	frozen smoothie	yogurt cake	low fat yogurt	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Converting the dataframe into a list of lists.

```
In [4]: #convert pandas dataframe into a list of lists
records = []
for i in range(0, 7499):
    records.append([str(dataset.values[i,j]) for j in range(0, 3)])
```

```
In [5]: print(records[0])
```

```
['burgers', 'meatballs', 'eggs']
```

```
In [6]: #generate a table
results = pd.DataFrame(records)
results.head(10)
```

Out[6]:

	0	1	2
0	burgers	meatballs	eggs
1	chutney	nan	nan
2	turkey	avocado	nan
3	mineral water	milk	energy bar
4	low fat yogurt	nan	nan
5	whole wheat pasta	french fries	nan
6	soup	light cream	shallot
7	frozen vegetables	spaghetti	green tea
8	french fries	nan	nan
9	eggs	pet food	nan

Application of Apriori algorithm.

```
In [7]: #Applying Apriori
from apyori import apriori

#min_length=2 minimum 2 items for each row

association_rules = apriori(records, min_support=0.0045, min_confidence=0.2, min_lift=2, min_length=2)
association_results = list(association_rules)


#total no of rules

print(len(association_results))
```

```
5
```

Parameters of Apriori algorithm:

min_support: chooses the items whose support values are higher than the parameter's value.

min_confidence: filters the rules whose confidence exceeds the parameter-specified confidence threshold.

min_lift: sets out the short-listed rules' minimum lift value.

min_length: The minimum number of elements in the rules that the user desires.

max_length: The maximum number of elements in the rules that the user desires

Choosing the first parameter to be extracted from the lists of lists.

```
In [8]: print(association_results[0])

RelationRecord(items=frozenset({'pasta', 'escalope'}), support=0.004800640085344712, ordered_statistics=[OrderedStatistic
(items_base=frozenset({'pasta'}), items_add=frozenset({'escalope'}), confidence=0.4044943820224719, lift=9.30461156682980
6)])
```

The first item of the list (grocery items):

It is evident that escalope and pasta are frequently purchased together. For the first rule, the support value is 0.0048. The value here is calculated by dividing the total number of transactions by the number of transactions that contain pasta. With a confidence level of 0.4044, the rule indicates that 40.44% of all transactions including pasta also contain escalope as well. The lift of 9.30 indicates that, in comparison to the default likelihood of the sale of escalope, escalope is 9.30 times more likely to be purchased by consumers who purchase pasta.

The following screenshot displays the rule, the support, the confidence, and lift for each rule.

```
In [9]: for item in association_results:

    pair = item[0]
    items = [x for x in pair]
    print("Rule: " + items[0] + " -> " + items[1] + " "+ "Support: " + str(item[1]) +" "+ "Confidence: " + str(item[2][0][2]

Rule: pasta -> escalope Support: 0.004800640085344712 Confidence: 0.4044943820224719 Lift: 9.304611566829806
Rule: shrimp -> frozen vegetables Support: 0.015068675823443126 Confidence: 0.21523809523809526 Lift: 2.325749965692329
Rule: tomatoes -> frozen vegetables Support: 0.01266835578077077 Confidence: 0.20084566596194506 Lift: 2.17023292370119
Rule: ground beef -> herb & pepper Support: 0.011334844645952793 Confidence: 0.23224043715846993 Lift: 2.6793400588482554
Rule: ground beef -> pepper Support: 0.0050673423123083075 Confidence: 0.2209302325581395 Lift: 2.548855098389982
```

```
In [10]: for item in association_results:

    # first index of the inner list
    # Contains base item and add item
    pair = item[0]
    items = [x for x in pair]
    print("Rule: " + items[0] + " -> " + items[1])

    #second index of the inner list
    print("Support: " + str(item[1]))

    #third index of the list located at 0th
    #of the third index of the inner list

    print("Confidence: " + str(item[2][0][2]))
    print("Lift: " + str(item[2][0][3]))
    print("=====================================")
```

```
Rule: pasta -> escalope
Support: 0.004800640085344712
Confidence: 0.4044943820224719
Lift: 9.304611566829806
=====================================
Rule: shrimp -> frozen vegetables
Support: 0.015068675823443126
Confidence: 0.21523809523809526
Lift: 2.325749965692329
=====================================
Rule: tomatoes -> frozen vegetables
Support: 0.012668355780770777
Confidence: 0.20084566596194506
Lift: 2.17023292370119
=====================================
Rule: ground beef -> herb & pepper
Support: 0.011334844645952793
Confidence: 0.23224043715846993
Lift: 2.6793400588482554
=====================================
Rule: ground beef -> pepper
Support: 0.005067342312308375
Confidence: 0.2209302325581395
Lift: 2.548855098389982
=====================================
```

In [11]:
```python
#generate a table
results = pd.DataFrame(association_results)
results.head(10)
```

Out[11]:

	items	support	ordered_statistics
0	(pasta, escalope)	0.004801	[((pasta), (escalope), 0.4044943820224719, 9.3...
1	(shrimp, frozen vegetables)	0.015069	[((shrimp), (frozen vegetables), 0.21523809523...
2	(tomatoes, frozen vegetables)	0.012668	[((tomatoes), (frozen vegetables), 0.200845665...
3	(ground beef, herb & pepper)	0.011335	[((herb & pepper), (ground beef), 0.2322404371...
4	(ground beef, pepper)	0.005067	[((pepper), (ground beef), 0.2209302325581395,...

Apriori with one-hot encoding.

In [13]:
```python
#Apriori Algorithm and One-Hot Encoding

#Apriori's algorithm transforms True/False or 1/0.
#Using TransactionEncoder, we convert the list to a One-Hot Encoded Boolean list.
#Products that customers bought or did not buy during shopping will now be represented by values 1 and 0.


#Let's transform the list, with one-hot encoding

from mlxtend.preprocessing import TransactionEncoder

a = TransactionEncoder()
a_data = a.fit(dataset).transform(dataset)
df = pd.DataFrame(a_data,columns=a.columns_)
df = df.replace(False,0)
df
```

Out[13]:

		a	b	c	d	e	f	g	h	i	...	p	r	s	t	u	v	w	x	y	z
0	0	0	0	0	0	0	0	0	True	True	...	True	True	True	0	0	0	0	0	0	0
1	0	True	0	0	True	0	0	0	0	0	...	0	0	True	0	0	0	0	0	0	0
2	0	True	0	True	True	0	0	0	0	0	...	0	0	0	0	0	True	0	0	0	0
3	True	True	True	0	0	True	0	True	0	True	...	0	0	True	True	0	True	0	True	0	0
4	True	True	0	0	0	True	0	True	0	0	...	True	True	True	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
7495	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
7496	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
7497	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
7498	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
7499	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

7500 rows × 26 columns

Clustering: *The process of grouping all of the data into groups or clusters according to the patterns seen in the data is called clustering. There is no target or dependent variable in this* *unsupervised learning problem, which implies there isn't a target to forecast. Stated differently, group comparable observations together and create distinct groups according to features and characteristics. This approach also aids in finding trends in large datasets and breaking them down into smaller groups or subsets.*

Data Clustering Techniques in Python

- K-means clustering
- Gaussian mixture models
- Spectral clustering

Gaussian Mixture Model (GMM)

Because Gaussian distributions have well-defined characteristics like mean, variance, and covariance, these models are helpful.

Unit Test: 1st Data source

https://www.kaggle.com/datasets/kondapuramshivani/mall-customerscsv

Viewing data source.

```
In [166]: import pandas as pd

          df = pd.read_csv("D:\\Mall_Customers.csv")

          print(df.head())
```

```
   CustomerID  Gender  Age  Annual Income (k$)  Spending Score (1-100)
0           1    Male   19                  15                      39
1           2    Male   21                  15                      81
2           3  Female   20                  16                       6
3           4  Female   23                  16                      77
4           5  Female   31                  17                      40
```

Applying Gaussian Mixture Model (GMM) on the source data, age vs spending money.

In [170]:

```
#initialize an instance of the GaussianMixture class
from sklearn.mixture import GaussianMixture

#inputs = age and spending score
X = df[['Age', 'Spending Score (1-100)']].copy()


#considering three clusters and fit the model to inputs (age and spending score):
n_clusters = 4
gmm_model = GaussianMixture(n_components=n_clusters)
gmm_model.fit(X)

#cluster lables
cluster_labels = gmm_model.predict(X)
X = pd.DataFrame(X)
X['cluster'] = cluster_labels


#plot each cluster within a for-loop
for k in range(0,n_clusters):
    data = X[X["cluster"]==k]
    plt.scatter(data["Age"],data["Spending Score (1-100)"])


#format out plot
plt.title("Clusters Identified by Guassian Mixture Model")
plt.ylabel("Spending Score (1-100)")
plt.xlabel("Age")
plt.show()
```

Outcomes:

- Spending is lower between the ages of 15 and 65 (Green) than it is between the ages of 20 and 40 (red and orange) and 40 and 70 (blue). It indicates that expenditures are declining relative to age increase.
- Most of the youth (those between the ages of 20 and 40 :- red and orange) spend more than the elderly do.

Unit Test: 2nd Data source:

https://www.kaggle.com/code/thiagopanini/predicting-credit-risk-eda-viz-pipeline/notebook

Applying Gaussian Mixture Model (GMM) on the source data, age vs credit values.

Viewing data source.

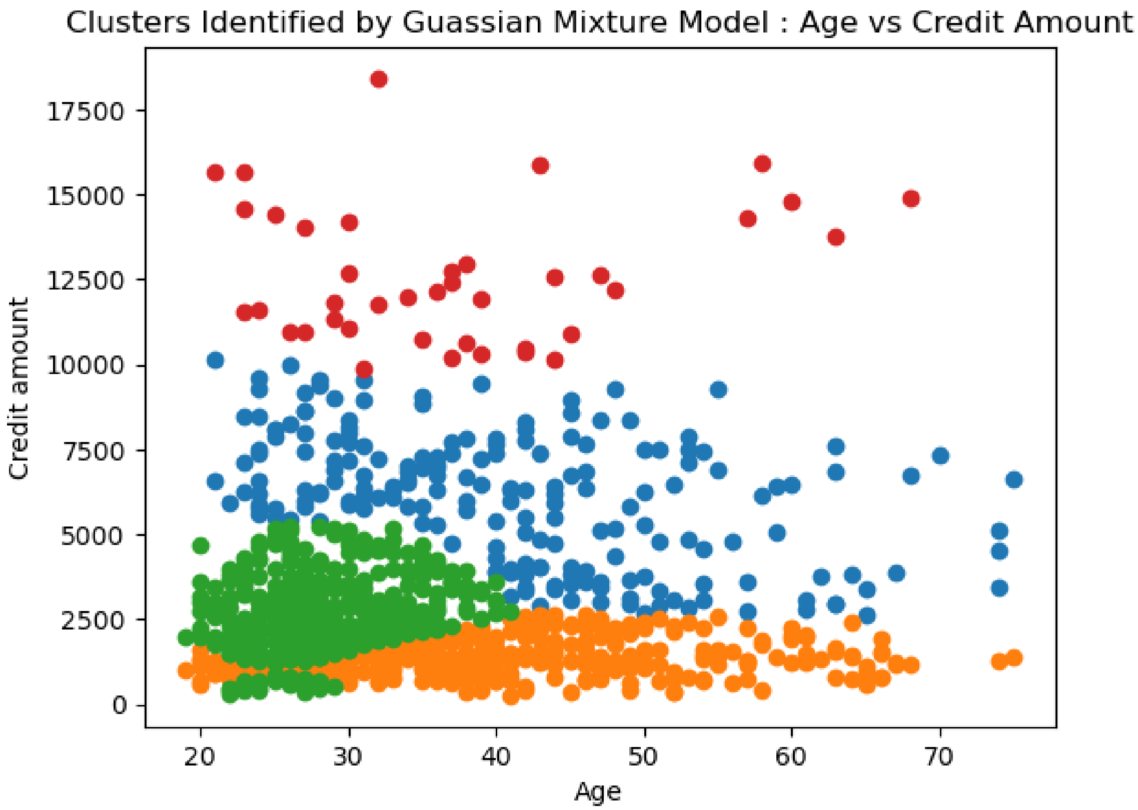
```
In [11]: import pandas as pd

         #https://www.kaggle.com/code/thiagopanini/predicting-credit-risk-eda-viz-pipeline/notebook
         df = pd.read_csv("D:\\german_credit_data.csv")

         print(df.head())
```

```
   Unnamed: 0  Age     Sex  Job Housing Saving accounts Checking account  \
0           0   67    male    2     own             NaN           little
1           1   22  female    2     own          little         moderate
2           2   49    male    1     own          little              NaN
3           3   45    male    2    free          little           little
4           4   53    male    2    free          little           little

   Credit amount  Duration              Purpose  Risk
0           1169         6             radio/TV  good
1           5951        48             radio/TV   bad
2           2096        12            education  good
3           7882        42  furniture/equipment  good
4           4870        24                  car   bad
```

Application: age vs credit values

```
In [27]: #initialize an instance of the GaussianMixture class
         from sklearn.mixture import GaussianMixture

         #inputs = age and spending score
         Y = df[['Age', 'Credit amount']].copy()


         #considering three clusters and fit the model to inputs (age and spending score):
         n_clusters = 4
         gmm_model = GaussianMixture(n_components=n_clusters)
         gmm_model.fit(Y)

         #cluster lables
         cluster_labels = gmm_model.predict(Y)
         Y = pd.DataFrame(Y)
         Y['cluster'] = cluster_labels


         #plot each cluster within a for-loop
         for k in range(0,n_clusters):
             data = Y[Y["cluster"]==k]
             plt.scatter(data["Age"],data["Credit amount"])


         #format out plot
         plt.title("Clusters Identified by Guassian Mixture Model : Age vs Credit Amount")
         plt.ylabel("Credit amount")
         plt.xlabel("Age")
         plt.show()
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory
leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment v
ariable OMP_NUM_THREADS=4.
  warnings.warn(
```



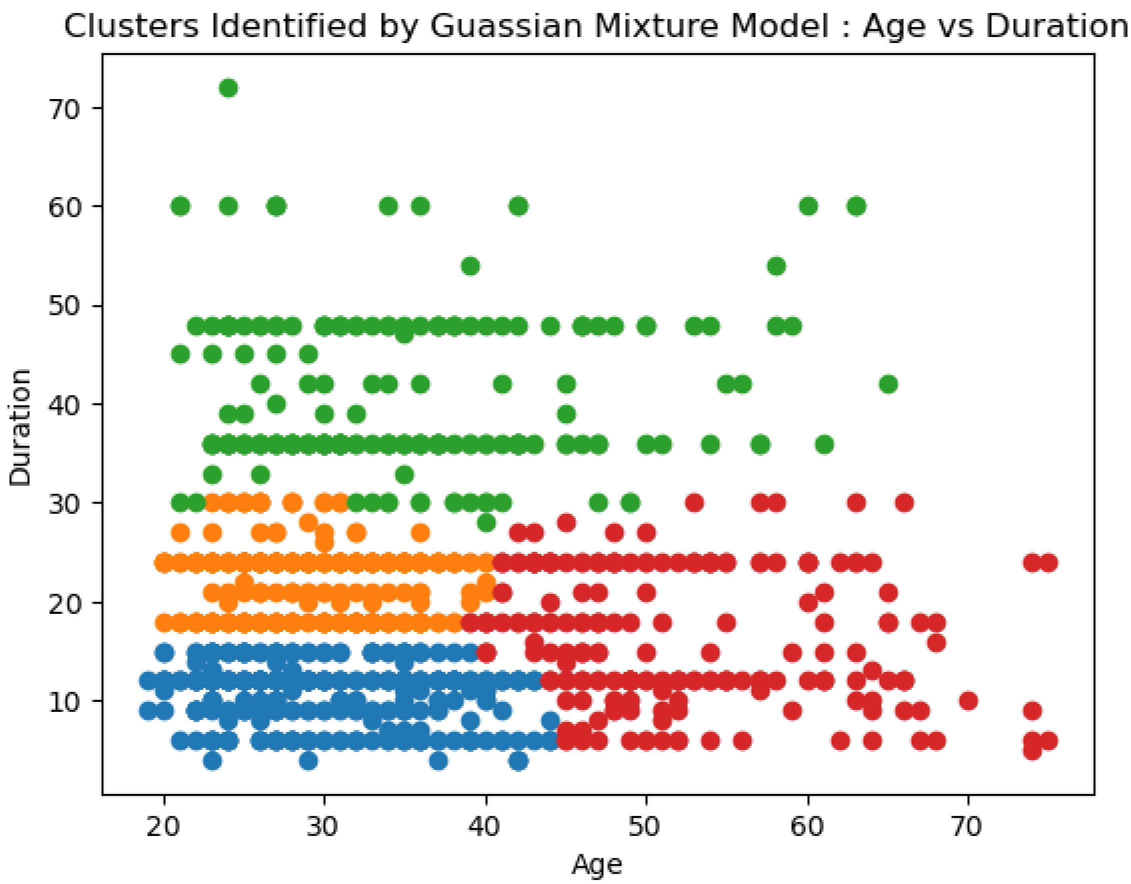
Outcomes:

- The majority of customers (orange) are between their ages of 20 and 70 and have received up to 2,500 loan values.
- Approximately fewer individuals (red) between the ages of 20 and 70 have received loan amount between 10,000 and 18,000.

Application: age vs credit facility's duration.

In [28]:
```
#initialize an instance of the GaussianMixture class
from sklearn.mixture import GaussianMixture

#inputs = age and spending score
Y = df[['Age', 'Duration']].copy()


#considering three clusters and fit the model to inputs (age and spending score):
n_clusters = 4
gmm_model = GaussianMixture(n_components=n_clusters)
gmm_model.fit(Y)

#cluster lables
cluster_labels = gmm_model.predict(Y)
Y = pd.DataFrame(Y)
Y['cluster'] = cluster_labels


#plot each cluster within a for-loop
for k in range(0,n_clusters):
    data = Y[Y["cluster"]==k]
    plt.scatter(data["Age"],data["Duration"])


#format out plot
plt.title("Clusters Identified by Guassian Mixture Model : Age vs Duration")
plt.ylabel("Duration")
plt.xlabel("Age")
plt.show()
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory
leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment v
ariable OMP_NUM_THREADS=4.
  warnings.warn(
```



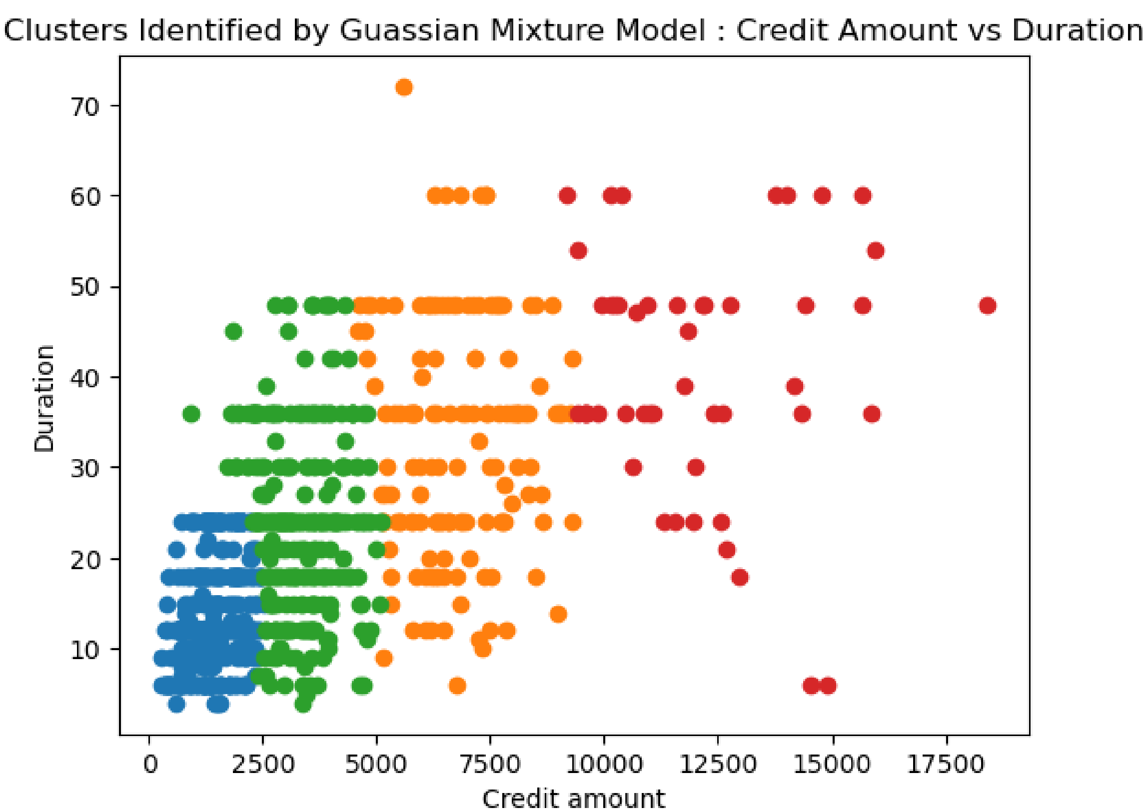
Observations:

- Consumers in their 20s to 45s (blue) have been given shorter repayment terms by the financial institutes; in contrast, customers in their 25s to 65s have been given roughly longer repayment terms (green), such as 60 or 70 months.
- Those in the age range of 40 to 75 have been granted a loan repayment duration of up to 30 months (red).

Application: credit facility amounts vs credit facility durations.

In [29]:
```python
#initialize an instance of the GaussianMixture class
from sklearn.mixture import GaussianMixture

#inputs = age and spending score
Y = df[['Credit amount', 'Duration']].copy()


#considering three clusters and fit the model to inputs (age and spending score):
n_clusters = 4
gmm_model = GaussianMixture(n_components=n_clusters)
gmm_model.fit(Y)

#cluster lables
cluster_labels = gmm_model.predict(Y)
Y = pd.DataFrame(Y)
Y['cluster'] = cluster_labels


#plot each cluster within a for-loop
for k in range(0,n_clusters):
    data = Y[Y["cluster"]==k]
    plt.scatter(data["Credit amount"],data["Duration"])


#format out plot
plt.title("Clusters Identified by Guassian Mixture Model : Credit Amount vs Duration")
plt.ylabel("Duration")
plt.xlabel("Credit amount")
plt.show()
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment v
ariable OMP_NUM_THREADS=4.
  warnings.warn(
```



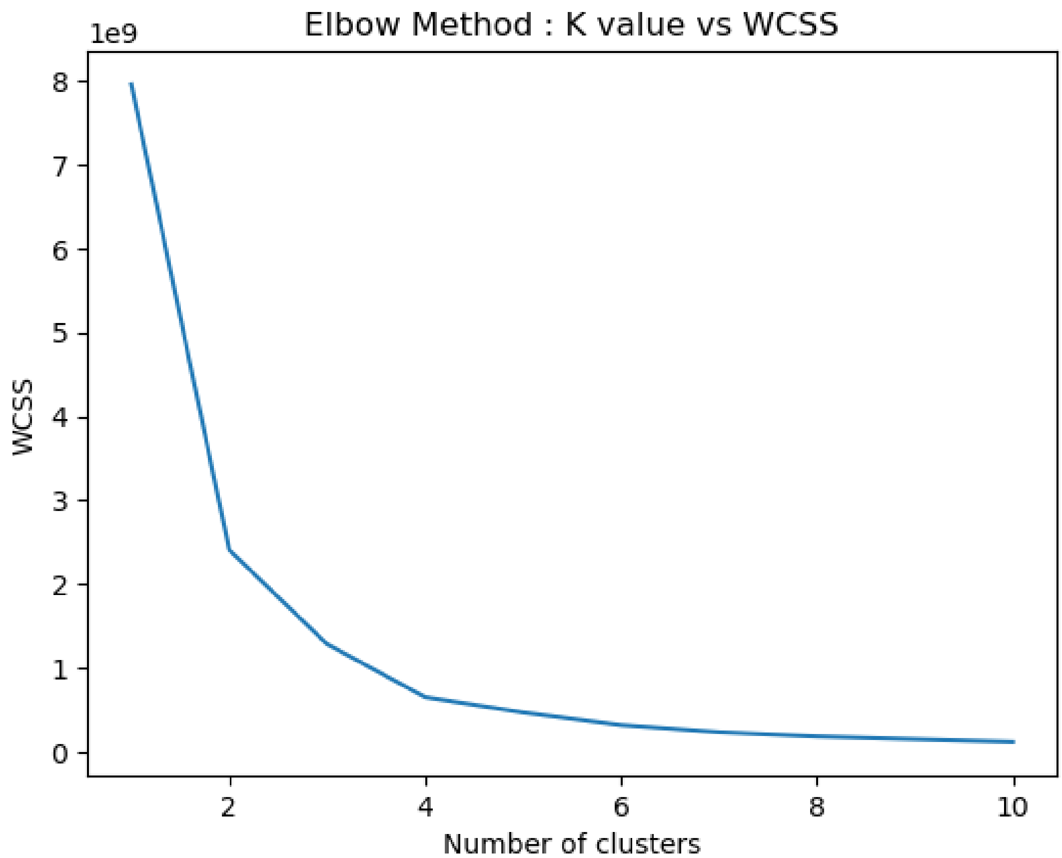
Observations:

- Higher credit facilities have a longer period of time to settle the credit facilities (orange and red), whereas lower credit facilities have shorter repayment terms (blue).
- Credit facilities approximately in between 5,000 to 10,000 have a long payback period—70 months (orange).

K-Means: The ideal value of the K is determined using the k-Means Elbow method, which is a popular unsupervised machine learning algorithm. The method finds the WCSS (Within-Cluster Sum of Square), which is the total squared distance between a cluster centroid and all of the cluster's points.

```
In [39]: Z = df.iloc[:, [7, 8]].values

         from sklearn.cluster import KMeans
         wcss = []
         for i in range(1, 11):
             kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
             kmeans.fit(Z)
             wcss.append(kmeans.inertia_)

         plt.plot(range(1, 11), wcss)
         plt.title("Elbow Method : K value vs WCSS")
         plt.xlabel('Number of clusters')
         plt.ylabel('WCSS')
         plt.show()
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` wi
ll change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory
```



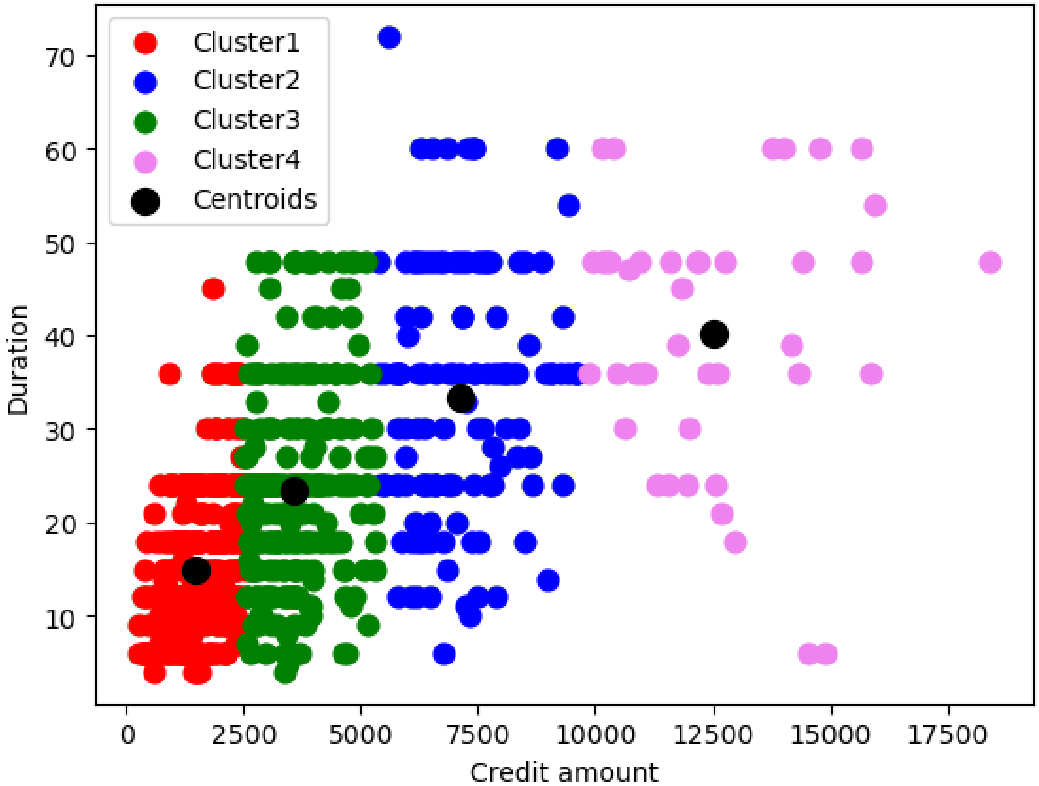
The K value, or the ideal number of clusters, is 4, which is also the point at which the elbow shape is formed.

```
In [42]: #train the model on the input data with a number of clusters 4

kmeans = KMeans(n_clusters = 4, init = "k-means++", random_state = 42)
y_kmeans = kmeans.fit_predict(X)


plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 60, c = 'red', label = 'Cluster1')
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 60, c = 'blue', label = 'Cluster2')
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 60, c = 'green', label = 'Cluster3')
plt.scatter(X[y_kmeans == 3, 0], X[y_kmeans == 3, 1], s = 60, c = 'violet', label = 'Cluster4')

plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 100, c = 'black', label = 'Centroids')

plt.xlabel('Credit amount')
plt.ylabel('Duration')
plt.legend()

plt.show()
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` wi
ll change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory
leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment v
ariable OMP_NUM_THREADS=4.
  warnings.warn(
```

Make use of the scatter plot to see the clusters.

There are four clusters in all, each with a black centroid and four distinct color visualizations.



- Red: lower credit amounts, mostly within 20 months, but occasionally exceeding 40 months and falling short of 50 months
- Green: Medium- credit amounts, typically within 25 months, but occasionally less than around 50 months.
- Blue: High credit amounts within 60 months, but not often beyond 70 months
- Violet: Excessive credit values, primarily for periods of 15 to 60 months

iv. Discuss the fundamental principles of the chosen techniques and the algorithms used, along with their practical applications.

v. Implement scalable pattern discovery techniques on the dataset to identify and analyze frequent patterns, sequential patterns, and sub-graph patterns.

v. Scalable Pattern Discovery

Data Source: Sample_Data_set_for_Task_3.xlsx

Reading the data source and viewing the first and the last 5 rows.

In [12]:
```
import pandas as pd

#reading the excel data source
reviews_df = pd.read_excel("D:\Sample_Data_set_for_Task_3.xlsx")
```

In [13]:
```
reviews_df.head()
```

Out[13]:

	ID	Review Text	Rating	Product_ID	Timestamp
0	1	Great product, excellent quality, and fast shi...	5	P123	2023-04-01
1	2	The item arrived damaged, but customer service...	3	P456	2023-04-01
2	3	The product did not meet my expectations.	2	P789	2023-04-02
3	4	Easy to use, and the results are amazing!	5	P159	2023-04-02
4	5	Overpriced for the quality offered.	2	P951	2023-04-03

In [14]:
```
reviews_df.tail()
```

Out[14]:

	ID	Review Text	Rating	Product_ID	Timestamp
95	96	Exceptional product, I highly recommend it.	5	P238	2023-05-18
96	97	The product didn't work properly, had to retur...	1	P339	2023-05-19
97	98	The product is decent, but I expected better p...	3	P440	2023-05-19
98	99	High-quality product, definitely worth the inv...	5	P541	2023-05-20
99	100	The product is not as advertised, very disappo...	1	P642	2023-05-20

In [15]:
```
#Viewing Review Text column
print(reviews_df["Review Text"])
```

```
0     Great product, excellent quality, and fast shi...
1     The item arrived damaged, but customer service...
2             The product did not meet my expectations.
3             Easy to use, and the results are amazing!
4                   Overpriced for the quality offered.
                              ...
95          Exceptional product, I highly recommend it.
96    The product didn't work properly, had to retur...
97    The product is decent, but I expected better p...
98    High-quality product, definitely worth the inv...
99    The product is not as advertised, very disappo...
Name: Review Text, Length: 100, dtype: object
```

In [16]:
```
# create the label
reviews_df["is_bad_review"] = reviews_df["Rating"].apply(lambda x: 1 if x < 5 else 0)
#print(reviews_df["is_bad_review"])


#select only relevant columns
reviews_df = reviews_df[["Review Text", "is_bad_review"]]
reviews_df.head()
```

Out[16]:

	Review Text	is_bad_review
0	Great product, excellent quality, and fast shi...	0
1	The item arrived damaged, but customer service...	1
2	The product did not meet my expectations.	1
3	Easy to use, and the results are amazing!	0
4	Overpriced for the quality offered.	1

```
In [17]: #this is to speed up computations - sample data
         reviews_df = reviews_df.sample(frac = 0.1, replace = False, random_state=42)
```

```
In [18]: #eliminate 'No Negative' or 'No Positive' from text
         #need to remove those parts from our texts - data cleaning
         reviews_df["Review Text"] = reviews_df["Review Text"].apply(lambda x: x.replace("No Negative", "").
                                                                     replace("No Positive", ""))
```

```
In [19]: # based on the POS rags, returns the wordnet object value
         from nltk.corpus import wordnet

         def get_wordnet_pos(pos_tag):
             if pos_tag.startswith('J'):
                 return wordnet.ADJ
             elif pos_tag.startswith('V'):
                 return wordnet.VERB
             elif pos_tag.startswith('N'):
                 return wordnet.NOUN
             elif pos_tag.startswith('R'):
                 return wordnet.ADV
             else:
                 return wordnet.NOUN
```

```
In [24]: import string
         from nltk import pos_tag
         from nltk.corpus import stopwords
         from nltk.tokenize import WhitespaceTokenizer
         from nltk.stem import WordNetLemmatizer

         #function - cleaning review text
         def clean_text(text):
             # lower text - simple letters
             text = text.lower()

             # tokenize text (splitting text into words) and remove puncutation
             text = [word.strip(string.punctuation) for word in text.split(" ")]

             # remove words that contain numbers
             text = [word for word in text if not any(c.isdigit() for c in word)]

             # remove stop words - unnecessary words remover
             stop = stopwords.words('english')
             text = [x for x in text if x not in stop]

             # remove empty tokens
             text = [t for t in text if len(t) > 0]

             # pos tag text - assign a tag to every word to define if it corresponds to a noun, a verb etc.
             pos_tags = pos_tag(text)

             # lemmatize text - transform every word into their root form
             text = [WordNetLemmatizer().lemmatize(t[0], get_wordnet_pos(t[1])) for t in pos_tags]

             # remove words with only one letter
             text = [t for t in text if len(t) > 1]

             # join all
             text = " ".join(text)
             return(text)

         # clean Review Text data
         reviews_df["review_clean"] = reviews_df["Review Text"].apply(lambda x: clean_text(x))

         #printing on screen after cleaning Review text
         print(reviews_df["review_clean"])
```

```
83              product arrive defect happy purchase
53        exceptional product highly recommend others
70                        item describe disappointed
45                         product expensive quality
44      high-quality material outstanding performance
39                      surprisingly good quality buy
22                       product average great expect
80                          product meet need return
10           impressive quality great customer support
0       great product excellent quality fast shipping
Name: review_clean, dtype: object
```

Feature engineering:

The explanation of the below python code:

- Adding sentiment analysis (neutrality score, positivity score, negativity score, an overall score that summarizes the previous scores)
- Integrate those 4 values as features in our dataset.
- Vader is a part of the NLTK module specially for sentiment analysis.
- Vader module sorts words into positive and negative categories using a lexicon.
- In order to calculate the sentiment scores, it additionally considers the statements' context.

```
In [25]: from nltk.sentiment.vader import SentimentIntensityAnalyzer

sid = SentimentIntensityAnalyzer()
reviews_df["sentiments"] = reviews_df["Review Text"].apply(lambda x: sid.polarity_scores(x))
reviews_df = pd.concat([reviews_df.drop(['sentiments'], axis=1), reviews_df['sentiments'].apply(pd.Series)], axis=1)

print(reviews_df)
```

```
                                         Review Text  is_bad_review  \
83  Product arrived with a defect, not happy with ...              1   
53  Exceptional product, highly recommend it to ot...              0   
70   The item is not as described, very disappointed.              1   
45      The product is too expensive for its quality.              1   
44  High-quality materials and outstanding perform...              0   
39         Surprisingly good quality, will buy again.              0   
22  The product is average, not as great as I expe...              1   
80   The product doesn't meet my needs, returning it.              1   
10     Impressive quality and great customer support.              0   
0   Great product, excellent quality, and fast shi...              0   

                                review_clean    neg    neu    pos  \
83           product arrive defect happy purchase  0.264  0.330  0.407   
53    exceptional product highly recommend others  0.000  0.589  0.411   
70                     item describe disappointed  0.608  0.392  0.000   
45                       product expensive quality  0.000  1.000  0.000   
44  high-quality material outstanding performance  0.000  0.429  0.571   
39                   surprisingly good quality buy  0.000  0.282  0.718   
22                    product average great expect  0.000  0.423  0.577   
80                        product meet need return  0.000  1.000  0.000   
10      impressive quality great customer support  0.000  0.165  0.835   
0   great product excellent quality fast shipping  0.000  0.339  0.661   

    compound    neg    neu    pos  compound    neg    neu    pos  compound  
83    0.3182  0.435  0.565  0.000   -0.6595  0.435  0.565  0.000   -0.6595  
53    0.4201  0.000  0.682  0.318    0.4201  0.000  0.682  0.318    0.4201  
70   -0.4767  0.326  0.674  0.000   -0.5256  0.326  0.674  0.000   -0.5256  
45    0.0000  0.000  1.000  0.000    0.0000  0.000  1.000  0.000    0.0000  
44    0.6124  0.000  0.500  0.500    0.6124  0.000  0.500  0.500    0.6124  
39    0.6249  0.000  0.440  0.560    0.6249  0.000  0.440  0.560    0.6249  
22    0.6249  0.292  0.708  0.000   -0.5096  0.292  0.708  0.000   -0.5096  
80    0.0000  0.000  1.000  0.000    0.0000  0.000  1.000  0.000    0.0000  
10    0.8779  0.000  0.229  0.771    0.8779  0.000  0.229  0.771    0.8779  
0     0.8316  0.000  0.391  0.609    0.8316  0.000  0.391  0.609    0.8316  
```

Adding number of characters and number of words column.

```
In [26]: #adding no of characters column
reviews_df["nb_chars"] = reviews_df["Review Text"].apply(lambda x: len(x))

#adding no of words column
reviews_df["nb_words"] = reviews_df["Review Text"].apply(lambda x: len(x.split(" ")))
```

```
In [ ]: #add some simple metrics for every text:

#number of characters in the text
#number of words in the text
```

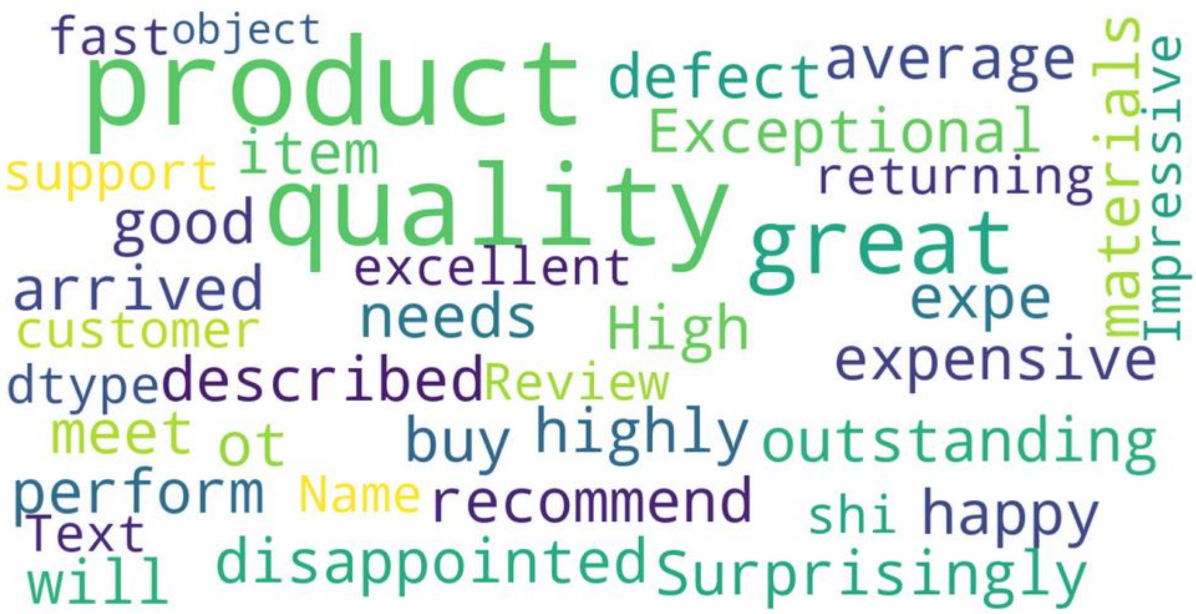
```
In [27]: # create doc2vec vector columns
from gensim.test.utils import common_texts
from gensim.models.doc2vec import Doc2Vec, TaggedDocument

documents = [TaggedDocument(doc, [i]) for i, doc in enumerate(reviews_df["review_clean"].apply(lambda x: x.split(" ")))]

# train a Doc2Vec model with the Review text data
model = Doc2Vec(documents, vector_size=5, window=2, min_count=1, workers=4)

# transform each document into a vector data
doc2vec_df = reviews_df["review_clean"].apply(lambda x: model.infer_vector(x.split(" "))).apply(pd.Series)
doc2vec_df.columns = ["doc2vec_vector_" + str(x) for x in doc2vec_df.columns]
reviews_df = pd.concat([reviews_df, doc2vec_df], axis=1)

print(reviews_df)
```

```
                                        Review Text  is_bad_review  \
83  Product arrived with a defect, not happy with ...              1
53  Exceptional product, highly recommend it to ot...              0
70   The item is not as described, very disappointed.              1
45      The product is too expensive for its quality.              1
44  High-quality materials and outstanding perform...              0
39         Surprisingly good quality, will buy again.              0
22  The product is average, not as great as I expe...              1
80   The product doesn't meet my needs, returning it.              1
10    Impressive quality and great customer support.               0
0   Great product, excellent quality, and fast shi...              0

                                   review_clean    neg    neu    pos  \
83           product arrive defect happy purchase  0.264  0.330  0.407
53    exceptional product highly recommend others  0.000  0.589  0.411
70                    item describe disappointed  0.608  0.392  0.000
45                     product expensive quality  0.000  1.000  0.000
44  high-quality material outstanding performance  0.000  0.429  0.571
39                surprisingly good quality buy  0.000  0.282  0.718
22                 product average great expect  0.000  0.423  0.577
80                     product meet need return  0.000  1.000  0.000
10      impressive quality great customer support  0.000  0.165  0.835
0   great product excellent quality fast shipping  0.000  0.339  0.661

    compound    neg    neu    pos  ...    neu    pos  compound  nb_chars  \
83    0.3182  0.435  0.565  0.000  ...  0.565  0.000   -0.6595        59
53    0.4201  0.000  0.682  0.318  ...  0.682  0.318    0.4201        51
70   -0.4767  0.326  0.674  0.000  ...  0.674  0.000   -0.5256        48
45    0.0000  0.000  1.000  0.000  ...  1.000  0.000    0.0000        45
44    0.6124  0.000  0.500  0.500  ...  0.500  0.500    0.6124        51
39    0.6249  0.000  0.440  0.560  ...  0.440  0.560    0.6249        42
22    0.6249  0.292  0.708  0.000  ...  0.708  0.000   -0.5096        51
80    0.0000  0.000  1.000  0.000  ...  1.000  0.000    0.0000        48
10    0.8779  0.000  0.229  0.771  ...  0.229  0.771    0.8779        46
0     0.8316  0.000  0.391  0.609  ...  0.391  0.609    0.8316        52

    nb_words  doc2vec_vector_0  doc2vec_vector_1  doc2vec_vector_2  \
83        10         -0.032347          0.042652          0.031815
53         7          0.056449         -0.011273         -0.027524
70         8          0.087560         -0.055528         -0.070035
45         8         -0.068945          0.008814          0.024403
44         5          0.035237         -0.100626          0.051644
39         6         -0.061514         -0.052183          0.049508
22        10         -0.065140         -0.078459         -0.068715
80         8          0.017255          0.087680          0.082314
10         6          0.041165          0.057640          0.076072
0          7          0.076666         -0.050150         -0.058772
```

```
    doc2vec_vector_3  doc2vec_vector_4
83         -0.069933         -0.085769
53          0.057146         -0.009971
70         -0.011132          0.057400
45          0.055156          0.039541
44          0.018209         -0.064013
39         -0.089758          0.075699
22          0.070870         -0.024085
80         -0.014926          0.074457
10          0.030686         -0.059177
0           0.065218          0.043386

[10 rows x 22 columns]
```

Use of wordcloud function.

In [33]:
```
# wordcloud function

from wordcloud import WordCloud
import matplotlib.pyplot as plt

def show_wordcloud(data, title = None):
    wordcloud = WordCloud(
        background_color = 'white',
        max_words = 200,
        max_font_size = 40,
        scale = 3,
        random_state = 42
    ).generate(str(data))

    fig = plt.figure(1, figsize = (20, 20))
    plt.axis('off')
    if title:
        fig.suptitle(title, fontsize = 20)
        fig.subplots_adjust(top = 2.3)

    plt.imshow(wordcloud)
    plt.show()

# print wordcloud
show_wordcloud(reviews_df["Review Text"])
```



Outcome: The majority of comments are favorable, indicating that customers are happy with their purchases.

Sorting the list by "nb_words"

In [48]:
```
#printing "nb_words" column
print(reviews_df["nb_words"])
```

```
83    10
53     7
70     8
45     8
44     5
39     6
22    10
80     8
10     6
0      7
Name: nb_words, dtype: int64
```

In [49]:
```
# highest positive sentiment reviews (with more than 5 words)
reviews_df[reviews_df["nb_words"] >= 5].sort_values("nb_words", ascending = False)[["Review Text", "nb_words"]].head(10)
```

Out[49]:

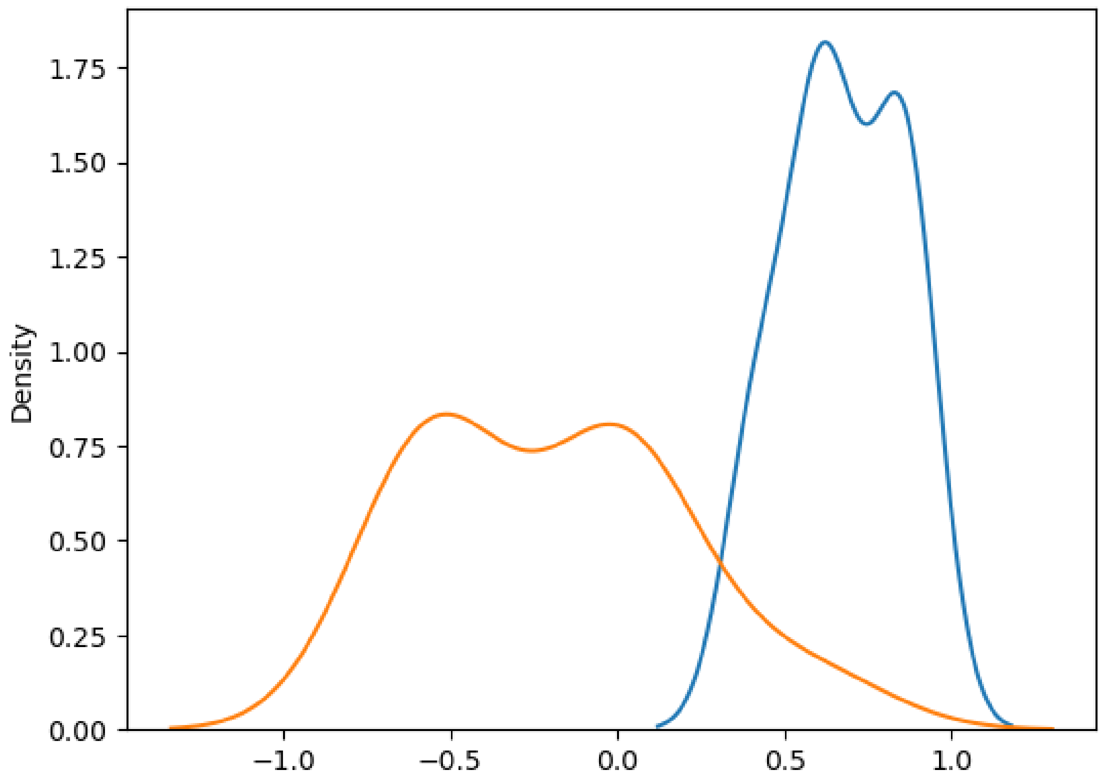
	Review Text	nb_words
83	Product arrived with a defect, not happy with ...	10
22	The product is average, not as great as I expe...	10
70	The item is not as described, very disappointed.	8
45	The product is too expensive for its quality.	8
80	The product doesn't meet my needs, returning it.	8
53	Exceptional product, highly recommend it to ot...	7
0	Great product, excellent quality, and fast shi...	7
39	Surprisingly good quality, will buy again.	6
10	Impressive quality and great customer support.	6
44	High-quality materials and outstanding perform...	5

```python
In [42]: # generating a plot to view positive and negative review feedbacks

import seaborn as sns

for x in [0, 1]:
    subset = reviews_df[reviews_df['is_bad_review'] == x]

    # Draw the density plot
    if x == 0:
        label = "Good reviews"
    else:
        label = "Bad reviews"
    sns.distplot(subset['compound'], hist = False, label = label)
```

```
C:\Users\ravi\AppData\Local\Temp\ipykernel_16752\1875207835.py:13: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(subset['compound'], hist = False, label = label)
C:\Users\ravi\AppData\Local\Temp\ipykernel_16752\1875207835.py:13: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(subset['compound'], hist = False, label = label)
```

The sentiment distribution between positive and negative reviews.

- The blue line indicates customers’ positive reviews.
- The orange line indicates customers’ negative reviews.
- It is evident that Vader module regards positive reviews for the majority of them as extremely positive. The bad reviews, on the other hand, typically have lower compound sentiment scores.



vi. Evaluate the efficiency and scalability of the implemented techniques.

- If low-quality data is present, it may result in erroneous or unlikely patterns.
- Sometimes even though a pattern is visible, it does not indicate anything that is relevant to any goal or point to any relevant information.
- Patterns provide the data additional significance and aid in improving understanding.
- Metrics like accuracy, precision, recall, or any other common assessment metrics that are used to determine how well NLP models perform are not provided by wordclouds.

Pattern Evaluation and Analysis

vii. Discuss the pattern evaluation metrics used to assess the quality and relevance of the discovered patterns.

In NLP, pattern evaluation metrics metrics help quantify how well a model is able to understand, generate, or manipulate language patterns.

3 common pattern of evaluation metrics,

- **Precision** is a classification model's capacity to select just the right information points.
  A model's **recall** is its capacity to locate every important case in a given data set.
  The harmonic mean of recall and precision is the **F1 score**.
- **Accuracy** is the proportion of our forecasts that came true.
- **Confusion Matrix** gives the user a direction to retrace steps and assists in assessing the performance of the model and identifying any issues.

The task mentioned above makes use of sentiment analysis. Sentiment analysis is a type of natural language processing (NLP) in which the sentiment or emotion expressed in a text is identified. While sentiment analysis is not a pattern evaluation metric in and of itself, it frequently uses pattern evaluation metrics to evaluate the effectiveness of sentiment analysis models.

A word cloud is a type of visualization where words are displayed in different sizes to indicate how frequently or how important they are in a particular text. Although word clouds are not widely used as metrics for pattern evaluation, they might function as an additional visual tool to obtain an understanding of the most frequently occurring or notable terms within a dataset.

viii. Investigate techniques for mining various patterns and compare their effectiveness in the given business scenario.

Application of text mining and analysis techniques,

- Facilitates the important insights to be extracted from massive amounts of unstructured text data.
- Assists businesses in comprehending the requirements, preferences, opinions, and comments of their clients. Text data from surveys, social media, product reviews, and customer service contacts can be analyzed for companies using methods like named entity identification, sentiment analysis, and topic modeling.

- Businesses may keep an eye on the tactics, innovations, and performance of their rivals by utilizing text mining and analysis. Businesses can gather and examine text data from sources including news articles, blogs, press releases, and patents by using methods like information extraction, text summarization, and text classification. Businesses can use this to measure their performance, gather competitive knowledge, and spot trends and best practices.

- Businesses can optimize their internal workflows, operations, and procedures with the use of text mining and analysis. Businesses can manage and organize text data from sources including emails, reports, documents, and manuals by using techniques like text clustering, text categorization, and text generation. Businesses can benefit from this by increasing their production, efficiency, quality, and compliance.

Decision Support and Recommendations

ix. Based on the identified patterns and extracted knowledge, provide actionable insights and recommendations to improve customer satisfaction.

Actionable insights and recommendations:

- Examine the favorable evaluations to find recurring themes or attributes that clients value. It is advised to highlight and emphasize these advantageous features in product descriptions and marketing collateral.

- Seek compliments regarding responsiveness, communication, or customer service. Make a commitment to customer service training and make sure that queries from customers are answered quickly and efficiently.
- Examine testimonials about the performance, durability, or quality of the product. Keep or improve the standards of product quality and think about emphasizing these qualities in marketing.
- Recognize clients that consistently express happiness and gratitude for the product. Create a loyalty program to reward and keep these clients, building enduring connections.
- Check to see if satisfied customers' reviews indicate easy returns or exchanges. Make your return and exchange policies clear and visible to prospective customers to build trust.

x. Discuss how the application of text mining and analysis techniques can support decision-making processes in the business context.

Application of text mining and analysis techniques,

- Facilitates the important insights to be extracted from massive amounts of unstructured text data.
- Assists businesses in comprehending the requirements, preferences, opinions, and comments of their clients. Text data from surveys, social media, product reviews, and customer service contacts can be analyzed for companies using methods like named entity identification, sentiment analysis, and topic modeling.

- Businesses may keep an eye on the tactics, innovations, and performance of their rivals by utilizing text mining and analysis. Businesses can gather and examine text data from sources including news articles, blogs, press releases, and patents by using methods like information extraction, text summarization, and text classification. Businesses can use this to measure their performance, gather competitive knowledge, and spot trends and best practices.

- Businesses can optimize their internal workflows, operations, and procedures with the use of text mining and analysis. Businesses can manage and organize text data from sources including emails, reports, documents, and manuals by using techniques like text clustering, text categorization, and text generation. Businesses can benefit from this by increasing their production, efficiency, quality, and compliance.

References

Abld Ali Awan. (2023, Sep). *What is Named Entity Recognition (NER)? Methods, Use Cases, and Challenges*. Retrieved from datacamp: https://www.datacamp.com/blog/what-is-named-entity-recognition-ner

Akash Bokade. (2023, Aug). *NLP workbook + Amazon Review Sentiment analysis*. Retrieved from kaggle: https://www.kaggle.com/code/akashbokade/nlp-workbook-amazon-review-sentiment-analysis

Chetna Khanna. (2021, Feb 10). *Text pre-processing: Stop words removal using different libraries*. Retrieved from Medium: https://towardsdatascience.com/text-pre-processing-stop-words-removal-using-different-libraries-f20bac19929a

Kevin Arvai. (2020, Jul 20). *K-Means Clustering in Python: A Practical Guide*. Retrieved from realpython: https://realpython.com/k-means-clustering-python/

Kevin Arvai. (n.d.). *K-Means Clustering in Python: A Practical Guide*. Retrieved from realpython: https://realpython.com/k-means-clustering-python/

Martin Porter. (2006, Jan). *The Porter Stemming Algorithm*. Retrieved from tartarus: https://www.tartarus.org/~martin/PorterStemmer/

Neri. (2023, Nov 03). *How To Implement Intent Classification In NLP [7 ML & DL Models] With Python Example*. Retrieved from spotintelligence: https://spotintelligence.com/2023/11/03/intent-classification-nlp/

Prateek Majumder. (2023, Sep 13). *Named Entity Recognition (NER) in Python with Spacy*. Retrieved from analyticsvidhya: https://www.analyticsvidhya.com/blog/2021/06/nlp-application-named-entity-recognition-ner-in-python-with-spacy/

Sadrach Pierre & Matthew Urwin. (2017, Oct 14). *builtin*. Retrieved from How to Form Clusters in Python: Data Clustering Methods: https://builtin.com/data-science/data-clustering-python

What is named entity recognition? (n.d.). Retrieved from ibm: https://www.ibm.com/topics/named-entity-recognition