Sentiment Analysis of Customer Feedback: Enhancing Products and Services with Precision

Data Collection and Loading

In [4]:

```
# Importing Libraries
import pandas as pd
from tqdm import tqdm

import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
nltk.download('punkt')

stop_words = set(stopwords.words("english"))
```

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

In [6]:

```
# Data Loading
train_df = pd.read_csv(r"C:\Users\ADMIN\Desktop\Amdari Project\Sentiment Analysis Amdari\datasets\e commerce revi
```

In [8]:

```
test_df = pd.read_csv(r"C:\Users\ADMIN\Desktop\Amdari Project\Sentiment Analysis Amdari\datasets\e commerce revie
```

In [10]:

```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3600010 entries, 0 to 3600009
Data columns (total 2 columns):
 #   Column  Dtype 
---  ------  ----- 
 0   labels  object
 1   text    object
dtypes: object(2)
memory usage: 54.9+ MB
```

In [12]:

```
test_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400000 entries, 0 to 399999
Data columns (total 2 columns):
 #   Column  Non-Null Count   Dtype 
---  ------  --------------   ----- 
 0   labels  400000 non-null  object
 1   text    400000 non-null  object
dtypes: object(2)
memory usage: 6.1+ MB
```

In [14]:

```
train_df.head(10)
```

Out[14]:

	labels	text
0	__label__2	Stuning even for the non-gamer: This sound tra...
1	__label__2	The best soundtrack ever to anything.: I'm rea...
2	__label__2	Amazing!: This soundtrack is my favorite music...
3	__label__2	Excellent Soundtrack: I truly like this soundt...
4	__label__2	Remember, Pull Your Jaw Off The Floor After He...
5	__label__2	an absolute masterpiece: I am quite sure any o...
6	__label__1	Buyer beware: This is a self-published book, a...
7	__label__2	Glorious story: I loved Whisper of the wicked ...
8	__label__2	A FIVE STAR BOOK: I just finished reading Whis...
9	__label__2	Whispers of the Wicked Saints: This was a easy...

In [16]:

```
# get the row at index 6
print(train_df.iloc[6]['text'])
```

```
Buyer beware: This is a self-published book, and if you want to know why--read a few paragraphs! Those 5 star revi
ews must have been written by Ms. Haddon's family and friends--or perhaps, by herself! I can't imagine anyone read
ing the whole thing--I spent an evening with the book and a friend and we were in hysterics reading bits and piece
s of it to one another. It is most definitely bad enough to be entered into some kind of a "worst book" contest. I
can't believe Amazon even sells this kind of thing. Maybe I can offer them my 8th grade term paper on "To Kill a M
ockingbird"--a book I am quite sure Ms. Haddon never heard of. Anyway, unless you are in a mood to send a book to
someone as a joke---stay far, far away from this one!
```

Text Processing

In [19]:

```
#First let change the label
###label 1: 1 and 2 stars ratings ==> negative
###label 2: 4 and 5 stars rating ==> positive
```

In [21]:

```
train_df['labels'].unique()
```

Out[21]:

```
array(['__label__2', '__label__1'], dtype=object)
```

In [23]:

```
##lets map the labels to sentiment words, positive, negative

mapping_values = {
    '__label__1': "negative",
    "__label__2": "positive"
}
```

In [25]:

```
train_df['labels'].map(mapping_values)
```

Out[25]:

```
0          positive
1          positive
2          positive
3          positive
4          positive
             ...
3600005    negative
3600006    negative
3600007    negative
3600008    negative
3600009    positive
Name: labels, Length: 3600010, dtype: object
```

In [27]:

```
#mapping labels columns
train_df['labels'] = train_df['labels'].map(mapping_values)
```

In [29]:

```
test_df['labels'] = test_df['labels'].map(mapping_values)
```

In [31]:

```
train_df.head(10)
```

Out[31]:

	labels	text
0	positive	Stuning even for the non-gamer: This sound tra...
1	positive	The best soundtrack ever to anything.: I'm rea...
2	positive	Amazing!: This soundtrack is my favorite music...
3	positive	Excellent Soundtrack: I truly like this soundt...
4	positive	Remember, Pull Your Jaw Off The Floor After He...
5	positive	an absolute masterpiece: I am quite sure any o...
6	negative	Buyer beware: This is a self-published book, a...
7	positive	Glorious story: I loved Whisper of the wicked ...
8	positive	A FIVE STAR BOOK: I just finished reading Whis...
9	positive	Whispers of the Wicked Saints: This was a easy...

In [33]:

```
test_df.head(10)
```

Out[33]:

	labels	text
0	positive	Great CD: My lovely Pat has one of the GREAT v...
1	positive	One of the best game music soundtracks - for a...
2	negative	Batteries died within a year ...: I bought thi...
3	positive	works fine, but Maha Energy is better: Check o...
4	positive	Great for the non-audiophile: Reviewed quite a...
5	negative	DVD Player crapped out after one year: I also ...
6	negative	Incorrect Disc: I love the style of this, but ...
7	negative	DVD menu select problems: I cannot scroll thro...
8	positive	Unique Weird Orientalia from the 1930's: Exoti...
9	negative	Not an "ultimate guide": Firstly,I enjoyed the...

In [35]:

```
text = "I love this product, it is good"
```

In [37]:

```
nltk.word_tokenize(text)
```

Out[37]:

```
['I', 'love', 'this', 'product', ',', 'it', 'is', 'good']
```

In [39]:

```
# Tokenize the text (split it into words)
words = nltk.word_tokenize(text)
```

In [41]:

```
# Remove stopwords from the text
filtered_words = [word for word in words if word.lower() not in stop_words]

# Reconstruct the text without stopwords
filtered_text = " ".join(filtered_words)

print(filtered_text)
```

```
love product , good
```

In [43]:

```
def remove_stopwords(text):
    """
    this function take a sentence
    tokenize.. the sentence
    filters out stopwords and return a more compactsentence
    """
    words = nltk.word_tokenize(text)
    filtered_words = [word for word in words if word.lower() not in stop_words]
    filtered_text = " ".join(filtered_words)
    return filtered_text
```

In [45]:

```
# Remove stopwords from the text
remove_stopwords(text)
```

```
Out[45]: 'love product , good'
```

```
In [47]: train_df["text"]
```

```
Out[47]: 0          Stuning even for the non-gamer: This sound tra...
         1          The best soundtrack ever to anything.: I'm rea...
         2          Amazing!: This soundtrack is my favorite music...
         3          Excellent Soundtrack: I truly like this soundt...
         4          Remember, Pull Your Jaw Off The Floor After He...
                                          ...
         3600005    Don't do it!!: The high chair looks great when...
         3600006    Looks nice, low functionality: I have used thi...
         3600007    compact, but hard to clean: We have a small ho...
         3600008    what is it saying?: not sure what this book is...
         3600009    Makes My Blood Run Red-White-And-Blue: I agree...
         Name: text, Length: 3600010, dtype: object
```

```
In [49]: train_df["text"].head(10).apply(remove_stopwords)
```

```
Out[49]: 0    Stuning even non-gamer : sound track beautiful...
         1    best soundtrack ever anything . : 'm reading l...
         2    Amazing ! : soundtrack favorite music time , h...
         3    Excellent Soundtrack : truly like soundtrack e...
         4    Remember , Pull Jaw Floor Hearing : 've played...
         5    absolute masterpiece : quite sure actually tak...
         6    Buyer beware : self-published book , want know...
         7    Glorious story : loved Whisper wicked saints ....
         8    FIVE STAR BOOK : finished reading Whisper Wick...
         9    Whispers Wicked Saints : easy read book made w...
         Name: text, dtype: object
```

```
In [51]: train_df["text"].head(10)
```

```
Out[51]: 0    Stuning even for the non-gamer: This sound tra...
         1    The best soundtrack ever to anything.: I'm rea...
         2    Amazing!: This soundtrack is my favorite music...
         3    Excellent Soundtrack: I truly like this soundt...
         4    Remember, Pull Your Jaw Off The Floor After He...
         5    an absolute masterpiece: I am quite sure any o...
         6    Buyer beware: This is a self-published book, a...
         7    Glorious story: I loved Whisper of the wicked ...
         8    A FIVE STAR BOOK: I just finished reading Whis...
         9    Whispers of the Wicked Saints: This was a easy...
         Name: text, dtype: object
```

```
In [57]: ##i would love to see a progress bar when we process for all the 3.6 million reviews
         total_rows = len(train_df)
         tqdm.pandas(total=total_rows)
         train_df['stop words'] = train_df['text'].progress_apply(remove_stopwords)
```

```
100%|██████████████████████████████████████████████████████████████████████| 3600010/3600010 [56:31<00:00, 1061.38
it/s]
```

```
In [59]: train_df
```

Out[59]:

	labels	text	stop words
0	positive	Stuning even for the non-gamer: This sound tra...	Stuning even non-gamer : sound track beautiful...
1	positive	The best soundtrack ever to anything.: I'm rea...	best soundtrack ever anything . : 'm reading l...
2	positive	Amazing!: This soundtrack is my favorite music...	Amazing ! : soundtrack favorite music time , h...
3	positive	Excellent Soundtrack: I truly like this soundt...	Excellent Soundtrack : truly like soundtrack e...
4	positive	Remember, Pull Your Jaw Off The Floor After He...	Remember , Pull Jaw Floor Hearing : 've played...
...	...	...	...
3600005	negative	Don't do it!!: The high chair looks great when...	n't ! ! : high chair looks great first comes b...
3600006	negative	Looks nice, low functionality: I have used thi...	Looks nice , low functionality : used highchai...
3600007	negative	compact, but hard to clean: We have a small ho...	compact , hard clean : small house , really wa...
3600008	negative	what is it saying?: not sure what this book is...	saying ? : sure book supposed . really rehash ...
3600009	positive	Makes My Blood Run Red-White-And-Blue: I agree...	Makes Blood Run Red-White-And-Blue : agree eve...

3600010 rows × 3 columns

In [63]:

```
total_rows = len(test_df)
tqdm.pandas(total=total_rows)
test_df['stop words'] = test_df['text'].progress_apply(remove_stopwords)
```

```
100%|██████████████████████████████████████████████████████████████████| 400000/400000 [14:01<00:00, 475.24
it/s]
```

In [65]:

```
test_df
```

Out[65]:

	labels	text	stop words
0	positive	Great CD: My lovely Pat has one of the GREAT v...	Great CD : lovely Pat one GREAT voices generat...
1	positive	One of the best game music soundtracks - for a...	One best game music soundtracks - game n't rea...
2	negative	Batteries died within a year ...: I bought thi...	Batteries died within year ... : bought charge...
3	positive	works fine, but Maha Energy is better: Check o...	works fine , Maha Energy better : Check Maha E...
4	positive	Great for the non-audiophile: Reviewed quite a...	Great non-audiophile : Reviewed quite bit comb...
...	...	...	...
399995	negative	Unbelievable- In a Bad Way: We bought this Tho...	Unbelievable- Bad Way : bought Thomas son huge...
399996	negative	Almost Great, Until it Broke...: My son reciev...	Almost Great , Broke ... : son recieved birthd...
399997	negative	Disappointed !!!: I bought this toy for my son...	Disappointed ! ! ! : bought toy son loves `` T...
399998	positive	Classic Jessica Mitford: This is a compilation...	Classic Jessica Mitford : compilation wide ran...
399999	negative	Comedy Scene, and Not Heard: This DVD will be ...	Comedy Scene , Heard : DVD disappointment get ...

400000 rows × 3 columns

In [67]:

```
#Create a bag of words and TF-IDF
```

A "Bag of Words" (BoW) is a simple and commonly used technique in natural language processing (NLP) and text analysis to represent text data as numerical features. It is used to transform a collection of text documents into a format that can be processed by machine learning algorithms. The idea behind the Bag of Words model is to disregard the order and structure of words in a text and focus only on the frequency of each word's occurrence.

The key idea is that the order of words and the grammatical structure of sentences are ignored, and the analysis is purely based on the presence or absence of specific words and their frequencies.

TF-IDF, which stands for "Term Frequency-Inverse Document Frequency," is a numerical statistic used in information retrieval and natural language processing (NLP) to evaluate the importance of a word within a document relative to a collection of documents, typically a corpus.

The TF-IDF score provides a measure of how important a term is within a specific document and across a collection of documents. Terms that appear frequently in a document but rarely in other documents receive higher TF-IDF scores, making them indicative of the content of that document.

In [70]:
```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [72]:
```
vectorizer = CountVectorizer() # You can adjust max_features as needed
train_bow = vectorizer.fit_transform(train_df['stop words'])
test_bow = vectorizer.transform(test_df['stop words'])
```

In [74]:
```
tfidf_vectorizer = TfidfVectorizer()  # You can adjust max_features as needed
train_tfidf = tfidf_vectorizer.fit_transform(train_df['stop words'])
test_tfidf = tfidf_vectorizer.transform(test_df['stop words'])
```

MODELLING AND EVALUATION

In [77]:
```
#Vader on normal Sentences
from sklearn.metrics import accuracy_score, classification_report
```

In [79]:
```
import nltk
# download the VADER lexicon and model
nltk.download('vader_lexicon')
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\ADMIN\AppData\Roaming\nltk_data...
```

Out[79]:
```
True
```

In [81]:
```
# import the SentimentIntensityAnalyzer class from vader
from nltk.sentiment.vader import SentimentIntensityAnalyzer

# Vader: pretrain model for analyzing sentiment of sentence
analyzer = SentimentIntensityAnalyzer()
```

In [91]:
```
## test out the sentiment analyzer with an example text

example_text  = "i love the orange flavor, good product"

sentiment_scores = analyzer.polarity_scores(example_text)

# The sentiment_scores dictionary will contain the scores.
print(sentiment_scores)
```

```
{'neg': 0.0, 'neu': 0.36, 'pos': 0.64, 'compound': 0.7964}
```

In [93]:
```
# getting the sentiment scores
compound_score = sentiment_scores['compound']

#now lets make a decision for the cut off for a postitive or negative score
if compound_score > 0:
    sentiment = "Positive"
else:
  sentiment = "Negative"

print(f"The sentiment is {sentiment} (Compound Score: {compound_score})")
```

```
The sentiment is Positive (Compound Score: 0.7964)
```

In [95]:
```
## apply all the text in our dataset, so lets first
## create the function, then we apply the function

def analyze_sentence(sentence, threshold = 0):
  sentiment_scores = analyzer.polarity_scores(sentence)
  compound_score = sentiment_scores['compound']

  if compound_score > threshold:
    sentiment = "positive"
  else:
    sentiment = "negative"

  return sentiment
```

In [97]:
```
inferences_0 = test_df['text'].progress_apply(analyze_sentence)
```

```
100%|████████████████████████████████████████████████████████████████████████| 400000/400000 [15:20<00:00, 434.65
it/s]
```

USING THE ACCURACY METRICS AND CLASSIFIACTION REPORT

```
In [100…  accuracy_score(inferences_0, test_df['labels'])
```

```
Out[100…  0.716675
```

```
In [102…  print(classification_report(test_df['labels'],inferences_0 ))
```

```
              precision    recall  f1-score   support

    negative       0.87      0.51      0.64    200000
    positive       0.65      0.92      0.76    200000

    accuracy                           0.72    400000
   macro avg       0.76      0.72      0.70    400000
weighted avg       0.76      0.72      0.70    400000
```

```
In [104…  # VADER ON STOP WORDS
          # now lets repeat on stopwords, lets see if by removing context irrelvant words we can improve the scores of vade
```

```
In [106…  inferences_1 = test_df['stop words'].progress_apply(analyze_sentence)
```

```
100%|████████████████████████████████████████████████████████████████████████| 400000/400000 [09:12<00:00, 724.37
it/s]
```

```
In [108…  # get the accuracy scores, then the classifcation report
          accuracy_score(inferences_1, test_df['labels'])
```

```
Out[108…  0.68083
```

```
In [110…  print(classification_report(test_df['labels'],inferences_1 ))
```

```
              precision    recall  f1-score   support

    negative       0.86      0.43      0.57    200000
    positive       0.62      0.93      0.75    200000

    accuracy                           0.68    400000
   macro avg       0.74      0.68      0.66    400000
weighted avg       0.74      0.68      0.66    400000
```

```
In [112…  ## TRAINING AND TESTING CUSTOM MODELS: Multinomial NB
          ## choosing it for its simplicity, speed and compatibility with bag of words and tfidf
```

```
In [114…  from sklearn.naive_bayes import MultinomialNB
```

```
In [116…  #create a classifier
          classifier = MultinomialNB()
```

```
In [120…  #fit on bag_of_words
          classifier.fit(train_bow, train_df['labels'])
```

```
Out[120…  ▾ MultinomialNB
          MultinomialNB()
```

```
In [122…  ##lets make predictions and evaluate the model

          y_pred = classifier.predict(test_bow)
          accuracy = accuracy_score(test_df['labels'], y_pred)

          #printing results
          print(f"Accuracy: {accuracy:.2f}")
          print(classification_report(test_df['labels'], y_pred))
```

```
Accuracy: 0.85
              precision    recall  f1-score   support

    negative       0.84      0.86      0.85    200000
    positive       0.85      0.84      0.85    200000

    accuracy                           0.85    400000
   macro avg       0.85      0.85      0.85    400000
weighted avg       0.85      0.85      0.85    400000
```

In [123…

```
#create and train a second classifier on tf-idf
classifier2 = MultinomialNB()
```

In [126…

```
classifier2.fit(train_tfidf, train_df["labels"])
```

Out[126…

```
▾ MultinomialNB
MultinomialNB()
```

In [128…

```
y_pred = classifier.predict(test_tfidf)
accuracy = accuracy_score(test_df['labels'], y_pred)
print(f"Accuracy: {accuracy:.2f}")
print(classification_report(test_df['labels'], y_pred))
```

```
Accuracy: 0.83
              precision    recall  f1-score   support

    negative       0.83      0.84      0.83    200000
    positive       0.84      0.82      0.83    200000

    accuracy                           0.83    400000
   macro avg       0.83      0.83      0.83    400000
weighted avg       0.83      0.83      0.83    400000
```

DEPLOYMENT: INFERENCE SCRIPT AND FLASK APP

In [131…

```
## create an inference function to receive a text, remove stopwords, convert to bow and pass to MUltinomialNB mod

stop_words = set(stopwords.words("english"))
def remove_stopwords(text,stop_words = stop_words):
  words = nltk.word_tokenize(text)
  # Remove stopwords from the text
  filtered_words = [word for word in words if word.lower() not in stop_words]
  # Reconstruct the text without stopwords
  filtered_text = " ".join(filtered_words)
  #print(filtered_text)

  return filtered_text

def inference(text):
  filtered_text = remove_stopwords(text)
  bow = vectorizer.transform([filtered_text])
  sentiment = classifier.predict(bow)
  return sentiment
```

In [133…

```
example_text = "i hate this book."
```

In [135…

```
inference(example_text)
```

Out[135…

```
array(['negative'], dtype='<U8')
```

In [137…

```
## FLASK APP

!pip install Flask
```

```
Requirement already satisfied: Flask in c:\users\admin\anaconda3\lib\site-packages (2.2.5)
Requirement already satisfied: Werkzeug>=2.2.2 in c:\users\admin\anaconda3\lib\site-packages (from Flask) (2.2.3)
Requirement already satisfied: Jinja2>=3.0 in c:\users\admin\anaconda3\lib\site-packages (from Flask) (3.1.3)
Requirement already satisfied: itsdangerous>=2.0 in c:\users\admin\anaconda3\lib\site-packages (from Flask) (2.0.
1)
Requirement already satisfied: click>=8.0 in c:\users\admin\anaconda3\lib\site-packages (from Flask) (8.1.7)
Requirement already satisfied: colorama in c:\users\admin\anaconda3\lib\site-packages (from click>=8.0->Flask) (0.
4.6)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\admin\anaconda3\lib\site-packages (from Jinja2>=3.0->Fl
ask) (2.1.3)
```

In [139…

```python
from flask import Flask

app = Flask(__name__)

@app.route('/')
def inference(text):
  filtered_text = remove_stopwords(text)
  bow = vectorizer.transform([filtered_text])
  sentiment = classifier.predict(bow)
  return sentiment

if __name__ == '__main__':
    app.run()
```

```
 * Serving Flask app '__main__'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server inst
ead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

In [ ]:

```python
if __name__ == '__main__':
    from werkzeug.serving import run_simple
    run_simple('localhost', 9000, app)
```