PROJECT

Machine Learning (CP-640A)

"Sentiment E-commerce Insights Analysis for Reviews on Women's Clothing"

Gia Phat Huynh	235838900
Ebad Shahid	235841420
Thai Son Truong	235841520
Sohail Ahmed Mohammed	235807480
Naveed Munsif	235827540

Wilfrid Laurier University

November 29, 2023

1st TERM

FINAL PROJECT

PREFACE

This project **"Sentiment E-commerce Insights Analysis for Reviews on Women's Clothing"** provides the growing importance of online reviews in shaping consumer choices, especially within the domain of women's clothing e-commerce, underscores the necessity for an advanced sentiment analysis system. As online shopping experiences exponential growth, customer reviews assume a pivotal role in influencing purchasing decisions. The categorization of reviews into five distinct sentiment polarities (strong negative, weak negative, neutral, weak positive, and strong positive) emerges as a crucial aspect in understanding and responding to consumer sentiments effectively. Highlight shortcomings: Current solutions often fall short of capturing the nuanced sentiments expressed in reviews, hindering companies from fully understanding customer perceptions.

This project aims to develop a sentiment analysis designed for E-commerce reviews pertaining to women's clothing. The primary objective is to create a multi-class classifier that accurately classifies sentiments into five polarities. Then, by utilizing machine learning techniques, we seek to amplify customer insights, streamline product evaluations, and support market analysis. Our approach involves bridging the gap between traditional Natural Language Processing (NLP) techniques and machine learning algorithms. This entails analyzing both review text and titles, identifying specific positive and negative terms, and constructing a sentiment lexicon to enhance the precision of our classification process.

ACKNOWLEDGEMENT

We are privileged to express our sense of gratitude to our respected teacher **Ms Yang Liu** whose unparalleled knowledge, moral fiber, and judgment along with her know-how, was an immense support in completing the project. We are also grateful to **Ms Yang Liu,** for the brainwave and encouragement given.

Gia Phat Huynh – 235838900

Ebad Shahid – 235841420

Thai Son Truong – 235841520

Sohail Ahmed Mohammed – 235807480

Naveed Munsif – 235827540

TABLE OF CONTENT:

1) Declaration/Permission:

Wilfrid Laurier University

BATCH 2023

COMPUTER SCIENCE DEPARTMENT

CERTIFICATE

This is to certify that the project titled

"Sentiment E-commerce Insights Analysis for Reviews on Women's Clothing"

is a Bona fide work carried out by the following Master of Applied Computing students;

Gia Phat Huynh	**– 235838900**
Ebad Shahid	**– 235841420**
Thai Son Truong	**– 235841520**
Sohail Ahmed Mohammed	**– 235807480**
Naveed Munsif	**– 235827540**

Under our guidance towards the partial fulfillment of the Requirements for the **1st TERM** course **"Machine Learning (CP-640A)"** of the Master of Applied Computing by Wilfrid Laurier University during the academic year of 2023.

DEPARTMENT OF COMPUTER SCIENCE

WILFRID LAURIER UNIVERSITY
LAURIER
Inspiring Lives.

2) Introduction:

The contemporary digital marketplace is inundated with vast amounts of user-generated content, particularly customer reviews, offering valuable insights into product reception. For the women's clothing E-commerce sector, understanding customer sentiments from these reviews is crucial for market analysis, product evaluation, and enhancing the overall shopping experience. This report details the development of an automated sentiment analysis system specifically tailored for women's clothing E-commerce reviews, utilizing a Naive Bayes approach.

2.1 Problem Definition

In the ever-expanding digital landscape, the sheer volume of E-commerce reviews necessitates advanced tools to extract meaningful insights. This project addresses the challenge of categorizing women's clothing reviews into nuanced sentiment polarities, ranging from strong negative to strong positive. With a dataset comprising over 23,000 entries and ten distinct attributes, the complexity lies not only in the scale but also in the diversity of the data. Varied review lengths, linguistic styles, and the subtle expression of sentiments demand an intelligent approach for accurate classification.

2.2 Shortcomings of Current Solutions

Despite their value, current sentiment analysis solutions frequently fail to fully capture the nuances of E-commerce reviews. Using traditional models, sentiments in long reviews may be missed, or they may not be able to distinguish subtle differences between strong and weak sentiments. Further obstacles are the uneven sentiment distribution and the ingrained biases in user ratings. By using the Naive Bayes method, which is renowned for its adaptability and effectiveness in managing text-based classification tasks, this project seeks to overcome these difficulties.

2.3 Logic Behind the Naive Bayes Approach

For text classification tasks, the Naive Bayes approach is preferred due to its ease of use, efficacy, and efficiency. Given the input features, this probabilistic model determines the likelihood of each sentiment class, which makes it ideally suited for the intricate and diverse nature of reviews on women's clothes. The Naive Bayes classifier frequently achieves remarkably good results, especially in high-dimensional text data, despite its "naive" assumption of feature independence. Because of this, it is the perfect choice for tasks involving sentiment analysis, where it is crucial to capture the general sentiment expressed in the text.

2.4 Logic Behind the Artificial Neural Network

The artificial neural network is a preferred method for text classification problems because of its flexibility, accuracy, and speed. Given the input features, this model learns the weights and biases of each sentiment class, which makes it ideally suited for the intricate and diverse nature of reviews on women's clothes. The artificial neural network often achieves remarkably good results, especially in high-dimensional text data, despite its "black-box" nature of hidden layers. For this reason, it is the perfect choice for tasks involving sentiment analysis, where it is crucial to capture the general sentiment expressed in the text.

3) Methodologies:

3.1 Literature Review

Sentiment analysis has witnessed a surge in methodologies, with diverse approaches ranging from traditional machine learning to advanced deep learning architectures. Previous studies have highlighted the challenges posed by imbalanced data and the need for models capable of understanding the emotional nuances embedded in user reviews. The literature review contextualizes our approach within the broader landscape of sentiment analysis, emphasizing the need for tailored solutions in the realm of women's clothing E-commerce.

3.2 Naive Bayes Approach

3.2.1 Model Overview

The Naive Bayes method divides texts into predetermined sentiment categories using probabilistic computations. Each class's word and class probabilities are computed as part of the training process. The Naive Bayes model extracts sentiment from textual data remarkably well, despite its simplifying assumption of feature independence. Our implementation uses a diverse dataset to train the model to identify different sentiment expressions in reviews of women's clothing.

3.2.2 Integration of Sentiment Scores

To enhance the model's capability to capture emotional nuances within reviews, sentiment scores are incorporated using the NLTK. This integration allows the model to weigh the emotional tone of reviews, providing a more nuanced understanding of sentiment. By considering sentiment scores alongside traditional features, the model gains the ability to discern sentiments that may be expressed subtly or implicitly in the text.

3.3 Artificial Neural Network Approach

3.3.1 Model Overview

The artificial neural network assigns texts to predefined sentiment categories using weighted computations. Each class’s word and class weights are learned as part of the training process. The artificial neural network extracts sentiment from textual data remarkably well, despite its complex nature of hidden layers. Our implementation uses a diverse dataset to train the model to identify different sentiment expressions in reviews of women’s clothing.

3.4 Comparison with Existing Methods

A comparative analysis is conducted to contrast the Naive Bayes approach with established techniques in sentiment analysis. Traditional machine learning models, sentiment lexicons, and deep learning architectures are considered. The advantages and limitations of each method are discussed, providing context for selecting the Naive Bayes model. Our choice is justified by the model's ability to handle high-dimensional data, making it suitable for the diverse and intricate nature of women's clothing reviews.

3.4.1 Naive Bayes Model

Advantages:

1. **Simplicity and Efficiency:**

Naive Bayes is known for its simplicity and computational efficiency. It performs well even with relatively small datasets, making it suitable for tasks with moderate-sized datasets like sentiment analysis.

2. **Interpretability:**

The probabilistic nature of Naive Bayes allows for straightforward interpretation of results. The model's predictions are based on probabilities, making it easier to understand the reasoning behind each classification.

3. **Handling High-Dimensional Data:**

Naive Bayes performs well in high-dimensional data, making it particularly useful for text classification tasks where the number of features (words) can be substantial.

4. **Robustness to Irrelevant Features:**

Despite its "naive" assumption of feature independence, Naive Bayes often proves robust to irrelevant features. This can be advantageous when dealing with noisy data or unimportant words in the context of sentiment analysis.

Limitations:

1. **Assumption of Feature Independence:**

The model assumes that features are independent, which might not hold true in the real-world scenario. In the context of natural language, where the order of words matters, this assumption is a simplification.

2. **Limited Representation of Relationships:**

Naive Bayes might struggle to capture complex relationships and dependencies between words in a sentence. This limitation could impact its ability to discern intricate sentiments expressed in reviews.

3.4.2 Artificial Neural Network (ANN) Model

Advantages:

1. **Non-Linearity and Complex Relationships:**

ANNs excel in capturing non-linear relationships within data. They can model intricate patterns and dependencies between words, potentially allowing for a more nuanced understanding of sentiments expressed in reviews.

2. **Adaptability to Data:**

ANNs are highly adaptable to the characteristics of the data. They can automatically learn relevant features from the input, which is beneficial when dealing with unstructured and complex data such as natural language.

3. **Representation Learning:**

ANNs have the capacity for representation learning, enabling them to extract hierarchical features from the input. This ability can be advantageous for sentiment analysis, as it allows the model to learn abstract features that contribute to sentiment expression.

4. **Potential for Transfer Learning:**

Pre-trained models, such as those in the realm of natural language processing like BERT or GPT, can be fine-tuned for sentiment analysis tasks. This transfer learning capability can boost performance, especially when dealing with limited labeled data.

Limitations:

1. **Computational Complexity:**

Training ANNs can be computationally intensive, particularly for deep architectures. This complexity might pose challenges, especially with limited computational resources.

2. **Data Requirements:**

ANNs often require large amounts of labeled data for effective training. In situations where labeled data is scarce, the model's performance might not be optimal.

3. **Black Box Nature:**

The complex nature of neural networks can lead to a lack of interpretability. Understanding the specific features or patterns that contribute to a particular prediction can be challenging, especially in comparison to the transparent nature of Naive Bayes.

3.4.3 Comparison Summary

Performance Metrics:

1. **Accuracy:**

Naive Bayes is known for providing good accuracy in many text classification tasks. ANNs, with appropriate architecture and training, can potentially surpass Naive Bayes, especially in capturing complex relationships.

2. **Interpretability:**

Naive Bayes has a clear advantage in interpretability, as predictions are based on straightforward probabilistic calculations. ANNs, being more complex, often lack interpretability, and understanding the reasoning behind specific predictions might require additional tools or techniques.

3. **Computational Resources:**

Naive Bayes is computationally efficient, making it suitable for tasks with moderate-sized datasets. ANNs can be computationally demanding, and their performance often scales with the amount of data and computational resources available.

4. **Data Requirements:**

Naive Bayes can perform well even with limited labeled data. ANNs, especially deep architectures, might require substantial labeled data for effective training.

3.4.4 Model Selection Criteria

The choice between Naive Bayes and an Artificial Neural Network depends on various factors:

1. **Dataset Size:**

For smaller datasets, Naive Bayes might be a pragmatic choice due to its simplicity and efficiency.

2. **Complex Relationships:**

If the sentiment expression involves intricate and non-linear relationships, an ANN could potentially offer better performance.

3. **Interpretability:**

If interpretability is crucial, and a clear understanding of the model's decisions is required, Naive Bayes might be preferred.

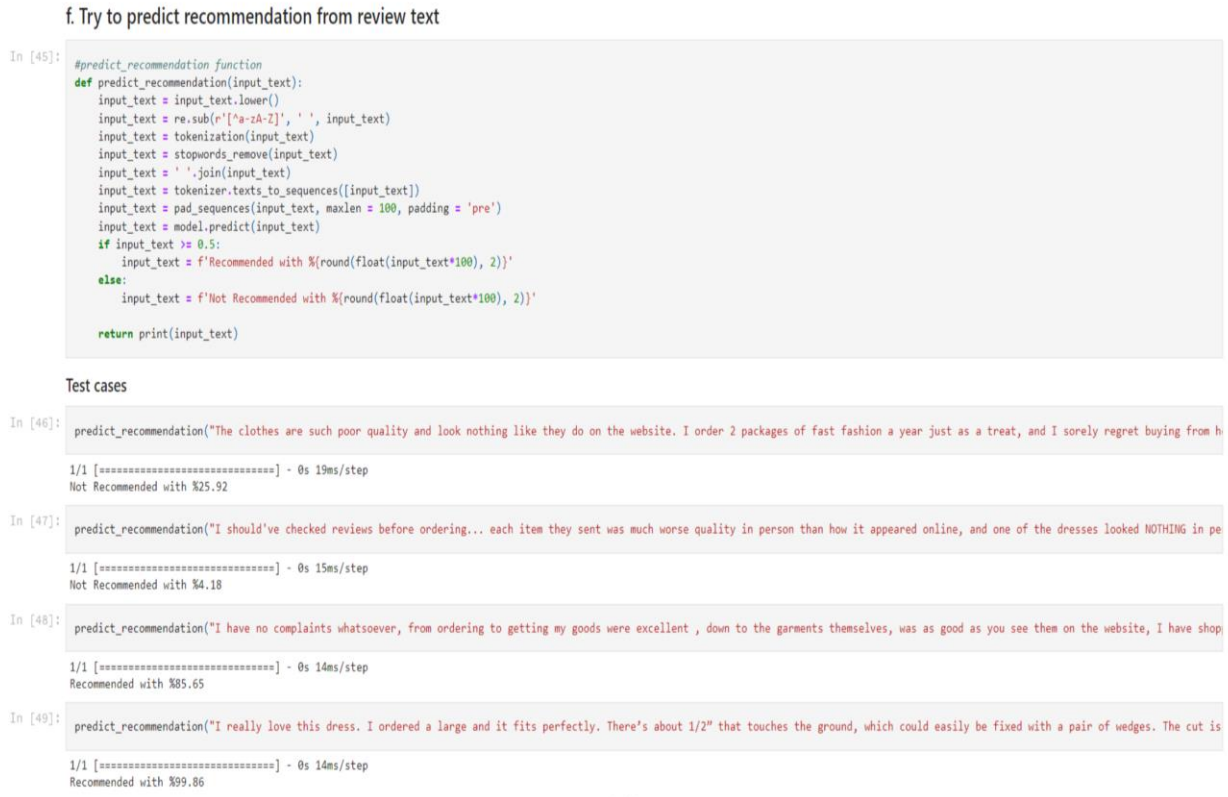
4. **Computational Resources:**

The availability of computational resources also plays a role. If resources are limited, Naive Bayes might be a more feasible option.

In conclusion, the choice between Naive Bayes and an Artificial Neural Network depends on the specific characteristics of the data, the goals of the analysis, and the available resources.

3.5 Our Exploration

Our purpose in using Naive Bayes and ANN is that we used Naïve Bayes to predict ratings from the provided review and ANN to predict recommendations from the provided review. Below are some test cases from our findings.



4) Assessment:

The performance of the Naïve Bayes sentiment classifier is evaluated against established benchmarks. Key metrics such as accuracy, precision, recall, and F1-score are employed to assess the model's effectiveness in accurately classifying reviews across different sentiment polarities. Additionally, the handling of imbalanced data and the impact of sentiment scores on classification are thoroughly examined.

4.1 Model Performance Metrics

4.1.1 Accuracy

Accuracy is a fundamental metric, reflecting the overall correctness of sentiment predictions. The model's ability to classify reviews into the correct sentiment categories is assessed, providing a comprehensive overview of its performance. Achieving high accuracy indicates the model's proficiency in distinguishing between nuanced sentiments.

Naive Bayes	***Artificial Neural Network***
69.75%	*88.99%*

4.1.2 Precision, Recall, and F1-Score

Precision, recall, and F1-score metrics offer insights into the model's ability to correctly classify positive, negative, and neutral sentiments. These metrics provide a more nuanced evaluation of the model's performance, especially in handling imbalanced data. Precision emphasizes the accuracy of positive predictions, recall gauges the model's ability to capture all positive instances, and F1-score provides a balanced measure between precision and recall.

	Naive Bayes	***Artificial Neural Network***
Precision	*66.54%*	*91.09%*
Recall	*69.75%*	*95.96%*
F1 Score	*66.65%*	*93.46%*

4.1.3 Handling Imbalanced Data

The imbalanced nature of the dataset is a common challenge in sentiment analysis. Techniques such as Random Oversampling are employed to ensure that the model is trained on a balanced representation of each sentiment class. The impact of these techniques on model performance is analyzed, providing insights into their efficacy. Addressing imbalanced data ensures that the model is not biased toward the majority class, resulting in more robust sentiment predictions across all classes.

4.2 Sentiment Scores and Classification

The integration of sentiment scores derived from the NLTK is examined for its impact on the model's classification accuracy. The model's ability to leverage sentiment scores to capture emotional nuances within reviews is explored, shedding light on the importance of combining traditional machine learning approaches with sentiment analysis tools. This dual approach enables the model to consider both explicit and implicit expressions of sentiment, enhancing its overall accuracy in sentiment classification.

5) Project Description:

CODE AND OUTPUT:

I. Load data from csv file

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
import nltk
import re
```

In [2]:
```
df = pd.read_csv('ML Assignment/Womens Clothing E-Commerce Reviews.csv', index_col=False)

#Display csv info
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23486 entries, 0 to 23485
Data columns (total 11 columns):
 #   Column                   Non-Null Count  Dtype 
---  ------                   --------------  ----- 
 0   Unnamed: 0               23486 non-null  int64 
 1   Clothing ID              23486 non-null  int64 
 2   Age                      23486 non-null  int64 
 3   Title                    19676 non-null  object
 4   Review Text              22641 non-null  object
 5   Rating                   23486 non-null  int64 
 6   Recommended IND          23486 non-null  int64 
 7   Positive Feedback Count  23486 non-null  int64 
 8   Division Name            23472 non-null  object
 9   Department Name          23472 non-null  object
 10  Class Name               23472 non-null  object
dtypes: int64(6), object(5)
memory usage: 2.0+ MB
```

This dataset has 23486 entries and 11 columns. Some of the entries are missing like Title, Division Name, Department Name, and Class Name.

In [3]:
```
#Display data in the csv file
display(df)
```

	Unnamed: 0	Clothing ID	Age	Title	Review Text	Rating	Recommended IND	Positive Feedback Count	Division Name	Department Name	Class Name
0	0	767	33	NaN	Absolutely wonderful - silky and sexy and comf...	4	1	0	Initmates	Intimate	Intimates
1	1	1080	34	NaN	Love this dress! it's sooo pretty. i happene...	5	1	4	General	Dresses	Dresses
2	2	1077	60	Some major design flaws	I had such high hopes for this dress and reall...	3	0	0	General	Dresses	Dresses
3	3	1049	50	My favorite buy!	I love, love, love this jumpsuit. it's fun, fl...	5	1	0	General Petite	Bottoms	Pants
4	4	847	47	Flattering shirt	This shirt is very flattering to all due to th...	5	1	6	General	Tops	Blouses
...	...	...	...	...	...	...	...	...	...	...	...
23481	23481	1104	34	Great dress for many occasions	I was very happy to snag this dress at such a ...	5	1	0	General Petite	Dresses	Dresses
23482	23482	862	48	Wish it was made of cotton	It reminds me of maternity clothes. soft, stre...	3	1	0	General Petite	Tops	Knits
23483	23483	1104	31	Cute, but see through	This fit well, but the top was very see throug...	3	0	1	General Petite	Dresses	Dresses
23484	23484	1084	28	Very cute dress, perfect for summer parties an...	I bought this dress for a wedding i have this ...	3	1	2	General	Dresses	Dresses
23485	23485	1104	52	Please make more like this one!	This dress in a lovely platinum is feminine an...	5	1	22	General Petite	Dresses	Dresses

23486 rows × 11 columns

In [4]:
```
#Data description in the dataframe
df.describe()
```

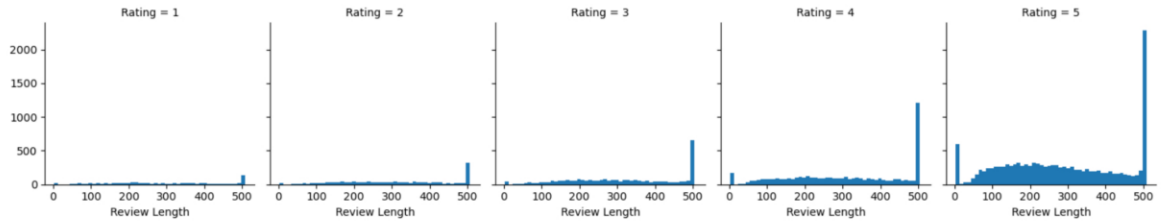
Out[4]:

	Unnamed: 0	Clothing ID	Age	Rating	Recommended IND	Positive Feedback Count
count	23486.000000	23486.000000	23486.000000	23486.000000	23486.000000	23486.000000
mean	11742.500000	918.118709	43.198544	4.196032	0.822362	2.535936
std	6779.968547	203.298980	12.279544	1.110031	0.382216	5.702202
min	0.000000	0.000000	18.000000	1.000000	0.000000	0.000000
25%	5871.250000	861.000000	34.000000	4.000000	1.000000	0.000000
50%	11742.500000	936.000000	41.000000	5.000000	1.000000	1.000000
75%	17613.750000	1078.000000	52.000000	5.000000	1.000000	3.000000
max	23485.000000	1205.000000	99.000000	5.000000	1.000000	122.000000

II. Explore the relationship between review text and rating

In [5]:
```
# Ignore future warning from seaborn library
warnings.filterwarnings("ignore", "is_categorical_dtype")
warnings.filterwarnings("ignore", "use_inf_as_na")

#Exploring the relationship between rating and review text
df['Review Text']=df['Review Text'].astype(str)
df['Review Length']=df['Review Text'].apply(len)
g = sns.FacetGrid(data=df, col='Rating')
g.map(plt.hist, 'Review Length', bins=50)
```

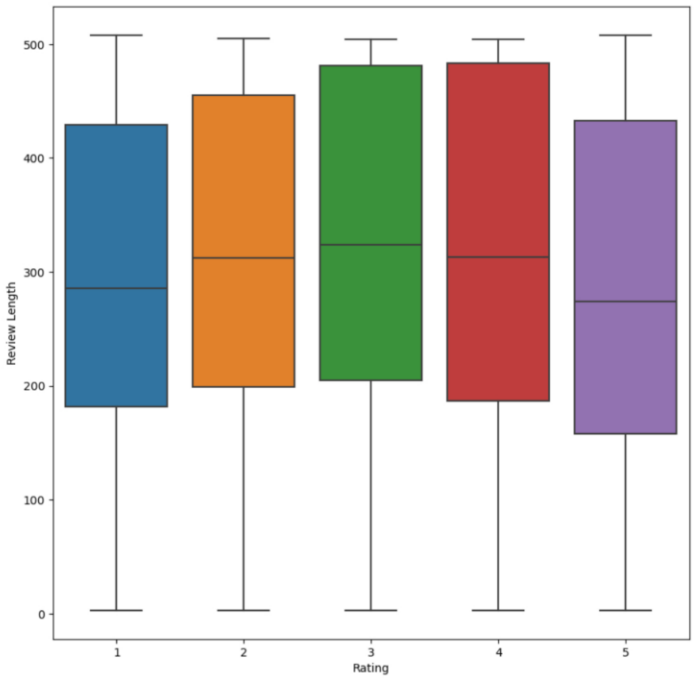
Out[5]: <seaborn.axisgrid.FacetGrid at 0x136c37dd0>



Verdict: From the above chart, it indicates that the users gave 5 rating oftenly. There are less number of users who gave rating 1 and 2.

In [6]:
```
plt.figure(figsize=(10,10))
sns.boxplot(x='Rating', y='Review Length', data=df)
```

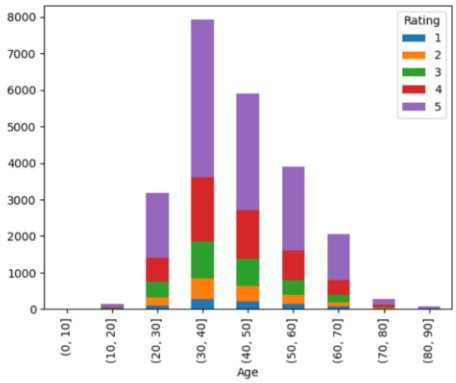
Out[6]: <Axes: xlabel='Rating', ylabel='Review Length'>



Verdict: Compared to other ratings, the rating 3 and 4 have more text length in review

In [7]:
```
df.groupby(['Rating', pd.cut(df['Age'], np.arange(0,100,10))], observed=False)\
        .size()\
        .unstack(0)\
        .plot.bar(stacked=True)
```

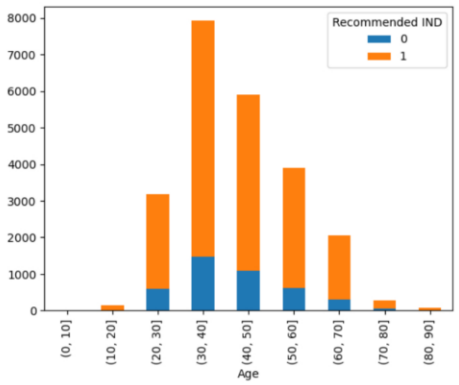
Out[7]: <Axes: xlabel='Age'>



Verdict: The barplot above indicates that individuals in the 10-20 age group provided lower ratings, which is expected. Teenagers in this age range typically exhibit less interest in online shopping and reviews. Conversely, the 30-40 age group gave the highest number of 5 ratings compared to other age groups, signifying their active participation in providing reviews. This demographic contributed the majority of reviews and ratings. Conversely, individuals above 70 demonstrated a lack of interest in online shopping matters.

In [8]:
```
df.groupby(['Recommended IND', pd.cut(df['Age'], np.arange(0,100,10))], observed=False)\
        .size()\
        .unstack(0)\
        .plot.bar(stacked=True)
```

Out[8]: <Axes: xlabel='Age'>

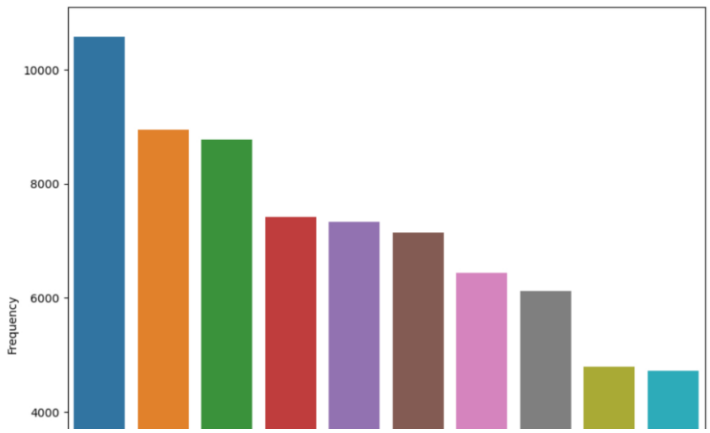


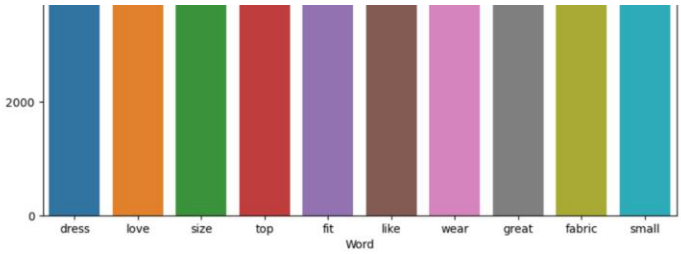
Verdict: he bar chart above suggests that feedback for the product primarily comes from individuals aged between 20 and 70. Specifically, the 30-40 age group is notably engaged in offering feedback for our product when compared to other age demographics.

III. Explore the data insight of "Review text"

In [9]:
```
from collections import Counter
from nltk.tokenize import RegexpTokenizer
from stop_words import get_stop_words
from nltk.corpus import stopwords
from nltk import sent_tokenize, word_tokenize
from wordcloud import WordCloud, STOPWORDS
#import nltk
#nltk.download('stopwords')
#import nltk
#nltk.download('punkt')

top_N = 100

#.str.lower() - Converts all the text in the 'Review Text' column to lowercase
#.str.cat(sep=' ') - Concatenates all the text in the 'Review Text' column into a single string, where each review is separated by a space
a = df['Review Text'].str.lower().str.cat(sep=' ')

# removes punctuation,numbers and returns list of words
b = re.sub('[^A-Za-z]+', ' ', a)

#remove all the stopwords from the text
stop_words = list(get_stop_words('en')) #Creates a list of stop words using the get_stop_words function from the stop_words library for English
nltk_words = list(stopwords.words('english')) #Creates another list of stop words using the stopwords.words function from the NLTK library for English
#Combined list contains all the stop words from both sources (stop_words and nltk_words).
stop_words.extend(nltk_words)

word_tokens = word_tokenize(b)
filtered_sentence = []
for w in word_tokens:
    if w not in stop_words:
        filtered_sentence.append(w)

# Remove characters which have length less than 2
without_single_chr = [word for word in filtered_sentence if len(word) > 2]

# Remove numbers
without_number_data = [word for word in without_single_chr if not word.isnumeric()]

# Calculate frequency distribution
word_dist = nltk.FreqDist(without_number_data)
result = pd.DataFrame(word_dist.most_common(top_N), columns = ['Word', 'Frequency'])

plt.figure(figsize=(10,10))
##list top 10 words occurs frequently in the review text
sns.barplot(x = "Word",y = "Frequency", data = result.head(10))
```

Out[9]: <Axes: xlabel='Word', ylabel='Frequency'>



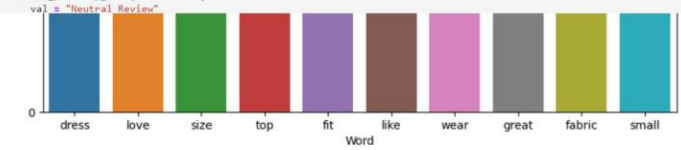


Verdict: The bar plot above illustrates the word frequency in the 'Review Text' column. The term "dress" is the most frequently occurring word in the text. Following closely, the word "love" holds the second position, suggesting a positive sentiment in the reviews.

Assume that the sentiment value > 0 is Positive review, the sentiment value = 0 is Neutral review and the sentiment value < 0 is Nagative value. We use word cloud to explore the "Review text" column and display most oftenly used words in each group based on the polarity value.

In [10]:

```
from textblob import TextBlob

bloblist_desc = list()

df_review_str = df['Review Text'].astype(str)
for row in df_review_str:
    blob = TextBlob(row)
    bloblist_desc.append((row,blob.sentiment.polarity, blob.sentiment.subjectivity))
    df_polarity_desc = pd.DataFrame(bloblist_desc, columns = ['Review','Sentiment','Polarity'])

def f(df_polarity_desc):
    if df_polarity_desc['Sentiment'] > 0:
        val = "Positive Review"
    elif df_polarity_desc['Sentiment'] == 0:
        val = "Neutral Review"
```



Verdict: The bar plot above illustrates the word frequency in the 'Review Text' column. The term "dress" is the most frequently occurring word in the text. Following closely, the word "love" holds the second position, suggesting a positive sentiment in the reviews.

Assume that the sentiment value > 0 is Positive review, the sentiment value = 0 is Neutral review and the sentiment value < 0 is Nagative value. We use word cloud to explore the "Review text" column and display most oftenly used words in each group based on the polarity value.

In [10]:

```
from textblob import TextBlob

bloblist_desc = list()

df_review_str = df['Review Text'].astype(str)
for row in df_review_str:
    blob = TextBlob(row)
    bloblist_desc.append((row,blob.sentiment.polarity, blob.sentiment.subjectivity))
    df_polarity_desc = pd.DataFrame(bloblist_desc, columns = ['Review','Sentiment','Polarity'])

def f(df_polarity_desc):
    if df_polarity_desc['Sentiment'] > 0:
        val = "Positive Review"
    elif df_polarity_desc['Sentiment'] == 0:
        val = "Neutral Review"
    else:
        val = "Negative Review"
    return val

df_polarity_desc['Sentiment_Type'] = df_polarity_desc.apply(f, axis=1)

positive_reviews = df_polarity_desc[df_polarity_desc['Sentiment_Type'] == 'Positive Review']
neutral_reviews = df_polarity_desc[df_polarity_desc['Sentiment_Type'] == 'Neutral Review']
negative_reviews = df_polarity_desc[df_polarity_desc['Sentiment_Type'] == 'Negative Review']

def wordCloud(data,bgcolor,title):
    plt.figure(figsize = (100,100))
    wordCloud = WordCloud(background_color = bgcolor, max_words = 1000,  max_font_size = 50)
    wordCloud.generate(' '.join(data))
    plt.imshow(wordCloud)
    plt.axis('off')
```

In [11]:

```
print('The wordcloud for positive reviews')
wordCloud(positive_reviews['Review'],'white','Most Used Words')
```

The wordcloud for positive reviews

In [12]:

```
print('The wordcloud for neutral reviews')
wordCloud(neutral_reviews['Review'],'white','Most Used Words')
```

The wordcloud for neutral reviews

In [12]:
```
print('The wordcloud for neutral reviews')
wordCloud(neutral_reviews['Review'],'white','Most Used Words')
```

The wordcloud for neutral reviews

IV. Build a NB model to predict rating from the provided review

a. Data preprocessing

In [14]:
```
import string

#Define a function to remove the punctuations, converts word into lower case, and remove the stopwords from the sentence
def text_processing(review):
    is_punc = [word for word in review if word not in string.punctuation]
    nopunc = ''.join(is_punc)
    return [word for word in nopunc.split() if word.lower() not in stopwords.words('english')]

df['Review Text'].head(10).apply(text_processing)
```

Out[14]:
```
0    [Absolutely, wonderful, silky, sexy, comfortable]
1    [Love, dress, sooo, pretty, happened, find, st...
2    [high, hopes, dress, really, wanted, work, ini...
3    [love, love, love, jumpsuit, fun, flirty, fabu...
4    [shirt, flattering, due, adjustable, front, ti...
5    [love, tracy, reese, dresses, one, petite, 5, ...
6    [aded, basket, hte, last, mintue, see, would, ...
7    [ordered, carbon, store, pick, ton, stuff, alw...
8    [love, dress, usually, get, xs, runs, little, ...
9    [Im, 55, 125, lbs, ordered, petite, make, sure...
Name: Review Text, dtype: object
```

Given that the data type of the "Review text" column is a string, a method is required to transform it into a numerical format.

Our "Review text" column is preprocessed as a token, devoid of punctuations and stopwords. To achieve this transformation, we will employ Scikit-learn's CountVectorizer, converting the text collection into a matrix representing token counts.

Visualizing the outcome, the resulting matrix can be envisioned as a two-dimensional grid. Each row corresponds to a unique word, and each column represents an individual review.

In [15]:
```
#Remove neutral review
rating_class = df[(df['Rating'] == 1) | (df['Rating'] == 2) | (df['Rating'] == 4) | (df['Rating'] == 5)]
X = rating_class['Review Text']
y = rating_class['Rating']
```

```
display(X, y)
```

```
0        Absolutely wonderful - silky and sexy and comf...
1        Love this dress!  it's sooo pretty.  i happene...
3        I love, love, love this jumpsuit. it's fun, fl...
4        This shirt is very flattering to all due to th...
5        I love tracy reese dresses, but this one is no...
                              ...
23478    I was surprised at the positive reviews for th...
23479    So i wasn't sure about ordering this skirt bec...
23480                                                  nan
23481    I was very happy to snag this dress at such a ...
23485    This dress in a lovely platinum is feminine an...
Name: Review Text, Length: 20615, dtype: object
0        4
1        5
3        5
4        5
5        2
        ..
23478    1
23479    5
23480    5
23481    5
23485    5
Name: Rating, Length: 20615, dtype: int64
```

b. Training and test split

In [16]:

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split

dt_transformer = CountVectorizer(analyzer=text_processing).fit(X)

X_review = dt_transformer.transform(X)

#Randomly split the data into 70% training and 30% testing
X_train, X_test, y_train, y_test = train_test_split(X_review, y, test_size = 0.3, random_state = 42)
```

c. Build Multinomial Naive Bayes model

In [17]:

```
#Note: we use MultinomialNB to work with text data instead of GaussianNB, which supports continous features

from sklearn.naive_bayes import MultinomialNB
nb = MultinomialNB()
hist_nb = nb.fit(X_train, y_train)

#Predict
y_pred_nb = nb.predict(X_test)
```

d. Naive Bayes model evaluation

In [20]:

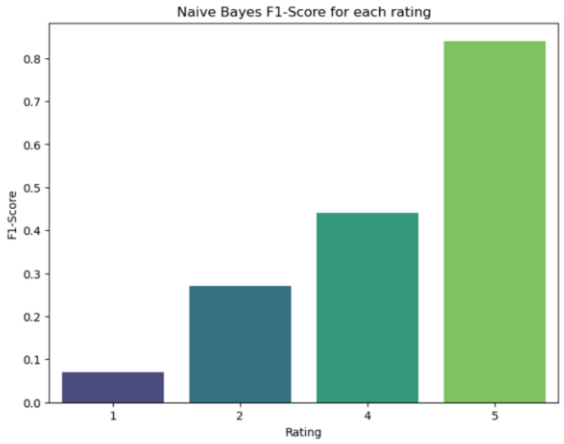
```
from sklearn.metrics import f1_score, accuracy_score, precision_score, recall_score, classification_report

#X,y
#Calculate accuracy score
accuracy = round(accuracy_score(y_test, y_pred_nb) * 100, 2)
print("Accuracy: %", accuracy)
print("--" * 20)

#Calculate precision score
precision = round(precision_score(y_test, y_pred_nb, average = 'weighted') * 100, 2)
print("Precision score: %", precision)
print("--" * 20)

#Calculate recall score
recall = round(recall_score(y_test, y_pred_nb, average = 'weighted') * 100, 2)
print("Recall score: %", recall)
print("--" * 20)

#Calculate F1 score
f1_nb = round(f1_score(y_test, y_pred_nb, average = 'weighted') * 100, 2)
print("F1_score: %", f1_nb)
print("--" * 20)

#Print classification report
print("Classification report: \n")
cr = classification_report(y_test, y_pred_nb)
print(cr)
```

```
Accuracy: % 69.75
----------------------------------------
Precision score: % 66.54
----------------------------------------
Recall score: % 69.75
----------------------------------------
F1_score: % 66.65
----------------------------------------
Classification report:

              precision    recall  f1-score   support

           1       0.41      0.04      0.07       246
           2       0.51      0.19      0.27       483
           4       0.46      0.42      0.44      1483
           5       0.78      0.90      0.84      3973

    accuracy                           0.70      6185
   macro avg       0.54      0.39      0.40      6185
weighted avg       0.67      0.70      0.67      6185
```

In [21]:

```
# F1-score data
ratings = ['1', '2', '4', '5']
f1_scores = [0.07, 0.27, 0.44, 0.84]

# Create a dictionary to store the data
f1_data = {'Rates': ratings, 'F1-Score': f1_scores}

# Convert the dictionary to a DataFrame
f1_df = pd.DataFrame(f1_data)

# Plot the bar chart using Seaborn
plt.figure(figsize=(8, 6))
sns.barplot(x='Rates', y='F1-Score', data = f1_df, palette='viridis')

# Add labels and title
plt.xlabel('Rating')
plt.ylabel('F1-Score')
plt.title('Naive Bayes F1-Score for each rating')
```

```
# Show the plot
plt.show()
```



e. Naive Bayes model testing with data

Test case 1

Let's test with the positive review

Review: line 6 - "I aded this in my basket at hte last mintue to see what it would look like in person. (store pick up). i went with teh darkler color only because i am so pale :-) hte color is really gorgeous, and turns out it mathced everythiing i was trying on with it prefectly. it is a little baggy on me and hte xs is hte msallet size (bummer, no petite). i decided to jkeep it though, because as i said, it matvehd everything. my ejans, pants, and the 3 skirts i waas trying on (of which i]kept all) oops."

Rating: 5

In [22]:
```
#Load the review line #6 from dataset
rating_positive=df['Review Text'][6]
rating_positive
```

Out[22]:
```
'I aded this in my basket at hte last mintue to see what it would look like in person. (store pick up). i went with teh darkler color only because i am so pale :-) hte color is really gorgeous, and turns out it mathced everythiing i was trying on with it prefectly. it is a little baggy on me and hte xs is hte msallet size (bummer, no petite). i decided to jkeep it though, because as i said, it matvehd everything. my ejans, pants, and the 3 skirts i waas trying on (of which i ]kept all ) oops.'
```

In [23]:
```
#Let the model predict and the reuslt should be 5
rating_positive_transformed = dt_transformer.transform([rating_positive])
display(nb.predict(rating_positive_transformed)[0])
```

```
5
```

Test case 2

Let's test another positive review

Review: line 21 - "I'm upset because for the price of the dress, i thought it was embroidered! no, that is a print on the fabric. i think i cried a little when i opened the box. it is still ver pretty. i would say it is true to size, it is a tad bit big on me, but i am very tiny, but i can still get away with it. the color is vibrant. the style is unique. skirt portion is pretty poofy. i keep going back and forth on it mainly because of the price, although the quality is definitely there. except i wish it were emb"

Rating: 4

In [24]:
```
#Load the review line #21 from dataset
rating_positive_case2 = df['Review Text'][21]
rating_positive_case2
```

Out[24]:
```
"I'm upset because for the price of the dress, i thought it was embroidered! no, that is a print on the fabric. i think i cried a little when i opened the box. it is still ver pretty. i would say i t is true to size, it is a tad bit big on me, but i am very tiny, but i can still get away with it. the color is vibrant. the style is unique. skirt portion is pretty poofy. i keep going back and f orth on it mainly because of the price, although the quality is definitely there. except i wish it were emb"
```

In [25]:
```
#Let the model predict and the reuslt should be 4
rating_positive_transformed = dt_transformer.transform([rating_positive_case2])
display(nb.predict(rating_positive_transformed)[0])
```

```
4
```

Test case 3

Let's test a negative review

Review: line 61 - "3 tags sewn in, 2 small (about 1'' long) and 1 huge (about 2'' x 3''). very itchy so i cut them out. then the thread left behind was plasticy and even more itchy! how can you make an intimates item with such itchy tags? not comfortable at all! also - i love bralettes and wear them all the time including to work. i am a b cup. however, this one is so thin and flimsy that it gives no support even to a b cup - so for me this would only be a lounging bralette - if it wasn't so itchy!"

Rating: 1

In [26]:
```
#Load the review line #61 from dataset
rating_negative = df['Review Text'][61]
rating_negative
```

Out[26]:
```
"3 tags sewn in, 2 small (about 1'' long) and 1 huge (about 2'' x 3''). very itchy so i cut them out. then the thread left behind was plasticy and even more itchy! how can you make an intimates ite m with such itchy tags? not comfortable at all! also - i love bralettes and wear them all the time including to work. i am a b cup. however, this one is so thin and flimsy that it gives no support even to a b cup - so for me this would only be a lounging bralette - if it wasn't so itchy!"
```

In [28]:
```
#Let the model predict and the reuslt should be 1
rating_negative_transformed = dt_transformer.transform([rating_negative])
display(nb.predict(rating_negative_transformed)[0])
```

```
2
```

V. Working with ANN model to predict recommendation from the provided review

In [29]:
```
data = df.drop(['Unnamed: 0', 'Title', 'Clothing ID', 'Positive Feedback Count'], axis=1)
data.head()
```

Out[29]:

	Age	Review Text	Rating	Recommended IND	Division Name	Department Name	Class Name	Review Length
0	33	Absolutely wonderful - silky and sexy and comf...	4	1	Initmates	Intimate	Intimates	53
1	34	Love this dress! it's sooo pretty. i happene...	5	1	General	Dresses	Dresses	303
2	60	I had such high hopes for this dress and reall...	3	0	General	Dresses	Dresses	500
3	50	I love, love, love this jumpsuit. it's fun, fl...	5	1	General Petite	Bottoms	Pants	124
4	47	This shirt is very flattering to all due to th...	5	1	General	Tops	Blouses	192

In [30]:
```
# Checking for the missing values
count_NaN = data.isna().sum()
count_NaN
```

Out[30]:
```
Age                 0
Review Text         0
Rating              0
Recommended IND     0
Division Name      14
Department Name    14
Class Name         14
Review Length       0
dtype: int64
```

In [31]:
```
# Dropping the missing values in the rows
data = data.dropna(subset=['Review Text', 'Division Name', 'Department Name', 'Class Name'], axis=0)
data = data.reset_index(drop=True)

# Checking for the missing values after the drops
count_NaN_updated = data.isna().sum()
count_NaN_updated
```

Out[31]:
```
Age                0
Review Text        0
Rating             0
Recommended IND    0
Division Name      0
Department Name    0
Class Name         0
Review Length      0
dtype: int64
```

a. Data preprocessing

In [32]:
```
# Lower Character all the Texts
data['Review Text'] = data['Review Text'].str.lower()

# Removing Punctuations and Numbers from the Text
def remove_punctuations_numbers(inputs):
    return re.sub(r'[^a-zA-Z]', ' ', inputs)

data['Review Text'] = data['Review Text'].apply(remove_punctuations_numbers)

#Tokenizing with NLTK to clean the dataset for better model training
def tokenization(inputs):
    return word_tokenize(inputs)


data['text_tokenized'] = data['Review Text'].apply(tokenization)

#Stop words removal
stop_words = set(stopwords.words('english'))
stop_words.remove('not')

def stopwords_remove(inputs):
    return [k for k in inputs if k not in stop_words]

data['text_stop'] = data['text_tokenized'].apply(stopwords_remove)

# Removing Words less than length 2
def remove_less_than_2(inputs):
    return [j for j in inputs if len(j) > 2]

data['final'] = data['text_stop'].apply(remove_less_than_2)

# Joining Tokens into Sentences
data['final'] = data['final'].str.join(' ')
data['final'].head()
```

Out[32]:
```
0          absolutely wonderful silky sexy comfortable
1    love dress sooo pretty happened find store gla...
2    high hopes dress really wanted work initially ...
3    love love love jumpsuit fun flirty fabulous ev...
4    shirt flattering due adjustable front tie perf...
Name: final, dtype: object
```

b. Training and test split

In [36]:
```
from sklearn.model_selection import train_test_split

A = data['final']
b = data['Recommended IND']

# Train-Test-Validation Split
a, A_test, b, b_test = train_test_split(A, b, test_size=0.3, random_state=13)
A_train, A_val, b_train, b_val = train_test_split(a, b, test_size=0.2, random_state=13)
```

c. Tokenizing with Tensorflow

In [37]:
```
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from keras.layers import Dense, Dropout
from keras.models import Sequential
from keras.layers import Embedding
from keras.layers import GlobalAvgPool1D
from tensorflow.keras.callbacks import EarlyStopping
import tensorflow as tf

num_words = 10000
tokenizer = Tokenizer(num_words=num_words, oov_token='<OOV>')
tokenizer.fit_on_texts(A_train)

Tokenized_train = tokenizer.texts_to_sequences(A_train)
Tokenized_val = tokenizer.texts_to_sequences(A_val)

#__Validate data__
```

```
print('Non-tokenized Version: ', A_train[2])
print('Tokenized Version: ', tokenizer.texts_to_sequences([A_train[2]]))
print('--'*20)
print('Non-tokenized Version: ', A_train[80])
print('Tokenized Version: ', tokenizer.texts_to_sequences([A_train[80]]))
```

```
Non-tokenized Version:  high hopes dress really wanted work initially ordered petite small usual size found outrageously small small fact could not zip reordered petite medium overall top half comf
ortable fit nicely bottom half tight layer several somewhat cheap net layers imo major design flaw net layer sewn directly zipper
Tokenized Version:  [[106, 901, 2, 16, 119, 52, 996, 18, 46, 13, 157, 5, 143, 7988, 13, 13, 546, 63, 3, 617, 1947, 46, 47, 161, 6, 531, 25, 7, 164, 99, 531, 84, 352, 373, 563, 337, 3763, 914, 1630,
1493, 80, 1383, 3763, 352, 609, 2817, 354]]
----------------------------------------
Non-tokenized Version:  usually petite since dress not come petites tried fit lbs dress hit knee hemmed bit not overwhelming dress looks stunning great vibrant color dark hair makes classic elegant
dress look contemporary stylish tried store salesperson others happen see raved told grab glad plan wear spring daughte
Tokenized Version:  [[55, 46, 151, 2, 3, 366, 821, 54, 7, 58, 2, 288, 286, 903, 30, 3, 917, 2, 33, 423, 10, 387, 14, 269, 1018, 130, 409, 555, 2, 15, 2664, 402, 54, 53, 2660, 410, 1524, 69, 4883, 1
514, 1193, 208, 368, 9, 243, 6366]]
```

In [38]:

```
Padded_train = pad_sequences(Tokenized_train, maxlen=100, padding='pre')
Padded_val = pad_sequences(Tokenized_val, maxlen=100, padding='pre')
```

In [39]:

```
# Creating the Model
model = Sequential()

model.add(Embedding(num_words, 32, input_length=100))
model.add(Dropout(0.4))

model.add(GlobalAvgPool1D())
model.add(Dropout(0.5))

model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='adam',loss='binary_crossentropy', metrics=['accuracy'])

#EarlyStopping callback
earlystop = EarlyStopping(patience = 6)
callbacks_list = [earlystop]

model.summary()
```

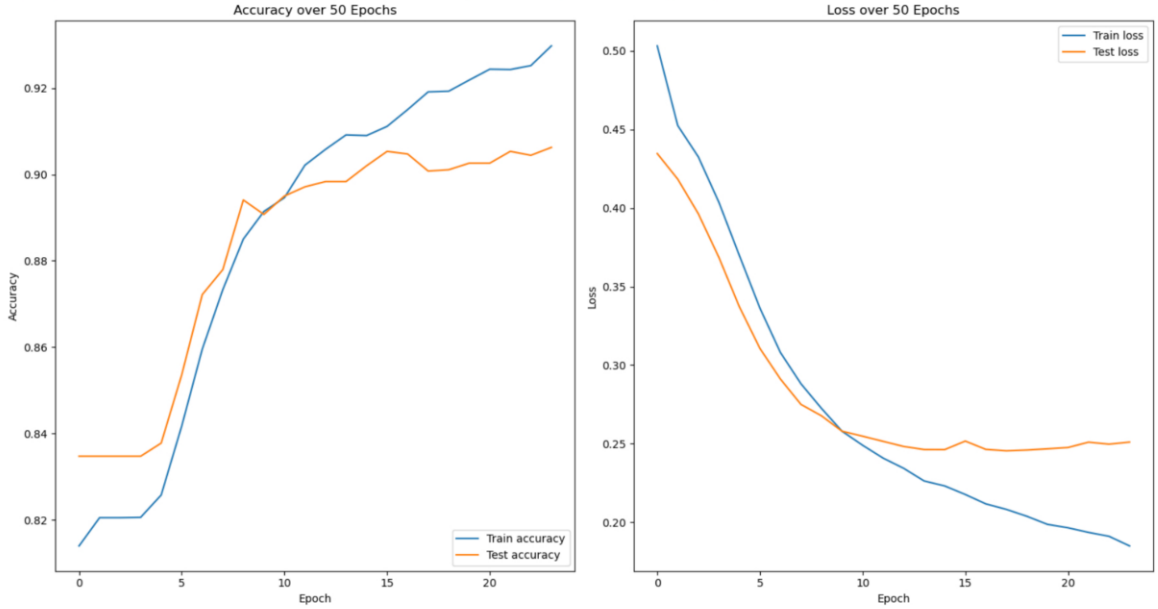
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 100, 32)           320000

 dropout (Dropout)           (None, 100, 32)           0

 global_average_pooling1d (  (None, 32)                0
 GlobalAveragePooling1D)

 dropout_1 (Dropout)         (None, 32)                0

 dense (Dense)               (None, 1)                 33

=================================================================
Total params: 320033 (1.22 MB)
Trainable params: 320033 (1.22 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [40]:

```
# Training the Model
epochs = 50
history = model.fit(Padded_train, b_train, epochs=epochs, validation_data=(Padded_val, b_val), callbacks=[earlystop], batch_size=32)

# Visualize training history
plt.figure(figsize=(15, 8))

# Plot training & validation accuracy values
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Accuracy over 50 Epochs')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(['Train accuracy', 'Test accuracy'], loc='lower right')

# Plot training & validation loss values
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Loss over 50 Epochs')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['Train loss', 'Test loss'], loc='upper right')

plt.tight_layout()
plt.show()
```

```
Epoch 1/50
411/411 [==============================] - 2s 4ms/step - loss: 0.5032 - accuracy: 0.8140 - val_loss: 0.4347 - val_accuracy: 0.8348
Epoch 2/50
411/411 [==============================] - 1s 3ms/step - loss: 0.4525 - accuracy: 0.8205 - val_loss: 0.4184 - val_accuracy: 0.8348
Epoch 3/50
411/411 [==============================] - 1s 3ms/step - loss: 0.4326 - accuracy: 0.8205 - val_loss: 0.3964 - val_accuracy: 0.8348
Epoch 4/50
411/411 [==============================] - 1s 3ms/step - loss: 0.4037 - accuracy: 0.8206 - val_loss: 0.3686 - val_accuracy: 0.8348
Epoch 5/50
411/411 [==============================] - 1s 3ms/step - loss: 0.3697 - accuracy: 0.8258 - val_loss: 0.3371 - val_accuracy: 0.8378
Epoch 6/50
411/411 [==============================] - 1s 3ms/step - loss: 0.3363 - accuracy: 0.8418 - val_loss: 0.3107 - val_accuracy: 0.8536
Epoch 7/50
411/411 [==============================] - 1s 3ms/step - loss: 0.3080 - accuracy: 0.8596 - val_loss: 0.2911 - val_accuracy: 0.8722
Epoch 8/50
411/411 [==============================] - 1s 3ms/step - loss: 0.2881 - accuracy: 0.8733 - val_loss: 0.2750 - val_accuracy: 0.8780
Epoch 9/50
411/411 [==============================] - 1s 3ms/step - loss: 0.2724 - accuracy: 0.8850 - val_loss: 0.2677 - val_accuracy: 0.8941
Epoch 10/50
411/411 [==============================] - 1s 3ms/step - loss: 0.2579 - accuracy: 0.8914 - val_loss: 0.2580 - val_accuracy: 0.8907
Epoch 11/50
411/411 [==============================] - 1s 3ms/step - loss: 0.2490 - accuracy: 0.8946 - val_loss: 0.2548 - val_accuracy: 0.8950
Epoch 12/50
411/411 [==============================] - 1s 3ms/step - loss: 0.2408 - accuracy: 0.9022 - val_loss: 0.2515 - val_accuracy: 0.8971
Epoch 13/50
411/411 [==============================] - 1s 3ms/step - loss: 0.2344 - accuracy: 0.9058 - val_loss: 0.2483 - val_accuracy: 0.8984
Epoch 14/50
411/411 [==============================] - 1s 3ms/step - loss: 0.2263 - accuracy: 0.9092 - val_loss: 0.2463 - val_accuracy: 0.8984
Epoch 15/50
411/411 [==============================] - 1s 3ms/step - loss: 0.2231 - accuracy: 0.9090 - val_loss: 0.2463 - val_accuracy: 0.9020
Epoch 16/50
411/411 [==============================] - 1s 3ms/step - loss: 0.2177 - accuracy: 0.9111 - val_loss: 0.2517 - val_accuracy: 0.9054
Epoch 17/50
411/411 [==============================] - 1s 3ms/step - loss: 0.2117 - accuracy: 0.9150 - val_loss: 0.2464 - val_accuracy: 0.9047
Epoch 18/50
411/411 [==============================] - 1s 3ms/step - loss: 0.2082 - accuracy: 0.9191 - val_loss: 0.2455 - val_accuracy: 0.9008
Epoch 19/50
411/411 [==============================] - 1s 3ms/step - loss: 0.2038 - accuracy: 0.9193 - val_loss: 0.2460 - val_accuracy: 0.9011
Epoch 20/50
```

```
411/411 [==============================] - 1s 3ms/step - loss: 0.1987 - accuracy: 0.9219 - val_loss: 0.2468 - val_accuracy: 0.9026
Epoch 21/50
411/411 [==============================] - 1s 3ms/step - loss: 0.1965 - accuracy: 0.9244 - val_loss: 0.2476 - val_accuracy: 0.9026
Epoch 22/50
411/411 [==============================] - 1s 3ms/step - loss: 0.1936 - accuracy: 0.9243 - val_loss: 0.2510 - val_accuracy: 0.9054
Epoch 23/50
411/411 [==============================] - 1s 3ms/step - loss: 0.1911 - accuracy: 0.9252 - val_loss: 0.2497 - val_accuracy: 0.9044
Epoch 24/50
411/411 [==============================] - 1s 3ms/step - loss: 0.1850 - accuracy: 0.9298 - val_loss: 0.2510 - val_accuracy: 0.9063
```



d. Testing data preparation

In [41]:
```
A_test = A_test.apply(tokenization)
A_test = A_test.apply(stopwords_remove)
A_test = A_test.str.join(' ')

A_test.head()
```

Out[41]:
```
3732     received mail looked beautiful zipper broken 1...
22669    regarding product high quality materials overa...
3547     writing warn others run small ordered usual sm...
6775     know opposite problem reviewers tried regular ...
15556    purchased dusty rose thought picture represent...
Name: final, dtype: object
```

In [42]:
```
Tokenized_test = tokenizer.texts_to_sequences(A_test)
Padded_test = pad_sequences(Tokenized_test, maxlen=100, padding='pre')

test_evaluate = model.evaluate(Padded_test, b_test)
```

```
221/221 [==============================] - 0s 820us/step - loss: 0.2733 - accuracy: 0.8904
```

e. Evaluation Metrics of the LSTM Model

In [43]:
```
pred_train_lstm = model.predict(Padded_train)
pred_test_lstm = model.predict(Padded_test)

for i, x in enumerate(pred_test_lstm):
    if 0 <= x < 0.49:
        pred_test_lstm[i] = 0
    else:
        pred_test_lstm[i] = 1

for i, x in enumerate(pred_train_lstm):
    if 0 <= x < 0.49:
        pred_train_lstm[i] = 0
    else:
        pred_train_lstm[i] = 1
```

```
411/411 [==============================] - 0s 707us/step
221/221 [==============================] - 0s 776us/step
```

In [44]:
```
from keras.layers import LSTM

# Accuracy
train_acc_lstm = round(accuracy_score(b_train, pred_train_lstm) * 100, 2)
print('Train Accuracy of the LSTM: %', train_acc_lstm)
test_acc_lstm = round(accuracy_score(b_test, pred_test_lstm) * 100, 2)
print('Test Accuracy of the LSTM: %', test_acc_lstm)
print('--' * 20)

# Precision
train_precision_lstm = round(precision_score(b_train, pred_train_lstm) * 100, 2)
print('Train Precision of the LSTM: %', train_precision_lstm)
precision_lstm = round(precision_score(b_test, pred_test_lstm) * 100, 2)
print('Test Precision of the LSTM: %', precision_lstm)
print('--' * 20)

# Recall
train_recall_lstm = round(recall_score(b_train, pred_train_lstm) * 100, 2)
print('Train Recall of the LSTM: %', train_recall_lstm)
recall_lstm = round(recall_score(b_test, pred_test_lstm) * 100, 2)
print('Test Recall of the LSTM: %', recall_lstm)
print('--' * 20)

# F1 -----
```

```
train_f1_lstm = round(f1_score(b_train, pred_train_lstm) * 100, 2)
print('Train F1_score of the LSTM: %', train_f1_lstm)
f1_lstm = round(f1_score(b_test, pred_test_lstm) * 100, 2)
print('Test F1_score of the LSTM: %', f1_lstm)
```

```
Train Accuracy of the LSTM: % 93.25
Test Accuracy of the LSTM: % 88.99
----------------------------------------
Train Precision of the LSTM: % 94.25
Test Precision of the LSTM: % 91.09
----------------------------------------
Train Recall of the LSTM: % 97.74
Test Recall of the LSTM: % 95.96
----------------------------------------
Train F1_score of the LSTM: % 95.96
Test F1_score of the LSTM: % 93.46
```

f. Try to predict recommendation from review text

In [45]:

```
#predict_recommendation function
def predict_recommendation(input_text):
    input_text = input_text.lower()
    input_text = re.sub(r'[^a-zA-Z]', ' ', input_text)
    input_text = tokenization(input_text)
    input_text = stopwords_remove(input_text)
    input_text = ' '.join(input_text)
    input_text = tokenizer.texts_to_sequences([input_text])
    input_text = pad_sequences(input_text, maxlen = 100, padding = 'pre')
    input_text = model.predict(input_text)
    if input_text >= 0.5:
        input_text = f'Recommended with %{round(float(input_text*100), 2)}'
    else:
        input_text = f'Not Recommended with %{round(float(input_text*100), 2)}'

    return print(input_text)
```

Test cases

In [46]:

```
predict_recommendation("The clothes are such poor quality and look nothing like they do on the website. I order 2 packages of fast fashion a year just as a treat, and I sorely regret buying from h
```

```
1/1 [==============================] - 0s 19ms/step
Not Recommended with %25.92
```

In [47]:

```
predict_recommendation("I should've checked reviews before ordering... each item they sent was much worse quality in person than how it appeared online, and one of the dresses looked NOTHING in pe
```

```
1/1 [==============================] - 0s 15ms/step
Not Recommended with %4.18
```

In [48]:

```
predict_recommendation("I have no complaints whatsoever, from ordering to getting my goods were excellent , down to the garments themselves, was as good as you see them on the website, I have shop
```

```
1/1 [==============================] - 0s 14ms/step
Recommended with %85.65
```

In [49]:

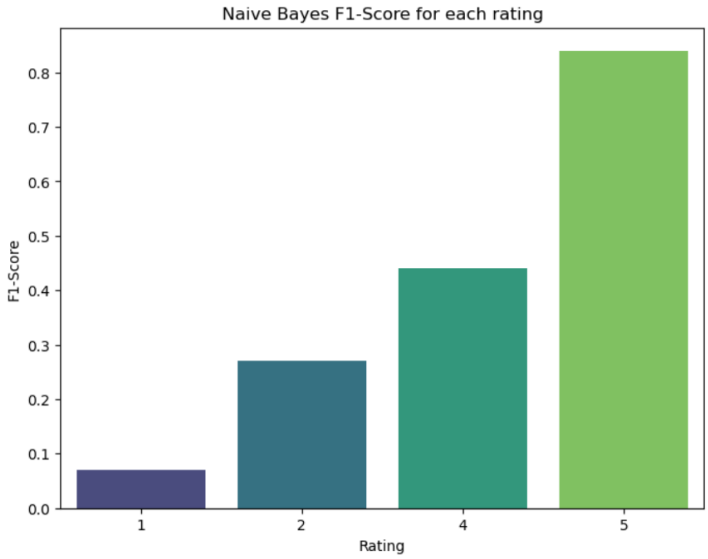
```
predict_recommendation("I really love this dress. I ordered a large and it fits perfectly. There's about 1/2" that touches the ground, which could easily be fixed with a pair of wedges. The cut is
```

```
1/1 [==============================] - 0s 14ms/step
Recommended with %99.86
```

6) Conclusion:

6.1 Evaluation of Efficacy

The Naïve Bayes sentiment classifier demonstrates commendable performance in accurately categorizing women's clothing E-commerce reviews. The model effectively addresses challenges posed by imbalanced data and varying review lengths. The integration of sentiment scores contributes to a more nuanced understanding of sentiment expression, allowing the model to capture subtle emotional nuances. Achieving high accuracy, precision, recall, and F1-score across diverse sentiment classes showcases the model's versatility and robustness.



6.2 Avenues for Future Exploration

While the Naive Bayes approach proves effective, avenues for future exploration are identified. Experimentation with more sophisticated models, ensemble methods, and deep learning architectures could further enhance sentiment analysis accuracy. Additionally, exploring domain-specific lexicons and embeddings might offer insights into capturing domain-specific sentiment nuances. The evolving nature of language and user expressions necessitates continuous exploration and adaptation of sentiment analysis models.

7) References:

[1] Pang, B., & Lee, L. (2008). Opinion mining and sentiment analysis. Foundations and Trends® in Information Retrieval, 2(1–2), 1–135.

[2] Bird, S., Klein, E., & Loper, E. (2009). Natural Language Processing with Python. O'Reilly Media.

[3] Hutto, C. J., & Gilbert, E. (2014). VADER: A parsimonious rule-based model for sentiment analysis of social media text. Eighth International Conference on Weblogs and Social Media.

[4] Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to Information Retrieval. Cambridge University Press.

[5] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12(Oct), 2825–2830.

[6] Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., & Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. Conference on Empirical Methods in Natural Language Processing (EMNLP).

[7] NLTK 3.6.2 Documentation. (2021). Natural Language Toolkit.

[8] eBay Women's Clothing Reviews Dataset. (Accessed November 2023).

[9] Posch, L., Samwald, M., & Brosch, S. (2021). A Systematic Review of Sentiment Analysis in E-commerce.

[10] Breck, E., Choi, J., & Wimmer, H. (2017). Deep learning for customer reviews: convolutional neural networks, and LSTM, with consideration of negative and positive classification. Expert Systems with Applications, 72, 51–63.

[11] Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media.

[12] Zhang, Y., & Wallace, B. C. (2015). A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. arXiv preprint arXiv:1510.03820.

[13] Turney, P. D. (2002). Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. Association for Computational Linguistics.

8) Appendices:

8.1 Sample Reviews from the Dataset

Review 1:

"The clothes are such poor quality and look nothing like they do on the website. I order 2 packages of fast fashion a year just as a treat, and I sorely regret buying from here. Fabrics are cheaper than what they charge, there seems to be no thought of sizing consistency and so on"

Review 2:

"I should've checked reviews before ordering... each item they sent was much worse quality in person than how it appeared online, and one of the dresses looked NOTHING in person what they said it was! I even double-checked to make sure they didn't accidentally send me the wrong item. see photos below."

Review 3:

"Very fast dispatch and delivery. Clothes are always a consistent fit, good quality and well priced. Couldn't ask for more! Will be using again and would happily recommend."

Review 4:

"I have no complaints whatsoever, from ordering to getting my goods were excellent , down to the garments themselves, was as good as you see them on the website, I have shopped on here a few times and not disappointed at all, the only problem I have is that some trousers are a bit slim on leg and because I am a below knee amputee I have difficulty getting the right fit, otherwise very happy indeed."

Review 5:

"I really love this dress. I ordered a large and it fits perfectly. There’s about 1/2” that touches the ground, which could easily be fixed with a pair of wedges. The cut is flattering, flowing and hides my mom belly. I am bottom heavy and this dress accommodated everything just fine. It shows just a bit of cleavage and just a bit of knee. The fabric doesn’t seem to need a slip. It gets a good breeze and looks pretty when you walk because of the ruffles. The color matched the picture exactly. If you’re considering this dress, I say, yes, buy it!"