Product Assessment Using NLP and Unsupervised Learning

BY

SAMARTH SINHA – 19BCE1670
KUNAL SUDHIR MISHRA – 19BCE1447
VANSH KOSHTI - 19BCE1747

A project report submitted to

Dr. BHARADWAJA KUMAR

SCHOOL OF COMPUTING SCIENCES AND ENGINEERING

In partial fulfillment of the requirements for the course of CSE4022 – Natural Language and Processing

VIT CHENNAI KELLAMBAKKAM – VANDULUR MAIN ROAD-632014
DECEMBER 2021

<u>BONAFIDE CERTIFICATE</u>

This is to Certified that this project report entitled "Product Assessment using NLP and Unsupervised Learning" is a bonafide work of **Kunal Sudhir Mishra(19BCE1447) Samarth Sinha (19BCE1670)** and **Vansh koshti(19BCE1747)** who carried out the J-component under my supervision and guidance.

Dr. BHARADWAJA KUMAR

Professor

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING (SCSE)

VIT UNIVERSITY, CHENNAI

CAMPUS CHENNAI-600127

ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Dr. Bharadwaja Kumar,** Professor, SCSE, for his consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

We are extremely grateful to the Dean of the SCOPE, VIT Chennai, for extending the facilities of the School towards our project and for the unstinting support. We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the course.

ABSTRACT

In the seasons of today, the world is running with Ecommerce stores surrounding us. About all business stages practically are E-commerce store. With simple access to the Internet all over and learning about the technique, the market for Ecommerce has blasted to radiant statures in the ongoing past. Product Reviews. Product Reviews furnish an Ecommerce store with one of the most valuable resources available i.e. Customer Feedback. We could develop an automation for the business to extract insights from their clothing reviews. Because it is not easy to read thousands of reviews and it is a time consuming task. This could be useful to understand what people are talking about, things they like or things they do not like about. Improve their products from user's feedbacks. This study focuses on analyzing social media data with NLP to predict what a customer would buy in a retail store. In this study, we measured a 0.3 increase in accuracy when only various forms of nouns were extracted and analyzed. Further research may include Named-Entity Recognition (NER), especially for proper nouns. The researchers believe that this study will contribute to changing the trajectory in which NLP is applied in the retail industry. Therefore, the methodology and design used herein will improve the existing approaches that have already been employed concerning NLP and social media data analysis.

INDEX

1. KEYWORDS

CountVectorizer: Convert a collection of text documents to a matrix of token counts.

Topic Modeling: Topic Modeling automatically discover the hidden themes from given documents. It is an unsupervised text analytics algorithm that is used for finding the group of words from the given document. These group of words represents a topic.

Word-cloud: visual representations of words that give greater prominence to words that appear more frequently

Sklearn: Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

Naive Bayes: Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

Classification: In the fields of computational linguistics and probability, an n-gram (sometimes also called Q-gram) is a contiguous sequence of n items from a given sample of text or speech. ... The n-grams typically are collected from a text or speech corpus. When the items are words, n-grams may also be called shingles.

INTRODUCTION

Natural Language Processing (NLP) works both on text and speech data along with different types of engineering data for the development of intelligent systems (Agarwal & Jayant, 2019; Gupta, Ahlawat, & Sagar, 2017; Alzubi, 2015). Natural language processing techniques can help analyze the social media text, including customer review on a particular product through text (Agarwal & Jayant, 2019). Retailers can understand customer preferences by analyzing their customers' social e-commerce shopping data on online purchasing site; a methodology termed as the Semantic Analysis of Customer Revies(SAC) (Atefeh, Diana, & Graeme, 2017; Vajjhala, Rakshit, Oshogbunu, & Salisu, 2020). This discipline analyzes and transforms social media data into social media intelligence for decisionmakers. Hence, insights from e-commerce review data enable executives to lead with contextual knowledge rather than intuition (Erik & Emanuel, 2018). Machine learning algorithms have several advantages, including the ability to reduce uncertainty and predict precisely (Agarwal & Jayant, 2019). Machine learning algorithms also allow real-time analysis and advance forecasting coupled with processing of large volumes of data (Agarwal & Jayant, 2019). Alzubi et al. (2020) found neural networkbased approaches to have better performance as compared to traditional methods in the context of measuring sentence similarity based on the feature engineering and linguistic tools. Hence, combining NLP with machine learning algorithms can provide vital insights into consumer behavior. NLP helps predict online consumer behavior by giving the computing machines the ability to process textual data through computer science, artificial intelligence, and linguistic algorithms. For example, NLP makes it possible to conduct brand perception analysis on social media platforms through the use of Named Entity Recognition (NER) (Erik & Emanuel, 2018).

ABOUT DATA SETS

Data Description

The dataset is obtained from Kaggle:
Link for Kaggle Dataset: https://www.kaggle.com/nicapotato/womens-ecommerce-clothing-reviews

- **Clothing ID**: Integer Categorical variable that refers to the specific piece being reviewed.
- **Age**: Positive Integer variable of the reviewers age.
- **Title**: String variable for the title of the review.
- **Review Text**: String variable for the review body. The company name is replaced by te word 'retailer'.
- **Rating**: Positive Ordinal Integer variable for the product score granted by the customer from 1 Worst, to 5 Best.
- **Recommended IND**: Binary variable stating where the customer recommends the product where 1 is recommended, 0 is not recommended.
- **Positive Feedback Count**: Positive Integer documenting the number of other customers who found this review positive.
- **Division Name**: Categorical name of the product high level division.
- **Department Name**: Categorical name of the product department name.
- **Class Name**: Categorical name of the product class name.

METHODOLOGY

Methodology for Product Assessment using NLP and Supervised Learning:

1. Collection of Data from Kaggle
2. Perform EDA
3. Perform Cleaning
4. Export as Pickle
5. Read Cleaned data from EDA
6. Understand what are the distribution of each rank using classification
7. Build different classification models
8. Then we perform Topics Modelings
9. Clustering
10. Visualisation
11. Word Cloud is created

Load the Dataset:

Load the dataset

```python
# load the dataset.
df = pd.read_csv('Womens Clothing E-Commerce Reviews.csv')
df.head()
```

Python

	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

EDA IS PERFORMED:

EDA

```python
# list of column names.
df.columns
```

Python

```
Index(['Unnamed: 0', 'Clothing ID', 'Age', 'Title', 'Review Text', 'Rating',
       'Recommended IND', 'Positive Feedback Count', 'Division Name',
       'Department Name', 'Class Name'],
      dtype='object')
```

Empty markdown cell, double click or press enter to edit.

```python
# there are 23486 rows and 11 columns.
df.shape
```

Python

```
(23486, 11)
```

```python
# take out the 'Unnamed: 0' and 'Clothing ID' column.
# don't think they will be useful for my analysis.
df = df.drop(['Unnamed: 0', 'Clothing ID'], axis=1)

# clean the white space from the column names.
df = df.rename(columns=lambda x: x.replace(' ', ''))
```

Python

```python
# there are NaN in Title, ReviewText, DivisionName, DepartmentName, ClassName column.
df.info()
```

Python

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23486 entries, 0 to 23485
Data columns (total 9 columns):
Age                      23486 non-null int64
Title                    19676 non-null object
ReviewText               22641 non-null object
Rating                   23486 non-null int64
RecommendedIND           23486 non-null int64
PositiveFeedbackCount    23486 non-null int64
DivisionName             23472 non-null object
DepartmentName           23472 non-null object
ClassName                23472 non-null object
dtypes: int64(4), object(5)
memory usage: 1.6+ MB
```

Further Steps -

- Check how many NA's do we have?
- Clean the NA's
- Export as Pickle

Export as Pickle

```python
df.to_pickle('cleaned_df.pkl')
```

Building Different Classification Models:

1. Importing Libraries for use

```python
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.decomposition import NMF, TruncatedSVD
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from collections import Counter
from imblearn.over_sampling import SMOTE
```

2. Read cleaned data from EDA

Read the cleaned data from EDA

```python
df = pd.read_pickle('cleaned_df.pkl')
df.head()
```

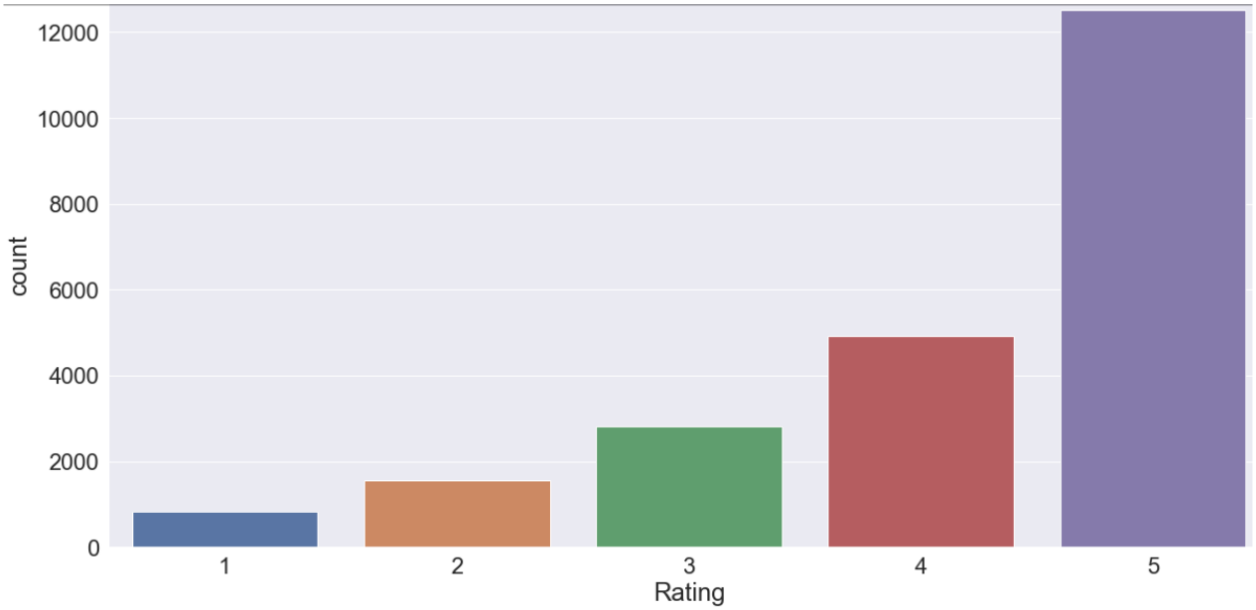
Python

	Age	ReviewText	Rating	RecommendedIND	PositiveFeedbackCount	DivisionName	DepartmentName	ClassName	CombinedText
0	33	Absolutely wonderful - silky and sexy and comf...	4	1	0	Initmates	Intimate	Intimates	Absolutely wonderful - silky and sexy and com...
1	34	Love this dress! it's sooo pretty. i happene...	5	1	4	General	Dresses	Dresses	Love this dress! it's sooo pretty. i happen...
2	60	I had such high hopes for this dress and reall...	3	0	0	General	Dresses	Dresses	Some major design flaws I had such high hopes ...
3	50	I love, love, love this jumpsuit. it's fun, fl...	5	1	0	General Petite	Bottoms	Pants	My favorite buy! I love, love, love this jumps...
4	47	This shirt is very flattering to all due to th...	5	1	6	General	Tops	Blouses	Flattering shirt This shirt is very flattering...

3. Understand what are the distribution of each rank

```python
plt.figure(figsize=(20,10))
sns.set(font_scale = 2)
sns.countplot(df.Rating)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x111edaac8>
```



4. Perform more Cleaning

```python
words_to_remove = ['love', 'dress', 'dresses', 'zip', 'zipper', 'fit', 'zippers', 'young', 'younger', 'pants', 'years']
text = 'I love things about dresses but not dress.'

import re
pattern = [f'(\\b{word}\\b)' for word in words_to_remove]
pattern = '|'.join(pattern)
re.sub(pattern, '', text)
```

```
'I  things about  but not .'
```

```python
df['ReviewTextLower'] = df.ReviewText
```

```python
df['ReviewTextLower'] = df.ReviewTextLower.str.lower()
```

```python
df['ReviewTextLower'].replace(to_replace=pattern, value='', regex=True, inplace=True)
```

```python
df.head()
```

Python

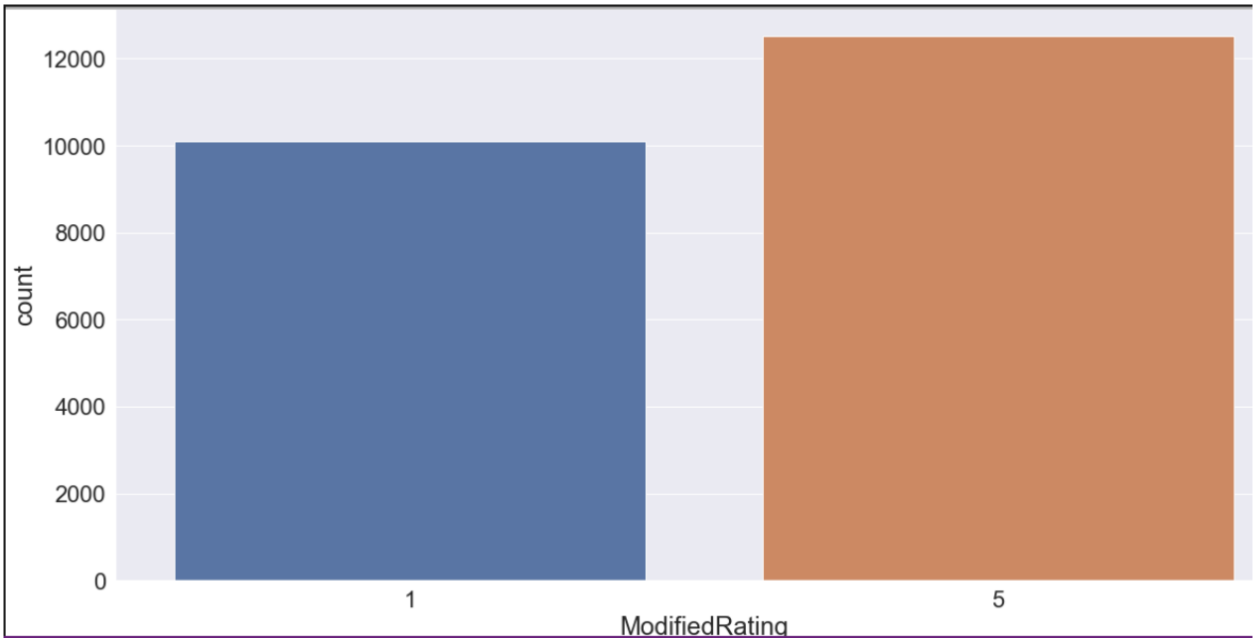
	Age	ReviewText	Rating	RecommendedIND	PositiveFeedbackCount	DivisionName	DepartmentName	ClassName	CombinedText	ReviewTextLower
0	33	Absolutely wonderful - silky and sexy and comf...	4	1	0	Initmates	Intimate	Intimates	Absolutely wonderful - silky and sexy and com...	absolutely wonderful - silky and sexy and comf...
1	34	Love this dress! it's sooo pretty. i happene...	5	1	4	General	Dresses	Dresses	Love this dress! it's sooo pretty. i happen...	this ! it's sooo pretty. i happened to find...
2	60	I had such high hopes for this dress and reall...	3	0	0	General	Dresses	Dresses	Some major design flaws I had such high hopes ...	i had such high hopes for this and really wan...
3	50	I love, love, love this jumpsuit. it's fun, fl...	5	1	0	General Petite	Bottoms	Pants	My favorite buy! I love, love, love this jumps...	i , , this jumpsuit. it's fun, flirty, and fa...
4	47	This shirt is very flattering to all due to th...	5	1	6	General	Tops	Blouses	Flattering shirt This shirt is very flattering...	this shirt is very flattering to all due to th...

5. Group different ranks together as my target rank.

```
df['ModifiedRating'] = df.Rating.replace([2, 3, 4], 1)
```

```
sns.set(font_scale = 2)
plt.figure(figsize=(20,10))
sns.countplot(df.ModifiedRating)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1180a8c18>
```




```
len(df[df.ModifiedRating == 1])
```

```
10101
```

```
len(df[df.ModifiedRating == 5])
```

```
12527
```

Two classes are not too imbalance and is in a safe range.

Topic Modelings is Performed:

1. Libraries is imported for use

```python
import numpy as np
import pandas as pd
import re

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.decomposition import NMF, TruncatedSVD
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
```

2. Import data from EDA

Import data from EDA

```python
df = pd.read_pickle('cleaned_df.pkl')
df.head()
```

Python

	Age	ReviewText	Rating	RecommendedIND	PositiveFeedbackCount	DivisionName	DepartmentName	ClassName	CombinedText
0	33	Absolutely wonderful - silky and sexy and comf...	4	1	0	Initmates	Intimate	Intimates	Absolutely wonderful - silky and sexy and com...
1	34	Love this dress! it's sooo pretty. i happene...	5	1	4	General	Dresses	Dresses	Love this dress! it's sooo pretty. i happen...
2	60	I had such high hopes for this dress and reall...	3	0	0	General	Dresses	Dresses	Some major design flaws I had such high hopes ...
3	50	I love, love, love this jumpsuit. it's fun, fl...	5	1	0	General Petite	Bottoms	Pants	My favorite buy! I love, love, love this jumps...
4	47	This shirt is very flattering to all due to th...	5	1	6	General	Tops	Blouses	Flattering shirt This shirt is very flattering...

3. Remove some frequent words

```python
words_to_remove = ['love', 'dress', 'dresses']
text = 'I love things about dresses but not dress.'

import re
pattern = [f'(\\b{word}\\b)' for word in words_to_remove]
pattern = '|'.join(pattern)
re.sub(pattern, '', text)
```

```
'I  things about  but not .'
```

```python
df['ReviewTextLower'] = df.ReviewText
```

```python
df['ReviewTextLower'] = df.ReviewTextLower.str.lower()
```

```python
df['ReviewTextLower'].replace(to_replace=pattern, value='', regex=True, inplace=True)
```

```
count_vectorizer = CountVectorizer(ngram_range=(1, 3),
                                   stop_words='english',
                                   token_pattern="\\b[a-z][a-z]+\\b",
                                   lowercase=True,
                                   max_df = 0.6, max_features=4000)
tfidf_vectorizer = TfidfVectorizer(ngram_range=(2, 3),
                                   stop_words='english',
                                   token_pattern="\\b[a-z][a-z]+\\b",
                                   lowercase=True,
                                   max_df = 0.6, max_features=4000)

cv_data = count_vectorizer.fit_transform(df.ReviewTextLower)
tfidf_data = tfidf_vectorizer.fit_transform(df.ReviewTextLower)
```

<u>Creating WordCloud</u>:

1. Vectorize the data

```
count_vectorizer = CountVectorizer(ngram_range=(1, 2),
                                   stop_words='english',
                                   token_pattern="\\b[a-z][a-z]+\\b",
                                   lowercase=True,
                                   max_df = 0.6, max_features=4000)
tfidf_vectorizer = TfidfVectorizer(ngram_range=(1, 2),
                                   stop_words='english',
                                   token_pattern="\\b[a-z][a-z]+\\b",
                                   lowercase=True,
                                   max_df = 0.6, max_features=4000)

cv_data = count_vectorizer.fit_transform(df.ReviewTextLower)
tfidf_data = tfidf_vectorizer.fit_transform(df.ReviewTextLower)
```

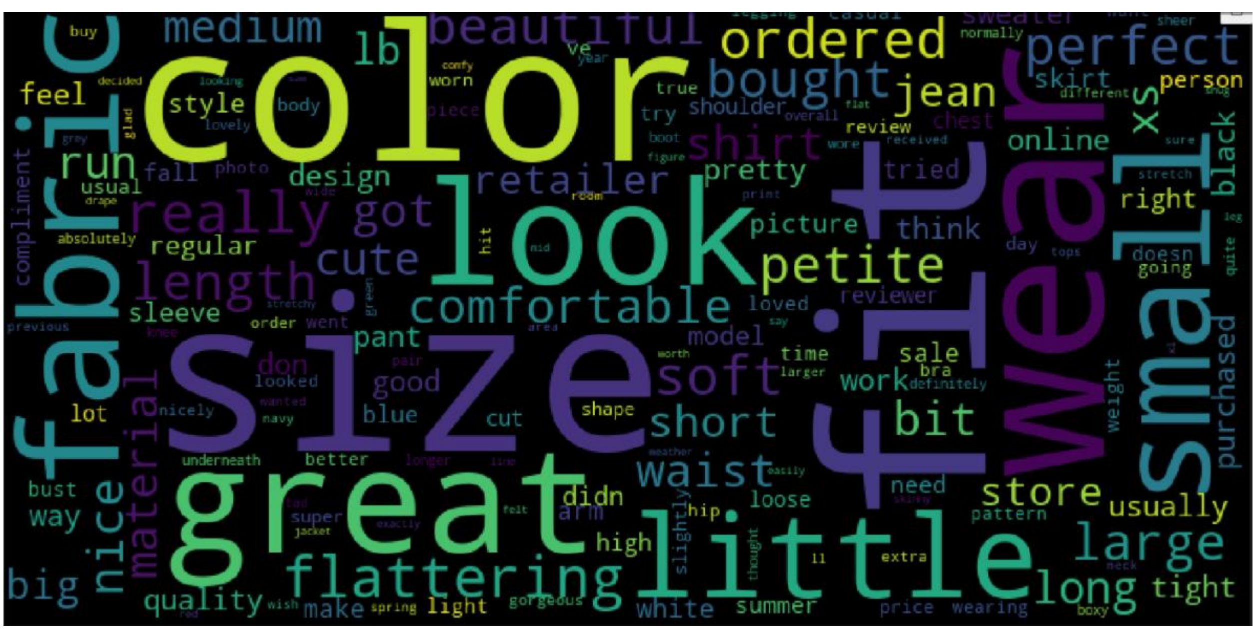
2. Create word cloud

```
for_wordcloud = count_vectorizer.get_feature_names()
for_wordcloud = for_wordcloud
for_wordcloud_str = ' '.join(for_wordcloud)

wordcloud = WordCloud(width=800, height=400, background_color ='white',
                      min_font_size = 7).generate(for_wordcloud_str)

plt.figure(figsize=(20, 10), facecolor=None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad=0)

plt.show()
```

3. Word Cloud is generated

If we want to change background color to black we can have:

```
for_wordcloud = count_vectorizer.get_feature_names()
for_wordcloud = for_wordcloud
for_wordcloud_str = ' '.join(for_wordcloud)

wordcloud = WordCloud(width=800, height=400, background_color ='black',
                      min_font_size = 7).generate(for_wordcloud_str)

plt.figure(figsize=(20, 10), facecolor=None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad=0)

plt.show()
```



CODE WITH RESULTS

CLASSIFICATION MODELS:

Build different classification models

Using ReviewTextLower column

```
# using ReviewTextLower column as my predictors
# using ModifiedRating column as my target variable
X = df['ReviewTextLower']
y = df['ModifiedRating']
```

```
# vectorization
count_vectorizer = CountVectorizer(ngram_range=(1, 2),
                                   stop_words='english',
                                   token_pattern="\\b[a-z][a-z]+\\b",
                                   lowercase=True,
                                   max_df = 0.6)
tfidf_vectorizer = TfidfVectorizer(ngram_range=(1, 2),
                                   stop_words='english',
                                   token_pattern="\\b[a-z][a-z]+\\b",
                                   lowercase=True,
                                   max_df = 0.6)

# transform my predictors
cv_data = count_vectorizer.fit_transform(X)
tfidf_data = tfidf_vectorizer.fit_transform(X)
```

```
len(count_vectorizer.vocabulary_)
```

```
255261
```

```
# split my data to 70/30
X_train, X_test, y_train, y_test = train_test_split(cv_data, y, test_size=0.3, random_state=42)
```

```
# train with multinomail Naive Bayes
nb = MultinomialNB()
nb.fit(X_train, y_train)
```

```
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
```

```
len(count_vectorizer.vocabulary_)
```

```
255261
```

```
# split my data to 70/30
X_train, X_test, y_train, y_test = train_test_split(cv_data, y, test_size=0.3, random_state=42)
```

```
# train with multinomail Naive Bayes
nb = MultinomialNB()
nb.fit(X_train, y_train)
```

```
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
```

```
# make prediction
y_pred = nb.predict(X_test)
```

```
# print out confusion matrix and classification report
print(confusion_matrix(y_test, y_pred))
print('\n')
print(classification_report(y_test, y_pred))
```

```
[[2129  896]
 [ 458 3306]]


             precision    recall  f1-score   support

          1       0.82      0.70      0.76      3025
          5       0.79      0.88      0.83      3764

avg / total       0.80      0.80      0.80      6789
```

```
# trying to make a prediction using this review
# this is an example for positive review
positive_example = df['ReviewTextLower'][1]
positive_example
```

```
' this !  it\'s sooo pretty.  i happened to find it in a store, and i\'m glad i did bc i never would have ordered it online bc it\'s petite.  i bought a petite and am 5\'8".  i  the length on me- hits just a little below the knee.  would definitely be a true midi on someone who is truly petite.'
```

```
# vectorize the positive_example
positive_example_vec = count_vectorizer.transform([positive_example])
# make prediction
nb.predict(positive_example_vec)[0]
```

Pytho

```
5
```

```
# trying to make a prediction using an negative review
negative_example = df['ReviewTextLower'][5]
negative_example
```

Pytho

```
'i  tracy reese , but this one is not for the very petite. i am just under 5 feet tall and usually wear a 0p in this brand. this  was very pretty out of the package but its a lot of . the skirt is long and very full so it overwhelmed my small frame. not a stranger to alterations, shortening and narrowing the skirt would take away from the embellishment of the garment. i  the color and the idea of the style but it just did not work on me. i returned this .'
```

```
negative_example_vec = count_vectorizer.transform([negative_example])
nb.predict(negative_example_vec)[0]
```

```
1
```

```python
negative_example = df['ReviewTextLower'][10]
negative_example
```

Python

```
' runs small esp where the zipper area runs. i ordered the sp which typically fits me and it was very tight! the material on the top looks and feels very cheap that even just pulling on it will cause it to rip the fabric. pretty disappointed as it was going to be my christmas  this year! needless to say it will be going back.'
```

```python
negative_example_vec = count_vectorizer.transform([negative_example])
nb.predict(negative_example_transformed)[0]
```

Python

```
1
```

Using Combinetext column

```python
df['CombinedTextLower'] = df.CombinedText
```

Python

```python
df['CombinedTextLower'] = df.CombinedTextLower.str.lower()
```

```
df['CombinedTextLower'].replace(to_replace=pattern, value='', regex=True, inplace=True)
```

```
X = df.CombinedTextLower
y = df.ModifiedRating
```

```
count_vectorizer = CountVectorizer(ngram_range=(1, 2),
                                   stop_words='english',
                                   token_pattern="\\b[a-z][a-z]+\\b",
                                   lowercase=True,
                                   max_df = 0.6)
tfidf_vectorizer = TfidfVectorizer(ngram_range=(1, 2),
                                   stop_words='english',
                                   token_pattern="\\b[a-z][a-z]+\\b",
                                   lowercase=True,
                                   max_df = 0.6)

cv_data = count_vectorizer.fit_transform(X)
tfidf_data = tfidf_vectorizer.fit_transform(X)
```

```
len(count_vectorizer.vocabulary_)
```

```
267875
```

```
X_train, X_test, y_train, y_test = train_test_split(cv_data, y, test_size=0.3, random_state=42)
```

```
nb = MultinomialNB()
nb.fit(X_train, y_train)
```

```
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
```

```
y_pred = nb.predict(X_test)
```

```
print(confusion_matrix(y_test, y_pred))
print('\n')
print(classification_report(y_test, y_pred))
```

```
[[2177  848]
 [ 424 3340]]
```

```
             precision    recall  f1-score   support

          1       0.84      0.72      0.77      3025
          5       0.80      0.89      0.84      3764

avg / total       0.82      0.81      0.81      6789
```

```python
logit = LogisticRegression()
logit.fit(X_train, y_train)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

```python
y_pred = logit.predict(X_test)
```

```python
print(confusion_matrix(y_test, y_pred))
print('\n')
print(classification_report(y_test, y_pred))
```

```
[[2250  775]
 [ 554 3210]]


             precision    recall  f1-score   support

          1       0.80      0.74      0.77      3025
          5       0.81      0.85      0.83      3764

avg / total       0.80      0.80      0.80      6789
```

```python
df.ReviewText[23]
```

Pytho

```
"Cute little dress fits tts. it is a little high waisted. good length for my 5'9 height. i like the dress, i'm just not in love with it. i dont think it looks or
feels cheap. it appears just as pictured."
```

```python
df.ReviewText[14]
```

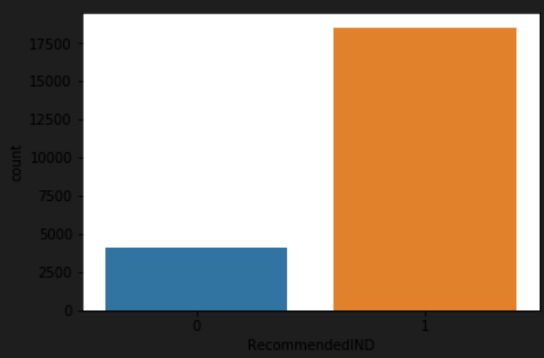
Pytho

```
'This is a nice choice for holiday gatherings. i like that the length grazes the knee so it is conservative enough for office related gatherings. the size small
fit me well - i am usually a size 2/4 with a small bust. in my opinion it runs small and those with larger busts will definitely have to size up (but then
perhaps the waist will be too big). the problem with this dress is the quality. the fabrics are terrible. the delicate netting type fabric on the top layer of
skirt got stuck in the zip'
```

predicting recommend or not

```
sns.countplot(df.RecommendedIND)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x11e983c88>
```




```
df.head()
```

```
df.head()
```

Python

	Age	ReviewText	Rating	RecommendedIND	PositiveFeedbackCount	DivisionName	DepartmentName	ClassName	CombinedText	ReviewTextLower	ModifiedRating	Cc
0	33	Absolutely wonderful - silky and sexy and comf...	4	1	0	Initmates	Intimate	Intimates	Absolutely wonderful - silky and sexy and com...	absolutely wonderful - silky and sexy and comf...	1	a
1	34	Love this dress! it's sooo pretty. i happene...	5	1	4	General	Dresses	Dresses	Love this dress! it's sooo pretty. i happen...	this ! it's sooo pretty. i happened to find...	5	tl
2	60	I had such high hopes for this dress and reall...	3	0	0	General	Dresses	Dresses	Some major design flaws I had such high hopes ...	i had such high hopes for this and really wan...	1	fl
3	50	I love, love, love this jumpsuit. it's fun, fl...	5	1	0	General Petite	Bottoms	Pants	My favorite buy! I love, love, love this jumps...	i , , this jumpsuit. it's fun, flirty, and fa...	5	tl
4	47	This shirt is very flattering to all due to	5	1	6	General	Tops	Blouses	Flattering shirt This shirt is very flattering...	this shirt is very flattering to all due to th...	5	

```
logit = LogisticRegression()
logit.fit(X_train_smoted, y_train_smoted)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

```
y_pred = logit.predict(X_test)
```

```
print(confusion_matrix(y_test, y_pred))
print('\n')
print(classification_report(y_test, y_pred))
```

```
[[ 844  386]
 [ 312 5247]]


             precision    recall  f1-score   support

          0       0.73      0.69      0.71      1230
          1       0.93      0.94      0.94      5559
```

TOPIC MODELINGS:

Try Topic Modelings

```python
# def functions for topic modelings
def display_topics(model, feature_names, no_top_words, topic_names=None):
    for ix, topic in enumerate(model.components_):
        if not topic_names or not topic_names[ix]:
            print("\nTopic ", ix)
        else:
            print("\nTopic: '",topic_names[ix],"'")
        print(", ".join([feature_names[i]
                        for i in topic.argsort()[:-no_top_words - 1:-1]]))

def display_topics2(model, feature_names, no_top_words=10, topic_names = None):
    for index, topic in enumerate(model.components_):
        if not topic_names or not topic_names[index]:
            print(f"\nTopic {index}")
        else:
            print(f"\nTopic {topic_names[index]}:")
        msg = ", ".join([f'{feature_names[i]} ({topic[i]:6.4f})'
                          for i in topic.argsort()[:-no_top_words-1:-1]])
        print(msg)
```

```python
# try using 50 dimensions
n_comp = 50
lsa_tfidf = TruncatedSVD(n_components=n_comp)
lsa_cv = TruncatedSVD(n_components=n_comp)
nmf_tfidf = NMF(n_components=n_comp)
nmf_cv = NMF(n_components=n_comp)

lsa_tfidf_data = lsa_tfidf.fit_transform(tfidf_data)
lsa_cv_data = lsa_cv.fit_transform(cv_data)
nmf_tfidf_data = nmf_tfidf.fit_transform(tfidf_data)
nmf_cv_data = nmf_cv.fit_transform(cv_data)
```

Python

```python
# topic modeling with lsa and tfidf
display_topics2(lsa_tfidf, tfidf_vectorizer.get_feature_names(),8)
```

Python

Output exceeds the size limit. Open the full output data in a text editor

```
Topic 0
true size (0.7192), fits true (0.2980), fits true size (0.2964), fit true (0.1697), fit true size (0.1690), looks great (0.0992), runs true (0.0958), runs true size

Topic 1
true size (0.3337), fits true (0.2025), fits true size (0.2014), fit true size (0.0718), fit true (0.0709), runs true size (0.0276), runs true (0.0247), true size fi
```

1. Topic modeling with lsa and countvectorizer

```python
# topic modeling with lsa and countvectorizer
display_topics2(lsa_cv, count_vectorizer.get_feature_names(),10)
```

```
Output exceeds the size limit. Open the full output data in a text editor

Topic 0
size (0.3559), fit (0.2668), like (0.2594), wear (0.2185), just (0.1979), great (0.1856), small (0.1772), fabric (0.1560), color (0.1481), ordered (0.1376)

Topic 1
size (0.7700), small (0.1356), true (0.1063), true size (0.1032), fit (0.0917), ordered (0.0893), size small (0.0494), large (0.0436), usual (0.0427), runs (0.0416)

Topic 2
like (0.6521), size (0.1364), look (0.1043), really (0.0914), just (0.0751), looked (0.0571), model (0.0542), fabric (0.0527), look like (0.0498), didn (0.0474)

Topic 3
wear (0.5233), small (0.4081), medium (0.1465), usually (0.1049), large (0.1010), runs (0.0785), usually wear (0.0760), shirt (0.0692), fits (0.0607), retailer (0.0

Topic 4
fit (0.5410), small (0.4336), medium (0.1533), ordered (0.1247), just (0.1090), xs (0.0945), large (0.0871), usually (0.0783), petite (0.0709), waist (0.0581)

Topic 5
just (0.6467), fabric (0.1602), color (0.1105), right (0.1011), beautiful (0.0919), really (0.0901), soft (0.0723), flattering (0.0687), bit (0.0675), little (0.062

Topic 6
small (0.4513), great (0.4445), ordered (0.1268), medium (0.1023), looks (0.1016), large (0.0743), runs (0.0739), little (0.0704), fits (0.0606), look (0.0606)
```

2. Topic modeling with nmf and tf-idf

```python
# topic modeling with nmf and tfidf
display_topics2(nmf_tfidf, tfidf_vectorizer.get_feature_names(),10)
```

```
Output exceeds the size limit. Open the full output data in a text editor

Topic 0
like (2.5308), look (1.3775), really (1.3128), fabric (1.1826), nice (0.8023), good (0.7609), looks (0.7597), material (0.6801), model (0.5913), pretty (0.5628)

Topic 1
wear (2.1296), perfect (1.6434), summer (0.8048), fall (0.4813), easy (0.3732), bought (0.3682), wait (0.3537), spring (0.3357), wait wear (0.3067), light (0.3060)

Topic 2
size (1.5858), true (1.2507), true size (1.2326), fits (0.5408), fits true (0.5220), fit true (0.2438), runs true (0.2037), size fits (0.1451), usual (0.1114), size

Topic 3
small (2.0832), medium (0.9048), usually (0.4495), ordered (0.4375), size small (0.3880), small medium (0.3003), size (0.2861), ordered small (0.2700), extra (0.259

Topic 4
shirt (2.6723), white (0.2181), great shirt (0.1486), shirt great (0.1293), cute shirt (0.1287), shirts (0.1227), like shirt (0.1154), bought shirt (0.1141), boxy (

Topic 5
great (2.4087), looks great (0.5386), looks (0.5052), fits great (0.3430), fits (0.2880), quality (0.2716), great quality (0.2714), great fit (0.2572), fit great (0

Topic 6
sweater (2.7870), warm (0.3254), sleeves (0.3219), soft (0.2718), long (0.2155), coat (0.2106), cozy (0.2075), beautiful sweater (0.1930), itchy (0.1921), great swe
```

3. Topic modeling with nmf and countvectorizer

```python
# topic modeling with nmf and countvectorizer
display_topics2(nmf_cv, count_vectorizer.get_feature_names(),10)
```

```
Output exceeds the size limit. Open the full output data in a text editor

Topic 0
flattering (5.2352), comfortable (5.2316), soft (4.7632), bought (3.2933), material (2.8323), pants (2.3196), super (2.2999), jeans (2.2132), cute (1.5795), colors

Topic 1
size (8.8916), true (1.3392), true size (1.2366), usual (0.4829), fits (0.4631), usual size (0.4094), smaller (0.3531), size small (0.3256), ordered size (0.3130),

Topic 2
like (7.6427), looks (0.9452), feel (0.5449), model (0.4726), looks like (0.4336), don (0.4022), feel like (0.3996), looked (0.3832), material (0.3193), look like (
```

Modeling Using StandardScaler:

```
# initialize vectorizers
count_vectorizer = CountVectorizer(ngram_range=(1, 2),
                                   stop_words='english',
                                   token_pattern="\\b[a-z][a-z]+\\b",
                                   lowercase=True,
                                   max_df = 0.6, max_features=4000)
tfidf_vectorizer = TfidfVectorizer(ngram_range=(1, 2),
                                   stop_words='english',
                                   token_pattern="\\b[a-z][a-z]+\\b",
                                   lowercase=True,
                                   max_df = 0.6, max_features=4000)

# transfomred my text data using vectorizers
cv_data = count_vectorizer.fit_transform(df.ReviewTextLower)
tfidf_data = tfidf_vectorizer.fit_transform(df.ReviewTextLower)

# initialized reducers with dimensions
n_comp = 5
lsa_tfidf = TruncatedSVD(n_components=n_comp)
lsa_cv = TruncatedSVD(n_components=n_comp)
nmf_tfidf = NMF(n_components=n_comp)
nmf_cv = NMF(n_components=n_comp)

# transformed my vectorizers data using reducers
lsa_tfidf_data = lsa_tfidf.fit_transform(tfidf_data)
lsa_cv_data = lsa_cv.fit_transform(cv_data)
nmf_tfidf_data = nmf_tfidf.fit_transform(tfidf_data)
nmf_cv_data = nmf_cv.fit_transform(cv_data)
```

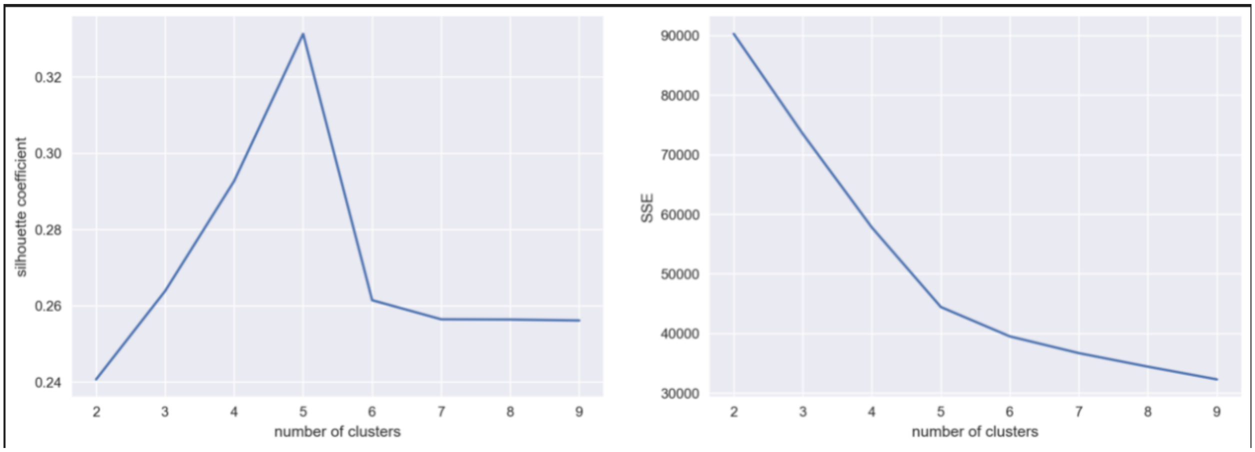
```
# initialize standardscaler
from sklearn.preprocessing import StandardScaler
SS = StandardScaler()

# transform my reducer data using standardscaler
lsa_tfidf_data_sclaed = SS.fit_transform(lsa_tfidf_data)
lsa_cv_data_sclaed = SS.fit_transform(lsa_cv_data)
nmf_tfidf_data_scaled = SS.fit_transform(nmf_tfidf_data)
nmf_cv_data_scaled = SS.fit_transform(nmf_cv_data)
```

```
display_topics2(lsa_tfidf, tfidf_vectorizer.get_feature_names(),8)
```

Topic Modeling using nmf and tf-idf

```
SSEs = []
Sil_coefs = []
for k in range(2,10):
    km = KMeans(n_clusters=k, random_state=42)
    km.fit(nmf_tfidf_data_scaled)
    labels = km.labels_
    Sil_coefs.append(silhouette_score(nmf_tfidf_data_scaled, labels, metric='euclidean'))
    SSEs.append(km.inertia_)
```

```python
fig, (ax1, ax2) = plt.subplots(1,2, figsize=(15,5), sharex=True, dpi=200)
k_clusters = range(2,10)
ax1.plot(k_clusters, Sil_coefs)
ax1.set_xlabel('number of clusters')
ax1.set_ylabel('silhouette coefficient')

# plot here on ax2
ax2.plot(k_clusters, SSEs)
ax2.set_xlabel('number of clusters')
ax2.set_ylabel('SSE');
```



TRYING SOME MORE GRAMS

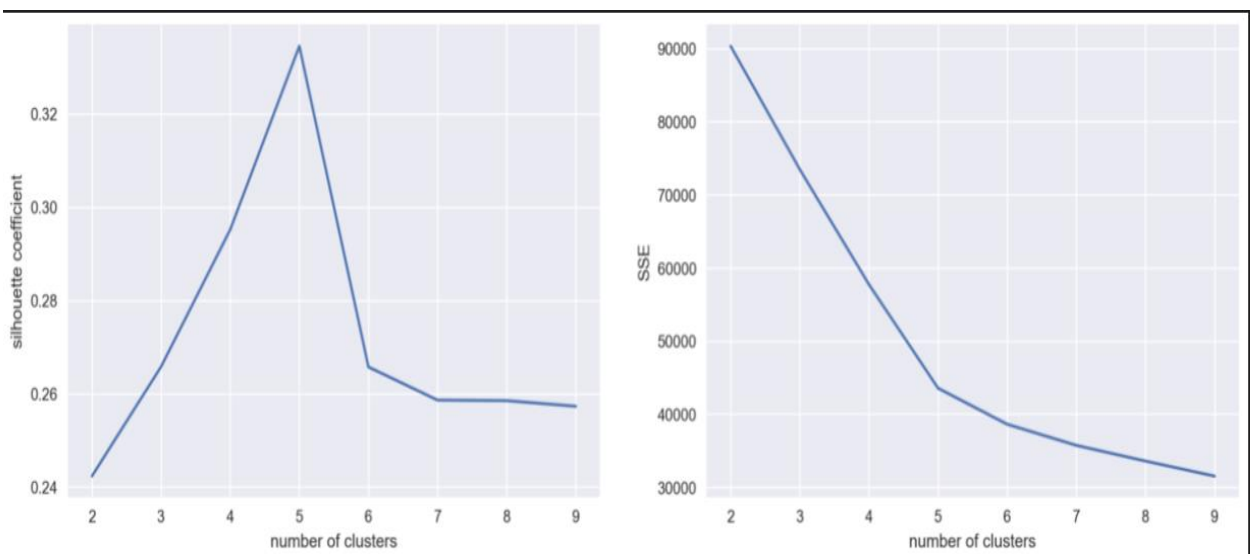
```python
# initialize vectorizers
count_vectorizer = CountVectorizer(ngram_range=(1, 3),
                                   stop_words='english',
                                   token_pattern="\\b[a-z][a-z]+\\b",
                                   lowercase=True,
                                   max_df = 0.6, max_features=4000)
tfidf_vectorizer = TfidfVectorizer(ngram_range=(1, 3),
                                   stop_words='english',
                                   token_pattern="\\b[a-z][a-z]+\\b",
                                   lowercase=True,
                                   max_df = 0.6, max_features=4000)

# transfomred my text data using vectorizers
cv_data = count_vectorizer.fit_transform(df.ReviewTextLower)
tfidf_data = tfidf_vectorizer.fit_transform(df.ReviewTextLower)
```

```python
# initialized reducers with dimensions
n_comp = 5
lsa_tfidf = TruncatedSVD(n_components=n_comp)
lsa_cv = TruncatedSVD(n_components=n_comp)
nmf_tfidf = NMF(n_components=n_comp)
nmf_cv = NMF(n_components=n_comp)

# transformed my vectorizers data using reducers
lsa_tfidf_data = lsa_tfidf.fit_transform(tfidf_data)
lsa_cv_data = lsa_cv.fit_transform(cv_data)
nmf_tfidf_data = nmf_tfidf.fit_transform(tfidf_data)
nmf_cv_data = nmf_cv.fit_transform(cv_data)
```

```python
# initialize standardscaler
from sklearn.preprocessing import StandardScaler
SS = StandardScaler()

# transform my reducer data using standardscaler
lsa_tfidf_data_sclaed = SS.fit_transform(lsa_tfidf_data)
lsa_cv_data_sclaed = SS.fit_transform(lsa_cv_data)
nmf_tfidf_data_scaled = SS.fit_transform(nmf_tfidf_data)
nmf_cv_data_scaled = SS.fit_transform(nmf_cv_data)
```

```python
SSEs = []
Sil_coefs = []
for k in range(2,10):
    km = KMeans(n_clusters=k, random_state=42)
    km.fit(nmf_tfidf_data_scaled)
    labels = km.labels_
    Sil_coefs.append(silhouette_score(nmf_tfidf_data_scaled, labels, metric='euclidean'))
    SSEs.append(km.inertia_)
```

```python
fig, (ax1, ax2) = plt.subplots(1,2, figsize=(15,5), sharex=True, dpi=200)
k_clusters = range(2,10)
ax1.plot(k_clusters, Sil_coefs)
ax1.set_xlabel('number of clusters')
ax1.set_ylabel('silhouette coefficient')

# plot here on ax2
ax2.plot(k_clusters, SSEs)
ax2.set_xlabel('number of clusters')
ax2.set_ylabel('SSE');
```



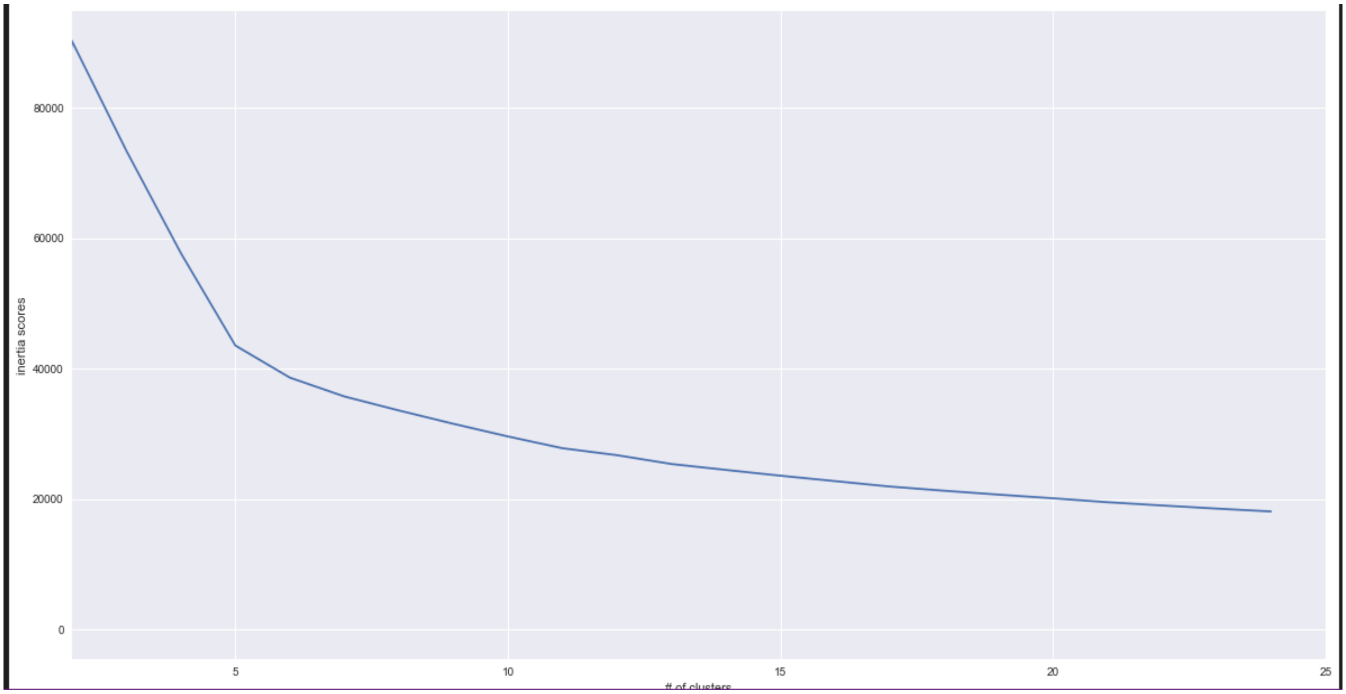
```python
inertia = [0,0]

for n_clusters in range(2, 25):
    km = KMeans(n_clusters = n_clusters)
    km.fit(nmf_tfidf_data_scaled)
    msg = f"""# clusters: {n_clusters:2d}   Inertia: {km.inertia_:8.6f}"""
    inertia.append(km.inertia_)
    print(msg)
```

```
# clusters:  2   Inertia: 90338.984804
# clusters:  3   Inertia: 73482.227460
# clusters:  4   Inertia: 57759.730141
# clusters:  5   Inertia: 43570.035756
# clusters:  6   Inertia: 38642.495455
# clusters:  7   Inertia: 35750.623578
# clusters:  8   Inertia: 33599.948861
# clusters:  9   Inertia: 31554.894180
# clusters: 10   Inertia: 29611.302321
# clusters: 11   Inertia: 27805.735090
# clusters: 12   Inertia: 26737.903024
# clusters: 13   Inertia: 25385.570846
# clusters: 14   Inertia: 24474.577757
# clusters: 15   Inertia: 23586.923368
# clusters: 16   Inertia: 22757.422779
# clusters: 17   Inertia: 21928.717501
# clusters: 18   Inertia: 21282.148283
```

```
# clusters: 19   Inertia: 20684.113418
# clusters: 20   Inertia: 20119.370720
# clusters: 21   Inertia: 19518.804959
# clusters: 22   Inertia: 19017.209461
# clusters: 23   Inertia: 18540.845862
# clusters: 24   Inertia: 18097.930785
```

```python
plt.figure(figsize=(20,10))
plt.plot(inertia)
plt.xlabel('# of clusters')
plt.xlim((2,25))
plt.ylabel('inertia scores')
#plt.ylim((650,1200))
```

```
... Text(0,0.5,'inertia scores')
```

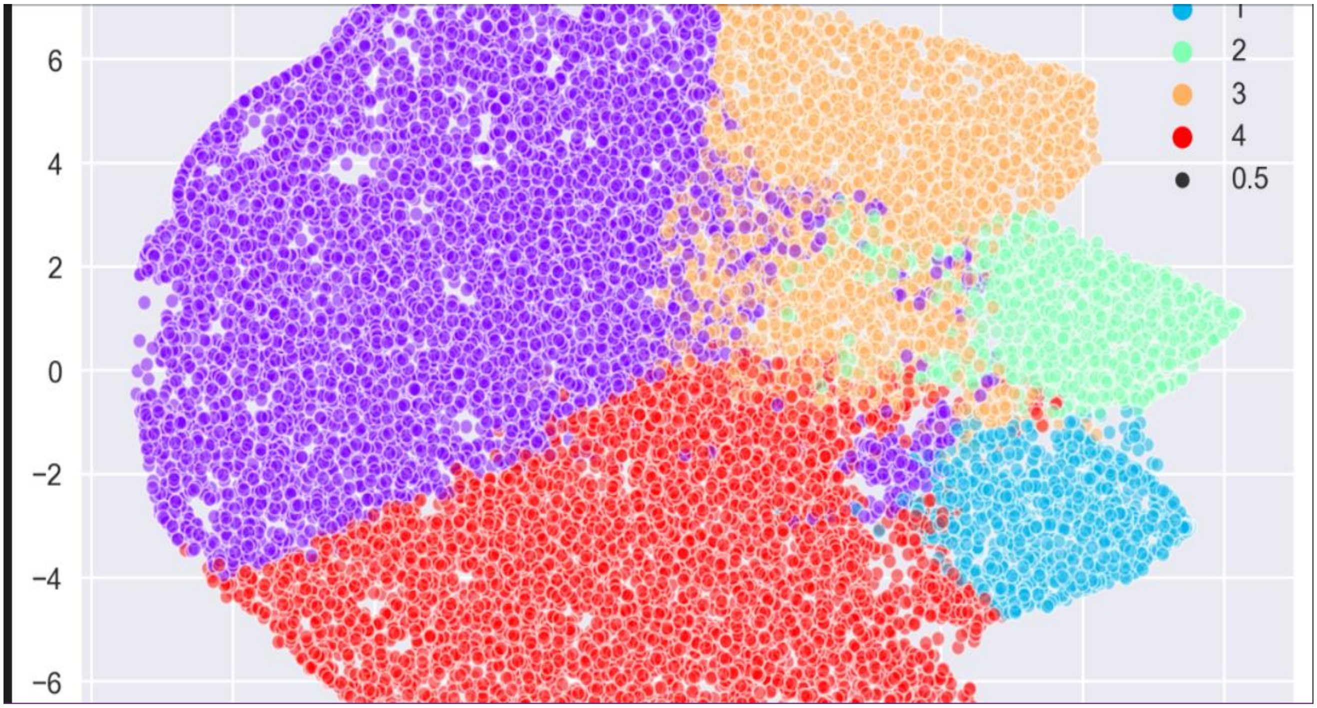



```python
# running cluster
k = 5
kmeans = KMeans(n_clusters=k, random_state=42)
kmeans.fit(nmf_tfidf_data_scaled)
centers = kmeans.cluster_centers_.argsort()[:,::-1]
terms = tfidf_vectorizer.get_feature_names()

for i in range(0,k):
    word_list=[]
    print("cluster%d:"% i)
    for j in centers[i,:15]:
        word_list.append(terms[j])
    print(word_list)
```

```
cluster0:
['able', 'absolutely', 'able wear', 'absolute', 'able try']
cluster1:
['absolutely', 'absolute', 'able wear', 'able try', 'able']
cluster2:
['able wear', 'able try', 'absolutely', 'absolute', 'able']
cluster3:
['absolute', 'able wear', 'absolutely', 'able try', 'able']
cluster4:
['able try', 'absolutely', 'able wear', 'absolute', 'able']
```

```
tsne = TSNE(n_components=2, verbose=1, perplexity=92, n_iter=300, random_state=42)
X_ne = tsne.fit_transform(nmf_tfidf_data_scaled[2000:])

figsize=(20,15)
plt.figure(dpi=300)
sns.scatterplot(X_ne[:, 0], X_ne[:, 1], hue=kmeans.labels_[2000:], alpha=0.5, size = 0.5, palette='rainbow', legend='full');
```

```
[t-SNE] Computing 277 nearest neighbors...
[t-SNE] Indexed 20628 samples in 0.093s...
[t-SNE] Computed neighbors for 20628 samples in 2.066s...
[t-SNE] Computed conditional probabilities for sample 1000 / 20628
[t-SNE] Computed conditional probabilities for sample 2000 / 20628
[t-SNE] Computed conditional probabilities for sample 3000 / 20628
[t-SNE] Computed conditional probabilities for sample 4000 / 20628
[t-SNE] Computed conditional probabilities for sample 5000 / 20628
[t-SNE] Computed conditional probabilities for sample 6000 / 20628
[t-SNE] Computed conditional probabilities for sample 7000 / 20628
[t-SNE] Computed conditional probabilities for sample 8000 / 20628
[t-SNE] Computed conditional probabilities for sample 9000 / 20628
[t-SNE] Computed conditional probabilities for sample 10000 / 20628
[t-SNE] Computed conditional probabilities for sample 11000 / 20628
[t-SNE] Computed conditional probabilities for sample 12000 / 20628
[t-SNE] Computed conditional probabilities for sample 13000 / 20628
[t-SNE] Computed conditional probabilities for sample 14000 / 20628
[t-SNE] Computed conditional probabilities for sample 15000 / 20628
[t-SNE] Computed conditional probabilities for sample 16000 / 20628
[t-SNE] Computed conditional probabilities for sample 17000 / 20628
[t-SNE] Computed conditional probabilities for sample 18000 / 20628
[t-SNE] Computed conditional probabilities for sample 19000 / 20628
[t-SNE] Computed conditional probabilities for sample 20000 / 20628
[t-SNE] Computed conditional probabilities for sample 20628 / 20628
[t-SNE] Mean sigma: 0.225878
```




```python
indices_max = [index for index, value in enumerate(kmeans.labels_) if value==0]
for rev_index in indices_max[:5]:
    print(rev_index, str(df.ReviewText[rev_index]))
    print("\n")
```

Python

```
1 Love this dress!  it's sooo pretty.  i happened to find it in a store, and i'm glad i did bc i never would have ordered it online bc it's petite.  i bought a petite and am 5'8".  i love the length on me- hits just a little below the knee.  would definitely be a true midi on someone who is truly petite.


5 I love tracy reese dresses, but this one is not for the very petite. i am just under 5 feet tall and usually wear a 0p in this brand. this dress was very pretty out of the package but its a lot of dress. the skirt is long and very full so it overwhelmed my small frame. not a stranger to alterations, shortening and narrowing the skirt would take away from the embellishment of the garment. i love the color and the idea of the style but it just did not work on me. i returned this dress.


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.
```

```
7 I ordered this in carbon for store pick up, and had a ton of stuff (as always) to try on and used this top to pair (skirts and pants). everything went with it. the color is really nice charcoal with shimmer, and went well with pencil skirts, flare pants, etc. my only compaint is it is a bit big, sleeves are long and it doesn't go in petite. also a bit loose for me, but no xxs... so i kept it and wil ldecide later since the light color is already sold out in hte smallest size...


11 This dress is perfection! so pretty and flattering.
```

5. Conclusion

Existing problems to resolve:

Topic Modeling: for example, what are the positive and negative things people are talking about that clothing/shoes. To see if I could find any topic by calculating frequencies of word or combination of words happen in a topic.

Separation of good and bad reviews using clustering: to separate out or find pattern of bad and good reviews for different products, so ones can send them to corresponding departments for attention by using clustering methods. This could be very hard since clustering method is an unsupervised machine learning technique that find hidden patterns from the data.

The project was focused on analyzing women clothing data with NLP to predict what a customer would buy in a retail store. In this study, we measured a 0.3 increase in accuracy when only various forms of nouns were extracted and analyzed. . Further research may include NamedEntity Recognition (NER), especially for proper nouns. The researchers believe that this study will change the trajectory in which NLP is applied in the retail industry. Therefore, the methodology and design used herein will improve the existing approaches that have already been employed concerning NLP and social media data analysis. Despite the gaps identified by the researcher, this study is limited in several ways. th any of our intended retailer's tweets were ousted in this research. Another limitation of our study was the limited number of rules and patterns applied, because of which we might have missed some of the cause-effect relations.