Jayanta_Analysis_of_E-Commerce_Women's_Clothing_Reviews_BID_Assignment_1166

March 24, 2021

BID Assignment

0.0.1 Name-Jayanta Ghosh

0.0.2 Roll no-190112

0.0.3 Seat No-1166

1 Analysis of E-Commerce Women's Clothing Reviews

In this project, I want to practice Natural Language Processing (NLP), Supervised Machine Learning and Unsupervised Machine Learning. After some research on what dataset I could obtain from the web, I found a women clothings dataset of a real e-commerce business. I think it could be cool and useful to a business if I 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 valuable for various reasons. For example: 1) Understand trends: to understand what people are talking about, things they like or things they do not like about.

2) Improve your products from users feedbacks.

3) To follow up with your user about the product that they don't like and further to understand the problem.

4) To decrease return rate, re-stocking fees is one of the big expenses for e-commerce to succeed or even stay alive.

1.1 Problems that I want to solve

So for the purpose of this project, I want to explore the followings:

1) 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.

2) "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.

3) Rankings of various products of Women's clothes provided in E-Commerce Website Data.

1.2 Project Design

1) Clean and perform Exploratory Data Analysis (EDA) on my data.
2) Vectorization of my cleaned text data (Count Vectorizer and TF-IDF).
3) Generate a WordCloud to see what are the most frequent words that people are talking about.
4) Perform Topic Modelings to see if I could find some clear different topics that people are talking about.
5) Use clustering methods to cluster out pattern from my text data and see if I could cluster out those bad reviews (or separate types of reviews). And use TSNE to visualize my clusters.
6) Perform a supervised learning problem with the Rating column from the dataset to classify good and bad reviews.
7) Achieved Rankings of Products provided in the dataset by E-Commerce by various lists.

1.3 Data And Technologies I used

The dataset I used could be obtained from Kaggle data source https://www.kaggle.com/nicapotato/womens-ecommerce-clothing-reviews, consists of 23486 entires of different clothings reviews and 11 different columns.

The tools that I have used in this project are numpy, pandas, matplotlib, seaborn, wordcloud, sklearn especially with CountVectorizer, TfidfVectorizer, Kmeans, TSNE, NMF, TruncatedSVD, silhouette_score, MultinomialNB and LogisticRegression.

1.4 Installing the required Packages

[5]:
```
! pip install imblearn
```

```
Collecting imblearn
  Downloading https://files.pythonhosted.org/packages/81/a7/4179e6ebfd654bd0eac0
b9c06125b8b4c96a9d0a8ff9e9507eb2a26d2d7e/imblearn-0.0-py2.py3-none-any.whl
Collecting imbalanced-learn (from imblearn)
  Downloading https://files.pythonhosted.org/packages/80/98/dc784205a7e3034e84d4
1ac4781660c67ad6327f2f5a80c568df31673d1c/imbalanced_learn-0.8.0-py3-none-any.whl
(206kB)
Requirement already satisfied: scikit-learn>=0.24 in
c:\users\welcome\appdata\local\programs\python\python36\lib\site-packages (from
imbalanced-learn->imblearn)
Requirement already satisfied: numpy>=1.13.3 in
c:\users\welcome\appdata\local\programs\python\python36\lib\site-packages (from
imbalanced-learn->imblearn)
Requirement already satisfied: scipy>=0.19.1 in
c:\users\welcome\appdata\local\programs\python\python36\lib\site-packages (from
imbalanced-learn->imblearn)
Requirement already satisfied: joblib>=0.11 in
```

```
c:\users\welcome\appdata\local\programs\python\python36\lib\site-packages (from
imbalanced-learn->imblearn)
Requirement already satisfied: threadpoolctl>=2.0.0 in
c:\users\welcome\appdata\local\programs\python\python36\lib\site-packages (from
scikit-learn>=0.24->imbalanced-learn->imblearn)
Installing collected packages: imbalanced-learn, imblearn
Successfully installed imbalanced-learn-0.8.0 imblearn-0.0

  Cache entry deserialization failed, entry ignored
  Cache entry deserialization failed, entry ignored
  Cache entry deserialization failed, entry ignored
  Cache entry deserialization failed, entry ignored
You are using pip version 9.0.3, however version 21.0.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip'
command.
```

[3]:
```
!pip install wordcloud
```

```
Collecting wordcloud
  Using cached https://files.pythonhosted.org/packages/a4/fd/0be730526bae8355cad
bea1c9e1cfbbfe5a92347f6937ea4474c467c04d1/wordcloud-1.8.1-cp36-cp36m-win_amd64.w
hl
Requirement already satisfied: matplotlib in
c:\users\welcome\appdata\local\programs\python\python36\lib\site-packages (from
wordcloud)
Requirement already satisfied: numpy>=1.6.1 in
c:\users\welcome\appdata\local\programs\python\python36\lib\site-packages (from
wordcloud)
Requirement already satisfied: pillow in
c:\users\welcome\appdata\local\programs\python\python36\lib\site-packages (from
wordcloud)
Requirement already satisfied: cycler>=0.10 in
c:\users\welcome\appdata\local\programs\python\python36\lib\site-packages (from
matplotlib->wordcloud)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in
c:\users\welcome\appdata\local\programs\python\python36\lib\site-packages (from
matplotlib->wordcloud)
Requirement already satisfied: python-dateutil>=2.1 in
c:\users\welcome\appdata\local\programs\python\python36\lib\site-packages (from
matplotlib->wordcloud)
Requirement already satisfied: kiwisolver>=1.0.1 in
c:\users\welcome\appdata\local\programs\python\python36\lib\site-packages (from
matplotlib->wordcloud)
Requirement already satisfied: six in
c:\users\welcome\appdata\local\programs\python\python36\lib\site-packages (from
cycler>=0.10->matplotlib->wordcloud)
Installing collected packages: wordcloud
Successfully installed wordcloud-1.8.1
```

```
You are using pip version 9.0.3, however version 21.0.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip'
command.
```

1.5 Importing Required Libraries

[141]:
```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.simplefilter(action='ignore')

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

2 Data Description

The dataset is obtained from Kaggle (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.

2.0.1 To-do list for Data Analysis

- load dataset
- EDA
- Preform cleanings

3 Load the dataset

[142]:
```
# load the dataset.
df = pd.read_csv('Womens Clothing E-Commerce Reviews.csv')
df.head()
```

[142]:

	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

3.1 Exploratory Data Analysis

[143]:
```
# list of column names.
df.columns
```

[143]:
```
Index(['Unnamed: 0', 'Clothing ID', 'Age', 'Title', 'Review Text', 'Rating',
       'Recommended IND', 'Positive Feedback Count', 'Division Name',
       'Department Name', 'Class Name'],
      dtype='object')
```

[144]:
```
# there are 23486 rows and 11 columns.
df.shape
```

[144]:
```
(23486, 11)
```

[145]:
```
# 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(' ', ''))
```

[146]:
```
# there are NaN in Title, ReviewText, DivisionName, DepartmentName, ClassName␣
↪column.
df.info()
```

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

3.1.1 How many NAs we have?

Columns Name	Amount of NAs	% of dataset
Title	3810	~16%
ReviewText	845	~3.5%
DivisionName	14	~0.05%
DepartmentName	14	~0.05%
ClassName	14	~0.05%

- Those NAs in **DivisionName**, **DepartmentName** and **ClassName** are the same, since there are only ~0.05% of those, I will drop them.
- For **ReviewText**, since we are preforming our NLP on that column primarily, we can't preform NLP if we don't have any text to analyze, so I will drop them.
- For **Title**, since it is text and I am doing NLP, eventually I probabily will need to prefrom NLP on that column and combine with the ReviewText column to see if there is any meaningful unsupervised learning results. There are 3810 (~16% of my whole dataset) NAs and I don't really want to drop all of them. So my solution is to create a new column called CombineText, which comebine the Title and ReviewText column together and makes the Title like the first sentense of the review.

3.1.2 Clean the NAs

[147]:
```
# dropping NAs for 4 columns.
subset = ['ReviewText', 'DivisionName', 'DepartmentName', 'ClassName']
df = df.dropna(subset=subset)

print('Now length of df is: ', len(df))
```

```
Now length of df is:  22628
```

ReviewText column will be my primary column for NLP.

[148]:
```
# first fill NAs in the Title column with space, so I can concatenate the Title␣
↪and ReviewText column together.
df.Title.fillna('', inplace=True)

# create a new column named CombinedText with Title and ReviewText.
df['CombinedText'] = df.Title + ' ' + df.ReviewText

# drop the Title column.
df.drop('Title', axis=1, inplace=True)
```

Beside the ReviewText column, I created another column called CombinedText, which is joining the Title and ReviewText column together. Because I think there could be some hidden data you can get from the review title as well.

[149]:
```
# general stats for the 4 numeric columns.
df.describe()
```

[149]:

	Age	Rating	RecommendedIND	PositiveFeedbackCount
count	22628.000000	22628.000000	22628.000000	22628.000000
mean	43.282880	4.183092	0.818764	2.631784
std	12.328176	1.115911	0.385222	5.787520
min	18.000000	1.000000	0.000000	0.000000
25%	34.000000	4.000000	1.000000	0.000000
50%	41.000000	5.000000	1.000000	1.000000
75%	52.000000	5.000000	1.000000	3.000000
max	99.000000	5.000000	1.000000	122.000000

[150]:
```
# there is no more NAs
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 22628 entries, 0 to 23485
Data columns (total 9 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Age                    22628 non-null  int64
 1   ReviewText             22628 non-null  object
 2   Rating                 22628 non-null  int64
```

```
 3   RecommendedIND          22628 non-null  int64
 4   PositiveFeedbackCount   22628 non-null  int64
 5   DivisionName            22628 non-null  object
 6   DepartmentName          22628 non-null  object
 7   ClassName               22628 non-null  object
 8   CombinedText            22628 non-null  object
dtypes: int64(4), object(5)
memory usage: 1.7+ MB
```

3.1.3 Export as Pickle

[151]:
```
df.to_pickle('cleaned_df.pkl')
```

Lastly I pickle my cleaned data for further usage.

3.2 Wordcloud Execution for Data

Next thing I do is to create a WordCloud to see what words people are talking/using the most. Before I do that, I need to:

3.2.1 Reading the cleaned data from EDA

[152]:
```
df = pd.read_pickle('cleaned_df.pkl')
df.head()
```

[152]:

	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…

3.3 Clean it a little bit more

Remove some of the less useful frequent words that could exist in the reviews, such as dress, dresses and etc.

[153]:
```
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)
```

[153]:
```
'I  things about  but not .'
```

modify my texts into all lower case

[154]:
```
df['ReviewTextLower'] = df.ReviewText
```

[155]:
```
df['ReviewTextLower'] = df.ReviewTextLower.str.lower()
```

[156]:
```
df['ReviewTextLower'].replace(to_replace=pattern, value='', regex=True,␣
↪inplace=True)
```

3.4 Vectorizer the text data

Then vectorize the text data by using Count and TF-IDF vectorizer.

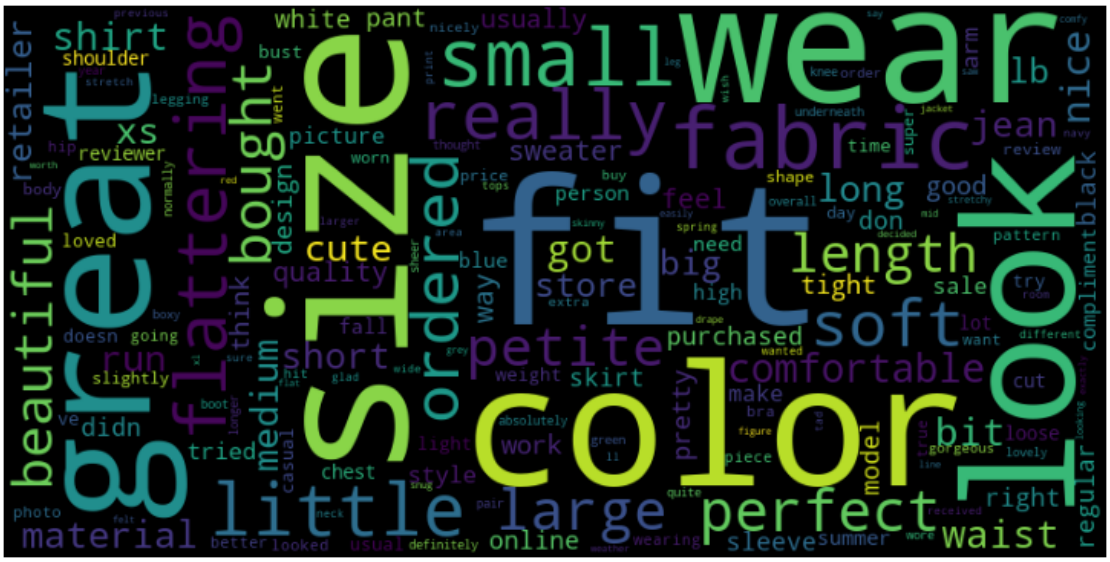
[157]:
```
count_vectorizer = CountVectorizer(ngram_range=(1, 2),
                                   stop_words='english',
                                   token_pattern="\\b[a-z][a-z]+\\b",
                                   lowercase=True,
                                   max_df = 0.6, max_features=4000)
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)
```

the code is basically saying vectorize the text into 1-gram and 2-gram (also tried with 3-gram), using the pre-set 'english' stop words from the package, everything and pattern is in lower case, ignores words that has a frequency of higher than 0.6 from the documents, with a maximum of 4000 features/dimensions.

3.5 Create the word cloud

Then I use the following code to create a WordCloud:

[158]:
```
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=(10, 10), facecolor=None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad=0)

plt.show()
```



3.6 Topic Modelings

There is one more step before I can do topic modeling, which is to use LSA and NMF to reduce the dimension of my input text data.

[159]:
```
# 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)
```

[160]:
```
# 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)
```

[161]:
```
# topic modeling with lsa and tfidf
display_topics2(lsa_tfidf, tfidf_vectorizer.get_feature_names(),8)
```

```
Topic 0
size (0.2144), fit (0.1781), like (0.1731), great (0.1686), wear (0.1614), just
(0.1463), small (0.1424), fabric (0.1303)

Topic 1
great (0.3615), jeans (0.1977), comfortable (0.1872), soft (0.1419), sweater
(0.1143), looks (0.1135), perfect (0.1063), shirt (0.1033)

Topic 2
size (0.4032), true (0.2697), true size (0.2672), great (0.2154), fits (0.1633),
perfect (0.1368), wear (0.1168), comfortable (0.1065)

Topic 3
small (0.3689), shirt (0.2601), wear (0.2385), medium (0.2324), large (0.2092),
cute (0.1412), runs (0.1403), usually (0.1279)

Topic 4
shirt (0.5399), true (0.2559), true size (0.2545), cute (0.2065), size (0.1805),
fits (0.1316), fits true (0.1172), runs (0.0976)

Topic 5
```

sweater (0.5852), beautiful (0.2066), color (0.1938), soft (0.1258), warm (0.0946), fits (0.0896), sleeves (0.0763), coat (0.0734)

Topic 6
great (0.6021), looks (0.2674), looks great (0.1960), small (0.1404), quality (0.1275), sweater (0.1144), look (0.0907), large (0.0881)

Topic 7
sweater (0.3686), cute (0.3173), jeans (0.2810), like (0.1933), super (0.1823), size (0.1545), pants (0.1479), look (0.1368)

Topic 8
shirt (0.3658), sweater (0.2458), petite (0.2109), fit (0.1722), xs (0.1654), ordered (0.1333), length (0.1240), store (0.1208)

Topic 9
cute (0.2987), super (0.2740), soft (0.1985), petite (0.1852), length (0.1619), super cute (0.1460), flattering (0.1443), comfortable (0.1434)

Topic 10
color (0.2369), store (0.2006), cute (0.1486), super (0.1481), fit (0.1479), online (0.1455), tried (0.1315), pants (0.1212)

Topic 11
fits (0.2816), cute (0.2539), skirt (0.1775), looks (0.1499), store (0.1478), petite (0.1452), xs (0.1450), compliments (0.1365)

Topic 12
like (0.3365), fits (0.2841), soft (0.2646), comfortable (0.2201), small (0.1949), perfectly (0.1714), fits perfectly (0.1257), fabric (0.1190)

Topic 13
color (0.2566), petite (0.2014), nice (0.1855), little (0.1573), xs (0.1546), pants (0.1288), summer (0.1138), white (0.1055)

Topic 14
skirt (0.5138), color (0.2585), jeans (0.2100), small (0.1304), ordered (0.1284), shirt (0.1253), blue (0.1065), petite (0.1022)

Topic 15
perfect (0.3231), color (0.3002), cute (0.2463), beautiful (0.2402), just (0.1617), looks (0.1468), jeans (0.1443), fits (0.1166)

Topic 16
comfortable (0.2985), large (0.2661), beautiful (0.2143), runs (0.1502), compliments (0.1487), little (0.1423), pants (0.1371), big (0.1208)

Topic 17

jeans (0.2917), just (0.2764), beautiful (0.1857), wear (0.1770), color (0.1144), flattering (0.1122), pretty (0.1082), right (0.1047)

Topic 18
quality (0.3240), pants (0.2843), nice (0.1965), really (0.1940), fits (0.1656), good (0.1543), got (0.1507), small (0.1495)

Topic 19
jeans (0.3330), quality (0.2657), fabric (0.2171), price (0.1725), good (0.1359), soft (0.1346), sale (0.1239), colors (0.1039)

Topic 20
just (0.4178), small (0.3367), soft (0.1955), right (0.1918), skirt (0.1913), just right (0.1303), look (0.1009), pants (0.0985)

Topic 21
beautiful (0.2570), ordered (0.2147), wear (0.1979), fabric (0.1745), color (0.1351), perfect (0.1343), quality (0.1218), petite (0.1194)

Topic 22
pants (0.2574), look (0.2547), small (0.2354), beautiful (0.2273), flattering (0.2240), cut (0.1158), blouse (0.1014), long (0.0980)

Topic 23
pants (0.3116), large (0.2290), super (0.2196), wear (0.1794), fits (0.1788), comfy (0.1512), perfectly (0.1398), color (0.1197)

Topic 24
material (0.3602), look (0.3192), really (0.2011), flattering (0.1497), pretty (0.1405), color (0.1287), large (0.1143), work (0.0946)

Topic 25
really (0.2771), fabric (0.2460), pants (0.2235), fits (0.2221), large (0.2170), nice (0.2071), ordered (0.1365), sweater (0.1101)

Topic 26
really (0.3313), colors (0.3156), pants (0.2559), beautiful (0.2255), material (0.1769), looks (0.1541), soft (0.1025), person (0.0920)

Topic 27
really (0.2703), got (0.2210), like (0.1901), soft (0.1767), great (0.1470), size (0.1381), compliments (0.1370), large (0.1140)

Topic 28
little (0.2624), beautiful (0.2383), fits (0.1552), look (0.1508), comfortable (0.1387), jeans (0.1162), tried (0.1104), color (0.1060)

Topic 29

comfortable (0.2643), long (0.2238), quality (0.2034), length (0.2030), good (0.1957), medium (0.1680), short (0.1572), color (0.1534)

Topic 30
really (0.2335), recommend (0.2254), xs (0.1845), highly (0.1669), highly recommend (0.1628), flattering (0.1570), big (0.1541), color (0.1337)

Topic 31
material (0.2390), little (0.2047), jacket (0.1722), medium (0.1722), pants (0.1592), nice (0.1579), store (0.1554), tried (0.1425)

Topic 32
nice (0.2224), jacket (0.2142), fit (0.2010), beautiful (0.1964), long (0.1813), sleeves (0.1463), short (0.1426), look (0.1349)

Topic 33
pretty (0.2792), true (0.2245), true size (0.2042), runs (0.1526), got (0.1399), large (0.1353), price (0.1287), pants (0.1252)

Topic 34
look (0.2523), really (0.2151), good (0.2108), soft (0.1917), got (0.1864), waist (0.1430), person (0.1328), looks (0.1221)

Topic 35
recommend (0.3425), highly (0.2676), highly recommend (0.2633), little (0.1851), medium (0.1574), white (0.1557), ordered (0.1301), quality (0.1290)

Topic 36
jacket (0.3243), black (0.2042), white (0.1825), soft (0.1539), pretty (0.1515), got (0.1502), looks (0.1461), blouse (0.1378)

Topic 37
nice (0.3473), pretty (0.2740), size (0.1590), price (0.1490), sale (0.1281), recommend (0.1217), got (0.1115), like (0.1031)

Topic 38
blouse (0.2757), white (0.1914), runs (0.1515), material (0.1392), perfect (0.1327), work (0.1251), black (0.1248), didn (0.1149)

Topic 39
beautiful (0.3043), medium (0.3015), got (0.2026), good (0.1688), xs (0.1597), flattering (0.1580), nice (0.1494), super (0.1422)

Topic 40
price (0.2560), long (0.2263), sale (0.2117), bought (0.2098), big (0.1662), look (0.1448), worth (0.1393), runs (0.1266)

Topic 41

```
work (0.3186), colors (0.2709), got (0.2244), didn (0.1993), waist (0.1365),
think (0.1287), pretty (0.1248), long (0.1212)

Topic 42
pretty (0.3713), retailer (0.2191), sleeves (0.1839), xs (0.1837), long
(0.1511), waist (0.1392), blouse (0.1277), style (0.1030)

Topic 43
bought (0.2942), bit (0.2066), super (0.1792), run (0.1782), does (0.1699),
beautiful (0.1675), work (0.1512), ordered (0.1436)

Topic 44
big (0.2785), bit (0.2227), perfectly (0.1635), super (0.1442), comfy (0.1382),
nice (0.1318), long (0.1219), runs (0.1164)

Topic 45
does (0.2516), got (0.2171), run (0.2029), style (0.1785), way (0.1734), does
run (0.1496), tight (0.1416), good (0.1410)

Topic 46
waist (0.4530), big (0.2635), white (0.1715), bit (0.1395), black (0.1305),
purchased (0.1273), recommend (0.1269), comfortable (0.1192)

Topic 47
short (0.2278), does (0.1580), right (0.1383), retailer (0.1344), nice (0.1310),
run (0.1276), white (0.1269), super (0.1123)

Topic 48
summer (0.2971), long (0.2786), white (0.1781), wearing (0.1738), comfy
(0.1688), way (0.1507), perfectly (0.1295), definitely (0.1209)

Topic 49
length (0.2831), jacket (0.2477), short (0.2222), summer (0.1788), pretty
(0.1770), good (0.1648), shorts (0.1228), perfectly (0.1201)
```

[162]:
```
# topic modeling with lsa and countvectorizer
display_topics2(lsa_cv, count_vectorizer.get_feature_names(),10)
```

```
Topic 0
size (0.3564), fit (0.2672), like (0.2597), wear (0.2188), just (0.1981), great
(0.1858), small (0.1774), fabric (0.1562), color (0.1482), ordered (0.1378)

Topic 1
size (0.7713), small (0.1353), true (0.1065), true size (0.1034), fit (0.0912),
ordered (0.0895), size small (0.0494), large (0.0436), usual (0.0428), runs
(0.0416)
```

Topic 2
like (0.6536), size (0.1369), look (0.1038), really (0.0916), just (0.0752), looked (0.0572), model (0.0542), fabric (0.0524), look like (0.0498), didn (0.0477)

Topic 3
wear (0.5252), small (0.4081), medium (0.1478), usually (0.1055), large (0.1009), runs (0.0783), usually wear (0.0764), shirt (0.0685), fits (0.0608), retailer (0.0528)

Topic 4
fit (0.5434), small (0.4347), medium (0.1521), ordered (0.1254), just (0.1086), xs (0.0929), large (0.0861), usually (0.0784), petite (0.0710), waist (0.0583)

Topic 5
just (0.6473), fabric (0.1600), color (0.1106), beautiful (0.1024), right (0.1016), really (0.0914), flattering (0.0715), soft (0.0674), little (0.0613), length (0.0593)

Topic 6
small (0.4533), great (0.4416), ordered (0.1260), medium (0.1024), looks (0.1019), large (0.0749), runs (0.0739), little (0.0725), look (0.0616), fits (0.0596)

Topic 7
just (0.5746), great (0.2303), like (0.1560), wear (0.1163), jeans (0.1133), right (0.0851), look (0.0625), just right (0.0551), looks (0.0475), fit (0.0435)

Topic 8
color (0.4746), sweater (0.3209), perfect (0.1707), ordered (0.1545), petite (0.1392), xs (0.1187), long (0.0873), blue (0.0803), length (0.0800), material (0.0751)

Topic 9
shirt (0.4906), little (0.3048), really (0.2428), cute (0.2255), look (0.1488), material (0.1359), bit (0.0946), flattering (0.0818), short (0.0773), comfortable (0.0679)

Topic 10
petite (0.3957), length (0.3172), perfect (0.2762), xs (0.1954), waist (0.1659), long (0.1654), regular (0.1484), ordered (0.1381), short (0.1146), skirt (0.1135)

Topic 11
look (0.5776), really (0.3112), sweater (0.1935), wear (0.1043), waist (0.1035), skirt (0.0927), think (0.0691), large (0.0662), didn (0.0648), way (0.0627)

Topic 12

ordered (0.3016), petite (0.2697), really (0.2500), xs (0.2456), wear (0.2121), great (0.1968), shirt (0.1418), tried (0.0887), regular (0.0817), looks (0.0782)

Topic 13
sweater (0.3795), little (0.3693), really (0.3138), large (0.2066), bit (0.1216), soft (0.0919), fit (0.0784), medium (0.0773), runs (0.0711), nice (0.0600)

Topic 14
sweater (0.4703), shirt (0.2984), large (0.2696), ordered (0.1988), look (0.1611), soft (0.1579), fabric (0.1296), sleeves (0.1025), medium (0.0893), long (0.0876)

Topic 15
little (0.6163), color (0.2538), large (0.1853), look (0.1522), bit (0.1491), waist (0.0943), wear (0.0561), big (0.0552), runs (0.0547), think (0.0475)

Topic 16
store (0.3558), bought (0.2984), tried (0.2850), xs (0.2020), retailer (0.1534), sweater (0.1465), flattering (0.1319), saw (0.1266), online (0.1181), soft (0.1082)

Topic 17
skirt (0.3949), beautiful (0.2535), large (0.2505), flattering (0.2438), fits (0.1950), material (0.1917), waist (0.1643), colors (0.1245), medium (0.1118), perfect (0.1038)

Topic 18
perfect (0.4385), large (0.4098), really (0.2131), medium (0.1792), store (0.1425), fabric (0.1171), bought (0.1142), tried (0.0888), length (0.0805), great (0.0733)

Topic 19
ordered (0.4733), comfortable (0.3013), large (0.2286), cute (0.2242), soft (0.2234), flattering (0.1950), pants (0.1574), material (0.1503), super (0.1487), jeans (0.1081)

Topic 20
large (0.3714), nice (0.3262), bit (0.2549), bought (0.2220), pants (0.1918), long (0.1521), soft (0.1447), length (0.1425), petite (0.1386), waist (0.1234)

Topic 21
skirt (0.4195), jeans (0.3905), looks (0.2981), cute (0.2666), waist (0.2521), sweater (0.1700), black (0.0999), store (0.0932), bit (0.0760), skinny (0.0724)

Topic 22
nice (0.4390), quality (0.1808), bought (0.1575), fits (0.1550), ordered (0.1500), black (0.1459), jeans (0.1450), beautiful (0.1309), colors (0.1274),

white (0.1256)

Topic 23
looks (0.4123), fits (0.3395), cute (0.2416), nice (0.1840), flattering (0.1677), xs (0.1346), bought (0.1288), look (0.1242), perfectly (0.1169), true (0.0922)

Topic 24
flattering (0.4077), beautiful (0.2046), bought (0.2021), jeans (0.1928), ordered (0.1842), pants (0.1718), long (0.1471), really (0.1219), black (0.1131), fits (0.1117)

Topic 25
nice (0.4950), flattering (0.3710), perfect (0.1635), cut (0.1629), material (0.1362), tried (0.1196), store (0.1088), medium (0.0852), didn (0.0819), looks (0.0817)

Topic 26
jeans (0.4124), beautiful (0.3522), looks (0.2402), soft (0.2389), material (0.1819), bit (0.1533), tried (0.1016), skinny (0.0786), large (0.0644), skinny jeans (0.0615)

Topic 27
bit (0.4526), cute (0.3275), pants (0.3128), got (0.1670), quality (0.1276), beautiful (0.1203), long (0.1146), think (0.1041), short (0.0983), fits (0.0904)

Topic 28
looks (0.3555), long (0.2689), length (0.2420), bought (0.2371), material (0.2205), good (0.1538), comfortable (0.1354), beautiful (0.1324), pants (0.1227), short (0.1214)

Topic 29
fits (0.3487), medium (0.2640), jeans (0.1936), cute (0.1830), got (0.1390), perfectly (0.1382), long (0.1382), store (0.1352), length (0.1295), tried (0.1272)

Topic 30
beautiful (0.2981), xs (0.2966), pants (0.2686), got (0.2141), large (0.1920), quality (0.1675), petite (0.1557), runs (0.1515), sweater (0.1341), big (0.1044)

Topic 31
pants (0.3819), looks (0.3347), comfortable (0.2875), got (0.1449), fits (0.1325), tried (0.1314), shirt (0.1297), store (0.1176), waist (0.1135), medium (0.1068)

Topic 32
skirt (0.3138), comfortable (0.2620), bit (0.2381), colors (0.1702), beautiful (0.1388), black (0.1260), petite (0.1201), nice (0.1197), got (0.1126), length

(0.0974)

Topic 33
beautiful (0.4759), nice (0.2536), cute (0.2337), pants (0.1886), colors (0.1522), waist (0.1103), tried (0.1100), store (0.0955), fits (0.0785), super (0.0721)

Topic 34
got (0.3977), waist (0.2587), comfortable (0.2253), work (0.1950), nice (0.1804), long (0.1745), colors (0.1341), pretty (0.1323), sleeves (0.1196), didn (0.1190)

Topic 35
comfortable (0.5935), quality (0.2714), good (0.1356), length (0.1254), bit (0.1046), bought (0.1016), tried (0.0943), waist (0.0888), high (0.0747), nice (0.0673)

Topic 36
long (0.4018), xs (0.3703), retailer (0.3415), sleeves (0.2067), usually (0.1728), skirt (0.1493), true (0.1110), arms (0.1041), tops (0.0971), true size (0.0941)

Topic 37
think (0.4119), colors (0.3081), pretty (0.2877), true (0.2858), true size (0.2413), don (0.2082), runs (0.1487), large (0.1411), petite (0.1318), jeans (0.0944)

Topic 38
length (0.4757), colors (0.3322), waist (0.2438), good (0.1768), got (0.1530), quality (0.1190), retailer (0.0810), soft (0.0779), blue (0.0754), online (0.0721)

Topic 39
black (0.3604), length (0.3202), work (0.2963), white (0.2467), material (0.2168), large (0.1398), didn (0.1302), true (0.0964), sweater (0.0948), xs (0.0872)

Topic 40
medium (0.3897), colors (0.3367), work (0.2724), retailer (0.2153), petite (0.2077), waist (0.1656), purchased (0.1473), cut (0.1380), comfortable (0.1123), quality (0.1050)

Topic 41
medium (0.3829), think (0.3729), length (0.2827), usually (0.2373), don (0.2105), xs (0.1432), big (0.1376), work (0.1294), beautiful (0.1194), retailer (0.1040)

Topic 42

```
black (0.3916), soft (0.2393), white (0.2383), quality (0.2003), beautiful
(0.1910), waist (0.1629), jacket (0.1123), true (0.1088), think (0.1052), medium
(0.0944)

Topic 43
material (0.3641), medium (0.2375), true (0.2243), true size (0.1956), waist
(0.1606), comfortable (0.1604), fabric (0.1295), tried (0.1137), black (0.1128),
beautiful (0.1075)

Topic 44
true (0.3791), true size (0.3253), work (0.1946), big (0.1540), medium (0.1482),
tried (0.1432), bought (0.1401), got (0.1322), didn (0.1295), good (0.1250)

Topic 45
retailer (0.3432), short (0.2620), true (0.2079), true size (0.1792), online
(0.1585), got (0.1553), right (0.1431), compliments (0.1081), nice (0.1076), ve
(0.0929)

Topic 46
pretty (0.6701), retailer (0.1751), petite (0.1391), did (0.1027), usually
(0.0979), got (0.0824), big (0.0808), bought (0.0797), sale (0.0794), blouse
(0.0792)

Topic 47
way (0.5312), big (0.5179), retailer (0.1582), cut (0.1572), runs (0.1147),
sleeves (0.1135), fits (0.0885), black (0.0807), arms (0.0785), soft (0.0781)

Topic 48
jacket (0.3593), cut (0.2581), sleeves (0.2268), tight (0.1806), right (0.1329),
arms (0.1285), body (0.1044), pants (0.0998), bought (0.0989), ve (0.0883)

Topic 49
jacket (0.5111), short (0.3067), black (0.2806), pretty (0.2385), big (0.1571),
does (0.1266), worn (0.0936), fall (0.0794), perfectly (0.0713), material
(0.0687)
```

[163]:
```
# topic modeling with nmf and tfidf
display_topics2(nmf_tfidf, tfidf_vectorizer.get_feature_names(),10)
```

```
Topic 0
fabric (12.7493), soft (2.0676), fabric soft (2.0094), design (0.8667), fabric
nice (0.8468), soft fabric (0.8302), heavy (0.7223), light (0.7198), quality
fabric (0.6972), style (0.6456)

Topic 1
wear (9.3299), usually wear (1.6612), wait (1.4259), easy (1.3125), wait wear
(1.2325), usually (1.1733), easy wear (1.0562), normally wear (0.8768), wear
```

size (0.8420), normally (0.6644)

Topic 2
true (3.4824), true size (3.3670), size (1.8984), fits true (1.1880), fit true (0.7309), runs true (0.5530), runs (0.2810), fits (0.2754), run true (0.2161), size great (0.1502)

Topic 3
small (4.7533), medium (1.7486), size small (0.7176), small medium (0.6751), runs small (0.5592), extra (0.5503), extra small (0.5392), usually (0.5303), wear small (0.4475), small fit (0.4302)

Topic 4
shirt (4.8714), great shirt (0.2303), shirts (0.2253), like shirt (0.1970), cute shirt (0.1948), boxy (0.1866), bought shirt (0.1814), shirt runs (0.1737), shirt great (0.1732), shirt fits (0.1713)

Topic 5
sweater (5.2311), warm (0.5705), soft (0.4807), cozy (0.3882), coat (0.3787), itchy (0.3555), sweaters (0.2873), knit (0.2871), winter (0.2841), sleeves (0.2640)

Topic 6
great (4.4934), looks great (0.4599), fit great (0.4151), great fit (0.4126), fits great (0.4021), great quality (0.3317), look great (0.3126), fall (0.2753), summer (0.2217), great length (0.2141)

Topic 7
jeans (5.9191), skinny (1.7820), skinny jeans (1.5035), pair (1.1919), leggings (0.9191), pair jeans (0.5222), worn (0.5201), pilcro (0.4918), boots (0.4536), great jeans (0.4491)

Topic 8
long (3.2352), short (2.3109), sleeves (1.7867), length (1.3386), torso (0.7682), arms (0.5712), long torso (0.4594), sleeve (0.4265), longer (0.3829), tall (0.3794)

Topic 9
cute (6.6370), super cute (1.1749), really cute (0.8120), cute comfortable (0.4763), design (0.3647), looks cute (0.3544), comfortable cute (0.3024), cute flattering (0.2941), cute design (0.2846), look cute (0.2374)

Topic 10
ordered (5.0901), ordered size (0.8717), reviews (0.8100), ordered small (0.6818), online (0.4919), ordered medium (0.4656), ordered online (0.4463), size ordered (0.4061), ordered xs (0.3972), did (0.3866)

Topic 11

fits (5.7937), perfectly (2.7996), fits perfectly (2.5452), fits great (0.9330), size fits (0.8092), small fits (0.7350), fits like (0.5426), fits true (0.5248), fit perfectly (0.4841), fits perfect (0.4787)

Topic 12
like (5.2442), feel (1.0319), feel like (0.7884), looked (0.6901), looked like (0.5486), don (0.5282), felt (0.4908), felt like (0.4612), wearing (0.4555), look like (0.4161)

Topic 13
fit (6.3363), fit perfectly (1.3819), perfectly (1.0881), size fit (0.6628), fit great (0.6248), great fit (0.5328), fit perfect (0.4857), better (0.4280), small fit (0.4033), fit like (0.3946)

Topic 14
skirt (4.1759), pencil (0.2438), skirts (0.2273), pencil skirt (0.2002), beautiful skirt (0.1890), lined (0.1615), knee (0.1386), skirt just (0.1348), pattern (0.1276), skirt great (0.1252)

Topic 15
color (4.6273), blue (0.8547), green (0.8204), pink (0.5387), red (0.4499), green color (0.3908), blue color (0.3564), orange (0.3340), color beautiful (0.3098), beautiful color (0.2780)

Topic 16
large (4.7641), runs (2.3514), runs large (1.7383), medium (0.9410), little large (0.4116), size large (0.4085), large size (0.3988), medium large (0.3650), bit large (0.3633), large usually (0.3373)

Topic 17
perfect (4.2476), summer (1.2121), fall (0.6582), fit perfect (0.5879), length (0.5645), perfect summer (0.5114), length perfect (0.4751), spring (0.4619), perfect length (0.3671), perfect fit (0.3654)

Topic 18
quality (5.5739), good (3.4252), good quality (1.7743), high (1.2399), high quality (1.0909), great quality (1.0058), quality fabric (0.5879), quality material (0.5017), quality good (0.4399), poor (0.4230)

Topic 19
white (3.1656), black (2.9518), blue (0.8874), black white (0.6754), purchased (0.5386), navy (0.4705), red (0.4641), tee (0.4533), lace (0.3958), sheer (0.3156)

Topic 20
just (3.5873), right (1.4094), just right (0.9389), fit just (0.2730), just didn (0.2253), just like (0.1972), fits just (0.1625), wasn (0.1489), just little (0.1465), length just (0.1398)

Topic 21
little (4.9703), little big (0.5186), runs little (0.5123), little large (0.4707), little bit (0.4159), little tight (0.3414), just little (0.3410), run little (0.3019), little snug (0.3018), longer (0.2965)

Topic 22
look (4.4811), look like (0.7894), make (0.6384), makes (0.6348), look great (0.5982), look good (0.4805), make look (0.4578), makes look (0.4273), doesn (0.3591), good (0.3538)

Topic 23
comfortable (5.0190), soft comfortable (1.2015), soft (0.7399), easy (0.4063), super comfortable (0.4033), comfortable flattering (0.3885), stylish (0.3395), casual (0.3240), cute comfortable (0.3082), comfortable wear (0.2677)

Topic 24
material (3.8145), soft (1.6726), material soft (0.9026), soft material (0.3040), quality material (0.2462), soft comfortable (0.2083), great material (0.2046), feels (0.1945), design (0.1873), material nice (0.1719)

Topic 25
nice (4.3984), fabric nice (0.4982), weight (0.4656), really nice (0.3984), nice quality (0.3407), nice weight (0.3147), color nice (0.2695), nice fabric (0.2595), material nice (0.2204), light (0.1955)

Topic 26
really (3.7420), really like (0.5215), really cute (0.5075), really wanted (0.4625), wanted (0.4602), really nice (0.2979), really pretty (0.2820), really liked (0.2292), liked (0.2112), really soft (0.1628)

Topic 27
looks (4.7916), looks great (1.8095), looks like (0.8395), model (0.7239), person (0.6839), better (0.5967), picture (0.5218), looks better (0.4786), looks good (0.3954), looks cute (0.3236)

Topic 28
pants (4.0125), pair (0.6722), pants fit (0.3262), pilcro (0.2471), stretch (0.2129), legs (0.2078), leg (0.2028), pants great (0.1978), pair pants (0.1717), pants comfortable (0.1683)

Topic 29
xs (3.2120), ordered xs (0.5165), lbs (0.5105), xxs (0.4932), usually (0.4032), size xs (0.3821), wear xs (0.3733), xs fit (0.3549), xs petite (0.3238), got xs (0.2893)

Topic 30
size (3.4405), usual (0.6089), usual size (0.5336), smaller (0.4514), normal

(0.3859), size small (0.3819), order (0.3754), normal size (0.3513), larger (0.3270), wear size (0.3200)

Topic 31
store (2.5762), tried (2.0755), online (1.5247), saw (1.3169), loved (0.9413), try (0.9053), tried store (0.7408), looked (0.6455), person (0.4764), saw store (0.4502)

Topic 32
flattering (3.3142), figure (0.4218), comfortable flattering (0.2717), fit flattering (0.2695), super flattering (0.2605), figure flattering (0.2557), flattering fit (0.2529), shape (0.2347), body (0.1782), flattering comfortable (0.1752)

Topic 33
pretty (3.9536), really pretty (0.4591), design (0.2851), lace (0.2842), color pretty (0.2010), feminine (0.1835), pretty color (0.1739), print (0.1678), pattern (0.1588), pretty person (0.1391)

Topic 34
bought (2.4581), colors (1.9054), wish (0.5061), bought colors (0.3408), buy (0.3027), came (0.1967), bought size (0.1883), vibrant (0.1791), wish came (0.1769), different (0.1601)

Topic 35
recommend (2.3284), highly (1.7512), highly recommend (1.7129), definitely (0.4446), definitely recommend (0.3613), recommend sizing (0.0985), absolutely (0.0972), versatile (0.0813), dressed (0.0806), sizing (0.0790)

Topic 36
price (2.1463), sale (1.8297), worth (1.0496), sale price (0.7848), worth price (0.5533), got (0.3861), definitely (0.2819), got sale (0.2623), think (0.2390), buy (0.2149)

Topic 37
jacket (3.0771), denim (0.4185), worn (0.3533), denim jacket (0.2921), leather (0.2432), warm (0.2192), piece (0.2188), fall (0.2160), versatile (0.2108), jean jacket (0.2069)

Topic 38
blouse (3.9249), gorgeous (0.3479), lovely (0.2931), feminine (0.2261), sleeves (0.2211), design (0.2204), sheer (0.2158), tight (0.2034), xl (0.1965), beautiful blouse (0.1906)

Topic 39
compliments (2.3518), received (1.4175), wore (1.3454), got (1.1244), received compliments (0.9825), time (0.7521), got compliments (0.4326), ve (0.4082), lots compliments (0.4065), lots (0.4043)

Topic 40
beautiful (4.3709), absolutely (0.4729), person (0.3759), beautiful color (0.3505), absolutely beautiful (0.3491), beautiful person (0.3346), color beautiful (0.3197), embroidery (0.2998), design (0.2468), colors beautiful (0.2419)

Topic 41
super (2.8365), comfy (1.1666), super soft (0.8911), super cute (0.8499), soft (0.6047), super comfy (0.5939), super comfortable (0.3656), super flattering (0.2986), soft comfy (0.2324), fabric super (0.1906)

Topic 42
does (1.9067), run (1.6917), does run (1.2185), run large (0.6338), run small (0.3598), run little (0.3305), large (0.2385), run big (0.2157), run bit (0.1589), does model (0.1446)

Topic 43
cut (2.5549), bra (1.1865), low (0.9222), tank (0.5450), underneath (0.4363), low cut (0.4307), cami (0.3959), need (0.3901), arm (0.3797), holes (0.3515)

Topic 44
big (2.8202), way (0.8987), runs (0.5723), little big (0.5525), runs big (0.4778), way big (0.4488), return (0.2482), looked (0.2279), arm (0.1821), holes (0.1752)

Topic 45
work (2.6072), didn (1.2234), didn work (0.5905), wanted (0.4046), wear work (0.3807), just didn (0.3283), did (0.2600), did work (0.2146), appropriate (0.1919), unfortunately (0.1793)

Topic 46
waist (2.7728), hips (0.8136), high (0.4776), tight (0.2748), elastic (0.2580), waist hips (0.2539), bust (0.2332), fitted (0.2114), area (0.2092), legs (0.2087)

Topic 47
bit (2.7600), little bit (0.3444), bit large (0.2820), runs bit (0.2315), quite (0.1930), tiny bit (0.1913), tiny (0.1894), bit longer (0.1869), bit long (0.1865), think (0.1683)

Topic 48
petite (2.6785), regular (1.1168), length (0.6987), regular size (0.3442), petite size (0.3388), ordered petite (0.3002), xs petite (0.2248), xxs (0.2137), knee (0.2098), sizes (0.2004)

Topic 49
retailer (2.7302), tops (0.6899), purchased (0.4995), usually (0.4530), retailer

tops (0.3874), favorite (0.3752), ve (0.3733), local retailer (0.3666), local (0.3627), time (0.2171)

[164]:
```
# topic modeling with nmf and countvectorizer
display_topics2(nmf_cv, count_vectorizer.get_feature_names(),10)
```

```
Topic 0
great (34.8864), looks great (2.3692), fit great (1.6852), fall (1.4489), look great (1.3142), summer (1.2223), fits great (1.1079), great fit (1.0296), great quality (0.9590), piece (0.9496)

Topic 1
size (9.2205), usual (0.5581), usual size (0.4689), smaller (0.4022), size small (0.3911), normal (0.3695), wear size (0.3604), size fit (0.3303), normal size (0.3255), ordered size (0.3177)

Topic 2
like (8.0721), feel (0.5839), feel like (0.4267), looked (0.3568), look like (0.3406), looked like (0.2860), model (0.2844), felt (0.2619), looks like (0.2596), really like (0.2455)

Topic 3
wear (11.3353), usually wear (1.0138), usually (0.7694), wait (0.5115), wear size (0.4973), normally wear (0.4959), easy (0.4451), bra (0.4299), wait wear (0.4191), underneath (0.4191)

Topic 4
fit (12.3894), fit perfectly (1.0444), perfectly (1.0345), size fit (0.5042), fit great (0.4127), better (0.3945), loose (0.3825), fit like (0.3422), fit perfect (0.3185), small fit (0.3079)

Topic 5
just (10.2839), right (1.4699), just right (0.8463), just didn (0.2985), fit just (0.2876), just like (0.2376), didn (0.2311), wasn (0.1960), feel (0.1697), just little (0.1682)

Topic 6
small (13.1752), size small (1.2311), runs (0.9961), extra (0.9297), extra small (0.7419), runs small (0.7109), small fit (0.6188), ordered small (0.5817), usually (0.5098), wear small (0.5013)

Topic 7
fabric (10.8660), fabric soft (0.5408), design (0.4070), light (0.3719), fabric nice (0.3051), pattern (0.3017), heavy (0.3004), weight (0.2861), style (0.2716), sheer (0.2424)

Topic 8
```

color (12.8771), blue (1.2585), green (1.0175), pink (0.6964), red (0.6449), orange (0.4829), style (0.4364), blue color (0.4201), light (0.4178), green color (0.3886)

Topic 9
shirt (12.6919), shirts (0.4138), boxy (0.2880), white (0.2793), underneath (0.2788), like shirt (0.2646), bought shirt (0.2601), great shirt (0.2262), tee (0.2102), white shirt (0.1921)

Topic 10
petite (10.9768), regular (3.0686), petite size (0.7465), xxs (0.6757), regular size (0.6698), ordered petite (0.6191), xs petite (0.6175), sizes (0.6084), sizing (0.4538), order (0.4324)

Topic 11
look (11.4232), look like (1.0984), make (0.9025), makes (0.7642), good (0.6308), look great (0.6266), make look (0.5524), doesn (0.5275), look good (0.4998), model (0.4421)

Topic 12
perfect (16.6837), summer (2.3718), fit perfect (1.2507), fall (1.1063), length perfect (0.8516), spring (0.7322), perfect summer (0.7145), perfect fit (0.6884), light (0.6654), weight (0.6650)

Topic 13
really (14.0695), really like (0.8697), wanted (0.8211), really cute (0.8160), really wanted (0.7414), really nice (0.5677), really pretty (0.4360), liked (0.3674), really liked (0.3515), really flattering (0.2286)

Topic 14
sweater (11.7549), warm (0.7064), coat (0.5543), knit (0.4986), sweaters (0.4707), winter (0.4575), cozy (0.4351), fall (0.4127), weight (0.3961), itchy (0.3729)

Topic 15
little (9.1460), little big (0.4413), little bit (0.4356), runs little (0.3771), longer (0.3452), little large (0.3431), wish (0.2959), snug (0.2894), just little (0.2777), run little (0.2759)

Topic 16
store (8.2355), tried (6.9518), online (3.8823), saw (3.2332), try (2.4271), retailer (2.1058), looked (1.4670), loved (1.3681), didn (1.3122), tried store (1.2684)

Topic 17
skirt (17.7828), skirts (0.7961), pencil (0.6590), lined (0.5682), knee (0.5286), pencil skirt (0.5138), hips (0.4853), lining (0.4262), denim (0.3927), does (0.3879)

Topic 18
large (11.1690), runs (2.9409), runs large (1.5918), run (1.1677), does (0.9095), run large (0.6809), size large (0.6370), does run (0.6332), normally (0.5116), bit large (0.5031)

Topic 19
ordered (15.4642), ordered size (1.3427), online (1.1540), reviews (1.0595), ordered small (0.9386), ordered xs (0.6445), ordered online (0.5828), ordered medium (0.5823), size ordered (0.5336), ordered usual (0.5317)

Topic 20
soft (15.7684), super (2.2581), super soft (1.7641), fabric soft (1.4255), soft comfortable (1.2563), material soft (0.9786), comfy (0.8693), warm (0.6220), cozy (0.5483), soft comfy (0.5389)

Topic 21
cute (11.9467), super (3.1398), super cute (1.8020), really cute (0.9518), comfy (0.5708), design (0.4410), looks cute (0.3372), cute comfortable (0.2827), price (0.2776), style (0.2617)

Topic 22
nice (20.5965), weight (1.2192), fabric nice (1.1482), really nice (1.0568), nice weight (0.7183), color nice (0.6441), nice quality (0.6270), fall (0.5855), touch (0.5517), material nice (0.5272)

Topic 23
looks (13.2200), looks great (2.4905), model (1.8390), looks like (1.5919), person (1.3105), better (1.0967), picture (0.8880), good (0.7119), does (0.6838), online (0.6583)

Topic 24
flattering (13.0092), figure (0.7945), shape (0.6391), super (0.5363), super flattering (0.5064), fit flattering (0.4745), flattering fit (0.4589), recommend (0.4239), body (0.4222), comfortable flattering (0.3938)

Topic 25
cut (8.4534), low (1.3272), bra (1.0892), low cut (0.6027), chest (0.5826), shoulders (0.5672), cut flattering (0.4319), neck (0.3938), high (0.3386), body (0.3220)

Topic 26
jeans (10.7472), skinny (1.9661), pair (1.7311), skinny jeans (1.4893), leggings (0.8121), worn (0.7827), pilcro (0.6860), denim (0.6468), stretch (0.6205), ag (0.5401)

Topic 27
bit (12.1306), little bit (0.8211), bit large (0.5526), quite (0.4916), loose

(0.4294), longer (0.4120), tiny (0.4009), tiny bit (0.3830), runs bit (0.3799), bit long (0.3757)

Topic 28
bought (14.4983), sale (0.8128), bought size (0.6786), pair (0.5716), buy (0.5529), wore (0.4883), green (0.4519), bought small (0.4436), ve (0.4239), compliments (0.4203)

Topic 29
fits (11.6705), perfectly (3.8350), fits perfectly (2.6554), size fits (0.9196), small fits (0.8090), fits great (0.7804), fits like (0.7766), fits true (0.7119), nicely (0.5661), fit perfectly (0.5635)

Topic 30
beautiful (13.2305), absolutely (0.8180), design (0.7002), person (0.6320), color beautiful (0.5862), blouse (0.5779), gorgeous (0.5714), lace (0.5146), beautiful color (0.4889), absolutely beautiful (0.4865)

Topic 31
pants (11.1882), pair (1.6299), retailer (0.7063), run (0.6578), stretch (0.4697), pilcro (0.4474), pants fit (0.4189), legs (0.4117), ankle (0.3660), leg (0.3613)

Topic 32
waist (9.4568), hips (1.5933), high (0.9186), waist hips (0.5210), fitted (0.4768), elastic (0.4728), tight (0.4343), right (0.4086), bust (0.3659), belt (0.3639)

Topic 33
quality (7.5487), good (4.1632), high (1.6476), price (1.4778), good quality (1.0977), high quality (0.9801), retailer (0.9199), design (0.6397), worth (0.5803), great quality (0.5689)

Topic 34
got (10.4267), compliments (1.5793), sale (1.2122), wore (1.2106), time (0.6505), got compliments (0.5940), price (0.5079), got size (0.4827), got small (0.4646), got sale (0.4342)

Topic 35
comfortable (9.3773), soft comfortable (0.8879), compliments (0.5540), wearing (0.5428), super comfortable (0.5354), easy (0.4389), feel (0.4069), casual (0.3813), comfortable flattering (0.3645), super (0.3504)

Topic 36
long (9.7346), torso (0.7051), sleeves (0.5681), long torso (0.5507), sleeve (0.4517), time (0.4384), tall (0.3728), legs (0.3713), sleeves long (0.3473), bit long (0.3443)

Topic 37
think (9.6051), don (4.5996), don think (1.4107), price (0.6666), sale (0.5241), need (0.4614), ll (0.4446), better (0.4273), definitely (0.4015), lot (0.3744)

Topic 38
length (10.6573), regular (0.7751), longer (0.7493), length perfect (0.6891), shorter (0.6510), knee (0.6503), right (0.5835), perfect length (0.5783), model (0.5640), good (0.4653)

Topic 39
material (11.3521), material soft (0.9600), design (0.5626), light (0.4970), cheap (0.3285), quality material (0.3192), material nice (0.3027), stretchy (0.2673), thought (0.2492), thicker (0.2449)

Topic 40
colors (10.8657), wish (0.8964), person (0.7820), vibrant (0.7603), blue (0.7162), different (0.5873), gorgeous (0.4939), came (0.4543), style (0.4195), colors vibrant (0.4184)

Topic 41
medium (10.7927), usually (2.4547), small medium (1.0507), size medium (0.9526), retailer (0.9257), usually wear (0.8537), ordered medium (0.7899), wear medium (0.7825), medium fit (0.7725), tops (0.7592)

Topic 42
black (6.9137), white (5.3938), blue (1.4114), purchased (1.1111), black white (0.7450), blouse (0.7317), navy (0.6278), red (0.5885), lace (0.5312), sheer (0.4934)

Topic 43
work (7.8856), didn (3.3356), did (0.9717), didn work (0.8914), wanted (0.8546), wear work (0.6078), just didn (0.5313), make (0.3445), appropriate (0.3414), did work (0.3403)

Topic 44
true (7.5866), true size (6.6722), size (3.9410), fits true (1.3017), fit true (1.1486), runs (0.9411), runs true (0.8067), say (0.2799), run true (0.2713), retailer (0.2359)

Topic 45
pretty (7.2378), blouse (0.6267), really pretty (0.4848), lace (0.4229), does (0.2829), bra (0.2519), person (0.2369), color pretty (0.2248), feminine (0.1913), sheer (0.1768)

Topic 46
way (7.4867), big (6.6669), looked (1.2183), way big (0.8193), runs (0.7831), going (0.6671), little big (0.5577), runs big (0.4487), loved (0.4211), return (0.3703)

```
Topic 47
xs (8.8920), usually (1.5567), lbs (0.9629), xxs (0.9023), retailer (0.8884),
ordered xs (0.7556), size xs (0.6384), wear xs (0.6028), xs petite (0.5523), xs
fit (0.5405)

Topic 48
sleeves (5.6095), jacket (4.5127), arms (2.3345), tight (1.7070), shoulders
(0.9661), body (0.7140), blouse (0.5542), fitted (0.3919), denim (0.3539), worn
(0.3145)

Topic 49
short (7.0804), torso (0.7333), waisted (0.5129), wide (0.4270), short waisted
(0.3947), short torso (0.3763), tall (0.3632), little short (0.3124), boxy
(0.3018), longer (0.2942)
```

We can generate different amount of topics, by testing with different numbers of topics to find the best number, and see if those topics make sense to you.

3.7 Clustering of Reviews

3.8 Standard Scaled: Non-Negative Matrix Factorization - Term Frequency–InverseDocument Frequency

3.8.1 2-grams Tokenization

[165]:
```
# 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)
```

[166]:
```
# 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)
```

It is better to standardize your input data to mean of 0 and standard deviation of 1 before you run clustering algorithms. Because your features might not all be on the same scale, on the other words, that might not be the same thing as increasing 1 unit from feature a comparing to increasing 1 unit from feature b.

[167]:
```
# 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)
```

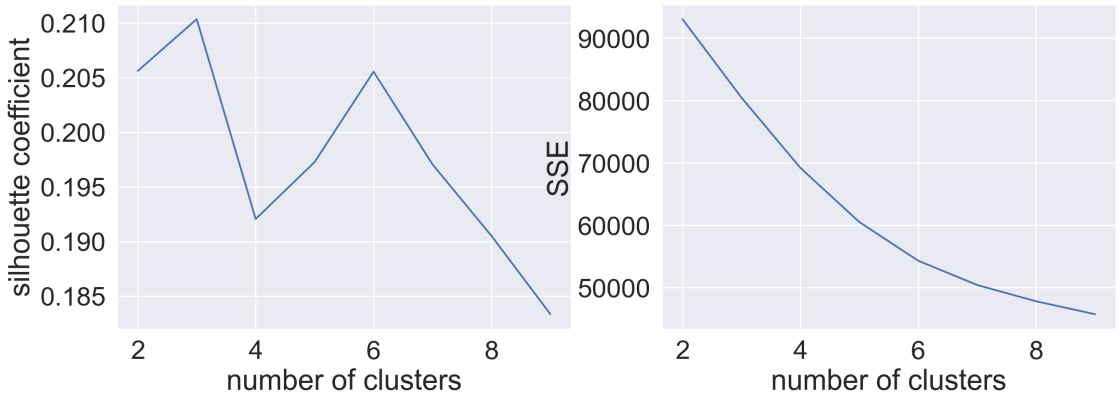
[168]:
```
display_topics2(lsa_tfidf, tfidf_vectorizer.get_feature_names(),8)
```

```
Topic 0
size (0.2144), fit (0.1781), like (0.1731), great (0.1686), wear (0.1614), just
(0.1463), small (0.1424), fabric (0.1303)

Topic 1
great (0.3693), jeans (0.1888), comfortable (0.1876), soft (0.1465), sweater
(0.1139), looks (0.1134), perfect (0.1063), shirt (0.1027)

Topic 2
size (0.4030), true (0.2633), true size (0.2608), great (0.1954), fits (0.1628),
perfect (0.1355), wear (0.1227), jeans (0.1209)

Topic 3
small (0.3684), medium (0.2491), shirt (0.2457), wear (0.2436), large (0.2203),
runs (0.1332), usually (0.1268), cute (0.1143)

Topic 4
shirt (0.5071), true (0.2406), true size (0.2380), cute (0.1987), size (0.1415),
fits (0.1320), fits true (0.1130), nice (0.1016)
```

Then you can use unsupervised machine learning algorithm to make clusters for different topics or different types of reviews. In this project, I used KMeans, and also used inertia and silhouette scores as proxy to help me identify what is the best number of clusters I should use. Then using TSNE to help me visualize the clusters generated.

[169]:
```
SSEs = []
Sil_coefs = []
for k in range(2,10):
    km = KMeans(n_clusters=k, random_state=42)
    km.fit(lsa_tfidf_data_sclaed)
    labels = km.labels_
    Sil_coefs.append(silhouette_score(lsa_tfidf_data_sclaed, labels,␣
 ↪metric='euclidean'))
    SSEs.append(km.inertia_)
```

[170]:
```
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');
```



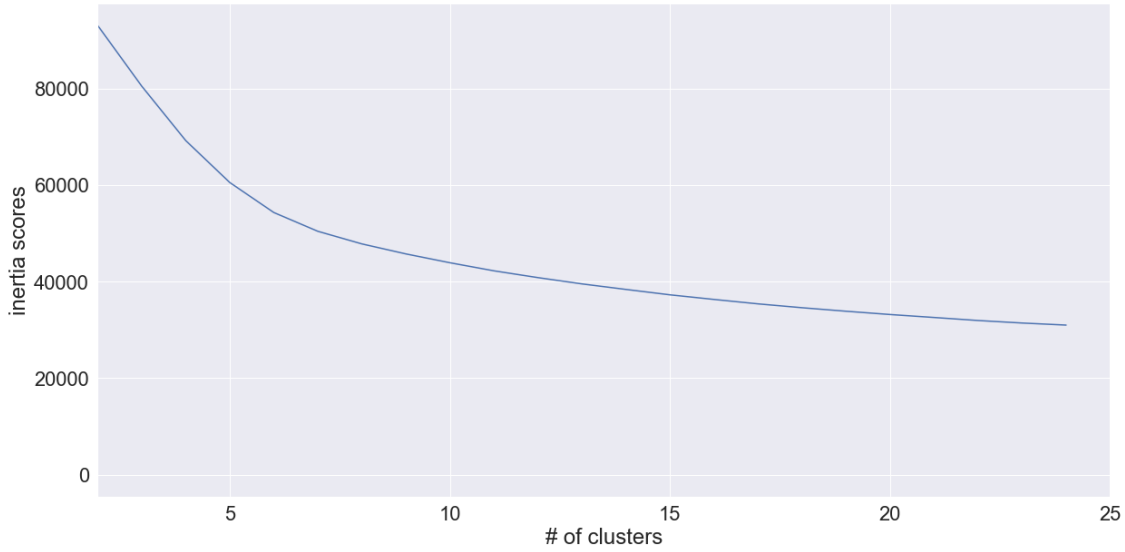
[171]:
```
inertia = [0,0]

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

```
# clusters:  2   Inertia: 93058.838693
# clusters:  3   Inertia: 80489.830025
# clusters:  4   Inertia: 69210.680182
```

```
# clusters:  5   Inertia: 60528.570877
# clusters:  6   Inertia: 54316.137949
# clusters:  7   Inertia: 50427.318115
# clusters:  8   Inertia: 47815.051992
# clusters:  9   Inertia: 45741.110852
# clusters: 10   Inertia: 43909.446494
# clusters: 11   Inertia: 42235.203859
# clusters: 12   Inertia: 40827.591612
# clusters: 13   Inertia: 39526.793406
# clusters: 14   Inertia: 38378.705178
# clusters: 15   Inertia: 37261.677344
# clusters: 16   Inertia: 36305.716985
# clusters: 17   Inertia: 35390.446504
# clusters: 18   Inertia: 34588.347502
# clusters: 19   Inertia: 33859.839891
# clusters: 20   Inertia: 33182.991687
# clusters: 21   Inertia: 32565.996098
# clusters: 22   Inertia: 31940.361807
# clusters: 23   Inertia: 31419.387674
# clusters: 24   Inertia: 30991.782242
```

[172]:
```
plt.figure(figsize=(20,10))
plt.plot(inertia)
plt.xlabel('# of clusters')
plt.xlim((2,25))
plt.ylabel('inertia scores')
#plt.ylim((650,1200))
```

[172]:
```
Text(0, 0.5, 'inertia scores')
```



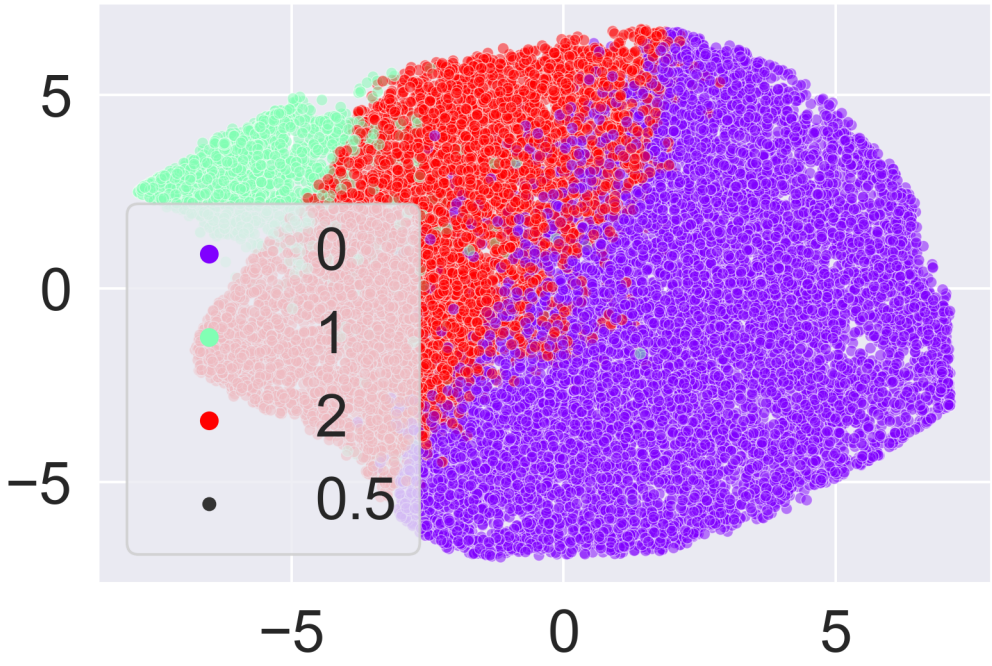
[173]:
```
# running cluster
k = 3
kmeans = KMeans(n_clusters=k, random_state=1)
kmeans.fit(lsa_tfidf_data_sclaed)
centers = kmeans.cluster_centers_.argsort()[:,::-1]
terms = tfidf_vectorizer.get_feature_names()

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

[174]:
```
tsne = TSNE(n_components=2, verbose=1, perplexity=92, n_iter=300)
X_ne = tsne.fit_transform(lsa_tfidf_data_sclaed[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 0.028s…
[t-SNE] Computed neighbors for 20628 samples in 2.682s…
[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: 0.372325
[t-SNE] KL divergence after 250 iterations with early exaggeration: 85.716904
[t-SNE] KL divergence after 300 iterations: 3.215184
```



[175]:
```
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 try', 'absolutely', 'absolute', 'able wear', 'able']
cluster1:
['able wear', 'absolutely', 'able', 'able try', 'absolute']
cluster2:
['able', 'absolute', 'able wear', 'absolutely', 'able try']
```

[176]:
```
kmeans.labels_
```

[176]: array([0, 2, 2, …, 0, 2, 0])

[177]:
```
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")
```

0 Absolutely wonderful - silky and sexy and comfortable

3 I love, love, love this jumpsuit. it's fun, flirty, and fabulous! every time i wear it, i get nothing but great compliments!

4 This shirt is very flattering to all due to the adjustable front tie. it is the perfect length to wear with leggings and it is sleeveless so it pairs well with any cardigan. love this shirt!!!

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.

11 This dress is perfection! so pretty and flattering.

[178]:
```
indices_max = [index for index, value in enumerate(kmeans.labels_) if value==1]
for rev_index in indices_max[:5]:
    print(rev_index, str(df.ReviewText[rev_index]))
    print("\n")
```

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

26 I have been waiting for this sweater coat to ship for weeks and i was so excited for it to arrive. this coat is not true to size and made me look short

and squat. the sleeves are very wide (although long). as a light weight fall coat the sleeves don't need to be as wide because you wouldn't be layerng too much underneath. the buttons need to be moved at least three inches in for a nicer fit. i thought about redoing the buttons myself but the sleeves looked even more out of proportion with a tigh

90 I love cute summer dresses and this one, especially because it is made out of linen, is unique. it is very well-made with a design that is quite flattering. i am 5 foot 6 and a little curvy with a 38 c bust and i got a size 10. it fits well although it is difficult to zip up because the material has no give. the perfect dress to wear to italy or france! now i just have to book my tickets!

91 This top is so much prettier in real life than it is on the model. the pattern and texture are both lovely, and the peplum is surprisingly flattering. it is definitely on the short side, but i think that gives it a modern look. the fabric does not stretch at all, but i still think it fits tts. if you have a very large chest you may want to go up a size, but otherwise i would order your normal size.

105 I bought this lovely silk/velvet shirt in the "sky" color but it is more on the teal blue side than sky blue, which disappointed me. it is definitely darker than appears in photo. still a luxurious well-made beauty with sassy appeal. it drapes like a snake slithering down your body. it comes with attitude.

[179]:
```
indices_max = [index for index, value in enumerate(kmeans.labels_) if value==2]
for rev_index in indices_max[:5]:
    print(rev_index, str(df.ReviewText[rev_index]))
    print("\n")
```

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.

2 I had such high hopes for this dress and really wanted it to work for me. i initially ordered the petite small (my usual size) but i found this to be outrageously small. so small in fact that i could not zip it up! i reordered it in petite medium, which was just ok. overall, the top half was comfortable and fit nicely, but the bottom half had a very tight under layer and several somewhat cheap (net) over layers. imo, a major design flaw was the net over layer sewn directly into the zipper - it c

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.

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…

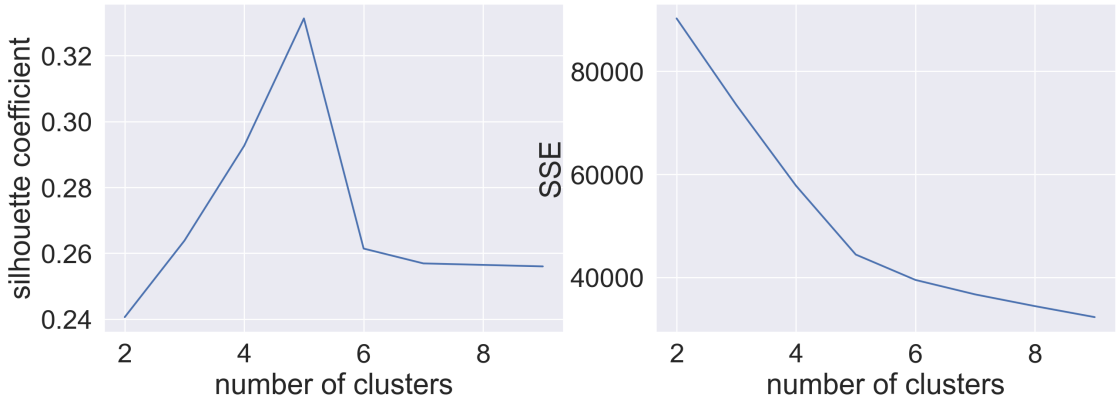
8 I love this dress. i usually get an xs but it runs a little snug in bust so i ordered up a size. very flattering and feminine with the usual retailer flair for style.

4 nmf_tfidf

[180]:
```
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_)
```

[181]:
```
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');
```



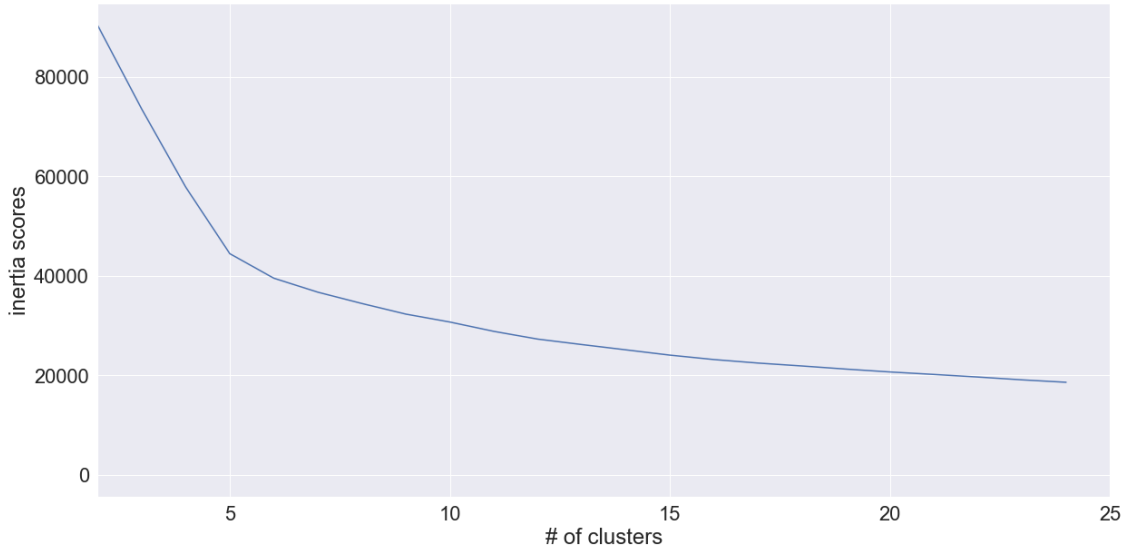
[182]:
```
inertia = [0,0]

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: 90278.007152
# clusters:  3   Inertia: 73474.534328
# clusters:  4   Inertia: 57776.866415
# clusters:  5   Inertia: 44419.244381
# clusters:  6   Inertia: 39489.609679
# clusters:  7   Inertia: 36680.706907
# clusters:  8   Inertia: 34407.798455
# clusters:  9   Inertia: 32272.817235
# clusters: 10   Inertia: 30678.677221
# clusters: 11   Inertia: 28802.094071
# clusters: 12   Inertia: 27241.494322
# clusters: 13   Inertia: 26149.893723
# clusters: 14   Inertia: 25086.664870
# clusters: 15   Inertia: 24025.871301
# clusters: 16   Inertia: 23134.714888
# clusters: 17   Inertia: 22433.028289
# clusters: 18   Inertia: 21843.331034
# clusters: 19   Inertia: 21217.445415
# clusters: 20   Inertia: 20643.078988
# clusters: 21   Inertia: 20144.198020
# clusters: 22   Inertia: 19619.974591
# clusters: 23   Inertia: 19054.747033
```

```
# clusters: 24    Inertia: 18567.503157
```

[183]:
```
plt.figure(figsize=(20,10))
plt.plot(inertia)
plt.xlabel('# of clusters')
plt.xlim((2,25))
plt.ylabel('inertia scores')
#plt.ylim((650,1200))
```

[183]:
```
Text(0, 0.5, 'inertia scores')
```



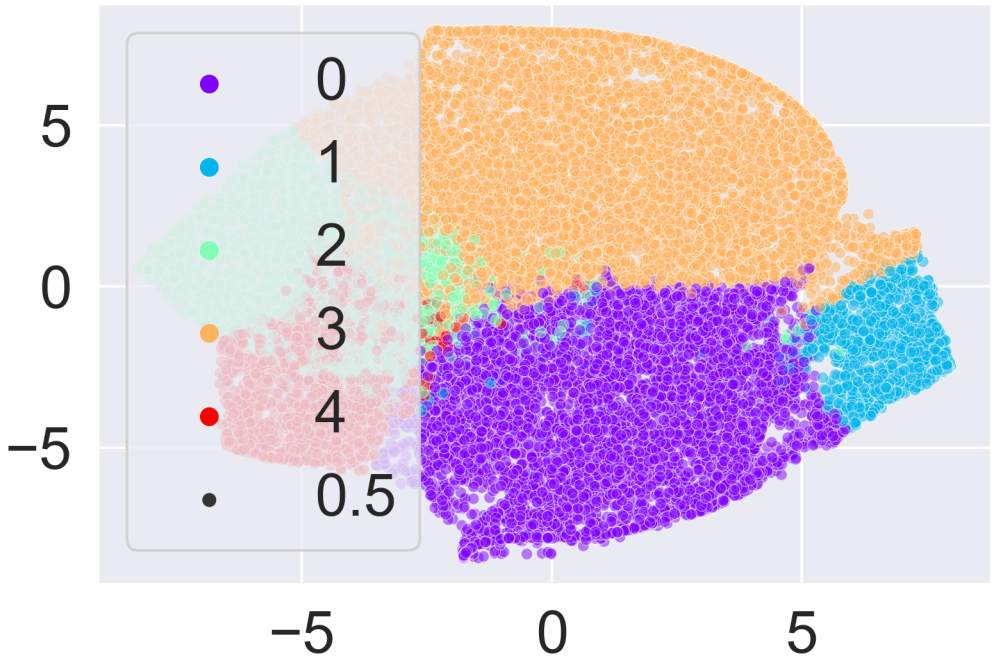
[184]:
```
# running cluster
k = 5
kmeans = KMeans(n_clusters=k, random_state=1)
kmeans.fit(nmf_tfidf_data_scaled)
centers = kmeans.cluster_centers_.argsort()[:,::-1]
terms = tfidf_vectorizer.get_feature_names()

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 try', 'absolutely', 'able wear', 'absolute', 'able']
cluster1:
['absolutely', 'absolute', 'able wear', 'able try', 'able']
```

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

[185]:
```
tsne = TSNE(n_components=2, verbose=1, perplexity=92, n_iter=300)
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 0.105s…
[t-SNE] Computed neighbors for 20628 samples in 2.434s…
[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: 0.227238
[t-SNE] KL divergence after 250 iterations with early exaggeration: 79.861053
[t-SNE] KL divergence after 300 iterations: 2.871678
```



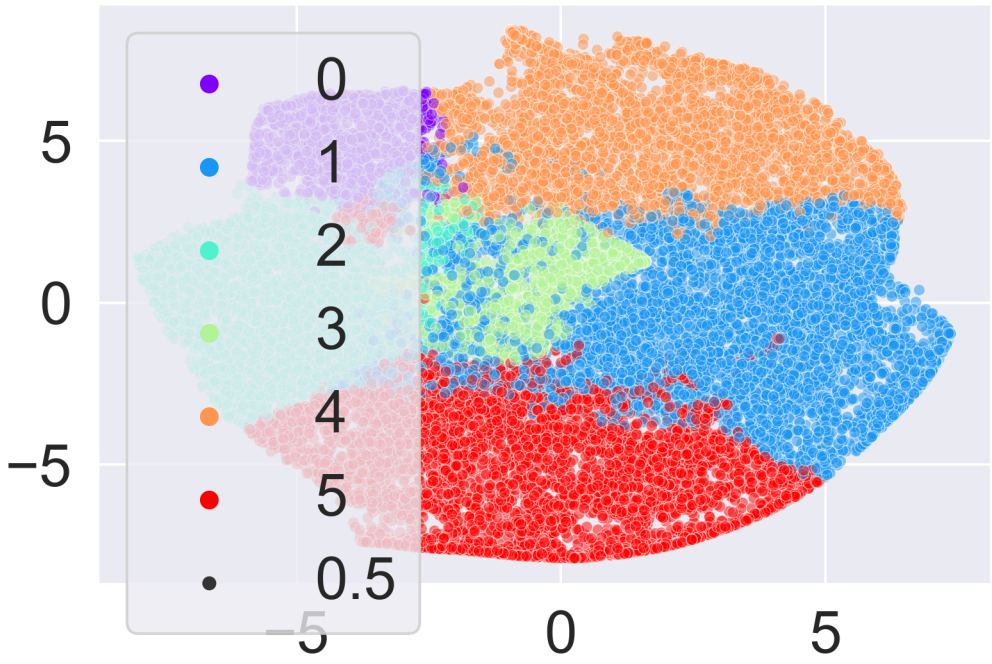
[186]:
```
# running cluster
k = 6
kmeans = KMeans(n_clusters=k, random_state=1)
kmeans.fit(nmf_tfidf_data_scaled)
centers = kmeans.cluster_centers_.argsort()[:,::-1]
terms = tfidf_vectorizer.get_feature_names()

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:
['absolutely', 'absolute', 'able wear', 'able try', 'able']
cluster1:
['able try', 'able', 'absolutely', 'able wear', 'absolute']
cluster2:
['absolute', 'able wear', 'absolutely', 'able try', 'able']
cluster3:
['able wear', 'absolutely', 'able try', 'absolute', 'able']
cluster4:
['able try', 'absolutely', 'able wear', 'absolute', 'able']
cluster5:
```

```
['able', 'absolutely', 'able wear', 'absolute', 'able try']
```

[187]:
```
tsne = TSNE(n_components=2, verbose=1, perplexity=92, n_iter=300)
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 0.105s…
[t-SNE] Computed neighbors for 20628 samples in 2.543s…
[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: 0.227238
[t-SNE] KL divergence after 250 iterations with early exaggeration: 80.322815
[t-SNE] KL divergence after 300 iterations: 2.896915
```



[188]:
```
indices_max = [index for index, value in enumerate(kmeans.labels_) if value==3]
for rev_index in indices_max[:5]:
    print(rev_index, str(df.ReviewText[rev_index]))
    print("\n")
```

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

41 This is a beautiful top. it's unique and not so ordinary. i bought my usual medium and i found that it fits tight across my chest. although i had a baby this year and i am nursing, so that could be why. if i bought again i would size up.

91 This top is so much prettier in real life than it is on the model. the pattern and texture are both lovely, and the peplum is surprisingly flattering. it is definitely on the short side, but i think that gives it a modern look. the fabric does not stretch at all, but i still think it fits tts. if you have a very large chest you may want to go up a size, but otherwise i would order your

normal size.

105 I bought this lovely silk/velvet shirt in the "sky" color but it is more on the teal blue side than sky blue, which disappointed me. it is definitely darker than appears in photo. still a luxurious well-made beauty with sassy appeal. it drapes like a snake slithering down your body. it comes with attitude.

110 This is so thin and poor quality. especially for the price. it felt like a thin pajama top. the buttons are terrible little shell buttons. this could not have been returned faster.

4.1 3-grams Tokenization

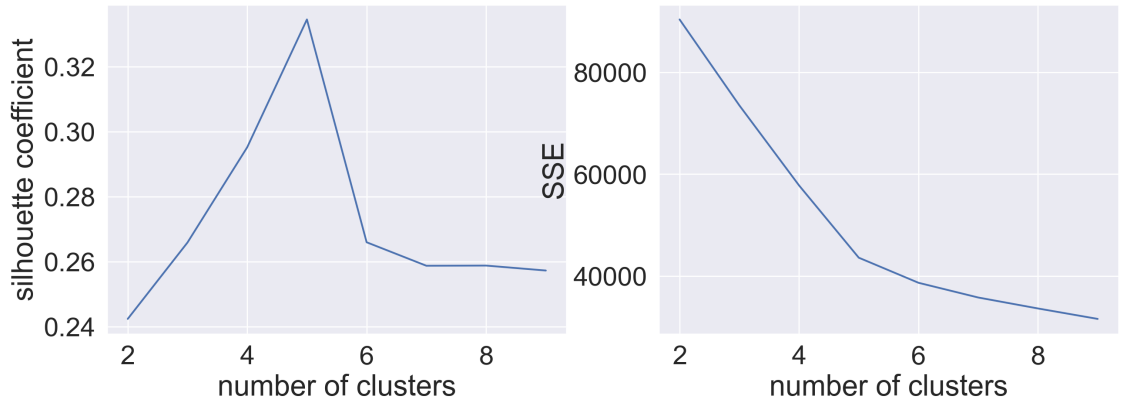
[189]:
```
# initialize vectorizers
count_vectorizer = CountVectorizer(ngram_range=(1, 3),
                                   stop_words='english',
                                   token_pattern="\\b[a-z][a-z]+\\b",
                                   lowercase=True,
                                   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)
```

[190]:
```
# 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)
```

[191]:
```
# 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)
```

[192]:
```
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_)
```

[193]:
```
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');
```

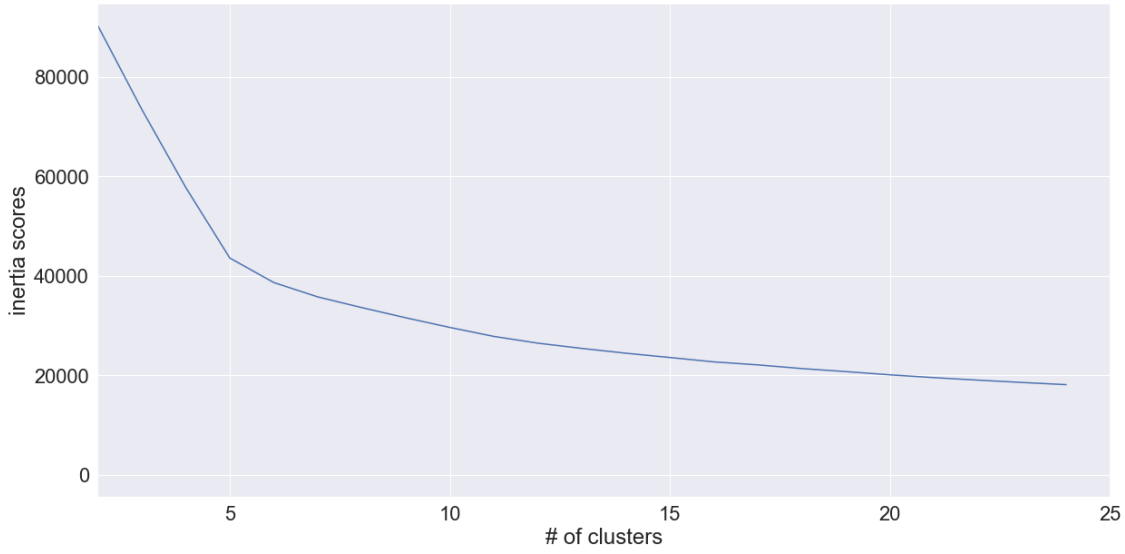


[194]:
```
inertia = [0,0]

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: 90340.393971
# clusters:  3   Inertia: 73482.435232
# clusters:  4   Inertia: 57759.568145
# clusters:  5   Inertia: 43570.098491
# clusters:  6   Inertia: 38643.199525
# clusters:  7   Inertia: 35751.132429
# clusters:  8   Inertia: 33599.519263
# clusters:  9   Inertia: 31551.116988
# clusters: 10   Inertia: 29611.734994
# clusters: 11   Inertia: 27806.471103
# clusters: 12   Inertia: 26448.721162
# clusters: 13   Inertia: 25389.954529
# clusters: 14   Inertia: 24427.662263
# clusters: 15   Inertia: 23563.130268
# clusters: 16   Inertia: 22681.318628
# clusters: 17   Inertia: 22078.271843
# clusters: 18   Inertia: 21337.114594
# clusters: 19   Inertia: 20729.231499
# clusters: 20   Inertia: 20079.719143
# clusters: 21   Inertia: 19512.206468
# clusters: 22   Inertia: 19012.125455
# clusters: 23   Inertia: 18541.104571
# clusters: 24   Inertia: 18102.877404
```

[195]:
```
plt.figure(figsize=(20,10))
plt.plot(inertia)
plt.xlabel('# of clusters')
plt.xlim((2,25))
plt.ylabel('inertia scores')
#plt.ylim((650,1200))
```

[195]:
```
Text(0, 0.5, 'inertia scores')
```



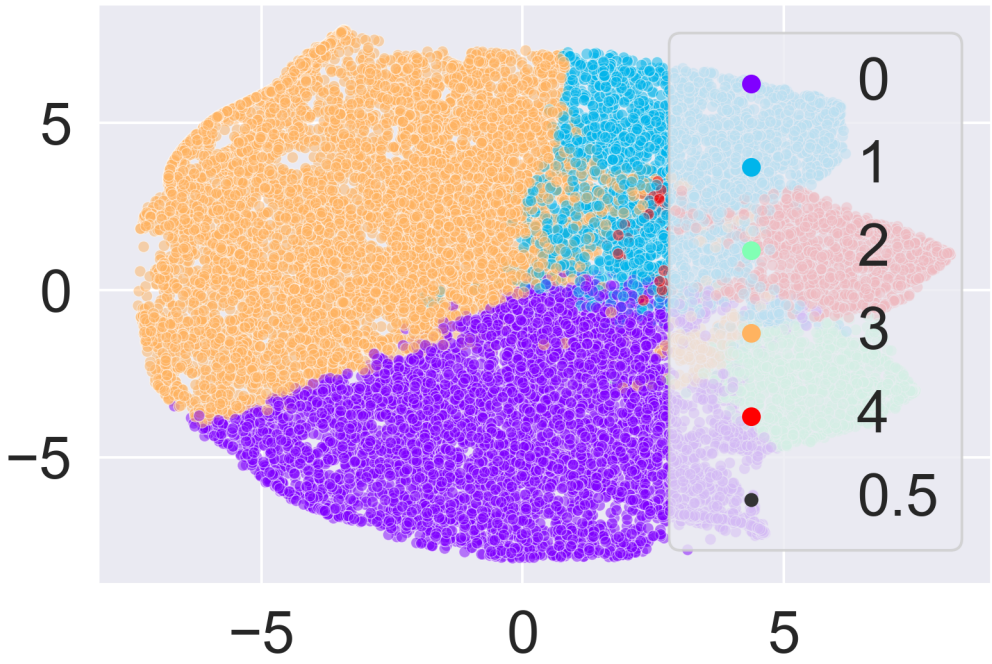
[196]:
```
# running cluster
k = 5
kmeans = KMeans(n_clusters=k, random_state=42)
kmeans.fit(nmf_tfidf_data_scaled)
centers = kmeans.cluster_centers_.argsort()[:,::-1]
terms = tfidf_vectorizer.get_feature_names()

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

[197]:
```
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 0.107s…
[t-SNE] Computed neighbors for 20628 samples in 2.576s…
[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
[t-SNE] KL divergence after 250 iterations with early exaggeration: 79.447639
[t-SNE] KL divergence after 300 iterations: 2.865235
```



[198]:
```
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")
```

```
0 Absolutely wonderful - silky and sexy and comfortable


3 I love, love, love this jumpsuit. it's fun, flirty, and fabulous! every time i
wear it, i get nothing but great compliments!


17 Took a chance on this blouse and so glad i did. i wasn't crazy about how the
blouse is photographed on the model. i paired it whit white pants and it worked
perfectly. crisp and clean is how i would describe it. launders well. fits
great. drape is perfect. wear tucked in or out - can't go wrong.


18 A flattering, super cozy coat.  will work well for cold, dry days and will
look good with jeans or a dressier outfit.  i am 5' 5'', about 135 and the small
fits great.


19 I love the look and feel of this tulle dress. i was looking for something
different, but not over the top for new year's eve. i'm small chested and the
```

top of this dress is form fitting for a flattering look. once i steamed the tulle, it was perfect! i ordered an xsp. length was perfect too.

After we identified how many clusters are the best, you can print out the documents that are the closest to the centroid of each clusters for examinations.

[199]:
```
indices_max = [index for index, value in enumerate(kmeans.labels_) if value==3]
for rev_index in indices_max[:5]:
    print(rev_index, str(df.ReviewText[rev_index]))
    print("\n")
```

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.

4.2 Classification of Reviews

Another thing we can try to separate the good or bad reviews from analyzing the text data is to perform a classification problem.

4.3 Read the cleaned data from EDA

[200]:
```
df = pd.read_pickle('cleaned_df.pkl')
df.head()
```

[200]:

	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…

In our data, we have a feature named Rating which is a rating score that a customer give to the product, while 1 is the least satisfied and 5 is the most satisfied.

We can set the Rating column as our target variable and our engineered CombinedText column as independent variable to see if we could build a classifier to automatically classify a comment.

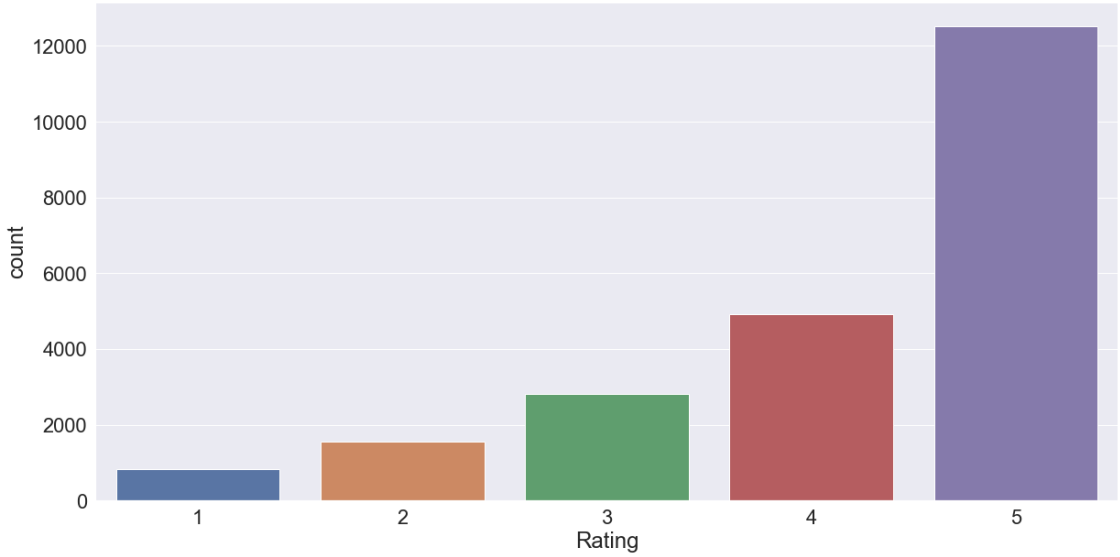
First thing I did was to group rank 1 to 4 together as bad review (labelled as 1), while rank 5 is our good review (labelled as 5). The two classes are not totally balance but they are in the acceptable range. I built classification models with naivesbayes and logistic classifiers.

4.4 Understand what are the distribution of each rank

4.5 Histogram of Review Ratings

[201]:
```
plt.figure(figsize=(20,10))
sns.set(font_scale = 2)
sns.countplot(df.Rating)
```

[201]:
```
<AxesSubplot:xlabel='Rating', ylabel='count'>
```



4.6 Perform more cleaning

[202]:
```
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)
```

[202]:
```
'I  things about  but not .'
```

[203]:
```
df['ReviewTextLower'] = df.ReviewText
```

[204]:
```
df['ReviewTextLower'] = df.ReviewTextLower.str.lower()
```

[205]:
```
df['ReviewTextLower'].replace(to_replace=pattern, value='', regex=True,␣
↪inplace=True)
```

[206]:
```
df.head()
```

[206]:

	Age	ReviewText	Rating	\
0	33	Absolutely wonderful - silky and sexy and comf…	4	
1	34	Love this dress! it's sooo pretty. i happene…	5	
2	60	I had such high hopes for this dress and reall…	3	
3	50	I love, love, love this jumpsuit. it's fun, fl…	5	
4	47	This shirt is very flattering to all due to th…	5	

```
   RecommendedIND  PositiveFeedbackCount    DivisionName DepartmentName  \
0               1                      0       Initmates       Intimate
1               1                      4         General        Dresses
2               0                      0         General        Dresses
3               1                      0  General Petite        Bottoms
4               1                      6         General           Tops

   ClassName                                        CombinedText  \
0  Intimates   Absolutely wonderful - silky and sexy and com…
1    Dresses   Love this dress!  it's sooo pretty.  i happen…
2    Dresses  Some major design flaws I had such high hopes …
3      Pants  My favorite buy! I love, love, love this jumps…
4    Blouses  Flattering shirt This shirt is very flattering…

                                    ReviewTextLower
0  absolutely wonderful - silky and sexy and comf…
1   this !  it's sooo pretty.  i happened to find…
2  i had such high hopes for this  and really wan…
3  i , ,  this jumpsuit. it's fun, flirty, and fa…
4  this shirt is very flattering to all due to th…
```

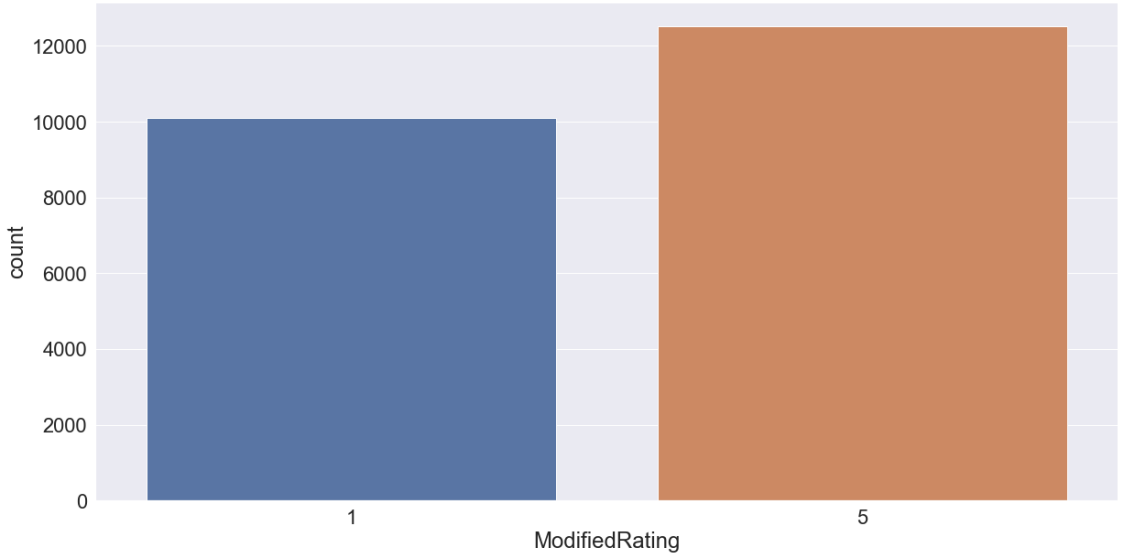
4.7 Group different ranks together as my target rank

4.8 Histogram of Bad vs Good Review Counts

[207]:
```
df['ModifiedRating'] = df.Rating.replace([2, 3, 4], 1)
```

[208]:
```
sns.set(font_scale = 2)
plt.figure(figsize=(20,10))
sns.countplot(df.ModifiedRating)
```

[208]:
```
<AxesSubplot:xlabel='ModifiedRating', ylabel='count'>
```



[209]:
```
len(df[df.ModifiedRating == 1])
```

[209]: 10101

[210]:
```
len(df[df.ModifiedRating == 5])
```

[210]: 12527

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

4.9 Build different classification models

Using ReviewTextLower column

[211]:
```
# using ReviewTextLower column as my predictors
# using ModifiedRating column as my target variable
X = df['ReviewTextLower']
y = df['ModifiedRating']
```

[212]:
```
# 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)
```

[213]:
```
len(count_vectorizer.vocabulary_)
```

[213]: 254500

[214]:
```
# 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)
```

[215]:
```
# train with multinomail Naive Bayes
nb = MultinomialNB()
nb.fit(X_train, y_train)
```

[215]: MultinomialNB()

[216]:
```
# make prediction
y_pred = nb.predict(X_test)
```

[217]:
```
# 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]
 [ 462 3302]]


              precision    recall  f1-score   support

           1       0.82      0.70      0.76      3025
           5       0.79      0.88      0.83      3764

    accuracy                           0.80      6789
   macro avg       0.80      0.79      0.79      6789
weighted avg       0.80      0.80      0.80      6789
```

[218]:
```
# trying to make a prediction using this review
# this is an example for positive review
positive_example = df['ReviewTextLower'][1]
positive_example
```

[218]: ' 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.'
```

[219]:
```
# vectorize the positive_example
positive_example_vec = count_vectorizer.transform([positive_example])
# make prediction
nb.predict(positive_example_vec)[0]
```

[219]:
```
5
```

[220]:
```
# trying to make a prediction using an negative review
negative_example = df['ReviewTextLower'][5]
negative_example
```

[220]:
```
'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 .'
```

[221]:
```
negative_example_vec = count_vectorizer.transform([negative_example])
nb.predict(negative_example_vec)[0]
```

[221]:
```
1
```

[222]:
```
negative_example = df['ReviewTextLower'][10]
negative_example
```

[222]:
```
' runs small esp where the  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.'
```

[223]:
```
negative_example_vec = count_vectorizer.transform([negative_example])
nb.predict(negative_example_vec)[0]
```

[223]:
```
1
```

Using Combinetext column

[224]:
```
df['CombinedTextLower'] = df.CombinedText
```

[225]:
```
df['CombinedTextLower'] = df.CombinedTextLower.str.lower()
```

[226]:
```
df['CombinedTextLower'].replace(to_replace=pattern, value='', regex=True,␣
↪inplace=True)
```

[227]:
```
X = df.CombinedTextLower
y = df.ModifiedRating
```

[228]:
```
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)
```

[229]:
```
len(count_vectorizer.vocabulary_)
```

[229]: 267875

[230]:
```
X_train, X_test, y_train, y_test = train_test_split(cv_data, y, test_size=0.3,␣
↪random_state=42)
```

[231]:
```
nb = MultinomialNB()
nb.fit(X_train, y_train)
```

[231]: MultinomialNB()

[232]:
```
y_pred = nb.predict(X_test)
```

[233]:
```
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

    accuracy                           0.81      6789
   macro avg       0.82      0.80      0.81      6789
weighted avg       0.82      0.81      0.81      6789
```

[234]:
```
logit = LogisticRegression()
logit.fit(X_train, y_train)
```

[234]:
```
LogisticRegression()
```

[235]:
```
y_pred = logit.predict(X_test)
```

[236]:
```
print(confusion_matrix(y_test, y_pred))
print('\n')
print(classification_report(y_test, y_pred))
```

```
[[2248  777]
 [ 554 3210]]


              precision    recall  f1-score   support

           1       0.80      0.74      0.77      3025
           5       0.81      0.85      0.83      3764

    accuracy                           0.80      6789
   macro avg       0.80      0.80      0.80      6789
weighted avg       0.80      0.80      0.80      6789
```

[237]:
```
df.ReviewText[23]
```

[237]:
```
"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."
```

[238]:
```
df.ReviewText[14]
```

[238]:
```
'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'
```

[239]:
```
df['ReviewText'][1]
```

[239]:
```
'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.'
```

[240]:
```
positive_example = df['CombinedTextLower'][1]
positive_example
```

[240]:
```
'  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.'
```

[241]:
```
positive_example_vec = count_vectorizer.transform([positive_example])
logit.predict(positive_example_vec)[0]
```

[241]:
```
5
```

[242]:
```
df['ReviewText'][5]
```

[242]:
```
'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.'
```

[243]:
```
negative_example=df['ReviewTextLower'][5]
negative_example
```

[243]:
```
'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 .'
```

[244]:
```
negative_example_vec = count_vectorizer.transform([negative_example])
logit.predict(negative_example_vec)[0]
```

[244]:
```
1
```

[245]:
```
negative_example=df['ReviewTextLower'][10]
negative_example
```

[245]:
```
' runs small esp where the  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.'
```

[246]:
```
negative_example_vec = count_vectorizer.transform([negative_example])
logit.predict(negative_example_vec)[0]
```

[246]:
```
1
```

[247]:
```
df.ReviewText[23484]
```

[247]:
```
"I bought this dress for a wedding i have this summer, and it's so cute.
unfortunately the fit isn't perfect. the medium fits my waist perfectly, but was
way too long and too big in the bust and shoulders. if i wanted to spend the
money, i could get it tailored, but i just felt like it might not be worth it.
side note - this dress was delivered to me with a nordstrom tag on it and i
found it much cheaper there after looking!"
```

[248]:
```
negative_example=df['ReviewTextLower'][23484]
negative_example
```

[248]:
```
"i bought this  for a wedding i have this summer, and it's so cute.
unfortunately the  isn't perfect. the medium fits my waist perfectly, but was
way too long and too big in the bust and shoulders. if i wanted to spend the
money, i could get it tailored, but i just felt like it might not be worth it.
side note - this  was delivered to me with a nordstrom tag on it and i found it
much cheaper there after looking!"
```

[249]:
```
negative_example_vec = count_vectorizer.transform([negative_example])
logit.predict(negative_example_vec)[0]
```

[249]:
```
1
```

[250]:
```
df.ReviewText[23478]
```

[250]:
```
'I was surprised at the positive reviews for this product. its terrible! it cuts
you in a weird place to make you look wide. the skirt is also not like the
picture. its darker and heavier. the material isnt great. i had to return.'
```

[251]:
```
negative_example=df['ReviewTextLower'][23478]
negative_example
```

[251]:
```
'i was surprised at the positive reviews for this product. its terrible! it cuts
you in a weird place to make you look wide. the skirt is also not like the
picture. its darker and heavier. the material isnt great. i had to return.'
```

[252]:
```
negative_example_vec = count_vectorizer.transform([negative_example])
logit.predict(negative_example_vec)[0]
```

[252]:
```
1
```

[253]:
```
negative_example=df['ReviewTextLower'][14]
negative_example
```

[253]:
```
'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
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  is the quality. the fabrics
are terrible. the delicate netting type fabric on the top layer of skirt got
stuck in the '
```

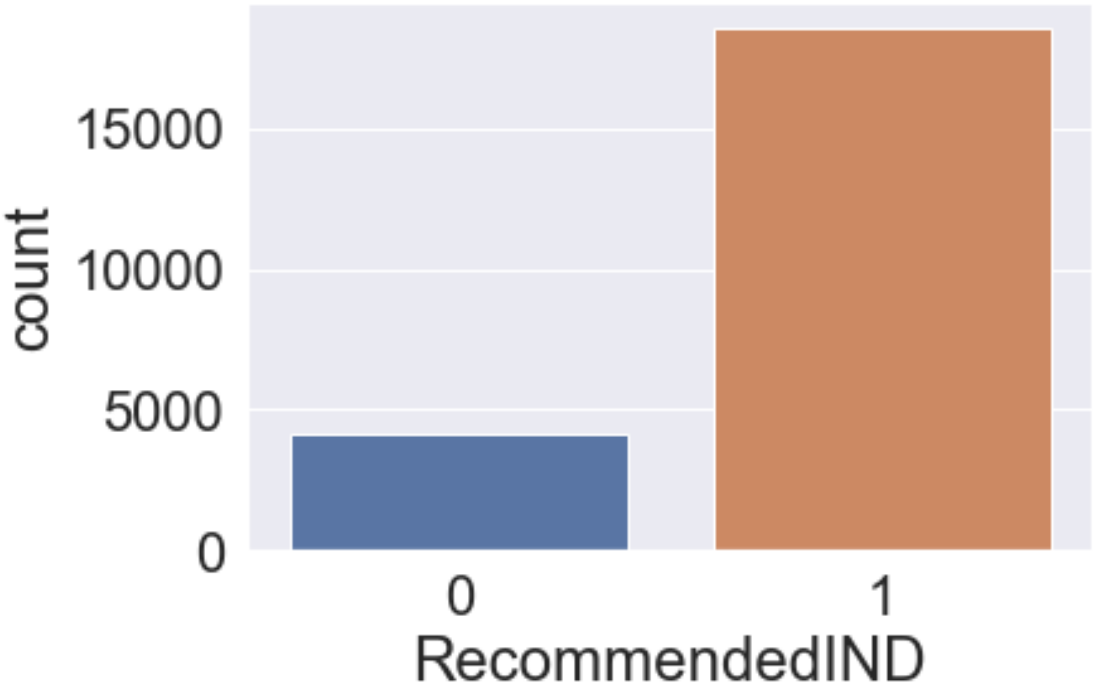
[254]:
```
negative_example_vec = count_vectorizer.transform([negative_example])
logit.predict(negative_example_vec)[0]
```

[254]:
```
1
```

5 predicting recommend or not

[255]:
```
sns.countplot(df.RecommendedIND)
```

[255]:
```
<AxesSubplot:xlabel='RecommendedIND', ylabel='count'>
```



[256]:
```
df.head()
```

[256]:
```
   Age                                          ReviewText  Rating  \
0   33  Absolutely wonderful - silky and sexy and comf…       4
1   34  Love this dress!  it's sooo pretty.  i happene…       5
2   60  I had such high hopes for this dress and reall…       3
3   50  I love, love, love this jumpsuit. it's fun, fl…       5
4   47  This shirt is very flattering to all due to th…       5

   RecommendedIND  PositiveFeedbackCount    DivisionName DepartmentName  \
0               1                      0       Initmates       Intimate
1               1                      4         General        Dresses
2               0                      0         General        Dresses
3               1                      0  General Petite        Bottoms
4               1                      6         General           Tops

   ClassName                                      CombinedText  \
0  Intimates   Absolutely wonderful - silky and sexy and com…
1    Dresses   Love this dress!  it's sooo pretty.  i happen…
2    Dresses  Some major design flaws I had such high hopes …
3      Pants  My favorite buy! I love, love, love this jumps…
4    Blouses  Flattering shirt This shirt is very flattering…

                                  ReviewTextLower  ModifiedRating  \
0  absolutely wonderful - silky and sexy and comf…               1
1   this !  it's sooo pretty.  i happened to find…               5
2  i had such high hopes for this  and really wan…               1
3  i , ,  this jumpsuit. it's fun, flirty, and fa…               5
4  this shirt is very flattering to all due to th…               5

                                  CombinedTextLower
0   absolutely wonderful - silky and sexy and com…
1    this !  it's sooo pretty.  i happened to fin…
2  some major design flaws i had such high hopes …
3  my favorite buy! i , ,  this jumpsuit. it's fu…
4  flattering shirt this shirt is very flattering…
```

[257]:
```
#rating_class = df[(df['Rating'] == 1) | (df['Rating'] == 5)]
X = df.CombinedTextLower
y = df.RecommendedIND
```

[258]:
```
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)
```

[259]:
```
len(count_vectorizer.vocabulary_)
```

[259]:
```
267875
```

[260]:
```
X_train, X_test, y_train, y_test = train_test_split(cv_data, y, test_size=0.3,␣
↪random_state=42, stratify=y)
```

[261]:
```
# using SMOTE to oversample
from imblearn.over_sampling import SMOTE

sm = SMOTE(random_state=42)
X_train_smoted, y_train_smoted = sm.fit_resample(X_train, y_train)
```

[262]:
```
print(Counter(y_train_smoted))
```

```
Counter({1: 12968, 0: 12968})
```

[263]:
```
nb = MultinomialNB()
nb.fit(X_train_smoted, y_train_smoted)
```

[263]:
```
MultinomialNB()
```

[264]:
```
y_pred = nb.predict(X_test)
```

[265]:
```
print(confusion_matrix(y_test, y_pred))
print('\n')
print(classification_report(y_test, y_pred))
```

```
[[ 753  477]
 [ 263 5296]]


              precision    recall  f1-score   support

           0       0.74      0.61      0.67      1230
           1       0.92      0.95      0.93      5559

    accuracy                           0.89      6789
   macro avg       0.83      0.78      0.80      6789
weighted avg       0.89      0.89      0.89      6789
```

[266]:
```
logit = LogisticRegression()
logit.fit(X_train_smoted, y_train_smoted)
```

[266]:
```
LogisticRegression()
```

[267]:
```
y_pred = logit.predict(X_test)
```

[268]:
```
print(confusion_matrix(y_test, y_pred))
print('\n')
print(classification_report(y_test, y_pred))
```

```
[[ 829  401]
 [ 490 5069]]


              precision    recall  f1-score   support

           0       0.63      0.67      0.65      1230
           1       0.93      0.91      0.92      5559

    accuracy                           0.87      6789
   macro avg       0.78      0.79      0.78      6789
weighted avg       0.87      0.87      0.87      6789
```

5.1 Recall Scores for both bad (rank 1) and good (rank 5) reviews

The metric that I used for model evaluation, I used recall score because I care the cases when I predicted the review is good review but actually it is not. The best recall score that I got is 0.74 without a lot of engineering. The score could be better if there is more time and exploration on the model.

6 Ranking of Products

[17]:
```
import pandas as pd
```

[18]:
```
df = pd.read_pickle('cleaned_df.pkl')
df.head()
```

[18]:

	Age	ReviewText	Rating	\
0	33	Absolutely wonderful - silky and sexy and comf…	4	
1	34	Love this dress! it's sooo pretty. i happene…	5	
2	60	I had such high hopes for this dress and reall…	3	
3	50	I love, love, love this jumpsuit. it's fun, fl…	5	
4	47	This shirt is very flattering to all due to th…	5	

	RecommendedIND	PositiveFeedbackCount	DivisionName	DepartmentName	\

```
0              1                      0       Initmates       Intimate
1              1                      4         General        Dresses
2              0                      0         General        Dresses
3              1                      0  General Petite        Bottoms
4              1                      6         General           Tops

   ClassName                                       CombinedText
0  Intimates   Absolutely wonderful - silky and sexy and com…
1    Dresses   Love this dress!  it's sooo pretty.  i happen…
2    Dresses  Some major design flaws I had such high hopes …
3      Pants  My favorite buy! I love, love, love this jumps…
4    Blouses  Flattering shirt This shirt is very flattering…
```

6.0.1 Rankings of Products by Division

[23]:
```
df_division=df[['DivisionName','Rating']]
df_division.groupby("DivisionName").mean()["Rating"].nlargest()
```

[23]:
```
DivisionName
Initmates         4.275596
General Petite    4.196631
General           4.165282
Name: Rating, dtype: float64
```

[26]:
```
df_division=df[['DivisionName','Rating']]
df_division_rank=df_division.groupby('DivisionName').mean().
 ↪sort_values(by='Rating',ascending=False)
df_division_rank["Rank"]=range(1,len(df_division_rank)+1)
df_division_rank.reset_index().set_index("Rank")
```

[26]:
```
         DivisionName    Rating
Rank
1           Initmates  4.275596
2     General Petite  4.196631
3             General  4.165282
```

6.0.2 Rankings of Products by Department

[27]:
```
df_department=df[['DepartmentName','Rating']]
df_department.groupby("DepartmentName").mean()["Rating"].nlargest()
```

[27]:
```
DepartmentName
Bottoms     4.278809
Intimate    4.271022
Jackets     4.254491
Tops        4.157743
Dresses     4.138812
```

```
Name: Rating, dtype: float64
```

[28]:
```
df_department=df[['DepartmentName','Rating']]
df_department_rank=df_department.groupby('DepartmentName').mean().
 ↪sort_values(by='Rating',ascending=False)
df_department_rank["Rank"]=range(1,len(df_department_rank)+1)
df_department_rank.reset_index().set_index("Rank")
```

[28]:

Rank	DepartmentName	Rating
1	Bottoms	4.278809
2	Intimate	4.271022
3	Jackets	4.254491
4	Tops	4.157743
5	Dresses	4.138812
6	Trend	3.838983

6.0.3 Rankings of Products by Class

[29]:
```
df_class=df[['ClassName','Rating']]
df_class.groupby("ClassName").mean()["Rating"].nlargest()
```

[29]:
```
ClassName
Layering    4.348485
Jeans       4.347826
Lounge      4.301943
Sleep       4.294393
Jackets     4.288433
Name: Rating, dtype: float64
```

[30]:
```
df_class=df[['ClassName','Rating']]
df_class_rank=df_class.groupby('ClassName').mean().
 ↪sort_values(by='Rating',ascending=False)
df_class_rank["Rank"]=range(1,len(df_class_rank)+1)
df_class_rank.reset_index().set_index("Rank")
```

[30]:

Rank	ClassName	Rating
1	Layering	4.348485
2	Jeans	4.347826
3	Lounge	4.301943
4	Sleep	4.294393
5	Jackets	4.288433
6	Intimates	4.278912
7	Pants	4.261481
8	Shorts	4.253289
9	Legwear	4.246835

```
10        Fine gauge  4.237960
11            Skirts  4.229236
12         Outerwear  4.181818
13              Swim  4.171687
14          Sweaters  4.168841
15             Knits  4.145698
16           Blouses  4.142809
17           Dresses  4.138812
18   Casual bottoms  4.000000
19          Chemises  4.000000
20             Trend  3.838983
```

6.1 Project Execution Understandings

- Unsupervised learning is really very different than supervised learning because of its nature.
- We will expect to spend a lot of time trying to understand how to cluster your data, there are a lot of clustering methods out there beside KMeans.
- Doing text analytics or NLP, we can expect to spend a lot of your time cleaning your text data for best results. For example, how to decide what stop words to use based on the context of your data and problem you want to solve, how to lemmatize, how to vectorize, how to reduce your dimensionality and avoid curse of dimensionality, and etc.
- Carried out rankings of products by each its respective columns lke Division, Department and Class.