

Customer Satisfaction Analysis using NLTK

Scenario

You have been contracted by an online clothing shop to assist them in making informed decisions regarding the product categories featured on their website, focusing on customer satisfaction. The shop offers five clothing categories: Tops, Bottoms, Jackets, Dresses, and Intimate. Your task is to determine which category has higher customer satisfaction. The shop has not maintained records of customer ratings, and the only available data consists of customer reviews in text format.

This lab involves cleaning and preprocessing the raw text reviews using Python NLTK, followed by conducting sentiment analysis to assess customer satisfaction across product categories. Through this systematic approach, we aim to provide the client with valuable insights that will aid them in optimizing their product offerings and overall customer experience.

This guided lab covers the following tasks:

- Task 1: Loading the customer reviews dataset
- Task 2: Clean and preprocess the reviews
- Task 3: Tokenize the reviews and removing the stop words
- Task 4: Exploring the tokens and product categories
- Task 5: Extracting the adjectives in the reviews
- Task 6: Finding the sentiment of each review

Importing modules

Install any necessary libraries (e.g., nltk, wordcloud, plotly ...) using:

```
!pip install some_library
```

Then, import the necessary modules by running the cell below.

In [239...]

```
import pandas as pd
import string
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)

# Importing Natural Language Processing toolkit
import nltk

# Downloading the NLTK english stop words
nltk.download('stopwords')

# Downloading the NLTK sentence tokenizer
```

```

nltk.download('punkt')
nltk.download('punkt_tab')

# Downloading the NLTK POS Tagger
nltk.download('averaged_perceptron_tagger')
nltk.download('averaged_perceptron_tagger_eng')

# Downloading the NLTK Vader Lexicon
nltk.download('vader_lexicon')

# Importing the NLTK english stop words
from nltk.corpus import stopwords

# Importing frequency distribution from NLTK
from nltk.probability import FreqDist

# Importing VADER dictionary. It is a rule-based sentiment analyzer
from nltk.sentiment import SentimentIntensityAnalyzer

# Importing data visualization modules
from wordcloud import WordCloud
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.io as pio
pio.renderers.default = "notebook"
import warnings
warnings.filterwarnings('ignore')
print("Modules imported!")

```

Modules imported!

```

[nltk_data]  Downloading package stopwords to
[nltk_data]      /Users/ahmedhanif/nltk_data...
[nltk_data]  Package stopwords is already up-to-date!
[nltk_data]  Downloading package punkt to
[nltk_data]      /Users/ahmedhanif/nltk_data...
[nltk_data]  Package punkt is already up-to-date!
[nltk_data]  Downloading package punkt_tab to
[nltk_data]      /Users/ahmedhanif/nltk_data...
[nltk_data]  Package punkt_tab is already up-to-date!
[nltk_data]  Downloading package averaged_perceptron_tagger to
[nltk_data]      /Users/ahmedhanif/nltk_data...
[nltk_data]  Package averaged_perceptron_tagger is already up-to-
[nltk_data]      date!
[nltk_data]  Downloading package averaged_perceptron_tagger_eng to
[nltk_data]      /Users/ahmedhanif/nltk_data...
[nltk_data]  Package averaged_perceptron_tagger_eng is already up-to-
[nltk_data]      date!
[nltk_data]  Downloading package vader_lexicon to
[nltk_data]      /Users/ahmedhanif/nltk_data...
[nltk_data]  Package vader_lexicon is already up-to-date!

```

Run the following cell to download the datasets `dataset.csv` and `reviews.csv`
:

In [240...]

```

from urllib.request import urlretrieve
urlretrieve("https://raw.githubusercontent.com/bouguelia/raw-materials/re
urlretrieve("https://raw.githubusercontent.com/bouguelia/raw-materials/re
print("Datasets downloaded!")

```

Datasets downloaded!

Task 1: Loading the customer reviews dataset

Load the data from `dataset.csv` into a pandas dataframe called `data` and display it.

In [241...]

```
# Your code here
data = pd.read_csv("./dataset.csv")
data.head()
```

Out[241...]

	product_review	product_category
0	I love, love, love this jumpsuit. it's fun, ni...	Bottoms
1	Beautifully made pants and on trend with the f...	Bottoms
2	I never would have given these pants a second ...	Bottoms
3	These pants are even better in person. the onl...	Bottoms
4	The silhouette and length of this skirt and le...	Bottoms

►💡 **Hint (click here to expand)**

►✅ **Solution (click here to reveal)**

Let's check the first `product_review`

In [242...]

```
# Your code here
data["product_review"].iloc[0]
```

Out[242...]

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

►✅ **Solution (click here to reveal)**

Checking the number of reviews per product category

In [243...]

```
# Your code here
data.product_category.value_counts()
```

Out[243...]

product_category	count
Bottoms	685
Dresses	681
Tops	680
Jackets	680
Intimate	650

Name: count, dtype: int64

►✅ **Solution (click here to reveal)**

Task 2: Clean and preprocess the reviews

- Lower casing

- Removing the punctuations

In [244... `data.head() # Displaying the original data`

		product_review	product_category
0	I love, love, love this jumpsuit. it's fun, ni...		Bottoms
1	Beautifully made pants and on trend with the f...		Bottoms
2	I never would have given these pants a second ...		Bottoms
3	These pants are even better in person. the onl...		Bottoms
4	The silhouette and length of this skirt and le...		Bottoms

Converting all the reviews to lower case:

In [245... `# Your code here`
`def to_lower_case(text):`
 `return text.lower()`
`data["product_review"] = data["product_review"].apply(to_lower_case)`
`data.product_review.sample(10)`

Out[245... 1302 i received this dress in the mail yesterday. r...
293 wonderful fabric. surprisingly great fit. supe...
1640 after surviving the winter months in various c...
1863 i saw this online and thought it was so pretty...
1471 i grabbed this pair of pj pants on sale, and i...
1638 this shirt fits very closely to the body. it h...
1366 just purchased this- love how light and airy i...
187 iordered hte petite after trying the rregualr ...
3251 this jacket is so adorable on! i have no idea ...
1741 i tried this on in white and considered whethe...
Name: product_review, dtype: object

►💡 [Hint \(click here to expand\)](#)

►✅ [Solution \(click here to reveal\)](#)

Remove the punctuations from all the reviews:

In [246... `# Your code here`
`import string`
`def remove_punctuation(text):`
 `return "".join([char for char in text if char not in string.punctuation])`
`print("Before:", data.product_review.iloc[0])`
`print(string.punctuation)`
`data.product_review = data.product_review.apply(remove_punctuation)`
`print("After:", data.product_review.iloc[0])`

Before: i love, love, love this jumpsuit. it's fun, nice, and fabulous! every time i wear it, i get nothing but great compliments!
 !"#\$%&'()*+,.-./:;=>?@[\\]^_`{|}~
 After: i love love love this jumpsuit its fun nice and fabulous every time i wear it i get nothing but great compliments

►💡 [Hint \(click here to expand\)](#)

►✅ [Solution \(click here to reveal\)](#)

Task 3: Tokenize the reviews and removing the stop words

- **Tokenization** is the process of breaking down a continuous stream of text, such as a sentence or a paragraph, into smaller units called tokens. These tokens typically correspond to words, but can also represent subword units like prefixes, suffixes, and stems.
- **Tokenization** facilitates the transformation of text into a format that machine learning algorithms can understand.

In [247...]

```
# Example:
tokens = nltk.word_tokenize(data.product_review[0])
print(tokens)

['i', 'love', 'love', 'love', 'this', 'jumpsuit', 'its', 'fun', 'nice', 'a
nd', 'fabulous', 'every', 'time', 'i', 'wear', 'it', 'i', 'get', 'nothin
g', 'but', 'great', 'compliments']
```

In [248...]

```
def tokenize(text):
    return nltk.word_tokenize(text)

data["product_review_tokenized"] = data.product_review.apply(tokenize) #
```

In [249...]

```
data.sample(5)
```

Out[249...]

	product_review	product_category	product_review_tokenized
1451	i was so stoked about this swim suit i bought...	Intimate	[i, was, so, stoked, about, this, swim, suit, ...]
1353	i ordered an xxs petite and this hangs like a ...	Dresses	[i, ordered, an, xxs, petite, and, this, hangs...]
1388	i have been eyeing this piece for months now i...	Intimate	[i, have, been, eyeing, this, piece, for, mont...]
1245	i tried this dress on in a 6 and a 4 a 4 is a ...	Dresses	[i, tried, this, dress, on, in, a, 6, and, a, ...]
1044	maybe its my curves or the way im only 54 but ...	Dresses	[maybe, its, my, curves, or, the, way, im, onl...]

Let's remove the Stop Words

Stop words are common words (e.g., "the," "and," "is") that appear frequently in a language and have little semantic value. Removing them is essential in natural language processing tasks to reduce data size, speed up processing, and improve the accuracy of algorithms by focusing on more informative words that convey the actual meaning of a text.

In [250...]

```
# Example:
english_stopwords = stopwords.words("English")
print(english_stopwords)

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

Let's remove the stop words from the `tokens` list

In [251...]

```
# Example:
[t for t in tokens if t not in english_stopwords]
```

Out[251...]

```
['love',
'love',
'love',
'jumpsuit',
'fun',
'nice',
'fabulous',
'every',
'time',
'wear',
'get',
'nothing',
'great',
'compliments']
```

Now, let's remove the stopwords from the tokenized reviews. Create a new column `'cleaned_tokens'` containing the tokenized reviews without the stopwords:

In [252...]

```
# Your code here
def clean_tokens(tokenized_words):
    filtered_words = []
    for word in tokenized_words:
        if word not in english_stopwords:
            filtered_words.append(word)

    return filtered_words
```

In [253...]

```
# test
tokenized_words_example = data.product_review_tokenized.iloc[0]
print("Before removing stopwords:", tokenized_words_example)
clean_tokens(tokenized_words_example)
```

Before removing stopwords: ['i', 'love', 'love', 'love', 'this', 'jumpsuit', 'its', 'fun', 'nice', 'and', 'fabulous', 'every', 'time', 'i', 'wear', 'it', 'i', 'get', 'nothing', 'but', 'great', 'compliments']

Out[253...]

```
['love',
 'love',
 'love',
 'jumpsuit',
 'fun',
 'nice',
 'fabulous',
 'every',
 'time',
 'wear',
 'get',
 'nothing',
 'great',
 'compliments']
```

In [254...]

```
data["cleaned_tokens"] = data.product_review_tokenized.apply(clean_tokens)
```

In [255...]

```
data.sample(5)
```

Out[255...]

	product_review	product_category	product_review_tokenized	cleaned_token
1199	i recommend this dress because it is beautiful...	Dresses	[i, recommend, this, dress, because, it, is, b...]	[recommend dress, beautifu stylish, well, m.]
1129	i was so sure i was going to love this dress W...	Dresses	[i, was, so, sure, i, was, going, to, love, th...]	[sure, going love, dress, sav rack, disappoi.
1510	i received this for christmas this year and ha...	Intimate	[i, received, this, for, christmas, this, year...]	[received christmas, yea already, worr sev.]
1300	i really loved this dress in the store when i ...	Dresses	[i, really, loved, this, dress, in, the, store...]	[really, loved dress, store tried, color, fa.
669	i typically wear a 26 in jeans and in these i ...	Bottoms	[i, typically, wear, a, 26, in, jeans, and, in...]	[typically, wea 26, jeans, go 25, fit, per.]

►  **Hint (click here to expand)**

►  **Solution (click here to reveal)**

Let's recreate the reviews from the cleaned tokens again:

```
In [256... # Example:  
tokens = data.cleaned_tokens[0]  
" ".join(tokens)
```

```
Out[256... 'love love love jumpsuit fun nice fabulous every time wear get nothing g  
reat compliments'
```

```
In [257... # Create a column 'product_review_cleaned' from the cleaned tokens  
data['product_review_cleaned'] = data.cleaned_tokens.apply(lambda x: " ".  
data.head()
```

	product_review	product_category	product_review_tokenized	cleaned_tokens
0	i love love love this jumpsuit its fun nice an...	Bottoms	[i, love, love, love, this, jumpsuit, its, fun...]	[love, love, love, jumpsuit, fun, nice, fabulo...
1	beautifully made pants and on trend with the f...	Bottoms	[beautifully, made, pants, and, on, trend, wit...]	[beautifully, made, pants, trend, flared, crop...
2	i never would have given these pants a second ...	Bottoms	[i, never, would, have, given, these, pants, a...]	[never, would, given, pants, second, look, onl...
3	these pants are even better in person the only...	Bottoms	[these, pants, are, even, better, in, person, ...]	[pants, even, better, person, downside, need, ...
4	the silhouette and length of this skirt and le...	Bottoms	[the, silhouette, and, length, of, this, skirt...]	[silhouette, length, skirt, length, flattering...

Task 4

Exploring the tokens and product categories

```
In [258... data.head()
```

Out [258...]

	product_review	product_category	product_review_tokenized	cleaned_tokens
0	i love love love this jumpsuit its fun nice an...	Bottoms	[i, love, love, love, this, jumpsuit, its, fun...]	[love, love, love, jumpsuit, fun, nice, fabulo...
1	beautifully made pants and on trend with the f...	Bottoms	[beautifully, made, pants, and, on, trend, wit...]	[beautifully, made, pants, trend, flared, crop...
2	i never would have given these pants a second ...	Bottoms	[i, never, would, have, given, these, pants, a...]	[never, would, given, pants, second, look, onl...
3	these pants are even better in person the only...	Bottoms	[these, pants, are, even, better, in, person, ...]	[pants, even, better, person, downside, need, ...
4	the silhouette and length of this skirt and le...	Bottoms	[the, silhouette, and, length, of, this, skirt...]	[silhouette, length, skirt, length, flattering...

Let's take a look at the product categories again

In [259...]

```
data.product_category.value_counts()
```

Out [259...]

product_category	count
Bottoms	685
Dresses	681
Tops	680
Jackets	680
Intimate	650

Name: count, dtype: int64

Let's combine all the tokens used in reviews for the Tops

In [260...]

```
tops_tokens = []
data_tops = data[data.product_category == "Tops"]

for x in data_tops.cleaned_tokens:
    tops_tokens.extend(x)

print(f"There are {len(tops_tokens)} reviews about the 'Tops' category")
```

There are 18527 reviews about the 'Tops' category

Let's find the 20 most common words (and their frequency) in the Tops products' reviews:

In [261...]

```
freq_dist = FreqDist(tops_tokens)
freq_dist.most_common(20)
```

```
Out[261...]: [('love', 337),
 ('top', 334),
 ('wear', 233),
 ('great', 229),
 ('size', 211),
 ('color', 185),
 ('shirt', 172),
 ('fit', 167),
 ('im', 149),
 ('perfect', 146),
 ('small', 141),
 ('like', 140),
 ('soft', 136),
 ('flattering', 128),
 ('little', 120),
 ('one', 119),
 ('sweater', 118),
 ('fits', 112),
 ('bought', 111),
 ('well', 109)]
```

```
In [262...]: def find_most_common_words(data, col, category, n=20):
    """
        Takes as input the dataframe, and column, and returns the most common words
    """
    top_tokens = []
    data_category = data[data[col] == category]

    for x in data_category.cleaned_tokens:
        top_tokens.extend(x)

    freq_dist = FreqDist(top_tokens)

    return freq_dist.most_common(n)
```

Now your turn: Find the 20 most common words in the `Dresses` products' reviews

```
In [263...]: # Your code here
find_most_common_words(data, "product_category", "Dresses")
```

```
Out[263...]: [('dress', 1062),  
 ('like', 321),  
 ('fabric', 254),  
 ('would', 220),  
 ('size', 207),  
 ('fit', 207),  
 ('back', 166),  
 ('look', 157),  
 ('really', 154),  
 ('love', 152),  
 ('im', 146),  
 ('ordered', 146),  
 ('material', 139),  
 ('small', 128),  
 ('looks', 119),  
 ('looked', 118),  
 ('much', 111),  
 ('also', 106),  
 ('top', 105),  
 ('wear', 105)]
```

►  **Solution (click here to reveal)**

Task 5: Extracting the adjectives used in the reviews

In this task, we employ the NLTK PoS tagger (**Part of Speech** tagger) to assign grammatical roles to the tokens within the reviews. By doing so, we extract only the adjectives present in the reviews. Then, we create a visual representation of these adjectives using a word cloud. This approach offers a visual summary of the descriptive terms used in the reviews, providing initial insights into customer sentiments and preferences.

- **Part of Speech (POS)** is the grammatical role of a word in a sentence. A part of speech is one of the nine types of English words: VERB, NOUN, ADJECTIVE, ADVERB, PRONOUN, PREPOSITION, DETERMINER, CONJUNCTION, INTERJECTION.
- PoS tagging is a helpful tool for getting useful info from text. By looking at adjectives and adverbs, we can understand the feelings in the text. If we focus on nouns, we can see what things people are talking about.

```
In [264...]: data.head()
```

Out [264...]

	product_review	product_category	product_review_tokenized	cleaned_tokens
0	i love love love this jumpsuit its fun nice an...	Bottoms	[i, love, love, love, this, jumpsuit, its, fun...]	[love, love, love, jumpsuit, fun, nice, fabulo...]
1	beautifully made pants and on trend with the f...	Bottoms	[beautifully, made, pants, and, on, trend, wit...]	[beautifully, made, pants, trend, flared, crop...]
2	i never would have given these pants a second ...	Bottoms	[i, never, would, have, given, these, pants, a...]	[never, would, given, pants, second, look, onl...]
3	these pants are even better in person the only...	Bottoms	[these, pants, are, even, better, in, person, ...]	[pants, even, better, person, downside, need, ...]
4	the silhouette and length of this skirt and le...	Bottoms	[the, silhouette, and, length, of, this, skirt...]	[silhouette, length, skirt, length, flattering...]

Part of Speech Tagging

Part of Speech: The grammatical role of a word in a sentence. A part of speech is one of the nine types of English words: VERB, NOUN, ADJECTIVE, ADVERB, PRONOUN, PREPOSITION, DETERMINER, CONJUNCTION, INTERJECTION.

(Optional) If needed, one could see more information about these using:

```
nltk.download('tagsets')
nltk.help.upenn_tagset()
```

Now let's see how PoS tagging works:

In [265...]

```
# Just printing the first review
print( data.product_review[0] )

# Tagging each token of the first review with nltk.pos_tag
nltk.pos_tag(data.product_review_tokenized[0])
```

```
i love love love this jumpsuit its fun nice and fabulous every time i wear
it i get nothing but great compliments
```

```
Out[265...]: [('i', 'NN'),  
 ('love', 'VBP'),  
 ('love', 'NN'),  
 ('love', 'NN'),  
 ('this', 'DT'),  
 ('jumpsuit', 'NN'),  
 ('its', 'PRP$'),  
 ('fun', 'NN'),  
 ('nice', 'JJ'),  
 ('and', 'CC'),  
 ('fabulous', 'JJ'),  
 ('every', 'DT'),  
 ('time', 'NN'),  
 ('i', 'NN'),  
 ('wear', 'VBP'),  
 ('it', 'PRP'),  
 ('i', 'JJ'),  
 ('get', 'VBP'),  
 ('nothing', 'NN'),  
 ('but', 'CC'),  
 ('great', 'JJ'),  
 ('compliments', 'NNS')]
```

Now, let's create a new column '`'POS_tokens'`', and use the PoS-tagger to assign *Part of Speech* to all the tokens of all of the reviews.

```
In [266...]: # Your code here  
def do_pos_tagging(text):  
    return nltk.pos_tag(text)  
  
# data["POS_tokens"] = data["product_review_tokenized"].apply(do_pos_tagg  
data["POS_tokens"] = data["product_review_tokenized"].apply(nltk.pos_tag)
```

```
In [267...]: data.sample(5)
```

Out [267...]

	product_review	product_category	product_review_tokenized	cleaned_token
1448	i used my birthday month discount to treat mys...	Intimate	[i, used, my, birthday, month, discount, to, t...]	[used, birthda...month, discoun...treat, gorge.
1539	they make your butt look great very whimsical ...	Intimate	[they, make, your, butt, look, great, very, wh...]	[make, but...look, grea...whimsica...design
2572	i found this blouse to be just a lovely spring...	Tops	[i, found, this, blouse, to, be, just, a, love...]	[found, blouse...lovely, spring...summer, top, n.
1006	no no noi dont know if it was the way it got f...	Dresses	[no, no, noi, dont, know, if, it, was, the, wa...]	[noi, dont, know...way, go...flattened...packagi.
881	got this in the blue and was so excited while ...	Dresses	[got, this, in, the, blue, and, was, so, excit...]	[got, blue...excited, look...cute, someone...cle.

▶ **Hint (click here to expand)**

▶ **Solution (click here to reveal)**

Let's extract the adjectives used in each review

In [268...]

```
def extract_adj(tokens):
    # Takes a list of (word, POS_tag) tuples and returns only the words
    # whose POS tags correspond to adjectives (JJ, JJR, JJS)
    adjectives = []
    for tok, tag in tokens:
        if tag in ["JJ", "JJR", "JJS"]:
            adjectives.append(tok)
    return adjectives

# Create a new column 'adjectives' by applying the function extract_adj to
data["adjectives"] = data.POS_tokens.apply(extract_adj)
data.head()
```

Out [268...]

	product_review	product_category	product_review_tokenized	cleaned_tokens
0	i love love love this jumpsuit its fun nice an...	Bottoms	[i, love, love, love, this, jumpsuit, its, fun...]	[love, love, love, jumpsuit, fun, nice, fabulo...]
1	beautifully made pants and on trend with the f...	Bottoms	[beautifully, made, pants, and, on, trend, wit...]	[beautifully, made, pants, trend, flared, crop...]
2	i never would have given these pants a second ...	Bottoms	[i, never, would, have, given, these, pants, a...]	[never, would, given, pants, second, look, onl...]
3	these pants are even better in person the only...	Bottoms	[these, pants, are, even, better, in, person, ...]	[pants, even, better, person, downside, need, ...]
4	the silhouette and length of this skirt and le...	Bottoms	[the, silhouette, and, length, of, this, skirt...]	[silhouette, length, skirt, length, flattering...]

Let's combine all the `adjectives` for the `Tops` category in one big string:

In []:

In [269...]

```
adj_tops = ""
data_tops = data[data.product_category == "Tops"]

for x in data_tops.adjectives:
    adj_tops += " ".join(x)

# print(adj_tops)
```

Let's visualize the adjectives `adj_tops` using a wordcloud

In [270...]

```
word_cloud = WordCloud(width=800, height=600, background_color='white').g

plt.imshow(word_cloud)
plt.axis('off')
plt.show()
```



Now your turn: Write a python function `visualize_adjectives(category)` that gets a product category name and combine the adjectives used in the reviews related to the input category and visualize them using a word cloud.

Then, try calling your function using `visualize_adjectives("Dresses")`.

```
In [271]: data.head()
```

	product_review	product_category	product_review_tokenized	cleaned_tokens
0	i love love love this jumpsuit its fun nice an...	Bottoms	[i, love, love, love, this, jumpsuit, its, fun...]	[love, love, love, jumpsuit, fun, nice, fabulo...]
1	beautifully made pants and on trend with the f...	Bottoms	[beautifully, made, pants, and, on, trend, wit...]	[beautifully, made, pants, trend, flared, crop...]
2	i never would have given these pants a second ...	Bottoms	[i, never, would, have, given, these, pants, a...]	[never, would, given, pants, second, look, onl...]
3	these pants are even better in person the only...	Bottoms	[these, pants, are, even, better, in, person, ...]	[pants, even, better, person, downside, need, ...]
4	the silhouette and length of this skirt and le...	Bottoms	[the, silhouette, and, length, of, this, skirt...]	[silhouette, length, skirt, length, flattering...]

```
In [272...]: # Your code here
def visualize_adjectives(data, col, category, specs):
    adj_category = ""
    if not category:
        data_category = data
    else:
        data_category = data[data[col] == category]

    for x in data_category.adjectives:
        adj_category += " ".join(x)

    word_cloud = WordCloud(width=specs["width"], height = specs["height"])
    plt.imshow(word_cloud)
    plt.axis("off")
    plt.show()

specs = {"width":1000, "height":1000,"background_color":"white"}
visualize_adjectives(data, "product_category", "Dresses", specs)
```



► **Solution** (click here to reveal)

Task 6

Finding the sentiment of each review

In this task, the NLTK sentiment analyzer is used to assign sentiment scores to individual text reviews. Polarity scores are calculated and visualized to compare sentiment across product categories. Based on this analysis, we can find out which category shows the highest number of reviews with negative sentiment.

```
In [273]: data.head()
```

Out[273...]

	product_review	product_category	product_review_tokenized	cleaned_tokens
0	i love love love this jumpsuit its fun nice an...	Bottoms	[i, love, love, love, this, jumpsuit, its, fun...]	[love, love, love, jumpsuit, fun, nice, fabulo...]
1	beautifully made pants and on trend with the f...	Bottoms	[beautifully, made, pants, and, on, trend, wit...]	[beautifully, made, pants, trend, flared, crop...]
2	i never would have given these pants a second ...	Bottoms	[i, never, would, have, given, these, pants, a...]	[never, would, given, pants, second, look, onl...]
3	these pants are even better in person the only...	Bottoms	[these, pants, are, even, better, in, person, ...]	[pants, even, better, person, downside, need, ...]
4	the silhouette and length of this skirt and le...	Bottoms	[the, silhouette, and, length, of, this, skirt...]	[silhouette, length, skirt, length, flattering...]

Here is an example showing how to find the sentiment of a review:

In [274...]

Example

```

sent = SentimentIntensityAnalyzer() # We imported this earlier from nltk.

# Let's pick the 1st (cleaned) review as an example:
review = data.product_review_cleaned[0]
print(review)

# Get the sentiment scores for that review
scores = sent.polarity_scores(review)
scores

```

love love love jumpsuit fun nice fabulous every time wear get nothing great compliments

Out[274...]

Sentiment scores:

- pos : The probability of positive sentiment
- neu : The probability of neutral sentiment
- neg : The probability of negative sentiment
- compound : The normalized compound score that takes values from -1 to 1

We can use the compound score to find the sentiment of each review.

- if compound score >= 0.05 then positive

- if compound score between -0.05 and 0.05 then `neutral`
- if compound score <=-0.05 then `negative`

Now let's create a method to find the sentiment of a review using the compound score

```
In [275...]: def polarity_score(review):
    # Initializing the Sentiment Analyzer
    sent = SentimentIntensityAnalyzer()

    # Extracting the sentiment polarity scores of a review
    scores = sent.polarity_scores(review)

    # Getting the compound score
    compound = scores['compound']

    if compound > 0.05:
        return "positive"
    elif compound < -0.5:
        return "negative"
    else:
        return "neutral"
```

```
In [276...]: # Example:

s1 = polarity_score("this product is amazing the quality is great")
print(s1)

s2 = polarity_score("this product is available in my country")
print(s2)

s3 = polarity_score("this cheap product has a very poor quality and did n")
print(s3)
```

positive
neutral
negative

Now, create a column called '`'sentiment'`' in your dataframe to label each review with a sentiment (positive, neutral, or negative) :

```
In [277...]: data["sentiment"] = data["product_review_cleaned"].apply(polarity_score)
```

```
In [278...]: data.sample(5)
```

Out [278...]

	product_review	product_category	product_review_tokenized	cleaned_toker
2152	perfect trans top skinnies or boyfriend and b...	Tops	[perfect, trans, top, skinnies, or, boyfriend,...	[perfect, tran top, skinnie boyfriend, boo.
526	softest denim they feel like theyve been thro...	Bottoms	[softest, denim, they, feel, like, theyve, bee...	[softest, denim feel, like, theyve wash, hun.
2207	i love this shirt it would be a fairly basic t...	Tops	[i, love, this, shirt, it, would, be, a, fairl...	[love, shir would, fair basic, tec neckl.
956	i wanted to love this dress it looks beautiful...	Dresses	[i, wanted, to, love, this, dress, it, looks, ...	[wanted, lov dress, look beautifu model.,
2065	great shirt to wear with white pants and wedge...	Tops	[great, shirt, to, wear, with, white, pants, a...	[great, shir wear, whit pants, wedge im.,

In [279...]

Your code here

- ▶💡 Hint ([click here to expand](#))
- ▶✅ Solution ([click here to reveal](#))

Now let's group the products by category and sentiment, and visualize the number of products in each group using a bar chart.

In [280...]

```
df = data.groupby(["product_category", "sentiment"]).size().reset_index()
df.head(10)
```

Out [280...]

	product_category	sentiment	counts
0	Bottoms	negative	3
1	Bottoms	neutral	5
2	Bottoms	positive	677
3	Dresses	negative	36
4	Dresses	neutral	82
5	Dresses	positive	563
6	Intimate	negative	5
7	Intimate	neutral	35
8	Intimate	positive	610
9	Jackets	negative	2

In [281...]

```
px.bar(df, x="product_category", y="counts", color="sentiment", barmode="
```

Exercise

In this exercise, a dataset `reviews.csv` is provided with 15000 review texts along with the corresponding sentiment for each review. Your goal is simply to assist the advertisement team of the company in creating a word cloud visualization specifically focusing on the adjectives used in reviews with positive sentiments.

Your task:

- Filter out the reviews with a positive sentiment.
- Apply PoS tagging to extract adjectives used in the reviews.
- Create a word cloud visualization for the extracted adjectives.

Importing modules:

In [282...]

```
import pandas as pd
import string

# Importing Natural Language Processing toolkit
import nltk

# Downloading the NLTK english stop words
nltk.download('stopwords')

# Importing data visualization modules
from wordcloud import WordCloud

print("Modules are imported! :)"
```

Modules are imported! :)

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/ahmedhanif/nltk_data...
[nltk_data]     Package stopwords is already up-to-date!
```

In [283...]

```
def preprocess_for_sentiment_analysis(df, col):
    df[col] = df[col].apply(to_lower_case).apply(remove_punctuation)
    df["tokenized"] = df[col].apply(tokenize)
    df["tokenized_clean"] = df["tokenized"].apply(clean_tokens).apply(lambda x: " ".join(x))
    return df
```

Preparing the data:

In []:

In [284...]

```
# loading the dataset "reviews.csv"
# TODO ...
# convert reviews to lower case
# TODO ...

# remove the punctuation
# TODO ...

# tokenization
# TODO ...
```

```
# display the dataframe (to check)
# TODO ...
```

In [285...]

```
data = pd.read_csv("./reviews.csv")
data.sample(5)
```

Out [285...]

	Sentiment	Review
1095	Negative	Troma should feel ashamed by the horrible qual...
11615	Negative	Protection was my intention but I ended up car...
11813	Positive	This was actually better than what most will t...
11891	Positive	This is a quick read and fast moving book. Don...
10145	Positive	We bought this DVD[ASIN:B00005JPH2 The Chronic...

Keep only the reviews with a positive sentiment:

In [286...]

```
# write your code here
positive_reviews_df = data[data.Sentiment == "Positive"]
positive_reviews_df.sample(5)
```

Out [286...]

	Sentiment	Review
490	Positive	these guys were the cream of the doom metal cr...
28	Positive	This book by Vincent Bugliosi is by far the be...
3454	Positive	There is so much about this book to comment on...
8770	Positive	"Of Mice And Men" is a terrific book! Especial...
1462	Positive	Its sad that the movie will only last a week o...

In [287...]

```
positive_reviews_df = preprocess_for_sentiment_analysis(positive_reviews_
```

In [288...]

```
positive_reviews_df.head()
```

Out [288...]

	Sentiment	Review	tokenized	tokenized_clean
0	Positive	this is the wonderfully engaging tale of how b...	[this, is, the, wonderfully, engaging, tale, o...	wonderfully engaging tale bugliosi successfull...
1	Positive	charles manson is part of the american conscie...	[charles, manson, is, part, of, the, american,...	charles manson part american conscience course...
2	Positive	helter skelter motive is a little far out real...	[helter, skelter, motive, is, a, little, far, ...	helter skelter motive little far real reason m...
4	Positive	i was 16 years old the summer of 1969 and reme...	[i, was, 16, years, old, the, summer, of, 1969...	16 years old summer 1969 remember vividly didn...
5	Positive	i had no idea the manson family was so much mo...	[i, had, no, idea, the, manson, family, was, s...	idea manson family much widespread one murder ...

Use the `nltk.pos_tag` function to find the role of the tokens in the sentences:

In [289...]

```
# write your code here
positive_reviews_df["POS_tokens"] = positive_reviews_df["tokenized"].apply(
```

Extract all the adjectives from the reviews with positive sentiment:

In [290...]

```
# write your code here
positive_reviews_df["adjectives"] = positive_reviews_df.POS_tokens.apply(
```

In [291...]

```
positive_reviews_df.head()
```

Out [291...]	Sentiment	Review	tokenized	tokenized_clean	POS_tokens	adjectiv
0	Positive	this is the wonderfully engaging tale of how b...	[this, is, the, wonderfully, engaging, tale, o...	wonderfully engaging tale bugliosi successfull...	[(this, DT), (is, VBZ), (the, DT), (wonderfull...	[engagir buglio lawenforcement bette
1	Positive	charles manson is part of the american conscie...	[charles, manson, is, part, of, the, american,...	charles manson part american conscience course...	[(charles, NNS), (manson, NN), (is, VBZ), (par...	[american scary, i, su good, oth les
2	Positive	helter skelter motive is a little far out real...	[helter, skelter, motive, is, a, little, far, ...	helter skelter motive little far real reason m...	[(helter, NN), (skelter, NN), (motive, NN), (i...	[little, real, ga
4	Positive	i was 16 years old the summer of 1969 and reme...	[i, was, 16, years, old, the, summer, of, 1969...	16 years old summer 1969 remember vividly didn...	[(i, NN), (was, VBD), (16, CD), (years, NNS), ...	[old, first, mai most, oth many, hot, st
5	Positive	i had no idea the manson family was so much mo...	[i, had, no, idea, the, manson, family, was, s...	idea manson family much widespread one murder ...	[(i, NN), (had, VBD), (no, DT), (idea, NN), (t...	[mansc widespread great, variou riveting

Use a word cloud to visualize all the adjectives used in the positive reviews:

In [292...]

```
# write your code here
specs = {"width":1000, "height":1000,"background_color":"white"}
visualize_adjectives(positive_reviews_df, "adjectives", "", specs)
```



Negative Sentiments

```
In [304... negative_reviews_df = preprocess_for_sentiment_analysis(data[data["Sentiment"] == "Negative"])
negative_reviews_df[["POS_tokens"]] = negative_reviews_df[["tokenized"]].apply(lambda x: x[0])
specifications = {"width":1000, "height":1000, "background_color": "white"}
```

```
In [305... negative_reviews_df[["adjectives"]] = negative_reviews_df.POS_tokens.apply(lambda x: x[0])
```

```
In [306... negative_reviews_df.head()
```

Out[306...]

Sentiment	Review	tokenized	tokenized_clean	POS_tokens	adjectives
3 Negative	what permissive issues could ban all photosa c...	[what, permissive, issues, could, ban, all, ph...	permissive issues could ban photosa crock scam...	[(what, WP), (permissive, JJ), (issues, NNS), ...]	[permissive, full, used, gratifying, dont]
6 Negative	have any of you not read the manson file it is...	[have, any, of, you, not, read, the, manson, f...	read manson file obvious control people lsd ob...	[(have, VBP), (any, DT), (of, IN), (you, PRP),...]	[obvious, obvious]
7 Negative	the fact of the matter is this book is total f...	[the, fact, of, the, matter, is, this, book, i...	fact matter book total fiction elaborate work ...	[(the, DT), (fact, NN), (of, IN), (the, DT), (...	[total, elaborate, vincent, due, superstitious...]
8 Negative	i realize that this is the first book a person...	[i, realize, that, this, is, the, first, book,...	realize first book person come searching manso...	[(i, NN), (realize, VBP), (that, IN), (this, D...]	[first, egotistical, better, much, available, o...]
9 Negative	bugliosis ghost writer for this one crafted a ...	[bugliosis, ghost, writer, for, this, one, cra...	bugliosis ghost writer one crafted skillful ta...	[(bugliosis, NN), (ghost, NN), (writer, NN), (...	[skillful, real, same, ridiculous, hard, better]

```
In [307]: visualize_adjectives(negative_reviews_df, "adjectives", "", specs) #####
```



In []:

In []:

In []: