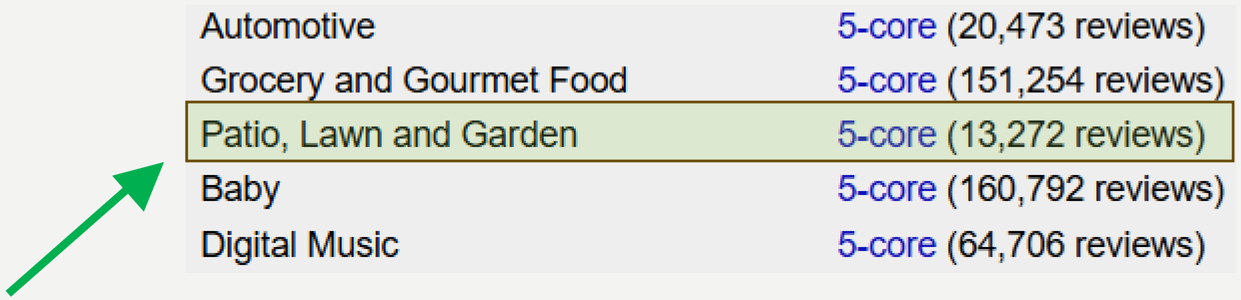
PA4

- Due at 11:59 PM on Apr 18, 2025
- The Jupyter notebook containing the code should be submitted on Blackboard in the Assignments section
- If there is any ambiguity or contradiction or confusion that you may have while working on the assignment, feel free to ask questions and clarifications in class or on Teams or discuss individually with Danish.
- Please do not post your code or code-snippet on Teams
- This is an individual assignment and focuses on data frames, data transformation and data visualization using Python.

DOWNLOADING, AND READING REVIEW DATA AND CHANGING COLUMN NAMES



Automotive	5-core (20,473 reviews)
Grocery and Gourmet Food	5-core (151,254 reviews)
Patio, Lawn and Garden	5-core (13,272 reviews)
Baby	5-core (160,792 reviews)
Digital Music	5-core (64,706 reviews)

- Download 5-core Patio, Lawn and Garden review data for office products from the following website: https://cseweb.ucsd.edu/~jmcauley/datasets/amazon/links.html

- Read the data from the file **reviews_Patio_Lawn_and_Garden_5.json.gz** into a Pandas data frame and change the names of the ***reviewText*** and ***overall*** columns to ***comment*** and ***rating***, respectively.

References:

Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering
R. He, J. McAuley
WWW, 2016
Image-based recommendations on styles and substitutes
J. McAuley, C. Targett, J. Shi, A. van den Hengel
SIGIR, 2015

ADDING MONTH, YEAR AND WORD COUNT COLUMNS

- Add the following columns in the data frame:
  - ***month***
    - Holds the values of the months extracted from the dates in the column ***reviewTime***.
  - ***year***
    - Holds the values of the years extracted from the dates in the column ***reviewTime***.
  - ***word_count***
    - Holds the values of the number of words of the textual comments in the column ***comment***. The number of words for each comment are calculated after all the characters except alphabets and blank spaces have been removed from the comments. For this, you can utilize regular expressions to first remove all characters except alphabets and white-spaces and then replace every group of contiguous white-spaces with just one white-space each. Finally remove the leading and trailing white-spaces. There may be some comments with no words. Such comments will have a word count of 0.

DESCRIPTIVE STATISTICS
MONTH, YEAR AND WORD COUNT COLUMNS

- Display the descriptive statistics of the ***month***, ***year***, ***rating*** and ***word_count*** columns (rounded to 2 decimals) as shown below:

	count	**mean**	**std**	**min**	**25%**	**50%**	**75%**	**max**
month	13272.0	6.07	2.70	1.0	4.0	6.0	8.0	12.0
year	13272.0	2012.59	1.53	2000.0	2012.0	2013.0	2014.0	2014.0
rating	13272.0	4.19	1.08	1.0	4.0	5.0	5.0	5.0
word_count	13272.0	155.53	148.35	0.0	60.0	112.0	202.0	2242.0

NUMBER OF REVIEWS BY MONTH

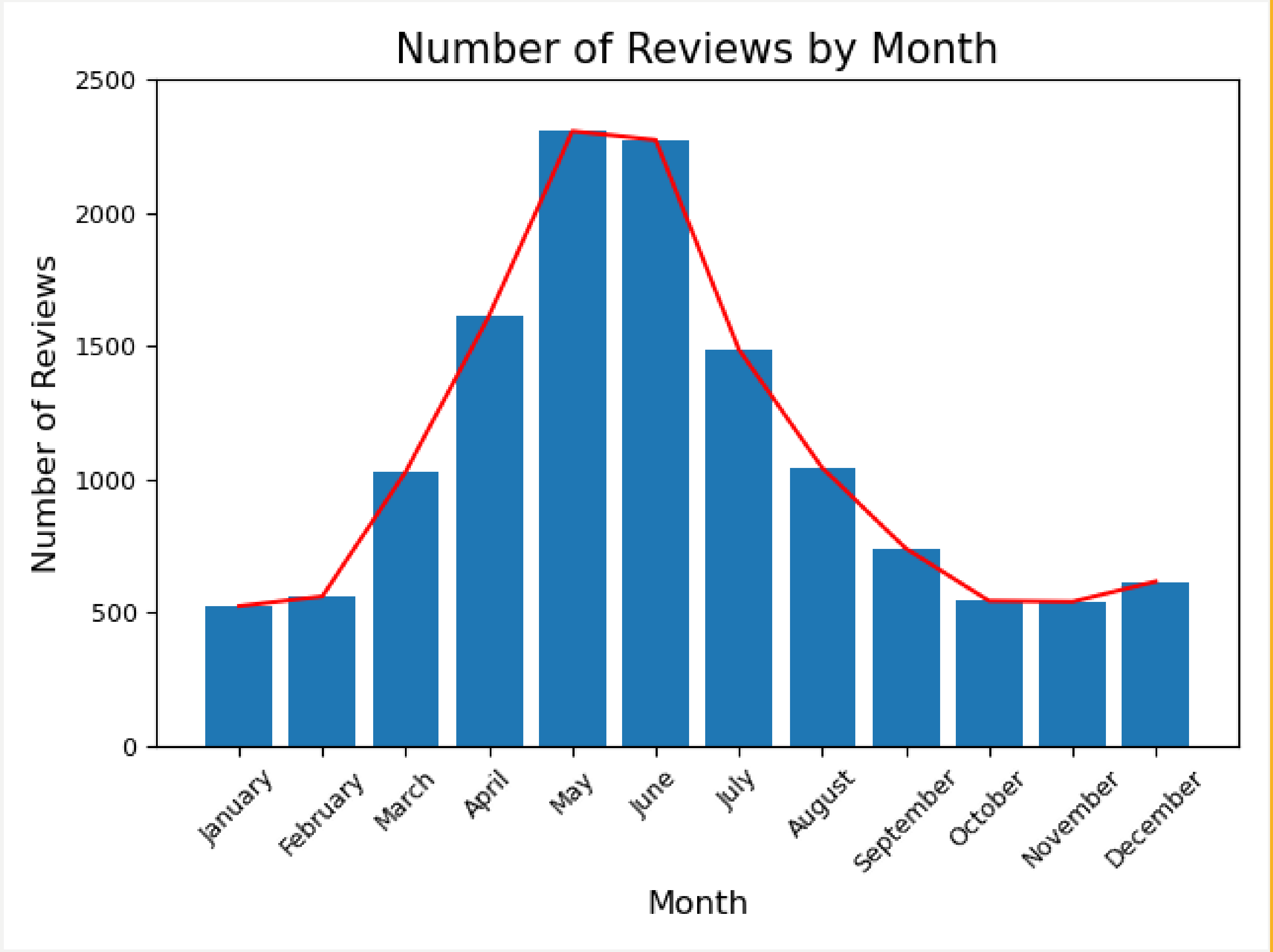
- Plot a combination of bar and line chart depicting the total number of reviews across the months and display/print a dataframe that depicts the number of reviews across the months as shown on the next slide. You can utilize matplotlib’s line and bar functions to create the plot.

- Each bar/line marker represents the total number of reviews across all the years for each month. For instance, in the review data, a total of 1486 reviews are from the Julys across all the years. The month names on the x-axis are rotated by 45 degrees.

- You can begin by creating a data frame with two columns, one storing the values of month numbers (1 to 12) and the other storing the total number of reviews for each month. The column *month_name* depicting the month names for the corresponding month (integer) should be inserted next to the column *month*. The data in the dataframe is sorted in ascending order of the month numbers (1 - 12).
  - To create the *month_name* column you can import and utilize the calendar package. For example, the output of calendar.month_name[3] is March.

NUMBER OF REVIEWS BY MONTH

	month	month_name	count
0	1	January	523
1	2	February	560
2	3	March	1028
3	4	April	1613
4	5	May	2307
5	6	June	2274
6	7	July	1486
7	8	August	1042
8	9	September	740
9	10	October	543
10	11	November	540
11	12	December	616



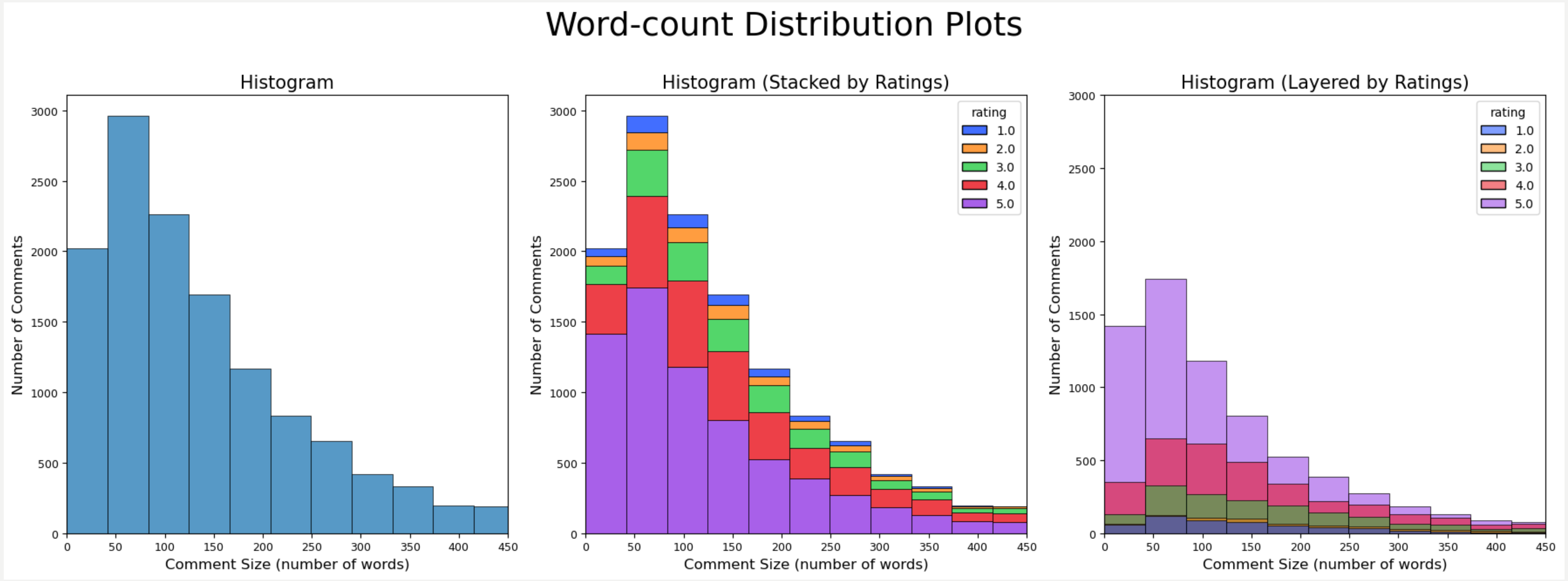
WORD-COUNT DISTRIBUTION

- Plot 3 histogram subplots to depict the distribution of *word_count* values as shown on the next slide. You can utilize seaborn's **histplot** function to create the plots. The histograms should be created as subplots in a matplotlib figure with 1 row and 3 columns. The value of the bins parameter in the histplot function should be set to 54.

- Utilize a for loop and appropriate if-elif-else conditions to set the values for parameters hue, multiple, alpha and palette of the histplot function so that the 3 subplots are created with just one call to the histplot function in the for loop. Set the titles of the subplots in these if-elif-else conditions as well. All of the features of the subplots should be programmed dynamically in the for loop as was shown with an example in the class.

- The value for the parameter palette for the layered and stacked histograms plots is 'bright'. The value for the parameter alpha for the layered histogram plot is 0.3.

WORD-COUNT DISTRIBUTION



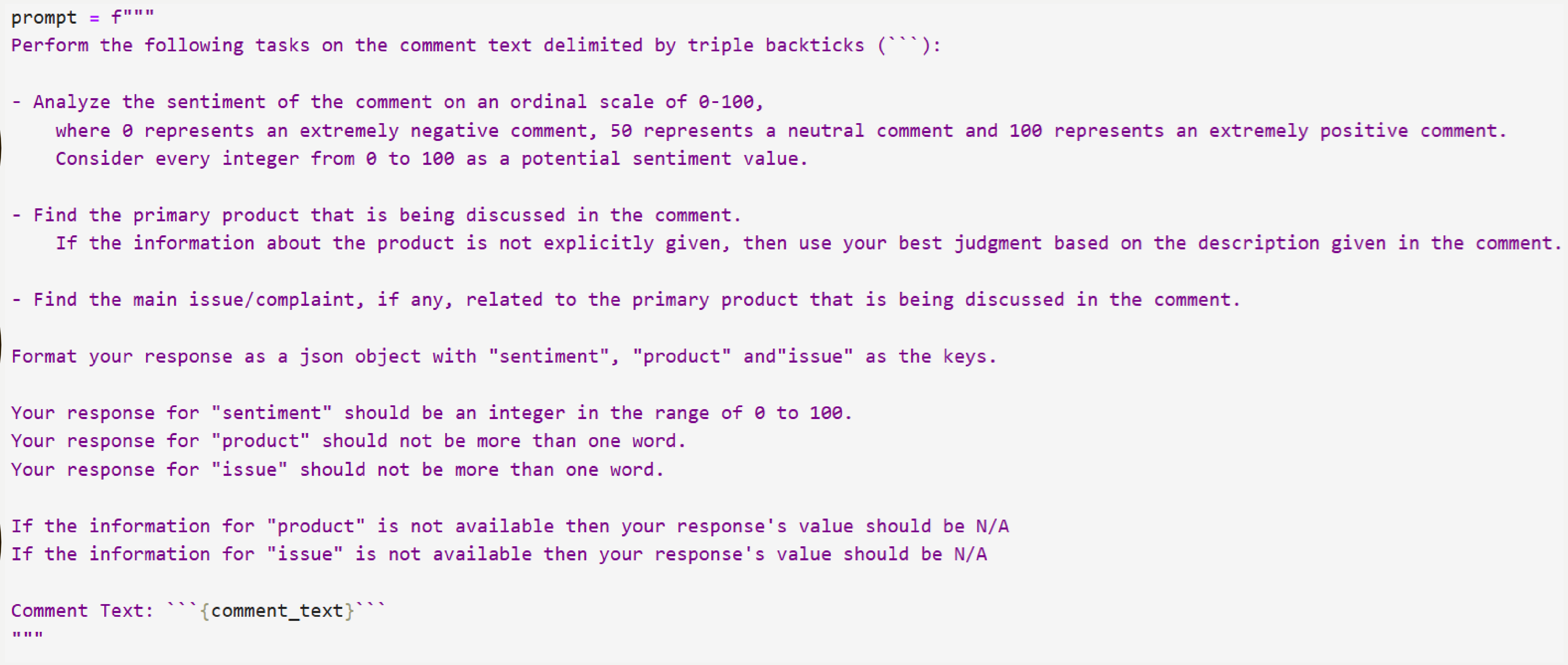
READING COMMENTS INFO

- Read “Reviews Info Data.csv” into a dataframe. This dataset contains a random sample of 300 reviews that were collected from the data you have been working with up till the previous slide.

- Apart from the columns that were already there, 5 additional columns have been added in this dataset: *sentiment*, *product, product_clean, issue* and *issue_clean*.

- The columns *sentiment*, *product* and *issue* were constructed by utilizing NLP techniques with a LLM model (GPT-4-turbo) on the comments in the *comment* column. The prompt used for the extracting the information from the comments that then was stored in these columns is given on the next slide for your reference. The columns *product_clean* and *issue_clean* were constructed by converting the values in the *product* and *issue* to lowercase and removing leading and trailing spaces (if any).

READING COMMENTS INFO



```
prompt = f"""
Perform the following tasks on the comment text delimited by triple backticks (```):

- Analyze the sentiment of the comment on an ordinal scale of 0-100,
    where 0 represents an extremely negative comment, 50 represents a neutral comment and 100 represents an extremely positive comment.
    Consider every integer from 0 to 100 as a potential sentiment value.

- Find the primary product that is being discussed in the comment.
    If the information about the product is not explicitly given, then use your best judgment based on the description given in the comment.

- Find the main issue/complaint, if any, related to the primary product that is being discussed in the comment.

Format your response as a json object with "sentiment", "product" and"issue" as the keys.

Your response for "sentiment" should be an integer in the range of 0 to 100.
Your response for "product" should not be more than one word.
Your response for "issue" should not be more than one word.

If the information for "product" is not available then your response's value should be N/A
If the information for "issue" is not available then your response's value should be N/A

Comment Text: ```{comment_text}```
"""
```

RATING SENTIMENT COMPARISON

- Create a scatterplot to compare the *sentiment* and *rating* values and print a dataframe depicting the number of reviews grouped by the *rating* (1, 2, 3, 4, 5) and *dif* (0, 1, 2, 3) categories as shown in the next slide. To create this dataframe, pandas' **crosstab** function can be utilized, and the scatterplot can be created using seaborn's **scatterplot** function.

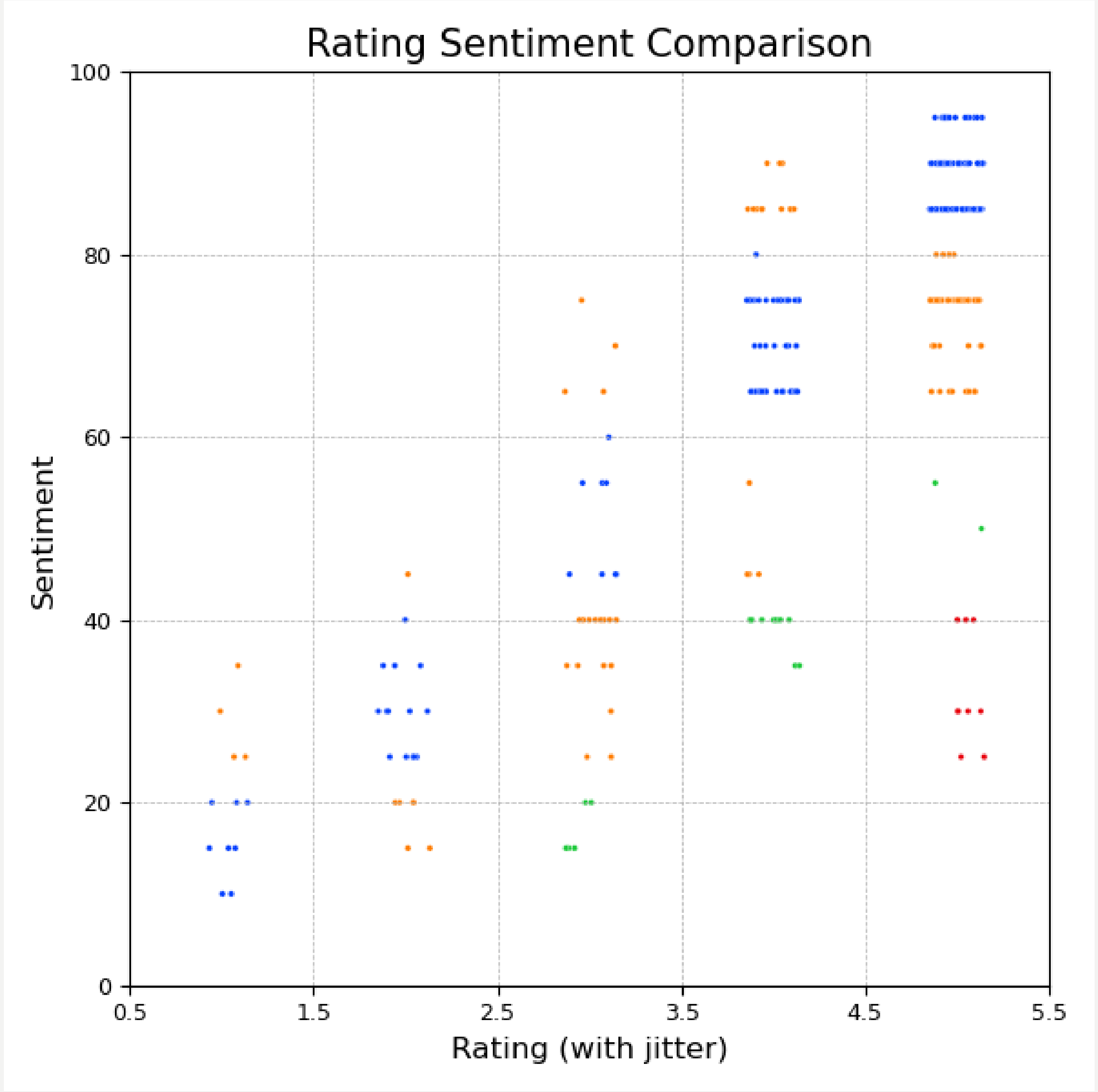
- For constructing the crosstab dataframe and scatterplot, you will create a column *dif*. The values in the column *dif* reflect the absolute differences between rating and scaled sentiment values. The scaled sentiment values are calculated by dividing the sentiment values by 20 and then calculating the ceilings (rounding up to nearest whole numbers). For example, here are the steps for scaling a sentiment value of 45: 45/20 => 2.25, math.ceil(2.25) => 3. Don't have to worry about sentiment = 0. The lowest sentiment value is 10. You can import the math package and use its ceil function. You can further refer the to table on the next slide that depicts scaled sentiment values for their corresponding sentiment ranges.

- For Rating (with jitter), you can construct and utilize a column *rating_jitter* which can be constructed by summing up the *rating* column and a collection of 300 random values drawn from a uniform distribution with a range of -0.15 to 0.15. You can use np.random.seed(42) to produce reproducible results. You can set the value of the argument palette to bright in the scatterplot function.

RATING SENTIMENT COMPARISON

Sentiment Range	Scaled Sentiment
1 <= sentiment <= 20	1
21 <= sentiment <= 40	2
41 <= sentiment <= 60	3
61 <= sentiment <= 80	4
81 <= sentiment <= 100	5

dif	0.0	1.0	2.0	3.0
rating				
1.0	8	4	0	0
2.0	14	7	0	0
3.0	8	19	5	0
4.0	48	16	12	0
5.0	92	56	2	9



SIZE & SENTIMENT DISTRIBUTIONS

- Create 2 boxplots (subplots) to depict the distributions of *word_count* and *sentiment* values (by Issues Found?) and display/print the count, mean, median values of the word counts and sentiments (by Issues Found?) as shown on the next slide. You can utilize seaborn’s **boxplot** function to create the plots. The boxplots should be created as subplots in a matplotlib figure with 1 row and 2 columns.

- For “Issues Found?”, create a column *issues_found* in which a value equals “yes” if the corresponding value in the *issue_clean* column is not missing (np.nan) otherwise it equals “no”.

- Utilize a for loop and appropriate if-else conditions to set the values for parameter y of the boxplot function so that the 2 subplots are created with just one call to the boxplot function in the for loop. Set the titles of the subplots in these if-else conditions as well. All of the features of the subplots should be programmed dynamically in the for loop as was shown with an example in the class.

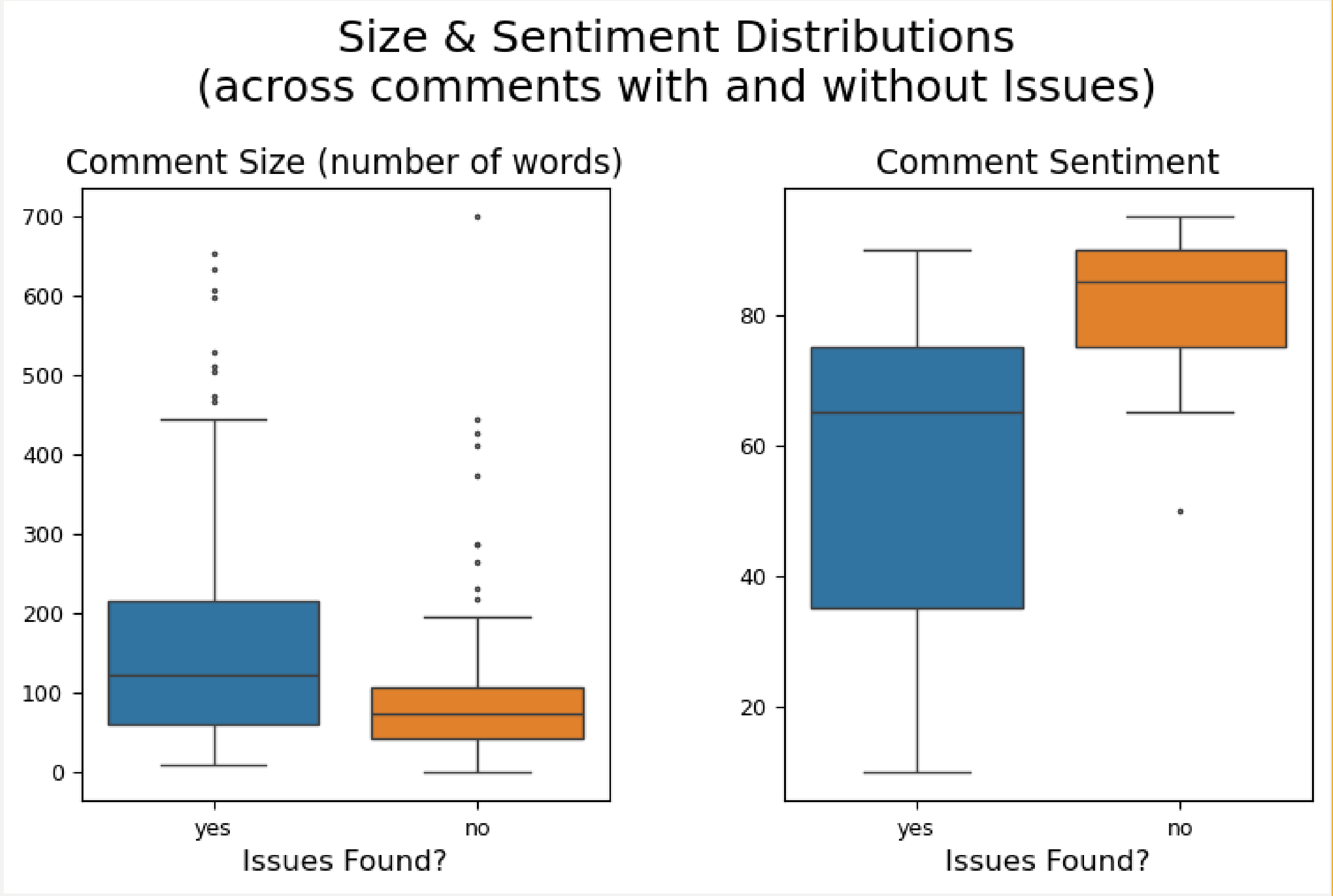
SIZE & SENTIMENT DISTRIBUTIONS

Average and median word counts (by issues found?):

	count	50%	mean
issues_found			
yes	184.0	123.0	161.29
no	116.0	72.5	99.96

Average and median sentiment (by issues found?):

	count	50%	mean
issues_found			
yes	184.0	65.0	54.84
no	116.0	85.0	83.36



RATING DISTRIBUTION (BY ISSUES FOUND?)

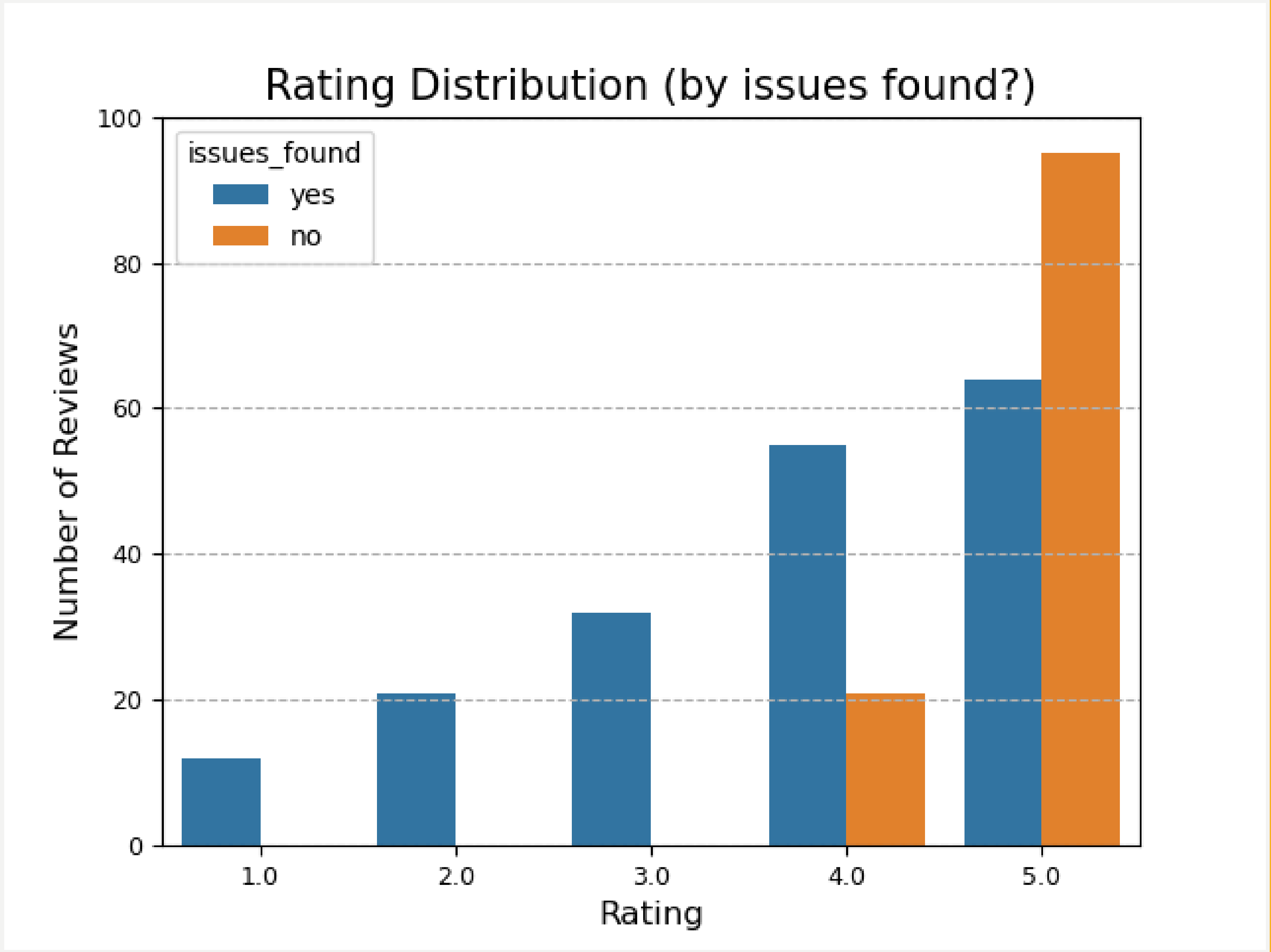
- Create a barplot (countplot) to depict *rating* distribution (by Issues Found?) and print/display a dataframe depicting the number of reviews grouped by the rating categories (1, 2, 3, 4, 5) and issues_found categories (no, yes) as shown in the next slide. To create this dataframe, pandas' **crosstab** function can be utilized.

- To display the column and row totals ("All"), set the argument margins in the crosstab function to True.

- Seaborn's **countplot** function can be used to create the plot. The values of the x and hue arguments in the countplot function should be equal to 'rating' and 'issues_found', respectively and the value of data argument should be set equal to the name of the dataframe containing the sample of 300 random reviews which you created earlier.

RATING DISTRIBUTION (BY ISSUES FOUND?)

issues_found	no	yes	All
rating			
1.0	0	12	12
2.0	0	21	21
3.0	0	32	32
4.0	21	55	76
5.0	95	64	159
All	116	184	300



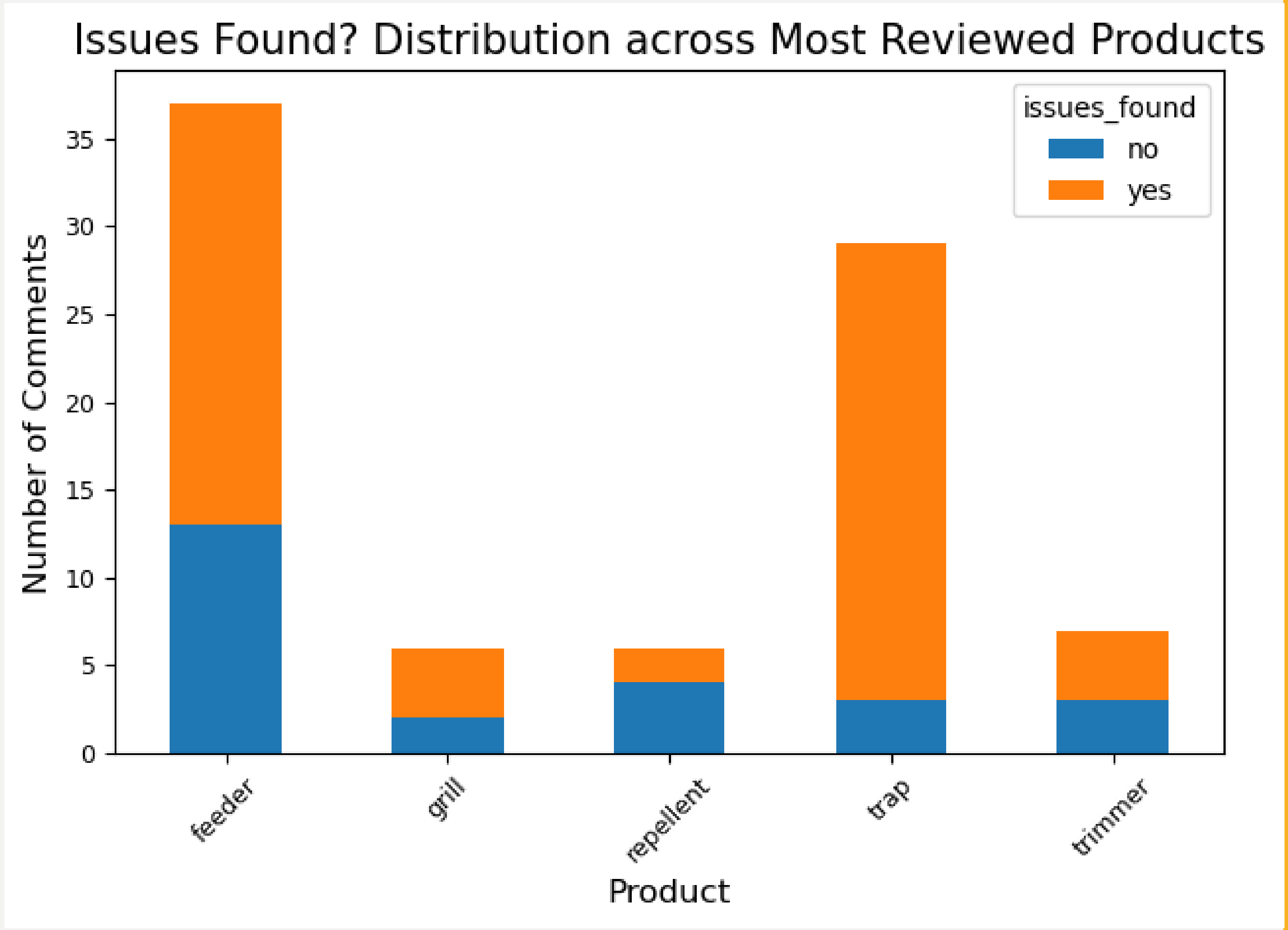
ISSUES FOUND? DISTRIBUTION ACROSS MOST REVIEWED PRODUCTS

- Create a barplot (countplot) to depict *issues_found* distribution (by *product_clean*) and print/display a dataframe depicting the number of reviews grouped by the products (5 *product_clean* values that occur most frequently in the reviews) and *issues_found* categories (no, yes) as shown on the next slide. To create this dataframe, pandas' **crosstab** function can be utilized.

- Pandas plot function can be used to create the plot. The values of the kind and stacked arguments in the plot function should be set to 'bar' and True, respectively.

ISSUES FOUND? DISTRIBUTION ACROSS MOST REVIEWED PRODUCTS

issues_found	no	yes
product_clean		
feeder	13	24
grill	2	4
repellent	4	2
trap	3	26
trimmer	3	4

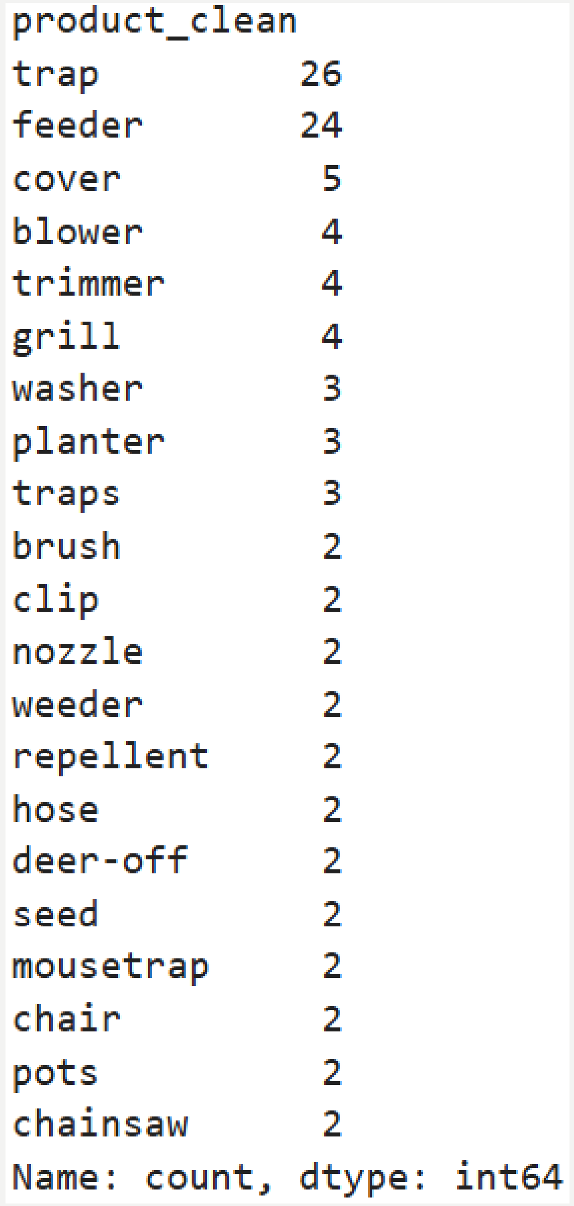


PRODUCT WORDCLOUDS

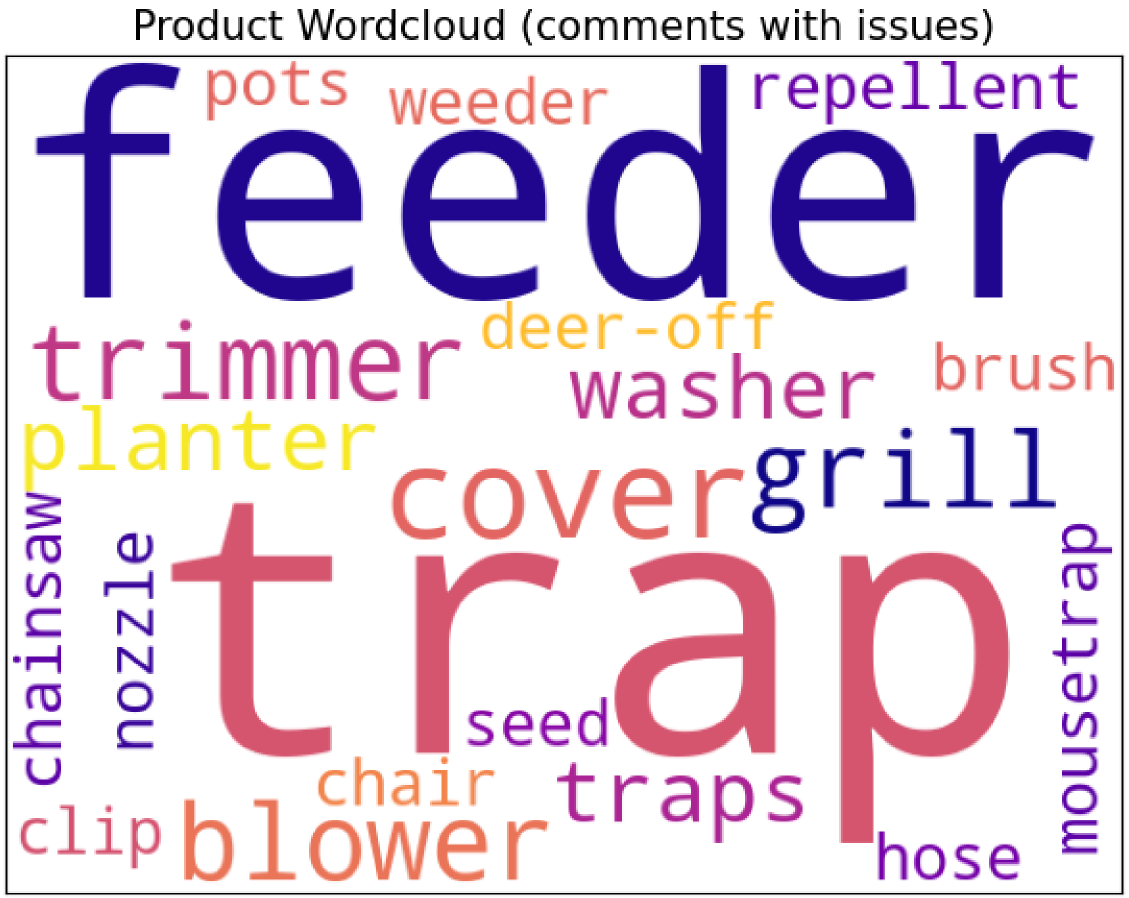
- Create 2 wordclouds depicting the frequencies of the products (*product_clean*) in comments with issues (issues_found = “yes”) and without issues (issues_found = “no”), as shown on the next two slides. Along with the wordclouds, print/display the corresponding frequencies of the products. For the wordclouds and frequency displays, keep only those products that appear at least twice in the comments.

- You can utilize the following arguments and their corresponding values for the wordclouds:
  - random_state = 42, background_color = 'white', colormap = 'plasma’, relative_scaling = 0.75, width = 600, height = 450

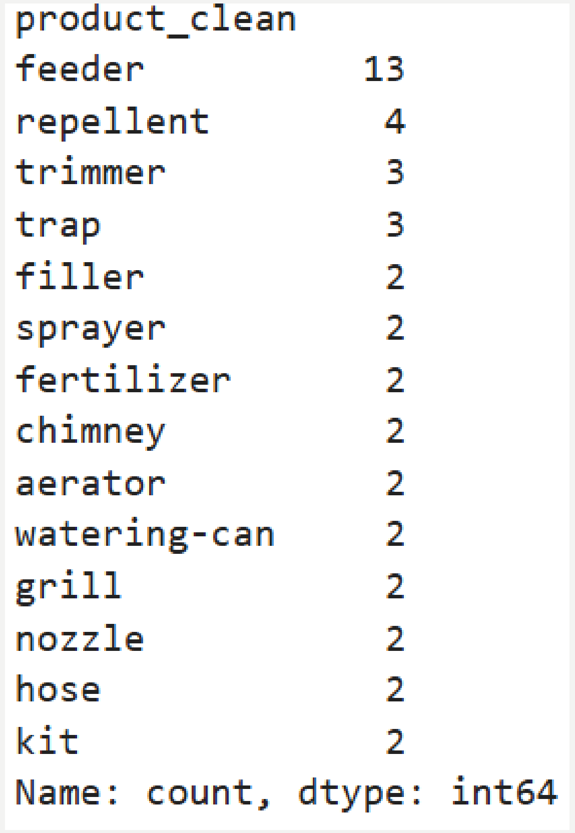
PRODUCT WORDCLOUD (COMMENTS WITH NO ISSUES FOUND)



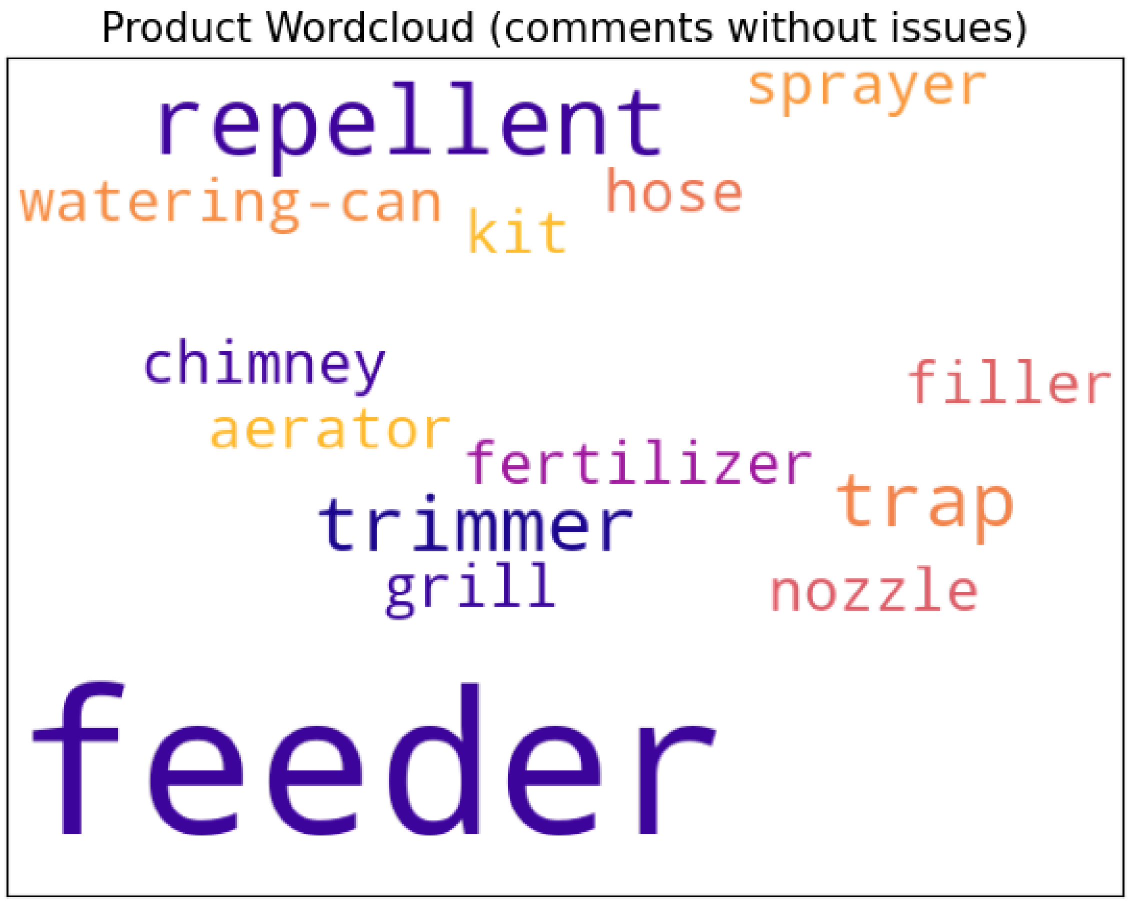
```
product_clean
trap          26
feeder        24
cover          5
blower         4
trimmer        4
grill          4
washer         3
planter        3
traps          3
brush          2
clip           2
nozzle         2
weeder         2
repellent      2
hose           2
deer-off       2
seed           2
mousetrap      2
chair          2
pots           2
chainsaw       2
Name: count, dtype: int64
```



PRODUCT WORDCLOUD (COMMENTS WITH NO ISSUES FOUND)



```
product_clean
feeder          13
repellent        4
trimmer          3
trap             3
filler           2
sprayer          2
fertilizer       2
chimney          2
aerator          2
watering-can     2
grill            2
nozzle           2
hose             2
kit              2
Name: count, dtype: int64
```



IMPORTANT NOTE

- The format, values and content of your the console outputs and plots should match the format, values and content of the console outputs and content shown in the corresponding slides (don’t worry about the spacing in the console outputs).
- Values in any of the dataframes or plots should not be hard-coded. They also should not be modified, delated or added individually or collectively using hard-coded values in a list, dictionary or any other Python data structure.
- The code for the construction of dataframe and/or columns and analysis for each of the deliverables should follow the instructions given on the corresponding slides. For those parts of the deliverables for which the instructions are not given on the slides, you will need to utilize an appropriate logic to derive the expected console outputs and plots.

PA4 GRADING (300 POINTS TOTAL)

Deliverables	Points
Adding the month, year and word_count columns	15
Descriptive statistics of month, year, rating and word_count	5
Number of reviews by month	30
Word-count distribution	35
Rating sentiment comparison	45
Size & sentiment distributions	45
Rating Distribution (by issues found)	30
Issues found? distribution across most reviewed products	35
Product Wordclouds	45
Efficient Program Structure (e.g. Code breakdown in appropriate methods, comments, markdowns)	15