“RATE MY BREW”

“INTRODUCTION TO DATA SCIENCE” FINAL PROJECT

BY REMEZ SAVOY AND BAR NAHMUKA

RESEARCH QUESTION

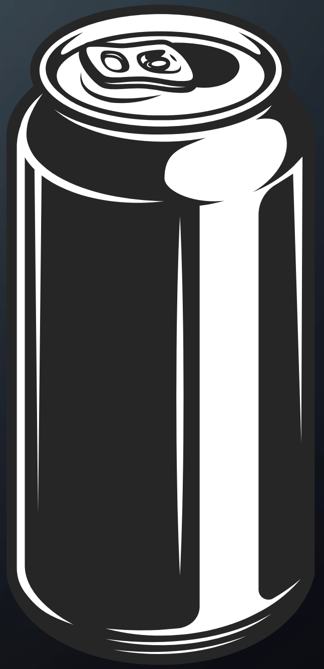
Can highly rated beers be predicted?

STAGES

- Data Acquisition
- Data Cleaning
- EDA
- Machine Learning

DATA ACQUISITION

- Scraping drizly.com
- Getting Beers ratings, countries and other beer properties

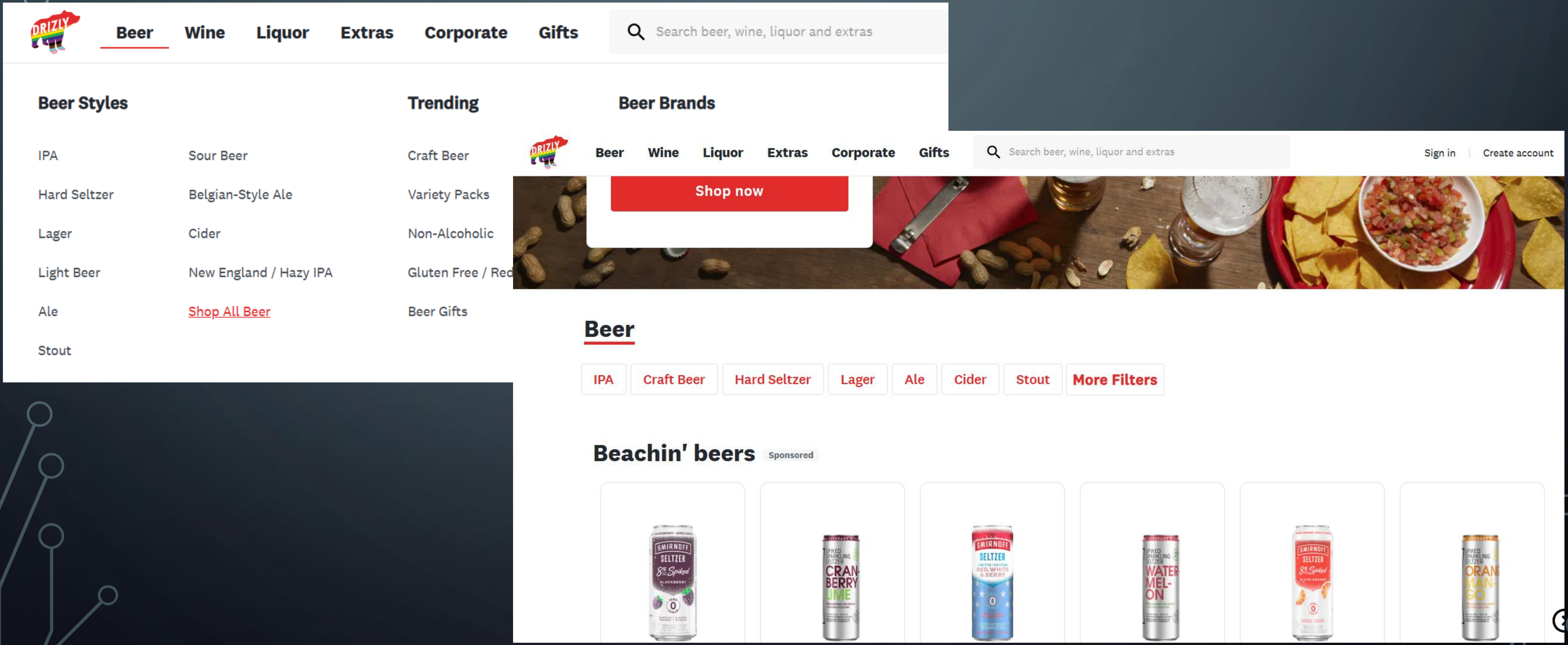


SCRAPING DRIZLY

- Gathering all beer data from drizly.com using Selenium

Work plan: Entering the beer browsing section at Drizly, and going through all beers and scraping them to a CSV file.

ENTERING THE BEER BROWSING SECTION



Beer Wine Liquor Extras Corporate Gifts

Search beer, wine, liquor and extras

Beer Styles

IPA

Hard Seltzer

Lager

Light Beer

Ale

Stout

Sour Beer

Belgian-Style Ale

Cider

New England / Hazy IPA

Shop All Beer

Trending

Craft Beer

Variety Packs

Non-Alcoholic

Gluten Free / Red

Beer Gifts

Beer Brands

Beer Wine Liquor Extras Corporate Gifts

Search beer, wine, liquor and extras

Sign in | Create account

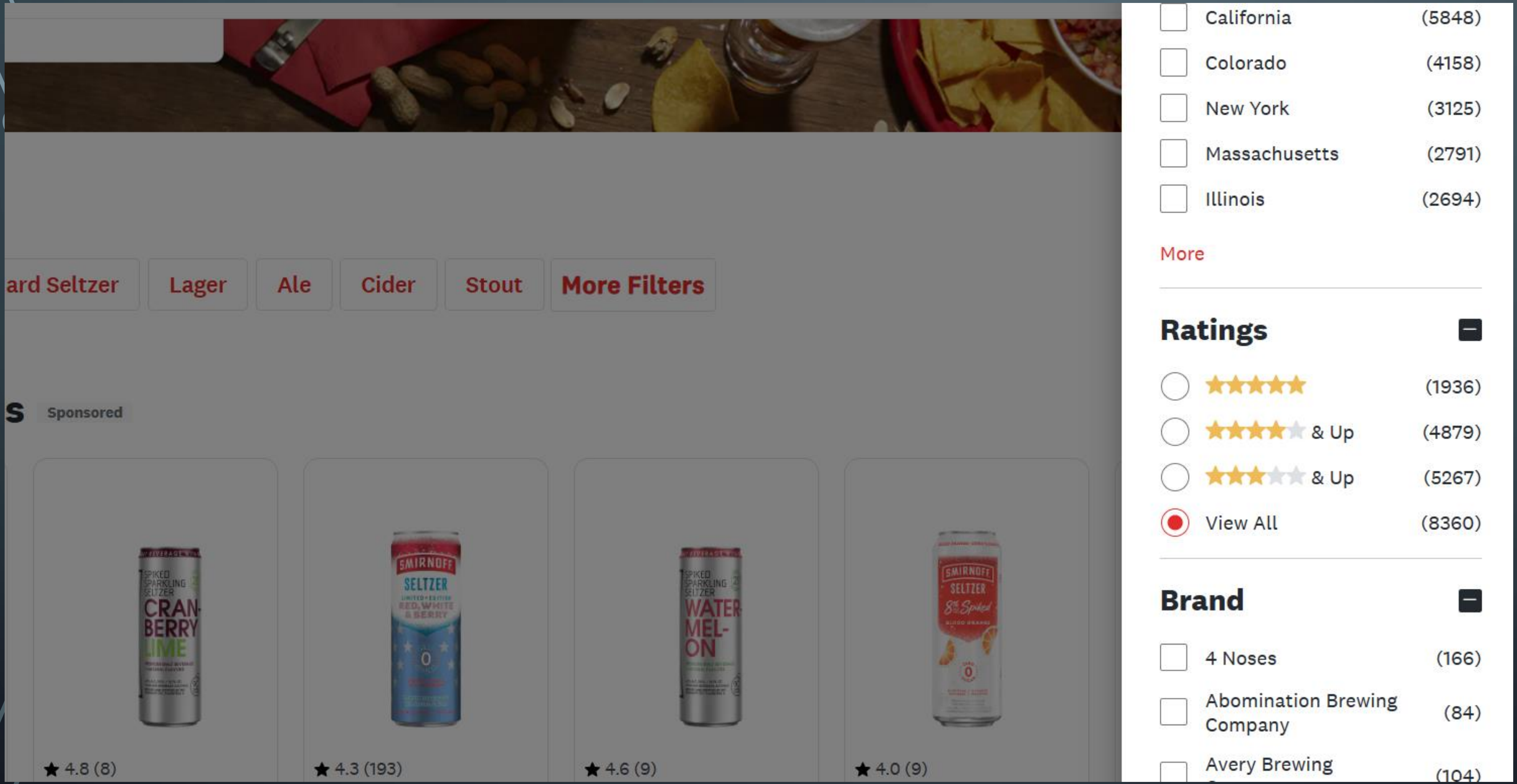
Shop now

Beer

IPA | Craft Beer | Hard Seltzer | Lager | Ale | Cider | Stout | **More Filters**

Beachin' beers Sponsored

FILTERING RESULTS



□ California	(5848)
□ Colorado	(4158)
□ New York	(3125)
□ Massachusetts	(2791)
□ Illinois	(2694)

More

Ratings

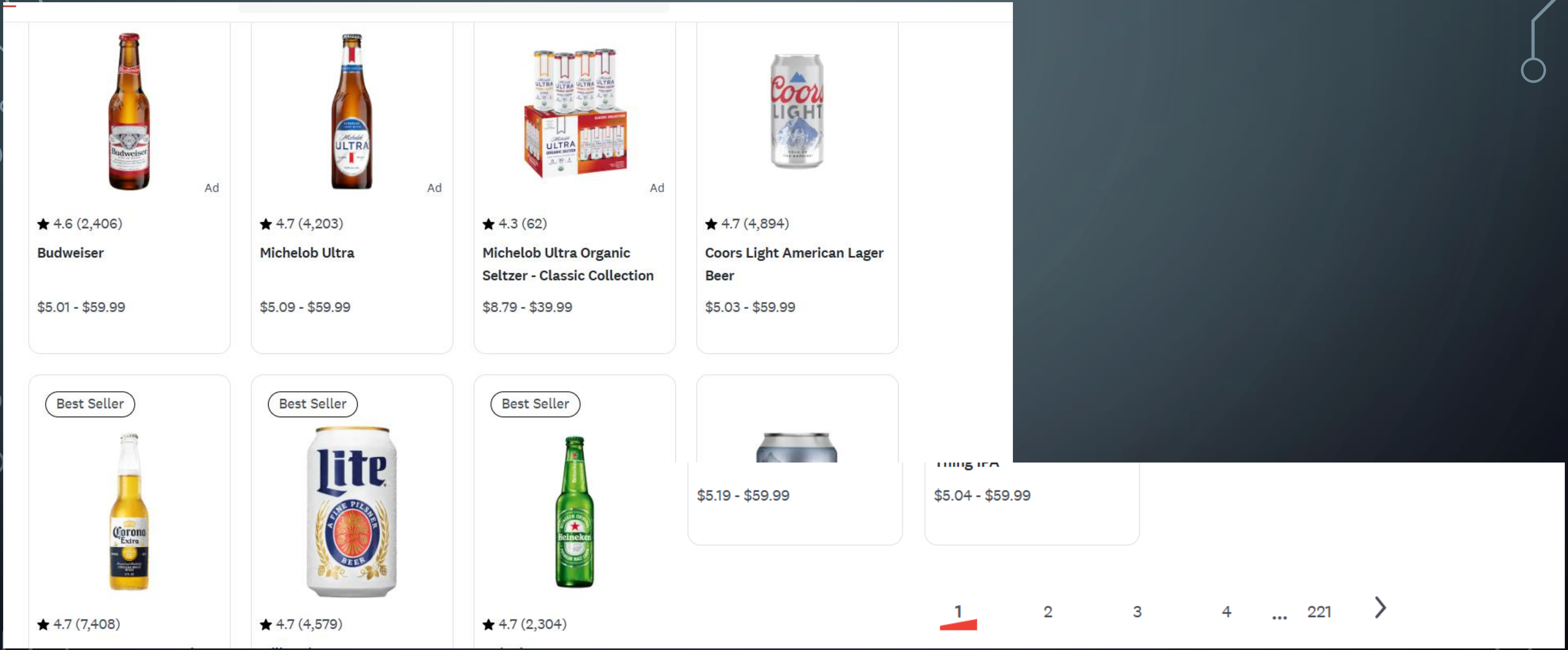
○ ★★★★★	(1936)
○ ★★★★☆ & Up	(4879)
○ ★★★☆☆ & Up	(5267)
● View All	(8360)

Brand

□ 4 Noses	(166)
□ Abomination Brewing Company	(84)
□ Avery Brewing	(104)

We filtered the results to show beers rated 3+

NOW WE GOT TO THIS SCREEN



Ad
★ 4.6 (2,406)
Budweiser
$5.01 - $59.99

Ad
★ 4.7 (4,203)
Michelob Ultra
$5.09 - $59.99

Ad
★ 4.3 (62)
Michelob Ultra Organic Seltzer - Classic Collection
$8.79 - $39.99

★ 4.7 (4,894)
Coors Light American Lager Beer
$5.03 - $59.99

Best Seller
★ 4.7 (7,408)

Best Seller
★ 4.7 (4,579)

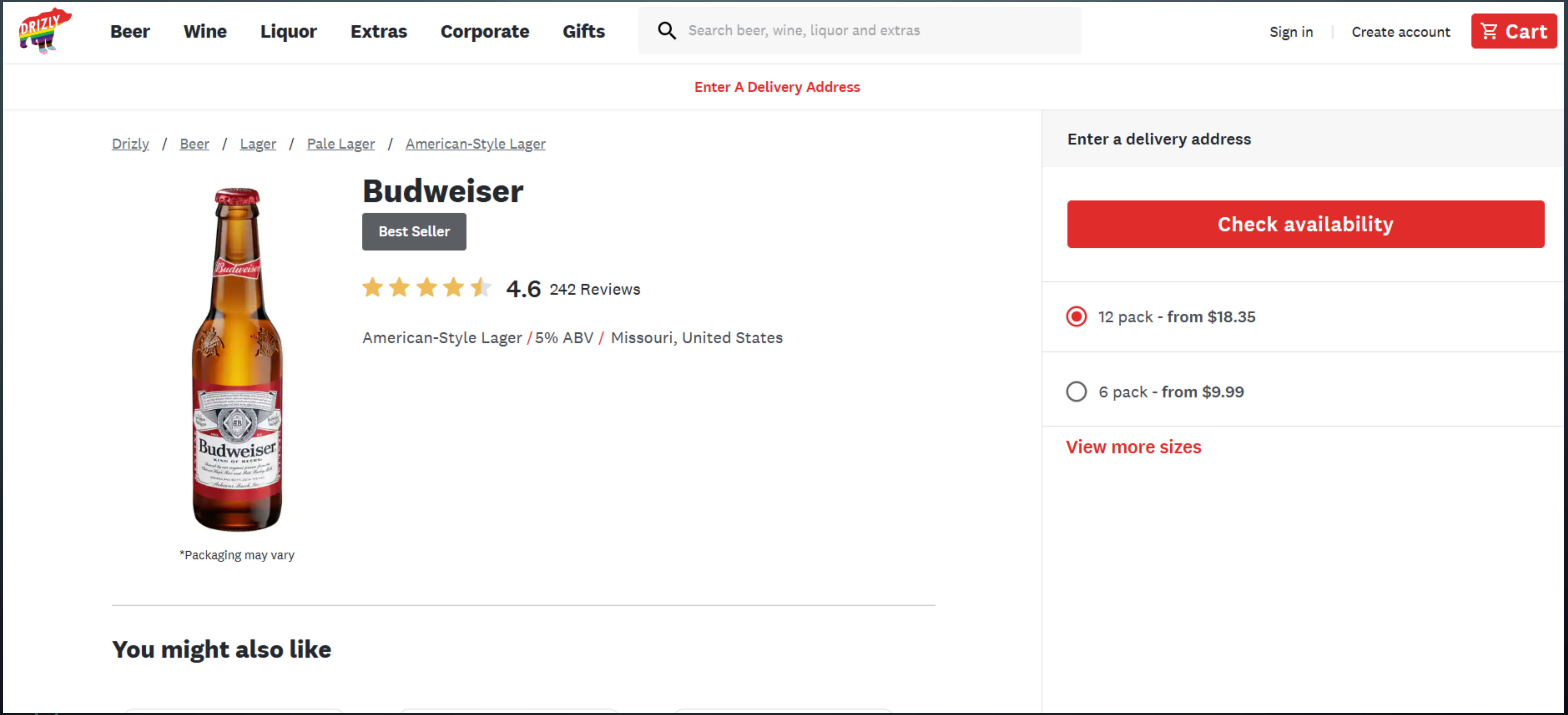
Best Seller
★ 4.7 (2,304)

$5.19 - $59.99

Thing IPA
$5.04 - $59.99

And then we went though all the pages and scraped the beer data

EXAMPLE FOR BEER PAGE



Drizly / Beer / Lager / Pale Lager / American-Style Lager

Budweiser

Best Seller

4.6 242 Reviews

American-Style Lager / 5% ABV / Missouri, United States

*Packaging may vary

Enter a delivery address

Check availability

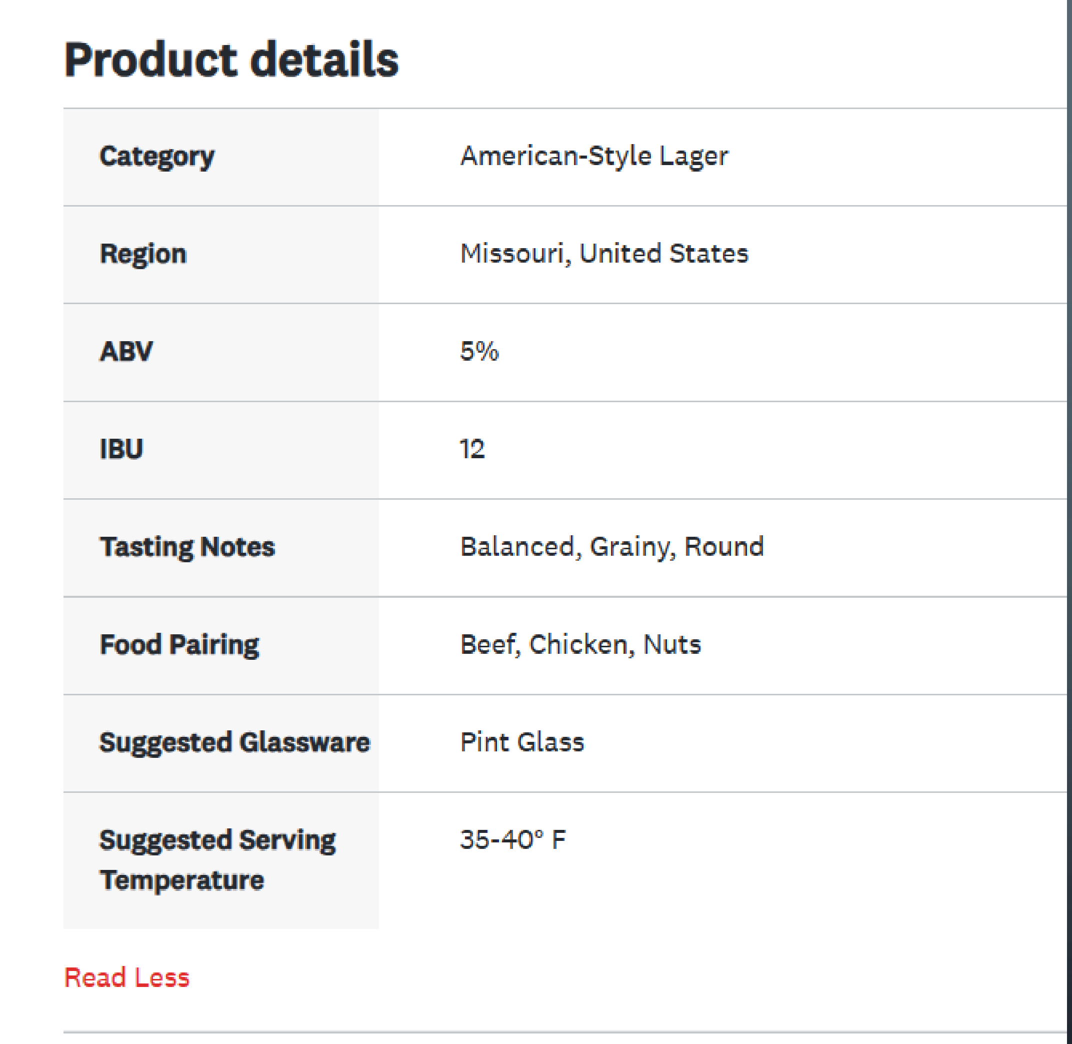
● 12 pack - from $18.35

○ 6 pack - from $9.99

View more sizes

You might also like

EACH BEER CONTAINS THE FOLLOWING DETAILS



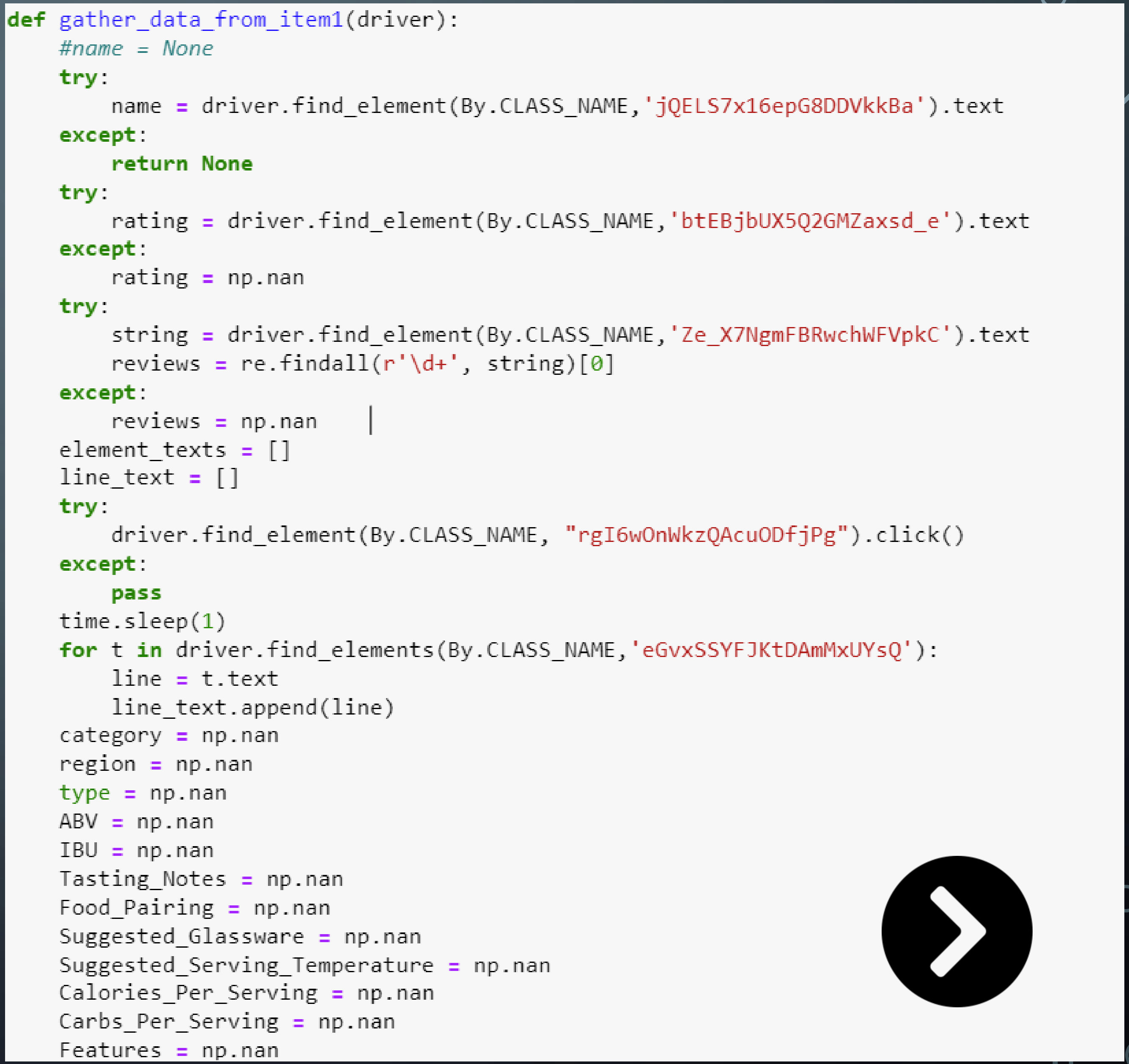
Product details

Category	American-Style Lager
Region	Missouri, United States
ABV	5%
IBU	12
Tasting Notes	Balanced, Grainy, Round
Food Pairing	Beef, Chicken, Nuts
Suggested Glassware	Pint Glass
Suggested Serving Temperature	35-40° F

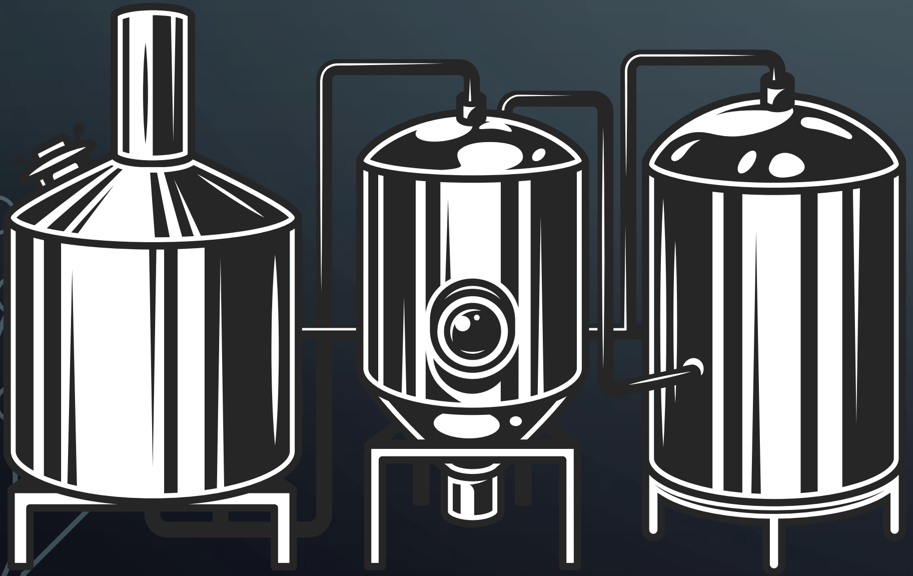
Read Less

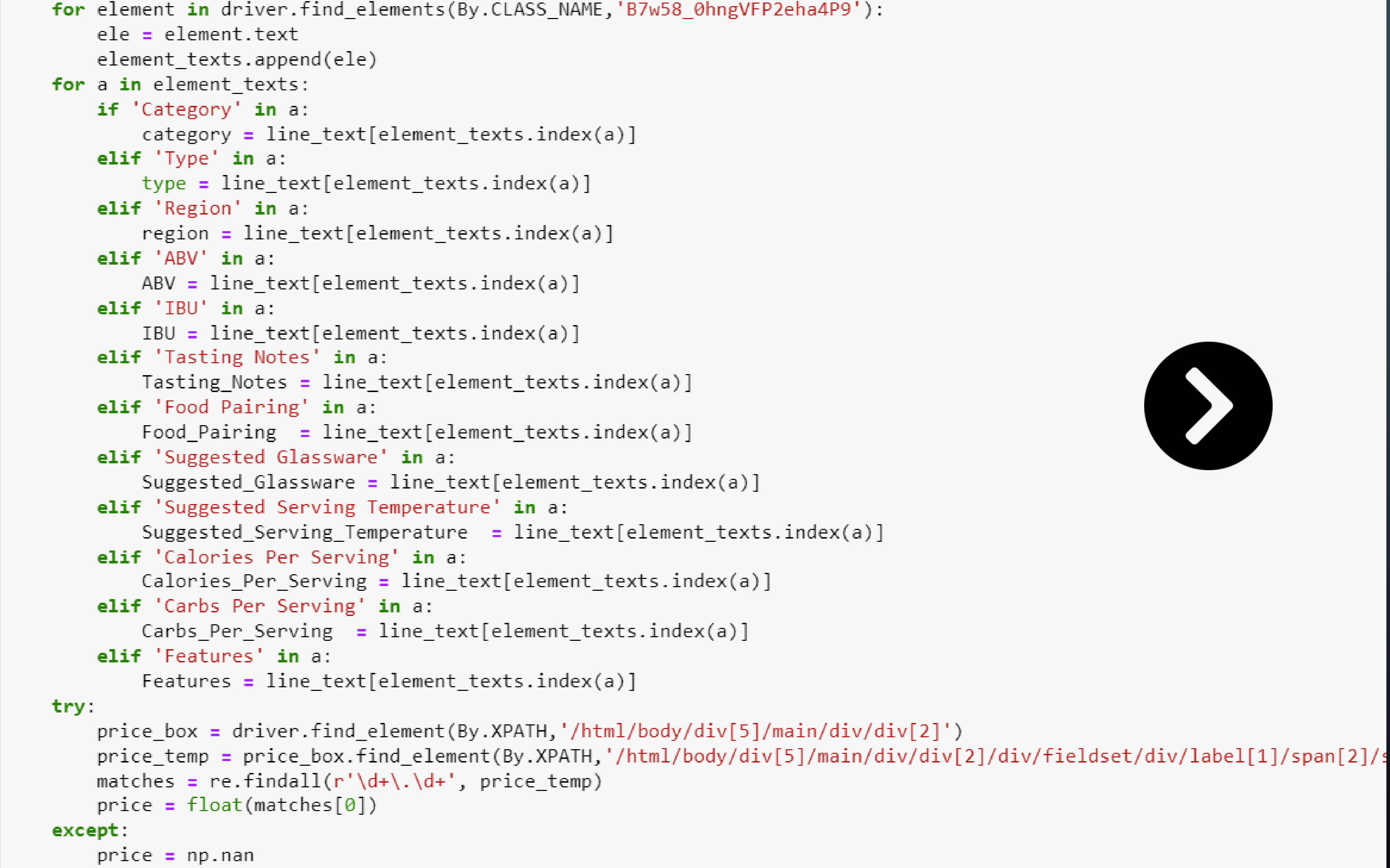
These are the details we scraped for each beer

SCRAPER CODE

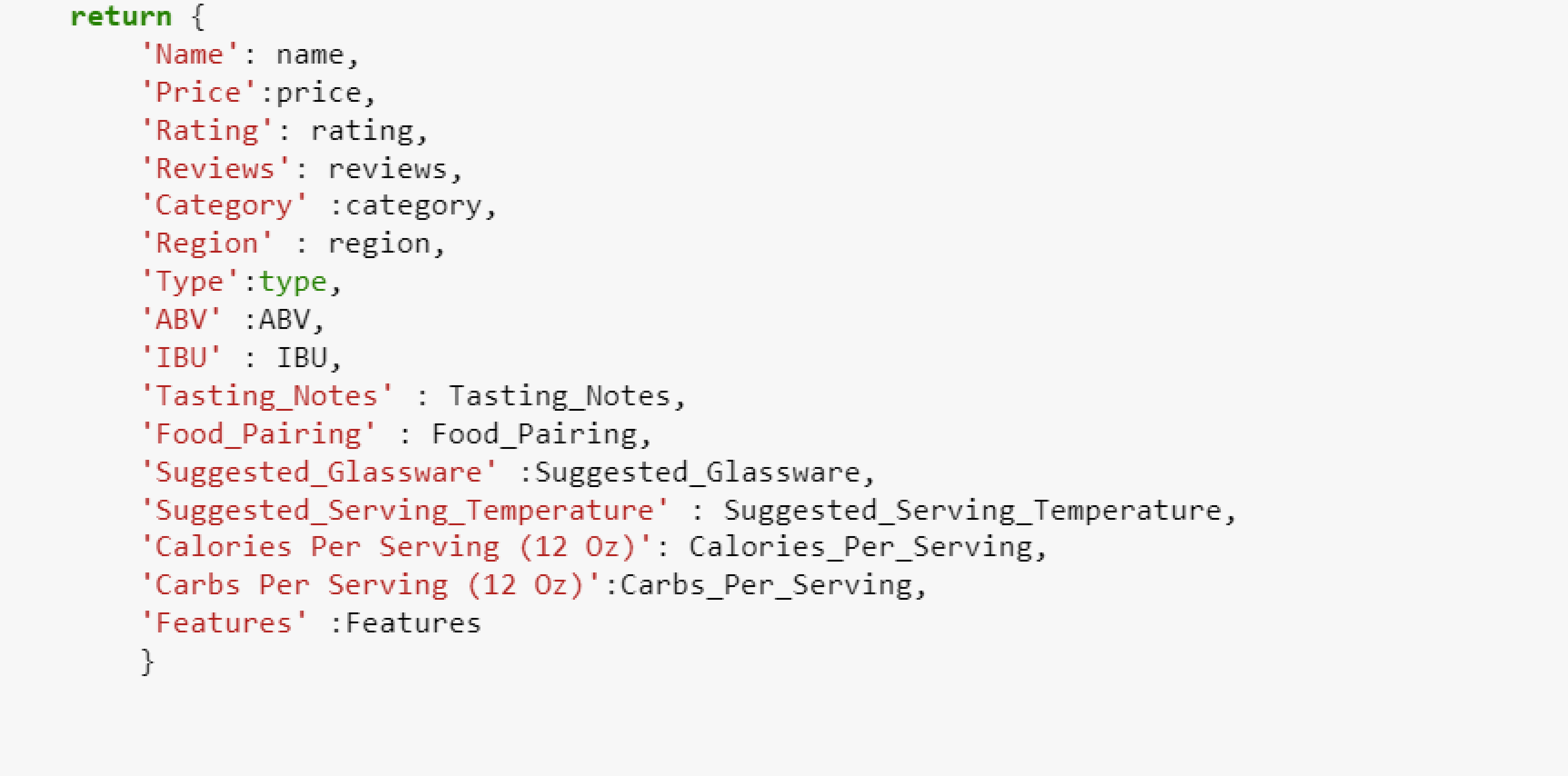


```python
def gather_data_from_item1(driver):
    #name = None
    try:
        name = driver.find_element(By.CLASS_NAME,'jQELS7x16epG8DDVkkBa').text
    except:
        return None
    try:
        rating = driver.find_element(By.CLASS_NAME,'btEBjbUX5Q2GMZaxsd_e').text
    except:
        rating = np.nan
    try:
        string = driver.find_element(By.CLASS_NAME,'Ze_X7NgmFBRwchWFVpkC').text
        reviews = re.findall(r'\d+', string)[0]
    except:
        reviews = np.nan
    element_texts = []
    line_text = []
    try:
        driver.find_element(By.CLASS_NAME, "rgI6wOnWkzQAcuODfjPg").click()
    except:
        pass
    time.sleep(1)
    for t in driver.find_elements(By.CLASS_NAME,'eGvxSSYFJKtDAmMxUYsQ'):
        line = t.text
        line_text.append(line)
    category = np.nan
    region = np.nan
    type = np.nan
    ABV = np.nan
    IBU = np.nan
    Tasting_Notes = np.nan
    Food_Pairing = np.nan
    Suggested_Glassware = np.nan
    Suggested_Serving_Temperature = np.nan
    Calories_Per_Serving = np.nan
    Carbs_Per_Serving = np.nan
    Features = np.nan
```



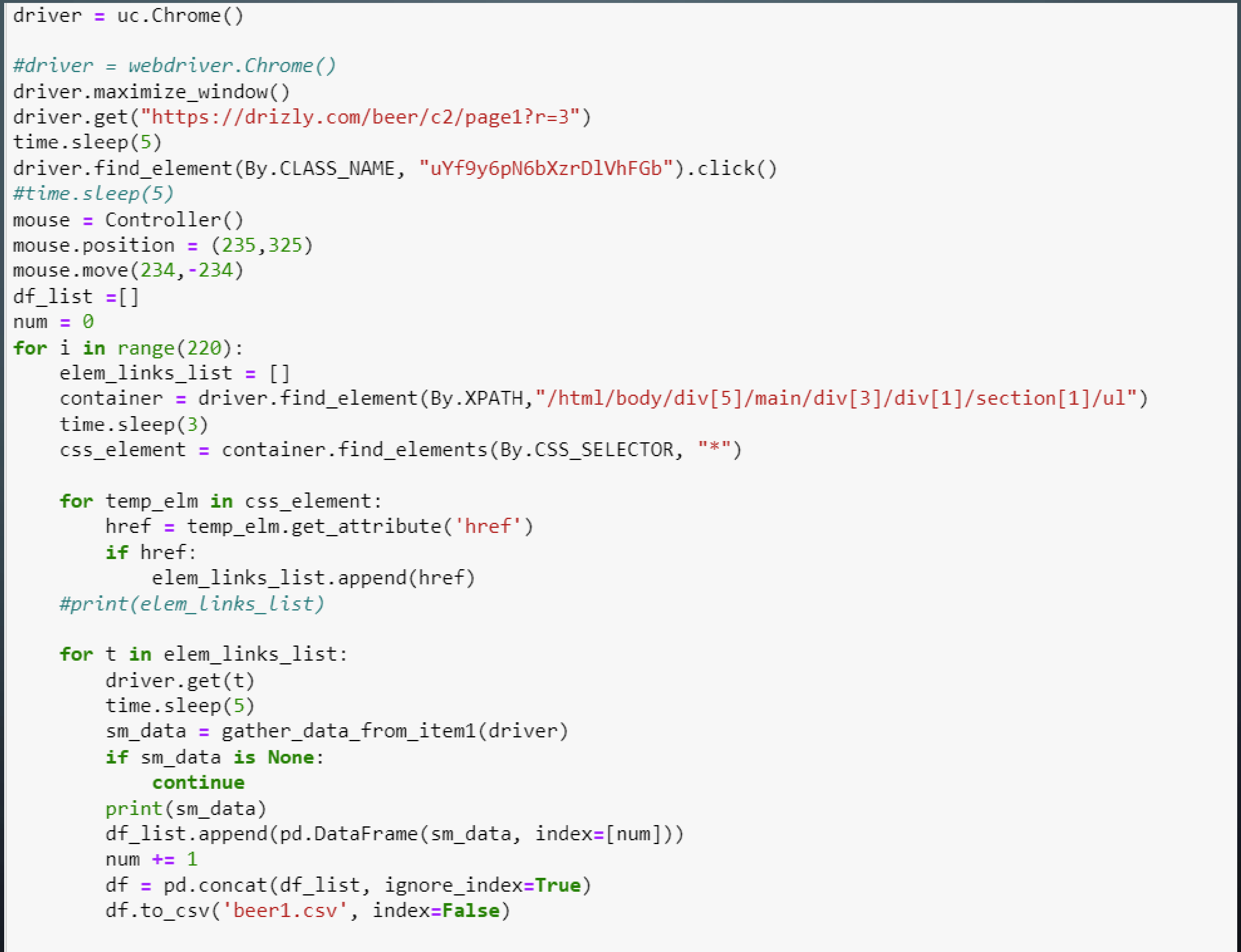


```
    for element in driver.find_elements(By.CLASS_NAME,'B7w58_0hngVFP2eha4P9'):
        ele = element.text
        element_texts.append(ele)
    for a in element_texts:
        if 'Category' in a:
            category = line_text[element_texts.index(a)]
        elif 'Type' in a:
            type = line_text[element_texts.index(a)]
        elif 'Region' in a:
            region = line_text[element_texts.index(a)]
        elif 'ABV' in a:
            ABV = line_text[element_texts.index(a)]
        elif 'IBU' in a:
            IBU = line_text[element_texts.index(a)]
        elif 'Tasting Notes' in a:
            Tasting_Notes = line_text[element_texts.index(a)]
        elif 'Food Pairing' in a:
            Food_Pairing  = line_text[element_texts.index(a)]
        elif 'Suggested Glassware' in a:
            Suggested_Glassware = line_text[element_texts.index(a)]
        elif 'Suggested Serving Temperature' in a:
            Suggested_Serving_Temperature  = line_text[element_texts.index(a)]
        elif 'Calories Per Serving' in a:
            Calories_Per_Serving = line_text[element_texts.index(a)]
        elif 'Carbs Per Serving' in a:
            Carbs_Per_Serving  = line_text[element_texts.index(a)]
        elif 'Features' in a:
            Features = line_text[element_texts.index(a)]
    try:
        price_box = driver.find_element(By.XPATH,'/html/body/div[5]/main/div/div[2]')
        price_temp = price_box.find_element(By.XPATH,'/html/body/div[5]/main/div/div[2]/div/fieldset/div/label[1]/span[2]/s
        matches = re.findall(r'\d+\.\d+', price_temp)
        price = float(matches[0])
    except:
        price = np.nan
```



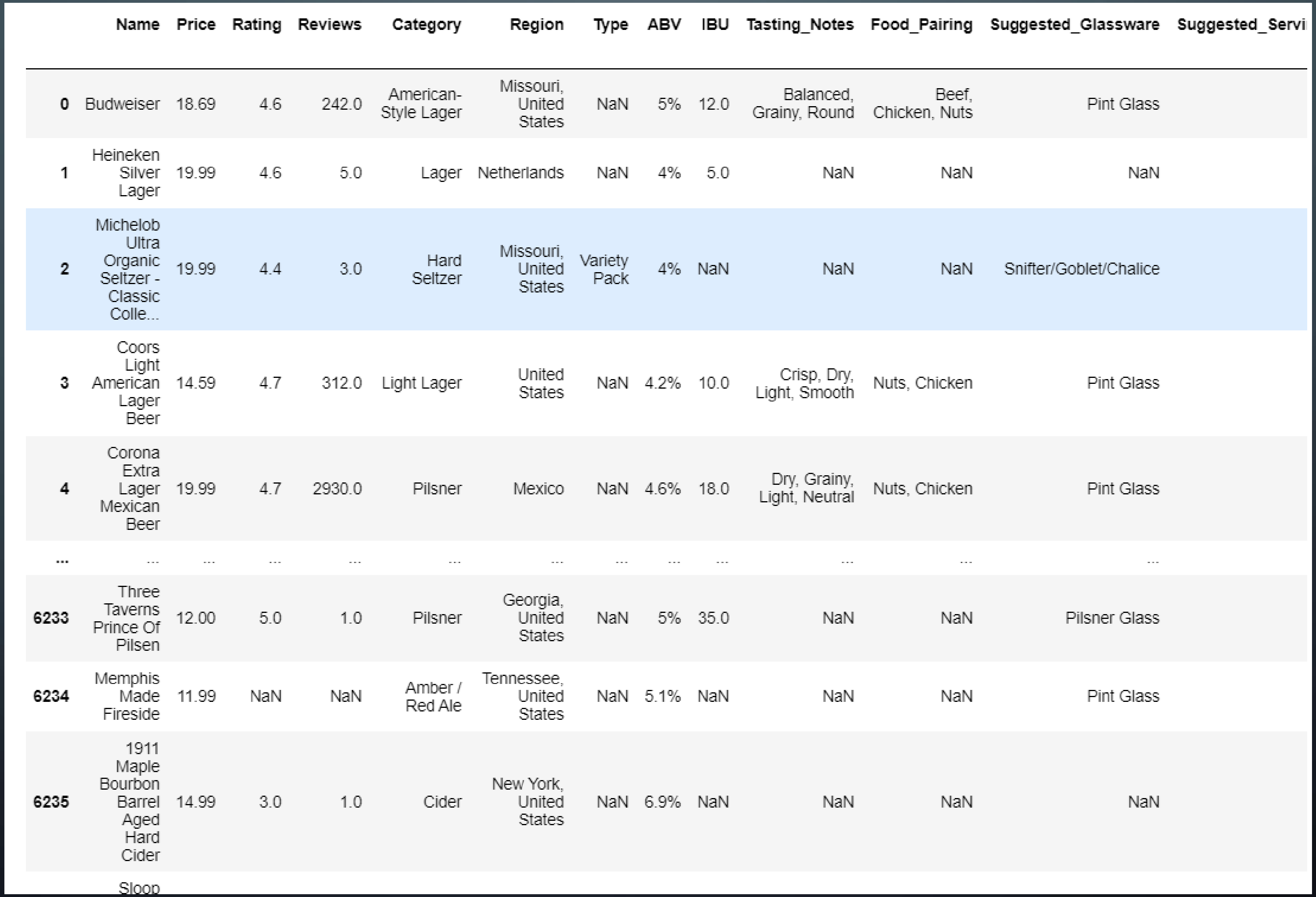
```
    return {
        'Name': name,
        'Price':price,
        'Rating': rating,
        'Reviews': reviews,
        'Category' :category,
        'Region' : region,
        'Type':type,
        'ABV' :ABV,
        'IBU' : IBU,
        'Tasting_Notes' : Tasting_Notes,
        'Food_Pairing' : Food_Pairing,
        'Suggested_Glassware' :Suggested_Glassware,
        'Suggested_Serving_Temperature' : Suggested_Serving_Temperature,
        'Calories Per Serving (12 Oz)': Calories_Per_Serving,
        'Carbs Per Serving (12 Oz)':Carbs_Per_Serving,
        'Features' :Features
        }
```

POPUP PROTECTION & SITE LINKING FROM SELENIUM



```
driver = uc.Chrome()

#driver = webdriver.Chrome()
driver.maximize_window()
driver.get("https://drizly.com/beer/c2/page1?r=3")
time.sleep(5)
driver.find_element(By.CLASS_NAME, "uYf9y6pN6bXzrDlVhFGb").click()
#time.sleep(5)
mouse = Controller()
mouse.position = (235,325)
mouse.move(234,-234)
df_list =[]
num = 0
for i in range(220):
    elem_links_list = []
    container = driver.find_element(By.XPATH,"/html/body/div[5]/main/div[3]/div[1]/section[1]/ul")
    time.sleep(3)
    css_element = container.find_elements(By.CSS_SELECTOR, "*")

    for temp_elm in css_element:
        href = temp_elm.get_attribute('href')
        if href:
            elem_links_list.append(href)
    #print(elem_links_list)

    for t in elem_links_list:
        driver.get(t)
        time.sleep(5)
        sm_data = gather_data_from_item1(driver)
        if sm_data is None:
            continue
        print(sm_data)
        df_list.append(pd.DataFrame(sm_data, index=[num]))
        num += 1
        df = pd.concat(df_list, ignore_index=True)
        df.to_csv('beer1.csv', index=False)
```

RESULT CSV [6238 rows x 16 columns]



	Name	Price	Rating	Reviews	Category	Region	Type	ABV	IBU	Tasting_Notes	Food_Pairing	Suggested_Glassware	Suggested_Servi
0	Budweiser	18.69	4.6	242.0	American-Style Lager	Missouri, United States	NaN	5%	12.0	Balanced, Grainy, Round	Beef, Chicken, Nuts	Pint Glass	
1	Heineken Silver Lager	19.99	4.6	5.0	Lager	Netherlands	NaN	4%	5.0	NaN	NaN	NaN	
2	Michelob Ultra Organic Seltzer - Classic Colle...	19.99	4.4	3.0	Hard Seltzer	Missouri, United States	Variety Pack	4%	NaN	NaN	NaN	Snifter/Goblet/Chalice	
3	Coors Light American Lager Beer	14.59	4.7	312.0	Light Lager	United States	NaN	4.2%	10.0	Crisp, Dry, Light, Smooth	Nuts, Chicken	Pint Glass	
4	Corona Extra Lager Mexican Beer	19.99	4.7	2930.0	Pilsner	Mexico	NaN	4.6%	18.0	Dry, Grainy, Light, Neutral	Nuts, Chicken	Pint Glass	
...	...	...	...	...	...	...	...	...	...	...	...	...	
6233	Three Taverns Prince Of Pilsen	12.00	5.0	1.0	Pilsner	Georgia, United States	NaN	5%	35.0	NaN	NaN	Pilsner Glass	
6234	Memphis Made Fireside	11.99	NaN	NaN	Amber / Red Ale	Tennessee, United States	NaN	5.1%	NaN	NaN	NaN	Pint Glass	
6235	1911 Maple Bourbon Barrel Aged Hard Cider	14.99	3.0	1.0	Cider	New York, United States	NaN	6.9%	NaN	NaN	NaN	NaN	
	Sloop												

COLUMNS

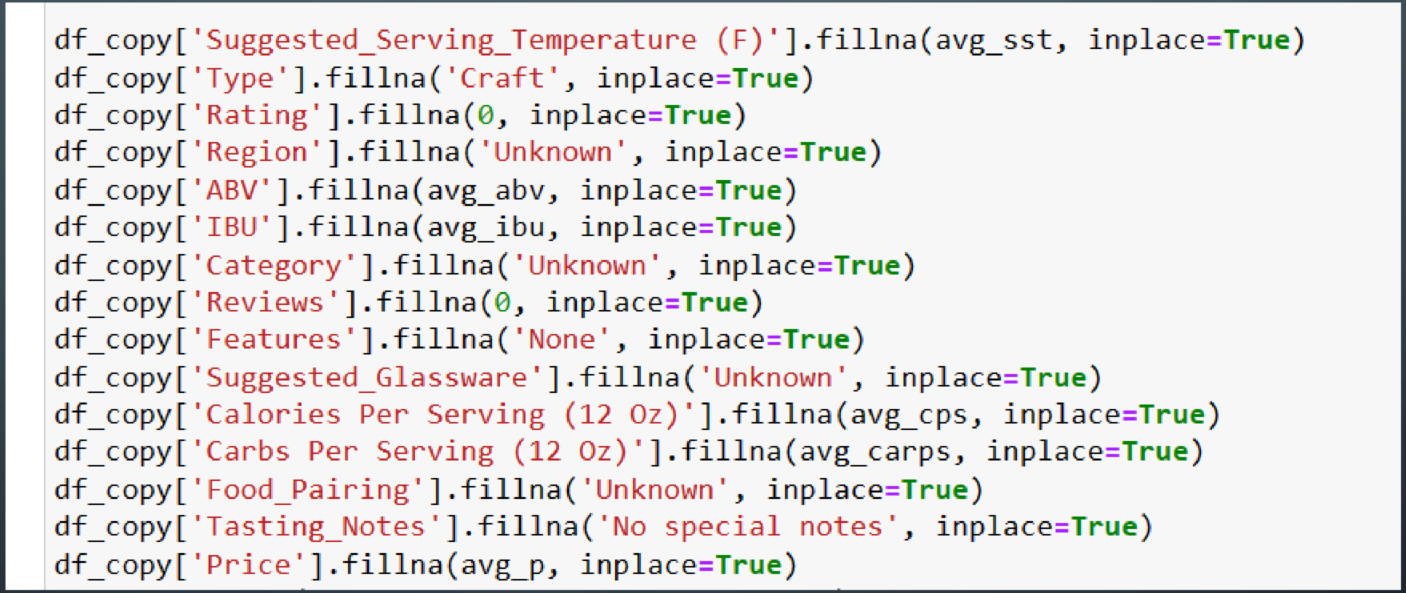
- Name
- Price
- Rating
- Reviews
- Category
- Region
- Type

- ABV
- IBU
- Tasting Notes
- Food Pairing
- Suggested Glassware
- Suggested Serving Temperature
- Calories Per Serving (12 Oz)
- Carbs Per Serving (12 Oz)
- Features

DATA CLEANING

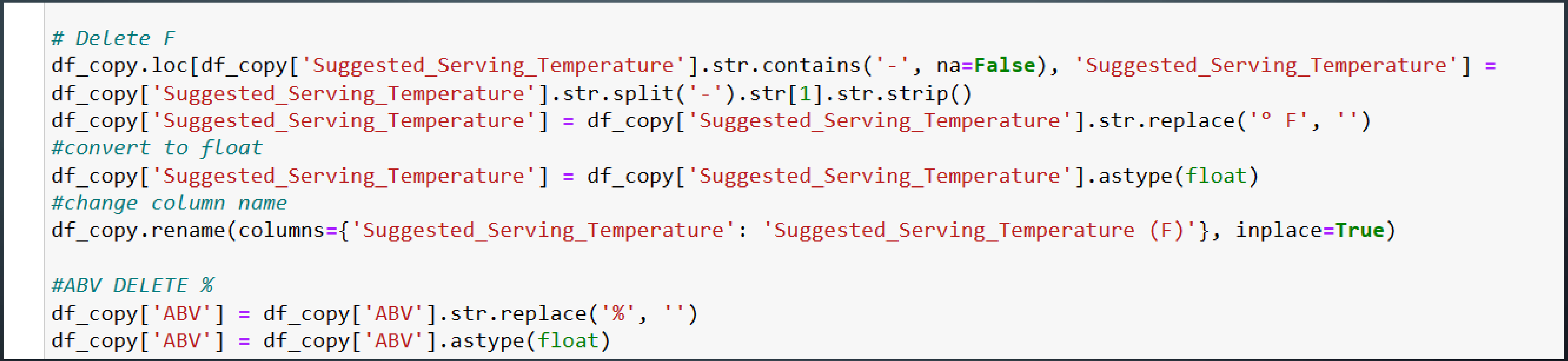
- Handling missing values
- Removing ‘%’ marks from data
- Converting all sort of data from object form to integers
- Removing beers rated ‘0’
- Removing city names and keeping only countries for each beer
- Dealing with outliers
- Dropping duplicates by ‘Name’

HANDLING MISSING VALUES



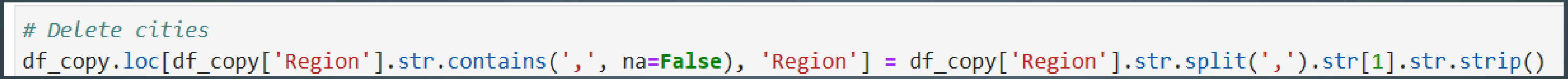
```
df_copy['Suggested_Serving_Temperature (F)'].fillna(avg_sst, inplace=True)
df_copy['Type'].fillna('Craft', inplace=True)
df_copy['Rating'].fillna(0, inplace=True)
df_copy['Region'].fillna('Unknown', inplace=True)
df_copy['ABV'].fillna(avg_abv, inplace=True)
df_copy['IBU'].fillna(avg_ibu, inplace=True)
df_copy['Category'].fillna('Unknown', inplace=True)
df_copy['Reviews'].fillna(0, inplace=True)
df_copy['Features'].fillna('None', inplace=True)
df_copy['Suggested_Glassware'].fillna('Unknown', inplace=True)
df_copy['Calories Per Serving (12 Oz)'].fillna(avg_cps, inplace=True)
df_copy['Carbs Per Serving (12 Oz)'].fillna(avg_carps, inplace=True)
df_copy['Food_Pairing'].fillna('Unknown', inplace=True)
df_copy['Tasting_Notes'].fillna('No special notes', inplace=True)
df_copy['Price'].fillna(avg_p, inplace=True)
```

REMOVING '%' MARKS FROM DATA



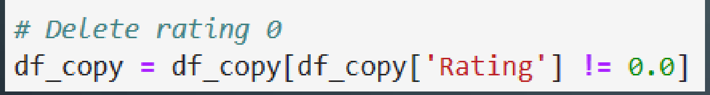
```
# Delete F
df_copy.loc[df_copy['Suggested_Serving_Temperature'].str.contains('-', na=False), 'Suggested_Serving_Temperature'] =
df_copy['Suggested_Serving_Temperature'].str.split('-').str[1].str.strip()
df_copy['Suggested_Serving_Temperature'] = df_copy['Suggested_Serving_Temperature'].str.replace('° F', '')
#convert to float
df_copy['Suggested_Serving_Temperature'] = df_copy['Suggested_Serving_Temperature'].astype(float)
#change column name
df_copy.rename(columns={'Suggested_Serving_Temperature': 'Suggested_Serving_Temperature (F)'}, inplace=True)

#ABV DELETE %
df_copy['ABV'] = df_copy['ABV'].str.replace('%', '')
df_copy['ABV'] = df_copy['ABV'].astype(float)
```

REMOVING CITY NAMES AND KEEPING ONLY COUNTRIES FOR EACH BEER



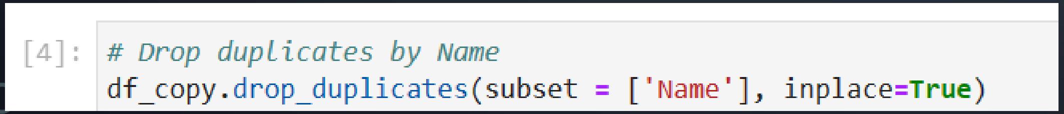
```
# Delete cities
df_copy.loc[df_copy['Region'].str.contains(',', na=False), 'Region'] = df_copy['Region'].str.split(',').str[1].str.strip()
```

REMOVING BEERS RATED ‘0’



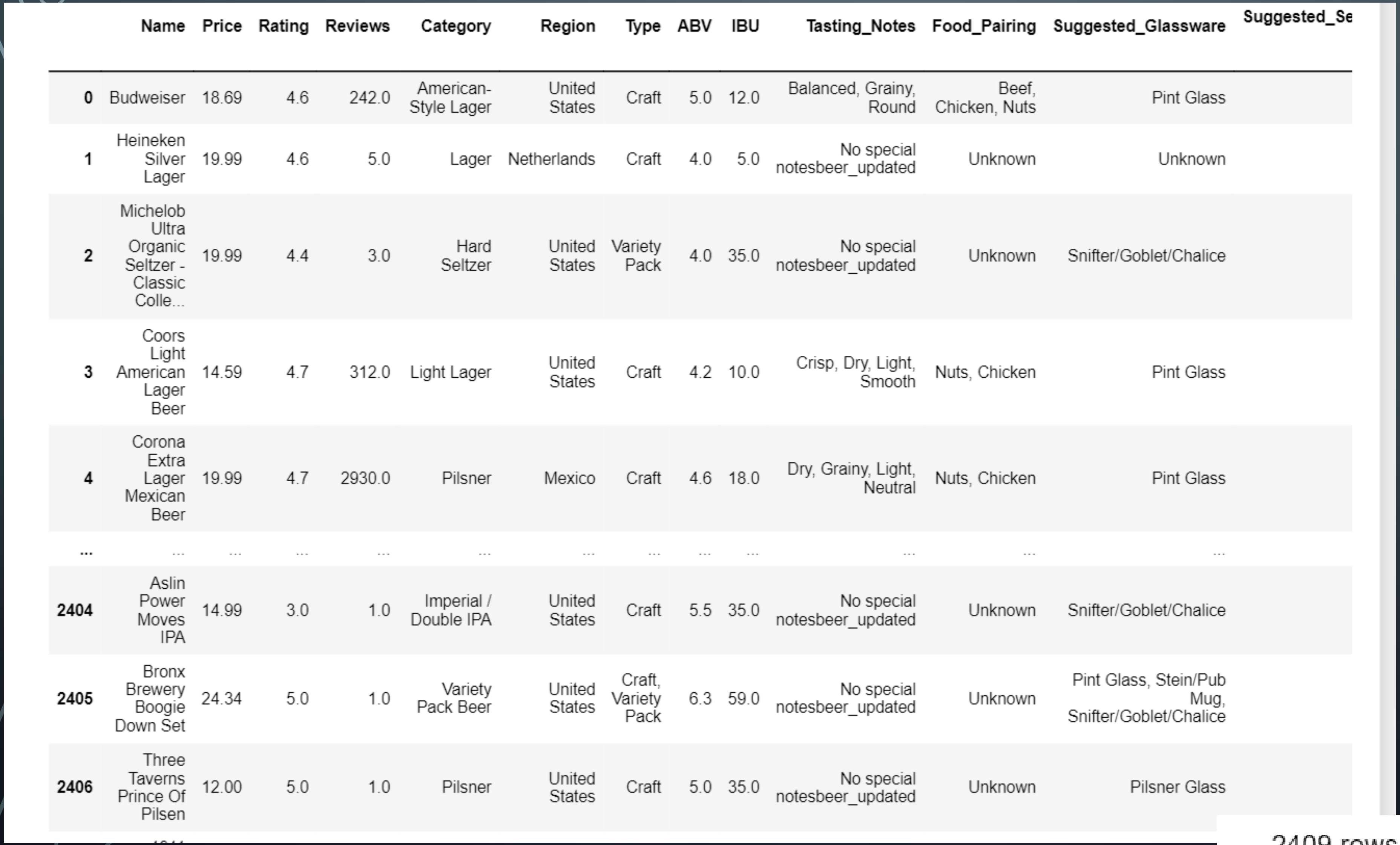
```
# Delete rating 0
df_copy = df_copy[df_copy['Rating'] != 0.0]
```

DROP DUPLICATES BY NAME



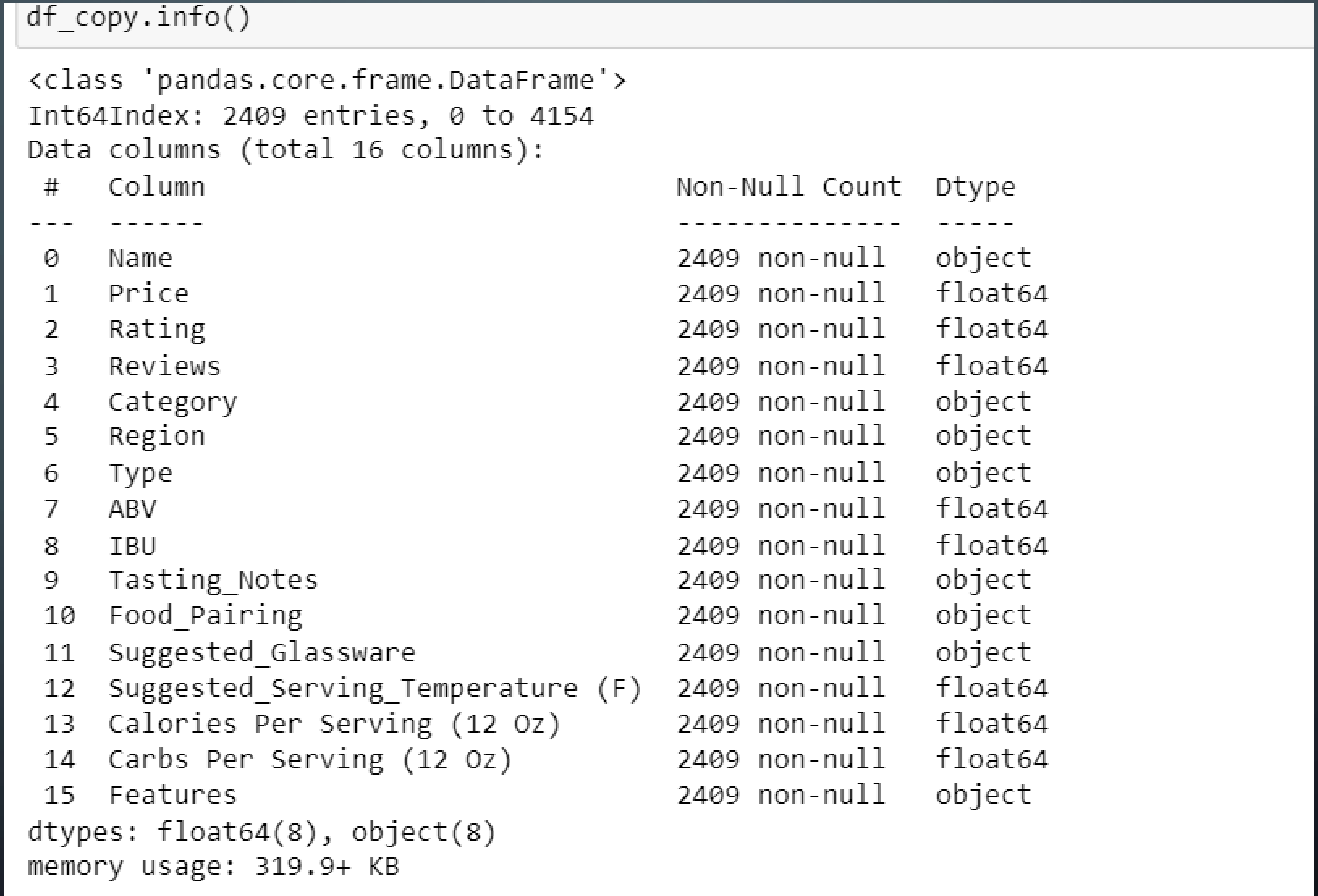
```
[4]: # Drop duplicates by Name
     df_copy.drop_duplicates(subset = ['Name'], inplace=True)
```

AFTER DATA CLEANING



	Name	Price	Rating	Reviews	Category	Region	Type	ABV	IBU	Tasting_Notes	Food_Pairing	Suggested_Glassware	Suggested_Se
0	Budweiser	18.69	4.6	242.0	American-Style Lager	United States	Craft	5.0	12.0	Balanced, Grainy, Round	Beef, Chicken, Nuts	Pint Glass	
1	Heineken Silver Lager	19.99	4.6	5.0	Lager	Netherlands	Craft	4.0	5.0	No special notesbeer_updated	Unknown	Unknown	
2	Michelob Ultra Organic Seltzer - Classic Colle...	19.99	4.4	3.0	Hard Seltzer	United States	Variety Pack	4.0	35.0	No special notesbeer_updated	Unknown	Snifter/Goblet/Chalice	
3	Coors Light American Lager Beer	14.59	4.7	312.0	Light Lager	United States	Craft	4.2	10.0	Crisp, Dry, Light, Smooth	Nuts, Chicken	Pint Glass	
4	Corona Extra Lager Mexican Beer	19.99	4.7	2930.0	Pilsner	Mexico	Craft	4.6	18.0	Dry, Grainy, Light, Neutral	Nuts, Chicken	Pint Glass	
...	...	...	...	...	...	...	...	...	...	...	...	...	
2404	Aslin Power Moves IPA	14.99	3.0	1.0	Imperial / Double IPA	United States	Craft	5.5	35.0	No special notesbeer_updated	Unknown	Snifter/Goblet/Chalice	
2405	Bronx Brewery Boogie Down Set	24.34	5.0	1.0	Variety Pack Beer	United States	Craft, Variety Pack	6.3	59.0	No special notesbeer_updated	Unknown	Pint Glass, Stein/Pub Mug, Snifter/Goblet/Chalice	
2406	Three Taverns Prince Of Pilsen	12.00	5.0	1.0	Pilsner	United States	Craft	5.0	35.0	No special notesbeer_updated	Unknown	Pilsner Glass	

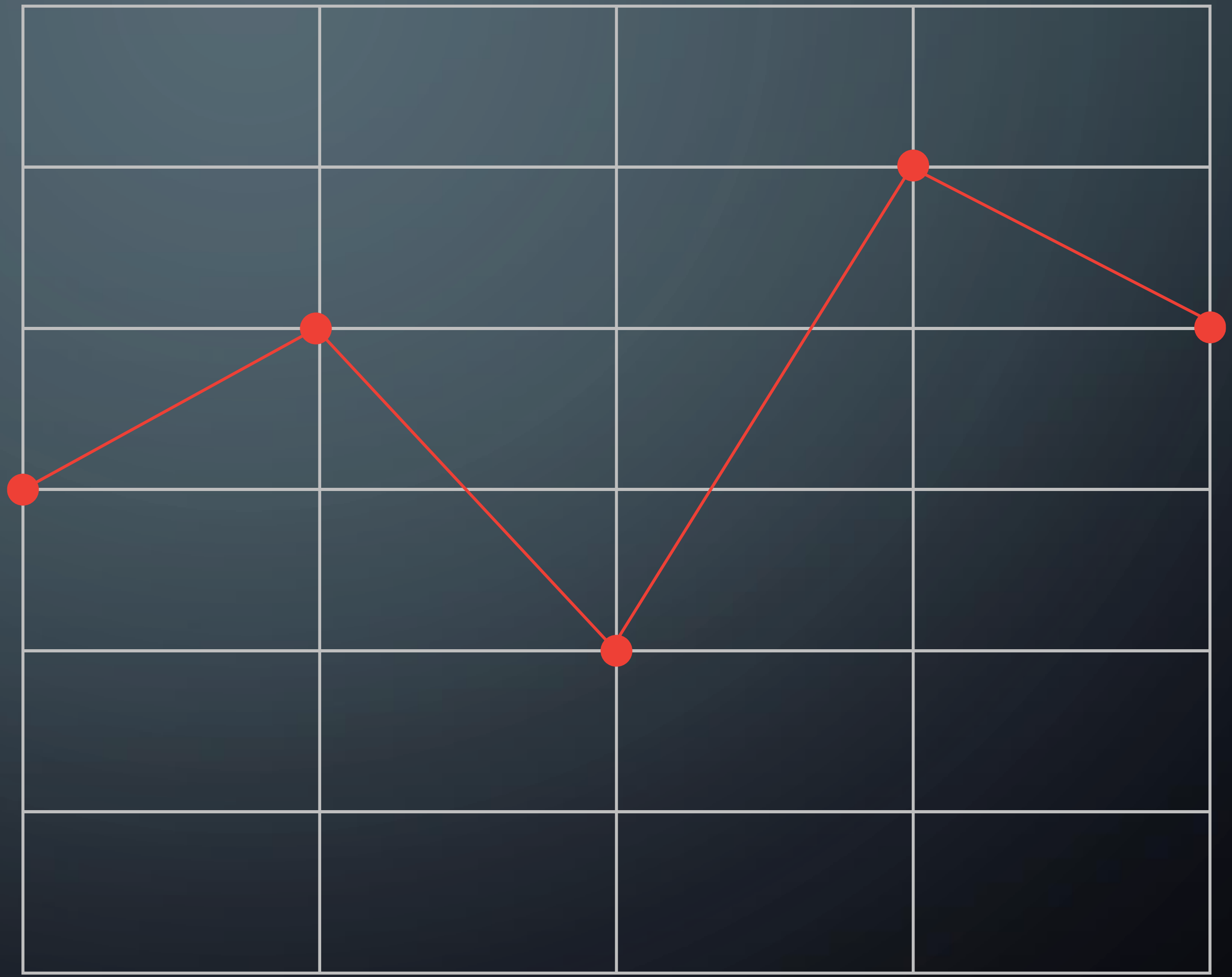
2409 rows × 16 columns



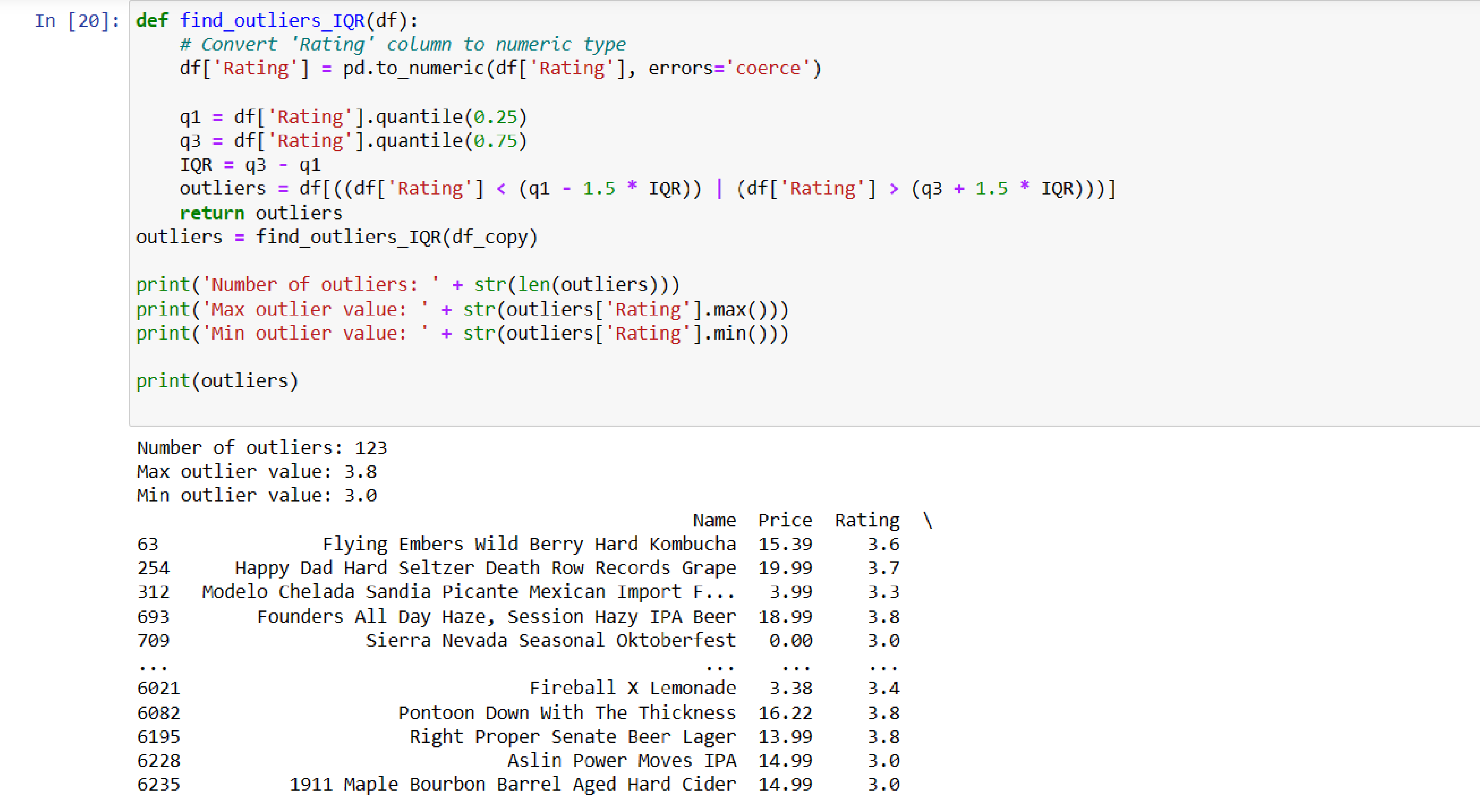
```
df_copy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2409 entries, 0 to 4154
Data columns (total 16 columns):
 #   Column                              Non-Null Count  Dtype  
---  ------                              --------------  -----  
 0   Name                                2409 non-null   object 
 1   Price                               2409 non-null   float64
 2   Rating                              2409 non-null   float64
 3   Reviews                             2409 non-null   float64
 4   Category                            2409 non-null   object 
 5   Region                              2409 non-null   object 
 6   Type                                2409 non-null   object 
 7   ABV                                 2409 non-null   float64
 8   IBU                                 2409 non-null   float64
 9   Tasting_Notes                       2409 non-null   object 
 10  Food_Pairing                        2409 non-null   object 
 11  Suggested_Glassware                 2409 non-null   object 
 12  Suggested_Serving_Temperature (F)   2409 non-null   float64
 13  Calories Per Serving (12 Oz)        2409 non-null   float64
 14  Carbs Per Serving (12 Oz)           2409 non-null   float64
 15  Features                            2409 non-null   object 
dtypes: float64(8), object(8)
memory usage: 319.9+ KB
```

EDA



DEALING WITH OUTLIERS

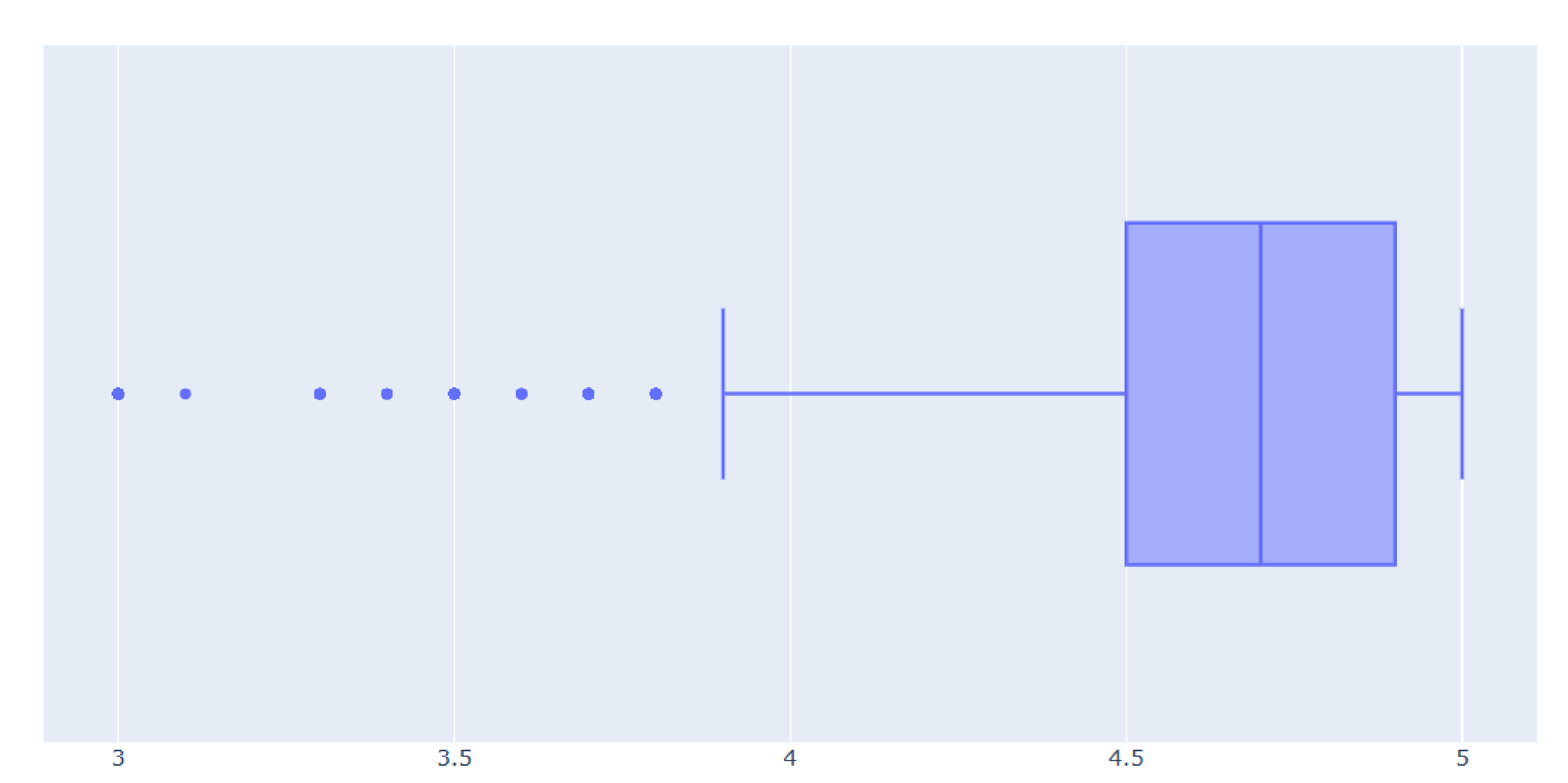


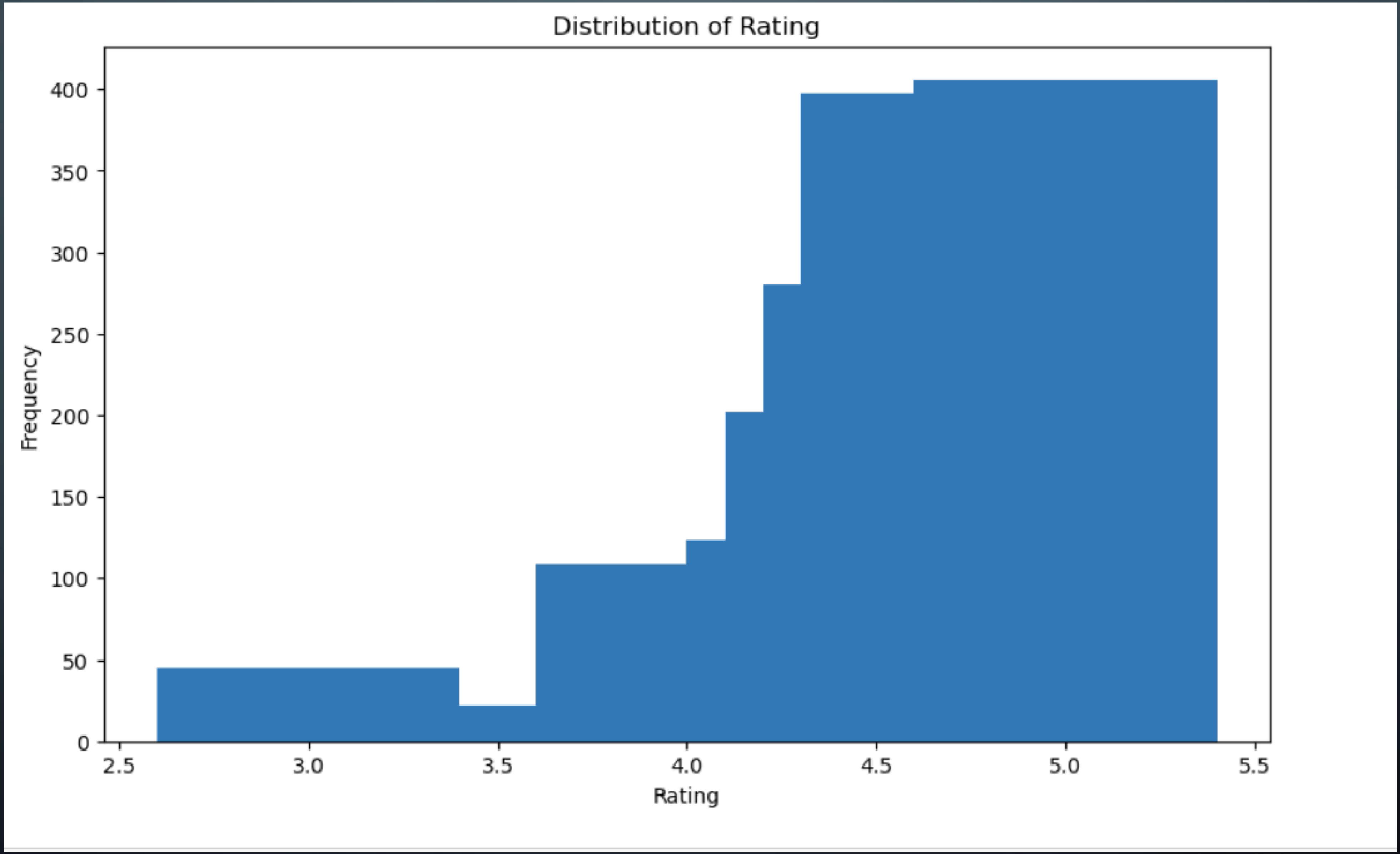
In [20]:

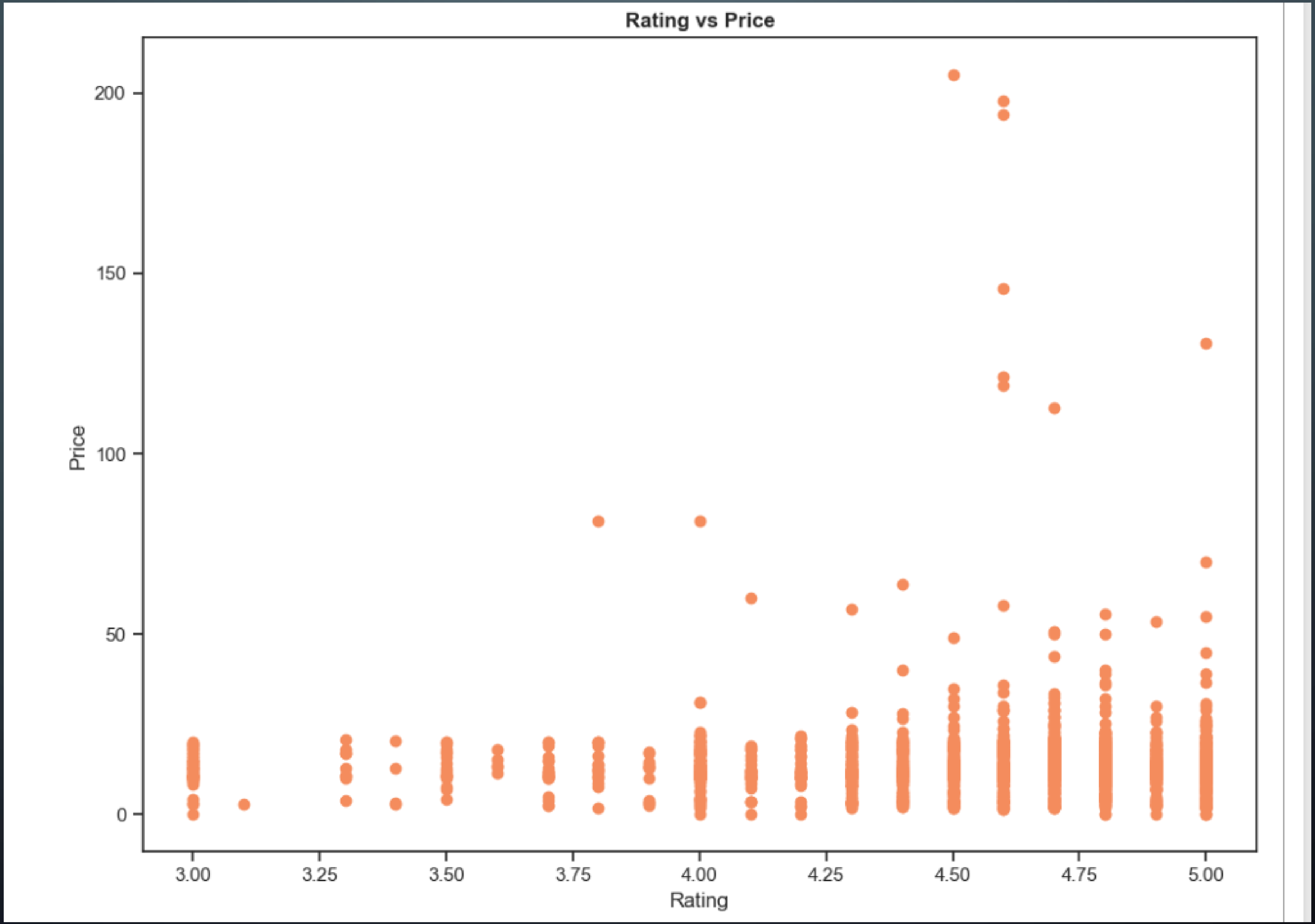
```
def find_outliers_IQR(df):
    # Convert 'Rating' column to numeric type
    df['Rating'] = pd.to_numeric(df['Rating'], errors='coerce')

    q1 = df['Rating'].quantile(0.25)
    q3 = df['Rating'].quantile(0.75)
    IQR = q3 - q1
    outliers = df[((df['Rating'] < (q1 - 1.5 * IQR)) | (df['Rating'] > (q3 + 1.5 * IQR)))]
    return outliers
outliers = find_outliers_IQR(df_copy)

print('Number of outliers: ' + str(len(outliers)))
print('Max outlier value: ' + str(outliers['Rating'].max()))
print('Min outlier value: ' + str(outliers['Rating'].min()))

print(outliers)
```

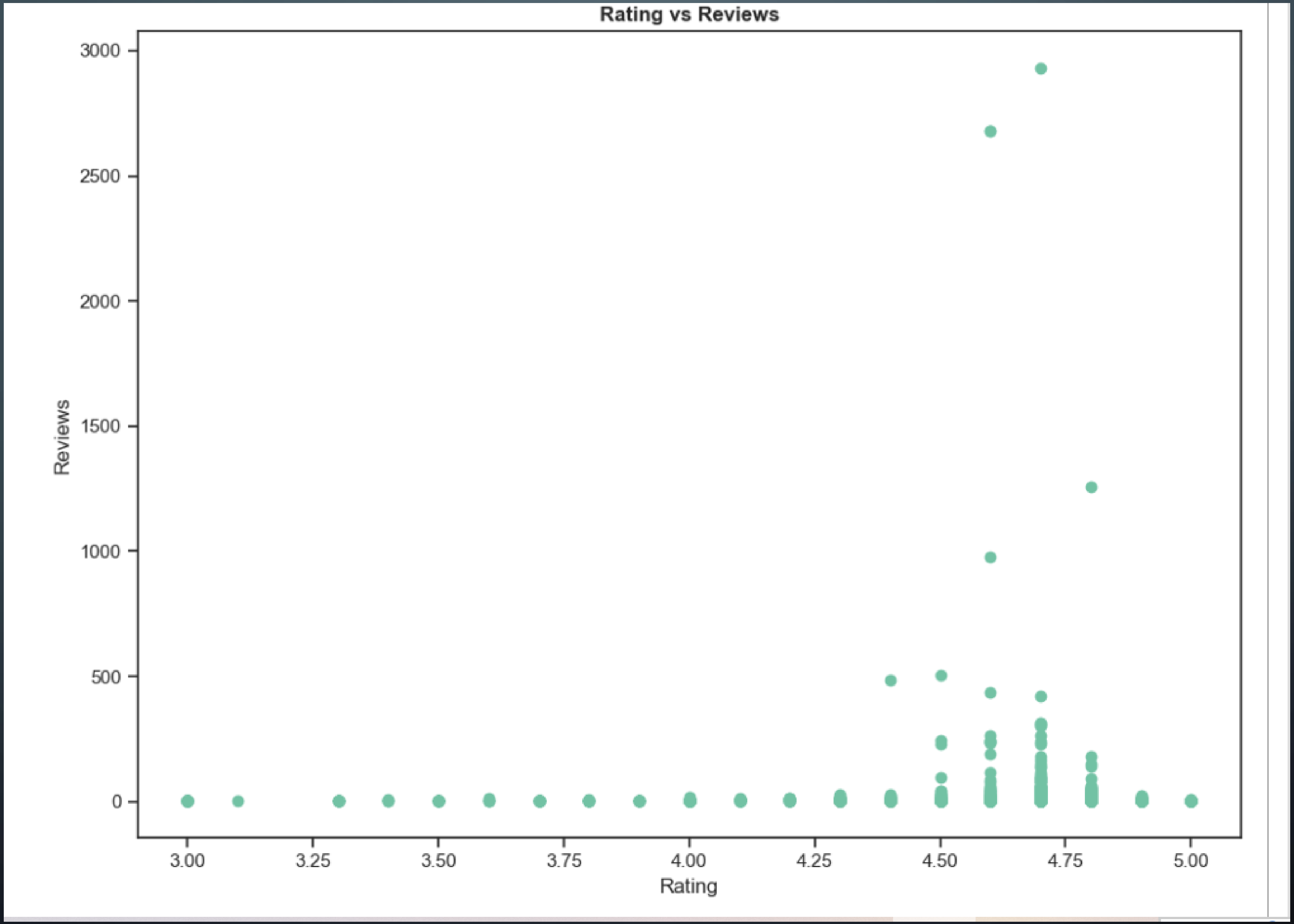
```
Number of outliers: 123
Max outlier value: 3.8
Min outlier value: 3.0
                                                   Name  Price  Rating  \
63               Flying Embers Wild Berry Hard Kombucha  15.39     3.6
254      Happy Dad Hard Seltzer Death Row Records Grape  19.99     3.7
312   Modelo Chelada Sandia Picante Mexican Import F...   3.99     3.3
693        Founders All Day Haze, Session Hazy IPA Beer  18.99     3.8
709                  Sierra Nevada Seasonal Oktoberfest   0.00     3.0
...                                                 ...    ...     ...
6021                                Fireball X Lemonade   3.38     3.4
6082                    Pontoon Down With The Thickness  16.22     3.8
6195                     Right Proper Senate Beer Lager  13.99     3.8
6228                              Aslin Power Moves IPA  14.99     3.0
6235          1911 Maple Bourbon Barrel Aged Hard Cider  14.99     3.0
```

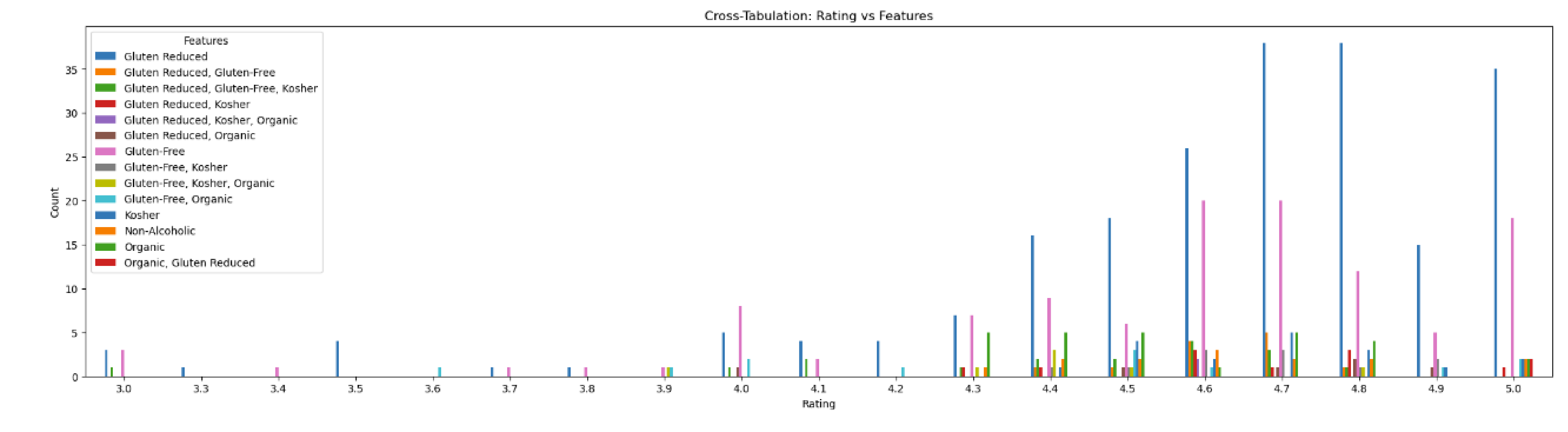
DEALING WITH OUTLIERS

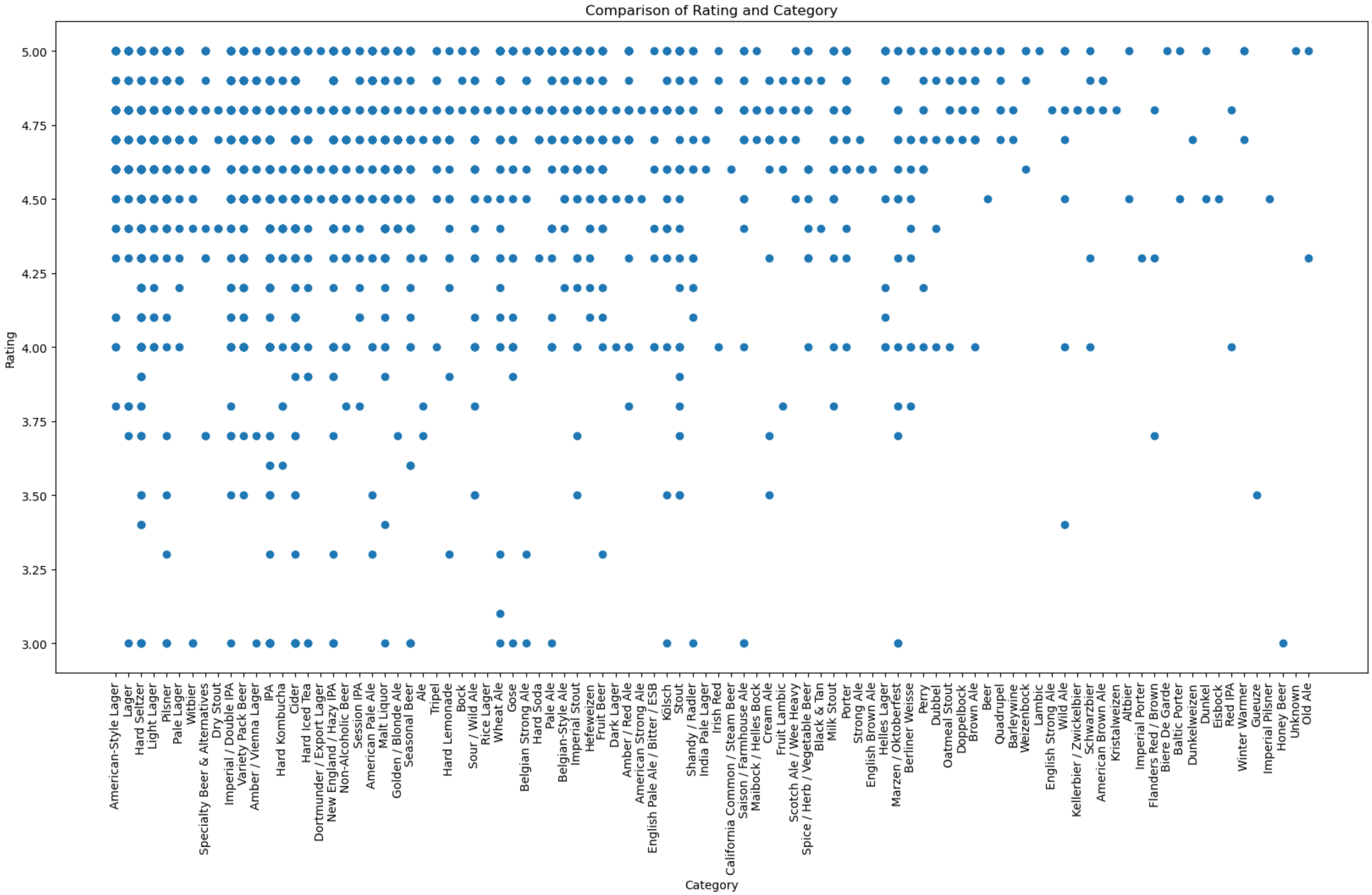


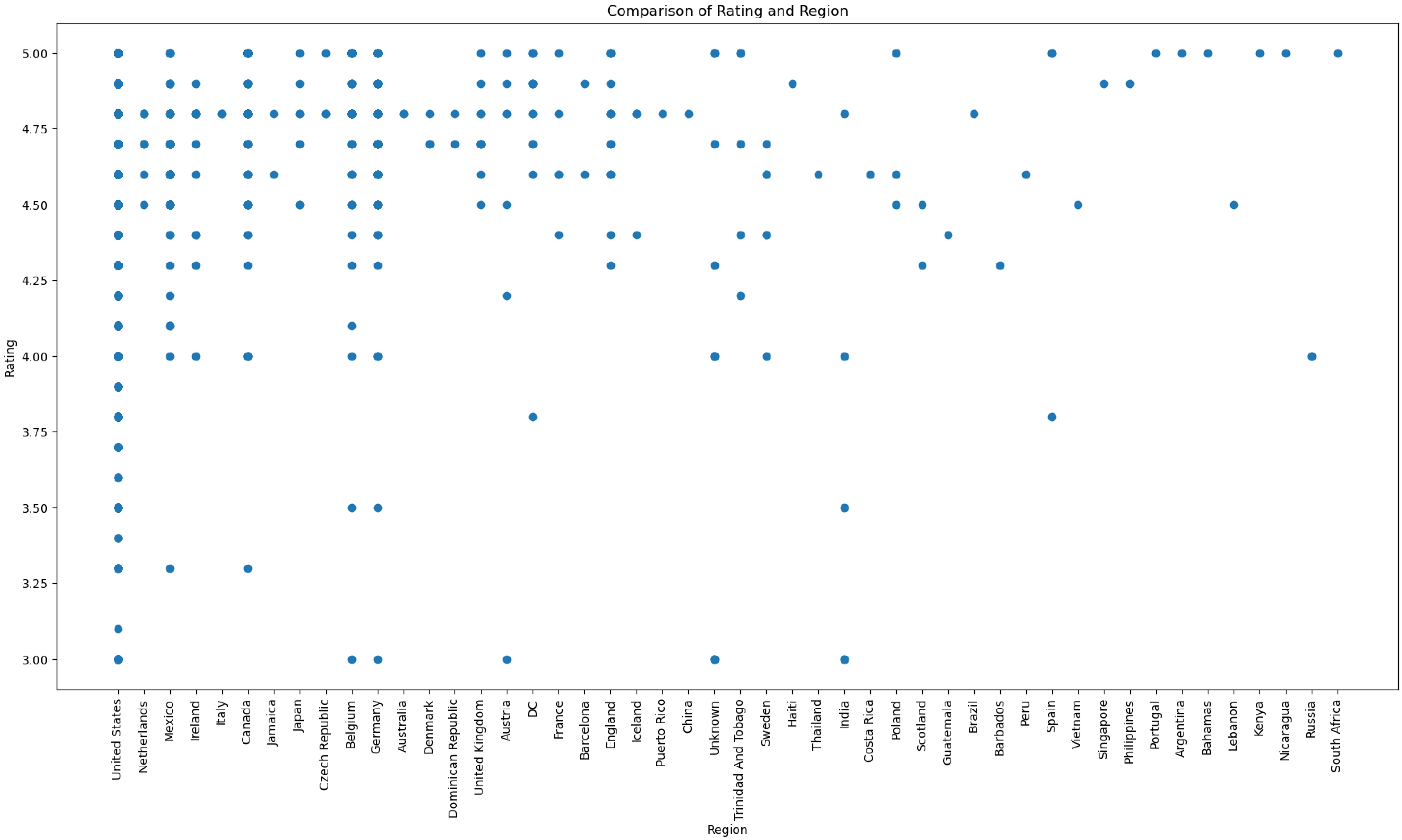

Distribution of Rating
Frequency
0
50
100
150
200
250
300
350
400
2.5
3.0
3.5
4.0
4.5
5.0
5.5
Rating


Rating vs Price
Price
200
150
100
50
0
3.00
3.25
3.50
3.75
4.00
4.25
4.50
4.75
5.00
Rating

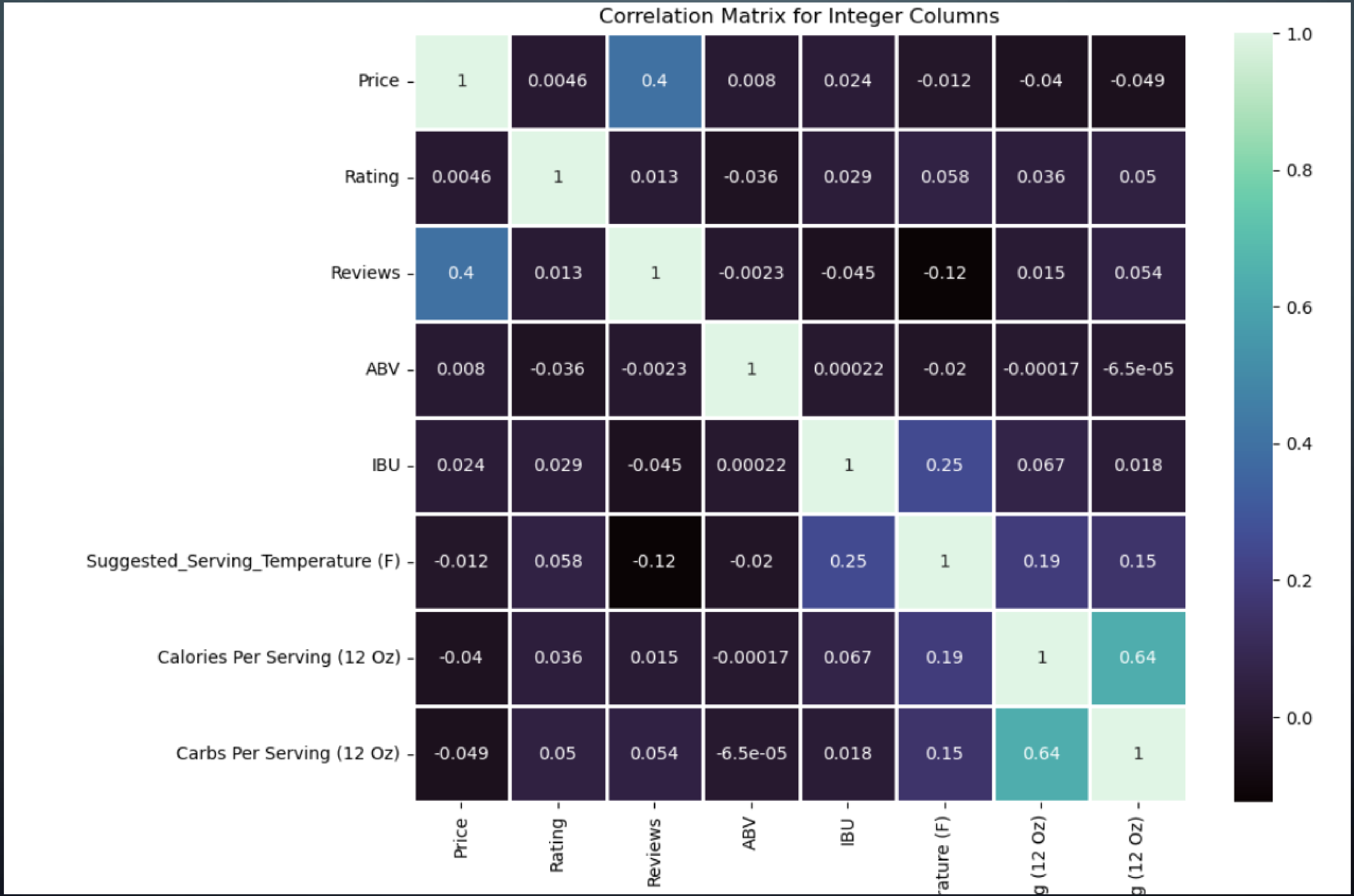

Rating vs Reviews
Reviews
3000
2500
2000
1500
1000
500
0
3.00
3.25
3.50
3.75
4.00
4.25
4.50
4.75
5.00
Rating


Cross-Tabulation: Rating vs Features
Features
Gluten Reduced
Gluten Reduced, Gluten-Free
Gluten Reduced, Gluten-Free, Kosher
Gluten Reduced, Kosher
Gluten Reduced, Kosher, Organic
Gluten Reduced, Organic
Gluten-Free
Gluten-Free, Kosher
Gluten-Free, Kosher, Organic
Gluten-Free, Organic
Kosher
Non-Alcoholic
Organic
Organic, Gluten Reduced
Count
35
30
25
20
15
10
5
0
3.0
3.3
3.4
3.5
3.6
3.7
3.8
3.9
4.0
4.1
4.2
4.3
4.4
4.5
4.6
4.7
4.8
4.9
5.0
Rating

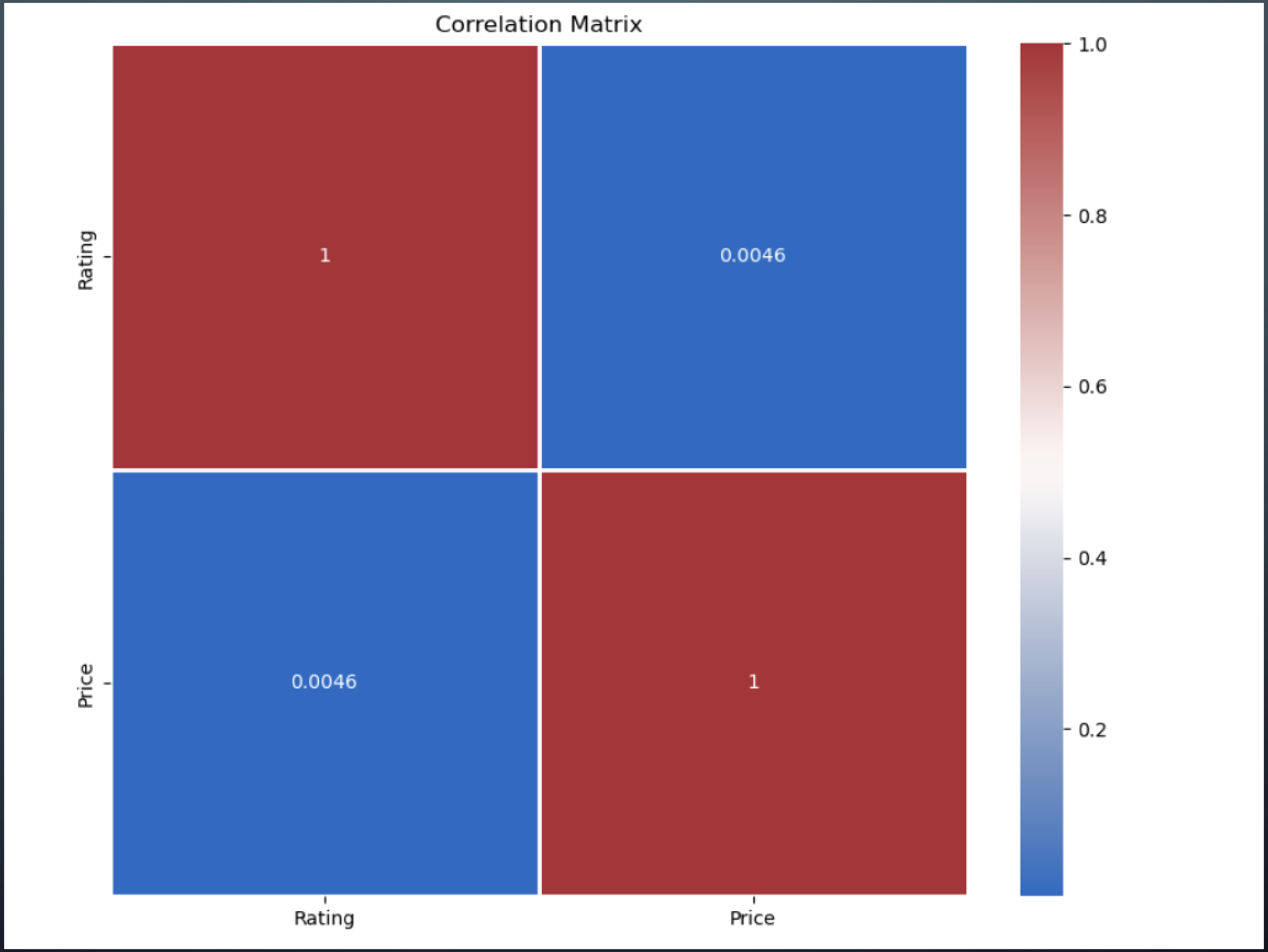

Comparison of Rating and Category
Rating
5.00
4.75
4.50
4.25
4.00
3.75
3.50
3.25
3.00
Category
American-Style Lager
Lager
Hard Seltzer
Light Lager
Pilsner
Pale Lager
Witbier
Specialty Beer & Alternatives
Dry Stout
Imperial / Double IPA
Variety Pack Beer
Amber / Vienna Lager
IPA
Hard Kombucha
Cider
Hard Iced Tea
Dortmunder / Export Lager
New England / Hazy IPA
Non-Alcoholic Beer
Session IPA
American Pale Ale
Malt Liquor
Golden / Blonde Ale
Seasonal Beer
Ale
Tripel
Hard Lemonade
Bock
Sour / Wild Ale
Rice Lager
Wheat Ale
Gose
Belgian Strong Ale
Hard Soda
Pale Ale
Belgian-Style Ale
Imperial Stout
Hefeweizen
Fruit Beer
Dark Lager
Amber / Red Ale
American Strong Ale
English Pale Ale / Bitter / ESB
Kölsch
Stout
Shandy / Radler
India Pale Lager
Irish Red
California Common / Steam Beer
Saison / Farmhouse Ale
Maibock / Helles Bock
Cream Ale
Fruit Lambic
Scotch Ale / Wee Heavy
Spice / Herb / Vegetable Beer
Black & Tan
Milk Stout
Porter
Strong Ale
English Brown Ale
Helles Lager
Marzen / Oktoberfest
Berliner Weisse
Perry
Dubbel
Oatmeal Stout
Doppelbock
Brown Ale
Beer
Quadrupel
Barleywine
Weizenbock
Lambic
English Strong Ale
Wild Ale
Kellerbier / Zwickelbier
Schwarzbier
American Brown Ale
Kristalweizen
Altbier
Imperial Porter
Flanders Red / Brown
Biere De Garde
Baltic Porter
Dunkelweizen
Dunkel
Eisbock
Red IPA
Winter Warmer
Gueuze
Imperial Pilsner
Honey Beer
Unknown
Old Ale

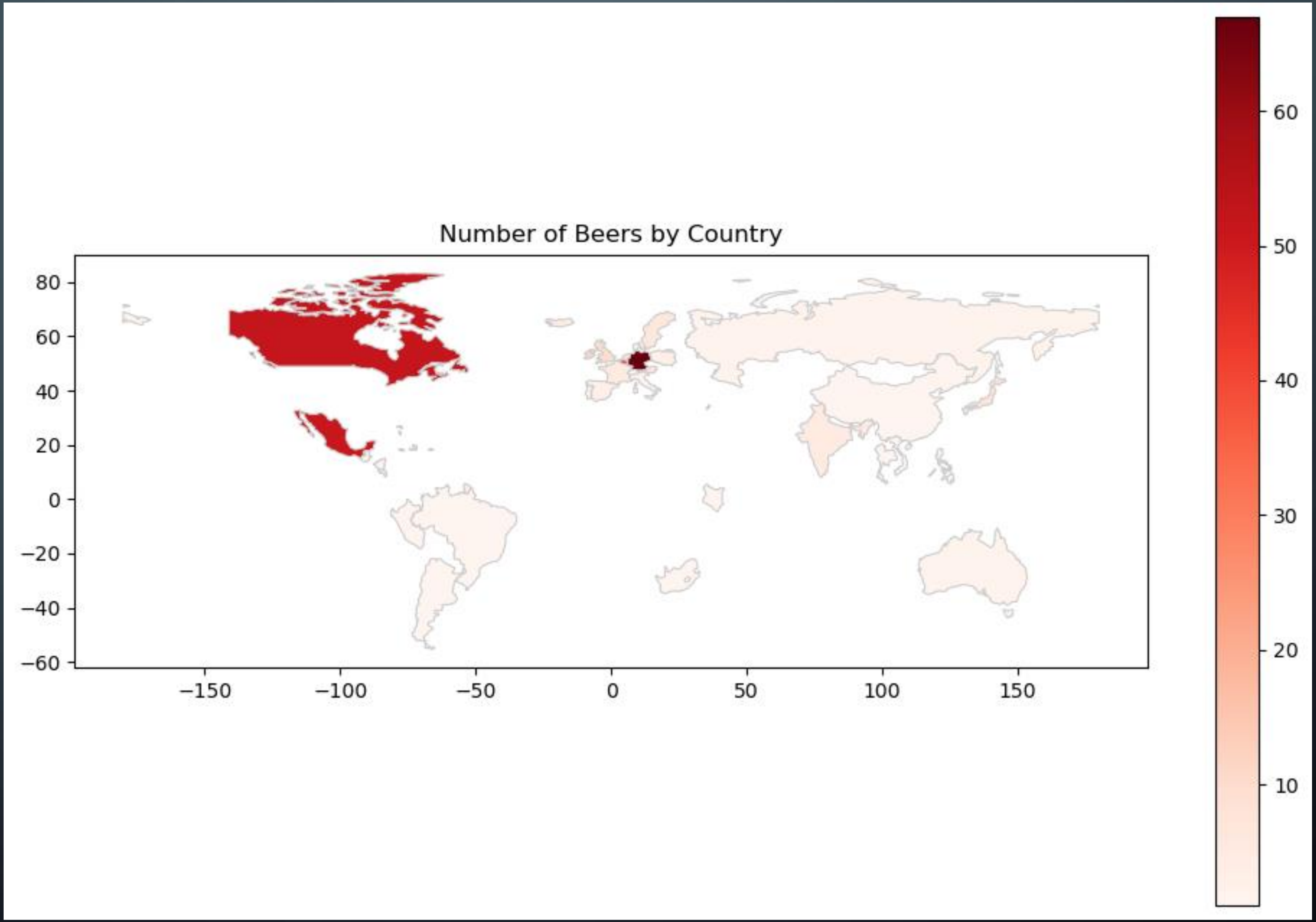

Comparison of Rating and Region
Rating
5.00
4.75
4.50
4.25
4.00
3.75
3.50
3.25
3.00
Region
United States
Netherlands
Mexico
Ireland
Italy
Canada
Jamaica
Japan
Czech Republic
Belgium
Germany
Australia
Denmark
Dominican Republic
United Kingdom
Austria
DC
France
Barcelona
England
Iceland
Puerto Rico
China
Unknown
Trinidad And Tobago
Sweden
Haiti
Thailand
India
Costa Rica
Poland
Scotland
Guatemala
Brazil
Barbados
Peru
Spain
Vietnam
Singapore
Philippines
Portugal
Argentina
Bahamas
Lebanon
Kenya
Nicaragua
Russia
South Africa

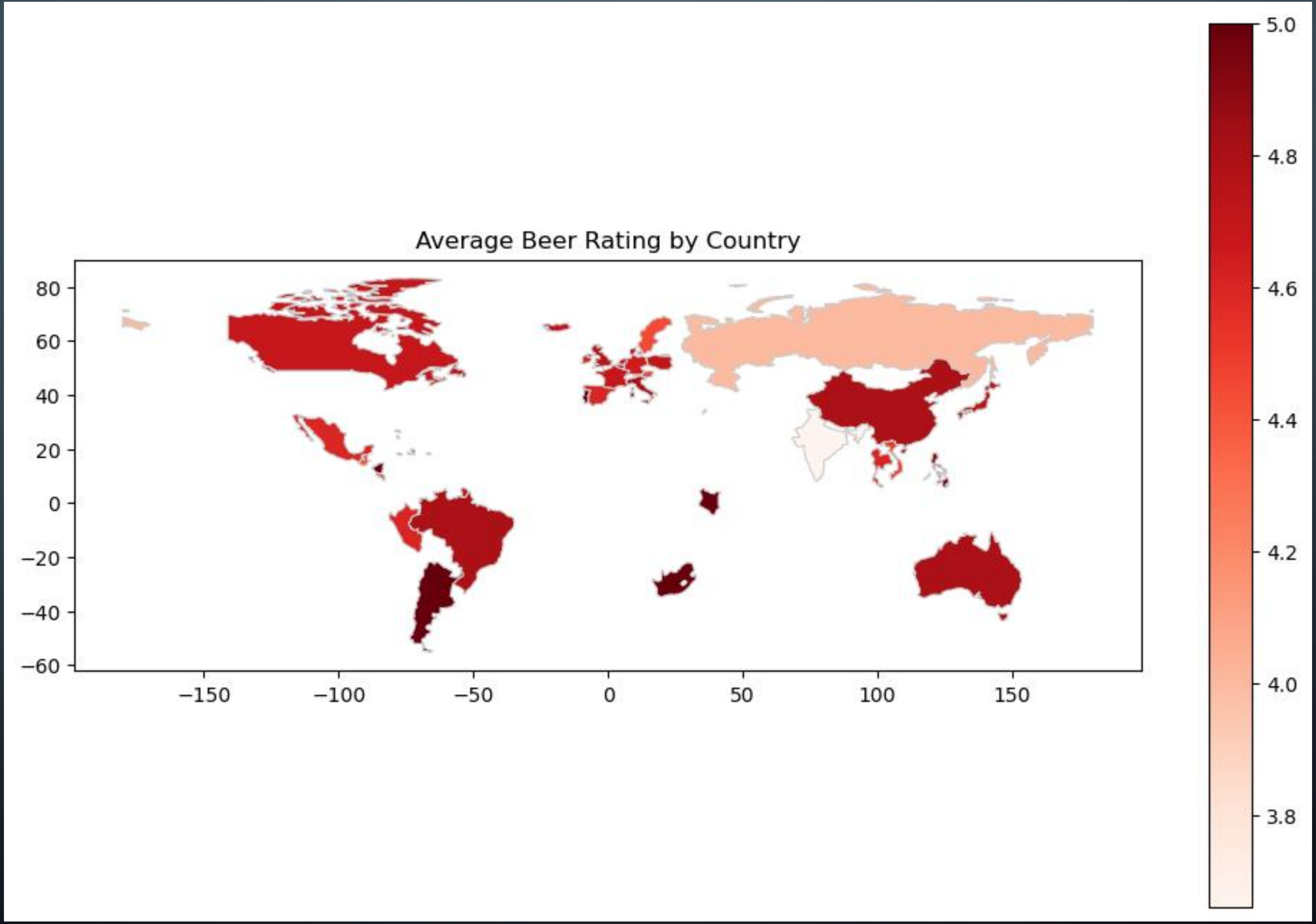
Correlation Matrix for Integer Columns

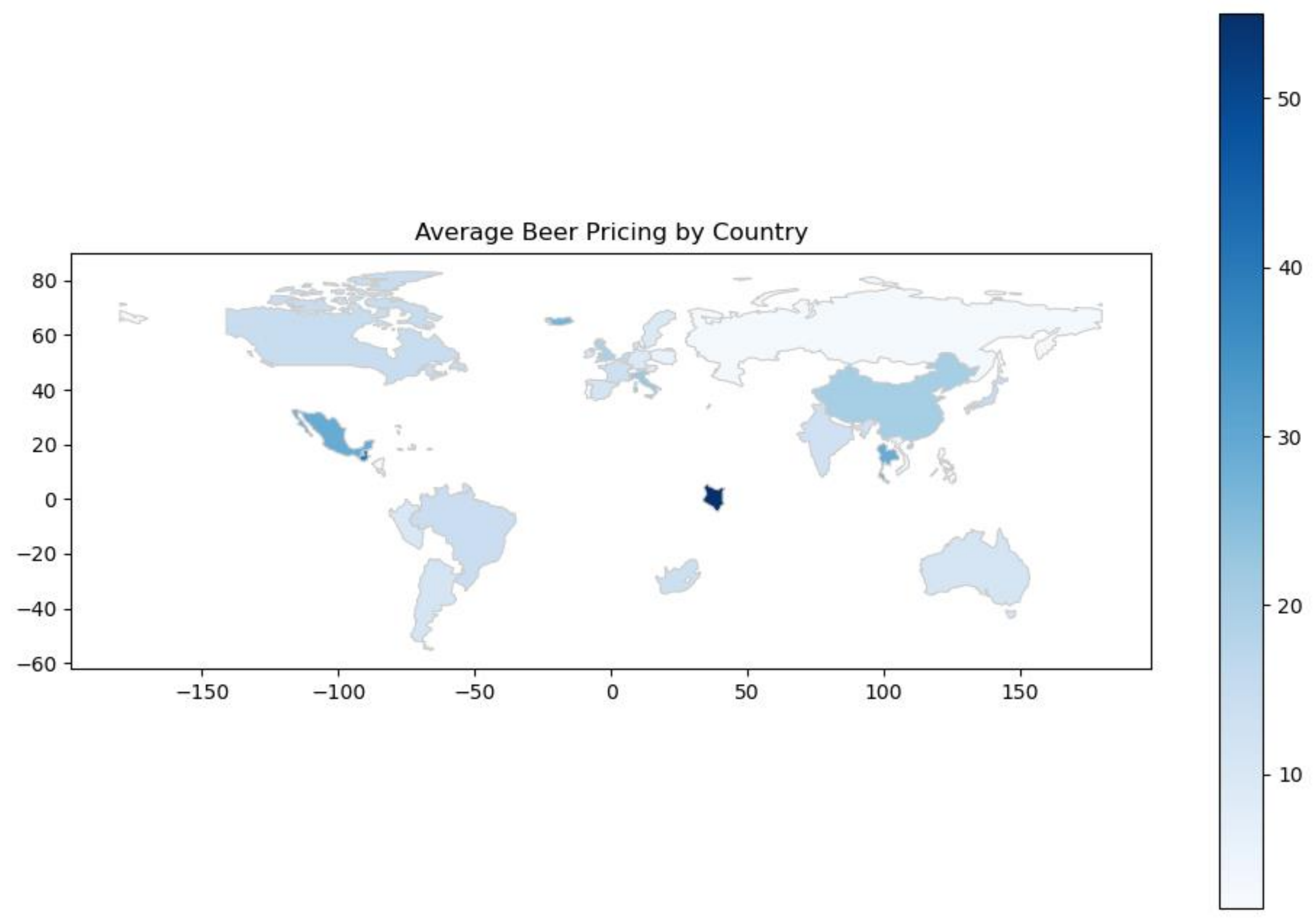


	Price	Rating	Reviews	ABV	IBU	rature (F)	g (12 Oz)	g (12 Oz)
Price	1	0.0046	0.4	0.008	0.024	-0.012	-0.04	-0.049
Rating	0.0046	1	0.013	-0.036	0.029	0.058	0.036	0.05
Reviews	0.4	0.013	1	-0.0023	-0.045	-0.12	0.015	0.054
ABV	0.008	-0.036	-0.0023	1	0.00022	-0.02	-0.00017	-6.5e-05
IBU	0.024	0.029	-0.045	0.00022	1	0.25	0.067	0.018
Suggested_Serving_Temperature (F)	-0.012	0.058	-0.12	-0.02	0.25	1	0.19	0.15
Calories Per Serving (12 Oz)	-0.04	0.036	0.015	-0.00017	0.067	0.19	1	0.64
Carbs Per Serving (12 Oz)	-0.049	0.05	0.054	-6.5e-05	0.018	0.15	0.64	1


Correlation Matrix
Rating
Price
1
0.0046
0.0046
1
Rating
Price
1.0
0.8
0.6
0.4
0.2


Number of Beers by Country
80
60
40
20
0
−20
−40
−60
−150
−100
−50
0
50
100
150
60
50
40
30
20
10


Average Beer Rating by Country
80
60
40
20
0
−20
−40
−60
−150
−100
−50
0
50
100
150
5.0
4.8
4.6
4.4
4.2
4.0
3.8


Average Beer Pricing by Country
80
60
40
20
0
−20
−40
−60
−150
−100
−50
0
50
100
150
50
40
30
20
10

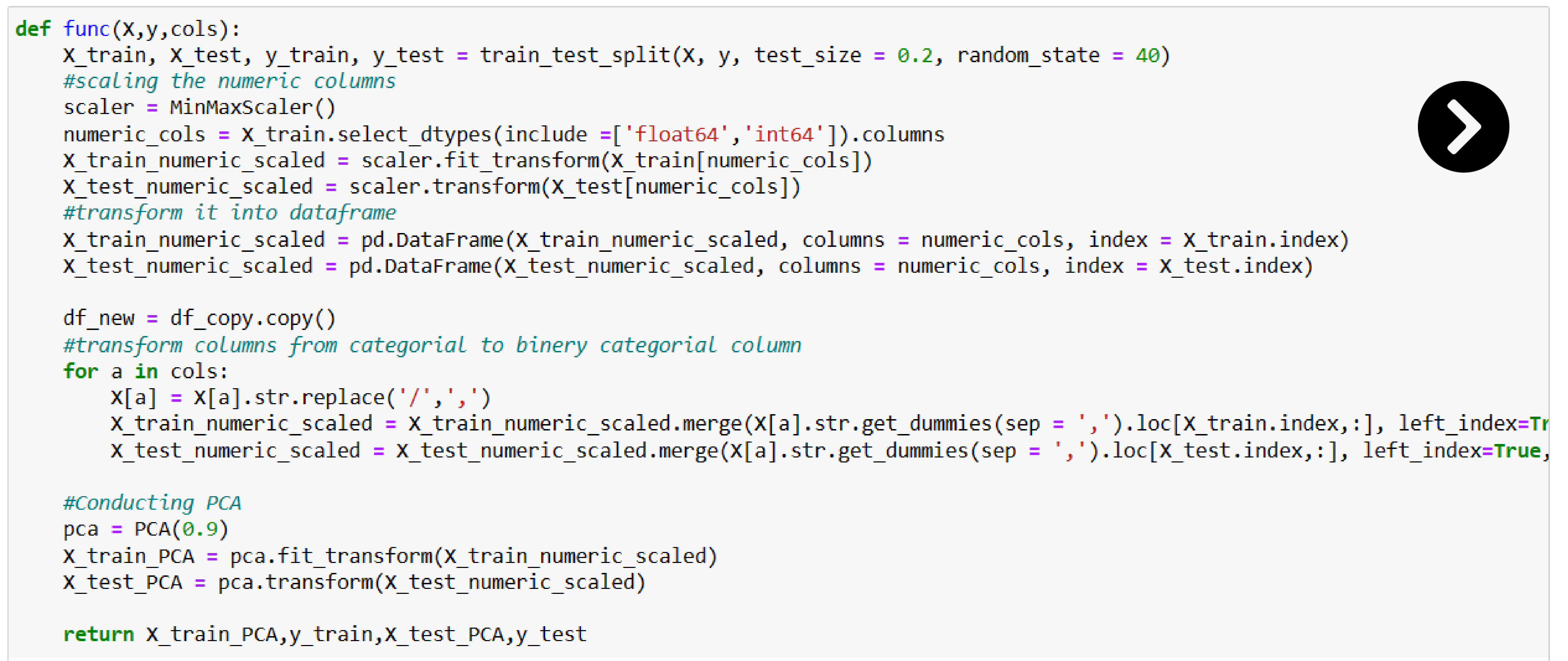
MACHINE LEARNING

- Preparing the data
- Comparing different models
- Hyperparameter tuning
- Final evaluation

PREPARING THE DATA

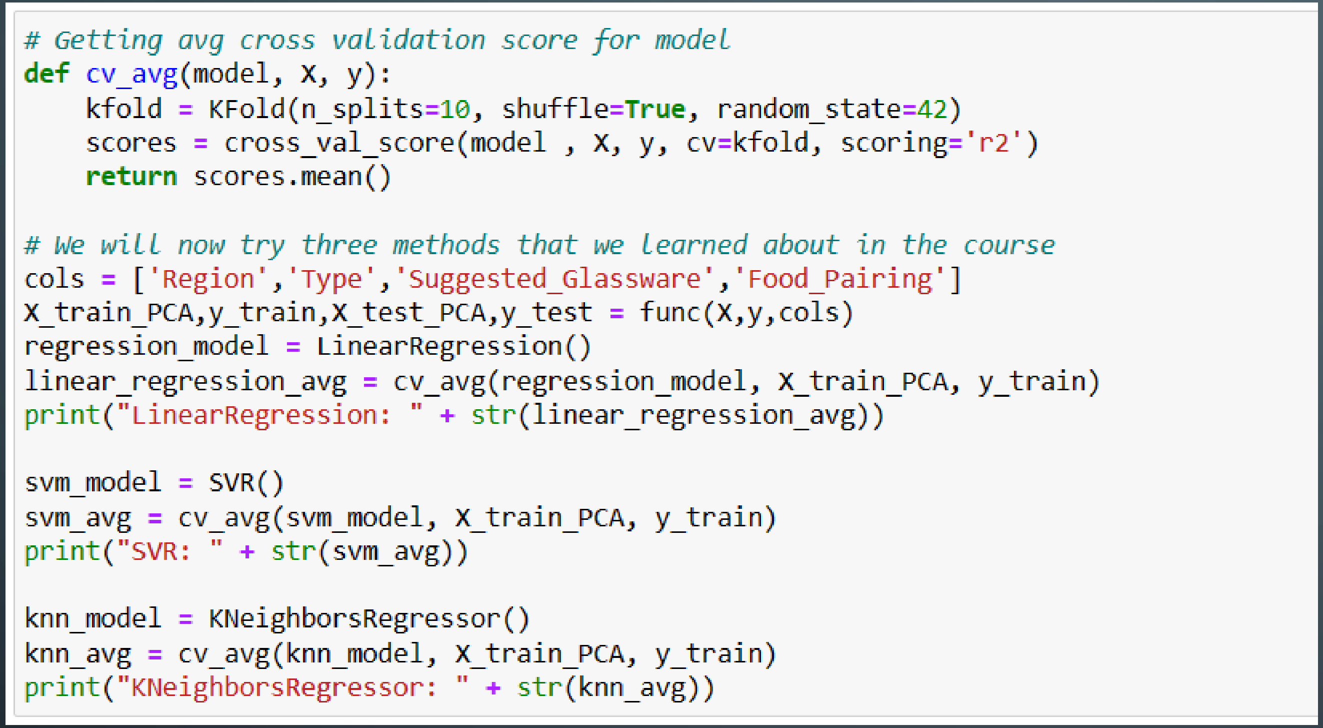
- Splitting data to train/test
- Scaling numeric features
- Min/max scaler features
- PCA

PREPARING THE DATA – CODE / FIRST MODEL



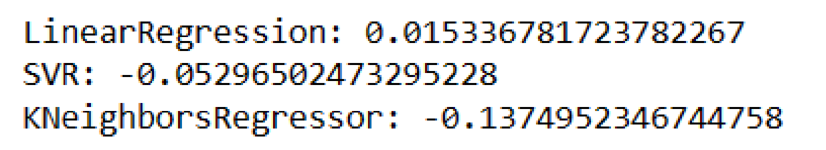
```
def func(X,y,cols):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 40)
    #scaling the numeric columns
    scaler = MinMaxScaler()
    numeric_cols = X_train.select_dtypes(include =['float64','int64']).columns
    X_train_numeric_scaled = scaler.fit_transform(X_train[numeric_cols])
    X_test_numeric_scaled = scaler.transform(X_test[numeric_cols])
    #transform it into dataframe
    X_train_numeric_scaled = pd.DataFrame(X_train_numeric_scaled, columns = numeric_cols, index = X_train.index)
    X_test_numeric_scaled = pd.DataFrame(X_test_numeric_scaled, columns = numeric_cols, index = X_test.index)

    df_new = df_copy.copy()
    #transform columns from categorial to binery categorial column
    for a in cols:
        X[a] = X[a].str.replace('/',',')
        X_train_numeric_scaled = X_train_numeric_scaled.merge(X[a].str.get_dummies(sep = ',').loc[X_train.index,:], left_index=Tr
        X_test_numeric_scaled = X_test_numeric_scaled.merge(X[a].str.get_dummies(sep = ',').loc[X_test.index,:], left_index=True,

    #Conducting PCA
    pca = PCA(0.9)
    X_train_PCA = pca.fit_transform(X_train_numeric_scaled)
    X_test_PCA = pca.transform(X_test_numeric_scaled)

    return X_train_PCA,y_train,X_test_PCA,y_test
```

PREPARING THE DATA – CODE / FIRST MODEL



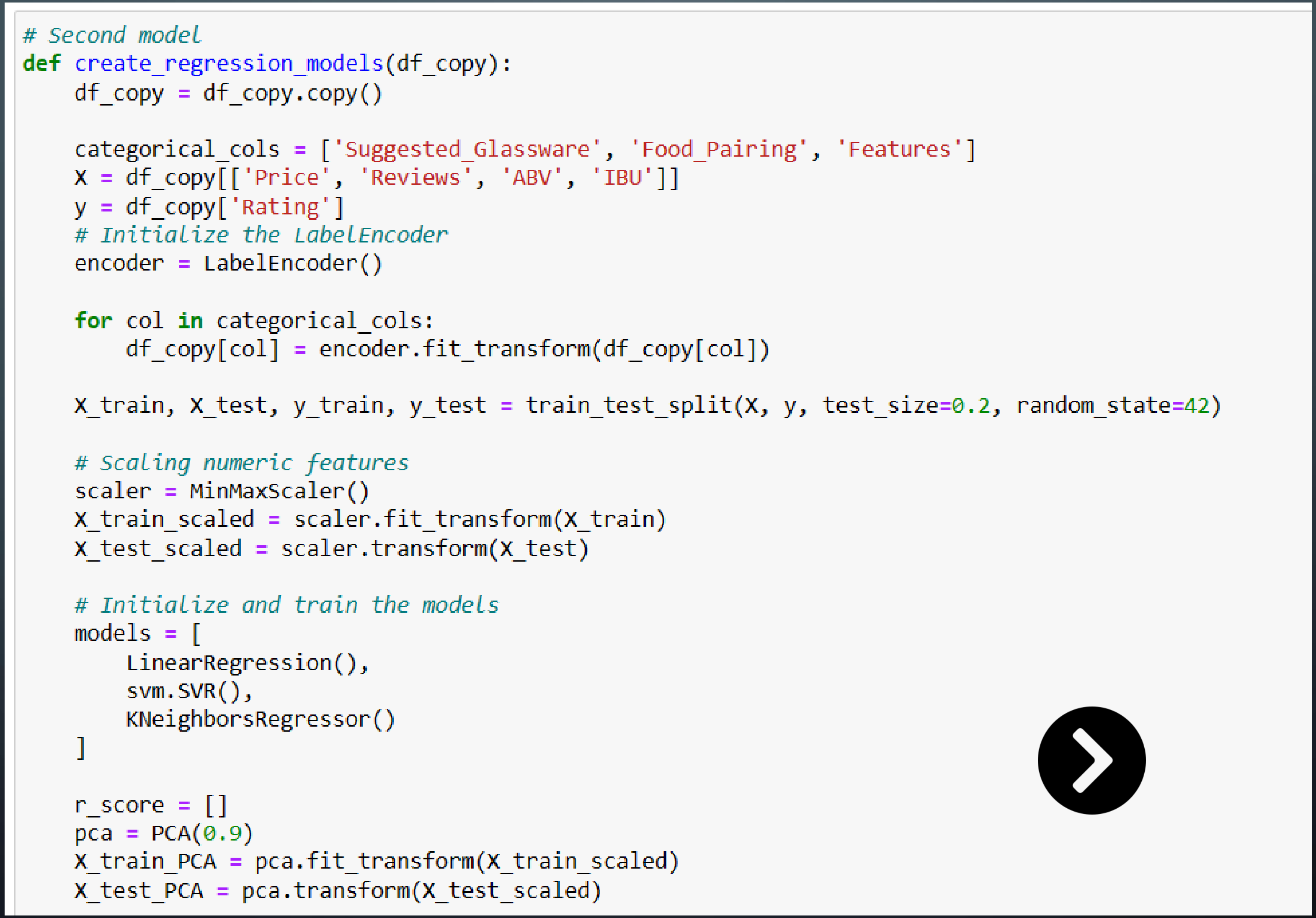
```python
# Getting avg cross validation score for model
def cv_avg(model, X, y):
    kfold = KFold(n_splits=10, shuffle=True, random_state=42)
    scores = cross_val_score(model , X, y, cv=kfold, scoring='r2')
    return scores.mean()

# We will now try three methods that we learned about in the course
cols = ['Region','Type','Suggested_Glassware','Food_Pairing']
X_train_PCA,y_train,X_test_PCA,y_test = func(X,y,cols)
regression_model = LinearRegression()
linear_regression_avg = cv_avg(regression_model, X_train_PCA, y_train)
print("LinearRegression: " + str(linear_regression_avg))

svm_model = SVR()
svm_avg = cv_avg(svm_model, X_train_PCA, y_train)
print("SVR: " + str(svm_avg))

knn_model = KNeighborsRegressor()
knn_avg = cv_avg(knn_model, X_train_PCA, y_train)
print("KNeighborsRegressor: " + str(knn_avg))
```

RESULTS



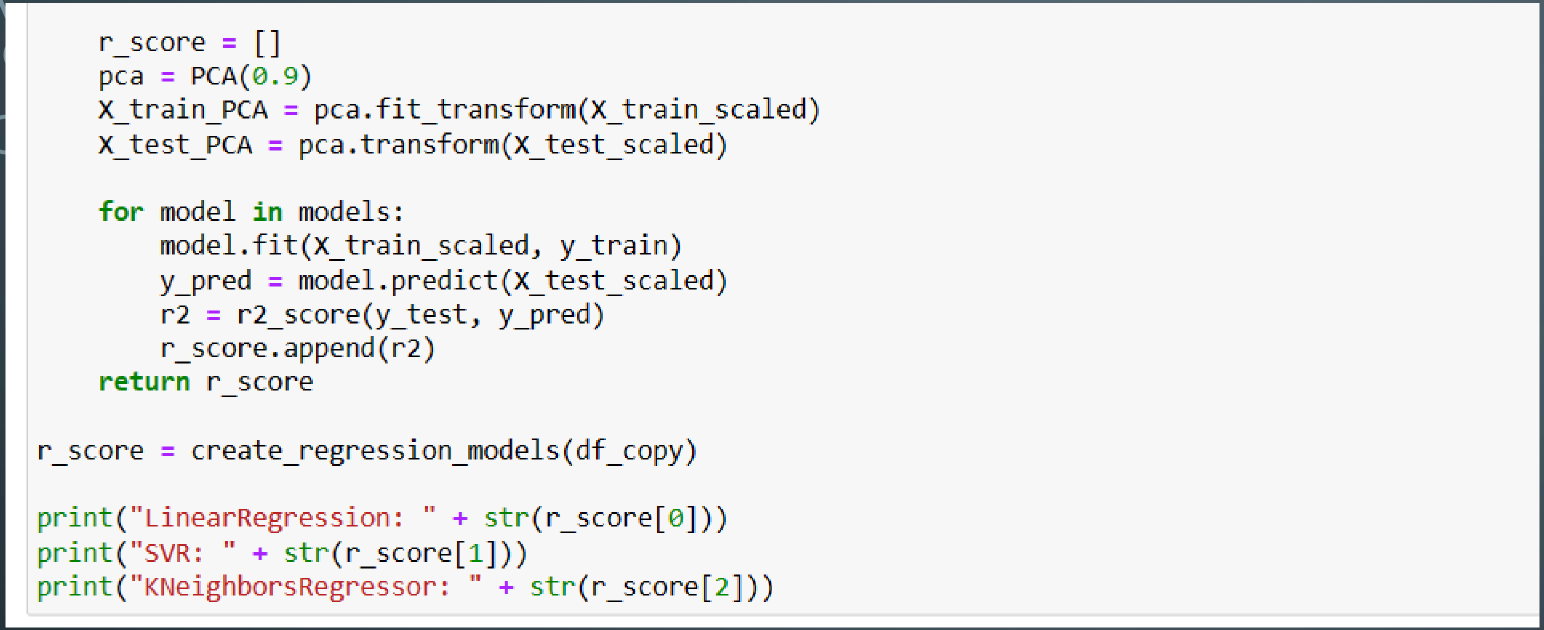
```
LinearRegression: 0.015336781723782267
SVR: -0.05296502473295228
KNeighborsRegressor: -0.1374952346744758
```

PREPARING THE DATA – CODE / SECOND MODEL



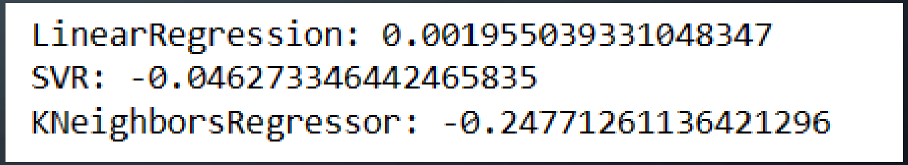
```
# Second model
def create_regression_models(df_copy):
    df_copy = df_copy.copy()

    categorical_cols = ['Suggested_Glassware', 'Food_Pairing', 'Features']
    X = df_copy[['Price', 'Reviews', 'ABV', 'IBU']]
    y = df_copy['Rating']
    # Initialize the LabelEncoder
    encoder = LabelEncoder()

    for col in categorical_cols:
        df_copy[col] = encoder.fit_transform(df_copy[col])

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # Scaling numeric features
    scaler = MinMaxScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)

    # Initialize and train the models
    models = [
        LinearRegression(),
        svm.SVR(),
        KNeighborsRegressor()
    ]

    r_score = []
    pca = PCA(0.9)
    X_train_PCA = pca.fit_transform(X_train_scaled)
    X_test_PCA = pca.transform(X_test_scaled)
```

PREPARING THE DATA – CODE / SECOND MODEL



```
    r_score = []
    pca = PCA(0.9)
    X_train_PCA = pca.fit_transform(X_train_scaled)
    X_test_PCA = pca.transform(X_test_scaled)

    for model in models:
        model.fit(X_train_scaled, y_train)
        y_pred = model.predict(X_test_scaled)
        r2 = r2_score(y_test, y_pred)
        r_score.append(r2)
    return r_score

r_score = create_regression_models(df_copy)

print("LinearRegression: " + str(r_score[0]))
print("SVR: " + str(r_score[1]))
print("KNeighborsRegressor: " + str(r_score[2]))
```

RESULTS



```
LinearRegression: 0.001955039331048347
SVR: -0.046273346442465835
KNeighborsRegressor: -0.24771261136421296
```

CONCLUSION

After examining the first model and the second model, we reached a better result in the first model than in the second model. But we did not reach a sufficient result in order to predict our research question and therefore our conclusion is that it is not possible way to predict the success of a beer using ratings of other beers.

The best prediction we’ve accomplished:

LinearRegression: 0.015336781723782267