scrapper_clean

April 26, 2021

[1]:
```
# !pip install webdriver-manager
```

[2]:
```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.chrome.options import Options  # for suppressing the␣
 ↪browser
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from bs4 import BeautifulSoup
import re
import pandas as pd
import os
import time
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

0.0.1 Scrapping data for Robusta coffee species

[3]:
```
cleaned_robusta_df = pd.read_csv('robusta_data_cleaned.csv')
```

Extract all the rows

[4]:
```
url = 'https://database.coffeeinstitute.org/coffees/robusta'

option = webdriver.ChromeOptions()
option.add_argument('headless') # use webdriver without opening the browser␣
 ↪window
driver = webdriver.Chrome(ChromeDriverManager().install(), options=option)

driver.implicitly_wait(30)
driver.get(url)
time.sleep(8)
```

```
====== WebDriver manager ======
Current google-chrome version is 89.0.4389
Get LATEST driver version for 89.0.4389
Driver [/Users/david/.wdm/drivers/chromedriver/mac64/89.0.4389.23/chromedriver]
found in cache
```

[5]:
```
robusta_main = BeautifulSoup(driver.page_source, 'lxml')
rows = robusta_main.find_all('tr', class_=['odd','even']) # extract odd and␣
↪even rows from the table
```

Extract the url content for each sample

[27]:
```
root_url = 'https://database.coffeeinstitute.org'
links = [root_url + row.find('a')['href'] for row in rows] # extract the link␣
↪for each row in the database
robusta_df = pd.DataFrame() # initialize df to store all records
print(links[:3])
```

```
['https://database.coffeeinstitute.org/coffee/722152',
'https://database.coffeeinstitute.org/coffee/758792',
'https://database.coffeeinstitute.org/coffee/805984']
```

Store the information of each sample in a DataFrame

[28]:
```
for link in links:
    driver.implicitly_wait(10)
    driver.get(link)
    # Wait until the element with TAG_NAME 'tr' has been loaded
    element = WebDriverWait(driver, 10).until(
        EC.presence_of_element_located((By.TAG_NAME, "tr"))
    )

    info = BeautifulSoup(driver.page_source, 'lxml')
    robusta_dic = {}

    for i in range(1,len(info.find_all('table'))):
        # find all 'th' the headers of the table
        table1 = info.find_all('table')[i].find_all('th')
        table_keys = [x.get_text() for x in table1]
        # find all the data of each header
        table2 = info.find_all('table')[i].find_all('td')
        table_values = [x.get_text() if x.get_text() != '' else np.nan for x in␣
↪table2]

        tmp_dic = dict(zip(table_keys, table_values))
        robusta_dic.update(tmp_dic) # update dictionary with the information␣
↪for robusta species

    df = pd.DataFrame(robusta_dic, index=[0]) # add index=[0] because the␣
↪values are scalars (not in a list)
    robusta_df = pd.concat([robusta_df,df], ignore_index=True, sort=False)
    time.sleep(5)

# quit the driver
# driver.quit()
```

[30]:
```
robusta_df.head(3)
```

[30]:
```
  Country of Origin Number of Bags           Farm Name Bag Weight Lot Number  \
0             India            100  Sethuraman Estate       60 kg  Lot No 22
1            Mexico            320                 n/a      60 kg    1540038
2             India            170  Sethuraman Estate       60 kg         27

                                  In-Country Partner           Mill  \
0  NKG Quality Service (a division of Bernhard Ro…  Kaapi Royale
1  NKG Quality Service (a division of Bernhard Ro…    AMSA - ECOM
2  NKG Quality Service (a division of Bernhard Ro…  Kaapi Royale

  Harvest Year       ICO Number        Grading Date  … Moisture  \
0         2020  14/1148/2020/11   August 20th, 2020  …     12 %
1         2019    016-2222-0409 November 3rd, 2020  …     10 %
2         2020  14/1148/2020/12   August 20th, 2020  …     12 %

          Color Category One Defects Category Two Defects Quakers      \
0         Green        0 full defects       0 full defects       0 NaN
1  Yellow-Green        1 full defects       7 full defects       3 NaN
2         Green        0 full defects       0 full defects       0 NaN

          Expiration                                 Certification Body  \
0   August 20th, 2021  NKG Quality Service (a division of Bernhard Ro…
1  November 3rd, 2021  NKG Quality Service (a division of Bernhard Ro…
2   August 20th, 2021  NKG Quality Service (a division of Bernhard Ro…

                        Certification Address         Certification Contact
0  Bahnhofstrasse 22 6300 Zug, Switzerland   Gloria Pedroza - +41417287296
1  Bahnhofstrasse 22 6300 Zug, Switzerland   Gloria Pedroza - +41417287296
2  Bahnhofstrasse 22 6300 Zug, Switzerland   Gloria Pedroza - +41417287296

[3 rows x 40 columns]
```

0.0.2 Scrapping data for Arabica coffee species

[31]:
```
cleaned_arabica_df = pd.read_csv('arabica_data_cleaned.csv')
```

Extract all the rows from different pages

[32]:
```
url = 'https://database.coffeeinstitute.org/coffees/arabica'

option = webdriver.ChromeOptions()
option.add_argument('headless')
driver = webdriver.Chrome(ChromeDriverManager().install(), options=option)

driver.implicitly_wait(30)
driver.get(url)
time.sleep(8)

arabica_main = BeautifulSoup(driver.page_source, 'lxml')
rows = arabica_main.find_all('tr', class_=['odd','even']) # extract odd and␣
 ↪even rows from the table

while len(arabica_main.find_all('a',class_='paginate_button next disabled')) <␣
 ↪1:
    page_buttons = driver.find_elements_by_class_name('paginate_button')
    page_buttons[-1].click() # click next
    time.sleep(6)
    arabica_main = BeautifulSoup(driver.page_source, 'lxml') # get arabica_main␣
 ↪for next page
    rows += arabica_main.find_all('tr', class_=['odd','even']) # add new rows
```

```
====== WebDriver manager ======
Current google-chrome version is 89.0.4389
Get LATEST driver version for 89.0.4389
Driver [/Users/david/.wdm/drivers/chromedriver/mac64/89.0.4389.23/chromedriver]
found in cache
```

Extract the url content for each sample

[33]:
```
root_url = 'https://database.coffeeinstitute.org'
links = [root_url + row.find('a')['href'] for row in rows] # extract the link␣
 ↪for each row in the database
arabica_df = pd.DataFrame() # initialize df to store all records
print(links[-3:])
```

```
['https://database.coffeeinstitute.org/coffee/813284',
'https://database.coffeeinstitute.org/coffee/564165',
'https://database.coffeeinstitute.org/coffee/799551']
```

Store the information for each sample in a DataFrame

[34]:
```
for link in links:
    driver.implicitly_wait(10)
    driver.get(link)
    # Wait until the element with TAG_NAME 'tr' has been loaded
    element = WebDriverWait(driver, 10).until(
        EC.presence_of_element_located((By.TAG_NAME, "tr"))
    )

    info = BeautifulSoup(driver.page_source, 'lxml')
    arabica_dic = {}

    for i in range(1,len(info.find_all('table'))):
        # find all 'th' the headers of the table
        table1 = info.find_all('table')[i].find_all('th')
        table_keys = [x.get_text() for x in table1]
        # find all the data of each header
        table2 = info.find_all('table')[i].find_all('td')
        table_values = [x.get_text() if x.get_text() != '' else np.nan for x in␣
 ↪table2]

        tmp_dic = dict(zip(table_keys, table_values))
        arabica_dic.update(tmp_dic) # update dictionary with the information␣
 ↪for arabica species

    df = pd.DataFrame(arabica_dic, index=[0]) # add index=[0] because the␣
 ↪values are scalars (not in a list)
    arabica_df = pd.concat([arabica_df,df], ignore_index=True, sort=False)
    time.sleep(4)

# quit the driver
# driver.quit()
```

[39]:
```
arabica_df.head(3)
```

[39]:
```
  Country of Origin Number of Bags   Farm Name Bag Weight Lot Number  \
0            Taiwan              3                 60 kg     202004
1             Kenya            320         N/A      69 kg    1542247
2          Ethiopia             37  Honey Drip      20 kg          1

                                    In-Country Partner         Mill  \
0  NKG Quality Service (a division of Bernhard Ro…
1  NKG Quality Service (a division of Bernhard Ro…  Tylor Winch
2                                Japan Coffee Exchange          non
```

```
  Harvest Year     ICO Number         Grading Date  … Moisture   Color  \
0         2020            NaN  September 1st, 2020  …     11 %   None
1         2019  037-1673-5107   October 30th, 2020  …     11 %  Green
2         2020            NaN      July 14th, 2020  …     11 %  Green

  Category One Defects Category Two Defects Quakers              Expiration  \
0      0 full defects      0 full defects       0 NaN  September 1st, 2021
1      0 full defects      1 full defects       0 NaN   October 30th, 2021
2      0 full defects      4 full defects       3 NaN      July 14th, 2021

                               Certification Body  \
0  NKG Quality Service (a division of Bernhard Ro…
1  NKG Quality Service (a division of Bernhard Ro…
2                             Japan Coffee Exchange

                            Certification Address  \
0          Bahnhofstrasse 22 6300 Zug, Switzerland
1          Bahnhofstrasse 22 6300 Zug, Switzerland
2  413-0002        -                  1173-58 Izu…

                        Certification Contact
0           Gloria Pedroza - +41417287296
1           Gloria Pedroza - +41417287296
2       Koju Matsuzawa - +81(0)9085642901

[3 rows x 40 columns]
```

0.0.3 Cleaning

[343]:
```python
# Drop empty column (after Quaker)
arabica_df.drop('', axis=1, inplace=True)
robusta_df.drop('', axis=1, inplace=True)
```

[370]:
```python
# Change the names of the columns to match the format of the cleaned datasets
new_names = []
for col in arabica_df.columns:
    new_names.append(col.replace(' ','.').replace('-','.'))

arabica_df.columns = new_names
robusta_df.columns = new_names
```

[374]:
```python
# Store the raw data
arabica_df.to_csv('arabica_raw.csv', index=False)
robusta_df.to_csv('robusta_raw.csv', index=False)
```

```
[390]: # arabica_df = pd.read_csv('arabica_raw.csv')
       # robusta_df = pd.read_csv('robusta_raw.csv')
```

```
[391]: # Region, Owner, Company, Farm.Name and Mill to lower
       cols_lower = ['Region', 'Owner', 'Company', 'Farm.Name', 'Mill']
       for col in cols_lower:
           arabica_df.loc[:,col] = arabica_df.loc[:,col].str.lower()
           robusta_df.loc[:,col] = robusta_df.loc[:,col].str.lower()
```

```
[392]: # Add column Species and Unnamed: 0 which keeps an index
       arabica_df['Species'] = 'Arabica'
       robusta_df['Species'] = 'Robusta'
       arabica_df.index += 1313
       robusta_df.index += 29
       arabica_df.index.rename('Unnamed: 0', inplace=True)
       robusta_df.index.rename('Unnamed: 0', inplace=True)
       arabica_df.reset_index(inplace=True)
       robusta_df.reset_index(inplace=True)
       arabica_df.head(3)
```

```
[392]:   Unnamed: 0 Country.of.Origin  Number.of.Bags   Farm.Name Bag.Weight  \
       0       1313            Taiwan               3                  60 kg
       1       1314             Kenya             320         NaN      69 kg
       2       1315          Ethiopia              37  honey drip      20 kg

         Lot.Number                            In.Country.Partner         Mill  \
       0     202004  NKG Quality Service (a division of Bernhard Ro…
       1    1542247  NKG Quality Service (a division of Bernhard Ro…  tylor winch
       2          1                           Japan Coffee Exchange          non

         Harvest.Year     ICO.Number  … Moisture   Color Category.One.Defects  \
       0         2020            NaN  …     11 %    None        0 full defects
       1         2019   037-1673-5107 …     11 %   Green        0 full defects
       2         2020            NaN  …     11 %   Green        0 full defects

         Category.Two.Defects Quakers           Expiration  \
       0       0 full defects       0  September 1st, 2021
       1       1 full defects       0   October 30th, 2021
       2       4 full defects       3       July 14th, 2021

                                     Certification.Body  \
       0  NKG Quality Service (a division of Bernhard Ro…
       1  NKG Quality Service (a division of Bernhard Ro…
       2                            Japan Coffee Exchange

                                  Certification.Address  \
       0           Bahnhofstrasse 22 6300 Zug, Switzerland
```

```
1          Bahnhofstrasse 22 6300 Zug, Switzerland
2   413-0002        -                 1173-58 Izu…

                  Certification.Contact  Species
0          Gloria Pedroza - +41417287296  Arabica
1          Gloria Pedroza - +41417287296  Arabica
2       Koju Matsuzawa - +81(0)9085642901  Arabica

[3 rows x 41 columns]
```

Create altitude_low_meters, altitude_high_meters, altitude_mean_meters Some of the values are very difficult to fix using regular expressions or an automated command, but besides 2 particular cases, we can use str methods to clean the column of altitude and get the values of the previously cleaned datasets to join them together.

[461]:
```
# 2 special cases to be cleaned
arabica_df.loc[4,'Altitude']='1500-2100'
arabica_df.loc[90,'Altitude']='1100'

# use regex to clean data and split by - into two columns (low and high␣
↪altitude)
df = arabica_df.Altitude.str.strip().str.replace(',|\.\d*|\+','').str.
↪split('-', expand=True)
df.columns = ['altitude_low_meters', 'altitude_high_meters']

# make the None values in high column equal to the low column
df.altitude_high_meters[pd.isna(df.altitude_high_meters)] = df.
↪altitude_low_meters[pd.isna(df.altitude_high_meters)]
df.altitude_low_meters[df.altitude_low_meters==''] = df.altitude_high_meters[df.
↪altitude_low_meters=='']
df['altitude_mean_meters'] = (df.altitude_low_meters.astype(int) + df.
↪altitude_high_meters.astype(int))/2

arabica_df = pd.concat([arabica_df, df], axis=1)
```

[468]:
```
# No need to preprocess the Robusta Altitude column, just match the format of␣
↪Arabica
robusta_df.Altitude = robusta_df.Altitude.astype(int)
robusta_df['altitude_low_meters'] = robusta_df['Altitude']
robusta_df['altitude_high_meters'] = robusta_df['Altitude']
robusta_df['altitude_mean_meters'] = robusta_df['Altitude']
```

Remove extra columns in all datasets

[493]:
```
# drop columns not present in previously cleaned data
drop_cols = ['Status', 'Overall', 'Defects', 'Certification.Address',
             'Certification.Contact']
arabica_df.drop(drop_cols, axis=1, inplace=True)
robusta_df.drop(drop_cols, axis=1, inplace=True)

# drop columns not present in new scrapped data
drop_cols2 = ['Owner.1', 'Cupper.Points', 'unit_of_measurement', 'Certification.
 ↪Address',
              'Certification.Contact']
cleaned_arabica_df.drop(drop_cols2, axis=1, inplace=True)
cleaned_robusta_df.drop(drop_cols2, axis=1, inplace=True)
```

To be able to do the final join of all the datasets, it is necessary to rename the columns of the previously cleaned robusta data. The following columns will be renamed to match with the current convention for robusta and arabica.

- Fragrance…Aroma -> Aroma

- Salt…Acid -> Acidity
- Bitter…Sweet -> Sweetness
- Mouthfeel -> Body
- Uniform.Cup -> Uniformity

[530]:
```
# rename columns in old robusta to match format of all datasets
robusta_dict = {'Fragrance...Aroma':'Aroma',
                'Salt...Acid':'Acidity', 'Mouthfeel':'Body',
                'Bitter...Sweet':'Sweetness',
                'Uniform.Cup':'Uniformity'}

cleaned_robusta_df.rename(columns=robusta_dict, inplace=True)
```

0.0.4 Join all dataframes

[536]:
```
# concatenate all dataframes by default outter, but all columns match with each␣
 ↪other
full_df = pd.concat([cleaned_arabica_df, arabica_df,
           cleaned_robusta_df, robusta_df],
         ignore_index=True, sort=False)
```

[575]:
```
# Convert Moisture to float
for i,val in enumerate(full_df.Moisture):
    try:
        float(val)
    except:
        full_df.loc[i,'Moisture'] = float(val.replace('%','').strip())/100
full_df.Moisture = full_df.Moisture.astype(float)
```

[572]:
```
# Conver Category.One.Defects to int
for i,val in enumerate(full_df['Category.One.Defects']):
    try:
        int(val)
    except:
        full_df.loc[i,'Category.One.Defects'] = int(val.replace('full␣
↪defects','').strip())
```

Checking numeric variables

[577]:
```
full_df.describe()
```

[577]:
```
        Unnamed: 0  Number.of.Bags        Aroma       Flavor   Aftertaste  \
count  1477.000000     1477.000000  1477.000000  1477.000000  1477.000000
mean    704.918754      161.854435     7.574821     7.531774     7.407928
std     425.286918      135.952825     0.372236     0.393496     0.397649
min       1.000000        0.000000     0.000000     0.000000     0.000000
25%     335.000000       16.000000     7.420000     7.330000     7.250000
50%     704.000000      200.000000     7.580000     7.580000     7.420000
75%    1073.000000      275.000000     7.750000     7.750000     7.670000
max    1443.000000     1280.000000     8.750000     8.830000     8.670000

          Acidity         Body      Balance   Uniformity    Clean.Cup  \
count  1477.000000  1477.000000  1477.000000  1477.000000  1477.000000
mean      7.545166     7.527393     7.526242     9.841273     9.843290
std       0.374695     0.363753     0.400143     0.549402     0.740145
min       0.000000     0.000000     0.000000     0.000000     0.000000
25%       7.330000     7.330000     7.330000    10.000000    10.000000
50%       7.580000     7.500000     7.500000    10.000000    10.000000
75%       7.750000     7.750000     7.750000    10.000000    10.000000
max       8.750000     8.580000     8.750000    10.000000    10.000000

         Sweetness  Total.Cup.Points     Moisture  Category.One.Defects  \
count  1477.000000       1477.000000  1477.000000           1477.000000
mean      9.856994         82.161381     0.090150              0.444144
std       0.612304          3.447233     0.046726              2.433762
min       0.000000          0.000000     0.000000              0.000000
25%      10.000000         81.170000     0.100000              0.000000
50%      10.000000         82.580000     0.110000              0.000000
75%      10.000000         83.670000     0.120000              0.000000
max      10.000000         90.580000     0.280000             63.000000

           Quakers  altitude_mean_meters
count  1476.000000           1247.000000
```

```
mean      0.266938        1732.473837
std       1.076978        8176.221765
min       0.000000           1.000000
25%       0.000000        1100.000000
50%       0.000000        1325.000000
75%       0.000000        1600.000000
max      18.000000      190164.000000
```

Above we observe that `altitude_mean_meters` has an unrealistic max value which can greatly affect distance based clustering methods. For this reason, the model will focus on records below 5000, assuming that this is an error in the scale or imputation.

[588]:
```
full_df = full_df[full_df['altitude_mean_meters']<5000].copy()
```

[1009]:
```
full_df.to_csv('arabica_robusta_cleaned.csv', index=False)
```

[ ]:
```

```