AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE
WYDZIAŁ ZARZĄDZANIA

Praca licencjacka

Screen scraping i jego zastosowania na przykładzie Coffee Research Institute oraz Allegro.pl

Autor: *Krzysztof Wis*
Promotor: *Iwona Skalna*
Kierunek studiów: *Informatyka i ekonometria*

Kraków, 2020

Spis treści

Terminy

Screen scraping - technika, za pomocą której program komputerowy wydobywa dane z wyjścia innego programu. Służący do tego program to tak zwany screen scraper. Głównym elementem odróżniającym screen scraping od parsingu jest to, że wyjście programu poddawane scrapingowi przeznaczone jest dla człowieka, a nie do interpretacji przez maszynę.

Element - Wszystko co widoczne (a także niewidoczne) na stronie internetowej to element. Każde pole, obraz, link i tekst to elementy. Element to ekwiwalent etykiety w HTML. W przypadku programów napisanych w ramach tej pracy zapisany w zmiennej element zawiera informacje takie jak jego dzieci, rodzica, lokalizację, wymiary, ewentualny tekst oraz etykietę.

Wstęp

Wprowadzenie i uzasadnienie aktualności tematu

Zazwyczaj transfer danych między programami odbywa się za pośrednictwem struktur danych przystosowanych do maszyn, a nie ludzi. Struktury takie są najczęściej dobrze uporządkowane, łatwe do odczytu i ograniczają dwuznaczność oraz duplikację do minimum. Bardzo często są one nieczytelne dla człowieka. Wyjście przeznaczone dla człowieka jest przeciwieństwem powyższego, formatowanie, nadmiarowe etykiety, komentarze, oraz inne informacje są nie tylko zbędne maszynie, ale mogą także utrudnić interpretację danych. Jednakże jeśli wyjście jest dostępne jedynie w takim przyjaznym człowiekowi formacie, screen scraping staje się jedynym zautomatyzowanym sposobem przeprowadzenia transferu danych.

Pierwotnie termin ten odnosił się do czytania danych z pamięci ekranu terminala komputerowego. Analogicznie screen scraping oznacza także skomputeryzowane przetwarzanie HTML-a na stronach WWW. W każdym przypadku screen scraper musi być zaprogramowany nie tylko do przetwarzania interesujących danych, lecz także do odrzucania niechcianych informacji i formatowania.

Screen scraping jest uznawany przez niektórych za nieelegancką technikę, używaną tylko jako ostateczność, kiedy żaden inny mechanizm nie jest dostępny. Oprócz tego, że w programowanie trzeba włożyć większy wysiłek, dane przeznaczone dla człowieka często zmieniają strukturę. Ludzie radzą sobie z tym bez problemu, programy komputerowe jednak w takich przypadkach przestają działać lub (co gorsze) zwracają niepoprawne wyniki. Dlatego też stworzenie narzędzia, które efektywnie pobiera ze złożonej strony interesujące nas dane może być ciekawym i przydatnym przedsięwzięciem.

Model ekonometryczny za pomocą równania lub układu równań pomaga wyjaśnić mechanizm zmian zachodzących w badanym obszarze. Opisuje powiązania między danymi wielkościami ekonomicznymi. Jest to formalny matematyczny zapis istniejących prawidłowości ekonomicznych. Zbudowanie modelu ekonometrycznego wymaga nie tylko dobrej znajomości teorii ekonomii oraz wiedzy matematyczno- ekonomicznej, lecz również znajomości praktyki ekonomicznej. Model ekonometryczny powinien posiadać nie tylko wartość poznawczą z punktu widzenia teorii ekonomii, ale również wartość praktyczną, czyli aby mógł służyć jako narzędzie wnioskowania w przyszłości. Takie modele są przydatne dla przedsiębiorców oraz ludzi operujących na rynkach finansowych.

Cel pracy

Celem pracy jest zaprezentowanie działania metod scrapingu na paru przykładach paru stron, a następnie zaprezentowanie użyteczności uzyskanych danych poprzez budowę modelu ekonometrycznego na podstawie danych z Cofee Quality Institute, który pozwoli wyciągać wnioski na temat optymalnych miejsc i metod hodowli ziaren kawy.

Kolejnym celem jest analiza struktury strony z wynikami wyszukiwania ze strony www.allegro.pl, ściągnięcie danych na temat produktów, przesłanie ich do bazy danych oraz ich analiza.

Tezy oraz hipotezy naukowe

Celem części pracy związanej z bazą danych kaw jest budowa narzędzia do Screen Scrapingu a następnie oczyszczenie tak uzyskanych danych i budowa modelu ekonometrycznego. Interesujące jest, czy otrzymane wyniki będą zgodne z logiką i z wiedzą z dziedziny hodowli kawy – czy nieotwarte ziarna w próbce wpływają na jakość negatywnie, czy wraz z wysokością n.p.m ona się polepsza, czy zielony faktycznie jest najlepszym kolorem próbki.

Celem części pracy związanej ze sklepem internetowym Allegro jest stworzenie narzędzia do pozyskiwania danych i zautomatyzowanie ich zapisu do bazy danych oraz ich analiza – na przykład sprawdzenie czy cena dostawy maleje przy wzroście wartości produktu, oraz jak częste są produkty z różnych półek cenowych.

Aspekt legalny

To, czy w świetle prawa Data Scrapping jest legalne, wciąż jest niejasne, pomimo wielu osądów w przeciągu ostatnich dwudziestu lat. Jeśli pozyskane dane są używane dla osobistego i prywatnego użytku, zwykle nie ma żadnego problemu. Jednakże jeśli dane te miałyby być opublikowane albo gdy scrapping jest agresywny i może spowodować awarię strony, albo zawartość jest chroniona prawem autorskim, może istnieć precedens do podjęcia akcji przeciwko użytkownikowi pozyskującemu dane.

W *Feist Publications, Inc. V. Rural Telephone*, rząd Stanów Zjednoczonych zadecydował, że scrapping i publikowanie numerów telefonów są legalne. Podobny przypadek w Australii, Telstra Corporation *Limited v. Phone Directories Company Pty Ltd,* zademonstrował że jedynie dane z identyfikowalnym autorem mogą być chronione prawem autorskim. Podobny przypadek w kwestii zdrapywanych danych, związany z ponownym użyciem opowiadań artykułów w ich zbiorze, został uznany za naruszenie prawa autorskiego – *Associated Press v. Meltwater.* W Danii, w sprawie *ofir.dk vs home.dk* osiągnięta została konkluzja, że zwykły crawling i deep linking są dozwolone.

Było też wiele przypadków kiedy różne firmy pozywały o agresywny scraping i próbowały go powstrzymać za pomocą prawa. Na jednej z takich rozpraw, *QVC v. Resultly,* orzeczono że o ile scrapping nie skutkował zniszczeniem prywatnej własności, nie może zostać uznany za celowe czynienie szkody, pomimo możliwości przyczynienia się do niestabilności strony.

Te przypadki pozwalają uważać, że kiedy pozyskane dane są publiczną informacją, (taką jak lokalizacje biznesowe i spisy numerów telefonów) to mogą być pozyskane legalnie. Jeśli jednak zdrapane dane są oryginalne (takie jak opinie, recenzje i prywatne dane), to prawdopodobnie zdrapywanie ich jest nielegalne. W każdym razie, należy pamiętać, że na stronie jest się gościem i istnieje możliwość zbanowania naszego numeru IP. Należy oczywiście uważać także na regulaminy.

Dane z Cofee Research Institute są dostępne publicznie i celowo udostępniane przez Instytut, więc w przypadku tej pracyprzypadku problemów prawnych być nie powinno. Tak samo nic nie wskazuje że dane z Allegro miałyby w jakiś sposób być niedozwolone do pozyskiwania.

Istnieje jeszcze kwestia Robot Exclusion Standard, znanego także jako plik *robots.txt*. Jest to znajdująca się w folderze głównym strony informacja, co robot indeksujący, taki jak Google, może indeksować. O ile ten plik na stronie Coffee Institute jest zasadniczo pusty, o tyle na stronie Allegro można znaleźć komendę disallow, nie pozwalającą na indeksowanie następujących części strony:

disallow: /showcase.php/
disallow: /SendMailToUser.php*
disallow: /report_item.php*
disallow: /events/clicks
disallow: /offer/
disallow: /logowanie*
disallow: /ape
disallow: /payments
disallow: /recommendations/
disallow: /cart-aggregator/

disallow: /favourites/
disallow: /usertraits/
disallow: /mediation/
disallow: *snapshot=*

Żaden z tych elementów nie nawiązuje do zdrapywanych w tej pracy części strony, szczególnie biorąc pod uwagę że zarówno strony z wynikami wyszukiwań jak i same strony ofert można znaleźć w wyszukiwarce Google.

Jak wyglądałaby jednak sytuacja jakby plik robots.txt zakazywał indeksowania danych pozyskiwanych w tej pracy? Otóż mimo, że plik ten wyraźnie wyraża wolę właściciela strony, aby nie można było zapisywać jego danych, i chroni go przed nadmiernym ruchem na stronie internetowej, nie jest w żaden sposób wiążący. Przykładowo, w rozprawie z roku 2003,w której Healthcare Advocates pozwało Health Advocate Inc o dostęp do danych „chronionych” plikiem robots.txt, sąd USA uznał, że prawo nie zostało naruszone.

Rozdział 1- Używane narzędzia

Język programowania Python

Python jest przejrzystym językiem wysokiego poziomu z ogromną i stale rosnącą liczbą bibliotek oraz jedną z największych społeczności. Jego największą zaletą jest wszechstronność zastosowań. Od niedawna coraz powszechniej przejmuje on rolę języków C++ i C w dziedzinie programowania sterowników sprzętowych. Python pozostaje również ulubionym narzędziem w sektorze data science, zastępując język R. Według badania przeprowadzonego w 2018 roku przez Analytics India Magazine, technologia ta używana jest przez 44% specjalistów zajmujących się data science.

Do pisania tej pracy został on przeze mnie wybrany ze względu na moją jego znajomość oraz jego szeroki potencjał do rozwiązywania różnych problemów poprzez szeroki wybór dostępnych bibliotek.

IDE PyCharm

PyCharm – zintegrowane środowisko programistyczne dla języka programowania Python firmy JetBrains. Wybrałem je ze względu na łatwą instalację bibliotek i łatwość użytku.

Jest oprogramowaniem wieloplatformowym pracującym na platformach systemowych: Microsoft Windows, GNU/Linux oraz macOS. Wydawany jest w wersji Professional Edition, która jest oprogramowaniem własnościowym oraz w wolnej wersji Community Edition, która pozbawiona jest jednak części funkcjonalności w porównaniu z wersją własnościową. Jest to jednak wybrana przeze mnie wersja ze względu na to, że jest darmowa.

gretl

Pakiet statystyczny typu open source wykorzystywany głównie w ekonometrii. Nazwa jest skrótem od Gnu Regression, Econometrics and Time-series Library. Zostanie on użyty przy budowie modelu ekonometrycznego. Potrafi budować modele, działać na plikach *.csv i pokazywać wykresy. Wybrałem go ze względu na jego możliwości i prostotę działania.

Selenium WebDriver

Selenium WebDriver to następca Selenium Remote Control. Przyjmuje on komendy (w „Selenese”, specjalnym języku programowania) i wysyła je do przeglądarki. Dzieje się to dzięki specjalnemu sterownikowi, który wysyła komendy do przeglądarki i otrzymuje wyniki z powrotem. Jego wersja istnieje dla wszystkich większych przeglądarek (Firefox, Chrome, Internet Explorer, Safari, Microsoft Edge).

Pakiet Pandas

Pandas to biblioteka napisana dla języka Python. Służy ona do manipulacji i analizy danych. Oferuje ona obsługę struktur z danymi. Jego nazwa pochodzi od „Panel Data”. W tym projekcie

jest ona używana głównie do konwersji skomplikowanych struktur danych pozyskiwanych ze stron na tabele z danymi, które są łatwo używalne.

BeautifulSoup

Biblioteka Beautiful Soup służy do wyciągania danych z plików XML i HTML. Współpracuje z parserami, zapewniając idiomatyczne sposoby nawigacji, wyszukiwania i modyfikowania drzewa parsowania.

MongoDB Compass

GUI dla MongoDB.

MongoDB to otwarty, nierelacyjny system zarządzania bazą danych napisany w języku C++. Charakteryzuje się dużą skalowalnością, wydajnością oraz brakiem ściśle zdefiniowanej struktury obsługiwanych baz danych. Zamiast tego dane składowane są jako dokumenty w stylu JSON, co umożliwia aplikacjom bardziej naturalne ich przetwarzanie, przy zachowaniu możliwości tworzenia hierarchii oraz indeksowania.

Inne narzędzia

- Funkcja „Inspect” to narzędzie w przeglądarce, które pokazuje kod HTML i CSS danego elementu. Jest podstawowym narzędziem, które pozwala zorientować się w zawartości strony.

Poniższa funkcja pomocnicza (Rysunek 1) służy do podświetlania elementu na stronie. Jest ona niezwykle użyteczna do lokalizowania konkretnego elementu. W końcowym kodzie nie będzie już używana, ale w trakcie pracy nad stronami była niezbędna. Poniżej znajduje się przykład podświetlenia jednego z elementów article (Rysunek 2)

```
def highlight(element, effect_time, color, border):
    driver = element._parent
    def apply_style(s):
        driver.execute_script("arguments[0].setAttribute('style', arguments[1]);",
                              element, s)
    original_style = element.get_attribute('style')
    apply_style("border: {0}px solid {1};".format(border, color))
    time.sleep(effect_time)
    apply_style(original_style)
```

Rysunek 1. Funkcja podświetlająca elementy na stronie

Rysunek 2. Efekt zastosowania powyższej funkcji

Funkcje lokalizujące elementy na stronie:

- .find_element_by_name
- .find_element_by_link_text
- .find_element_by_partial_link_text
- .find_element_by_xpath
- .find_element_by_tag_name
- .find_element_by_class_name

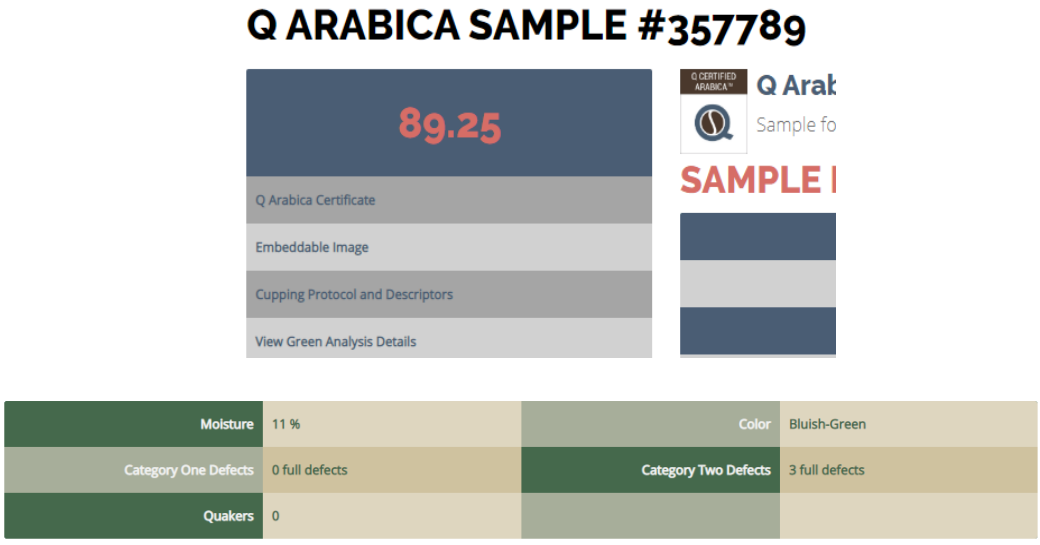
Rozdział 2- Data scraping ze strony Coffee Quality Institute i budowa podstawowego modelu ekonometrycznego.

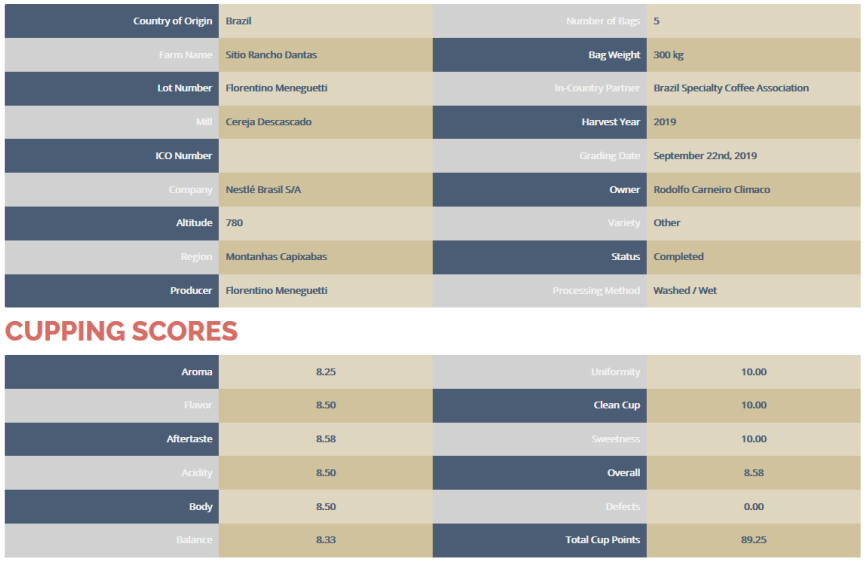
Dziedzina problemu

Ze stroną https://database.coffeeinstitute.org/ jest kilka problemów. Po pierwsze, dostęp do bazy wymaga zalogowania. Na szczęście konto jest darmowe, więc bardzo prosto było założyć konto i napisać automatyczne logowanie. Po drugie, autorzy strony nie stworzyli ani pliku robot.txt, ani pliku Sitemap, co znacząco utrudni orientowanie się w stronie. Po trzecie mechanizmy strony są zbudowane na podstawie skryptów, co uniemożliwia uzyskanie danych bezpośrednio ze źródła strony.

Dane do pozyskania

Z danych , w kontekście budowy modelu, interesujące są oczywiście tylko niektóre. Są to rok zbioru, wysokość nad poziomem morza, rodzaj (variety), metoda procesowania, wilgotność, kolor, defekty pierwszej i drugiej kategorii, nieotwarte nasiona (quakers), oraz sam wynik punktowy. Wynik jest po prostu sumą Cupping Scores. Te punktacje też zostaną pobrane, aby później możliwe było zbudowanie macierzy ich wzajemnych korelacji. Jak te dane wyglądają na stronie, przedstawia (Rysunek 3)





Country of Origin	Brazil	Number of Bags	5
Farm Name	Sitio Rancho Dantas	Bag Weight	300 kg
Lot Number	Florentino Meneguetti	In-Country Partner	Brazil Specialty Coffee Association
Mill	Cereja Descascado	Harvest Year	2019
ICO Number		Grading Date	September 22nd, 2019
Company	Nestlé Brasil S/A	Owner	Rodolfo Carneiro Climaco
Altitude	780	Variety	Other
Region	Montanhas Capixabas	Status	Completed
Producer	Florentino Meneguetti	Processing Method	Washed / Wet

Aroma	8.25	Uniformity	10.00
Flavor	8.50	Clean Cup	10.00
Aftertaste	8.58	Sweetness	10.00
Acidity	8.50	Overall	8.58
Body	8.50	Defects	0.00
Balance	8.33	Total Cup Points	89.25

Rysunek 3. Dane do pozyskania ze strony

Proces pozyskiwania danych ze strony internetowej.

```
driver = webdriver.Firefox()
waitForResponse()
driver.get('https://database.coffeeinstitute.org/login')
```

Rysunek 4. Uruchomienie przeglądarki i otwarcie strony Instytutu

Po pierwsze należało pobrać webdriver – wersję przeglądarki kompatybilną z pakietem Selenium – oraz umieścić go w PATH. Spośród dostępnych przeglądarek został wybrany Firefox, ze względu na jej szybkie działanie i fakt, że Chrome ma problemy z obsługą przesyłanych do niego poleceń kliknięcia. Driver będzie zmienną oznaczającą po prostu otwartą przeglądarkę.

WaitForResponse to zwyczajna komenda time.sleep(...). Dzięki temu że jest w osobnej funkcji mogę zmieniać czas oczekiwania – przydatne przy powolniejszych komputerach i połączeniach internetowych. I wygląda to bardziej estetycznie. Funkcja z oczywistych powodów jest używana w programie często.

driver.get to funkcja otwierająca podany adres.

```
username = driver.find_element_by_name("username")
password = driver.find_element_by_name("password")

username.send_keys(login_email)
password.send_keys(login_password)
driver.find_element_by_class_name("submit").click()
waitForResponse()
```

Rysunek 5. Logowanie na stronę

Po pierwsze, na stronie trzeba było zlokalizować pola „username” i „password”, używając jednej z metod wyszukiwania elementów. Na szczęście te akurat pola, mają standardowe i celowe nazwy. Nie zawsze tak jest.

Następnie trzeba było wpisać do tych pól login i hasło (Są to stringi, nie zamieszczona została ich deklaracja, ponieważ są to prywatne dane). Na końcu należało zlokalizować obiekt o jasno nazwanej klasie „submit” i kliknąć go.

```
coffees =
driver.find_element_by_xpath('//html/body/header/nav[@id="main"]/div[@class="containe
r"]/div[@class="in"]/a[@href="/coffees"]').click()
waitForResponse()
```

Rysunek 6. Nawigacja do panelu z wyborem typu kaw

Została użyta funkcja *find_element_by_xpath* aby praktycznie bezpośrednio wskazać konkretny element na stronie. Konstrukcja */X[@Y=”Z”]* oznacza że poszukiwana jest etykietka X o atrybucie Y równym Z. Technicznie możliwe jest, że funkcja znalazłaby więcej niż jeden element, ale kryteria są na to zbyt konkretne w tym przypadku.

```
driver.maximize_window()
driver.find_element_by_link_text('Arabica Coffees').click()
waitForResponse()
```

Rysunek 7. Nawigacja do listy kaw Arabica

Link do bazy danych z Kawami Arabica jest akurat przykryty przez napis „Login Successful”, więc prostym sposobem na ominięcie tego problemu jest zmaksymalizowanie strony. Element można znaleźć po tekście linku.

```
file1 = open("data.csv", "a")
file1.write("Id,Score,Country,Altitude,Variety,Processing,Arome,Flavor,Aftertaste,Aci
dity,Body,Balance,Uniformity,Clean Cup,Sweetness,Cupper
Points,Defects,Moisture,Defects 1,Quakers,Color,Defects 2\n")
```

Rysunek 8. Przygotowanie pliku na wyniki

W tym zadaniu do zbierania danych będzie używany plik *.csv. Zwróciwszy uwagę na formatowanie plików, którego wymaga potem program gretl, można stwierdzić że jest to najprostsze i najefektywniejsze rozwiązanie.

```
while True:
   maintable = driver.find_elements_by_xpath('//td')
```

```
for i in range(1,350,8):
      (zasadnicza funkcjonalność, opisana dalej)
try:
      linkNext = driver.find_element_by_link_text('Next')
      linkNext.click()
except:
   break
```

Rysunek 9. Sposób iterowania po stronie

W powyższy sposób dokonana zostaje iteracja po każdej kawie w głównej tabeli. Jak widać na poniższym zrzucie ekranu, tabela posiada osiem kolumn, przez co iterując po elementach w głównej tabeli, zmienna *i* zwiększa się o osiem przy każdej iteracji. Iterowanie po wszystkich stronach jest bardzo proste. Jeśli na stronie znajduje się link do kolejnej strony, zostaje on kliknięty. Jeśli nie, iterowanie zostaje zakończone.

	ID	Species	Country	Owner	Grade	ICP	Completed
	#357789	Arabica	Brazil	Rodolfo Carneiro Climaco	89.25	bsca	Sep 22nd, 2019
	#961547	Arabica	Ethiopia	Blossom Valley International	87.17	JPCE	Jul 14th, 2020
	#274278	Arabica	Brazil	Rodolfo Carneiro Climaco	86.75	bsca	Oct 2nd, 2019
	#507718	Arabica	Ethiopia	Blossom Valley International	86.58	JPCE	Jul 14th, 2020
	#308592	Arabica	Brazil	Rodolfo Carneiro Climaco	86.58	bsca	Sep 22nd, 2019
	#282143	Arabica	Brazil	Rodolfo Carneiro Climaco	86.33	bsca	Sep 22nd, 2019
	#758571	Arabica	Ethiopia	Al-impex Business plc	86.25	metad	Jun 29th, 2020
	#139012	Arabica	Brazil	Rodolfo Carneiro Climaco	85.75	bsca	Sep 22nd, 2019
	#508424	Arabica	Brazil	Rodolfo Carneiro Climaco	85.75	bsca	Sep 22nd, 2019
	#529406	Arabica	Ethiopia	Al-impex Business plc	85.67	metad	Jun 13th, 2020
	#619793	Arabica	Brazil	Rodolfo Carneiro Climaco	85.50	bsca	Sep 22nd, 2019

Rysunek 10. Tabela z kawami na stronie Instytutu

```
try:
        IDnumber = maintable[i].text
except:
        break

actions.key_down(Keys.CONTROL).click(maintable[i]).perform()
#actions.key_up(Keys.CONTROL).perform()

#Otwieranie strony konkretnej próbki w nowej karcie

link = maintable[i].find_element_by_tag_name("a").get_attribute('href')
driver.execute_script("window.open('');")
driver.switch_to.window(driver.window_handles[1])
driver.get(link)
```

Rysunek 11. Otwieranie strony próbki kawy w nowej karcie

Jeśli na stronie nie ma już elementu maintable[i], to znaczy że strona nie jest cała wypełniona próbkami i program dotarł do ostatniej. Należy w tym momencie zakończyć wykonywanie programu.

Wykomentowany tekst to domyślny sposób na zwyczajne otwarcie linku w nowej karcie: CTRL i wciśnięcie przycisku na myszce, po czym zwolnienie go. Po paru godzinach prób zidentyfikowania problemu zdecydowałem się na obejście problemu poprzez skopiowanie linku, otwarcie nowej karty i otworzenie w niej linku.

„window.open(‘‘);” to tak naprawdę krótki skrypt w JavaScript.

```
tables = driver.find_elements(By.TAG_NAME, "table")
waitForResponse()

arr = []
arr.append(IDnumber)
t = BeautifulSoup(tables[0].get_attribute('outerHTML'), "html.parser")
df = pd.read_html(str(t))
row = df[0].columns[0]
arr.append(row[0]) #points
```

Rysunek 12. Pozyskiwanie ilości punktów ze strony

Arr to tabela, zawierająca kolejne dane które będą wpisane do pliku i używane w modelu.

Powyższy kod pokazuje jak z pierwszej tabeli, pierwszej kolumny i pierwszego w niej elementu wyciągnąć faktyczny tekst. Używana jest w tym momencie biblioteka BeautifulSoup. Dzięki niej możliwe jest sparsowanie kodu HTML do bardziej używalnej formy.

```
t = BeautifulSoup(tables[1].get_attribute('outerHTML'), "html.parser")
df = pd.read_html(str(t))
frame = df[0]

arr.append(frame[1][0])
arr.append(frame[1][6])
arr.append(frame[3][6])
arr.append(frame[3][8])

t = BeautifulSoup(tables[2].get_attribute('outerHTML'), "html.parser")
df = pd.read_html(str(t))
frame = df[0]

arr.append(str(frame[1][0]))
arr.append(str(frame[1][1]))
arr.append(str(frame[1][2]))
arr.append(str(frame[1][3]))
arr.append(str(frame[1][4]))
arr.append(str(frame[1][5]))
arr.append(str(frame[3][0]))
```

```
arr.append(str(frame[3][1]))
arr.append(str(frame[3][2]))
arr.append(str(frame[3][3]))
arr.append(str(frame[3][4]))

t = BeautifulSoup(tables[3].get_attribute('outerHTML'), "html.parser")
df = pd.read_html(str(t))
frame = df[0]

arr.append(frame[1][0])
arr.append(frame[1][1])
arr.append(frame[1][2])
arr.append(frame[3][0])
arr.append(frame[3][1])
```

Rysunek 13. Wyciąganie danych z tabeli na stronie

W powyższym fragmencie programu odbywa się iteracja po każdej z tabel z (Rysunku 3). Potrzebne dane zostają z nich zczytywane i umieszczane w tabeli arr.

```
savstr = ""
for x in arr:
   savstr = savstr + "\"" +str(x) + "\","
savstr = savstr[:-1]

file1.write(savstr+"\n")

driver.close()
driver.switch_to.window(driver.window_handles[0])
```

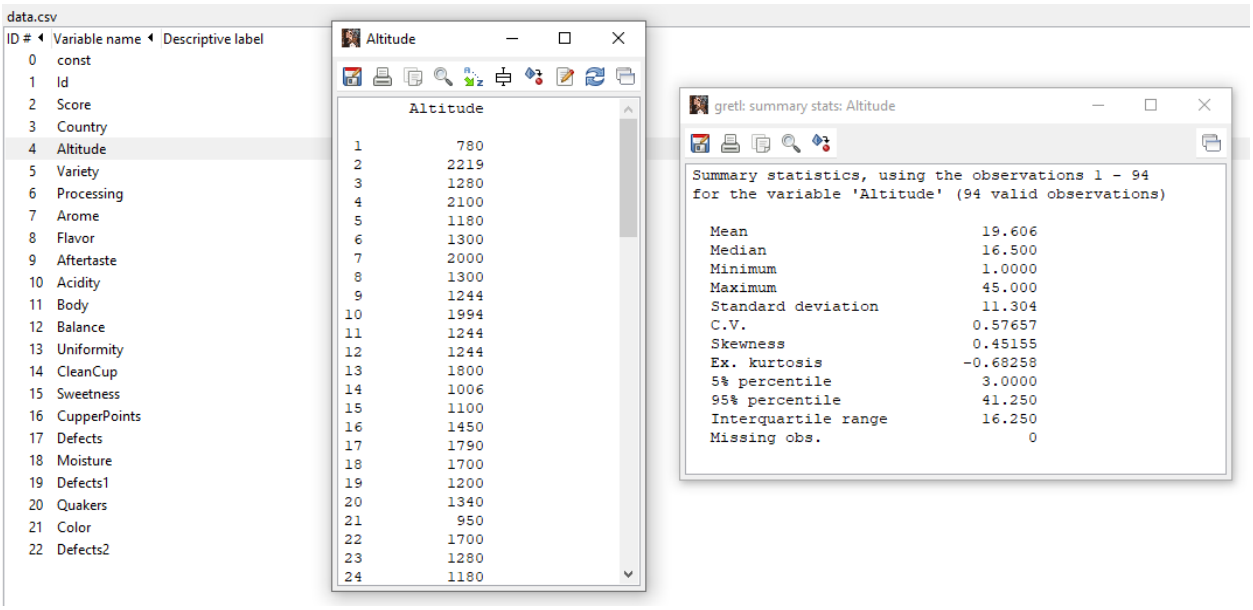
Rysunek 14. Zapis danych do pliku

Następnie po kolei, po każdym elemencie, dane z tabeli *arr* zapisywane są do Stringa savstr, zwracając uwagę na znaki cudzysłowia i przecinkach między nimi. Potem następuje jego zapis do pliku, w nowej linii. Na koniec następuje zamknięcie karty i powrót do punktu wyjścia. Jak te dane wyglądają otwarte w Notatniku, widać na (Rysunku 15)

```
Id,Score,Country,Altitude,Variety,Processing,Arome,Flavor,Aftertaste,Acidity,Body,Balance,Uniformity,Clean Cup,Sweetness,Cupper Points,Defects,Moisture,Defects 1,Quakers,Color,Defects 2
"#357789","89.25","Brazil","780","Other","Washed / Wet","8.25","8.5","8.58","8.5","8.5","8.33","10.0","10.0","10.0","8.58","0.0","10 %","0 full defects","0","Bluish-Green","5 full defects"
"#274278","86.75","Brazil","1280","Other","Washed / Wet","7.92","8.17","8.17","8.25","8.08","8.08","10.0","10.0","10.0","8.08","0.0","12 %","0 full defects","2","nan","0 full defects"
"#308592","86.58","Brazil","1180","Other","Washed / Wet","8.17","7.92","8.0","8.17","8.25","8.08","10.0","10.0","10.0","8.0","0.0","10 %","0 full defects","0","Bluish-Green","2 full defects"
"#282143","86.33","Brazil","1300","Other","Washed / Wet","7.83","8.0","8.0","8.08","8.08","8.17","10.0","10.0","10.0","8.17","0.0","11 %","0 full defects","0","Bluish-Green","1 full defects"
"#641141","85.83","Panama","1400.1700","Typica","Washed / Wet","7.92","8.08","7.75","8.17","8.0","7.83","10.0","10.0","10.0","8.08","0.0","11 %","0 full defects","3","Bluish-Green","0 full defects"
"#139012","85.75","Brazil","1300","Other","Washed / Wet","7.92","8.08","7.83","7.92","7.92","8.0","10.0","10.0","10.0","8.08","0.0","11 %","0 full defects","0","Bluish-Green","0 full defects"
"#508424","85.75","Brazil","1244","Other","Washed / Wet","7.92","8.08","7.92","8.0","7.83","7.92","10.0","10.0","10.0","8.08","0.0","12 %","0 full defects","0","Bluish-Green","1 full defects"
"#529406","85.67","Ethiopia","1994","Ethiopian Yirgacheffe","Washed / Wet","8.0","8.0","7.83","8.0","8.0","8.0","10.0","10.0","10.0","7.83","0.0","10 %","0 full defects","0","Green","7 full defects"
"#619793","85.50","Brazil","1244","Other","Washed / Wet","7.92","7.92","7.83","8.0","7.92","8.0","10.0","10.0","10.0","7.92","0.0","11 %","0 full defects","0","Bluish-Green","0 full defects"
"#200218","85.50","Brazil","1244","Other","Washed / Wet","7.75","8.08","8.0","8.0","7.92","7.83","10.0","10.0","10.0","7.92","0.0","11 %","0 full defects","0","Bluish-Green","5 full defects"
"#429563","85.33","Ethiopia","1850","Ethiopian Heirlooms","Washed / Wet","7.58","7.83","7.92","8.08","8.17","7.83","10.0","10.0","10.0","7.92","0.0","9 %","0 full defects","1","Blue-Green","5 full defects"
"#428993","85.25","Brazil","1006","Other","Natural / Dry","7.83","8.0","8.0","7.75","7.83","7.92","10.0","10.0","10.0","7.92","0.0","11 %","0 full defects","0","Bluish-Green","2 full defects"
"#592575","85.25","Brazil","1100","Bourbon","Pulped natural / honey","7.92","7.83","7.83","7.92","7.92","7.83","10.0","10.0","10.0","8.0","0.0","11 %","0 full defects","0","Bluish-Green","3 full defects"
"#135582","85.17","Honduras","1450","Catuai","Washed / Wet","7.92","7.92","7.75","7.92","7.92","7.83","10.0","10.0","10.0","7.92","0.0","9 %","0 full defects","0","Green","1 full defects"
"#939387","85.08","Uganda","1790","SL28","Washed / Wet","7.83","8.08","7.92","7.67","7.83","7.75","10.0","10.0","10.0","8.0","0.0","10 %","0 full defects","0","Green","0 full defects"
"#835570","85.00","Ethiopia","1700","Ethiopian Heirlooms","Natural / Dry","7.58","8.17","7.75","8.25","7.75","7.5","10.0","10.0","10.0","8.0","0.0","10 %","0 full defects","1","Green","3 full defects"
"#588079","84.92","Mexico","1200","Gesha","Washed / Wet","8.0","7.92","7.58","8.0","7.67","7.67","10.0","10.0","10.0","8.08","0.0","11 %","0 full defects","0","Blue-Green","4 full defects"
```

Rysunek 15. Ostateczny wygląd gotowego pliku z danymi

Wynikiem jest plik *.csv, gotowy do importu do programu statystycznego gretl. Jak wyglądają dane w nim otwarte widać na (Rysunku 16)



Rysunek 16. Dane otwarte w programie gretl

Jakie cechy czynią kawę dobrą? – budowa modelu za pomocą MNK

Po imporcie do programu gretl, gotowy jest dataset do analizy. Jak widać na załączonym zrzucie ekranu, możliwe jest otrzymanie statystyk dotyczących każdej z danych. Niestety, są dostępne jedynie 94 rekordy – strona zwyczajnie przeszła przez jakiś rodzaj restrukturyzacji podczas pisania tej pracy i zmalała ilość dostępnych rekordów – ale i tak powinno być możliwe zbudowanie modelu.

Najpierw trzeba jednak dopasować niektóre zmienne do potrzeb modelu. Defekty „x full defects" zmienić na „x", Zmienną tekstową „color" zmienić na binarne zmienne za pomocą narzędzia dummify („Green" – 1, „Blue" – 0, „Bluish – Green" – 0), podobnie jak zmienną Processing.

Ostateczna punktacja jest sumą osobnych punktów, takich jak Flavor, Balance i Uniformity. Zawieranie ich w modelu byłoby więc oczywiście bezsensowne.

Do stworzenia modelu będą zatem używane następujące zmienne:

Color – Kolor próbki.
- **DColor_1** – Niebieskawo-Zielony
- **DColor_2** – Zielony
- **DColor_3** – Żaden
- **DColor_4** – Niebiesko-Zielony

Processing – Metoda procesowania próbki
- **Dprocessing_1** – Myte / Wilgotne
- **Dprocessing_2** – Inne
- **Dprocessing_3** – Naturalne/Suche
- **Dprocessing_4** – Usuwana skóra

Defects1, **Defects2** – Defekty pierwszego i drugiego typu
Quakers – Kwakry, niedojrzałe nasiona
Moisture – wilgotność, w procentach
Altitude – wysokość nad poziomem morza

Analiza Głównych Składowych

Analiza głównych składowych (ang. Principal Component Analysis, w skrócie PCA) jest procedurą statystyczną, która polega na ortogonalnej trasformacji układu badanych zmiennych X w zbiór nowych nieobserwowanych zmiennych Y , które są w rzeczywistości kobinacją liniową tych obserwowanych zmiennych.

Cel przeprowadzenia PCA:

- redukcja liczby zmiennych opisujących zjawiska (bez utraty informacji)
- wykrywanie struktury w związkach między zmiennymi
- weryfikacja wykrytych prawidłowości i powiązań
- rozpoznawanie jednostek nietypowych
- klasyfikacja obiektów w nowych przestrzeniach zdefiniowanych przez utworzone czynniki (grupowanie)
- graficzna prezentacja konfiguracji porównywanych zmiennych

Metoda często ujawnia zależności, których wcześniej się nie domyślano,a co za tym idzie PCA pozwala na zupełnie nową interperację danych.

Główne składowe są tak wyznaczane, aby wariancje kolejnych składowych były coraz mniejsze. Głównych składowych można wyznaczyć tyle, ile było zmiennych pierwotnych. Jednak zazwyczaj kilka pierwszych wystarcza do wyjaśnienia większości wariancji układu zmiennych.

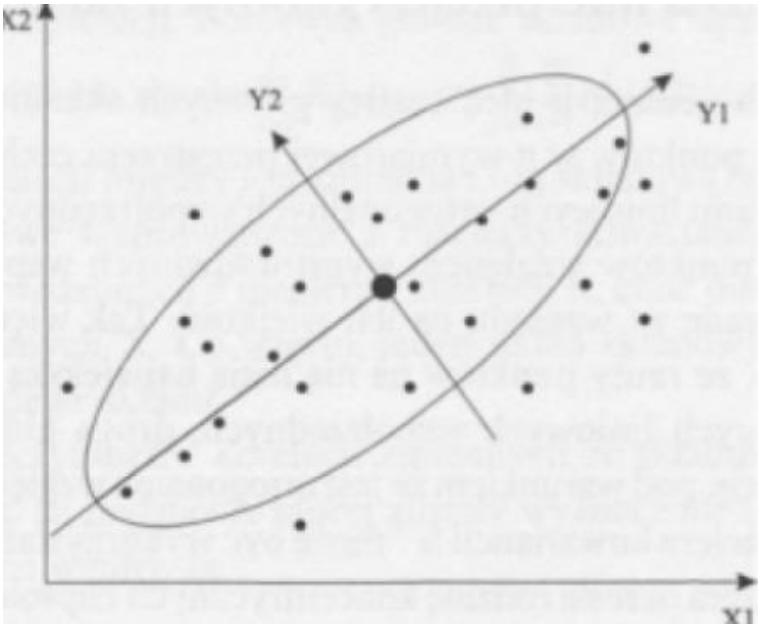
Znaczenie i użyteczność głównej składowej jest mierzona wielkością wyjaśnianej przez nią całkowitej wielkości.

Własności składowych głównych:

- są liniową kombinacją obserwowanych zmiennych
- są ortogonalne względem siebie
- kolejne składowe wyjaśniają malejącą ilość łącznej wariancji zmiennych
- suma wariancji składowych jest równa sumie wariancji zmiennych pierwotnych

Z geometrycznego punktu widzenia ideą analizy głównych składowych jest opisanie zmienności układu **n** punktów w **p** wymiarowej przestrzeni cech poprzez wprowadzenie nowego układu liniowych, ortogonalnych współrzędnych. Wariancje danych punktów względem wprowadzonych współrzędnych są uporządkowane malejąco. Rzuty punktów na pierwszą składową mają największą wariancję ze wszystkich możliwych liniowych współrzędnych.

Rozważmy dwie zmienne X1, X2 oraz n pomiarów (x_i1, x_i2), (i = 1, 2, . . . , n). Pomiary są przedstawione na układzie współrzędnych na płaszczyźnie w formie diagramu korelacyjnego. Kierunek zgodnie z którym dane są bardziej rozproszone wyznacza nową oś, która reprezentuje pierwszą główną składową Y1. Druga oś biegnąca pod kątem 90 stopni do pierwszej, wyznacza kierunek drugiej składowej Y2. Obie osie współrzędnych X1, X2 są transformowane poprzez przesunięcie środka układu do punktu średnich ($\bar{x}_1, \bar{x}_2$), a następnie obrócone w taki sposób, że otrzymujemy współrzędne Y1, Y2 głównych składowych.



Rysunek 17. Graficzna reprezentacja istoty wyodrębnienia głównych składowych – diagram korelacyjny

Punktem wyjścia PCA są macierz korelacji bądź macierz kowariancji utworzone ze zbioru wyjściowego. Zawierają one całą informację niezbędną do wyznaczenia głónych składowych. Algorytm w obydwu wersjach jest identyczny, jednak uzyskane wyniki są zupełnie różne.

W przypadku użycia macierzy kowariancji (Σ) największy wpływ na wynik mają zmienne o największej wariancji. Stąd Σ możemy użyć, gdy analizujemy zbiór zmiennych o porównywalnych wielkościach (np. procentowe zmiany kursów akcji).

W przeciwnym przypadku decydujemy się na macierz korelacji (ρ). Użycie macierzy korelacji odpowiada wstępnej normalizacji zbioru pierwotnego, tak aby każda zmienna miała na wejściu identyczną wariancję.

W naszym modelu będziemy używać macierzy korelacji. Różnorodność danych wejściowych jest bardzo duża – mamy dane binarne, takie jak Kolor Zielony, dane procentowe, takie jak Wilgotność, oraz dane ilościowe, takie jak Defekty.

Używanie programu gretl do PCA

Szukanie składowych głównych jest procesem złożonym, przez co wyjaśnienie go i zaprezentowanie zajęłoby bardzo dużą część tej pracy. Na szczęście program gretl posiada narzędzie pozwalające automatycznie znaleźć składowe główne oraz policzyć ich Wartości Własne, mówiące jak dużą część wariancji w modelu wyjaśnia dana składowa – czyli jak dobrze podsumowywuje ona zmienne w modelu.

Dla wspomnianych wcześniej zmiennych, program był w stanie policzyć 13 Wartości Własnych – ilość równa liczbie zmiennych. (Color i Processing zostały przeze mnie podzielone na zmienne binarne) Kolumna „proportion" oznacza jaką część wariancji w modelu jest wyjaśniane przez daną Wartość. Wartości w niej sumują się oczywiście do 1. Wynik widać na (Rysunku 18). Eigenvalue = Wartość Własna, Component = Składowa Główna.

Eigenanalysis of the Correlation Matrix

Component	Eigenvalue	Proportion	Cumulative
1	2.4089	0.1853	0.1853
2	1.9063	0.1466	0.3319
3	1.4346	0.1104	0.4423
4	1.2324	0.0948	0.5371
5	1.2102	0.0931	0.6302
6	1.1319	0.0871	0.7173
7	1.0285	0.0791	0.7964
8	0.9494	0.0730	0.8694
9	0.6573	0.0506	0.9200
10	0.5723	0.0440	0.9640
11	0.4683	0.0360	1.0000
12	0.0000	0.0000	1.0000
13	0.0000	0.0000	1.0000

Rysunek 18. Wartości własne Składowych Głównych

Niestety, jak widać na powyższym rysunku, żadna ze składowych nie wyjaśnia wyraźnej większości wariancji w modelu. Całym celem tej analizy było znalezienie takich składowych, aby wykres z dwoma z nich dobrze reprezentował wszystkie zmienne. Żadna inna kombinacja zmiennych nie poprawiła tej sytuacji. Niestety, faktyczne dane ze świata, jakimi są dane z Instytutu, nie zawsze dają się dobrze podsumować.

Jak w takim razie wyglądają najlepsze z powyższych wektorów? Widać to na (Rysunku 19).

	PC1	PC2	PC3	PC4
Altitude	0.379	-0.045	0.350	-0.270
Moisture	-0.183	-0.041	0.386	-0.321
Defects1	0.114	-0.084	-0.331	-0.410
Quakers	-0.056	0.235	0.011	-0.477
Defects2	0.326	0.160	-0.400	-0.193
DColor_1	-0.191	-0.521	-0.146	-0.350
DColor_2	0.256	0.628	0.100	0.063
DColor_3	-0.128	-0.104	-0.033	-0.002
DColor_4	-0.053	-0.229	0.088	0.485
DProcessing_1	-0.543	0.278	-0.116	-0.051
DProcessing_2	0.024	0.116	-0.095	0.038
DProcessing_3	0.521	-0.282	-0.145	0.104
DProcessing_4	0.112	-0.091	0.612	-0.125

Rysunek 19. Wektory Własne

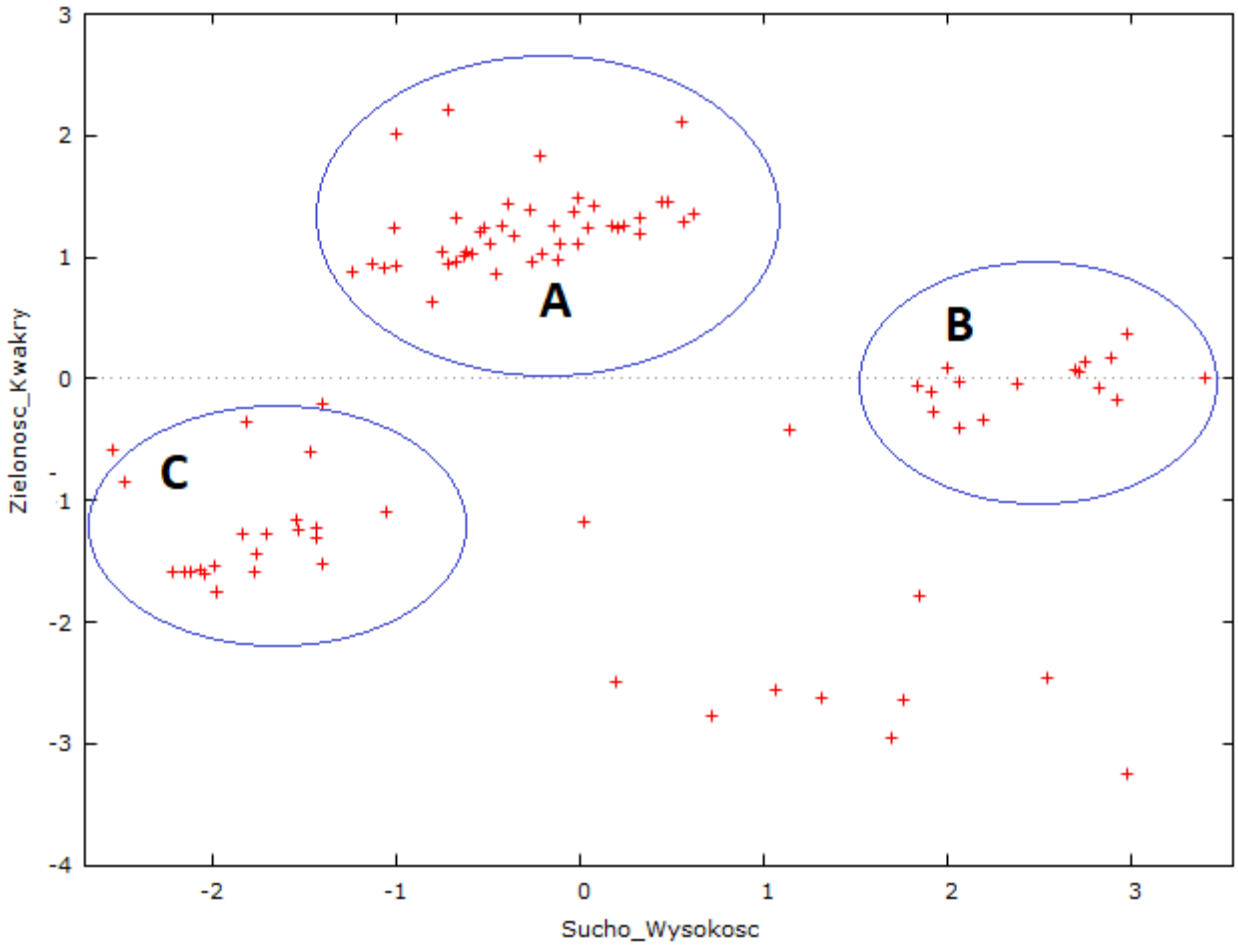
Znowu, wyniki nie wyglądają zachęcająco. Na Składową Główna PC1 najbardziej wpływają następujące cechy próbki:

- Wysokość n.p.m,
- Nie bycie mytą
- Bycie suszoną

Na Składową Główną PC2 najbardziej wpływają:

- Bycie zieloną
- Nie bycie niebieskawo-zieloną
- Istnienie kwakrów

Niech PC1 nazywa się Sucho_Wysokość, a PC2 nazywa się Zieloność_Kwakry. Niestety, razem stanowią ledwie 33% wariancji w modelu, ale i tak warto stworzyć wykres.(Rysunek 20)



Rysunek 20. Przedstawienie 13 zmiennych na wykresie 2D, za pomocą metody PCA

Zadziwiająco, dane faktycznie układają się w grupy. Wyraźnie widać grupy **A** – zielone kawy na średniej wysokości z większą ilością kwakrów, **B** – kawy na dużej wysokości oraz **C** – niebieskawe kawy na małej wysokości, z mniejszą ilością kwakrów. Cały sens metody PCA został zasadniczo osiągnięty – można zobaczyć prawidłowości.

Niestety, ze względu na specyfikę danych, małą ich ilość oraz prawdopodobnie zwyczajny pech, powyższy wykres nie przedstawia zbyt wiele interesujących informacji, ale pokazuje bardzo dobrze zastosowanie metody PCA.

Budowa modelu za pomocą MNK

Po użyciu Metody Najmniejszych Kwadratów powstaje poniższy model. Niestety, nie wygląda na to że wszystkie zmienne faktycznie wpływają na końcowy wynik. Okazuje się też, że metoda procesowania „Pulped natural / honey" ma dokładną współliniowość do koloru niebiesko – zielonego.

```
Model 1: OLS, using observations 1-94
Dependent variable: Score
Omitted due to exact collinearity: DColor_4 DProcessing_4

                  coefficient   std. error   t-ratio    p-value
  ------------------------------------------------------------
  const             0.904923     9.95084     0.09094   0.9278
  DColor_1          3.80370      5.88403     0.6464    0.5198
  DColor_2          9.32292      5.56656     1.675     0.0978   *
  DColor_3         -0.474355     7.68316    -0.06174   0.9509
  DProcessing_1     2.20173      7.01468     0.3139    0.7544
  DProcessing_2   -16.3015      15.1143     -1.079     0.2840
  DProcessing_3     2.66137      7.26190     0.3665    0.7149
  Defects2          0.164598     0.561209    0.2933    0.7700
  Defects1          1.85552      2.28633     0.8116    0.4194
  Moisture          2.45750      1.10660     2.221     0.0291   **
  Altitude          0.628205     0.138622    4.532     1.97e-05 ***
  Quakers           0.414617     0.810415    0.5116    0.6103

Mean dependent var   28.97872   S.D. dependent var   15.21529
Sum squared resid    12901.84   S.E. of regression   12.54350
R-squared            0.400749   Adjusted R-squared   0.320362
F(11, 82)            4.985236   P-value(F)           6.79e-06
Log-likelihood      -364.7062   Akaike criterion     753.4125
Schwarz criterion    783.9320   Hannan-Quinn         765.7402

Excluding the constant, p-value was highest for variable 25 (DColor_3)
```

Rysunek 21. Wstępny wynik zastosowania MNK

Dziwne jest to, że zmienne, które na logikę powinny mieć największy wpływ na wynik – defekty i kwakry – mają p – value tak wysokie że nie mają praktycznie żadnej wartości dowodowej. Metoda procesowania podobnie.

Końcowy model

Po usunięciu z modelu tych zmiennych widać końcowy efekt:

	coefficient	std. error	t-ratio	p-value	
DColor_1	7.35118	3.98238	1.846	0.0683	*
DColor_2	12.5166	3.50192	3.574	0.0006	***
DColor_3	3.14659	6.44557	0.4882	0.6266	
DColor_4	2.96067	6.09655	0.4856	0.6284	
Moisture	2.42733	1.02930	2.358	0.0206	**
Altitude	0.669388	0.115872	5.777	1.13e-07	***

Rysunek 22. Końcowy model ekonometryczny

1) Okazuje się że im wyższa wilgotność, tym lepszy wynik. W rzeczywistości uznaje się, że wilgotność przekraczająca 12% jest nieoptymalna, z 10% będącą dolną akceptowalną granicą. Tymczasem z danych na stronie wynika, że im większa wilgotność tym lepiej. Spowodowane jest to prawdopodobnie specyfiką próbek kaw reprezentowanych na stronie.
2) Jedyny kolor który ma jakikolwiek rzeczywisty wpływ na jakość kawy to Zielony. I to jak pozytywny! Bycie zielonym, w przeciwieństwie do nie bycia zielonym zdobywa dodatkowe 5 % – 9 %. Okazuje się, że faktycznie – zielone nasiona kawy uznawane są za będące bardzo wysokiej jakości.
3) Im wyżej nad poziomem morza, tym kawa lepsza. Ten punkt przynajmniej nie dziwi. Jest szeroko akceptowane, że wraz z wysokością nad poziomem morza idzie jakość i złożoność smaku kawy.

Ostatecznie, najlepszym wzorem, który da się osiągnąć, jest:

OCENA PRÓBKI = 7.35 * NIEBIESKAWO-ZIELONY KOLOR (0/1) + 12.51 * ZIELONY KOLOR(0/1) + 2.42 * WILGOTNOŚĆ (w procentach) + 0.67 * WYSOKOŚĆ N.P.M (w metrach)

Czy cechy dobrej kawy korelują ze sobą?

Jeśli chodzi o punkty w każdych kategoriach, można na przykład zobaczyć w jaki sposób korelują ze sobą nawzajem. Okazuje się, że prawie wyłącznie pozytywnie – lepsza kawa to po prostu lepsza kawa i jeśli osiąga wysoki wynik w na przykład Smaku, osiąga też wysokie wyniki w Balansie albo Kwasowości. Jedynym wyjątkiem jest słodkość, która nie jest wyraźnie skorelowana z czymkolwiek – jest swoją osobną kategorią. Wyniki widać na (Rysunku 23)

```
Correlation Coefficients, using the observations 1 - 94
5% critical value (two-tailed) = 0.2028 for n = 94

     Aroma          Flavor      Aftertaste         Acidity
    1.0000          0.7958          0.8099          0.7338 Aroma
                    1.0000          0.8323          0.7991 Flavor
                                    1.0000          0.8028 Aftertaste
                                                    1.0000 Acidity

      Body         Balance      Uniformity        CleanCup
    0.7698          0.7245          0.0965          0.0805 Aroma
    0.7312          0.7941          0.1489          0.1590 Flavor
    0.8469          0.8595          0.1332          0.1138 Aftertaste
    0.7665          0.7780          0.0562          0.0759 Acidity
    1.0000          0.8085          0.0830          0.0830 Body
                    1.0000          0.0747          0.0747 Balance
                                    1.0000          0.8855 Uniformity
                                                    1.0000 CleanCup

  Sweetness    CupperPoints
    -0.0235          0.7814 Aroma
     0.0108          0.8201 Flavor
     0.0619          0.8267 Aftertaste
     0.0217          0.7763 Acidity
     0.0841          0.7707 Body
     0.1077          0.8089 Balance
    -0.0181          0.1402 Uniformity
    -0.0181          0.1671 CleanCup
     1.0000         -0.0236 Sweetness
                     1.0000 CupperPoints
```

Rysunek 23. Korelacja między punktami częściowymi

Rozdział 3 – Data scraping wyników z Allegro.pl i wprowadzenie wyników do bazy danych

Dziedzina problemu

Celem w tej części pracy jest pobranie podstawowych informacji o każdym produkcie ze strony Allegro.pl wyświetlającym się po wpisaniu w oknie wyszukiwania konkretnego terminu i zapisanie ich w bazie danych. Zastosowany przeze mnie termin wyszukiwania to „kawa arabica 100% 1kg”.

Dane do pozyskania

Dane do zapisania to Nazwa produktu, Cena, Cena z dostawą, czy oferta jest od Super Sprzedawcy oraz Marka kawy. Marka jest oczywiście cechą szczególną, nie występującą w każdym możliwym wyszukiwaniu, ale w zależności od produktu którego rekordy użytkownik programu chciałby ściągnąć, można z łatwością zmienić element kodu aby szukać „Serii” albo „Rozmiaru”. Element z którego pobierane są dane widać na (Rysunku 24)

Rysunek 24. Wygląd elementu zawierającego dane

Proces pozyskiwania danych ze strony internetowej

```
login_email = 'xxx@gmail.com'
login_password = 'xxx'
search_term = "kawa arabica 100% 1kg"
sort_text = "cena z dostawą: od najniższej"
#trafność: największa
#trafność: najlepsza cena
#cena: od najniższej
#cena: od najwyższej
#cena z dostawą: od najniższej
#cena z dostawą: od najwyższej
#popularność: największa
#czas do końca: najmniej
#czas dodania: najnowsze
```

Rysunek 25. Deklaracja używanych zmiennych

Na początek zostały podjęte podobne kroki co w poprzednim przykładzie. Na początek zadeklarowane zostały potrzebne w działaniu programu zmienne. sort_text to zmienna, dzięki której można decydować jak sortowane mają być wyniki, co decyduje o późniejszej kolejności danych w bazie.

```
driver = webdriver.Firefox()

driver.get('https://allegro.pl/login/form')
waitForResponse()

dalejButton = driver.find_elements_by_xpath("//*[text()[contains(.,'dalej')]]")
dalejButton[1].click()

username = driver.find_element_by_name("username")
password = driver.find_element_by_name("password")
waitForResponse()

username.send_keys(login_email)
password.send_keys(login_password)
password.send_keys(Keys.ENTER)
```

Rysunek 26. Logowanie do Allegro

Po wczytaniu strony, znalezione zostają elementy służące do logowania i wysłane zostają do nich login i hasło. Kwestia przycisku „dalej”, którą można zobaczyć w powyższym kodzie to wczytujące się po wczytaniu strony okno proszące o zgodę na przetwarzanie danych osobowych.

```
found = 0
while found == 0:
    try:
        search = driver.find_element_by_xpath('//input[@data-role="search-input"]')
        found = 1
    except:
        found = 0
search.click()
search.send_keys(search_term)
search.send_keys(Keys.ENTER)

waitForResponse()

sort = driver.find_element_by_xpath('//select[@data-value="m"]')
select = selenium.webdriver.support.select.Select(sort)
select.select_by_visible_text(sort_text)
```

Rysunek 27. Ustawianie parametrów wyszukiwania

Wyszukiwanie pola do wpisania terminu wyszukiwania okazało się często sprawiać problemy, więc za pomocą prostej konstrukcji *while* wyszukiwane jest to pole aż do skutku.

Po znalezieniu pola „search-input” wysyłane zostaje do niego zapytanie, po czym wybrany zostaje element widoczny na (Rysunku 28), a następnie odpowiedni sposób sortowania.

Rysunek 28. Rozwijane menu z rodzajem sortowania

```
client = MongoClient('localhost:27017')
db = client.Allegro
while cont == 1:
    Products = driver.find_elements_by_xpath('//article[@data-item="true"]')
    for p in Products:
        (…)
    try:
        NextButton = driver.find_elements_by_xpath('//i[@class="_nem5f _1av8u"]')
        NextButton[0].click()
        waitForResponse()
    except:
        cont = 0
```

Rysunek 29. Połączenie z bazą danych i iteracja po produktach

Tworzone jest połączenie z MongoDB, z bazą danych „Allegro”, znajdującą się na komputerze. To do niej będą wysyłane pozyskane dane. MongoDB to szeroko używany, profesjonalny system i ma szeroką funkcjonalność, więc wysłane do niego dane mogą być potem używane w dowolny sposób.

Iteracja następuje po każdej stronie wyników, i po każdym na niej produkcie - <article>. Istnienie kolejnej strony z wynikami sugeruje istnienie przycisku „dalej”, który program próbuje znaleźć i kliknąć. Niestety, w tym miejscu kod strony www.Allegro.pl nie jest przejrzysty. Nazwa klasy przycisku dalej jest zwyczajnie niezrozumiała, i trzeba go było znaleźć funkcją „inspect”.

```
Name = ""
Price = 0.0
PriceShipping = 0.0
Super = False
Brand = ""

info = p.text
```

Rysunek 30. Deklaracja zmiennych z danymi oferty

Zostają zadeklarowane zmienne z odpowiednimi danymi. Następnie do zmiennej „info” przypisany zostaje tekst elementu. Na szczęście, w tym przypadku dane są w kodzie strony. Teraz trzeba tylko obrobić je i przypisać do zmiennych.

Zmienna *info* wygląda w następujący sposób: *'Oferta sponsorowana\nNa energię - KAWA ZIELONA Mielona Arabica 1kg\nOd\nSuper Sprzedawcy\n27,90 zł\nz kurierem\n35,90 zł z dostawą\ndostawa w środę\n2 osoby kupiły'*

```
info = info.replace("dostępne warianty\n","")
info = info.replace("Oferta sponsorowana\n", "")
info = info.replace("Oficjalny sklep\n", "")

if ("od\nSuper Sprzedawcy" in info):
    Super = True
    info = info.replace("od\nSuper Sprzedawcy","")

rows = info.split(sep="\n")

Name = rows[0]
rows.pop(0)

for r in rows:
    if ("Marka" in r):
        Brand = r.replace("Marka","")
    if ("zł" in r and "%" not in r):
        cut = re.search(r'\d+,\d\d', r).group()
        prc = float(cut.replace(",","."))
        if (Price == 0.0):
            Price = prc
        else:
            PriceShipping = prc

if(PriceShipping == 0):
    PriceShipping = Price
```

Rysunek 31. Parsowanie i zapis danych

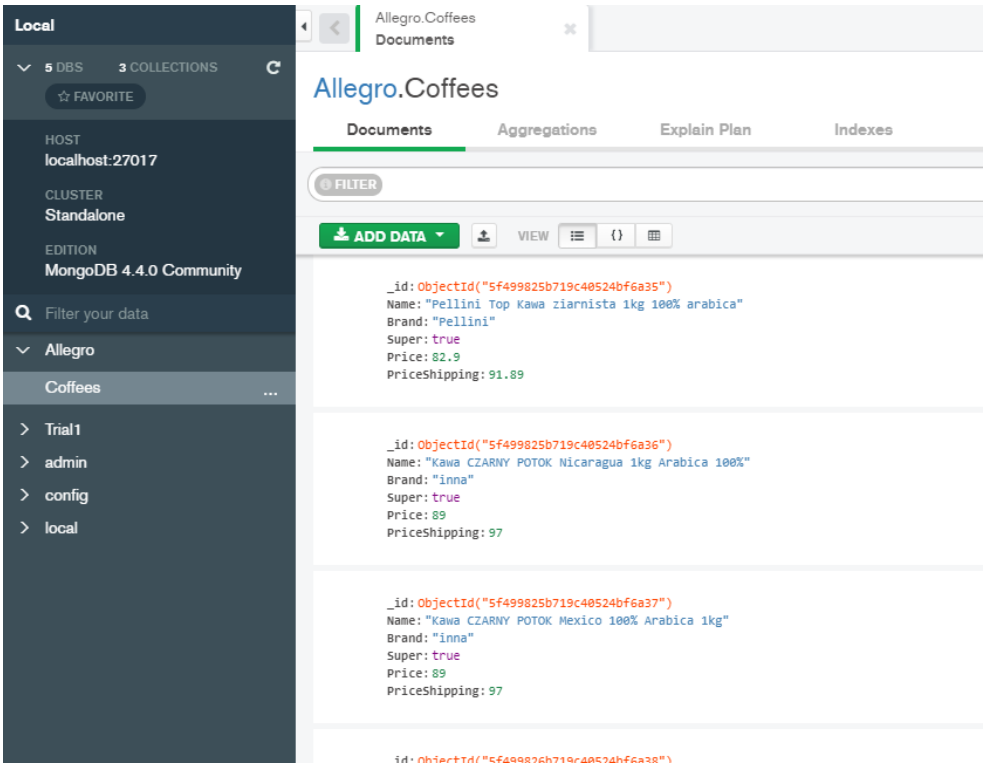
Następuje oczyszczenie zawartości zmiennej „info”. Usunięte zostają, czasami zdarzające się „dostępne warianty”, „Oferta sponsorowana” i „Oficjalny sklep”. Jeśli produkt pochodzi od Super Sprzedawcy, jest to zapisywane w zmiennej a następnie linijka z tą informacją zostaje usunięta. Kolejna jest już nazwa produktu. Następuje iteracja po wszystkich linijkach (uzyskanych za pomocą funkcji Split). Marka to po prostu to, co jest po słowie „Marka”. Pierwsze wystąpienie kwoty to cena produktu bez dostawy, drugie to z dostawą. Omijama jest linia ze znakiem %, ponieważ mówi ona o zniżce. (np -50%)

```
db.Coffees.insert_one(
    {
        "Name": Name,
        "Brand": Brand,
        "Super": Super,
```

```
        "Price": Price,
        "PriceShipping": PriceShipping
    })
```

Rysunek 32. Zapis do bazy danych

Następnie do bazy danych zapisywany jest jeden rekord. MongoClient obsługuje wszystko automatycznie. W sumie, dla zapytania "kawa arabica 100% 1kg" jest 810 rekordów.



Rysunek 33. Uzyskana baza danych Coffees widoczna w programie MongoDB Compass

Odczytywanie danych z bazy i przykładowa analiza

Aby udowodnić, że dane w bazie MongoDB są łatwo dostępne i użyteczne, pokazane będzie teraz przeze mnie jak je wyciągnąć z niej i wykorzystać. Używana będzie teraz biblioteka matplotlib.

W osobnym pliku zostają odczytane dane z bazy:

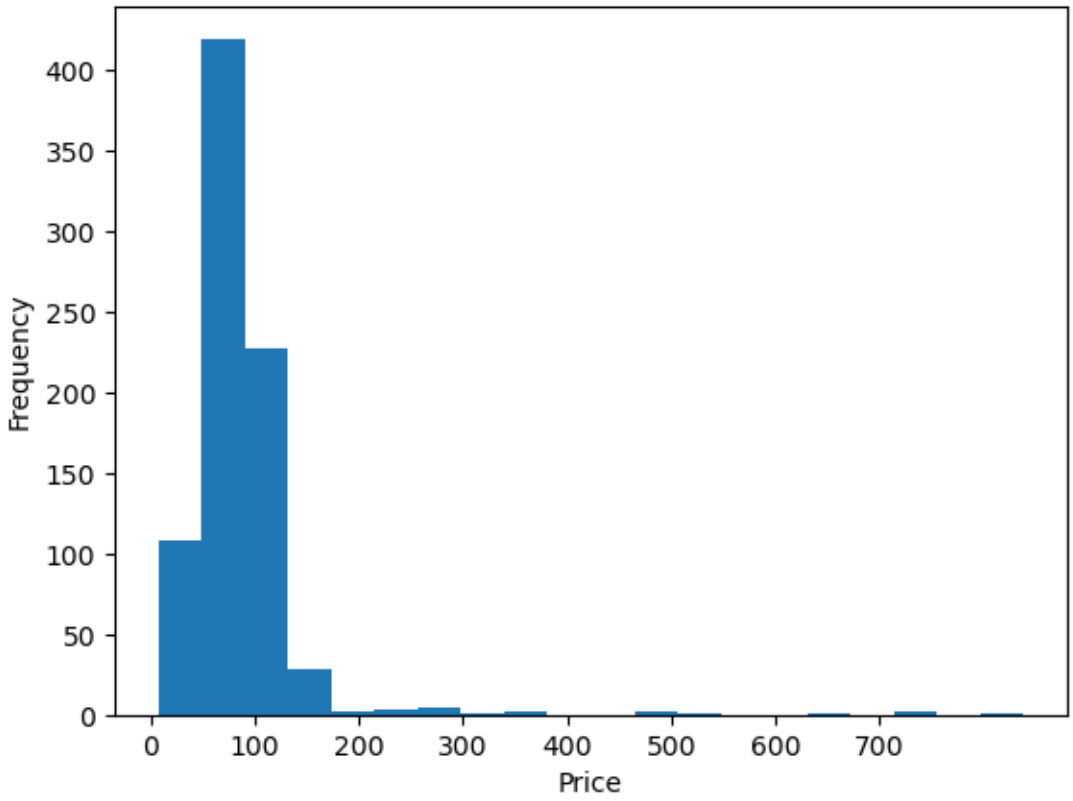
```
import matplotlib.pyplot as plt

client = MongoClient('localhost:27017')
db = client.Allegro

data = pd.DataFrame(list(db.Coffees.find()))
plt.hist(data['Price'],20)
plt.xticks(range(0,800,100))
plt.xlabel("Price")
plt.ylabel("Frequency")
```

Rysunek 34. Budowa histogramu

Można na przykład stworzyć histogram z rozkładem cen kilogramowych kaw Arabica na Allegro. Na (Rysunku x12) widać, że większość kaw kosztuje kilkadziesiąt złotych.



Rysunek 35. Histogram cen kilogramowych kaw Arabica

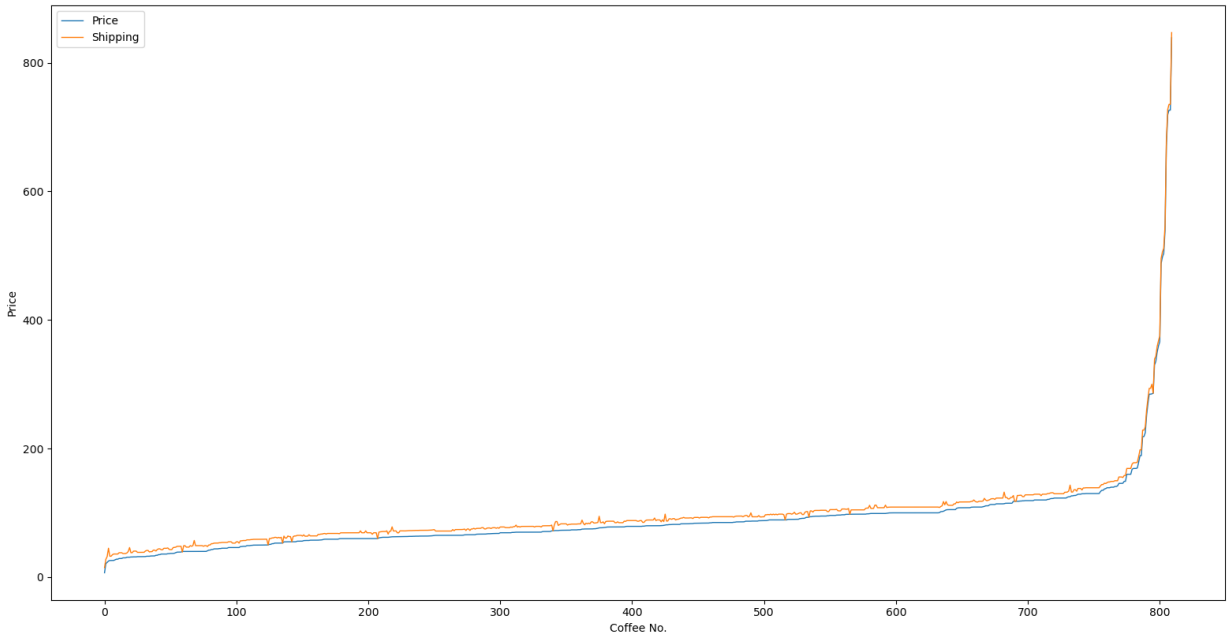
```
data = data.sort_values('Price')

PriceList = data['Price'].tolist()
ShippingList = data['PriceShipping'].tolist()

plt.xlabel('Coffee No.')
plt.ylabel('Price')
plt.plot(PriceList, label='Price', linewidth=1)
plt.plot(ShippingList, label='Shipping', linewidth=1)
plt.show()
plt.legend()
```

Rysunek 36. Budowa wykresu liniowego

Można też pokazać porównanie cen kawy bez dostawy i z dostawą. Po użyciu standardowej biblioteki matplotlib widać, że ceny dostawy raczej maleją wraz z ceną samego produktu.



Rysunek 37. Wykres ceny bez dostawy vs ceny z dostawą

Podsumowanie

Data Scraping jest przydatną techniką, jeśli chcemy pozyskać dane ze strony internetowej, a nie ma łatwo dostępnych sposobów na zdobycie ich. Jak zaprezentowałem, z różnych stron można pozyskać dane i wykorzystać je w profesjonalnych programach do analizy oraz przechowywania danych. Za pomocą szerokiego wachlarza dostępnych darmowo narzędzi, jest możliwe pozyskanie dowolnych danych z różnych stron. Może to być bardzo pomocne jeśli strona celowo lub nie nie daje nam interesujących nas danych gotowych do ściągnięcia. Dopóki mieścimy się w zakresie czynności legalnych, Data Scraping może pozwolić poszerzać naszą wiedzę.

Allegro okazało się stroną, która niechętnie wita automatyzację. Pola oraz atrybuty wyglądają na celowo nazwane nieintuicyjnie. Wszystko jest jednak możliwe, każde pole jest ostatecznie możliwe do znalezienia i kliknięcia. Proces automatyzacji jest organiczny i łatwo dostosowywalny do dokładnego zamiaru użytkownika. Dane przekazane do bazy danych mogą być łatwo dostępne dzięki bazie danych MongoDB i analizowane dzięki szerokiemu wachlarzowi pakietów w języku Python.

Strona Cofee Research Institute jest ustrukturyzowana w dostępny sposób, a dane są dokładne i łatwo pozyskiwalne. Niestety, sama baza danych nie jest zbyt wielka – podczas pisania tej pracy strona ograniczyła ilość dostępnych rekordów. W związku z tym sama analiza statystyczna próbek kaw Arabica nie przyniosła zbyt wielkich rezultatów – jedyne, do czego można było definitywnie dojść, to obserwacja, że dla znajdujących się na stronie próbek wysokość nad poziomem morza, wilgotność oraz bycie koloru zielonego są czynnikami pozytywnymi – co jest faktycznie prawdą, z zastrzeżeniem, że w szerszym kontekście eksperci uważają, że zbyt wielka wilgotność próbki może być jej wadą.

Zaprezentowane w tej pracy techniki przedstawiają niektóre możliwości Data Scrapingu i stanowią przykład jego zastosowania w Data Science.

Literatura

[1] Katharine Jarmul, Python Web Scraping: Hands-on data scraping and crawling using PyQT, Selenium, HTML and Python, 2nd Edition

[2] https://database.coffeeinstitute.org/

[3] https://www.freecodecamp.org/news/the-ultimate-guide-to-web-scraping-with-node-js-daa2027dcd3/

[4] https://www.techbeamers.com/locate-elements-selenium-python/#locate-element-by-link-text

[5] https://www.allegro.pl/

[6] (Youtube) StatQuest: Principal Component Analysis (PCA), Step-by-Step