UNIVERSITY OF WARSAW

Web Scraping and Social Media Scraping Project

„Scrapping of cars database from ebay.com"

by DSBA-1 students:

Sergey Amarin (id: 436854)
Oleksii Zymin (id: 430995)

Cars become an important part of lifestyle of every family. Nowadays we have thousands of vehicles on the streets and car trading market is huge. In current work we tried to scrap a database of cars and its specifications from announcements on ebay.com – one of the biggest marketplaces in USA and world.

Cars announcements situated in ebay Motors section > Cars&Trucks (https://www.ebay.com/b/CarsTrucks/6001/bn_1865117?rt=nc&LH_ItemCondition=3000%7C1000%7C2500&_stpos=10025). A user can select various options for search and filtering. At this moment there are more than 40 thousand announcements. And taking into consideration the fact that ebay will not show you more than 10000 announcements in Search results (https://community.ebay.com/t5/Selling-Q-A/Search-limit-Only-10-000-items/qaq-p/19862360), we decided to divide scrapping process into 3 parts:

1) Get list of links for every brand to avoid 10000 limits (no brand has such number of announcements)
2) Get list of links to all announcements of every brand;
3) Get car's specifications from every announcement.

We have made 3 scrappers: BeautifulSoup, Selenium and Scrapy.

Beautiful soup (created by Oleksii Zymin)

We get list of brands with 'find_all' function, because all the brands situated in the list of Search options in the right sidebar. For the second stage we assigned brand's 'url' variable, grab the html code and scrap links to every 48 announcements from a page, but there are 2 links for each announcement's html, so we took them all and then put odds to the list of links. On this stage we have face an unexpected behavior from ebay.com: BeautifulSoup could not grab any info from 5th page and above for every brand. We have come up with conduction that there is anti-scrapping protection and unfortunately, we could not break it. But still we have had at least 192 announcements for every brand. Finally, we used every link and scrapped item's specifications, which are situated into the table. These specifications can differ from car to car, because there are obligatory fields, like brand, year, year as well optional: number of doors, trim, options. We scrapped columns' names and values separately. We used dictionary to collect them all together, but as a negative effect get some NA values which then worked out during data cleaning process. Also, we scrapped price separately, covered all 3 variants of price: fixed, bid and ended bid.

Selenium (created by Sergey Amarin)

In selenium code we used ‘click()’ method to close all cookies banners and to get to the necessary page. We get list of brands by xpath and then ignored link of ‘Auriel’ cars and ‘Skoda’ cars, because there no announcements, and Selenium got a lot of junk from ebay’s recommendations. Before we select brand, we introduce 3 sec pauses, because list of brands is loading for 2-3 secs after the loading of the page. Selenium clicks on every brand and get all the links even from 5th pages and above. To get announcements link from all the pages we created a while loop with limitations by the last page number for every brand (calculated separately). Xpaths for links can be slightly different and we omit these risks with ‘try’ function for every observed combination. Finally, we got all item’s specifications from the table the same way as in BeautifulSoup: get columns names and values separately and gather it via dictionary.

Scrapy (brands and links – Sergey Amarin, cars – Oleksii Zymin)

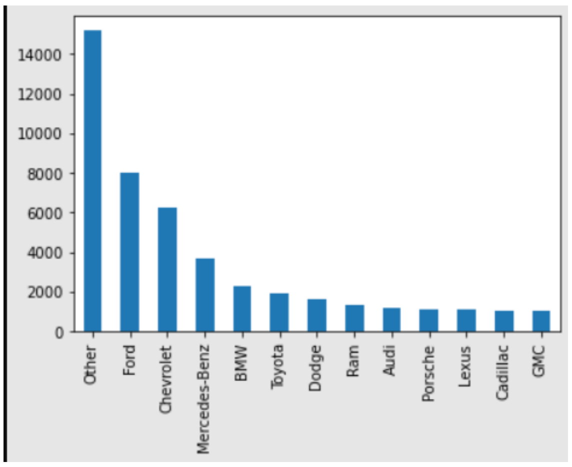
For scrapy we used the same 3 steps and created a separate spider for each one. But with links gathering we faced the same problem as in BeautifulSoup scrapping: scrapy could not get any info from pages above or equal 5. So, its strength our hypothesis that ebay has anti-scrapping protection. Another specific of Scrapy appears with writing csv file in the end: it takes column names from the first car and then fit all the others to this structure. This problem was discussed at Campuswire (https://campuswire.com/c/GA5783CE0/feed/109), but unfortunately solution was not found. So, to deal with this way of behavior, we found the car with maximum useful specifics and put it into the beginning of our table, and to be sure that this car will be scrapped the first we multiply the first car to 16, because it is the number of one-time responses sent by scrapy to the site. This way we get 100% chance, that needed car will be the first, so we get needed table structure in the end and can easily delete the first 16 cars from it to omit our maneuver.

Time comparison

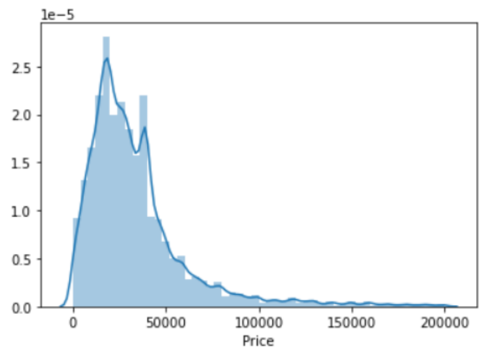
Step/Scrapper	BeautifulSoup	Selenium	Scrapy
Brands	1.81 secs	13.45 secs	1.58 secs
Links (100 units)	3.94 secs	6.36 secs	0.53 secs
Cars (100 units)	126.33 secs	239.34	8.59 secs

From table above we can clearly observe that Scrapy is the faster scrapper. And Selenium is the slowest.

Analysis:

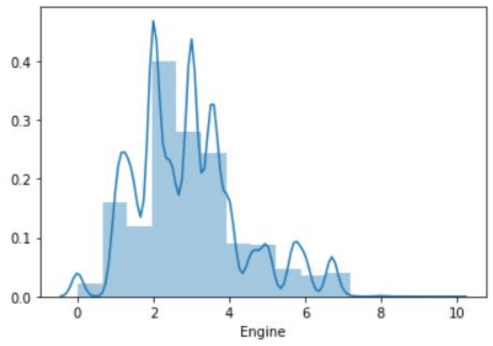


On American cars market local producers are dominating. However, car makers from Germany and Japan have significant shares.

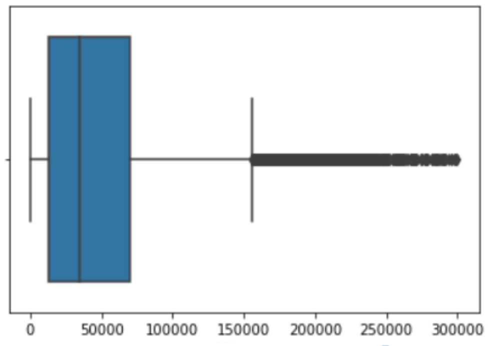


Price distribution is skewed to the left and there are a lot of outliers to the right. Mean price is 39 382 USD.

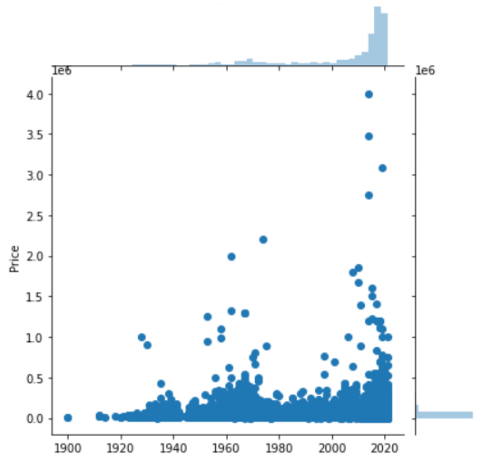
```
count      45787.00
mean       39382.55
std        65001.13
min            0.00
25%        16632.50
50%        27590.00
75%        41995.00
max      3999996.00
```



Mean engine volume is 2.94 L, while the most popular is around 2.5 L.



Average mileage is 50 879 miles.



For this scatterplot we can observe trend that newer cars are more expensive.

Conclusion

Even though Scrapy is the fasters scrapping method, we faced several issues with it. Firstly, it could not avoid Ebay's anti-scrapping protection even with ROBOTS_OBEY = False, so we had limitations with the number of links to cars. Secondly, Scrapy do not extend the output table columns when writing output to csv. It limits our grabbed data, but from another side could be used as filter, not to grab junk data. Output from every method was exported to csv files and needed data cleaning processing.

Selenium needs the most time to scrap, but it avoids anti-scrapping protection, which makes it useful tool. Also, it has a bunch of options for scrolling, clicking, input and etc., which can be beneficiary used in heavy websites.

BeaultifulSoup is relatively fast but have no interactions options and XPATH is a useful way to point the data.

Finally, the most efficient way to grab the data in our project would be:

1) Scrap brands and links with Selenium;
2) Scrap cars with Scrapy.

This way we will bet the best parts from every scrapper avoid anti-scrapping protection and briefly get filtered data without junk.