Web scraping Carvana

BAX: 422 Final Project

University of California, Davis

Master of Science in Business Analytics (2020-21)

Section 1

Jessica Padolina

Yuthika Agarwalla

Weitian (Lance) Xie

I. Executive Summary

Pricing is an important factor to consider in order to be successful in the used car market. The target demographic is price-sensitive and financially conscious consumers who are hoping to get a fair deal on their purchase. But, it's not as easy to put a price tag on a used car because there's lots of attributes to consider that affect the condition, depreciation, and remaining life of the car. This paper aims to describe methodology for scraping data from the website of Carvana, an online used car retailer, and the best method for storing the data for use in answering business questions. The dataset is filtered for a San Francisco zip code and the BMW brand. Access to such a dataset can prove beneficial in estimating value, popular features, and preferable conditions for a second-hand BMW to other dealerships in the city or similar car markets like New York City. Furthermore, our web scraping methodology can be applied to similar search-oriented websites for collecting data in domains outside of the used car market.

II. Background and Context:

The used car industry is massive with a global value of approximately $1.402 billion in 2020 and 40 million used car units sold in the US annually. The pandemic has only fueled the used car market due to multiple factors like shortage of new cars, more price-conscious consumers, and a desire to avoid mass transit methods during this dangerous time.

San Francisco, especially, is a tricky car market due to having one of the highest walking scores nationally. Abundant public transit methods and astronomical parking costs deter most of the population from buying and owning cars - especially new cars that lose half their value upon leaving the lot. And, earlier this year, San Francisco declared Market Street "car free" in a pilot to mitigate traffic and increase safety (Bliss). But circumstances like surge pricing on ride-sharing options and unreliable timings on public transportation can cause automobile ownership to become a necessity, despite the inconveniences.

A persistent problem in the used car business is profitable and attractive pricing (Sudhir, K.). It's not beneficial to have cars sitting on the lots for long periods; the goal is to turn over inventory fast (CarGurus, Inc.). But it's not as straightforward as pricing new cars which come with a predetermined price tag; used car sales are fraught with bargaining and ongoing promotions. So, this area is ripe for

statistical analysis to determine two things: optimal pricing for individual units and the order in which to sell inventory to derive the most financial value from the fleet, before depreciation affects potential profits.

III. Data Source, Web scraping, and Database Design

Data Source

Our data source is the Carvana website, which can be accessed at carvana.com. It is an online used car retail shop that offers all the services of a brick and mortar car dealership. Unlike traditional dealerships, customers can simply browse available vehicles, purchase a vehicle completely online, and either pick up or have the car delivered to them. The website provides users a 360-degree view of the car along with typical features and pricing information.

After a user enters relevant search terms, Carvana displays all car listings within several pages of tiles. The tiles only display basic information of each vehicle listing such as a photo, year, make, mileage and price. Each inventory car in a tile has an unique URL that contains its Vehicle ID. Clicking on the tile/URL loads an entire page of additional car's features, options and packages. Moreover, there are Carvana-specific data points such as number of likes or views on that car so a potential customer can gauge popularity and future availability of the unit.

We believe that Carvana provides a comprehensive source of vehicle data. With careful scraping routines, we should be able to build a powerful dataset for used car value prediction. We will cover the variable specifics in section *Dataset Choices*.

Web-scraping Routines

Our general structure of Web-scraping consists of 4 major steps:

1: Send Zip Code to Carvana Website

2: Extract URLs of all car listings and save locally.

3: Open each car listing URL. Save contents locally.

4: Scrape the variables and save to SQL database.

Step 1: Sending Zip code

There are many tasks that run on the background of the Carvana Zip Code search page. For example, it sets a cookie before you enter the zip code already and it sends the zip code verification through their internal API. Due to the complexity, it is difficult to use standard GET and POST requests to mimic the zip code search process. Thus, we decided to use the Selenium Package and Chrome webdriver. This package controls our Chrome browser through Python and interacts with the website. It does an excellent job at mimicking human interaction with a browser.

First we set the search term "bmw" as part of the URL to look up. We then identified the web element (search bar) that allows us to enter the zip code. We used the click() command to interact with the search bar. Once the search bar has been clicked on and is ready for input, we asked our web driver to press the "Backspace" key five times to clear all the pre-filled zip codes. Finally, we sent the key "94103" to the search bar and asked the driver to press "Enter".

Pop-up windows are our next roadblock. It is possible that the zip code search window does not automatically close. It is also possible that Carvana sends us an email advertisement pop-up window. In both cases, the search/pop-up window prevents us from accessing the main content of the page. As a solution, we designed functions that look for "close" buttons of the pop-up windows. Carvana stores the buttons either in an svg element or a button type element. Should the web driver identify such a close button, it will interact with the button and close the pop-up window. Now, we can move forward with the search results.

Step 2: Extract Car Listing URLs

To extract all URLs, we will need to find the web contents from the first search page to the last search page. Although carvana uses "pgn" to define the individual page number, there is no element that defines the total page number of the search results. This is because sometimes there are advertisements rather than car listings on random tiles. The total number of pages changes dynamically as you go through each page. We find it difficult to go through the pages by changing the "pgn" variable, because the exact number of pages keeps changing. We noticed that on each page, there is an enabled "next" button that

will take you to the next page. On the last page of the search results, the "next" button still exists but will be disabled. Thus, we designed the following logic on Selenium in order to scrape car listing URLs. First, save the source code of the current page and look for the "next" button. Second, if the "next" button is enabled, click it. This will take us to the next page. Save the source code again on the next page and continue looking for the "next" button then click it. After many iterations, the algorithm will eventually find a "next" button that is disabled. This indicates that we are on the last search page, and we have saved all the relevant page contents. With the source code of each page saved locally, finding the url of each car became a breeze.

Step 3: Scraping contents from Car listing URLs

Originally, we used standard GET requests to acquire web contents. It worked smoothly and we collected all the contents fairly quickly. However, upon closer inspection on the downloaded contents, we found a lot of information missing. The root cause is that Carvana intentionally designed its website to only display some information at first. Once you scroll down, the page automatically loads more content. Unfortunately, many variables of interest are hidden behind the "load more" feature on the website. We struggled to find a working solution using GET.

Selenium web driver helped us again in this step. We used the Chrome web driver to open each car listing URL. Once the page was open, we used Python to command the web driver to scroll down to the bottom of the page. Most of the elements would finish loading after a few seconds. We can then safely download the source codes that contain all variables using this method.

One important thing to note is that to achieve comprehensiveness, we sacrificed efficiency. Taking into consideration the loading time, the function spent nearly 2 hours in scraping all source codes, and this was just for a sample set of 661 cars. The 661 BMW listings were acquired on March 13, 2020. As Carvana sells and refills its inventory, it is very likely that the inventory listings will change on a daily basis. Our algorithm will most likely pull different results each time it is run. For the sake of consistency, we decided to stick to the listings on March 13 for our analysis.

Step 4: Scrape Variables and Save to Database:

Having collected each car result as its own text file, we were ready to scrape all variables of interest using RegEx. Using BeautifulSoup would have achieved the same results too. Our program initialized empty lists to store *n* instances of each variable, *n* being the number of text files saved. Then it iterated through each file, opened it, searched for RegEx patterns, formatted the patterns, saved each result into their respective lists, and sent the result to our SQL table. We also ensured that each variable saved as 0 or an empty string if no result had been found. The vast majority of variables were easy to locate and scrape, albeit time-consuming. The most challenging variable to scrape was "Engine Type", which was a string consisting of 3 items (example: "4-cyl, Turbo, 3.0 Liter"). While originally we wanted to split this variable into 3 separate strings, it proved to be time consuming as some Engine Types did not follow the standard format (having two items instead of three, showing liter as "L"). To save time, we stored the entire string and scraped the Cylinder Count from another field as its own variable. Though we intended to engineer Engine Type in our machine learning project, we ended up not using it.

The main reason for storing the variables as lists as well as sending each instance to SQL is to ensure the code scraped the same amount *n* for each variable, and for testing. The end result is a dataset containing 661 rows and 27 columns. It's noteworthy that some columns contain more nulls than others, because some pages did not contain the variables we sought. For variables like "imperfections", null values properly indicate the lack of that variable level and are thus induced to be 0. For others, like "series", this variable was not scraped and is merely represented by a blank space. We will handle missing value imputation in our machine learning project.

Dataset Choices

While it is possible to scrape features of multiple brands, we needed to restrict our search down to a reasonable range for a time restricted project involving a beginner's level of experience with web scraping. For that reason, we specified our single brand of interest to be BMW, and focused on cars available within the San Francisco zip code 94103. From this sample, we wanted to scrape as many features as we believed to be useful when analyzing prices and attributes.

The dataset is a mix of car specific columns, condition specific columns, and Carvana specific columns. The car specific columns like model, series, cylinder count, color are included to identify what features of a car are favorable and which model/series in the BMW brand are listed at higher prices. Condition specific columns like imperfections can show the magnitude of depreciation differences in usage conditions like mileage can have on cars. And Carvana specific columns like view count and save count are indicative of what consumers are responding to or considering to be a worthwhile deal. An amalgamation of all this information can be highly deterministic of pricing strategies and decisions.

Database Design

The data is being stored in an RDBMS because it has a tabular structure, is static, and has low volume or scalability. Every column is being added into a single table because our analytical intentions require using all of the data in tandem. We considered splitting it into separate tables to divide between car features and Carvana features, but that is not necessary for our business use case.

Some of the less obvious columns we included in the dataset are:

1. *Great_deal:* This is a tag on a Carvana listing's tile if it's classified as a good deal by the company. It would be interesting to see if the existence of this tag increases views and saves, and ultimately verifying if these listings are in fact a better deal than other listings or if it's a marketing gimmick.
2. *Photo_count:* The number of photos attached to the listing details on Carvana, and if that justifies a higher price.
3. *Tax_title_reg:* Taxes, title, and registration. This is an additional cost that gets added to the sale price of a car for some listings.

Data Types

Columns:	
id	int AI PK
VIN	varchar(20)
StockNo	int
model	varchar(50)
series	varchar(500)
price	int
tax_title_reg	int
mileage	int
cylinder_count	int
engine_desc	varchar(50)
horsepower	int
torque	int
fuel_description	varchar(50)
drive_trains	varchar(50)
seats	int
mpg_city	int
mpg_hwy	int
ext_color	varchar(50)
int_color	varchar(50)
transmission1	varchar(50)
transmission2	varchar(1000)
doors	int
great_deal	tinyint(1)
saves	int
views	int
photo_count	int
imperfections	int

IV. Business Use Case

Carvana is a public company with a market capitalization of $50 billion, and a stock price on a steady rise. This indicates that it's been successful in the used car industry, where price is a central factor in many of the transactions and strategic price can be the difference between successful turnover of inventory or cars idly sitting in a lot. From this, Carvana's pricing strategy of its inventory can be considered accurate. Using this as a source dataset to predict car prices would be valuable for other car dealership businesses breaking into the same space or working to price new units of their inventory.

Not only price, but we can use the saves and views objects to determine what's popular amongst car buyers. This can help decision making around which cars to acquire, which is just as important as pricing because it's not easy to maintain a profitable fleet of used cars. Especially this year with low

manufacturing, there are not many avenues for dealerships to get specific cars from. So, it becomes even more vital to pinpoint which models are worth creatively hunting for.

Because we have collected information about so many features of a car, it is possible to conduct analysis that can uncover which features are more important or deal breakers for potential buyers versus which features or lack thereof they're willing to overlook. This will play an important role for used cars dealerships because they don't have extensive choice when deciding which car to add to the inventory. The supply usually comes from trade-ins, cash buys, or automobile auctions. If it can be concluded that customers aren't picky about the color of the seats, then the dealerships do not have to worry about it as much as the mileage on the car, for example. This can ease the process of car buying for the dealerships.

V. References

Bliss, Laura (2020). “What Happened After Market Street Went Car-Free”, Bloomberg CityLab (10 March)

CarGurus, Inc. “CarGurus Research Outlines Key Challenges Facing Used Car Dealerships in 2020.” GlobeNewswire News Room, "GlobeNewswire", 14 Feb. 2020

Sudhir, K. “Competitive Pricing Behavior in the Auto Market: A Structural Analysis.” Marketing Science, vol. 20, no. 1, 2001, pp. 42–60. JSTOR