Price Discrimination in Online Marketplaces

Timothy Ose

April 19th, 2017

Abstract

This paper looks at the impact IP address location data has on the prices customers see while shopping at various online retailers, within the United States. The paper analyzes data from Amazon, E-Bay, Overstock.com, and Wal-Mart.com and regarding 128GB iPad Air 2s, 6 foot unbranded HDMI Cables, and generic 500 sheet reams of 8.5 X 11 unlined printer paper. First, the paper looks at where, if at all, customers would see different prices on the same item. Next, it looks at search results, recommended products, and suggested 3rd party seller options to see if this data is targeted by location. Finally, where search and/or recommended product targeting is found to occur, the paper attempts to determine if the targeted search results have a statistically significant difference in price from the product initially being sought. Using an OLS regression, the paper finds no evidence for price discrimination on any of the products analyzed within the study, and finds no evidence of 3rd party seller targeting as well. One retailer, E-Bay, is found to target search results and product recommendations to customers visiting the site from Alaska or Hawaii. Furthermore, the paper finds statistically significant evidence that the prices of alternate search results and recommended products are higher than the item being sought, though confounding factors may exist.

1. Introduction

1.1 Overview

In today's world, computers and many of the advantages they bring are becoming an ever larger part of the economy. Specifically, the advent of big data computing has allowed companies to obtain and analyze information that would have been impossible to collect in the past. This type of computing works by feeding a computer program an enormous amount of data, without a predetermined theoretical model for analysis, and allowing the program to search for patterns and trends. Recently, this kind of computing has begun to revolutionize a number of fields, but none more profoundly than online shopping and online advertising.

With the enormous amount of information computers are able to store about your activity, from previous online purchases to what links you have followed on social media sites, companies are able to obtain the quantity of data necessary for effective big data computing. Thus, these companies are able to ensure that you are shown advertisements for things you are more likely to be interested in, are able to direct you to products you are more likely to buy, and perhaps are able to change the listed prices in order to increase their profits. For example, your IP address, internet cookies, and browser fingerprint are all sent to a website you choose to visit. All of this information can then be compiled and compared to similar data received from other computers that visited the same website, allowing companies to predict your activity. My study will attempt to track the impact various IP addresses have on the prices you see for certain common products on leading online retail websites. It will also determine if this data has any impact on the results of searches, recommended products, and 3rd party seller options you are shown.

1.2 Importance of the Issue

This issue is important for far more than academic reasons; knowledge of what impacts these fluctuations could have a significant impact upon the lives of many who use online services every day. Price discrimination based on big data analyses could have a profound impact on the lives of many in our society, and a knowledge of what it targets and where it is prevalent would be immensely valuable. If this kind of price targeting is indeed widespread, individuals that are giving up larger amounts of personal information would likely not be seeing the best possible prices on items they would like to buy while online shopping. A knowledge of which companies target this information, which specific data points they target, and how much these targets impact prices could make informed shopping much easier and could allow people to receive much better

prices on their purchases. Experiencing the online environment through a carefully crafted filter dramatically cuts your ability to discern the truth, and being able to understand and escape that filter would be very valuable for anyone who wants a more authentic and well-rounded view of what online marketplaces can offer.

2. Previous Literature

Price discrimination in online marketplaces has been explored, however it has been looked at primarily with focuses very different from my own. One study was conducted by the American Marketing Association and tried to determine if online car listings would adjust prices based on a consumers' use of various online infomediary websites (Agarwal, et. all). These price changes would be based on a specific theory, however, as car companies would adjust these prices having already made the assumption that consumers who visited infomediary websites would be more informed. It does not attempt to track price changes based on patterns in seemingly insignificant data, as are discerned in a big data analysis. Another study, conducted by the University of Pennsylvania, looked at price discrimination in online travel markets (Clemons, et. all). The goal of this study, however, was simply to see if such discrimination exists, not to track the price changes as a function of any information given to the websites by a consumer's computer. The specific focus of my study on tracking changes in price due to changes in information that could be easily used in big data computing allows me the opportunity to uncover unique results.

Recently, ProPublica conducted a study similar to this one, though covering the prices shown for services, rather than products. Specifically, the study looks at the prices shown to customers for Princeton Review SAT help sessions by ZIP code (Larson et. all). The study finds as much as $1,800 dollars in variation of cost based on income, and also finds that higher prices

are almost twice as likely to impact Asian-American residents, indicating potential racial profiling (Larson et. all). While this study is interesting, there are numerous reasons why online price discrimination is more likely in services, such as this. For one, this is a very specialized product that likely has few substitutes. Furthermore, while shipping costs of products such as iPads and HDMI cables are trivial, the travel costs required to attend a prep class in a different ZIP code are almost certainly non-trivial. The existence of this price discrimination, though, does highlight the importance of determining if it occurs in product markets.

Another interesting study conducted at MIT looks at prices for numerous products compared to their prices at physical stores (Clemons et. all). The study finds that prices are identical most of the time, but does find variability by country and several other factors. While not the focus of the study, the researchers also found no evidence of price variability by IP address (Clemons et. all).

Tangentially related to this is the issue of advertisement targeting. In this field, however, more definitive evidence of targeting exists. While few have attempted to untangle what exactly firms choose to target, many other studies have been conducted. Some issues that have been explored are to what extent this kind of advertising violates personal privacy, and is this kind of advertising actually viable and effective. In terms of personal privacy, researchers at Stanford and NYU built a study that attempts to determine if tracking algorithms used to obtain personal data overstep any ethical or legal boundaries (Barocas, et. all). This study has a very unique goal—it seeks to uncover potential wrongdoings rather than seeking to understand the inner working of the algorithms. Also, the 39th Hawaii International Conference on System Sciences published a paper using data mining to determine what kind of information is good at predicting the purchase patterns of a consumer (Cheng et. all). While study has been conducted on many

problems similar to the one I am approaching, results in this experiment would still provide significant new insight into the way online sellers conduct their business.

3. Methodology

3.1 Previous Attempts and Failures

Initially, my experiment was intended to be much different than it is now. The focus was spread out between advertisement targeting and price discrimination and intended to pinpoint the factors responsible for both, rather than focus on a specific factor (e.g. IP address). I began by developing a regression in which I had 4 dummy variables for IP Location (international vs US), online fingerprint, cookies, and purchase history at the various websites being explored. If the dummy variable was a 1, then the website would receive my information, if it was a 0, the given information would be hidden. Data for IP address was randomized and the status (international or domestic) was determined later. The statistical models that were analyzed were as follows:

$$\mu_{i,j} = \beta_1 * C + \beta_2 * H + \beta_3 * F + \beta_4 * L + K$$

where $\mu_{i,j}$ is the mean price of item i at website j; C is the binary cookies bucket; H is the binary purchase history bucket; F is the binary fingerprint bucket; L is the binary bucket IP address location; and K is a constant intercept term, for price discrimination; and

$$P_{i,j} = \beta_1 * C + \beta_2 * H + \beta_3 * F + \beta_4 * L + K$$

where $P_{i,j}$ is the proportion of adds in category i at website j; C is the binary cookies bucket; H is the binary purchase history bucket; F is the binary fingerprint bucket; L is the binary bucket IP address location; and K is a constant intercept term for advertisement targeting.

This analysis was meant to determine which factor had the biggest impact on price and/or on the proportion of advertisements you would see from various categories. My exploration into

this analysis hit many roadblocks with developing the required code to gather the data, determining if all of the variation was occurring correctly, and with the eventual analysis of the results. While not successful, this line of inquiry did lead me to develop my final methods of analysis.

3.2 Data Collection

My data collection methodology was driven by the python webbrowser package. I developed a python script (available in the appendix of the paper) that would interact with the command line capabilities of a VPN system to visit the webpages I needed. These webpages were the page of search results for each product at each website, and the actual product page for each product at each website. Once the algorithm had pulled up each page, I recorded the data into an excel spreadsheet. Next, I had the algorithm revisit each of the webpages and checked my recorded data against the new pages. If variation was found between identical pages on the revisit, this could indicate that variation seemingly due to IP was actually due to the time of the visit, even if the time difference was very small. No variation was ever found on revisits.

From each search result page, I collected the price and short description of the top 5 results. From each product page, I first recorded the price of the product, then I recorded the top 5 recommended products and their prices. Finally, on Amazon only, I recorded the rankings of the top 3 options of 3rd party sellers and any available data on their shipping prices to the IP being used.

The websites I visited were Amazon, Overstock.com, E-Bay, and Wal-Mart.com. Target was initially included in the data collection, but rejected the requests from many of the VPN IP addresses. For consistency, I omitted Target from my analysis. I picked these websites as I felt they gave a representative sample of the various kinds of firms that conduct online retail sales.

Further, the products I analyzed were the 128 GB iPad Air 2, 6 foot unbranded HDMI cords, and 500 sheet reams of 8.5 X 11 unlined generic printer paper. I thought the various prices, branding levels, uses, and customer bases would provide an interesting window into the online marketplace as a whole. All data collection was done in incognito browsing with cookies turned off as I did not want browsing data generated during the data collection to impact site visits later on in the data collection process.

3.3 OLS Regression

My final regression model was based entirely on IP address and used a series of dummy variables. The exact model was as follows:

$$P_{i,\,j} = \beta_1 * L_1 + \ldots + \beta_{20} * L_{20} + K$$

where $P_{i,\,j}$ is the price of item i at site j, L_m is the dummy variable corresponding to IP location m, β_m is the corresponding OLS coefficient, and K is a constant intercept term. The locations of each dummy can be found on the following map and were chosen in the interest of geographic diversity and under the constraint of what was available through the VPN. A table listing all locations follows:

Location 1	**Anchorage, Alaska**
Location 2	**Phoenix, Arizona**
Location 3	**Los Angeles, California**
Location 4	**San Francisco, California**
Location 5	**Wilmington, Delaware**
Location 6	**Miami, Florida**
Location 7	**Atlanta, Georgia**
Location 8	**Honolulu, Hawaii**
Location 9	**Chicago, Illinois**

Location 10	**Des Moines, Iowa**
Location 11	**Louisville, Kentucky**
Location 12	**Boston, Massachusetts**
Location 13	**St. Paul, Minnesota**
Location 14	**Billings, Montana**
Location 15	**Las Vegas, Nevada**
Location 16	**New York City, New York**
Location 17	**Portland, Oregon**
Location 18	**Nashville, Tennessee**
Location 19	**Dallas, Texas**
Location 20	**Washington, DC**

3.4 Search, Recommended, and 3rd Party Sellers

The data for search result, recommended product, and 3rd party seller differences, collected as described above, is purely qualitative. As such, determining if these differences exist did not require any method of statistical analysis. I compared results across locations and noted any differences I found. While developing a method of rigorous analysis for this may have been worthwhile had substantial variability existed, I found almost total homogeneity except in a few cases, as will be discussed in the results section.

3.5 Recommended Products Analysis

My final analysis was to look at variation in search and recommended products, where they existed, to determine if there was a significant difference in price between the product being sought and other search results given or products recommended. I ran this analysis only where I

found evidence of variability in the suggested product and/or search result areas for several reasons. First, if the firm had shown willingness to target IP address, then, in my opinion, there would be a higher likelihood that that firm would be willing to use search results to encourage customer to by higher priced products. Also, and much more importantly, without any variability in product recommendation or search results, I would have been running this analysis on a sample with N = 5, which is far too small for any meaningful result.

First, I looked at recommended products and, for each product individually, conducted a single sample t test against H_0: The mean of reccomended product prices is equal to the price of the product being sought. This should determine if the price of the primary product we selected is randomly sampled from the distribution of the search results/product recommendations. The sample used was the set of unique products showing up in recommendations. While doing this, I did not include accessories (e.g. iPad cases) as this would skew the data away from what I was trying to measure: do the product recommendations attempt to get you to buy a nicer version of the product you are looking for? I conducted this analysis for each product individually, and then for all 3 together by normalizing price based on the mean price of all products including the actual product being sought. I then conducted the same analysis, again for each of the products, based on the prices of other items that came up in the search results, and finally conducted the analysis on the set including both reccomended products and search results.

4. Results

4.1 Results of Failed Analysis

I was able to gather and analyze data for one iteration of my previous experiment, but ran into many issues with finding any relevant results and with confidence in the results I did find. For example, while it is well known that many firms target advertising, my method of looking at

the proportion of advertisements from various categories was unable to capture this targeting Therefore, any results in this area were unable to achieve their goal of determining what firms focused on when targeting advertisements. With regard to price discrimination, the binary bucket for location did introduce variability when looking at international IPs but this was likely entirely due to exchange rate volatility rather than price discrimination. As such, my results in that section would have been misleading at best. With these setbacks in mind, I developed my final experiment which I believe effectively measures variability only due to IP address and answers the question it is trying to ask.

4.2 Summary Statistics

The following tables show the average price on each item in question, as well as the standard deviation for that price. Data is broken out by retailer.

	Amazon	**Overstock**	**E-Bay**	**Wal-Mart**
iPad	$444.99	$664.62	$439.00	$479.95
St Error	*0.00*	*0.00*	*0.00*	*0.00*
HDMI	$5.99	$4.49	$4.00	$4.99
St Error	*0.00*	*0.00*	*0.00*	*0.00*
Paper	$7.99	$17.49	$6.67	$7.49
St Error	*0.00*	*0.00*	*0.00*	*0.00*

The following table shows mean price and standard deviation data for the product recommendations associated with each product. Data is structured in the same way as the previous table.

	Amazon	Overstock	E-Bay	Wal-Mart
iPad	$24.16	$495.50	$280.77	$111.99
St Error	*14.68*	*47.89*	*66.78*	*66.13*
HDMI	$42.93	$6.74	$8.00	$7.12
St Error	*40.93*	*1.32*	*6.29*	*4.93*
Paper	$33.22	$31.66	$19.50	$9.24
St Error	*23.39*	*22.21*	*1.66*	*2.87*

The fact that the Amazon values (and Wal-Mart to a lesser extent) are substantially more varied from the prices of the products in this table is no accident. While many of these firms focused their reccomended products sections on recommending other similar products, Amazon uses the section primarily to recommend related products that would be purchased with the initial product in question. For example, Amazon puts cases reccomended with iPads, TVs reccomended with HDMI cords, and printers reccomended with paper. The low standard deviation for E-Bay's paper recommendations is due to the fact that E-Bay primarily reccomended different colors of the same brand of colored paper, which I treated as distinct products. Also, E-Bays low mean value for iPads is due to the prevalence of used products rather than recommending less expensive accessories.

The final table gives the same information as the previous two for search results.

	Amazon	Overstock	E-Bay	Wal-Mart
iPad	$463.59	$250.02	$389.06	$420.24
St Error	*59.01*	*313.86*	*84.01*	*57.21*
HDMI	$8.60	$9.51	$6.60	$6.57
St Error	*2.39*	*5.25*	*6.17*	*2.70*
Paper	$10.13	$25.21	$12.20	$7.89
St Error	*2.95*	*13.06*	*6.04*	*3.63*

The most notable result in this table is the low iPad mean value for Overstock which is due to the fact that they do not carry many versions of iPads causing accessories to show up high in the search rankings on that website.

4.3 Regression Results

Each regression shows no evidence of any kind of price variability based on IP Address; in fact, not one price change due to IP occurred at any point throughout the data. The standard error was found to be zero for each product at each retailer. This makes us conclude that it is highly unlikely that any of these firms engage in location based price discrimination on any of these products.

4.4 Search, Recommended, and 3rd Party Seller Results

For Amazon, Overstock, and Wal-Mart all search results, recommended products, and 3rd party seller suggestions were identical to the position of each these items occupied on the webpage. For E-Bay, the same was true within the lower 48 states, however in Alaska and Hawaii substantial variation occurred in both the rankings of the products listed and the actual products appearing for both search results and reccomended products. For iPads, variability occurred in both of these categories. For HDMI cords, only search variability was found, and for paper, only product recommendation variability was found. While only 2 IPs showed any of this variability, the fact that both are outside the lower 48 states indicates that it is unlikely this variability is due to chance and may be due to the fact that individual sellers on E-Bay do not have the economies of scale to make shipping a trivial cost to remote locations. As such, E-Bay may downgrade single-item seller products within the search results and product recommendation in order to avoid cancelled transactions due to high shipping costs.

4.5 Recommended Product Analysis Results

In the following results I compared the price of the primary product sought (e.g. the iPad Air 2) to the distribution of the prices of the 1) recommended products 2) search results for that product 3) and a compilation of the two. The results of my one sample T-Test with H_0: The mean of recommended product prices is equal to the price of the product being sought follow:

Reccomended Products:

Analysis	P-Values	T-Statistic
iPad	0.000003	-7.447171
HDMI	0.222717	1.391822
Paper	0.000012	8.609453
Indexed	0.005933	2.961724

Search Results:

Analysis	P-Values	T-Statistic
iPad	0.123370	-1.699883
HDMI	0.349000	1.032899
Paper	0.059096	2.615200
Indexed	0.087193	1.798598

Compilation:

Analysis	P-Values	T-Statistic
iPad	0.000008	-5.680412
HDMI	0.101528	1.786785
Paper	0.000021	6.254313
Indexed	0.000027	4.808868

This data shows that, despite the significant downward statistical pressure from iPads, E-Bay does, in general, try to show higher priced items in the search results and recommendations on their website. The complied indexed results are significant beyond the 1% level, indicating strong evidence that EBay does use the search results and product recommendations as a way to upsell customers.

4.6 Discussion

While I think the conclusions regarding IP address targeting are very convincing, there is more research that could be done regarding the other tests and issues I looked into. First, while I found that, for these products, differences occurred in search results outside the lower 48, it would be worth testing to see if this result hold for all products in general. Also, adding other IP addresses outside of the US could provide interesting data. If more research was to be done in this area, a way to quantify differences in search results or product recommendations would need to be developed. An option would be to look at a set of products that show up in the search for another product and see if there is a statistically significant difference in the ranking they have in the search results for that product based on location. This could be repeated for numerous products and search results in order to get a better idea of if this difference occurs with all products outside the lower 48 states.

Furthermore, while very significant, the data regarding the price differences in search results and product recommendations could be improved upon. Both paper and HDMI cords are fairly low cost products in their category so it is hard to even think of lower priced products that would show up in a search for either one. These products positioning within their submarkets may be skewing the data. To see if this kind of targeting is occurring overall, looking at a large basket of items and doing the same analysis could potentially isolate for the market position of each individual product and give a more robust suite of results.

5. Conclusion

Using data gathered specifically for this analysis, we can conclude that there is no evidence of online marketplaces, linked to physical stores or otherwise, targeting IP addresses with different prices. In fact, no variation at all was observed in listed price. Beyond that

however, this paper did determine that E-Bay may change the search results and product recommendations seen by those in remote areas, likely do to the high impact shipping costs can have on the small firms or individuals who use E-Bay as their marketplace. While there is not have enough data to determine if this kind of search targeting is widespread, it seems to exist frequently outside of the lower 48 states.

Finally, by looking at differences between the prices of the products we were looking for and the other search results and product recommendations found when looking for those products, this paper found evidence that E-Bay attempts to show search results and recommend products with higher prices than the product being sought. While there are potential confounding factors, further research in this area could conclude whether or not search targeting is used to encourage customers to purchase higher value products.

Works Cited

Cavallo, Alberto F. *Are Online and Offline Prices Similar*? *Evidence from Large Multi-Channel Retailers*. National Bureau of Economic Research, Mar. 2016. Web. Apr. 2017.

Clemons, Eric K., Il-Horn Hann, and Lorin M. Hitt. "Price Dispersion and Differentiation in Online Travel: An Empirical Investigation."*Management Science*. INFORMS, 1 Apr. 2002, Web. 25 Apr. 2016.

Larson, Jeff, Surya Mattu, and Julia Angwin. "Unintended Consequences of Geographic Targeting." ProPublica (2016): n. pag. Web.

Toubiana, Vincent, Arvind Narayanan, Dan Boneh, Helen Nissenbaum, and Solon Barocas. "Adnostic: Privacy Preserving Targeted Advertising." Proceedings Network and Distributed System Symposium, Mar. 2010. Web.

Viswanathan, Siva, Jason Kuruzovich, Sanjay Gosain, and Ritu Agarwal. "Online Infomediaries and Price Discrimination: Evidence from the Automotive Retailing Sector." *AMA Journals*. American Marketing Association, Jul. 2007. Web. 25 Apr. 2016.

Yang, Wan-Shiou, Jia-Ben Dia, Hung-Chi Cheng, and Hsing-Tzu Lin. "Mining Social Networks for Targeted Advertising." Hawaii International Conference on System Sciences, Jan. 2006. Web. 25 Apr. 2016.

6. Appendix – Python Scripts

6.1 iPad Data Gathering

```
import webbrowser as web

##Products are iPad Air 2 128 GB Space Gray, 500 Sheet ream of paper, 6 foot HDMI
cord

##iPad Air 2 128 GB Space Gray

##Pure Online Retailers
##Amazon
web.open_new('https://www.amazon.com/s/ref=nb_sb_noss_2?url=search-
alias%3Daps&field-keywords=ipad+air+2+128+GB&rh=i%3Aaps%2Ck%3Aipad+air+2+128+GB')
web.open_new_tab('https://www.amazon.com/Apple-MGTX2LL-9-7-inches-Tablet-
Space/dp/B00OTWQXSI/ref=sr_1_2?ie=UTF8&qid=1491934470&sr=8-
2&keywords=ipad+air+2+128+GB')
##Overstock
web.open_new_tab('https://www.overstock.com/search?keywords=ipad+air+2+128gb&taxon
omy=sto2&ralt=sto22,sto40,sto7&TID=AR:TRUE&searchtype=Header')
web.open_new_tab('https://www.overstock.com/Electronics/Apple-iPad-Air-2-64GB-
Gold-with-Accessories-
Bundle/10747055/product.html?refccid=JKBZJJFYEJ6ZGV2ELDGXGL7LPQ&searchidx=1')
##E-Bay
web.open_new_tab('http://www.ebay.com/sch/i.html?_from=R40&_trksid=p2050601.m570.l
1313.TR12.TRC2.A0.H0.Xipad+air+2+128gb.TRS0&_nkw=ipad+air+2+128gb&_sacat=0')
web.open_new_tab('http://www.ebay.com/itm/Apple-iPad-Air-2-128GB-Wi-Fi-9-7in-
Space-Gray-/292084446753?hash=item440196f621:g:mN8AAOSwc49Y7tn5')
##Store Websites
##Wal-Mart
web.open_new_tab('https://www.walmart.com/search/?query=iPad%20air%202%20128%20GB'
)
web.open_new_tab('https://www.walmart.com/ip/Apple-iPad-Air-2-MGTX2LL-A-9-7-
inches-128-GB-Tablet-Space-Gray/160348958')

web.open_new_tab('http://geoip.hidemyass.com/')
```

6.2 HDMI Data Gathering

```
import webbrowser as web

##6-foot HDMI

##Pure Online Retailers
##Amazon
web.open_new('https://www.amazon.com/s/ref=nb_sb_noss_2?url=search-
alias%3Daps&field-keywords=6+foot+hdmi+cord')
web.open_new_tab('https://www.amazon.com/AmazonBasics-Rated-Wall-Installation-
Cable/dp/B014I8TIV6/ref=sr_1_1?ie=UTF8&qid=1492466602&sr=8-
1&keywords=6+foot+hdmi+cord')
##Overstock
web.open_new_tab('https://www.overstock.com/search?keywords=6+foot+HDMI+cable&Sear
chType=Header')
web.open_new_tab('https://www.overstock.com/Electronics/INSTEN-AccStation-6-foot-
High-Speed-M-M-HDMI-
Cable/6370343/product.html?refccid=O5MABO552ZMIOBSILC2NF55X4A&searchidx=3')
```

```
##E-Bay
web.open_new_tab('http://www.ebay.com/sch/i.html?_from=R40&_trksid=p2050601.m570.l
1313.TR0.TRC0.H0.X6+foot+hdmi+cor.TRS0&_nkw=6+foot+hdmi+cord&_sacat=0')
web.open_new_tab('http://www.ebay.com/itm/6FT-HDMI-CABLE-CORD-CONNECT-TV-TO-
LAPTOP-COMPUTER-PC-DVD-BLU-RAY-MEDIA-PLAYER-
/122449944717?hash=item1c8295e08d:g:q28AAOSwcdRY8-CQ')
##Store Websites
##Wal-Mart
web.open_new_tab('https://www.walmart.com/search/?query=6%20foot%20hdmi%20cord')
web.open_new_tab('https://www.walmart.com/ip/FosPower-xFFFD-4x1-HDMI-Switcher-4-
port-Splitter-with-IR-Remote-3D-4Kx2K-1080p-with-Picture-in-Picture-PiP-
Function/40654441')

web.open_new_tab('http://geoip.hidemyass.com/')
```

6.3 Paper Data Gathering

```
import webbrowser as web

##6-foot HDMI

##Pure Online Retailers
##Amazon
web.open_new('https://www.amazon.com/s/ref=nb_sb_noss_2?url=search-
alias%3Daps&field-keywords=6+foot+hdmi+cord')
web.open_new_tab('https://www.amazon.com/AmazonBasics-Rated-Wall-Installation-
Cable/dp/B014I8TIV6/ref=sr_1_1?ie=UTF8&qid=1492466602&sr=8-
1&keywords=6+foot+hdmi+cord')
##Overstock
web.open_new_tab('https://www.overstock.com/search?keywords=6+foot+HDMI+cable&Sear
chType=Header')
web.open_new_tab('https://www.overstock.com/Electronics/INSTEN-AccStation-6-foot-
High-Speed-M-M-HDMI-
Cable/6370343/product.html?refccid=O5MABO552ZMIOBSILC2NF55X4A&searchidx=3')
##E-Bay
web.open_new_tab('http://www.ebay.com/sch/i.html?_from=R40&_trksid=p2050601.m570.l
1313.TR0.TRC0.H0.X6+foot+hdmi+cor.TRS0&_nkw=6+foot+hdmi+cord&_sacat=0')
web.open_new_tab('http://www.ebay.com/itm/6FT-HDMI-CABLE-CORD-CONNECT-TV-TO-
LAPTOP-COMPUTER-PC-DVD-BLU-RAY-MEDIA-PLAYER-
/122449944717?hash=item1c8295e08d:g:q28AAOSwcdRY8-CQ')
##Store Websites
##Wal-Mart
web.open_new_tab('https://www.walmart.com/search/?query=6%20foot%20hdmi%20cord')
web.open_new_tab('https://www.walmart.com/ip/FosPower-xFFFD-4x1-HDMI-Switcher-4-
port-Splitter-with-IR-Remote-3D-4Kx2K-1080p-with-Picture-in-Picture-PiP-
Function/40654441')

web.open_new_tab('http://geoip.hidemyass.com/')
```

6.4 iPad OLS

```
##iPad-Amazon

import pandas as pd
import statsmodels
import statsmodels.formula.api as sm
dt = pd.DataFrame({"Price": [444.99, 444.99, 444.99, 444.99, 444.99, 444.99,
444.99, 444.99, 444.99, 444.99, 444.99, 444.99, 444.99, 444.99, 444.99, 444.99,
444.99, 444.99, 444.99, 444.99, 444.99, 444.99, 444.99, 444.99, 444.99, 444.99,
444.99, 444.99, 444.99, 444.99, 444.99, 444.99, 444.99, 444.99, 444.99, 444.99,
444.99, 444.99, 444.99, 444.99],
    "loc1": [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc2': [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc3': [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc4': [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc5': [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc6': [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc7': [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc8': [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc9': [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc10': [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc11': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc12': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc13': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc14': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
    'loc15': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
    'loc16': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
    'loc17': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
    'loc18': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
    'loc19': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
    'loc20': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]})
result = sm.ols(formula="Price ~ loc1 + loc2 + loc3 + loc4 + loc5 \
+ loc6 + loc7 + loc8 + loc9 + loc10 + loc11 + loc12 + loc13 + loc14\
+ loc15 + loc16 + loc17 + loc18 + loc19 + loc20", data=dt).fit()

print(result.summary())

##iPad-Overstock
```

```
dt2 = pd.DataFrame({"Price": [664.62, 664.62, 664.62, 664.62, 664.62, 664.62,
664.62, 664.62, 664.62, 664.62, 664.62, 664.62, 664.62, 664.62, 664.62, 664.62,
664.62, 664.62, 664.62, 664.62, 664.62, 664.62, 664.62, 664.62, 664.62, 664.62,
664.62, 664.62, 664.62, 664.62, 664.62, 664.62, 664.62, 664.62, 664.62, 664.62,
664.62, 664.62, 664.62, 664.62],
    "loc1": [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc2': [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc3': [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc4': [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc5': [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc6': [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc7': [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc8': [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc9': [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc10': [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc11': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc12': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc13': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc14': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
    'loc15': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
    'loc16': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
    'loc17': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
    'loc18': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
    'loc19': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
    'loc20': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0]})
result = sm.ols(formula="Price ~ loc1 + loc2 + loc3 + loc4 + loc5 \
+ loc6 + loc7 + loc8 + loc9 + loc10 + loc11 + loc12 + loc13 + loc14\
+ loc15 + loc16 + loc17 + loc18 + loc19 + loc20", data=dt2).fit()

print(result.summary())

##iPad-E-Bay

dt3 = pd.DataFrame({"Price": [439, 439, 439, 439, 439, 439, 439, 439, 439, 439,
439, 439, 439, 439, 439, 439, 439, 439, 439, 439, 439, 439, 439, 439, 439, 439,
439, 439, 439, 439, 439, 439, 439, 439, 439, 439, 439, 439, 439, 439],
    "loc1": [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc2': [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
```

```
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc3': [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc4': [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc5': [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc6': [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc7': [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc8': [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc9': [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc10': [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc11': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc12': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc13': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
    'loc14': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
    'loc15': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
    'loc16': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
    'loc17': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
    'loc18': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
    'loc19': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0],
    'loc20': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]})
result = sm.ols(formula="Price ~ loc1 + loc2 + loc3 + loc4 + loc5 \
+ loc6 + loc7 + loc8 + loc9 + loc10 + loc11 + loc12 + loc13 + loc14\
+ loc15 + loc16 + loc17 + loc18 + loc19 + loc20", data=dt3).fit()

print(result.summary())

##iPad-Wal-Mart

dt4 = pd.DataFrame({"Price": [479.95, 479.95, 479.95, 479.95, 479.95, 479.95,
479.95, 479.95, 479.95, 479.95, 479.95, 479.95, 479.95, 479.95, 479.95, 479.95,
479.95, 479.95, 479.95, 479.95, 479.95, 479.95, 479.95, 479.95, 479.95, 479.95,
479.95, 479.95, 479.95, 479.95, 479.95, 479.95, 479.95, 479.95, 479.95, 479.95,
479.95, 479.95, 479.95, 479.95],
    "loc1": [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc2': [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc3': [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc4': [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc5': [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
```

```
    'loc6': [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc7': [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc8': [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc9': [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc10': [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc11': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc12': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
    'loc13': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
    'loc14': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
    'loc15': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
    'loc16': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
    'loc17': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
    'loc18': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0],
    'loc19': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
    'loc20': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]})
result = sm.ols(formula="Price ~ loc1 + loc2 + loc3 + loc4 + loc5 \
+ loc6 + loc7 + loc8 + loc9 + loc10 + loc11 + loc12 + loc13 + loc14\
+ loc15 + loc16 + loc17 + loc18 + loc19 + loc20", data=dt4).fit()

print(result.summary())
```

6.5 HDMI OLS

```
##HDMI-Amazon

import pandas as pd
import statsmodels
import statsmodels.formula.api as sm
dt = pd.DataFrame({"Price": [5.99, 5.99, 5.99, 5.99, 5.99, 5.99, 5.99, 5.99, 5.99,
5.99, 5.99, 5.99, 5.99, 5.99, 5.99, 5.99, 5.99, 5.99, 5.99, 5.99, 5.99, 5.99,
5.99, 5.99, 5.99, 5.99, 5.99, 5.99, 5.99, 5.99, 5.99, 5.99, 5.99, 5.99, 5.99,
5.99, 5.99, 5.99, 5.99, 5.99],
    "loc1": [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc2': [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc3': [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc4': [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc5': [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc6': [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc7': [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc8': [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc9': [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc10': [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc11': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc12': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc13': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
    'loc14': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
    'loc15': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
    'loc16': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
    'loc17': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
    'loc18': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
    'loc19': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0],
    'loc20': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]})
result = sm.ols(formula="Price ~ loc1 + loc2 + loc3 + loc4 + loc5 \
+ loc6 + loc7 + loc8 + loc9 + loc10 + loc11 + loc12 + loc13 + loc14\
+ loc15 + loc16 + loc17 + loc18 + loc19 + loc20", data=dt).fit()
print(result.params)
print(result.summary())

##HDMI-Overstock

dt2 = pd.DataFrame({"Price": [4.49, 4.49, 4.49, 4.49, 4.49, 4.49, 4.49, 4.49,
4.49, 4.49, 4.49, 4.49, 4.49, 4.49, 4.49, 4.49, 4.49, 4.49, 4.49, 4.49, 4.49,
4.49, 4.49, 4.49, 4.49, 4.49, 4.49, 4.49, 4.49, 4.49, 4.49, 4.49, 4.49, 4.49,
4.49, 4.49, 4.49, 4.49, 4.49, 4.49],
    "loc1": [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc2': [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc3': [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc4': [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc5': [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc6': [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc7': [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc8': [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc9': [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc10': [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc11': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc12': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc13': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
    'loc14': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
    'loc15': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
    'loc16': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
    'loc17': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
    'loc18': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
    'loc19': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0],
    'loc20': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]})
result = sm.ols(formula="Price ~ loc1 + loc2 + loc3 + loc4 + loc5 \
+ loc6 + loc7 + loc8 + loc9 + loc10 + loc11 + loc12 + loc13 + loc14\
+ loc15 + loc16 + loc17 + loc18 + loc19 + loc20", data=dt2).fit()
print(result.params)
print(result.summary())

##HDMI-E-Bay

dt3 = pd.DataFrame({"Price": [4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4],
    "loc1": [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc2': [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc3': [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc4': [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc5': [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc6': [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc7': [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc8': [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc9': [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc10': [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc11': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc12': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc13': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
    'loc14': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
    'loc15': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
```

```
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
    'loc16': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
    'loc17': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
    'loc18': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
    'loc19': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0],
    'loc20': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]})
result = sm.ols(formula="Price ~ loc1 + loc2 + loc3 + loc4 + loc5 \
+ loc6 + loc7 + loc8 + loc9 + loc10 + loc11 + loc12 + loc13 + loc14\
+ loc15 + loc16 + loc17 + loc18 + loc19 + loc20", data=dt3).fit()
print(result.params)
print(result.summary())

##HDMI-Wal-Mart

dt4 = pd.DataFrame({"Price": [4.99, 4.99, 4.99, 4.99, 4.99, 4.99, 4.99, 4.99,
4.99, 4.99, 4.99, 4.99, 4.99, 4.99, 4.99, 4.99, 4.99, 4.99, 4.99, 4.99, 4.99,
4.99, 4.99, 4.99, 4.99, 4.99, 4.99, 4.99, 4.99, 4.99, 4.99, 4.99, 4.99, 4.99,
4.99, 4.99, 4.99, 4.99, 4.99, 4.99],
    "loc1": [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc2': [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc3': [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc4': [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc5': [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc6': [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc7': [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc8': [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc9': [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc10': [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc11': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc12': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc13': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc14': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
    'loc15': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
    'loc16': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
    'loc17': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
    'loc18': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
    'loc19': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
    'loc20': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0]})
result = sm.ols(formula="Price ~ loc1 + loc2 + loc3 + loc4 + loc5 \
+ loc6 + loc7 + loc8 + loc9 + loc10 + loc11 + loc12 + loc13 + loc14\
+ loc15 + loc16 + loc17 + loc18 + loc19 + loc20", data=dt4).fit()
print(result.params)
print(result.summary())
```

```
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0],
    'loc20': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]})
result = sm.ols(formula="Price ~ loc1 + loc2 + loc3 + loc4 + loc5 \
+ loc6 + loc7 + loc8 + loc9 + loc10 + loc11 + loc12 + loc13 + loc14\
+ loc15 + loc16 + loc17 + loc18 + loc19 + loc20", data=dt4).fit()
print(result.params)
print(result.summary())
```

6.6 Paper OLS

```
##Paper-Amazon

import pandas as pd
import statsmodels
import statsmodels.formula.api as sm
dt = pd.DataFrame({"Price": [7.99, 7.99, 7.99, 7.99, 7.99, 7.99, 7.99, 7.99, 7.99,
7.99, 7.99, 7.99, 7.99, 7.99, 7.99, 7.99, 7.99, 7.99, 7.99, 7.99, 7.99, 7.99,
7.99, 7.99, 7.99, 7.99, 7.99, 7.99, 7.99, 7.99, 7.99, 7.99, 7.99, 7.99, 7.99,
7.99, 7.99, 7.99, 7.99, 7.99],
    "loc1": [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc2': [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc3': [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc4': [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc5': [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc6': [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc7': [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc8': [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc9': [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc10': [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc11': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc12': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
    'loc13': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
    'loc14': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
    'loc15': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
    'loc16': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
    'loc17': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
    'loc18': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0],
    'loc19': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
```

```
    'loc20': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]})
result = sm.ols(formula="Price ~ loc1 + loc2 + loc3 + loc4 + loc5 \
+ loc6 + loc7 + loc8 + loc9 + loc10 + loc11 + loc12 + loc13 + loc14\
+ loc15 + loc16 + loc17 + loc18 + loc19 + loc20", data=dt).fit()
print(result.params)
print(result.summary())

##Paper-Overstock

dt2 = pd.DataFrame({"Price": [17.49, 17.49, 17.49, 17.49, 17.49, 17.49, 17.49,
17.49, 17.49, 17.49, 17.49, 17.49, 17.49, 17.49, 17.49, 17.49, 17.49, 17.49, 17.49,
17.49, 17.49, 17.49, 17.49, 17.49, 17.49, 17.49, 17.49, 17.49, 17.49, 17.49, 17.49,
17.49, 17.49, 17.49, 17.49, 17.49, 17.49, 17.49, 17.49, 17.49, 17.49, 17.49, 17.49],
    "loc1": [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc2': [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc3': [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc4': [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc5': [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc6': [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc7': [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc8': [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc9': [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc10': [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc11': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc12': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc13': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
    'loc14': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
    'loc15': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
    'loc16': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
    'loc17': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
    'loc18': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
    'loc19': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0],
    'loc20': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]})
result = sm.ols(formula="Price ~ loc1 + loc2 + loc3 + loc4 + loc5 \
+ loc6 + loc7 + loc8 + loc9 + loc10 + loc11 + loc12 + loc13 + loc14\
+ loc15 + loc16 + loc17 + loc18 + loc19 + loc20", data=dt2).fit()
print(result.params)
print(result.summary())
```

```
##Paper-E-Bay

dt3 = pd.DataFrame({"Price": [6.67, 6.67, 6.67, 6.67, 6.67, 6.67, 6.67, 6.67,
6.67, 6.67, 6.67, 6.67, 6.67, 6.67, 6.67, 6.67, 6.67, 6.67, 6.67, 6.67, 6.67,
6.67, 6.67, 6.67, 6.67, 6.67, 6.67, 6.67, 6.67, 6.67, 6.67, 6.67, 6.67, 6.67,
6.67, 6.67, 6.67, 6.67, 6.67, 6.67],
    "loc1": [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc2': [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc3': [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc4': [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc5': [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc6': [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc7': [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc8': [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc9': [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc10': [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc11': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc12': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc13': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc14': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
    'loc15': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
    'loc16': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
    'loc17': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
    'loc18': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
    'loc19': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
    'loc20': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0]})
result = sm.ols(formula="Price ~ loc1 + loc2 + loc3 + loc4 + loc5 \
+ loc6 + loc7 + loc8 + loc9 + loc10 + loc11 + loc12 + loc13 + loc14\
+ loc15 + loc16 + loc17 + loc18 + loc19 + loc20", data=dt3).fit()
print(result.params)
print(result.summary())

##Paper-Wal-Mart

dt4 = pd.DataFrame({"Price": [7.49, 7.49, 7.49, 7.49, 7.49, 7.49, 7.49, 7.49,
7.49, 7.49, 7.49, 7.49, 7.49, 7.49, 7.49, 7.49, 7.49, 7.49, 7.49, 7.49, 7.49,
7.49, 7.49, 7.49, 7.49, 7.49, 7.49, 7.49, 7.49, 7.49, 7.49, 7.49, 7.49, 7.49,
7.49, 7.49, 7.49, 7.49, 7.49, 7.49],
    "loc1": [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
```

```
    'loc2': [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc3': [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc4': [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc5': [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc6': [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc7': [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc8': [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc9': [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc10': [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc11': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc12': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
    'loc13': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
    'loc14': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
    'loc15': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
    'loc16': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
    'loc17': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
    'loc18': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
    'loc19': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0],
    'loc20': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]})
result = sm.ols(formula="Price ~ loc1 + loc2 + loc3 + loc4 + loc5 \
+ loc6 + loc7 + loc8 + loc9 + loc10 + loc11 + loc12 + loc13 + loc14\
+ loc15 + loc16 + loc17 + loc18 + loc19 + loc20", data=dt4).fit()
print(result.params)
print(result.summary())
```

6.7 Reccomended Product and Search Result Analysis

```
import pandas as pd
import statsmodels
import statsmodels.formula.api as sm
import numpy as n
import scipy.stats as sp

#E-Bay

#iPads

iPad_Price = 439
Suggested_iPads = [439, 310, 367.77, 379.99, 261.81, 370, 189.99, 249.99, 365,
200, 240, 260, 325, 257, 255]
```

```
[t_stat1, p1] = sp.ttest_1samp(Suggested_iPads, iPad_Price)

#HDMI

HDMI_Price = 4
Suggested_HDMI = [4, 2.14, 5.78, 5.99, 7.36, 18.71]
[t_stat2, p2] = sp.ttest_1samp(Suggested_HDMI, HDMI_Price)

#Paper

Paper_Price = 6.67
Suggested_Paper = [6.67, 20.02, 20.25, 20.25, 20.03, 16.39, 20.03, 21.22, 19.87,
20.27]
[t_stat3, p3] = sp.ttest_1samp(Suggested_Paper, Paper_Price)

#Indexed

iPad_factor = 100/n.mean(Suggested_iPads)
HDMI_factor = 100/n.mean(Suggested_HDMI)
Paper_factor = 100/n.mean(Suggested_Paper)

Indexed_Prices = [iPad_factor*iPad_Price, HDMI_factor*HDMI_Price,
Paper_factor*Paper_Price]
Suggested_Indexed = []
Suggested_Indexed = list(n.array(Suggested_iPads)*iPad_factor) +
list(n.array(Suggested_HDMI)*HDMI_factor) +
list(n.array(Suggested_Paper)*Paper_factor)

[t_stat4, p4] = sp.ttest_1samp(Suggested_Indexed, n.mean(Indexed_Prices))

data = pd.DataFrame({'Analysis': ['iPad', 'HDMI', 'Paper', 'Indexed'], 'T-
Statistic': [t_stat1, t_stat2, t_stat3,
                    t_stat4], 'P-Values': [p1, p2, p3, p4]})

print('\n')
print(data)

#Searches

#iPads

Suggested_iPads2 = [439, 450, 399.99, 294.99, 279.99, 395, 325, 415, 290, 589]
[t_stat5, p5] = sp.ttest_1samp(Suggested_iPads2, iPad_Price)

#HDMI

Suggested_HDMI2 = [4, 4.69, 3, 5.99, 2.95, 18.99]
[t_stat6, p6] = sp.ttest_1samp(Suggested_HDMI2, HDMI_Price)

#Paper

Suggested_Paper2 = [9.79, 7.99, 11.35, 7.01, 14.49]
[t_stat7, p7] = sp.ttest_1samp(Suggested_Paper2, Paper_Price)

#Indexed

iPad_factor2 = 100/n.mean(Suggested_iPads2)
HDMI_factor2 = 100/n.mean(Suggested_HDMI2)
Paper_factor2 = 100/n.mean(Suggested_Paper2)
```

```
Indexed_Prices2 = [iPad_factor2*iPad_Price, HDMI_factor2*HDMI_Price,
Paper_factor2*Paper_Price]
Suggested_Indexed2 = []
Suggested_Indexed2 = list(n.array(Suggested_iPads2)*iPad_factor2) +
list(n.array(Suggested_HDMI2)*HDMI_factor2) +
list(n.array(Suggested_Paper2)*Paper_factor2)

[t_stat8, p8] = sp.ttest_1samp(Suggested_Indexed2, n.mean(Indexed_Prices2))

data2 = pd.DataFrame({'Analysis': ['iPad', 'HDMI', 'Paper', 'Indexed'], 'T-
Statistic': [t_stat5, t_stat6, t_stat7,
                    t_stat8], 'P-Values': [p5, p6, p7, p8]})

print('\n')
print(data2)

#Combined

#iPads

Suggested_iPads3 = Suggested_iPads + Suggested_iPads2
[t_stat9, p9] = sp.ttest_1samp(Suggested_iPads3, iPad_Price)

#HDMI

Suggested_HDMI3 = Suggested_HDMI + Suggested_HDMI2
[t_stat10, p10] = sp.ttest_1samp(Suggested_HDMI3, HDMI_Price)

#Paper

Suggested_Paper3 = Suggested_Paper + Suggested_Paper2
[t_stat11, p11] = sp.ttest_1samp(Suggested_Paper3, Paper_Price)

#Indexed

iPad_factor3 = 100/n.mean(Suggested_iPads3)
HDMI_factor3 = 100/n.mean(Suggested_HDMI3)
Paper_factor3 = 100/n.mean(Suggested_Paper3)

Indexed_Prices3 = [iPad_factor3*iPad_Price, HDMI_factor3*HDMI_Price,
Paper_factor3*Paper_Price]
Suggested_Indexed3 = []
Suggested_Indexed3 = list(n.array(Suggested_iPads2)*iPad_factor3) +
list(n.array(Suggested_HDMI3)*HDMI_factor2) +
list(n.array(Suggested_Paper3)*Paper_factor2)

[t_stat12, p12] = sp.ttest_1samp(Suggested_Indexed3, n.mean(Indexed_Prices3))

data3 = pd.DataFrame({'Analysis': ['iPad', 'HDMI', 'Paper', 'Indexed'], 'T-
Statistic': [t_stat9, t_stat10, t_stat11,
                    t_stat12], 'P-Values': [p9, p10, p11, p12]})

print('\n')
print(data3)
```