Project Report

Project by:

Akshay Mukundwar
Abhinav Rahate
Abhijeet Patil
Soham Wani

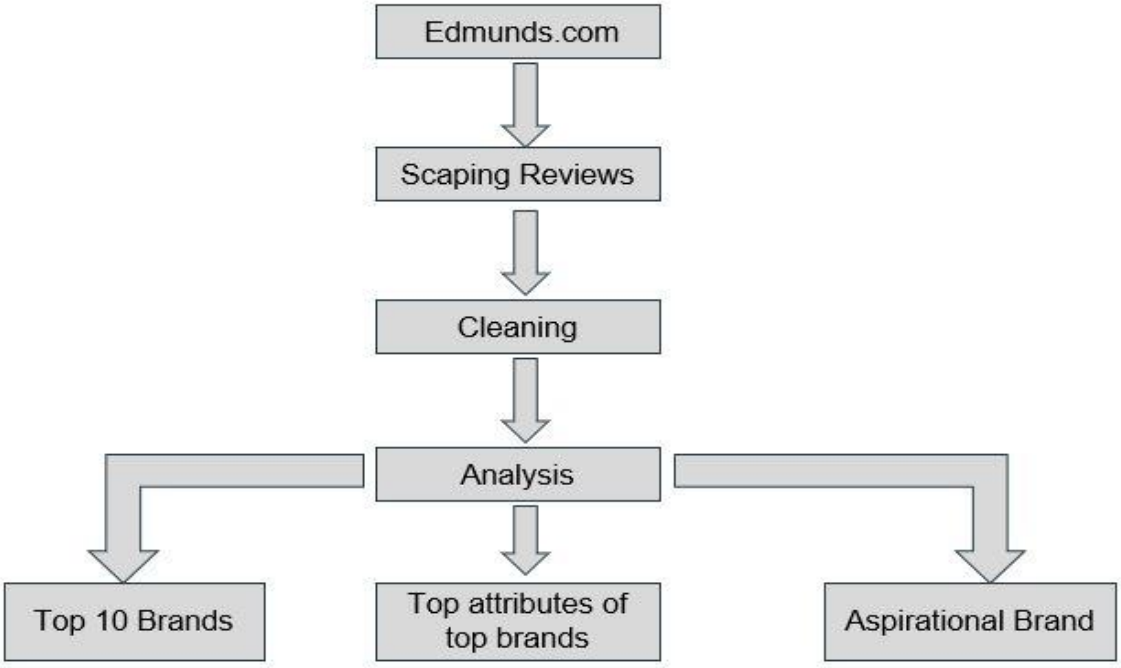
INTRODUCTION

- **CONTEXT**
  As people mostly use the manual process while looking for a car, and it's not the perfect way they want.
  1. People are not sure what to look, where to look and which one to select?
  2. Testing every car is not possible!
  3. Which car fits my budget?
  4. Unaware of specifics.

- **AIM AND OBJECTIVE**
  - To produce a system that can help the user to select/sort any particular car based on initial reviews of people.
  - To analyze and provide an option to the user based on his selection (Manufacturer/ Car Attributes).
  - To ease the customer's task whenever they want to buy a car.
  - Our Approach:
    - Know about website Edmuds.com
    - Web Scraping with the help of Selenium
    - Scraping the reviews from the website i.e (Edmuds.com)
    - Cleaning the Reviews

- **Analyzing**
  1. Top Attributes of Brands
  2. Top 10 Brands
  3. Aspirational Brands

OUR APPROACH



What is Edmunds.com?

- Edmunds.com Inc. (stylized as Edmunds) is an American online resource for automotive information. Edmunds give Car Shoppers Alerts and Advice to Help Them Maximize Their Savings.
- The Edmunds.com Web site includes prices for new and used vehicles, dealer and inventory listings, a database of national and regional incentives and rebates, vehicle test drive reviews, and tips and advice on all aspects of car purchases and ownership.
- Edmunds.com provides data through its "True Market Value" pricing tools, which launched in 2000. The Edmunds.com True Market Value New Vehicle Calculator displays the estimated average price consumers are paying when buying new vehicles.

- The Edmunds.com True Market Value Used Vehicle Appraiser estimates the actual transaction prices for used vehicles bought and sold by dealers and private parties.

Web Scraping Using Selenium—Python

Selenium: Selenium is a Web Browser Automation Tool Primarily, it is for automating web applications for testing purposes, but is certainly not limited to just that. It allows you to open a browser of your choice & perform tasks as a human being would, such as:

- Clicking buttons
- Entering information in forms
- Searching for specific information on the web pages

What is Web Scraping?

As the name suggests, this is a technique used for extracting data from websites. It is an automated process where an application processes the HTML of a Web Page to extract data for manipulation such as converting the Web page to another format and copying it into a local database or spreadsheet for later retrieval or analysis.

- How to navigate through multiple pages of a website and scrape large amounts of data using Selenium in Python

What is web-scraping?

Web scraping is a technique for extracting information from the internet automatically using software that simulates human web surfing.

How is web-scraping useful?

Web scraping helps us extract large volumes of data about customers, products, people, stock markets, etc. It is usually difficult to get this kind of information on a large scale using traditional data collection methods. We can utilize the data collected from a website such as e-commerce portal, social media channels to understand customer behaviors and sentiments, buying patterns, and brand attribute associations which are critical insights for any business.

Instructions regarding installations of packages.

a. Python version: We will be using Python 3.0, however, feel free to use Python 2.0 by making slight adjustments. We will be using a jupyter notebook, so you don't need any command line knowledge.

b. Selenium package: You can install selenium package using the following command

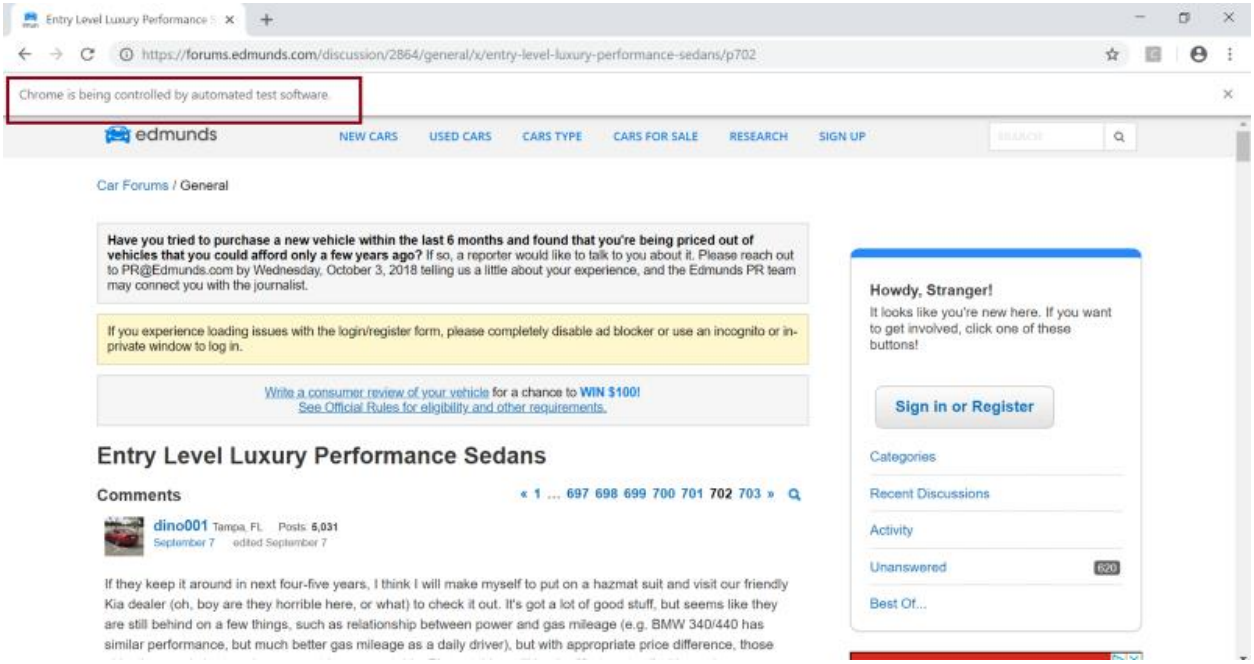
!pip install selenium

c. Chrome driver: Please install the latest version of chrome driver.

- The first and foremost thing while scraping a website is to understand the structure of the website. We will be scraping Edmunds.com, a car forum. This website aids people in their car buying decisions. People can post their reviews about different cars in the discussion forums (very similar to how one posts reviews on Amazon). We will be scraping the discussion about entry level luxury car brands.
- We will scrape ~5000 comments from different users across multiple pages. We will scrape user id, date of comment and comments and export it into a CSV file for any further analysis.
- We will first import important packages in our Notebook —
    - Importing packages
    - from selenium import webdriver
    - import pandas as pd
- Let's now create a new instance of google chrome. This will help our program open an URL in google chrome.

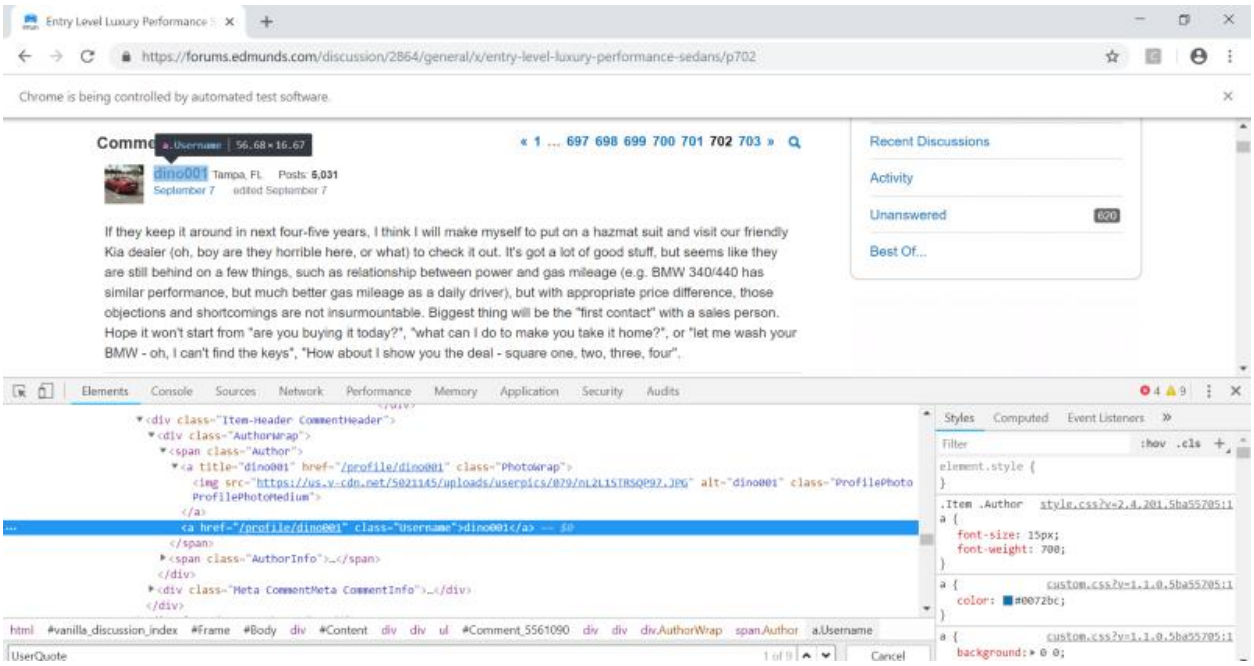
  driver = webdriver.Chrome('Path in your computer where you have installed chromedriver')
- Let's now access google chrome and open our website. By the way, chrome knows that you are accessing it through an automated software!

driver.get('https://forums.edmunds.com/discussion/2864/general/x/entry-level-luxury-performance-sedans/p702')



- We will inspect 3 items (user id, date, and comment) on our web page and understand how we can extract them.

1. **User id:** Inspecting the userid, we can see the highlighted text represents the XML code for user id.

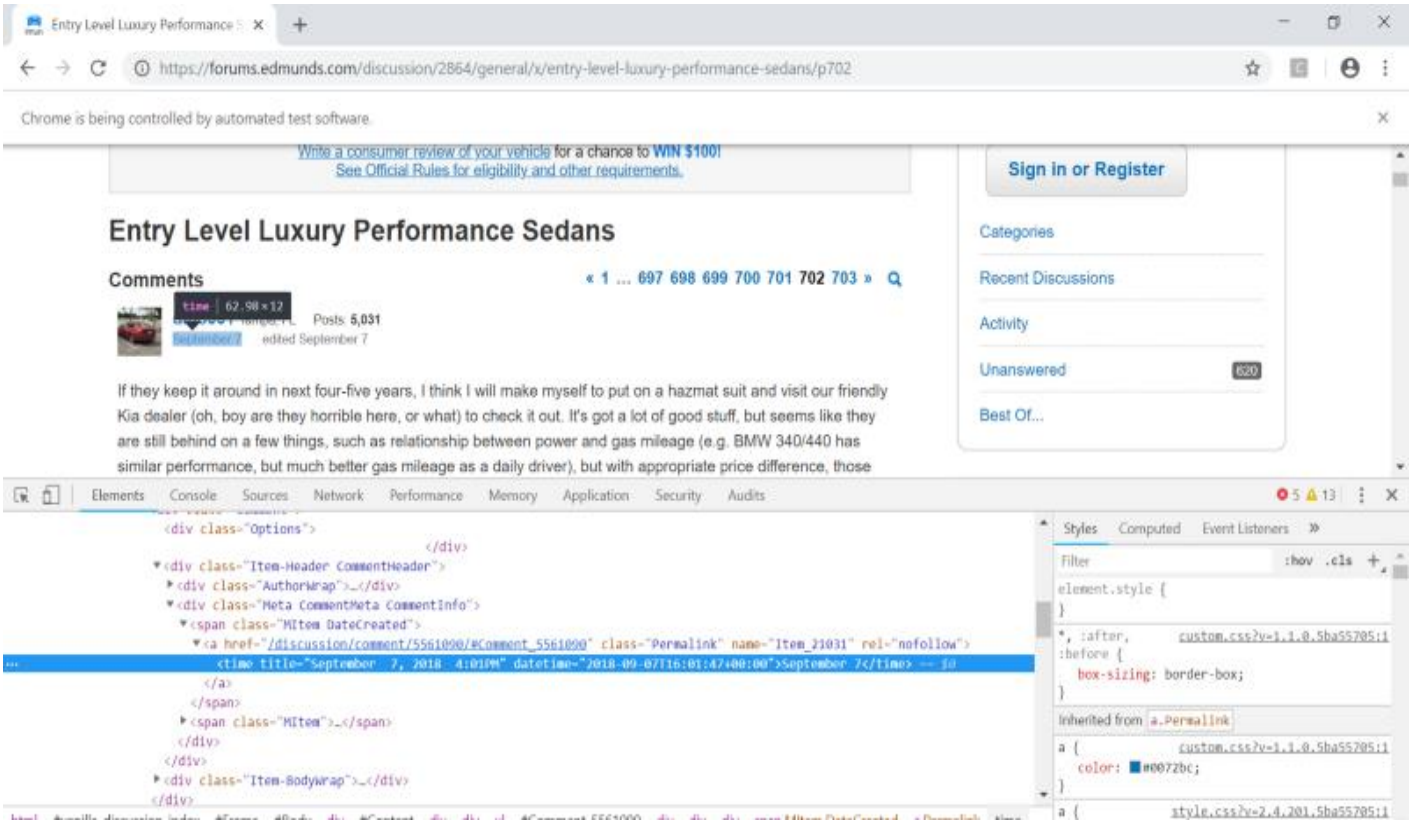


- The XML path (XPath)for the userid is shown below. There is an interesting thing to note here that the XML path contains a comment id, which uniquely denotes each comment on the website. This will be very helpful as we try to recursively scrape multiple comments.
  //*[@id=”Comment_5561090"]/div/div[2]/div[1]/span[1]/a[2]

- If we see the XPath in the picture, we will observe that it contains the user id ‘dino001’.

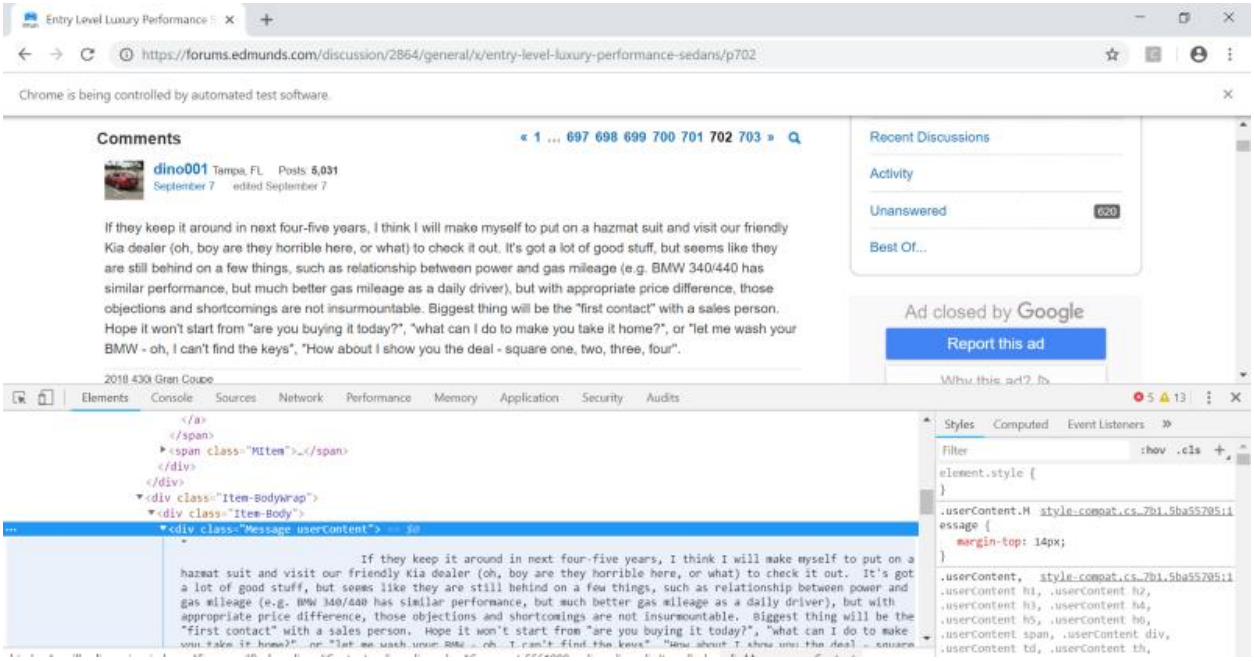
- **How do we extract the values inside an XPath?**
  Selenium has a function called “*find_elements_by_xpath*”. We will pass our XPath into this function and get a selenium element. Once we have the element, we can extract the text inside our XPath using the ‘*text*’ function. In our case, the text is basically the user id (‘dino001’).
- **userid_element = driver.find_elements_by_xpath('//*[@id="Comment_5561090"]/div/div[2]/div[1]/span[1]/a[2]')[0]**
- **userid = userid_element.text**

2. Comment Date: Similar to the user id, we will now inspect the date when the comment was posted.



- Let's also see the XPath for the comment date. Again note the unique comment id in the XPath.
- **//*[@id="Comment_5561090"]/div/div[2]/div[2]/span[1]/a/time**
- So, how do we extract the date from the above XPath?
- We will again use the function "find_elements_by_xpath" to get the selenium element. Now, if we carefully observe the highlighted text in the picture, we will see that the date is stored inside the 'title' attribute. We can access the values inside attributes using the function 'get_attribute'. We will pass the tag name in this function to get the value inside the same.
- **user_date = driver.find_elements_by_xpath('//*[@id="Comment_5561090"]/div/div[2]/div[2]/span[1]/a/time')[0]**
- **date = user_date.get_attribute('title')**

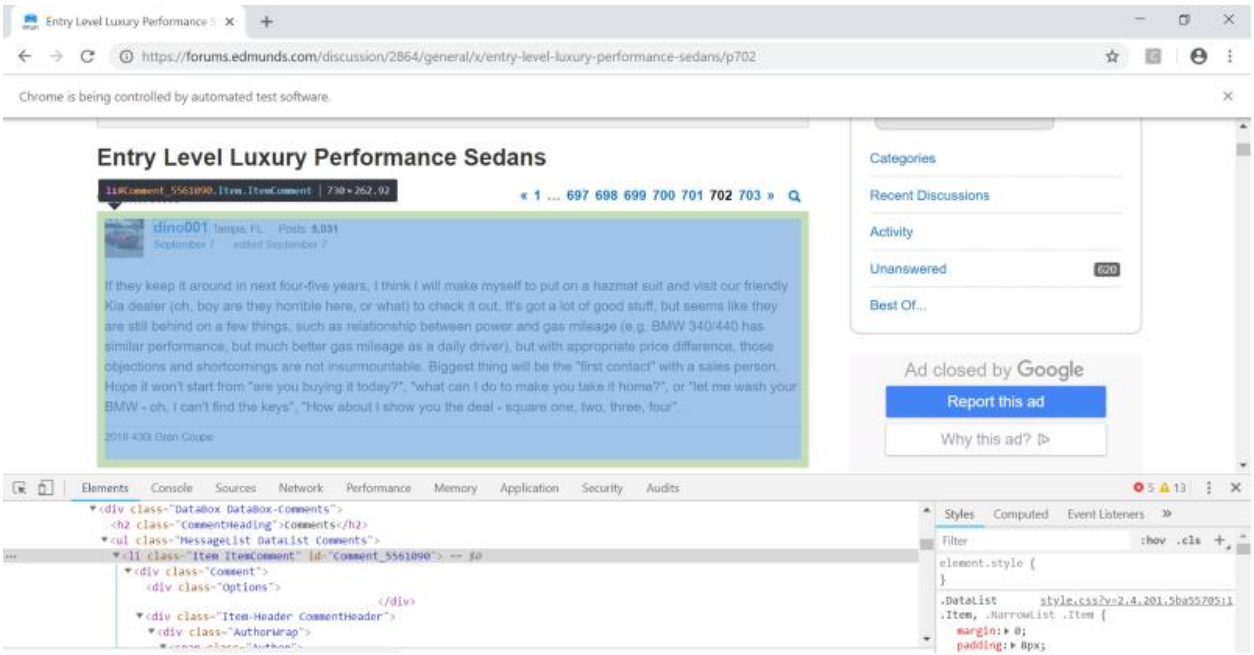
3. Comments: Lastly, let's explore how to extract the comments of each user.



- Below is the XPath for the user comment – //*[@id="Comment_5561090"]/div/div[3]/div/div[1]
- Once again, we have the comment id in our XPath. Similar to the userid we will extract the comment from the above XPath
- user_message = driver.find_elements_by_xpath('//*[@id="Comment_5561090"]/div/div[3]/div/div[1]')[0]
- comment = user_message.text

Now how to recursively extract these items for 5000 users?

- As discussed above, we will use the comment ids, which are unique for a comment to extract different users data. If we see the XPath for the entire comment block, we will see that it has a comment id associated with it. **//*[@id="Comment_5561090"]**



- The following code snippet will help us extract all the comment ids on a particular web page. We will again use the function '*find_elements_by_xpath*' on the above XPath and extract the ids from the '*id*' attribute.

  ids = driver.find_elements_by_xpath("//*[contains(@id,'Comment_')]")

  comment_ids = []

  for i in ids:

  comment_ids.append(i.get_attribute('id'))

- The above code gives us a list of all the comment ids from a particular web page.
- Now we will bring all the things we have seen so far into one big code, which will recursively help us extract 5000 comments. We can extract user ids, date and comments for each user on a particular web page by looping through all the comment ids we found in the previous code.
- Below is the code snippet to extract all comments from a particular web page.

driver = webdriver.Chrome('C:/Users/AbPa/chromedriver.exe')

driver.get('https://forums.edmunds.com/discussion/2864/general/x/entry-level-luxury-performance-sedans/p715')

```
#Creating empty data frame to store user_id, dates and comments from ~5K users.
comments = pd.DataFrame(columns = ['Date','user_id','comments'])

j = 715
while (j>=1):
   # Running while loop only till we get 5K comments
   if (len(comments)<5000):
      url = 'https://forums.edmunds.com/discussion/2864/general/x/entry-level-luxury-
performance-sedans/p' + str(j)
      driver.get(url)
      ids = driver.find_elements_by_xpath("//*[contains(@id,'Comment_')]")
      comment_ids = []
      for i in ids:
         comment_ids.append(i.get_attribute('id'))

      for x in comment_ids:
         #Extract dates from for each user on a page
         user_date = driver.find_elements_by_xpath('//*[@id="' + x
+'"]/div/div[2]/div[2]/span[1]/a/time')[0]
         date = user_date.get_attribute('title')

         #Extract user ids from each user on a page
         userid_element = driver.find_elements_by_xpath('//*[@id="' + x
+'"]/div/div[2]/div[1]/span[1]/a[2]')[0]
         userid = userid_element.text

         #Extract Message for each user on a page
         user_message = driver.find_elements_by_xpath('//*[@id="' + x
+'"]/div/div[3]/div/div[1]')[0]
         comment = user_message.text

         #Extracting Block Quote if Present
         block_quote = driver.find_element_by_xpath('//*[@id="' + x +
'"]/div/div[3]/div/div[1]')
         block_quote_class = block_quote.find_elements_by_class_name('UserQuote')
         block_text = ''
         if len(block_quote_class)>0:
            block_text = block_quote_class[0].text
```

```
        #Replacing block quotes
        comment = comment.replace(block_text,"")

       #Adding date, userid and comment for each user in a dataframe
        comments.loc[len(comments)] = [date,userid,comment]
    j=j-1
else:
    break
```

Cleaning Data

- Reviews cleaning is very important task, here we removed punctuation and converted reviews in lower case.
- As we are analyzing for brands, we have to convert all model names with brands name.
- Stop words
- Tokenization and Lemmatizing
- Counting frequency
- Words such as mileage, mpg, Gas, fuel, economy has been mapped to efficiency in order to find most associated attributes for a car

Analysis

Top 15 Brands:

We Analysis the top 15 Brands on the basis of Frequency count of particular brands come into the reviews

As it is clearly shown in the below table and Graph:

Index	Type	Size	Value
0	tuple	2	('bmw', 1305)
1	tuple	2	('audi', 1015)
2	tuple	2	('acura', 575)
3	tuple	2	('mercedes', 367)
4	tuple	2	('honda', 356)
5	tuple	2	('cadillac', 342)
6	tuple	2	('ford', 239)
7	tuple	2	('volkswagen', 227)
8	tuple	2	('infiniti', 213)
9	tuple	2	('hyundai', 199)
10	tuple	2	('lexus', 196)
11	tuple	2	('kia', 144)
12	tuple	2	('toyota', 127)
13	tuple	2	('volvo', 121)
14	tuple	2	('subaru', 112)

Calculate lift ratios for associations between brands: What is the lift ratio?

Consider the following example from a supermarket transactions database:

	Tea	Not tea	**Total**
Coffee	150	750	900
Not coffee	50	50	100
Total	200	800	1000

Let us now evaluate the association rule Tea => Coffee. The support of this rule is 100/1000 or 10%. The confidence of the rule is 150/200 or 75%. At first sight, this association rule seems very appealing given its high confidence. However, closer inspection reveals that the prior probability of buying coffee equals 900/1000 or 90%. Hence, a customer who buys tea is less likely to buy coffee than a customer about whom we have no information. The lift, also referred to as the interestingness measure, takes this into account by incorporating the prior probability of the rule consequent as follows:

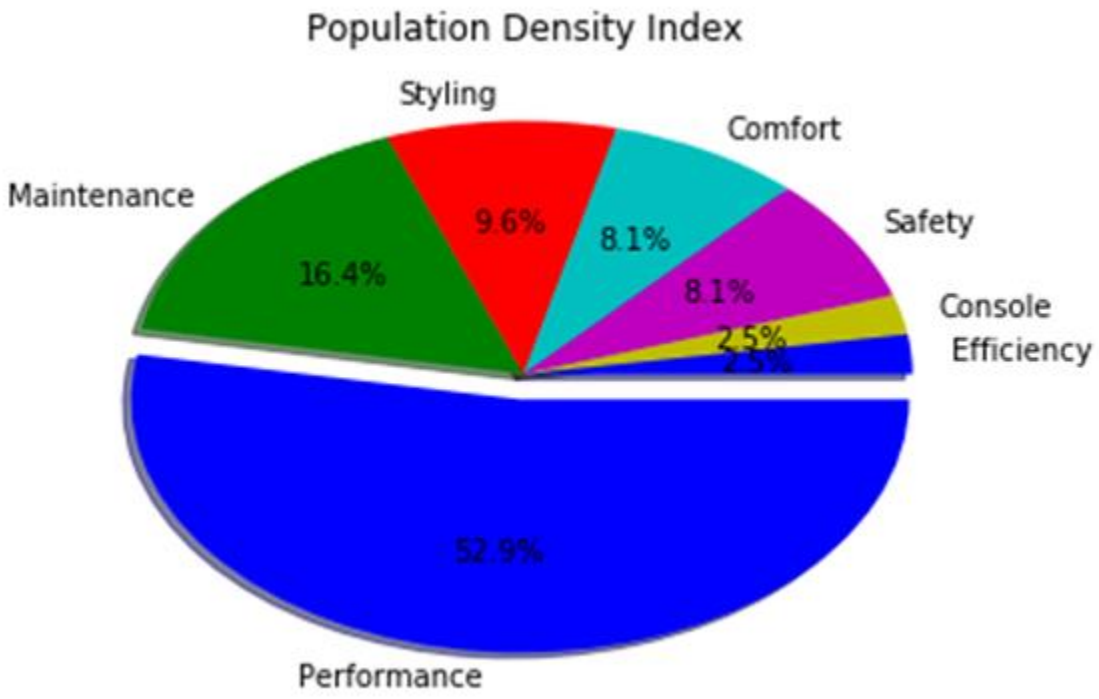
A lift value less (larger) than 1 indicates a negative (positive) dependence or substitution (complimentary) effect. In our example, the lift value equals 0.89, which clearly indicates the expected substitution effect between coffee and tea.

So, from the above table, we can easily analyze the association between two brands

For example, BMW is highly associated with the Mercedes, so on the basis of this analysis, we can say that a person who is interested in BMW can also look into the Mercedes or we can refer Mercedes as an option with BMW.

It will help in making Marketing Strategies.

Top attributes of top brands



With the help of the above pie chart, most of the people are interested in the performance of the car.

Calculate lift ratios for associations between top 5 attributes

Index	performance	maintenance	styling	comfort	safety
bmw	1.40337	1.40167	1.52719	1.45737	1.23873
audi	1.54587	1.61386	1.97249	1.79924	1.67594
acura	1.47303	1.47189	2.16826	1.99207	1.56188
mercedes	1.34116	1.21504	1.73576	2.14944	1.12278
honda	1.51594	1.40596	1.9939	1.86678	1.81041

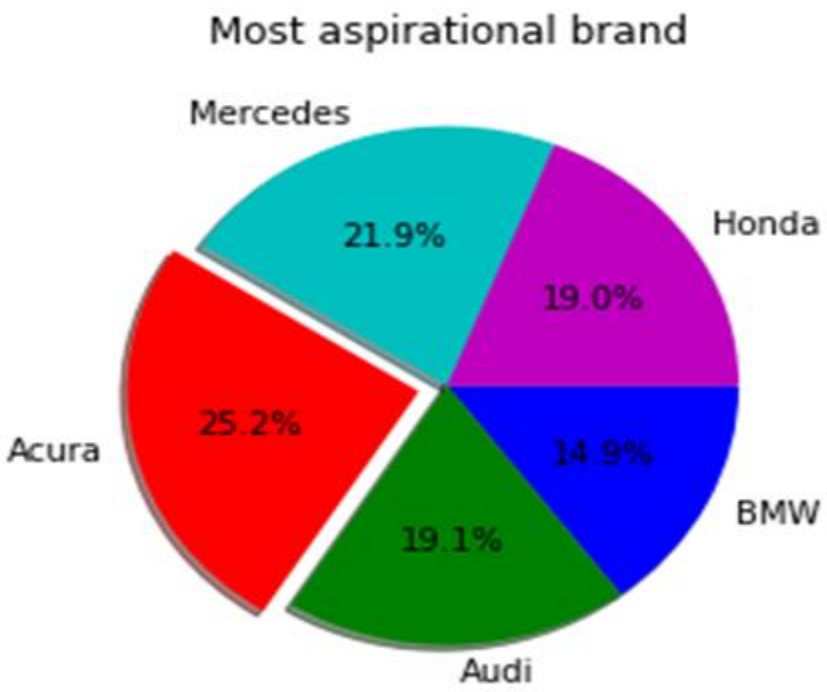
From the above table, we can easily find out the association between the top 5 attributes

For example:

Mercedes is mostly known for its comfort level so we can clearly see from the above table it is highly associated we comfort attribute but on the other side if we see Mercedes car not safe because it is not highly associated with safety attribute.

It is very helpful to the manufacturing department to more focus on the safety of a car.

Aspirational Brand:



From the above pie chart, we can find out the most aspirational brand, and the winner is Acura with 25.2%.

Lift Ratio:

Index	aspiration
bmw	1.44529
audi	1.85824
acura	2.41844
mercedes	2.1341
honda	1.88575

Observation

While BMW has claimed that they are the "ultimate driving machine", is that how people feel on Edmunds?

The association of BMW with performance is lower than that of Audi or Acura.

This shows that probably people don't associate BMW with a performance that much.

What is interesting is that people associate BMW more with comfort and styling than performance.

Whereas,

Acura gives comparatively the best result with respect to every attribute.

Hence, Acura could be the most preferred or recommended brand for any person.

Deployment

FLASK:

Flask is a lightweight WSGI web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. It began as a simple wrapper around Werkzeug and Jinja and has become one of the most popular Python web application frameworks.
Flask offers suggestions but doesn't enforce any dependencies or project layout. It is up to the developer to choose the tools and libraries they want to use. There are many extensions provided by the community that makes adding new functionality easy.

Conclusions

Production Manager:

a.) **BMW**: While we are marketing BMW as the 'ultimate driving machine' people are not strongly associating BMW with performance. Thus, think about improving the quality of engine/materials and maybe see what are the key performance features people are talking about and potentially try to include them in our cars.

b.) **Mercedes**: While Mercedes is doing a great job at comfort, people think that it is not safer to drive Mercedes. From a product's perspective, we could consider including additional high-quality safety features to try to alleviate any concerns with the same.

Marketing:

a.) **Audi:** For all 5 attributes, it seems like Audi is very closely related to Acura and Honda, and from a marketing perspective, we would want to create more brand differentiation to really uniquely position Audi in the luxury car marketplace.

b.) **Honda:** For all 5 attributes, it seems like Honda is very closely related to Acura, and from a marketing perspective, we would want to create more brand differentiation to really uniquely position Honda in the luxury car marketplace.

c.) **Mercedes**: This brand ranks the highest in terms of comfort, but scored the lowest in terms of safety. From a marketing and branding perspective, they are doing a good job of highlighting the comfort and styling of the car but may want to consider including the vehicle's safety aspects to having a more well-rounded brand image.