Web scrapping cars24.com

INDEX

Web scrapping cars24.com

Code Implementation

- 5.1 Setup and Initialization
- 5.2 Web Scraping Logic
- 5.3 Data Extraction
- 5.4 Data Storage
- 5.5 Error Handling and Logging
- 5.6 Legal and Ethical Considerations

Results

- 6.1 Summary of Data Collected
- 6.2 Data Structure and Format
- 6.3 Analysis and Insights

Conclusion

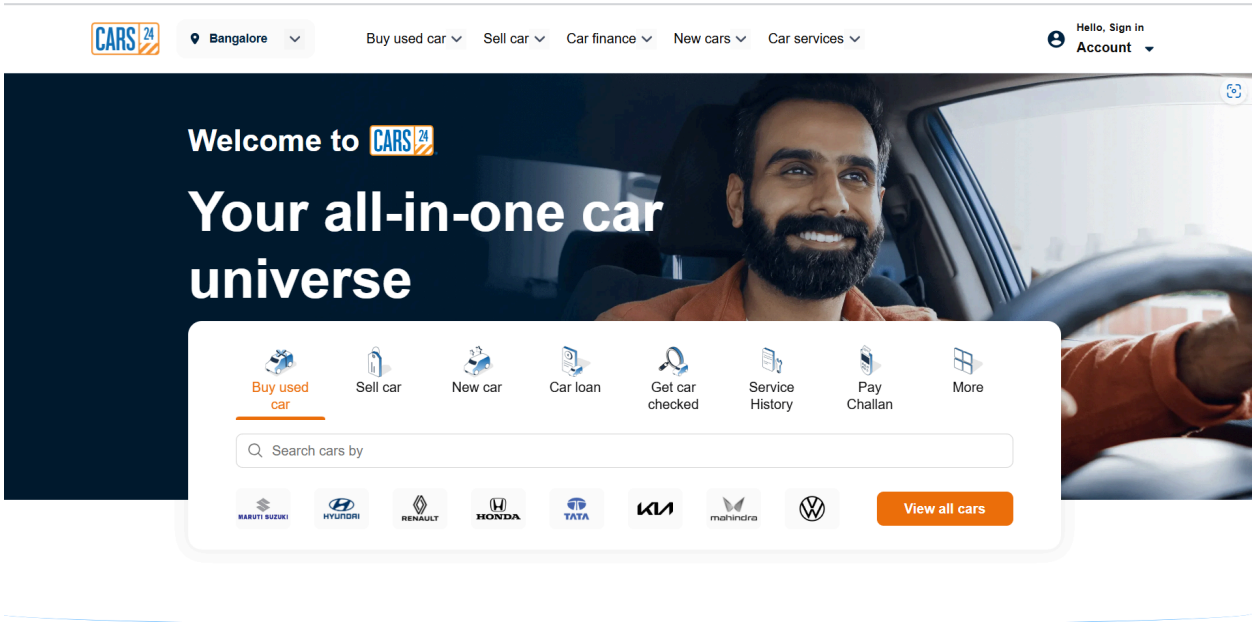
- 7.1 Key Findings
- 7.2 Challenges Faced
- 7.3 Future Work

References

- 8.1 Documentation and Guides
- 8.2 Libraries and Tools Documentation
- 8.3 Legal and Ethical Guidelines

Web scrapping cars24.com

project web scraping cars24 using selenium



Introduction

Web scrapping :

Web scraping is the process of automatically extracting data from websites. It involves accessing web pages, parsing the HTML content, and retrieving specific pieces of information. This data can then be stored, analyzed, or used for various purposes like building databases, generating reports, or feeding into machine learning models.

Key Concepts in Web Scraping:

1. **HTML Parsing**: Websites are typically built using HTML. Web scrapers parse the HTML code to extract the desired data.

2. **Libraries and Tools**: There are several popular libraries and tools for web scraping, such as:

Web scrapping cars24.com

- **BeautifulSoup**: A Python library that makes it easy to navigate and search through HTML and XML documents.
- **Scrapy**: An open-source web crawling framework written in Python, which provides tools to scrape websites and process the data.
- **Selenium**: A tool used to automate web browsers, often used when websites use JavaScript to load content dynamically.
- **Puppeteer**: A Node.js library that provides a high-level API to control Chrome or Chromium over the DevTools Protocol.

3. **HTTP Requests**: Web scraping tools typically make HTTP requests to fetch the raw HTML content of web pages. Libraries like `requests` in Python or `axios` in JavaScript are often used.

4. **Data Extraction**: After parsing the HTML, you extract specific elements, such as tables, lists, images, or text. This involves identifying HTML tags, classes, IDs, and other attributes.

5. **Handling Dynamic Content**: Some websites load content dynamically using JavaScript. In such cases, tools like Selenium or Puppeteer, which can interact with the browser, are used to capture the content after it has been rendered.

6. **Ethics and Legal Considerations**: Not all websites allow web scraping. It's essential to check the site's `robots.txt` file to see if scraping is permitted. Additionally, scraping too frequently or scraping sensitive data can lead to legal issues or getting your IP blocked.

7. **Storage and Processing**: Once the data is extracted, it can be stored in various formats like CSV, JSON, or databases for further analysis.

Web scrapping cars24.com

Objective of the project

The objective of web scraping Cars24.com is to automate the process of extracting specific data related to used car listings from the website. This includes details such as the car's name, model, price, year of manufacture, kilometers driven, fuel type, and transmission type. The extracted data can then be stored in a structured format (like CSV, JSON, or a database) for further analysis, reporting, or integration into other applications.

Data Collection:

- Automatically gather detailed information on used cars listed on Cars24.com.
- Extract relevant attributes such as year of manufacture,make of the car, car model, kilometers driven,owner, fuel type,transmission type,price,location.

Automation:

- Develop a script that can repeatedly perform the data extraction process, allowing for periodic updates or continuous monitoring of new car listings.
- Ensure the script can handle dynamic content and pagination effectively.

Data Storage:

- Store the extracted data in a structured and accessible format (e.g., CSV, JSON) for easy retrieval and analysis.
- Ensure data integrity and accuracy during the extraction process.

Legal Compliance:

- Adhere to the legal and ethical guidelines for web scraping, including respecting the website's `robots.txt` file and terms of service.
- Implement rate limiting to avoid overloading the server and prevent IP blocking.

Web scrapping cars24.com

Error Handling:

- Implement robust error handling to manage potential issues such as missing elements, network failures, or changes in the website's structure.

Scalability:

- Design the script to be scalable, allowing it to be adapted for other similar websites or expanded to include additional data points as needed.

Web scrapping cars24.com

Details Extracted from Cars24.com

Year :

When scraping data from Cars24.com, one of the key details you might want to extract is the year of manufacture of the cars listed. This data point is typically included in the car's listing and provides information on how old the vehicle is.

Locating the Year of Manufacture:

- **HTML Tags**: Identify the specific HTML tag or class that contains the year of manufacture. This might be within a , or another HTML element, usually accompanied by other details like the car model and variant.

- **CSS Selectors or XPath**: Use CSS selectors or XPath to precisely target the element containing the year. For example, the year might be located within a class attribute like “ year “.

Make :

When scraping data from Cars24.com, the make of the car is another crucial detail to extract. The "make" refers to the manufacturer or brand of the car, such as Toyota, Honda, Ford, etc. Extracting this information can help in categorizing the listings by brand, which is useful for various analyses.

1. Locating the Car Make:

- **HTML Tags**: The make of the car is typically displayed alongside or within the same element as the model. It could be in a <h2> ,<h3>, or tag, often together with the model in a combined string like "Toyota Corolla" or "Ford Mustang".
- **CSS Selectors or XPath:** Identify the correct element using CSS selectors or XPath to accurately extract the make from the combined string or from a separate element if available.

Web scrapping cars24.com

Model :

When scraping data from Cars24.com, another essential detail to extract is the **model** of the car. The car model indicates the specific version or variant of a car produced by a manufacturer, such as a Honda Civic, Ford Mustang, or Toyota Corolla. Here's how you can approach extracting the car model:

1. Locating the Car Model:

- **HTML Tags**: The car model is often displayed prominently on the listing page, typically within a specific <h2> , <h3> , or tag. The tag might have a class or id attribute that makes it identifiable, such as " model ".

- **CSS Selectors or XPath**: Use CSS selectors or XPath expressions to locate the exact element that contains the car model information.

Kilometers driven :

When scraping data from Cars24.com, another important detail to extract is the **kilometers driven** (or mileage) of the car. This metric indicates the total distance the car has traveled, which is a key factor for potential buyers as it gives an idea of the car's wear and tear.

1. Locating the Kilometers Driven:

- **HTML Tags**: The mileage is usually displayed prominently in the car listing, often alongside other key details like the year of manufacture, price, and model. It might be contained within a , or similar HTML element, often labeled with a class like "km-driven".
- **CSS Selectors or XPath**: You can use CSS selectors or XPath to locate and extract the element containing the kilometers driven.

Web scrapping cars24.com

Owner :

When scraping data from Cars24.com, another valuable detail to extract is the number of previous owners of the car. This information helps potential buyers understand the car's history, with fewer previous owners often being perceived as a sign of better maintenance and lower wear and tear.

1. Locating the Owner Information:

- **HTML Tags**: The number of previous owners is usually included in the car listing, often near other key details like the year of manufacture, mileage, and price. It

  might be in a , , or another element with a class or ID that identifies it, such as “ owner “ .

- **CSS Selectors or XPath**: Use CSS selectors or XPath to specifically target the element that contains the owner information.

Fuel - Type :

When scraping data from Cars24.com, the fuel type of the car is another critical detail to extract. The fuel type (such as petrol, diesel, CNG, electric, etc.) is a key factor that influences a buyer's decision, as it affects the running cost, environmental impact, and performance of the vehicle.

1. Locating the Fuel Type:

- **HTML Tags**: The fuel type is typically displayed alongside other key details like the model, year, and mileage. It may be found within a , , or another element with a specific class or ID, such as fuel type.
- **CSS Selectors or XPath**: Use CSS selectors or XPath to precisely target and extract the element that contains the fuel type information.

Web scrapping cars24.com

Transmission :

When scraping data from Cars24.com, extracting the transmission type of the car is another key detail. The transmission type (manual, automatic, CVT, etc.) is crucial for buyers as it affects driving experience, fuel efficiency, and maintenance costs.

1. Locating the Transmission Type:

- **HTML Tags**: The transmission type is typically listed alongside other important details like fuel type, mileage, and model. It may be found within a , , or other HTML elements, often with a class or ID like “transmission “.
- **CSS Selectors or XPath**: Use CSS selectors or XPath to accurately locate and extract the element containing the transmission information.

Price :

When scraping data from Cars24.com, extracting the price of each car is crucial. The price is a primary factor for potential buyers and is essential for analyses related to market value, pricing trends, and sales strategies.

1. Locating the Price:

- **HTML Tags**: The price is typically displayed prominently in car listings, often in a , , or similar HTML element. It may have a class or ID like “ Price”.
- **CSS Selectors or XPath**: Use CSS selectors or XPath to target the element that contains the price information.

Location :

When scraping data from Cars24.com, extracting the location of each car is important for understanding regional market dynamics and preferences. The location typically indicates where the car is available for purchase, which can be crucial for potential buyers who are looking for cars within a specific area.

Web scrapping cars24.com

1. Locating the Location Information:

- **HTML Tags**: The location information is usually displayed on the car listing page, often within a , , or other HTML elements. It might be labeled with a class or ID such as “Location”.

- **CSS Selectors or XPath**: Use CSS selectors or XPath expressions to accurately target and extract the element containing the location information.

Web scrapping cars24.com

Tools and Libraries Used

Here's a list of common Python packages and libraries used for web scraping, along with a brief description of each:

1. Requests

- **Description**: Used for making HTTP requests to fetch web pages.
- **Installation**: pip install requests

2. BeautifulSoup (bs4)

- **Description**: Used for parsing HTML and XML documents and extracting data from them.
- **Installation**: pip install BeautifulSoup4

3. Selenium

- **Description**: Used for automating web browsers and handling dynamic content that is loaded via JavaScript.
- **Installation**: pip install selenium

4. Pandas

- **Description**: Useful for data manipulation and analysis, especially when handling large datasets.
- **Installation**: pip install pandas

Web scrapping cars24.com

5. Selenium WebDriver Manager:

- **Description**: Manages browser drivers for Selenium, simplifying the setup process.
- **Installation**: pip install webdriver-manager

6. Requests-HTML:

- **Description**: Provides an easy-to-use API for making HTTP requests and handling HTML content, including JavaScript rendering.
- **Installation**: pip install requests-html

These packages cover a wide range of web scraping needs, from making requests and parsing HTML to handling dynamic content and processing data. Depending on the specific requirements of your scraping project, you may use one or more of these libraries.

Web scrapping cars24.com

Code Implementation

Discussing the steps in the project

Importing libraries

```
import requests
from bs4 import BeautifulSoup
import pandas as pd
import time
```

```
url =
"https://www.cars24.com/buy-used-car?sort=bestmatch
&serveWarrantyCount=true&gaId=776885423.1723467593&
listingSource=FilterTags&storeCityId="

year =[]
make = []
model = []
fuel_type = []
price = []
km_driven = []
transmission = []
wheels = []
EMI_per_month = []
location = []
```

Web scrapping cars24.com

```
city_codes =
['4709','2','2378','3686','1692','5','777']
soup = []

for code in city_codes:
    response = requests.get(url + code)
                                    response =
BeautifulSoup(response.content,'html.parser')
    soup.append(response)

for soup in soup:
             car_details   =   soup.find_all('h3',
{'class':'_11dVb'})

    #year,make,model
    for detail in car_details:
          detail_list = detail.text.strip().split("
")
        #year
        car_year = detail_list[0]
        year.append(car_year)
        #make of the car
        car_make = detail_list[1]
        make.append(car_make)
        #model of the car
        car_model1 = detail_list[2]
        car_model2 = detail_list[3]
```

Web scrapping cars24.com

```
        model.append(car_model1 + " " + car_model2)

    #fueltype
    ul = soup.find_all('ul', {'class':'_3J2G-'})
    for li in ul:
            count = li.text.split("km")[1]
            if 'CNG' in count :
                fuel_type.append('CNG')
            else :
                count = count[0:6]
                fuel_type.append(count)

    #prices
              prices    =    soup.find_all('strong',
{'class':'_3RL-I'})
    # Loop through and print each price
    for cost in prices:
            count = cost.text.replace(' Lakh',"")
            count = count.replace('₹','')
            price.append(count)

    #kilometer_driven
               kilometer    =    soup.find_all('ul',
{'class':'_3J2G-'})
    # Loop through and print each price
    for km in kilometer:
        count = km.text.split(" ")[0]
        km_driven.append(count)
```

Web scrapping cars24.com

```
    #transmission
          transmission_type   =   soup.find_all('ul',
{'class':'_3J2G-'} )

    # Loop through and print each price
    for tm in transmission_type:

        tm_type = tm.text.split("km")[1]
        if 'CNG' in tm_type :
            count = tm_type[3:]
            transmission.append(count)
        else :
            count =  tm_type[6:]
            transmission.append(count)

    #wheels_type
      wheels_type  =  soup.find_all('span',  {'class':
'_3JoYA'})
    for tag in wheels_type :
        count = tag.text
        wheels.append(count)

    #emi per month
     emi  =  soup.find_all('span',{'class':  '_2O0yU'}
)
    for tag in emi :
        count =tag.text.split(" ")[2]
```

Web scrapping cars24.com

```
        emi_ = count.replace("/month", "").strip()
        emi_ = emi_.replace("₹", "")
        EMI_per_month.append(emi_)

    #location
    #pune,new-delhi,mumbai,hyderabad,banglore
                                locations        =
soup.find_all('p',{'class':'_3dGMY'})
    # Loop through and print each price
    for loc in locations:

         # Find all the list items (span elements)
        span_elements = loc.find_all('span')
                #  The  last  span  element  is  the
location_test
        location_test = span_elements[-1].text

        # Add the transmission type to the list
        location.append(location_test)
```

Web scrapping cars24.com

```
# Find the minimum length among all lists
min_length = min(len(year), len(make), len(model),
len(fuel_type), len(transmission),

len(price),len(km_driven),len(wheels),len(EMI_per_m
onth), len(location))

# Slice all lists to the minimum length
year = year[:min_length]
make = make[:min_length]
model = model[:min_length]
fuel_type = fuel_type[:min_length]
price = price[:min_length]
km_driven = km_driven[:min_length]
wheels = wheels[:min_length]
transmission = transmission[:min_length]
EMI_per_month = EMI_per_month[:min_length]
location = location[:min_length]
df = pd.DataFrame({

    'YEAR': year,
    'MAKE' : make,
    'MODEL' : model,
    'FUEL_TYPE': fuel_type,
    'PRICE': price,
    'KM_DRIVEN' : km_driven,
    'TRANSMISSION': transmission,
```

Web scrapping cars24.com

```
    'WHEEL_TYPE' : wheels,
    'EMI_PER_MONTH': EMI_per_month,
    'LOCATION' : location


})
```

```
df.to_csv('E:/internship/cars24webscrapping/cars24.csv')
```

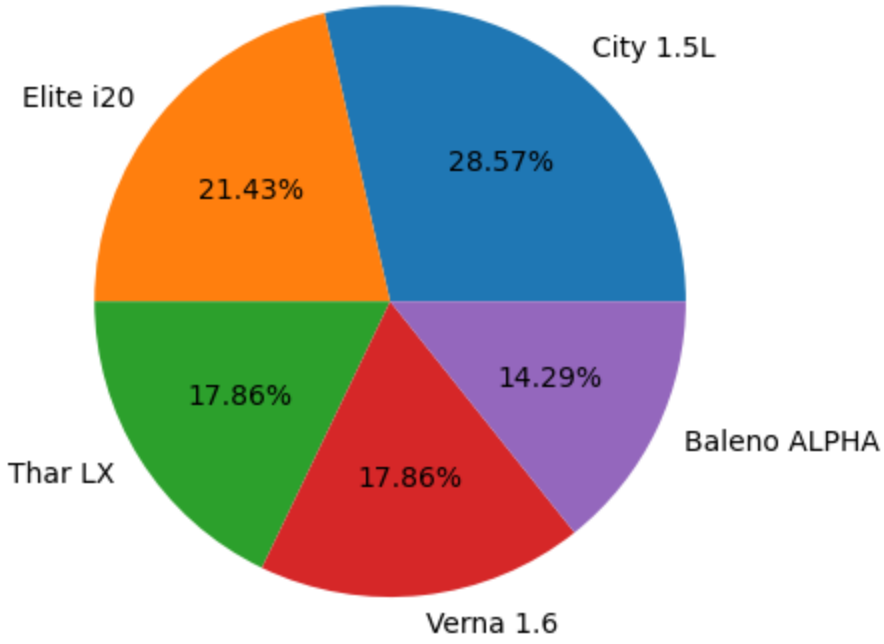
Results

Sample of CSV file is here:

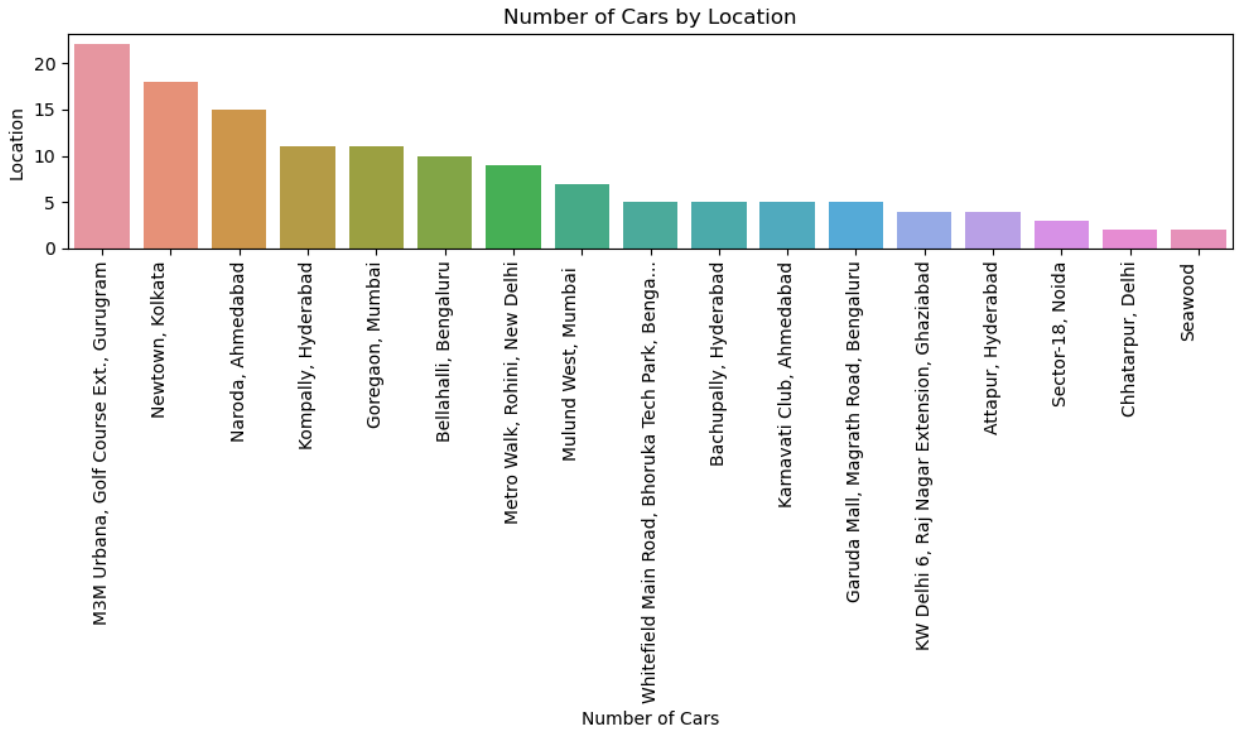
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1		YEAR	MAKE	MODEL	FUEL_TYPI	PRICE	KM_DRIVE	TRANSMIS	WHEEL_T	EMI_PER_	LOCATION				
2	0	2015	Hyundai	Creta S	Diesel	8.5	61,718	Manual	Alloy whee	18,481	Garuda Mall, Magrath Road, Bengaluru				
3	1	2013	Honda	City 1.5L	Petrol	4.64	84,818	Manual	Alloy whee	12,219	Whitefield Main Road, Bhoruka Tech Park, Benga...				
4	2	2015	Renault	Duster 110	Diesel	5.74	94,510	Manual	Reg. servic	12,768	Bellahalli, Bengaluru				
5	3	2021	Tata	PUNCH AC	Petrol	8.49	8,945	Automatic	Top Mode	16,160	Garuda Mall, Magrath Road, Bengaluru				
6	4	2016	Ford	Ecosport T	Petrol	6.16	71,617	Manual	Spl. reg. nc	12,043	Garuda Mall, Magrath Road, Bengaluru				
7	5	2017	Renault	Duster 85	Diesel	7.75	71,918	Manual	Reg. servic	15,151	Garuda Mall, Magrath Road, Bengaluru				
8	6	2015	Mahindra	XUV500 W	Diesel	7.85	98,338	Manual	Alloy whee	17,462	Bellahalli, Bengaluru				
9	7	2019	Toyota	Glanza G	Petrol	6.81	53,435	Manual	Alloy whee	13,314	Bellahalli, Bengaluru				
10	8	2020	Tata	ALTROZ XZ	Petrol	7.35	11,336	Manual	Low run cc	14,370	Bellahalli, Bengaluru				
11	9	2017	Datsun	Redi Go	Petrol	2.62	43,814	Manual	Alloy whee	5,122	Bellahalli, Bengaluru				
12	10	2018	Maruti	Baleno RS	Petrol	6.54	61,005	Manual	Alloy whee	12,786	Whitefield Main Road, Bhoruka Tech Park, Benga...				

Pie chart of top 5 models of cars:

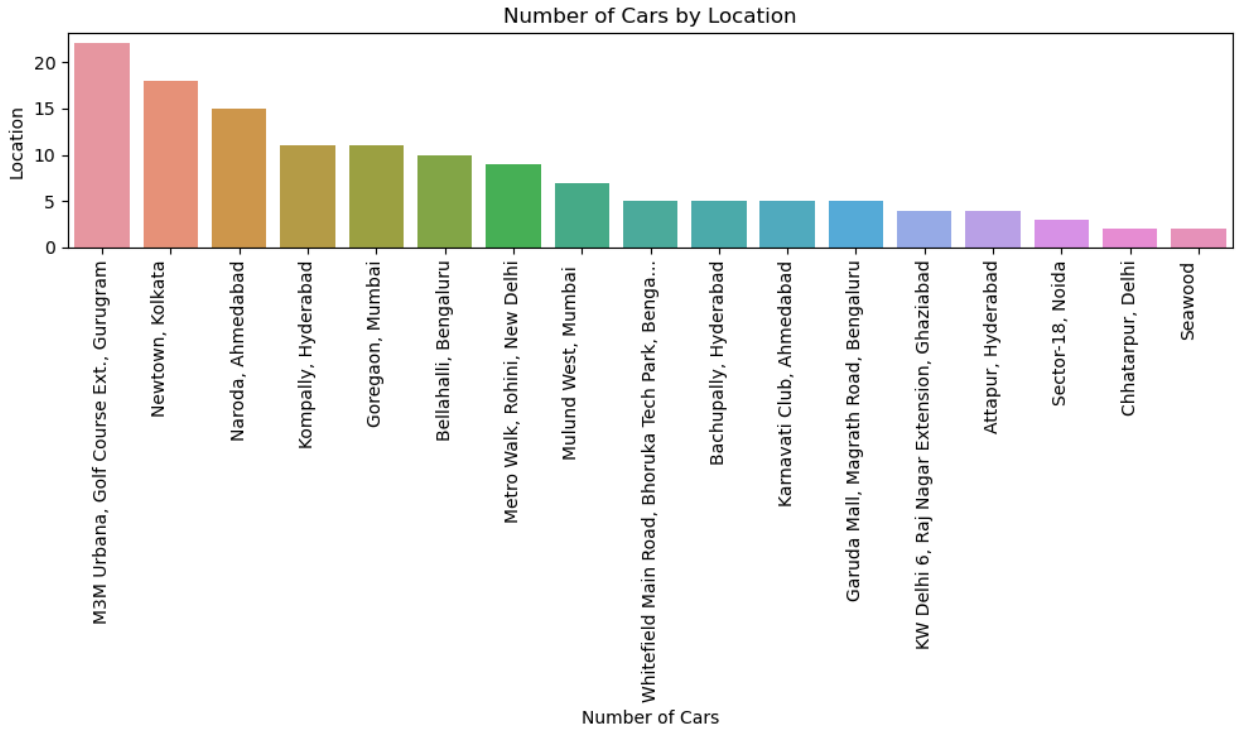
Web scrapping cars24.com



Number of cars by location :

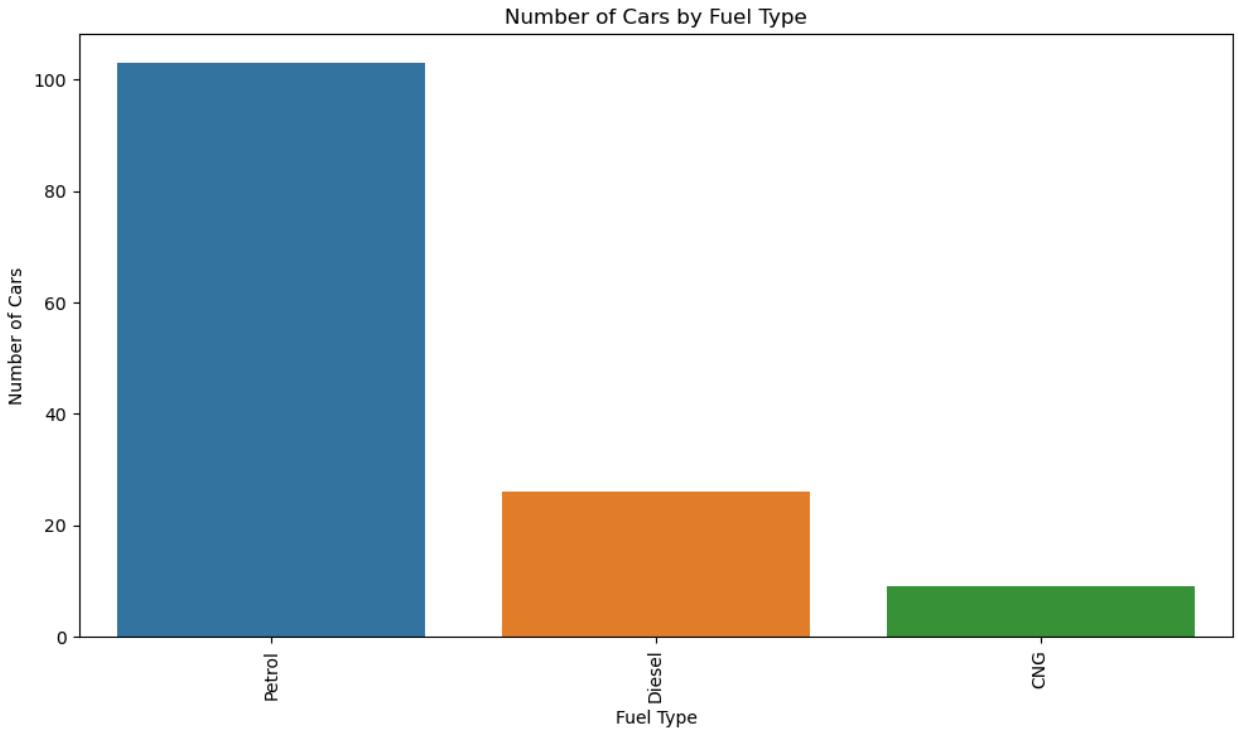


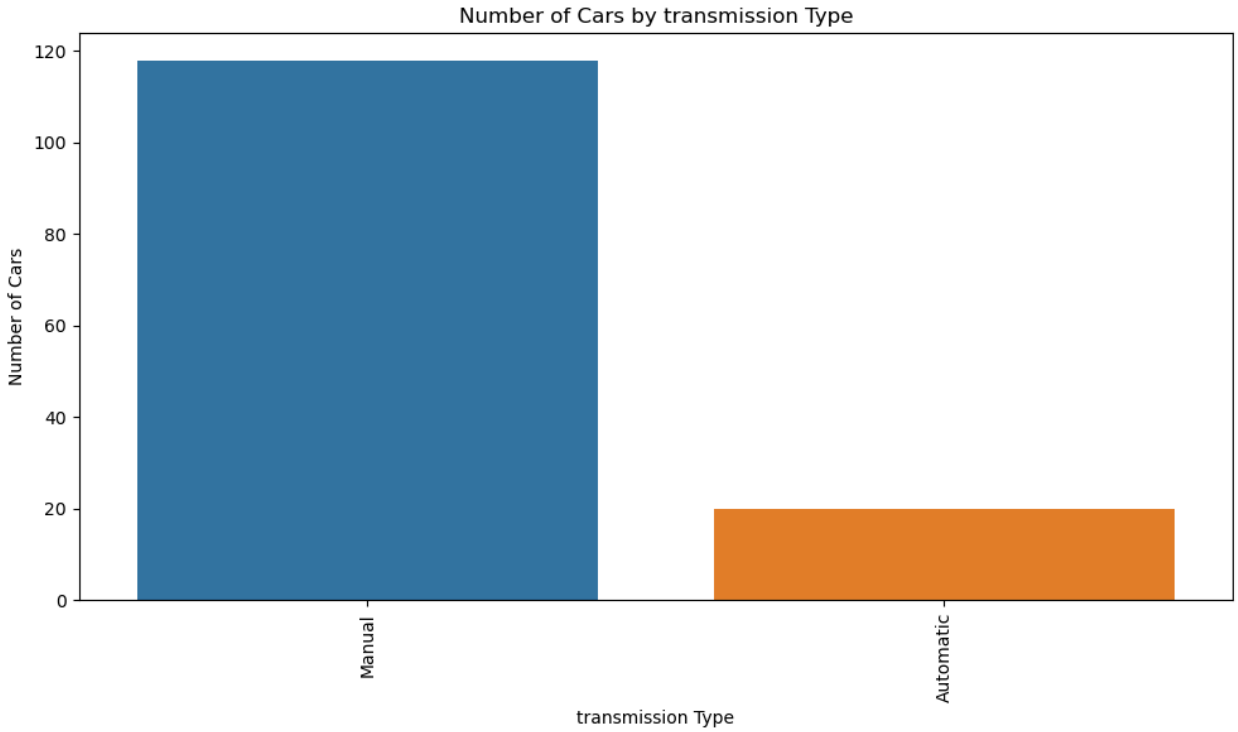
Web scrapping cars24.com



Number of cars by Fuel Type

Web scrapping cars24.com





Web scrapping cars24.com

Conclusion

This project demonstrates the power of web scraping in automating the extraction of valuable data from websites like Cars24.com. By using Selenium, we were able to gather detailed information on used cars, including the make, model, year, kilometers driven, price, and more. The data was then organized into a structured format, making it easy to analyze and use for further insights.

The process highlights the importance of understanding HTML structure, handling dynamic content, and adhering to ethical and legal guidelines. Ultimately, this project showcases how web scraping can simplify data collection and provide meaningful insights into online markets**.**