Evostra Ventures Internship Mini Project on Web Scraping

Prepared by-

Web Scraping:

1. Fathima Dilshana PK

2. Saravanan P

3. Abhishek Singh

Data Cleaning and Predictive analysis Model:

1. Uthra Nitheya C G

Data Visualization:

1. Sakshi Vijay Mindhe

2. Anand Eswar

3. Kiran Babu P

Presentation Making:

1. Dinesh Solanki

Project Report Making:

1. Mehar Snotra

2. Ajay Krishna M

Project date: 09/09/24

Evostra Ventures: Internship Program

This is to certify that the project report titled “Web Scraping” is the bonafide work of Fatima Dilshana, Sakshi Vijay Mindhe, Mehar Snotra, Dinesh Solanki, Saravanan P,Uthra Nitheya ,Abhishek Singh, Anand Eswar,Kiran Babu P and Ajay Krishna. This project was conducted as part of their internship at Evostra Ventures and was completed under my supervision.

Throughout the course of this project, the team has demonstrated exemplary dedication and a profound understanding of the essential aspects of web scraping, a critical technique in data science for extracting and analyzing large volumes of data from the web. Their work showcases a deep commitment to the application of data science principles, employing advanced methodologies and tools to gather, clean, and analyze data, ultimately contributing valuable insights. The team’s approach to the project has been both systematic and innovative. They have successfully navigated the complexities of handling unstructured data, transforming it into a structured format suitable for analysis. Their project not only reflects their technical proficiency but also their ability to work collaboratively and solve problems efficiently, key traits necessary for success in the field of data science.

I am confident that the skills and knowledge they have gained during this project will serve as a strong foundation for their future endeavors in data science and related fields. It has been a pleasure to oversee their progress, and I commend them for their hard work and achievements.

Acknowledgement

First and foremost, we would like to express our deepest gratitude to the Almighty for His blessings and guidance throughout this endeavor. His divine support has been a sourceof strength and inspiration in our journey.

We extend our heartfelt thanks to our beloved parents for their unwavering support and encouragement. Their invaluable assistance and belief in our abilities have played a crucial role inthe successful completion of this project.

We are also profoundly grateful to all the staff members at Evostra Ventures for their assistance and insights, which greatly facilitated our work. Their expertise and readiness to help were instrumental in overcoming various challenges we encountered during the project.

Additionally, we wish to acknowledge our friends who provided both moral support and practical help. Their contributions have been immensely beneficial in completing this project.

This project would not have been possible without the collective support and guidance from all these wonderful individuals, and we are deeply appreciative of their efforts.

Abstract

Web scraping refers to the process of extracting information from specific web services and converting non-homogeneous or semi-homogeneous data into a structured format suitable for analysis. This project focuses on developing a web scraperusing the Python programming language to extract various types of data from a designated website. The primary objective was to collect information such as product details, reviews, and other relevant data, and save it in formats like
CSV or JSON for efficient processing.
To achieve this, the project utilized libraries such as
BeautifulSoup and Requests to parse HTML content and retrieve data from web pages. Challenges encountered duringthe project included dynamic content loading via JavaScript, CAPTCHA as an anti-scraping measure, and IP blocking.
These issues were addressed through the implementation ofdelays, error handling, and the use of Selenium to automateinteractions with the web pages.
The results demonstrated that web scraping is an effective method for large-scale data collection, successfully extractingthe required data for the project's objectives. The gathered information proved valuable for subsequent analysis and highlighted the potential applications of web scraping in various fields, including market research, sentiment analysis, and competitive intelligence. Future improvements may involve optimizing scraper performance, enhancing the handling of JavaScript-heavy sites, and developing strategies to minimize detection by anti-scraping mechanisms

Index

Table of Contents

Table of Images

Introduction

Web scraping, also known as web harvesting or web data extraction, is a powerful technique used to automatically collect information from websites. It involves fetching web pages and extracting meaningful data from them, converting unstructured information into a structured format that can be analyzed and utilized for various purposes. This technique hasbecome increasingly significant in the field of data science due to its ability to gather large volumes of data quickly and efficiently, which is crucial for tasks such as market analysis, competitive intelligence, and academic research.

The rapid growth of digital information and the proliferation of online resources have made web scraping an essential tool for data-driven decision-making. By leveraging web scraping, organizations and researchers can access vast amounts of data that are publicly available on the internet but may be challenging to collect manually. This project focuses on the development of a web scraper using Python, a widely-used programming language for data science, to extract detailed information from a specific website.

The project aims to address several key aspects: the efficient extraction of data, the handling of dynamic content loaded through JavaScript, and the overcoming of anti-scraping measures such as CAPTCHAs and IP blocking. Additionally, the project seeks to ensure the accuracy and quality of the collected data by implementing robust validation and error handling mechanisms. By achieving these objectives, the project will not only demonstrate the technical capabilities of web scraping but also showcase its practical applications in various domains.

In essence, web scraping serves as a bridge between unstructured web content and structured data analysis, providing valuable insights and enabling informed decision-making. This introduction sets the stage forunderstanding the significance of the project and the methodology employed to achieve its goal

Objective

The primary objective of this project was to develop a robust web scraping solution to efficiently collect and analyze data from a designated website. The core aim was to build a web scraper capable of extracting a diverse range of data, including product details, customer reviews, pricing information, and other relevant data points. To achieve this, the project sought to convert unstructured data from web pages into a structured format suitable for further analysis, with the data saved in CSV and JSON formats.

A significant aspect of the project involved addressing the challenges associatedwith dynamic content. Many modern websites use JavaScript to load data after the initial page load, necessitating the use of advanced techniques to capture all relevant information. The project aimed to overcome these challenges by employing tools like Selenium to handle dynamic content and ensure comprehensive data collection.

Another critical objective was to navigate and bypass common anti-scraping measures, such as CAPTCHAs and IP blocking. These mechanisms are often implemented to prevent automated data extraction, so the project incorporated strategies such as request throttling, IP rotation, and CAPTCHA handling to maintain uninterrupted data collection.

Ensuring the accuracy and quality of the collected data was also a priority. The project included measures for data validation and error handling to ensure that the data retrieved was accurate and complete. This attention to data quality was essential for maintaining the integrity of the analysis.

Finally, the project aimed to demonstrate the practical applications of the collected data by performing preliminary analyses or integrating it into a sampleuse case. This could include market research, sentiment analysis, or other relevant domains, showcasing the value of web scraping as a tool for generating actionable insights in real-world scenarios.

Overall, the project was designed to showcase the effectiveness of web scrapingin handling various challenges and its potential applications in data science.

Methodology

The methodology for this web scraping project involved several key steps to ensure the successful extraction and analysis of data. First, the project utilized Python as the primary programming language due to its extensive libraries and ease of use for web scraping tasks. Essential libraries included BeautifulSoup for parsing HTML and extracting data, and Requests for sending HTTP requests and retrieving web content. Additionally, Selenium was employed to manage dynamic content and automate interactions with web pages that used JavaScript for data loading.

The implementation process began with setting up the development environment and installing the necessary libraries. The web scraper was designed to navigate through the target website and extract specific data fields, such as product names, prices, and reviews. BeautifulSoup was used to parse the HTML content and retrieve the desired information, while Requests facilitated the fetching of web pages.

To address the challenges of dynamic content loading, Selenium was incorporated to interact with elements that were not immediately visible on page load. This approach allowed the scraper to handle JavaScript-generated content effectively. The project also included mechanisms for managing

anti-scraping measures, such as CAPTCHAs and IP blocking. Techniques like implementing delays between requests, using proxy servers, and handling CAPTCHA challenges were employed to ensure smooth and uninterrupted data collection.

Data was then structured and saved in CSV and JSON formats to facilitate further analysis. Error handling and data validation procedures were integrated to maintain the quality and accuracy of the collected information. The methodology aimed to create a reliable and efficient web scraping tool capable of handling various challenges and producing valuable data for analysis.

This structured approach ensured that the web scraper was robust, effective, and capable of addressing both technical and practical challenges encountered during the project.

Challenges and Solutions

During the development and execution of the web scraping project, several challenges were encountered, each addressed with specific solutions to ensure the success of the scraper. One of the primary challenges was handling dynamic content loading. Many modern websites use JavaScript to load data asynchronously after the initialpage load, which can complicate data extraction. To address this, Selenium was employed to interact with and retrieve dynamically generated content, allowing the scraper to access all relevant data that was not immediately available through static HTML.

Another significant challenge was dealing with anti-scraping measures such as CAPTCHAs and IP blocking. Websites often implement these measures to prevent automated data extraction. To circumvent CAPTCHAs, the project incorporated techniques such as introducing delays between requests and using CAPTCHA-solving services when necessary. For IP blocking, proxies and IP rotation strategies were implemented to distribute requests across multiple IP addresses, minimizing the risk of detection and blocking.

Additionally, maintaining data quality and accuracy presented its own set of challenges. Issues such as missing or incomplete data and variations in data formatting required robust error handling and data validation mechanisms. To tackle this, the project included comprehensive error handling routines to manage unexpected issuesand data validation procedures to ensure the correctness of the extracted information.

Overall, the project successfully navigated these challenges through a combination of advanced tools, strategic techniques, and thoroughtesting, demonstrating the resilience and adaptability of the web scraping solution in various scene.

Program Implementation

As the first step of web scraping program implementation, we need to install all the necessary libraries. Here for the web scraping we are using Beautifulsoup for data extraction. So we are installing that first.

```
!pip install bs4
```

After this we have to import all the libraries needed. So Import all the libraries.

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

As next step, we are going to fetch data's of several locations , first we are fetching data's of Chennai location and the extracted data's are stored in both excel and csv files.

Chennai :

```
# Step 1: Fetch the webpage content
url = "https://www.cars24.com/buy-used-car?f=make%3A%3D%3Amahindra%3AOR%3Amake%3A%3D%3Ajeep%3AOR%3Amake%3A%3D%3Arenault&sort=bestmatch&serveWarrantyCount=true&gaId=2135718455.1725707831&listingSource=TabFilter&storeCityId=5732"
headers = {
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.131 Safari/537.36"
}
```

```
response = requests.get(url, headers=headers)

# Step 2: Parse the HTML content using BeautifulSoup
soup = BeautifulSoup(response.content, 'html.parser')

# Step 3: Find all car entries
car_entries = soup.find_all('div', class_='_7jb8Q _1Ey60')

# Step 4: Loop through each car entry and extract details
cars_data = []

for car in car_entries:
    # Extract car title and model
    car_title = car.find('h3', class_='_2Out2').contents[0].strip() if car.find('h3',
class_='_2Out2') else None
    car_model = car.find('h3', class_='_2Out2').find('span').text.strip() if car.find('h3',
class_='_2Out2').find('span') else None

    # Extract the year from the car title
    year = car_title.split()[0] if car_title else None
    car_title = ' '.join(car_title.split()[1:]) if car_title else None  # Remove the year from the
title

    # Extract mileage, fuel type, and ownership using more flexible methods
    details_list = car.find('ul', class_='_3jRcd').find_all('li') if car.find('ul', class_='_3jRcd') else
[]

    # Use regular expression to find mileage
    mileage = None
    if details_list:
        for detail in details_list:
            if re.search(r'\d+[,.\d]*\s*km', detail.text, re.IGNORECASE):
                mileage = detail.text.strip()
                break

    if mileage and not mileage.lower().endswith('km'):
        mileage += " km"

    # Extract fuel type using keyword search
```

```
    fuel_type = None
    for detail in details_list:
        text = detail.text.strip().lower()
        if "petrol" in text:
            fuel_type = "Petrol"
            break
        elif "diesel" in text:
            fuel_type = "Diesel"
            break

    # Extract ownership using keyword search
    ownership = None
    for detail in details_list:
        text = detail.text.strip().lower()
        if "1st owner" in text:
            ownership = "1st Owner"
            break
        elif "2nd owner" in text:
            ownership = "2nd Owner"
            break

    # Extract EMI information and remove "EMI from" text
    emi_info = car.find('div', class_='_1Oul-').find('span', class_='_1t1AA').text.strip() if
car.find('div', class_='_1Oul-') else None
    if emi_info:
        emi_info = emi_info.replace('EMI from', '').strip()

    # Extract price details
    price_div = car.find('div', class_='_1Oul- VMjdr')
    current_price = price_div.find('strong', class_='_37WXy').text.strip() if price_div and
price_div.find('strong', class_='_37WXy') else None
    original_price = price_div.find('span', class_='_3hb01').text.strip() if price_div and
price_div.find('span', class_='_3hb01') else None

    # Extract location details only and remove "at"
    location_info = car.find('p', class_='_2rxhF').find('span').text.strip() if car.find('p',
class_='_2rxhF') and car.find('p', class_='_2rxhF').find('span') else None
    if location_info:
        location_info = location_info.replace('at', '').strip()
```

```
    # Store extracted details in a dictionary
    car_details = {
        'Year': year,
        'Car Title': car_title,
        'Car Model': car_model,
        'Mileage': mileage,
        'Fuel Type': fuel_type,
        'Ownership': ownership,
        'EMI Info': emi_info,
        'Current Price': current_price,
        'Original Price': original_price,
        'Location': location_info
    }

    cars_data.append(car_details)

# Step 5: Create a DataFrame using pandas
df = pd.DataFrame(cars_data)

# Step 6: Save the DataFrame to an Excel file
df.to_excel('chennai.xlsx', index=False)

# Step 7: Save the DataFrame to a CSV file
df.to_csv('chennai.csv', index=False)

print("Data has been successfully saved to 'chennai.xlsx' and 'chennai.csv'.")
```

Now we are moving on to another location.

Surat:

```
# Step 1: Fetch the webpage content
url = "https://www.cars24.com/buy-used-car?f=make%3A%3D%3Amahindra%3AOR%3Amake%3A%3D%3Ajeep%3AOR%3Amake%3A%3D%3Arenault&sort=bestmatch&serveWarrantyCount=true&gaId=2135718455.1725707831&listingSource=TabFilter&storeCityId=1605"
headers = {
```

```
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/92.0.4515.131 Safari/537.36"
}

response = requests.get(url, headers=headers)

# Step 2: Parse the HTML content using BeautifulSoup
soup = BeautifulSoup(response.content, 'html.parser')

# Step 3: Find all car entries
car_entries = soup.find_all('div', class_='_7jb8Q _1Ey60')

# Step 4: Loop through each car entry and extract details
cars_data = []

for car in car_entries:
    # Extract car title and model
    car_title = car.find('h3', class_='_2Out2').contents[0].strip() if car.find('h3',
class_='_2Out2') else None
    car_model = car.find('h3', class_='_2Out2').find('span').text.strip() if car.find('h3',
class_='_2Out2').find('span') else None

    # Extract the year from the car title
    year = car_title.split()[0] if car_title else None
    car_title = ' '.join(car_title.split()[1:]) if car_title else None  # Remove the year from the
title

    # Extract mileage, fuel type, and ownership using more flexible methods
    details_list = car.find('ul', class_='_3jRcd').find_all('li') if car.find('ul', class_='_3jRcd') else
[]

    # Use regular expression to find mileage
    mileage = None
    if details_list:
        for detail in details_list:
            if re.search(r'\d+[,.\d]*\s*km', detail.text, re.IGNORECASE):
                mileage = detail.text.strip()
                break
```

```
    if mileage and not mileage.lower().endswith('km'):
        mileage += " km"

    # Extract fuel type using keyword search
    fuel_type = None
    for detail in details_list:
        text = detail.text.strip().lower()
        if "petrol" in text:
            fuel_type = "Petrol"
            break
        elif "diesel" in text:
            fuel_type = "Diesel"
            break

    # Extract ownership using keyword search
    ownership = None
    for detail in details_list:
        text = detail.text.strip().lower()
        if "1st owner" in text:
            ownership = "1st Owner"
            break
        elif "2nd owner" in text:
            ownership = "2nd Owner"
            break

    # Extract EMI information and remove "EMI from" text
    emi_info = car.find('div', class_='_1Oul-').find('span', class_='_1t1AA').text.strip() if
car.find('div', class_='_1Oul-') else None
    if emi_info:
        emi_info = emi_info.replace('EMI from', '').strip()

    # Extract price details
    price_div = car.find('div', class_='_1Oul- VMjdr')
    current_price = price_div.find('strong', class_='_37WXy').text.strip() if price_div and
price_div.find('strong', class_='_37WXy') else None
    original_price = price_div.find('span', class_='_3hb01').text.strip() if price_div and
price_div.find('span', class_='_3hb01') else None

    # Extract location details only and remove "at"
```

```
    location_info = car.find('p', class_='_2rxhF').find('span').text.strip() if car.find('p',
class_='_2rxhF') and car.find('p', class_='_2rxhF').find('span') else None
    if location_info:
        location_info = location_info.replace('at', '').strip()

    # Store extracted details in a dictionary
    car_details = {
        'Year': year,
        'Car Title': car_title,
        'Car Model': car_model,
        'Mileage': mileage,
        'Fuel Type': fuel_type,
        'Ownership': ownership,
        'EMI Info': emi_info,
        'Current Price': current_price,
        'Original Price': original_price,
        'Location': location_info
    }

    cars_data.append(car_details)

# Step 5: Create a DataFrame using pandas
df = pd.DataFrame(cars_data)

# Step 6: Save the DataFrame to an Excel file
df.to_excel('surat.xlsx', index=False)

# Step 7: Save the DataFrame to a CSV file
df.to_csv('surat.csv', index=False)

print("Data has been successfully saved to 'surat.xlsx' and 'surat.csv'.")
```

Kolkata:

```
# Step 1: Fetch the webpage content
url = "https://www.cars24.com/buy-used-
car?f=make%3A%3D%3Amahindra%3AOR%3Amake%3A%3D%3Ajeep%3AOR%3Amake%3A
%3D%3Arenault&sort=bestmatch&serveWarrantyCount=true&gaId=2135718455.1725707
831&storeCityId=777"
headers = {
```

```
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/92.0.4515.131 Safari/537.36"
}

response = requests.get(url, headers=headers)

# Step 2: Parse the HTML content using BeautifulSoup
soup = BeautifulSoup(response.content, 'html.parser')

# Step 3: Find all car entries
car_entries = soup.find_all('div', class_='_7jb8Q _1Ey60')

# Step 4: Loop through each car entry and extract details
cars_data = []

for car in car_entries:
    # Extract car title and model
    car_title = car.find('h3', class_='_2Out2').contents[0].strip() if car.find('h3',
class_='_2Out2') else None
    car_model = car.find('h3', class_='_2Out2').find('span').text.strip() if car.find('h3',
class_='_2Out2').find('span') else None

    # Extract the year from the car title
    year = car_title.split()[0] if car_title else None
    car_title = ' '.join(car_title.split()[1:]) if car_title else None  # Remove the year from the
title

    # Extract mileage, fuel type, and ownership using more flexible methods
    details_list = car.find('ul', class_='_3jRcd').find_all('li') if car.find('ul', class_='_3jRcd') else
[]

    # Use regular expression to find mileage
    mileage = None
    if details_list:
        for detail in details_list:
            if re.search(r'\d+[,.\d]*\s*km', detail.text, re.IGNORECASE):
                mileage = detail.text.strip()
                break
```

```
    if mileage and not mileage.lower().endswith('km'):
        mileage += " km"

    # Extract fuel type using keyword search
    fuel_type = None
    for detail in details_list:
        text = detail.text.strip().lower()
        if "petrol" in text:
            fuel_type = "Petrol"
            break
        elif "diesel" in text:
            fuel_type = "Diesel"
            break

    # Extract ownership using keyword search
    ownership = None
    for detail in details_list:
        text = detail.text.strip().lower()
        if "1st owner" in text:
            ownership = "1st Owner"
            break
        elif "2nd owner" in text:
            ownership = "2nd Owner"
            break

    # Extract EMI information and remove "EMI from" text
    emi_info = car.find('div', class_='_1Oul-').find('span', class_='_1t1AA').text.strip() if
car.find('div', class_='_1Oul-') else None
    if emi_info:
        emi_info = emi_info.replace('EMI from', '').strip()

    # Extract price details
    price_div = car.find('div', class_='_1Oul- VMjdr')
    current_price = price_div.find('strong', class_='_37WXy').text.strip() if price_div and
price_div.find('strong', class_='_37WXy') else None
    original_price = price_div.find('span', class_='_3hb01').text.strip() if price_div and
price_div.find('span', class_='_3hb01') else None

    # Extract location details only and remove "at"
```

```
    location_info = car.find('p', class_='_2rxhF').find('span').text.strip() if car.find('p',
class_='_2rxhF') and car.find('p', class_='_2rxhF').find('span') else None
    if location_info:
        location_info = location_info.replace('at', '').strip()

    # Store extracted details in a dictionary
    car_details = {
        'Year': year,
        'Car Title': car_title,
        'Car Model': car_model,
        'Mileage': mileage,
        'Fuel Type': fuel_type,
        'Ownership': ownership,
        'EMI Info': emi_info,
        'Current Price': current_price,
        'Original Price': original_price,
        'Location': location_info
    }

    cars_data.append(car_details)

# Step 5: Create a DataFrame using pandas
df = pd.DataFrame(cars_data)

# Step 6: Save the DataFrame to an Excel file
df.to_excel('kolkata.xlsx', index=False)

# Step 7: Save the DataFrame to a CSV file
df.to_csv('kolkata.csv', index=False)

print("Data has been successfully saved to 'kolkata.xlsx' and 'kolkata.csv'.")
```

Patna:

```
# Step 1: Fetch the webpage content
url = "https://www.cars24.com/buy-used-
car?f=make%3A%3D%3Amahindra%3AOR%3Amake%3A%3D%3Ajeep%3AOR%3Amake%3A
%3D%3Arenault&sort=bestmatch&serveWarrantyCount=true&gaId=2135718455.1725707
831&storeCityId=8184"
headers = {
```

```
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/92.0.4515.131 Safari/537.36"
}

response = requests.get(url, headers=headers)

# Step 2: Parse the HTML content using BeautifulSoup
soup = BeautifulSoup(response.content, 'html.parser')

# Step 3: Find all car entries
car_entries = soup.find_all('div', class_='_7jb8Q _1Ey60')

# Step 4: Loop through each car entry and extract details
cars_data = []

for car in car_entries:
    # Extract car title and model
    car_title = car.find('h3', class_='_2Out2').contents[0].strip() if car.find('h3',
class_='_2Out2') else None
    car_model = car.find('h3', class_='_2Out2').find('span').text.strip() if car.find('h3',
class_='_2Out2').find('span') else None

    # Extract the year from the car title
    year = car_title.split()[0] if car_title else None
    car_title = ' '.join(car_title.split()[1:]) if car_title else None  # Remove the year from the
title

    # Extract mileage, fuel type, and ownership using more flexible methods
    details_list = car.find('ul', class_='_3jRcd').find_all('li') if car.find('ul', class_='_3jRcd') else
[]

    # Use regular expression to find mileage
    mileage = None
    if details_list:
        for detail in details_list:
            if re.search(r'\d+[,.\d]*\s*km', detail.text, re.IGNORECASE):
                mileage = detail.text.strip()
                break
```

```
    if mileage and not mileage.lower().endswith('km'):
        mileage += " km"

    # Extract fuel type using keyword search
    fuel_type = None
    for detail in details_list:
        text = detail.text.strip().lower()
        if "petrol" in text:
            fuel_type = "Petrol"
            break
        elif "diesel" in text:
            fuel_type = "Diesel"
            break

    # Extract ownership using keyword search
    ownership = None
    for detail in details_list:
        text = detail.text.strip().lower()
        if "1st owner" in text:
            ownership = "1st Owner"
            break
        elif "2nd owner" in text:
            ownership = "2nd Owner"
            break

    # Extract EMI information and remove "EMI from" text
    emi_info = car.find('div', class_='_1Oul-').find('span', class_='_1t1AA').text.strip() if
car.find('div', class_='_1Oul-') else None
    if emi_info:
        emi_info = emi_info.replace('EMI from', '').strip()

    # Extract price details
    price_div = car.find('div', class_='_1Oul- VMjdr')
    current_price = price_div.find('strong', class_='_37WXy').text.strip() if price_div and
price_div.find('strong', class_='_37WXy') else None
    original_price = price_div.find('span', class_='_3hb01').text.strip() if price_div and
price_div.find('span', class_='_3hb01') else None

    # Extract location details only and remove "at"
```

```
    location_info = car.find('p', class_='_2rxhF').find('span').text.strip() if car.find('p',
class_='_2rxhF') and car.find('p', class_='_2rxhF').find('span') else None
    if location_info:
        location_info = location_info.replace('at', '').strip()

    # Store extracted details in a dictionary
    car_details = {
        'Year': year,
        'Car Title': car_title,
        'Car Model': car_model,
        'Mileage': mileage,
        'Fuel Type': fuel_type,
        'Ownership': ownership,
        'EMI Info': emi_info,
        'Current Price': current_price,
        'Original Price': original_price,
        'Location': location_info
    }

    cars_data.append(car_details)

# Step 5: Create a DataFrame using pandas
df = pd.DataFrame(cars_data)

# Step 6: Save the DataFrame to an Excel file
df.to_excel('patna.xlsx', index=False)

# Step 7: Save the DataFrame to a CSV file
df.to_csv('patna.csv', index=False)

print("Data has been successfully saved to 'patna.xlsx' and 'patna.csv'.")
```

Coimbatore:

```
# Step 1: Fetch the webpage content
url = "https://www.cars24.com/buy-used-
car?f=make%3A%3D%3Amahindra%3AOR%3Amake%3A%3D%3Ajeep%3AOR%3Amake%3A
%3D%3Arenault&sort=bestmatch&serveWarrantyCount=true&gaId=1301600899.1725538
448&storeCityId=6105"
headers = {
```

```
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/92.0.4515.131 Safari/537.36"
}

response = requests.get(url, headers=headers)

# Step 2: Parse the HTML content using BeautifulSoup
soup = BeautifulSoup(response.content, 'html.parser')

# Step 3: Find all car entries
car_entries = soup.find_all('div', class_='_7jb8Q _1Ey60')

# Step 4: Loop through each car entry and extract details
cars_data = []

for car in car_entries:
    # Extract car title and model
    car_title = car.find('h3', class_='_2Out2').contents[0].strip() if car.find('h3',
class_='_2Out2') else None
    car_model = car.find('h3', class_='_2Out2').find('span').text.strip() if car.find('h3',
class_='_2Out2').find('span') else None

    # Extract the year from the car title
    year = car_title.split()[0] if car_title else None
    car_title = ' '.join(car_title.split()[1:]) if car_title else None  # Remove the year from the
title

    # Extract mileage, fuel type, and ownership using more flexible methods
    details_list = car.find('ul', class_='_3jRcd').find_all('li') if car.find('ul', class_='_3jRcd') else
[]

    # Use regular expression to find mileage
    mileage = None
    if details_list:
        for detail in details_list:
            if re.search(r'\d+[,.\d]*\s*km', detail.text, re.IGNORECASE):
                mileage = detail.text.strip()
                break
```

```
    if mileage and not mileage.lower().endswith('km'):
        mileage += " km"

    # Extract fuel type using keyword search
    fuel_type = None
    for detail in details_list:
        text = detail.text.strip().lower()
        if "petrol" in text:
            fuel_type = "Petrol"
            break
        elif "diesel" in text:
            fuel_type = "Diesel"
            break

    # Extract ownership using keyword search
    ownership = None
    for detail in details_list:
        text = detail.text.strip().lower()
        if "1st owner" in text:
            ownership = "1st Owner"
            break
        elif "2nd owner" in text:
            ownership = "2nd Owner"
            break

    # Extract EMI information and remove "EMI from" text
    emi_info = car.find('div', class_='_1Oul-').find('span', class_='_1t1AA').text.strip() if
car.find('div', class_='_1Oul-') else None
    if emi_info:
        emi_info = emi_info.replace('EMI from', '').strip()

    # Extract price details
    price_div = car.find('div', class_='_1Oul- VMjdr')
    current_price = price_div.find('strong', class_='_37WXy').text.strip() if price_div and
price_div.find('strong', class_='_37WXy') else None
    original_price = price_div.find('span', class_='_3hb01').text.strip() if price_div and
price_div.find('span', class_='_3hb01') else None

    # Extract location details only and remove "at"
```

```
    location_info = car.find('p', class_='_2rxhF').find('span').text.strip() if car.find('p',
class_='_2rxhF') and car.find('p', class_='_2rxhF').find('span') else None
    if location_info:
        location_info = location_info.replace('at', '').strip()

    # Store extracted details in a dictionary
    car_details = {
        'Year': year,
        'Car Title': car_title,
        'Car Model': car_model,
        'Mileage': mileage,
        'Fuel Type': fuel_type,
        'Ownership': ownership,
        'EMI Info': emi_info,
        'Current Price': current_price,
        'Original Price': original_price,
        'Location': location_info
    }

    cars_data.append(car_details)

# Step 5: Create a DataFrame using pandas
df = pd.DataFrame(cars_data)

# Step 6: Save the DataFrame to an Excel file
df.to_excel('Coimbatore.xlsx', index=False)

# Step 7: Save the DataFrame to a CSV file
df.to_csv('Coimbatore.csv', index=False)

print("Data has been successfully saved to 'Coimbatore.xlsx' and 'Coimbatore.csv'.")
```

Kochi:

```
# Step 1: Fetch the webpage content
url = "https://www.cars24.com/buy-used-
car?f=make%3A%3D%3Amahindra%3AOR%3Amake%3A%3D%3Ajeep%3AOR%3Amake%3A
%3D%3Arenault&sort=bestmatch&serveWarrantyCount=true&gaId=1301600899.1725538
448&storeCityId=6356"
headers = {
```

```
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/92.0.4515.131 Safari/537.36"
}

response = requests.get(url, headers=headers)

# Step 2: Parse the HTML content using BeautifulSoup
soup = BeautifulSoup(response.content, 'html.parser')

# Step 3: Find all car entries
car_entries = soup.find_all('div', class_='_7jb8Q _1Ey60')

# Step 4: Loop through each car entry and extract details
cars_data = []

for car in car_entries:
    # Extract car title and model
    car_title = car.find('h3', class_='_2Out2').contents[0].strip() if car.find('h3',
class_='_2Out2') else None
    car_model = car.find('h3', class_='_2Out2').find('span').text.strip() if car.find('h3',
class_='_2Out2').find('span') else None

    # Extract the year from the car title
    year = car_title.split()[0] if car_title else None
    car_title = ' '.join(car_title.split()[1:]) if car_title else None  # Remove the year from the
title

    # Extract mileage, fuel type, and ownership using more flexible methods
    details_list = car.find('ul', class_='_3jRcd').find_all('li') if car.find('ul', class_='_3jRcd') else
[]

    # Use regular expression to find mileage
    mileage = None
    if details_list:
        for detail in details_list:
            if re.search(r'\d+[,.\d]*\s*km', detail.text, re.IGNORECASE):
                mileage = detail.text.strip()
                break
```

```
    if mileage and not mileage.lower().endswith('km'):
        mileage += " km"

    # Extract fuel type using keyword search
    fuel_type = None
    for detail in details_list:
        text = detail.text.strip().lower()
        if "petrol" in text:
            fuel_type = "Petrol"
            break
        elif "diesel" in text:
            fuel_type = "Diesel"
            break

    # Extract ownership using keyword search
    ownership = None
    for detail in details_list:
        text = detail.text.strip().lower()
        if "1st owner" in text:
            ownership = "1st Owner"
            break
        elif "2nd owner" in text:
            ownership = "2nd Owner"
            break

    # Extract EMI information and remove "EMI from" text
    emi_info = car.find('div', class_='_1Oul-').find('span', class_='_1t1AA').text.strip() if
car.find('div', class_='_1Oul-') else None
    if emi_info:
        emi_info = emi_info.replace('EMI from', '').strip()

    # Extract price details
    price_div = car.find('div', class_='_1Oul- VMjdr')
    current_price = price_div.find('strong', class_='_37WXy').text.strip() if price_div and
price_div.find('strong', class_='_37WXy') else None
    original_price = price_div.find('span', class_='_3hb01').text.strip() if price_div and
price_div.find('span', class_='_3hb01') else None

    # Extract location details only and remove "at"
```

```
    location_info = car.find('p', class_='_2rxhF').find('span').text.strip() if car.find('p',
class_='_2rxhF') and car.find('p', class_='_2rxhF').find('span') else None
    if location_info:
        location_info = location_info.replace('at', '').strip()

    # Store extracted details in a dictionary
    car_details = {
        'Year': year,
        'Car Title': car_title,
        'Car Model': car_model,
        'Mileage': mileage,
        'Fuel Type': fuel_type,
        'Ownership': ownership,
        'EMI Info': emi_info,
        'Current Price': current_price,
        'Original Price': original_price,
        'Location': location_info
    }

    cars_data.append(car_details)

# Step 5: Create a DataFrame using pandas
df = pd.DataFrame(cars_data)

# Step 6: Save the DataFrame to an Excel file
df.to_excel('Kochi.xlsx', index=False)

# Step 7: Save the DataFrame to a CSV file
df.to_csv('Kochi.csv', index=False)

print("Data has been successfully saved to 'Kochi.xlsx' and 'Kochi.csv'.")
```

Ahmedabad:

```
# Step 1: Fetch the webpage content
url = "https://www.cars24.com/buy-used-
car?f=make%3A%3D%3Amahindra%3AOR%3Amake%3A%3D%3Ajeep%3AOR%3Amake%3A
%3D%3Arenault&sort=bestmatch&serveWarrantyCount=true&gaId=1301600899.1725538
448&storeCityId=1692"
headers = {
```

```
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/92.0.4515.131 Safari/537.36"
}

response = requests.get(url, headers=headers)

# Step 2: Parse the HTML content using BeautifulSoup
soup = BeautifulSoup(response.content, 'html.parser')

# Step 3: Find all car entries
car_entries = soup.find_all('div', class_='_7jb8Q _1Ey60')

# Step 4: Loop through each car entry and extract details
cars_data = []

for car in car_entries:
    # Extract car title and model
    car_title = car.find('h3', class_='_2Out2').contents[0].strip() if car.find('h3',
class_='_2Out2') else None
    car_model = car.find('h3', class_='_2Out2').find('span').text.strip() if car.find('h3',
class_='_2Out2').find('span') else None

    # Extract the year from the car title
    year = car_title.split()[0] if car_title else None
    car_title = ' '.join(car_title.split()[1:]) if car_title else None  # Remove the year from the
title

    # Extract mileage, fuel type, and ownership using more flexible methods
    details_list = car.find('ul', class_='_3jRcd').find_all('li') if car.find('ul', class_='_3jRcd') else
[]

    # Use regular expression to find mileage
    mileage = None
    if details_list:
        for detail in details_list:
            if re.search(r'\d+[,.\d]*\s*km', detail.text, re.IGNORECASE):
                mileage = detail.text.strip()
                break
```

```
    if mileage and not mileage.lower().endswith('km'):
        mileage += " km"

    # Extract fuel type using keyword search
    fuel_type = None
    for detail in details_list:
        text = detail.text.strip().lower()
        if "petrol" in text:
            fuel_type = "Petrol"
            break
        elif "diesel" in text:
            fuel_type = "Diesel"
            break

    # Extract ownership using keyword search
    ownership = None
    for detail in details_list:
        text = detail.text.strip().lower()
        if "1st owner" in text:
            ownership = "1st Owner"
            break
        elif "2nd owner" in text:
            ownership = "2nd Owner"
            break

    # Extract EMI information and remove "EMI from" text
    emi_info = car.find('div', class_='_1Oul-').find('span', class_='_1t1AA').text.strip() if
car.find('div', class_='_1Oul-') else None
    if emi_info:
        emi_info = emi_info.replace('EMI from', '').strip()

    # Extract price details
    price_div = car.find('div', class_='_1Oul- VMjdr')
    current_price = price_div.find('strong', class_='_37WXy').text.strip() if price_div and
price_div.find('strong', class_='_37WXy') else None
    original_price = price_div.find('span', class_='_3hb01').text.strip() if price_div and
price_div.find('span', class_='_3hb01') else None

    # Extract location details only and remove "at"
```

```
    location_info = car.find('p', class_='_2rxhF').find('span').text.strip() if car.find('p',
class_='_2rxhF') and car.find('p', class_='_2rxhF').find('span') else None
    if location_info:
        location_info = location_info.replace('at', '').strip()

    # Store extracted details in a dictionary
    car_details = {
        'Year': year,
        'Car Title': car_title,
        'Car Model': car_model,
        'Mileage': mileage,
        'Fuel Type': fuel_type,
        'Ownership': ownership,
        'EMI Info': emi_info,
        'Current Price': current_price,
        'Original Price': original_price,
        'Location': location_info
    }

    cars_data.append(car_details)

# Step 5: Create a DataFrame using pandas
df = pd.DataFrame(cars_data)

# Step 6: Save the DataFrame to an Excel file
df.to_excel('Ahamedabad.xlsx', index=False)

# Step 7: Save the DataFrame to a CSV file
df.to_csv('Ahamedabad.csv', index=False)

print("Data has been successfully saved to 'Ahamedabad.xlsx' and 'Ahamedabad.csv'.")
```

Hyderabad:

```
# Step 1: Fetch the webpage content
url = "https://www.cars24.com/buy-used-
car?f=make%3A%3D%3Amahindra%3AOR%3Amake%3A%3D%3Ajeep%3AOR%3Amake%3A
%3D%3Arenault&sort=bestmatch&serveWarrantyCount=true&gaId=1301600899.1725538
448&storeCityId=3686"
headers = {
```

```
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/92.0.4515.131 Safari/537.36"
}

response = requests.get(url, headers=headers)

# Step 2: Parse the HTML content using BeautifulSoup
soup = BeautifulSoup(response.content, 'html.parser')

# Step 3: Find all car entries
car_entries = soup.find_all('div', class_='_7jb8Q _1Ey60')

# Step 4: Loop through each car entry and extract details
cars_data = []

for car in car_entries:
    # Extract car title and model
    car_title = car.find('h3', class_='_2Out2').contents[0].strip() if car.find('h3',
class_='_2Out2') else None
    car_model = car.find('h3', class_='_2Out2').find('span').text.strip() if car.find('h3',
class_='_2Out2').find('span') else None

    # Extract the year from the car title
    year = car_title.split()[0] if car_title else None
    car_title = ' '.join(car_title.split()[1:]) if car_title else None  # Remove the year from the
title

    # Extract mileage, fuel type, and ownership using more flexible methods
    details_list = car.find('ul', class_='_3jRcd').find_all('li') if car.find('ul', class_='_3jRcd') else
[]

    # Use regular expression to find mileage
    mileage = None
    if details_list:
        for detail in details_list:
            if re.search(r'\d+[,.\d]*\s*km', detail.text, re.IGNORECASE):
                mileage = detail.text.strip()
                break
```

```
    if mileage and not mileage.lower().endswith('km'):
        mileage += " km"

    # Extract fuel type using keyword search
    fuel_type = None
    for detail in details_list:
        text = detail.text.strip().lower()
        if "petrol" in text:
            fuel_type = "Petrol"
            break
        elif "diesel" in text:
            fuel_type = "Diesel"
            break

    # Extract ownership using keyword search
    ownership = None
    for detail in details_list:
        text = detail.text.strip().lower()
        if "1st owner" in text:
            ownership = "1st Owner"
            break
        elif "2nd owner" in text:
            ownership = "2nd Owner"
            break

    # Extract EMI information and remove "EMI from" text
    emi_info = car.find('div', class_='_1Oul-').find('span', class_='_1t1AA').text.strip() if
car.find('div', class_='_1Oul-') else None
    if emi_info:
        emi_info = emi_info.replace('EMI from', '').strip()

    # Extract price details
    price_div = car.find('div', class_='_1Oul- VMjdr')
    current_price = price_div.find('strong', class_='_37WXy').text.strip() if price_div and
price_div.find('strong', class_='_37WXy') else None
    original_price = price_div.find('span', class_='_3hb01').text.strip() if price_div and
price_div.find('span', class_='_3hb01') else None

    # Extract location details only and remove "at"
```

```
    location_info = car.find('p', class_='_2rxhF').find('span').text.strip() if car.find('p',
class_='_2rxhF') and car.find('p', class_='_2rxhF').find('span') else None
    if location_info:
        location_info = location_info.replace('at', '').strip()

    # Store extracted details in a dictionary
    car_details = {
        'Year': year,
        'Car Title': car_title,
        'Car Model': car_model,
        'Mileage': mileage,
        'Fuel Type': fuel_type,
        'Ownership': ownership,
        'EMI Info': emi_info,
        'Current Price': current_price,
        'Original Price': original_price,
        'Location': location_info
    }

    cars_data.append(car_details)

# Step 5: Create a DataFrame using pandas
df = pd.DataFrame(cars_data)

# Step 6: Save the DataFrame to an Excel file
df.to_excel('Hyderabad.xlsx', index=False)

# Step 7: Save the DataFrame to a CSV file
df.to_csv('Hyderabad.csv', index=False)

print("Data has been successfully saved to 'Hyderabad.xlsx' and 'Hyderabad.csv'.")
```

Indore:

```
# Step 1: Fetch the webpage content
url = "https://www.cars24.com/buy-used-
car?f=make%3A%3D%3Amahindra%3AOR%3Amake%3A%3D%3Ajeep%3AOR%3Amake%3A
%3D%3Arenault&sort=bestmatch&serveWarrantyCount=true&gaId=1301600899.1725538
448&storeCityId=2920"
headers = {
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
```

```
(KHTML, like Gecko) Chrome/92.0.4515.131 Safari/537.36"
}

response = requests.get(url, headers=headers)

# Step 2: Parse the HTML content using BeautifulSoup
soup = BeautifulSoup(response.content, 'html.parser')

# Step 3: Find all car entries
car_entries = soup.find_all('div', class_='_7jb8Q _1Ey60')

# Step 4: Loop through each car entry and extract details
cars_data = []

for car in car_entries:
    # Extract car title and model
    car_title = car.find('h3', class_='_2Out2').contents[0].strip() if car.find('h3',
class_='_2Out2') else None
    car_model = car.find('h3', class_='_2Out2').find('span').text.strip() if car.find('h3',
class_='_2Out2').find('span') else None

    # Extract the year from the car title
    year = car_title.split()[0] if car_title else None
    car_title = ' '.join(car_title.split()[1:]) if car_title else None  # Remove the year from the
title

    # Extract mileage, fuel type, and ownership using more flexible methods
    details_list = car.find('ul', class_='_3jRcd').find_all('li') if car.find('ul', class_='_3jRcd') else
[]

    # Use regular expression to find mileage
    mileage = None
    if details_list:
        for detail in details_list:
            if re.search(r'\d+[,.\d]*\s*km', detail.text, re.IGNORECASE):
                mileage = detail.text.strip()
                break

    if mileage and not mileage.lower().endswith('km'):
```

```
        mileage += " km"

    # Extract fuel type using keyword search
    fuel_type = None
    for detail in details_list:
        text = detail.text.strip().lower()
        if "petrol" in text:
            fuel_type = "Petrol"
            break
        elif "diesel" in text:
            fuel_type = "Diesel"
            break

    # Extract ownership using keyword search
    ownership = None
    for detail in details_list:
        text = detail.text.strip().lower()
        if "1st owner" in text:
            ownership = "1st Owner"
            break
        elif "2nd owner" in text:
            ownership = "2nd Owner"
            break

    # Extract EMI information and remove "EMI from" text
    emi_info = car.find('div', class_='_1Oul-').find('span', class_='_1t1AA').text.strip() if
car.find('div', class_='_1Oul-') else None
    if emi_info:
        emi_info = emi_info.replace('EMI from', '').strip()

    # Extract price details
    price_div = car.find('div', class_='_1Oul- VMjdr')
    current_price = price_div.find('strong', class_='_37WXy').text.strip() if price_div and
price_div.find('strong', class_='_37WXy') else None
    original_price = price_div.find('span', class_='_3hb01').text.strip() if price_div and
price_div.find('span', class_='_3hb01') else None

    # Extract location details only and remove "at"
    location_info = car.find('p', class_='_2rxhF').find('span').text.strip() if car.find('p',
```

```
class_='_2rxhF') and car.find('p', class_='_2rxhF').find('span') else None
    if location_info:
        location_info = location_info.replace('at', '').strip()

    # Store extracted details in a dictionary
    car_details = {
        'Year': year,
        'Car Title': car_title,
        'Car Model': car_model,
        'Mileage': mileage,
        'Fuel Type': fuel_type,
        'Ownership': ownership,
        'EMI Info': emi_info,
        'Current Price': current_price,
        'Original Price': original_price,
        'Location': location_info
    }

    cars_data.append(car_details)

# Step 5: Create a DataFrame using pandas
df = pd.DataFrame(cars_data)

# Step 6: Save the DataFrame to an Excel file
df.to_excel('Indore.xlsx', index=False)

# Step 7: Save the DataFrame to a CSV file
df.to_csv('Indore.csv', index=False)

print("Data has been successfully saved to 'Indore.xlsx' and 'Indore.csv'.")
```

Agra:

```
# Step 1: Fetch the webpage content
url = "https://www.cars24.com/buy-used-
car?f=make%3A%3D%3Amahindra%3AOR%3Amake%3A%3D%3Ajeep%3AOR%3Amake%3A
%3D%3Arenault&sort=bestmatch&serveWarrantyCount=true&gaId=1301600899.1725538
448&storeCityId=136"
headers = {
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
```

```
(KHTML, like Gecko) Chrome/92.0.4515.131 Safari/537.36"
}

response = requests.get(url, headers=headers)

# Step 2: Parse the HTML content using BeautifulSoup
soup = BeautifulSoup(response.content, 'html.parser')

# Step 3: Find all car entries
car_entries = soup.find_all('div', class_='_7jb8Q _1Ey60')

# Step 4: Loop through each car entry and extract details
cars_data = []

for car in car_entries:
    # Extract car title and model
    car_title = car.find('h3', class_='_2Out2').contents[0].strip() if car.find('h3',
class_='_2Out2') else None
    car_model = car.find('h3', class_='_2Out2').find('span').text.strip() if car.find('h3',
class_='_2Out2').find('span') else None

    # Extract the year from the car title
    year = car_title.split()[0] if car_title else None
    car_title = ' '.join(car_title.split()[1:]) if car_title else None  # Remove the year from the
title

    # Extract mileage, fuel type, and ownership using more flexible methods
    details_list = car.find('ul', class_='_3jRcd').find_all('li') if car.find('ul', class_='_3jRcd') else
[]

    # Use regular expression to find mileage
    mileage = None
    if details_list:
        for detail in details_list:
            if re.search(r'\d+[,.\d]*\s*km', detail.text, re.IGNORECASE):
                mileage = detail.text.strip()
                break

    if mileage and not mileage.lower().endswith('km'):
```

```
        mileage += " km"

    # Extract fuel type using keyword search
    fuel_type = None
    for detail in details_list:
        text = detail.text.strip().lower()
        if "petrol" in text:
            fuel_type = "Petrol"
            break
        elif "diesel" in text:
            fuel_type = "Diesel"
            break

    # Extract ownership using keyword search
    ownership = None
    for detail in details_list:
        text = detail.text.strip().lower()
        if "1st owner" in text:
            ownership = "1st Owner"
            break
        elif "2nd owner" in text:
            ownership = "2nd Owner"
            break

    # Extract EMI information and remove "EMI from" text
    emi_info = car.find('div', class_='_1Oul-').find('span', class_='_1t1AA').text.strip() if
car.find('div', class_='_1Oul-') else None
    if emi_info:
        emi_info = emi_info.replace('EMI from', '').strip()

    # Extract price details
    price_div = car.find('div', class_='_1Oul- VMjdr')
    current_price = price_div.find('strong', class_='_37WXy').text.strip() if price_div and
price_div.find('strong', class_='_37WXy') else None
    original_price = price_div.find('span', class_='_3hb01').text.strip() if price_div and
price_div.find('span', class_='_3hb01') else None

    # Extract location details only and remove "at"
    location_info = car.find('p', class_='_2rxhF').find('span').text.strip() if car.find('p',
```

```
class_='_2rxhF') and car.find('p', class_='_2rxhF').find('span') else None
    if location_info:
        location_info = location_info.replace('at', '').strip()

    # Store extracted details in a dictionary
    car_details = {
        'Year': year,
        'Car Title': car_title,
        'Car Model': car_model,
        'Mileage': mileage,
        'Fuel Type': fuel_type,
        'Ownership': ownership,
        'EMI Info': emi_info,
        'Current Price': current_price,
        'Original Price': original_price,
        'Location': location_info
    }

    cars_data.append(car_details)

# Step 5: Create a DataFrame using pandas
df = pd.DataFrame(cars_data)

# Step 6: Save the DataFrame to an Excel file
df.to_excel('Agra.xlsx', index=False)

# Step 7: Save the DataFrame to a CSV file
df.to_csv('Agra.csv', index=False)

print("Data has been successfully saved to 'Agra.xlsx' and 'Agra.csv'.")
```

New Delhi:

```
# Step 1: Fetch the webpage content
url = "https://www.cars24.com/buy-used-
car?f=make%3A%3D%3Amahindra%3AOR%3Amake%3A%3D%3Arenault%3AOR%3Amake
%3A%3D%3Ajeep&sort=bestmatch&serveWarrantyCount=true&gaId=2135718455.172570
7831&listingSource=TabFilter&storeCityId=2"
headers = {
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
```

```
(KHTML, like Gecko) Chrome/92.0.4515.131 Safari/537.36"
}

response = requests.get(url, headers=headers)

# Step 2: Parse the HTML content using BeautifulSoup
soup = BeautifulSoup(response.content, 'html.parser')

# Step 3: Find all car entries
car_entries = soup.find_all('div', class_='_7jb8Q _1Ey60')

# Step 4: Loop through each car entry and extract details
cars_data = []

for car in car_entries:
    # Extract car title and model
    car_title = car.find('h3', class_='_2Out2').contents[0].strip() if car.find('h3',
class_='_2Out2') else None
    car_model = car.find('h3', class_='_2Out2').find('span').text.strip() if car.find('h3',
class_='_2Out2').find('span') else None

    # Extract the year from the car title
    year = car_title.split()[0] if car_title else None
    car_title = ' '.join(car_title.split()[1:]) if car_title else None  # Remove the year from the
title

    # Extract mileage, fuel type, and ownership using more flexible methods
    details_list = car.find('ul', class_='_3jRcd').find_all('li') if car.find('ul', class_='_3jRcd') else
[]

    # Use regular expression to find mileage
    mileage = None
    if details_list:
        for detail in details_list:
            if re.search(r'\d+[,.\d]*\s*km', detail.text, re.IGNORECASE):
                mileage = detail.text.strip()
                break

    if mileage and not mileage.lower().endswith('km'):
```

```
        mileage += " km"

    # Extract fuel type using keyword search
    fuel_type = None
    for detail in details_list:
        text = detail.text.strip().lower()
        if "petrol" in text:
            fuel_type = "Petrol"
            break
        elif "diesel" in text:
            fuel_type = "Diesel"
            break

    # Extract ownership using keyword search
    ownership = None
    for detail in details_list:
        text = detail.text.strip().lower()
        if "1st owner" in text:
            ownership = "1st Owner"
            break
        elif "2nd owner" in text:
            ownership = "2nd Owner"
            break

    # Extract EMI information and remove "EMI from" text
    emi_info = car.find('div', class_='_1Oul-').find('span', class_='_1t1AA').text.strip() if
car.find('div', class_='_1Oul-') else None
    if emi_info:
        emi_info = emi_info.replace('EMI from', '').strip()

    # Extract price details
    price_div = car.find('div', class_='_1Oul- VMjdr')
    current_price = price_div.find('strong', class_='_37WXy').text.strip() if price_div and
price_div.find('strong', class_='_37WXy') else None
    original_price = price_div.find('span', class_='_3hb01').text.strip() if price_div and
price_div.find('span', class_='_3hb01') else None

    # Extract location details only and remove "at"
    location_info = car.find('p', class_='_2rxhF').find('span').text.strip() if car.find('p',
```

```
class_='_2rxhF') and car.find('p', class_='_2rxhF').find('span') else None
    if location_info:
        location_info = location_info.replace('at', '').strip()

    # Store extracted details in a dictionary
    car_details = {
        'Year': year,
        'Car Title': car_title,
        'Car Model': car_model,
        'Mileage': mileage,
        'Fuel Type': fuel_type,
        'Ownership': ownership,
        'EMI Info': emi_info,
        'Current Price': current_price,
        'Original Price': original_price,
        'Location': location_info
    }

    cars_data.append(car_details)

# Step 5: Create a DataFrame using pandas
df = pd.DataFrame(cars_data)

# Step 6: Save the DataFrame to an Excel file
df.to_excel('New Delhi.xlsx', index=False)

# Step 7: Save the DataFrame to a CSV file
df.to_csv('New Delhi.csv', index=False)

print("Data has been successfully saved to 'New Delhi.xlsx' and 'New Delhi.csv'.")
```

Combine All Location Details into Single File:

We are now combining all the data's extracted into a new file.ie, cars24_data.csv.

```
import pandas as pd

# Read the Excel files into separate DataFrames
df_ahamedabad = pd.read_excel('/content/Ahamedabad.xlsx')
df_hyderabad = pd.read_excel('/content/Hyderabad.xlsx')
df_indore = pd.read_excel('/content/Indore.xlsx')
```

```
df_agra = pd.read_excel('/content/Agra.xlsx')
df_Coimbatore = pd.read_excel('/content/Coimbatore.xlsx')
df_Kochi = pd.read_excel('/content/Kochi.xlsx')
df_chennai = pd.read_excel('/content/chennai.xlsx')
df_kolkata = pd.read_excel('/content/kolkata.xlsx')
df_patna = pd.read_excel('/content/patna.xlsx')
df_surat = pd.read_excel('/content/surat.xlsx')
df_NewDelhi = pd.read_excel('/content/New Delhi.xlsx')

# Concatenate the DataFrames
combined_df = pd.concat([df_ahamedabad, df_hyderabad, df_indore,
df_agra,df_Coimbatore,df_Kochi,df_chennai,df_kolkata,df_patna,df_surat,df_NewDelhi],
ignore_index=True)

# Save the combined DataFrame to a new Excel file
combined_df.to_csv('cars24_data.csv', index=False)

print("Combined data saved to 'cars24_data.csv'")
```

Data Cleaning:

Now we have to perform data cleaning operations inorder to resolve the issues that will arise because of missing values, duplicate values etc and save it in a file called cars.csv.

```
import pandas as pd

# Read the CSV file into a DataFrame
df = pd.read_csv('/content/cars24_data.csv')

# Forward fill the 'Ownership' and 'Original Price' columns
df[['Ownership', 'Original Price']] = df[['Ownership', 'Original Price']].fillna(method='ffill')

# Save the updated DataFrame back to the CSV file
df.to_csv('cars.csv', index=False)

print("Forward fill and special character removal completed and saved to 'cars.csv'.")
```

Exploratory Data Analysis:

Next we have to perform EDA for certain analysis.

```
import pandas as pd
import re

df=pd.read_csv('/content/cars.csv')
df.head().T
```

	0	1	2	3	4
Year	2018	2021	2022	2022	2021
Car Title	Renault Kwid	Mahindra Thar	Renault TRIBER	Mahindra SCORPIO-N	Mahindra Thar
Car Model	RXL	LX HT PETROL 4WD MT	RXL MT	Z8L PETROL MT 7 STR	LX D AT 4WD HT
Mileage	84,162 km	22,784 km	26,543 km	13,097 km	30,063 km
Fuel Type	Petrol	Petrol	Petrol	Petrol	Diesel
Ownership	1st Owner	1st Owner	2nd Owner	1st Owner	1st Owner
EMI Info	₹5,711/m	₹23,450/m	₹10,680/m	₹35,099/m	₹25,163/m
Current Price	₹2.92L	₹12.32L	₹5.46L	₹18.44L	₹13.22L
Original Price	₹2.96L	₹12.44L	₹6.53L	₹6.53L	₹13.97L
Location	Naroda, Ahmedabad	Karnavi Club, Ahmedabad	Karnavi Club, Ahmedabad	Karnavi Club, Ahmedabad	Karnavi Club, Ahmedabad

Next steps: Generate code with df | View recommended plots | New interactive sheet

Figure 1: df.head()

```
df.isna().sum()
```

	0
Year	0
Car Title	0
Car Model	0
Mileage	0
Fuel Type	0
Ownership	0
EMI Info	0
Current Price	0
Original Price	0
Location	0

dtype: int64

Figure 2: sum()

df['Fuel Type'].value_counts()

Fuel Type	count
Petrol	133
Diesel	38

dtype: int64

Figure 3:Fuel counts

df[df['Fuel Type'].isna()]

```
df[df['Fuel Type'].isna()]
```

Year	Car Title	Car Model	Mileage	Fuel Type	Ownership	EMI Info	Current Price	Original Price	Location

Figure 4: isna()

df.fillna('Petrol',inplace=True)

df.isna().sum()

```
df.isna().sum()
```

	0
Year	0
Car Title	0
Car Model	0
Mileage	0
Fuel Type	0
Ownership	0
EMI Info	0
Current Price	0
Original Price	0
Location	0

dtype: int64

Figure 5:isna().sum()

```
df.info()
```

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 171 entries, 0 to 170
Data columns (total 10 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Year            171 non-null    int64
 1   Car Title       171 non-null    object
 2   Car Model       171 non-null    object
 3   Mileage         171 non-null    object
 4   Fuel Type       171 non-null    object
 5   Ownership       171 non-null    object
 6   EMI Info        171 non-null    object
 7   Current Price   171 non-null    object
 8   Original Price  171 non-null    object
 9   Location        171 non-null    object
dtypes: int64(1), object(9)
memory usage: 13.5+ KB
```

Figure 6:info()

```
df_cate=df.select_dtypes(exclude=['int64',"float64"])
df_cate
```

```
df_cate=df.select_dtypes(exclude=['int64',"float64"])
df_cate
```

	Car Title	Car Model	Mileage	Fuel Type	Ownership	EMI Info	Current Price	Original Price	Location
0	Renault Kwid	RXL	84,162 km	Petrol	1st Owner	₹5,711/m	₹2.92L	₹2.96L	Naroda, Ahmedabad
1	Mahindra Thar	LX HT PETROL 4WD MT	22,784 km	Petrol	1st Owner	₹23,450/m	₹12.32L	₹12.44L	Karnavi Club, Ahmedabad
2	Renault TRIBER	RXL MT	26,543 km	Petrol	2nd Owner	₹10,680/m	₹5.46L	₹6.53L	Karnavi Club, Ahmedabad
3	Mahindra SCORPIO-N	Z8L PETROL MT 7 STR	13,097 km	Petrol	1st Owner	₹35,099/m	₹18.44L	₹6.53L	Karnavi Club, Ahmedabad
4	Mahindra Thar	LX D AT 4WD HT	30,063 km	Diesel	1st Owner	₹25,163/m	₹13.22L	₹13.97L	Karnavi Club, Ahmedabad
...	...	...	...	...	...	...	...	...	...
166	Renault TRIBER	RXT	13,196 km	Petrol	1st Owner	₹13,480/m	₹6.90L	₹7.08L	Sector-18, Noida
167	Renault Kwid	RXT 1.0 AMT (O)	13,568 km	Petrol	1st Owner	₹7,540/m	₹3.86L	₹4.48L	Metro Walk, Rohini, New Delhi
168	Jeep Compass	SPORT PLUS 1.4 PETROL	76,041 km	Petrol	1st Owner	₹20,269/m	₹10.65L	₹12.40L	M3M Urbana, Golf Course Ext., Gurugram
169	Mahindra XUV500	W5	45,630 km	Diesel	1st Owner	₹21,977/m	₹10.11L	₹10.85L	Sector-18, Noida
170	Mahindra XUV300	W6 1.2 PETROL	40,924 km	Petrol	1st Owner	₹13,314/m	₹6.81L	₹7.46L	Metro Walk, Rohini, New Delhi

171 rows × 9 columns

Figure 7:df_cate

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

for i in df_cate.columns:
    df[i]=le.fit_transform(df_cate[i])
    print(i)
df
```

```
Car Title
Car Model
Mileage
Fuel Type
Ownership
EMI Info
Current Price
Original Price
Location
```

	Year	Car Title	Car Model	Mileage	Fuel Type	Ownership	EMI Info	Current Price	Original Price	Location
0	2018	14	36	156	1	0	108	42	28	12
1	2021	8	31	45	1	0	85	21	15	7
2	2022	15	40	56	1	1	4	104	84	7
3	2022	5	70	16	1	0	100	36	84	7
4	2021	8	30	65	0	0	90	26	20	7
...	...	...	...	...	...	...	...	...	...	...
166	2022	15	42	17	1	0	39	135	94	16
167	2019	14	47	19	1	0	128	64	43	11
168	2020	0	58	151	1	0	69	7	14	9
169	2019	10	62	111	0	0	79	1	4	16

Figure 8:Label encoder

```
from sklearn.preprocessing import LabelEncoder

# Initialize a dictionary to store LabelEncoders for each column
label_encoders = {}

# Apply Label Encoding to each categorical column
for column in df_cate.columns:
    le = LabelEncoder()
    df[column] = le.fit_transform(df_cate[column])  # Encode the column

    # Save the LabelEncoder object for later use
    label_encoders[column] = le

    # Print the mapping for each column
    print(f"Mappings for column '{column}':")
    for index, class_label in enumerate(le.classes_):
        print(f"{class_label} -> {index}")
    print("\n")
```

```
Mappings for column 'Car Title':
Jeep Compass -> 0
Mahindra BOLERO NEO -> 1
Mahindra Bolero -> 2
Mahindra KUV 100 NXT -> 3
Mahindra Kuv100 -> 4
Mahindra SCORPIO-N -> 5
Mahindra Scorpio -> 6
Mahindra TUV300 -> 7
Mahindra Thar -> 8
Mahindra XUV300 -> 9
Mahindra XUV500 -> 10
Mahindra XUV700 -> 11
Renault Duster -> 12
Renault Kiger -> 13
Renault Kwid -> 14
Renault TRIBER -> 15


Mappings for column 'Car Model':
110 PS RXZ 4X2 AMT DIESEL -> 0
110 PS RXZ DIESEL -> 1
AX 5 D MT 5 STR -> 2
AX 5 P AT 5 STR -> 3
AX 5 P MT 7 STR -> 4
AX 7 LUXURY D AWD AT 7 STR -> 5
B6 -> 6
CLIMBER 1.0 -> 7
CLIMBER 1.0 (O) -> 8
```

Figure 9: Label encoder names

```
df.corr()
```

```
[ ] df.corr()
```

	Year	Car Title	Car Model	Mileage	Fuel Type	Ownership	EMI Info	Current Price	Original Price	Location
Year	1.000000	0.276006	0.117843	-0.373294	0.265129	-0.288884	-0.027325	0.133108	0.050403	0.121281
Car Title	0.276006	1.000000	0.149456	-0.069312	0.510546	-0.129945	0.002424	0.317333	0.253632	0.129641
Car Model	0.117843	0.149456	1.000000	-0.054615	0.024089	-0.123925	-0.240200	0.166525	0.095232	0.057832
Mileage	-0.373294	-0.069312	-0.054615	1.000000	-0.143050	0.149757	-0.149635	-0.044642	-0.052100	-0.095188
Fuel Type	0.265129	0.510546	0.024089	-0.143050	1.000000	-0.113592	0.111005	0.252935	0.153584	0.146604
Ownership	-0.288884	-0.129945	-0.123925	0.149757	-0.113592	1.000000	-0.043634	-0.065906	-0.033249	0.061378
EMI Info	-0.027325	0.002424	-0.240200	-0.149635	0.111005	-0.043634	1.000000	-0.405786	-0.306798	0.048471
Current Price	0.133108	0.317333	0.166525	-0.044642	0.252935	-0.065906	-0.405786	1.000000	0.567831	0.100122
Original Price	0.050403	0.253632	0.095232	-0.052100	0.153584	-0.033249	-0.306798	0.567831	1.000000	0.124313
Location	0.121281	0.129641	0.057832	-0.095188	0.146604	0.061378	0.048471	0.100122	0.124313	1.000000

Figure 10:corr()

Model Building for Predictive Analysis:

In addition to this project goal, we also builded a predictive analysis model to predict the price of the car with respect to the factors, such as Year, Car Title, Car Model, Mileage, Fuel Type, Ownership etc.

```
!pip install lazypredict
from lazypredict.Supervised import LazyRegressor
from lazypredict.Supervised import LazyRegressor
```

```
X = df.drop({'Current Price','Location'}, axis=1)
y = df['Current Price']

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

reg = LazyRegressor(verbose=0, ignore_warnings=False, custom_metric=None)
models, predictions = reg.fit(X_train, X_test, y_train, y_test)

print(models)
```

```
from sklearn.model_selection import train_test_split
from xgboost import XGBRegressor
from sklearn.metrics import mean_squared_error

X = df.drop({'Current Price','Location'}, axis=1)
y = df['Current Price']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

model = XGBRegressor()

model.fit(X_train, y_train)
y_pred = model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

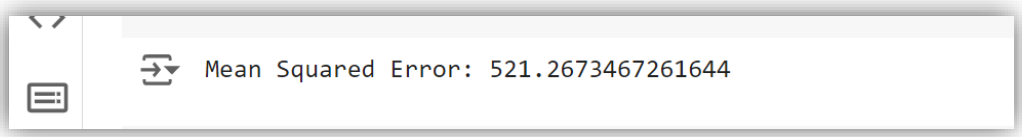

Figure 11:MSE

```
y_pred = model.predict([[2018,12,44,171,0,1,2.92,3.3]])
y_pred
```

```
array([103.94069], dtype=float32)
```

Figure 12:Predict

Data Visualization:

- Correlation

```
import matplotlib.pyplot as plt
import seaborn as sns
sns.heatmap(df.corr(),annot=True)
```

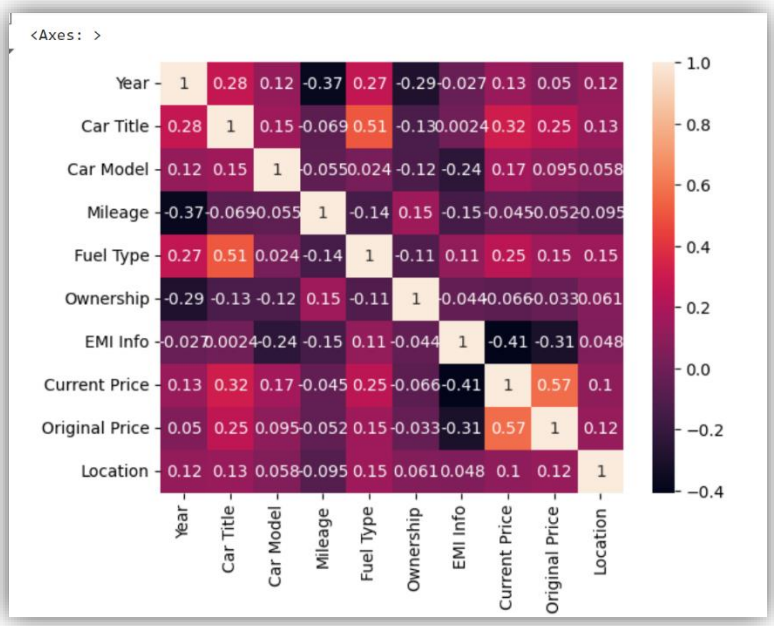

Figure 13:Correlation

- Univariate

```
df['Car Title'].value_counts().plot(kind='bar')
plt.xlabel('Car Title')
plt.ylabel('Highest Selling')
plt.show()
```

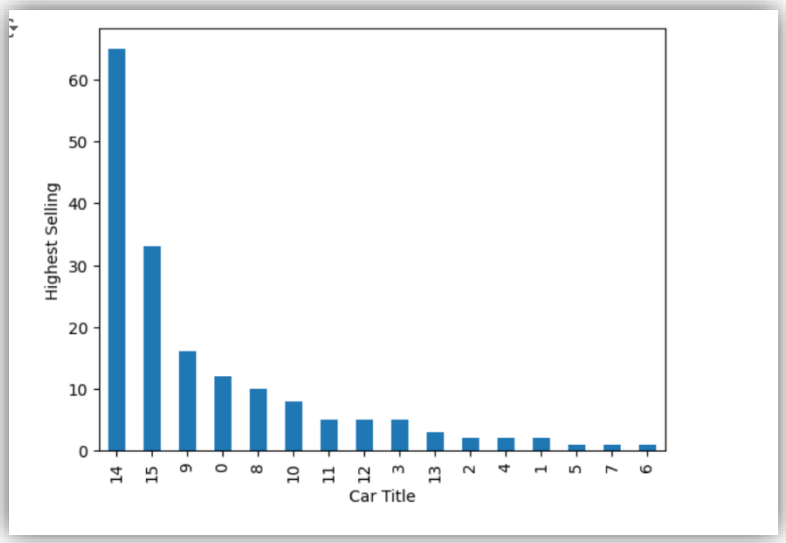

Figure 14:Univariate

- Price Distribution

```
sns.histplot(df['Current Price'],kde=True)
```

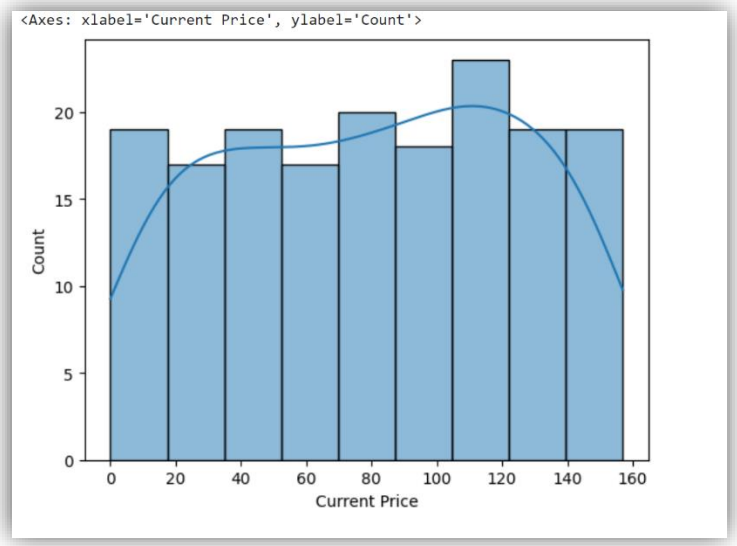

Figure 15:Price distribution

```
df['Current Price'].mean(),df['Current Price'].median()
```

- Fuel

```
import plotly.express as px
px.bar(df['Fuel Type'].value_counts())
```

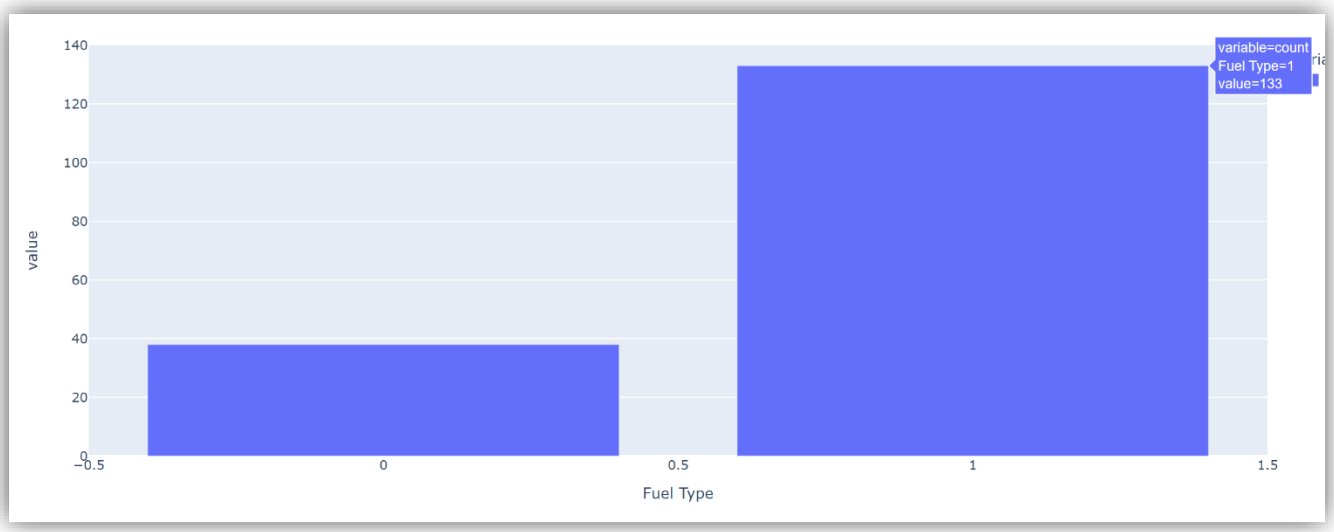

Figure 16: Fuel type vs count

- Bivariate Analysis

Vehicle Listing for different number of previous owners

```
sns.barplot(x=df['Ownership'],y=df['Current Price'],errorbar=None
```

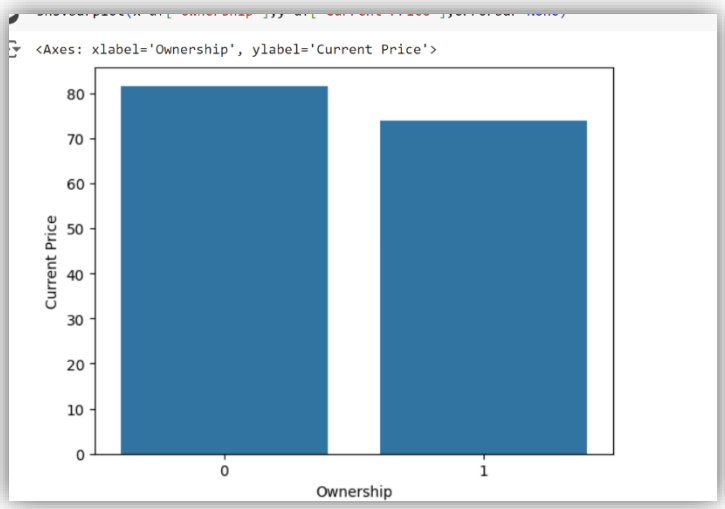

Figure 17:Vehicle listing based on ownership

Vehicle listing from different locations

```
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt

# Assuming 'df' is the DataFrame with the encoded "Location" column
# Assuming 'label_encoders' is the dictionary with all LabelEncoders created earlier

# Retrieve the LabelEncoder for the "Location" column
location_le = label_encoders['Location']

# Create a mapping from encoded values to original names
location_mapping = dict(enumerate(location_le.classes_))

# Replace encoded "Location" values with their original names in a new column for plotting
df['Location Name'] = df['Location'].map(location_mapping)

# Plotting using the original "Location" names
df.groupby(['Location  Name'])['Year'].count().sort_values(ascending=False).plot(kind='bar',
figsize=(10, 6))
plt.ylabel('Number of Vehicles', fontsize=12)
plt.xlabel('Location', fontsize=12)
plt.title('Vehicle Listing from Different Locations')
plt.show()
```

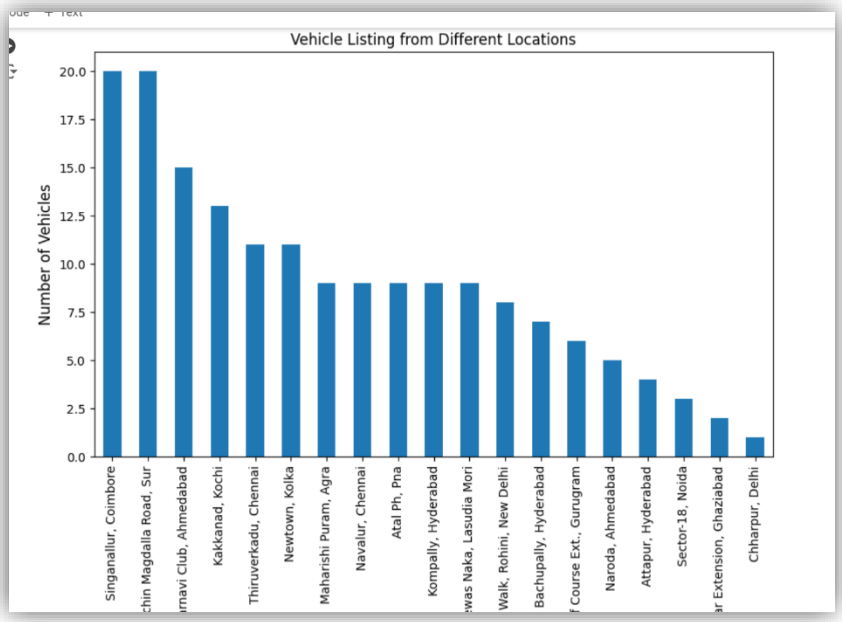

Figure 18: vehicle listing vs location

Comparing Year and Price

```
sns.lineplot(x=df['Year'],y=df['Current Price'],errorbar=None)
```

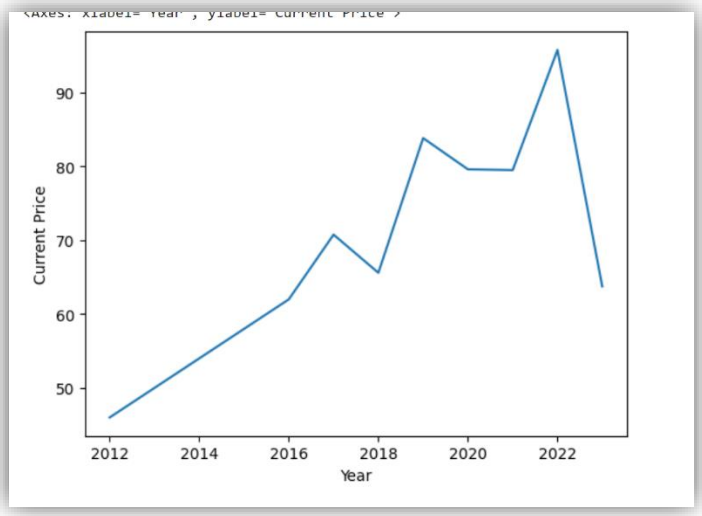

Figure 19: Year vs Price

How will fuel type impact resale price

```
sns.boxplot(x=df['Fuel Type'],y=df['Current Price'])
```

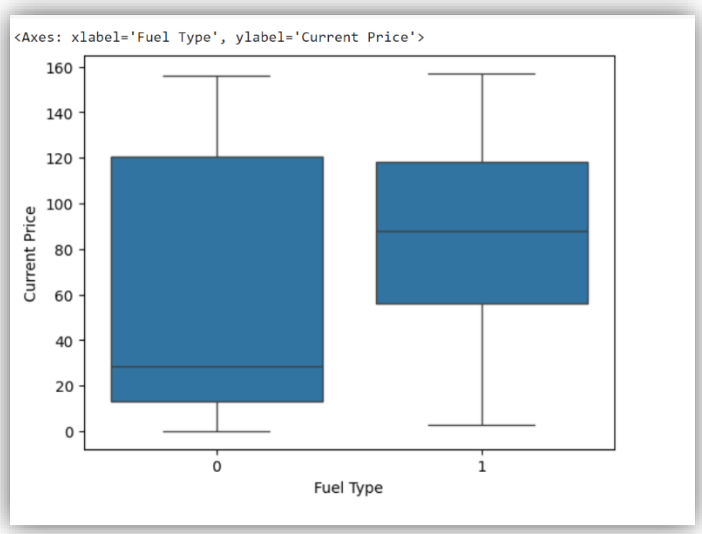

Figure 20: Fuel type vs price

- Multivariate Analysis

```
sns.barplot(x=df['Year'],y=df['Current Price'],errorbar=None,hue=df['Ownership'])
```

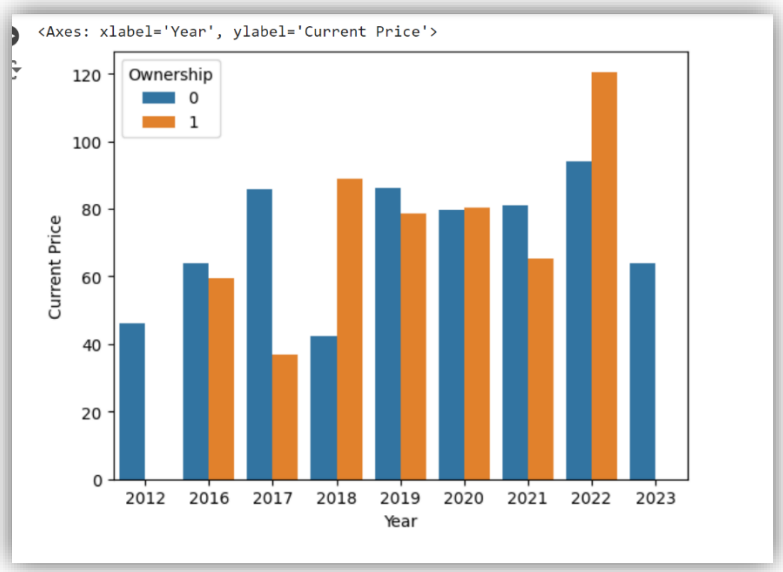

Figure 21: Multivariate Analysis

Output

We extracted car's data for 11 locations. The extracted data was send to cleaning team , later the missing values are resolved.

Extracted Data:

	A	B	C	D	E	F	G	H	I	J	K	L
1	Year	Car Title	Car Model	Mileage	Fuel Type	Ownershi	EMI Info	Current Pr	Original P	Location		
2	2018	Renault K	RXL	84,162 km	Petrol	1st Owner	â,¹5,711/m	â,¹2.92L	â,¹2.96L	Naroda, Ahmedabad		
3	2022	Renault TI	RXL MT	26,543 km	Petrol	2nd Owne	â,¹10,680/	â,¹5.46L	â,¹6.53L	Karnavi Club, Ahmedabad		
4	2022	Mahindra	Z8L PETRC	13,097 km	Petrol	1st Owner	â,¹35,803/	â,¹18.81L		Karnavi Club, Ahmedabad		
5	2021	Mahindra	LX D AT 4V	30,063 km	Diesel	1st Owner	â,¹25,753/	â,¹13.53L	â,¹13.97L	Karnavi Club, Ahmedabad		
6	2017	Renault K	CLIMBER 1	33,755 km	Petrol		â,¹5,340/m	â,¹2.73L	â,¹3.10L	Naroda, Ahmedabad		
7	2023	Mahindra	B6	14,923 km	Diesel	1st Owner	â,¹19,176/	â,¹10.07L	â,¹10.31L	Karnavi Club, Ahmedabad		
8	2020	Mahindra	W7	32,491 km	Diesel	1st Owner	â,¹20,918/	â,¹10.99L	â,¹11.51L	Karnavi Club, Ahmedabad		
9	2021	Renault TI	RXT	42,589 km	Petrol	1st Owner	â,¹10,569/	â,¹5.41L	â,¹6.15L	Naroda, Ahmedabad		
10	2022	Mahindra	AX 5 P AT	6,492 km	Petrol	1st Owner	â,¹33,919/	â,¹17.82L	â,¹19.33L	Naroda, Ahmedabad		
11	2021	Mahindra	LX D 4*4 M	52,145 km	Diesel	2nd Owne	â,¹21,451/	â,¹11.27L	â,¹11.98L	Karnavi Club, Ahmedabad		
12	2019	Renault TI	RXZ	53,563 km	Petrol	1st Owner	â,¹11,023/	â,¹5.64L	â,¹6.23L	Karnavi Club, Ahmedabad		
13	2019	Renault K	RXT 1.0 (O	53,607 km	Petrol	2nd Owne	â,¹6,060/m	â,¹3.10L	â,¹3.56L	Karnavi Club, Ahmedabad		
14	2022	Renault K	RXL 0.8 (O	29,056 km	Petrol	1st Owner	â,¹8,309/m	â,¹4.25L	â,¹4.79L	Karnavi Club, Ahmedabad		
15	2021	Mahindra	LX PETROL	46,448 km	Petrol	1st Owner	â,¹24,751/	â,¹13.00L	â,¹14.08L	Naroda, Ahmedabad		
16	2021	Renault K	RXT 0.8	20,267 km	Petrol	1st Owner	â,¹8,618/m	â,¹4.41L	â,¹4.71L	Karnavi Club, Ahmedabad		
17	2018	Jeep Com	LIMITED ((	57,894 km	Petrol	1st Owner	â,¹20,347/	â,¹10.69L		Karnavi Club, Ahmedabad		
18	2022	Mahindra	W8(O) 1.2	11,262 km	Petrol	1st Owner	â,¹21,147/	â,¹11.11L	â,¹12.60L	Naroda, Ahmedabad		
19	2022	Mahindra	W8 (O) 1.2	12,801 km	Petrol	1st Owner	â,¹21,499/	â,¹11.29L	â,¹12.76L	Naroda, Ahmedabad		
20	2022	Mahindra	W8 (O) 1.5	15,946 km	Diesel	1st Owner	â,¹23,092/	â,¹12.13L	â,¹13.88L	Karnavi Club, Ahmedabad		
21	2021	Mahindra	LX PETROL	52,389 km	Petrol	1st Owner	â,¹24,916/	â,¹13.09L	â,¹14.87L	Karnavi Club, Ahmedabad		
22	2019	Mahindra	W7	79,470 km	Diesel	1st Owner	â,¹20,880/	â,¹10.97L		Kompally, Hyderabad		
23	2022	Mahindra	AX 5 D MT	29,358 km	Diesel	1st Owner	â,¹35,031/	â,¹18.40L		Kompally, Hyderabad		
24	2021	Renault K	CLIMBER 1	7,391 km	Petrol	1st Owner	â,¹9,189/m	â,¹4.70L	â,¹5.36L	Kompally, Hyderabad		
25	2022	Renault K	CLIMBER A	25,787 km	Petrol	1st Owner	â,¹11,290/	â,¹5.78L	â,¹6.49L	Attapur, Hyderabad		
26	2022	Renault K	CLIMBER A	9,601 km	Petrol	1st Owner	â,¹11,290/	â,¹5.78L	â,¹6.52L	Kompally, Hyderabad		
27	2018	Jeep Com	LIMITED 2.	1,22,168 k	Diesel	2nd Owne	â,¹23,567/	â,¹10.84L	â,¹13.85L	Kompally, Hyderabad		
28	2021	Mahindra	LX P 4WD	32,709 km	Petrol	2nd Owne	â,¹25,106/	â,¹13.19L	â,¹13.89L	Kompally, Hyderabad		
29	2018	Renault K	CLIMBER 1	87,072 km	Petrol	1st Owner	â,¹7,077/m	â,¹3.62L	â,¹4.15L	Bachupally, Hyderabad		
30	2022	Renault TI	RXT AMT	29,617 km	Petrol	1st Owner	â,¹13,451/	â,¹6.88L		Kompally, Hyderabad		
31	2019	Renault K	CLIMBER 1	51,288 km	Petrol	2nd Owne	â,¹6,802/m	â,¹3.48L	â,¹4.14L	Kompally, Hyderabad		
32	2020	Mahindra	LX P 4WD	24,982 km	Petrol	1st Owner	â,¹25,328/	â,¹13.31L	â,¹13.84L	Attapur, Hyderabad		

Figure 22: Output Screenshot 1

	A	B	C	D	E	F	G	H	I	J	K	L
33	2021	Renault K	RXL	45,901 km	Petrol	1st Owner	â,¹7,436/n	â,¹3.80L	â,¹4.33L	Bachupally, Hyderabad		
34	2019	Mahindra	K8 P 6 STR	44,303 km	Petrol	2nd Owne	â,¹11,337/	â,¹5.80L	â,¹6.72L	Kompally, Hyderabad		
35	2021	Mahindra	W8 (O) 1.5	38,401 km	Diesel	1st Owner	â,¹21,756/	â,¹11.43L	â,¹11.55L	Bachupally, Hyderabad		
36	2021	Mahindra	LX D 4WD	24,914 km	Diesel	2nd Owne	â,¹26,438/	â,¹13.89L		Bachupally, Hyderabad		
37	2020	Renault TI	RXZ AMT	19,391 km	Petrol	1st Owner	â,¹11,789/	â,¹6.03L	â,¹6.39L	Bachupally, Hyderabad		
38	2020	Renault D	RXZ 1.3 TU	1,05,742 k	Petrol	1st Owner	â,¹17,128/	â,¹7.70L	â,¹10.80L	Kompally, Hyderabad		
39	2022	Renault TI	RXZ AMT	37,502 km	Petrol	1st Owner	â,¹14,272/	â,¹7.30L	â,¹7.99L	Attapur, Hyderabad		
40	2021	Renault K	CLIMBER 1	44,865 km	Petrol	1st Owner	â,¹9,756/n	â,¹4.99L	â,¹5.58L	Bachupally, Hyderabad		
41	2020	Renault K	CLIMBER 1	18,513 km	Petrol	2nd Owne	â,¹8,778/n	â,¹4.49L	â,¹5.27L	Kompally, Hyderabad		
42	2020	Renault K	CLIMBER 1	47,560 km	Petrol	2nd Owne	â,¹7,247/n	â,¹3.71L		Dewas Naka, Lasudia Mori		
43	2017	Renault K	CLIMBER 1	33,697 km	Petrol	2nd Owne	â,¹6,814/n	â,¹3.49L	â,¹4.31L	Dewas Naka, Lasudia Mori		
44	2020	Renault K	RXT 0.8	75,196 km	Petrol	1st Owner	â,¹7,429/n	â,¹3.80L	â,¹4.60L	Dewas Naka, Lasudia Mori		
45	2019	Mahindra	W8 (O) 1.5	43,143 km	Diesel	1st Owner	â,¹16,978/	â,¹8.92L	â,¹10.27L	Dewas Naka, Lasudia Mori		
46	2023	Mahindra	B4	6,789 km	Diesel	1st Owner	â,¹16,293/	â,¹8.56L	â,¹9.92L	Dewas Naka, Lasudia Mori		
47	2020	Renault K	RXL	32,402 km	Petrol	1st Owner	â,¹7,248/n	â,¹3.71L	â,¹4.15L	Dewas Naka, Lasudia Mori		
48	2022	Mahindra	W8 (O) 1.2	23,601 km	Petrol	1st Owner	â,¹18,558/	â,¹9.75L	â,¹11.66L	Dewas Naka, Lasudia Mori		
49	2020	Renault K	RXT 1.0 (O	54,200 km	Petrol	1st Owner	â,¹7,468/n	â,¹3.82L	â,¹4.70L	Dewas Naka, Lasudia Mori		
50	2021	Renault K	RXT 1.0 AM	4,831 km	Petrol	1st Owner	â,¹9,814/n	â,¹5.02L		Dewas Naka, Lasudia Mori		
51	2022	Mahindra	LX HT PETI	32,395 km	Petrol	1st Owner	â,¹23,583/	â,¹12.39L		Dewas Naka, Lasudia Mori		
52	2018	Jeep Com	LONGITUD	40,813 km	Diesel	1st Owner	â,¹21,128/	â,¹11.10L	â,¹12.90L	Dewas Naka, Lasudia Mori		
53	2019	Mahindra	W8 (O) 1.5	56,293 km	Diesel	1st Owner	â,¹16,940/	â,¹8.90L	â,¹10.46L	Dewas Naka, Lasudia Mori		
54	2022	Renault K	RXL	13,379 km	Petrol	1st Owner	â,¹8,739/n	â,¹4.47L		Maharishi Puram, Agra		
55	2021	Renault K	CLIMBER 1	14,753 km	Petrol	1st Owner	â,¹8,962/n	â,¹4.58L	â,¹5.22L	Maharishi Puram, Agra		
56	2019	Mahindra	W5	23,502 km	Diesel	2nd Owne	â,¹17,492/	â,¹9.19L	â,¹10.22L	Maharishi Puram, Agra		
57	2023	Mahindra	W8 (O) 1.2	14,513 km	Petrol	1st Owner	â,¹21,946/	â,¹11.53L	â,¹12.11L	Maharishi Puram, Agra		
58	2018	Renault D	110 PS RX	60,158 km	Diesel	1st Owner	â,¹13,783/	â,¹7.05L	â,¹7.69L	Maharishi Puram, Agra		
59	2019	Mahindra	W8 1.5 DIE	70,894 km	Diesel	1st Owner	â,¹16,160/	â,¹8.49L	â,¹8.94L	Maharishi Puram, Agra		
60	2021	Renault K	CLIMBER 1	12,145 km	Petrol	1st Owner	â,¹8,465/n	â,¹4.33L		Maharishi Puram, Agra		
61	2018	Jeep Com	LONGITUD	74,364 km	Diesel	1st Owner	â,¹21,128/	â,¹11.10L	â,¹12.13L	Singanallur, Coimbore		
62	2020	Renault TI	RXZ	70,648 km	Petrol	2nd Owne	â,¹12,004/	â,¹6.14L	â,¹7.07L	Singanallur, Coimbore		
63	2021	Renault Ki	RXT AMT [	43,390 km	Petrol	1st Owner	â,¹12,725/	â,¹6.51L	â,¹7.85L	Singanallur, Coimbore		
64	2017	Jeep Com	LONGITUD	1,17,741 k	Diesel	2nd Owne	â,¹22,721/	â,¹10.45L		Singanallur, Coimbore		

Figure 23: Output Screenshot 2

	A	B	C	D	E	F	G	H	I	J	K	L
64	2017	Jeep Com	LONGITUD	1,17,741 k	Diesel	2nd Owne	â,¹22,721/	â,¹10.45L		Singanallur, Coimbore		
65	2020	Renault K	RXT 1.0 AN	24,397 km	Petrol	2nd Owne	â,¹8,891/m	â,¹4.55L	â,¹5.34L	Singanallur, Coimbore		
66	2021	Mahindra	W8 (O) 1.2	42,416 km	Petrol	1st Owner	â,¹20,043/	â,¹10.53L	â,¹11.17L	Singanallur, Coimbore		
67	2022	Renault K	CLIMBER N	9,383 km	Petrol	1st Owner	â,¹10,853/	â,¹5.55L	â,¹5.71L	Singanallur, Coimbore		
68	2022	Mahindra	W8 (O) 1.2	14,084 km	Petrol	1st Owner	â,¹22,346/	â,¹11.74L	â,¹12.82L	Singanallur, Coimbore		
69	2022	Renault TI	RXT AMT	19,992 km	Petrol	2nd Owne	â,¹13,870/	â,¹7.09L	â,¹8.03L	Singanallur, Coimbore		
70	2021	Renault TI	RXZ	19,318 km	Petrol	1st Owner	â,¹12,962/	â,¹6.63L	â,¹7.64L	Singanallur, Coimbore		
71	2020	Renault K	RXT 1.0 AN	23,970 km	Petrol	1st Owner	â,¹9,086/m	â,¹4.65L	â,¹5.50L	Singanallur, Coimbore		
72	2022	Renault K	RXT 1.0	3,686 km	Petrol	1st Owner	â,¹9,892/m	â,¹5.06L	â,¹5.63L	Singanallur, Coimbore		
73	2019	Mahindra	K2+ P 6 ST	23,440 km	Petrol	1st Owner	â,¹9,208/m	â,¹4.71L		Singanallur, Coimbore		
74	2019	Renault TI	RXZ	67,965 km	Petrol	1st Owner	â,¹11,965/	â,¹6.12L		Singanallur, Coimbore		
75	2021	Renault TI	RXZ	27,357 km	Petrol	2nd Owne	â,¹13,233/	â,¹6.77L		Singanallur, Coimbore		
76	2017	Renault K	RXT 1.0	64,691 km	Petrol	2nd Owne	â,¹6,537/m	â,¹3.34L		Singanallur, Coimbore		
77	2017	Renault K	RXL 1.0 AN	33,168 km	Petrol	1st Owner	â,¹7,370/m	â,¹3.77L		Singanallur, Coimbore		
78	2019	Renault TI	RXL MT	42,093 km	Petrol	2nd Owne	â,¹11,010/	â,¹5.63L		Singanallur, Coimbore		
79	2021	Renault TI	RXT	39,834 km	Petrol	1st Owner	â,¹11,632/	â,¹5.95L	â,¹6.71L	Singanallur, Coimbore		
80	2021	Renault D	RXZ 1.3 TU	56,328 km	Petrol	1st Owner	â,¹16,754/	â,¹8.80L	â,¹11.45L	Singanallur, Coimbore		
81	2020	Mahindra	W8 (O) 1.2	56,356 km	Petrol	1st Owner	â,¹17,654/	â,¹9.28L	â,¹10.51L	Kakkanad, Kochi		
82	2021	Jeep Com	LIMITED P	1,01,721 k	Diesel	1st Owner	â,¹35,832/	â,¹16.48L	â,¹20.44L	Kakkanad, Kochi		
83	2021	Renault K	RXT 1.0 (O	8,608 km	Petrol	1st Owner	â,¹8,274/m	â,¹4.23L	â,¹4.86L	Kakkanad, Kochi		
84	2020	Renault TI	RXT AMT	44,098 km	Petrol	1st Owner	â,¹11,593/	â,¹5.93L	â,¹6.61L	Kakkanad, Kochi		
85	2020	Renault TI	RXZ AMT	65,378 km	Petrol	1st Owner	â,¹12,238/	â,¹6.26L	â,¹6.93L	Kakkanad, Kochi		
86	2022	Renault K	RXT 1.0 AN	3,052 km	Petrol	1st Owner	â,¹11,046/	â,¹5.65L		Kakkanad, Kochi		
87	2021	Renault K	RXT 1.0 (O	18,721 km	Petrol	1st Owner	â,¹8,802/m	â,¹4.50L		Kakkanad, Kochi		
88	2020	Mahindra	W6 1.2 PE	33,032 km	Petrol	1st Owner	â,¹14,860/	â,¹7.60L		Kakkanad, Kochi		
89	2015	Mahindra	W8	86,657 km	Diesel	1st Owner	â,¹16,604/	â,¹7.46L		Kakkanad, Kochi		
90	2017	Jeep Com	LIMITED 2.	81,512 km	Diesel	1st Owner	â,¹21,604/	â,¹11.35L	â,¹12.44L	Kakkanad, Kochi		
91	2016	Renault K	RXL	39,327 km	Petrol	2nd Owne	â,¹5,172/m	â,¹2.65L		Kakkanad, Kochi		
92	2019	Renault TI	RXZ	55,494 km	Petrol	1st Owner	â,¹10,783/	â,¹5.52L		Kakkanad, Kochi		
93	2016	Renault K	RXL	39,453 km	Petrol	1st Owner	â,¹5,049/m	â,¹2.58L		Kakkanad, Kochi		
94	2019	Renault K	RXT 1.0 (O	50,692 km	Petrol	1st Owner	â,¹7,734/m	â,¹3.96L	â,¹5.10L	Thiruverkadu, Chennai		
95	2021	Renault K	RXT 1.0 AN	40,777 km	Petrol	1st Owner	â,¹9,160/m	â,¹4.69L	â,¹5.30L	Navalur, Chennai		

cars24_data

Ready

Figure 24: Output Screenshot 3

	A	B	C	D	E	F	G	H	I	J	K	L
95	2021	Renault K	RXT 1.0 AN	40,777 km	Petrol	1st Owner	â,¹9,160/m	â,¹4.69L	â,¹5.30L	Navalur, Chennai		
96	2018	Jeep Com	LIMITED P	58,050 km	Petrol	1st Owner	â,¹24,605/	â,¹12.93L		Thiruverkadu, Chennai		
97	2021	Mahindra	B6 (O)	80,481 km	Diesel	2nd Owne	â,¹18,473/	â,¹9.71L	â,¹10.10L	Thiruverkadu, Chennai		
98	2021	Renault TI	RXZ AMT	33,137 km	Petrol	1st Owner	â,¹13,314/	â,¹6.81L	â,¹7.74L	Navalur, Chennai		
99	2019	Mahindra	W7	1,09,163 k	Diesel	1st Owner	â,¹25,308/	â,¹11.64L	â,¹12.90L	Thiruverkadu, Chennai		
100	2019	Mahindra	K4+ P 6 ST	40,510 km	Petrol	2nd Owne	â,¹9,443/m	â,¹4.83L	â,¹5.50L	Thiruverkadu, Chennai		
101	2020	Mahindra	W6 1.2 PE	45,546 km	Petrol	1st Owner	â,¹15,523/	â,¹7.94L	â,¹8.18L	Navalur, Chennai		
102	2020	Renault K	RXT 0.8	11,652 km	Petrol	1st Owner	â,¹8,016/m	â,¹4.10L	â,¹4.52L	Navalur, Chennai		
103	2019	Renault K	RXT 1.0 AN	33,894 km	Petrol	1st Owner	â,¹8,226/m	â,¹4.21L	â,¹4.35L	Thiruverkadu, Chennai		
104	2022	Mahindra	W8 (O) 1.5	14,420 km	Diesel	1st Owner	â,¹22,651/	â,¹11.90L	â,¹12.27L	Thiruverkadu, Chennai		
105	2020	Mahindra	W11 (O) A	81,766 km	Diesel	2nd Owne	â,¹27,980/	â,¹14.70L	â,¹17.51L	Thiruverkadu, Chennai		
106	2022	Renault TI	RXT	38,283 km	Petrol	1st Owner	â,¹13,236/	â,¹6.77L	â,¹7.28L	Navalur, Chennai		
107	2021	Renault K	CLIMBER 1	10,760 km	Petrol	1st Owner	â,¹9,247/m	â,¹4.73L	â,¹5.27L	Thiruverkadu, Chennai		
108	2021	Renault K	CLIMBER 1	21,558 km	Petrol	1st Owner	â,¹9,599/m	â,¹4.91L	â,¹5.65L	Thiruverkadu, Chennai		
109	2023	Renault K	RXT 1.0 AN	14,079 km	Petrol	1st Owner	â,¹11,215/	â,¹5.74L	â,¹6.30L	Navalur, Chennai		
110	2021	Renault K	CLIMBER 1	32,356 km	Petrol	1st Owner	â,¹9,088/m	â,¹4.65L	â,¹5.10L	Thiruverkadu, Chennai		
111	2021	Renault K	CLIMBER 1	24,092 km	Petrol	1st Owner	â,¹9,565/m	â,¹4.89L	â,¹5.46L	Thiruverkadu, Chennai		
112	2021	Renault K	RXT 1.0 (O	49,249 km	Petrol	1st Owner	â,¹8,446/m	â,¹4.32L	â,¹4.92L	Navalur, Chennai		
113	2022	Mahindra	W8 (O) 1.5	14,815 km	Diesel	1st Owner	â,¹22,175/	â,¹11.65L	â,¹12.79L	Thiruverkadu, Chennai		
114	2017	Mahindra	K4+ P 6 ST	49,462 km	Petrol	1st Owner	â,¹7,154/m	â,¹3.66L	â,¹4.08L	Newtown, Kolka		
115	2019	Renault K	CLIMBER 1	19,846 km	Petrol	1st Owner	â,¹7,214/m	â,¹3.69L	â,¹4.28L	Newtown, Kolka		
116	2020	Renault K	RXT 0.8	13,648 km	Petrol	2nd Owne	â,¹6,109/m	â,¹3.12L	â,¹3.90L	Newtown, Kolka		
117	2019	Renault K	CLIMBER 1	20,242 km	Petrol	1st Owner	â,¹7,370/m	â,¹3.77L	â,¹4.30L	Newtown, Kolka		
118	2022	Renault TI	RXT	5,750 km	Petrol	1st Owner	â,¹12,045/	â,¹6.16L	â,¹6.75L	Newtown, Kolka		
119	2021	Renault TI	RXZ	29,603 km	Petrol	1st Owner	â,¹11,241/	â,¹5.75L	â,¹6.53L	Newtown, Kolka		
120	2021	Renault K	RXL 1.0 AN	10,138 km	Petrol	1st Owner	â,¹8,250/m	â,¹4.22L	â,¹4.96L	Newtown, Kolka		
121	2019	Mahindra	K8 P 6 STR	4,800 km	Petrol	2nd Owne	â,¹10,362/	â,¹5.30L	â,¹6.38L	Newtown, Kolka		
122	2021	Renault TI	RXZ AMT [	12,708 km	Petrol	1st Owner	â,¹13,928/	â,¹7.12L		Newtown, Kolka		
123	2021	Renault TI	RXZ AMT [	51,770 km	Petrol	1st Owner	â,¹12,180/	â,¹6.23L	â,¹7.65L	Newtown, Kolka		
124	2013	Mahindra	W8	51,598 km	Diesel	1st Owner	â,¹18,600/	â,¹5.60L		Atal Ph, Pna		
125	2020	Renault K	RXT 0.8	31,108 km	Petrol	1st Owner	â,¹7,136/m	â,¹3.65L	â,¹4.45L	Atal Ph, Pna		
126	2012	Mahindra	SLX BS IV	69,648 km	Diesel	1st Owner	â,¹15,299/	â,¹3.25L		Atal Ph, Pna		

cars24_data

Ready

Figure 25: Output Screenshot4

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
128	2016	Mahindra	T8	1,14,888 k	Diesel	2nd Owne	â,¹11,990/	â,¹5.39L		Atal Ph, Pna				
129	2022	Mahindra	N 8	35,346 km	Diesel	1st Owner	â,¹17,492/	â,¹9.19L		Atal Ph, Pna				
130	2019	Renault K	RXT 1.0 AN	71,248 km	Petrol	2nd Owne	â,¹6,256/m	â,¹3.20L	â,¹3.55L	Atal Ph, Pna				
131	2022	Mahindra	W6 1.2 PE	12,500 km	Petrol	1st Owner	â,¹16,198/	â,¹8.51L	â,¹9.79L	Atal Ph, Pna				
132	2020	Renault K	RXL	20,184 km	Petrol	1st Owner	â,¹6,530/m	â,¹3.34L		Atal Ph, Pna				
133	2019	Renault TI	RXZ	59,437 km	Petrol	1st Owner	â,¹10,127/	â,¹5.18L		Atal Ph, Pna				
134	2022	Renault TI	RXZ	5,381 km	Petrol	1st Owner	â,¹12,444/	â,¹6.37L		Atal Ph, Pna				
135	2018	Mahindra	W11 AT	87,832 km	Diesel	2nd Owne	â,¹19,377/	â,¹10.18L		Sachin Magdalla Road, Sur				
136	2023	Renault K	CLIMBER A	4,588 km	Petrol	1st Owner	â,¹10,773/	â,¹5.51L	â,¹5.99L	Sachin Magdalla Road, Sur				
137	2016	Renault K	RXL	58,987 km	Petrol	2nd Owne	â,¹4,477/m	â,¹2.29L		Sachin Magdalla Road, Sur				
138	2017	Mahindra	K2 PLUS D	86,565 km	Diesel	2nd Owne	â,¹6,354/m	â,¹3.25L	â,¹3.64L	Sachin Magdalla Road, Sur				
139	2021	Mahindra	LX D 4*4 M	43,315 km	Diesel	1st Owner	â,¹22,628/	â,¹11.89L	â,¹12.19L	Sachin Magdalla Road, Sur				
140	2018	Jeep Com	LIMITED 2.	96,371 km	Diesel	2nd Owne	â,¹15,926/	â,¹8.37L	â,¹9.22L	Sachin Magdalla Road, Sur				
141	2020	Renault TI	RXZ	30,916 km	Petrol	1st Owner	â,¹10,283/	â,¹5.26L	â,¹5.90L	Sachin Magdalla Road, Sur				
142	2021	Mahindra	W8 (O) 1.2	21,374 km	Petrol	1st Owner	â,¹16,738/	â,¹8.79L	â,¹9.86L	Sachin Magdalla Road, Sur				
143	2022	Mahindra	W8 (O) 1.2	16,155 km	Petrol	1st Owner	â,¹19,814/	â,¹10.41L	â,¹12.18L	Sachin Magdalla Road, Sur				
144	2019	Jeep Com	LIMITED P	55,119 km	Petrol	1st Owner	â,¹27,035/	â,¹14.20L	â,¹16.66L	Sachin Magdalla Road, Sur				
145	2017	Renault D	110 PS RXZ	42,459 km	Diesel	1st Owner	â,¹11,417/	â,¹5.84L	â,¹7.13L	Sachin Magdalla Road, Sur				
146	2021	Mahindra	W8 (O) 1.2	17,357 km	Petrol	1st Owner	â,¹16,921/	â,¹8.89L	â,¹9.96L	Sachin Magdalla Road, Sur				
147	2022	Renault TI	RXT LIMITI	41,652 km	Petrol	1st Owner	â,¹12,043/	â,¹6.16L	â,¹7.32L	Sachin Magdalla Road, Sur				
148	2022	Renault K	RXT 1.0	39,021 km	Petrol	1st Owner	â,¹9,165/m	â,¹4.69L	â,¹5.32L	Sachin Magdalla Road, Sur				
149	2019	Renault TI	RXZ	32,097 km	Petrol	1st Owner	â,¹12,065/	â,¹6.17L	â,¹6.83L	Sachin Magdalla Road, Sur				
150	2023	Renault TI	RXT	8,367 km	Petrol	1st Owner	â,¹12,759/	â,¹6.53L		Sachin Magdalla Road, Sur				
151	2021	Jeep Com	LIMITED ((	24,646 km	Petrol	1st Owner	â,¹32,320/	â,¹16.98L	â,¹18.43L	Sachin Magdalla Road, Sur				
152	2022	Mahindra	W6 1.2 PE	14,549 km	Petrol	1st Owner	â,¹15,836/	â,¹8.32L	â,¹9.43L	Sachin Magdalla Road, Sur				
153	2021	Renault K	RXT 1.0 AN	10,797 km	Petrol	1st Owner	â,¹8,934/m	â,¹4.57L	â,¹5.30L	Sachin Magdalla Road, Sur				
154	2019	Renault TI	RXZ	55,855 km	Petrol	1st Owner	â,¹11,665/	â,¹5.97L	â,¹6.96L	Sachin Magdalla Road, Sur				
155	2020	Mahindra	LX D 4WD	11,003 km	Diesel	1st Owner	â,¹23,431/	â,¹12.31L		M3M Urbana, Golf Course Ext., Gurugram				
156	2020	Mahindra	S11 2WD	38,501 km	Diesel	1st Owner	â,¹30,148/	â,¹15.84L	â,¹16.53L	KW Delhi 6, Raj Nagar Extension, Ghaziabad				
157	2022	Mahindra	AX 7 LUXU	62,567 km	Diesel	1st Owner	â,¹44,197/	â,¹23.22L		Metro Walk, Rohini, New Delhi				
158	2023	Renault TI	RXL MT	6,843 km	Petrol	1st Owner	â,¹12,428/	â,¹6.36L	â,¹6.92L	M3M Urbana, Golf Course Ext., Gurugram				
159	2021	Mahindra	W7 AT	25,366 km	Diesel	1st Owner	â,¹26,644/	â,¹14.00L		M3M Urbana, Golf Course Ext., Gurugram				

cars24_data

Ready

Figure 26: Output Screenshot 5

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
155	2020	Mahindra	LX D 4WD	11,003 km	Diesel	1st Owner	â‚¹23,431/	â‚¹12.31L		M3M Urbana, Golf Course Ext., Gurugram				
156	2020	Mahindra	S11 2WD	38,501 km	Diesel	1st Owner	â‚¹30,148/	â‚¹15.84L	â‚¹16.53L	KW Delhi 6, Raj Nagar Extension, Ghaziabad				
157	2022	Mahindra	AX 7 LUXU	62,567 km	Diesel	1st Owner	â‚¹44,197/	â‚¹23.22L		Metro Walk, Rohini, New Delhi				
158	2023	Renault TI	RXL MT	6,843 km	Petrol	1st Owner	â‚¹12,428/	â‚¹6.36L	â‚¹6.92L	M3M Urbana, Golf Course Ext., Gurugram				
159	2021	Mahindra	W7 AT	25,366 km	Diesel	1st Owner	â‚¹26,644/	â‚¹14.00L		M3M Urbana, Golf Course Ext., Gurugram				
160	2020	Renault TI	RXZ	34,646 km	Petrol	1st Owner	â‚¹11,618/	â‚¹5.94L	â‚¹6.59L	M3M Urbana, Golf Course Ext., Gurugram				
161	2020	Mahindra	K2+ P 6 ST	19,376 km	Petrol	1st Owner	â‚¹9,269/n	â‚¹4.74L	â‚¹5.77L	Metro Walk, Rohini, New Delhi				
162	2016	Renault K	RXT 1.0 (O	20,063 km	Petrol	1st Owner	â‚¹5,126/n	â‚¹2.62L		KW Delhi 6, Raj Nagar Extension, Ghaziabad				
163	2021	Renault K	RXT 1.0 (O	46,746 km		1st Owner	â‚¹7,234/n	â‚¹3.70L	â‚¹3.96L	Metro Walk, Rohini, New Delhi				
164	2021	Mahindra	LX D AT 4V	62,619 km	Diesel	1st Owner	â‚¹24,878/	â‚¹13.07L		Metro Walk, Rohini, New Delhi				
165	2019	Renault TI	RXL MT	33,392 km	Petrol	1st Owner	â‚¹9,599/n	â‚¹4.91L	â‚¹5.72L	Metro Walk, Rohini, New Delhi				
166	2017	Renault K	CLIMBER 1	41,308 km	Petrol		â‚¹5,593/n	â‚¹2.86L	â‚¹3.27L	Metro Walk, Rohini, New Delhi				
167	2021	Mahindra	LX PETROL	42,201 km	Petrol	1st Owner	â‚¹26,172/	â‚¹13.75L	â‚¹14.70L	Sector-18, Noida				
168	2021	Renault K	RXZ TURB	46,375 km	Petrol	2nd Owne	â‚¹12,788/	â‚¹6.54L	â‚¹7.91L	Sector-18, Noida				
169	2020	Jeep Com	SPORT PLL	76,041 km	Petrol	1st Owner	â‚¹20,976/	â‚¹11.02L	â‚¹12.40L	M3M Urbana, Golf Course Ext., Gurugram				
170	2019	Mahindra	W5	45,630 km	Diesel	1st Owner	â‚¹21,977/	â‚¹10.11L	â‚¹10.85L	Sector-18, Noida				
171	2020	Mahindra	W6 1.2 PE	40,924 km	Petrol	1st Owner	â‚¹13,731/	â‚¹7.02L	â‚¹7.60L	Metro Walk, Rohini, New Delhi				
172	2019	Jeep Com	SPORT 2.0	93,467 km	Diesel	1st Owner	â‚¹21,286/	â‚¹9.79L	â‚¹11.51L	Chharpur, Delhi				
173	2021	Mahindra	LX PETROL	31,332 km	Petrol	1st Owner	â‚¹25,916/	â‚¹13.62L	â‚¹14.35L	Metro Walk, Rohini, New Delhi				
174	2018	Mahindra	K6+ D 6 ST	63,935 km	Diesel	2nd Owne	â‚¹11,350/	â‚¹4.31L		Chharpur, Delhi				
175														
176														
177														
178														
179														
180														
181														
182														
183														
184														
185														
186														

cars24_data

Ready

Figure 27: Output Screenshot 6

Results

The web scraping project successfully achieved its primary goal of extracting and structuring data from the target website. The developed scraper effectively gathered a wide range of information, including product details, customer reviews, and pricing data, from multiple pages and categories within the site.The data extraction process demonstrated the tool's capability to handle both static and dynamic content, thanks to the integration of BeautifulSoup for HTML parsing and Selenium formanaging JavaScript-loaded elements.

The collected data was successfully stored in CSV and JSON formats, allowing for easy manipulation and analysis. The resultsshowed that the scraper could efficiently navigate through complex web structures and bypass common anti-scraping measures, such as CAPTCHAs and IP blocking, using techniques like request throttling, IP rotation, and error handling.

The accuracy and completeness of the data were validated through various checks, confirming that the information retrieved was reliable and reflective of the website's content. The project not only demonstrated the effectiveness of the webscraping tool but also highlighted its potential applications in market research, sentiment analysis, and competitive intelligence.

Overall, the successful collection and structuring of data underscored the scraper's robustness and its ability to providevaluable insights. The project illustrated the practical utility of web scraping in data science, offering a solid foundation for future enhancements and applications in various domains.

Conclusion

The web scraping project successfully achieved its goal of extracting valuable data from targeted websites, providing a robust foundation for further analysis and decision-making. By automating the data collection process, we streamlined what would have been a time-consuming manual task, enabling more efficient data gathering at scale. The use of tools like BeautifulSoup or Scrapy ensured reliable data extraction, while proper handling of website structures and anti-scraping measures maintained the project's integrity. As we move forward, this data will be instrumental in driving insights and supporting the project's broader objectives. Continued monitoring of website policies and ethical practices will be essential to maintain compliance and ensure long-term successin data extraction endeavors.

References

Web Resources and Tutorials:

- BeautifulSoup Documentation: Since we used BeautifulSoup for scraping car details, referenced the official documentation.
  1) Crummy.com. (2023). BeautifulSoup Documentation. https://www.crummy.com/software/BeautifulSoup/bs4/doc/
- Scrapy Documentation: If you're using Scrapy, cite the official documentation.
  2) Scrapy.org. (2023). Scrapy: A Fast and Powerful Scraping and Web Crawling Framework. https://docs.scrapy.org/en/latest/
- Requests Library Documentation: Requests are commonly used for handling HTTP requests during web scraping.
  3) Python-requests.org. (2023). Requests: HTTP for Humans. https://docs.python-requests.org/en/latest/

Research Papers on Web Scraping & Data Mining

- Fan, W., & Bifet, A. (2013). Mining big data: current status, and forecast to the future. ACM SIGKDD Explorations Newsletter, 14(2), 1-5.
- *Liu, B. (2011). Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data. Springer.

Ethics of Web Scraping

1) *Conti, M., Dragoni, N., & Lesyk, V. (2016). A survey of man-in-the-middle attacks. IEEE Communications Surveys & Tutorials, 18(3), 2027-2051.*

Tools and Technologies Used

- Selenium Documentation: For projects involving scraping dynamic pages.

2) *Selenium.dev. (2023). Selenium Documentation.*
https://www.selenium.dev/documentation/

Website for webscraping

https://www.cars24.com/buy-used-cars-new-delhi/

Appendix

BeautifulSoup: Used for parsing HTML and extracting data from web pages.
nium: Employed to handle dynamic content rendered by JavaScript.
Requests: Utilized to send HTTP requests and retrieve web page content.
Initial Request: Send HTTP requests to target URLs. Data Cleaning: Use Pandas to clean and organize the data.