Team A Mini Project: Car Data Analysis

Introduction

Mumbai, one of India's largest and most dynamic cities, offers a rich tapestry of consumer preferences and buying patterns in the automotive market. This report aims to analyze data on the most commonly used car brands in Mumbai. These brands dominate the market and provide a diverse range of options catering to various consumer needs.

Our dataset encompasses key attributes such as brand name, fuel type, price, transmission type, and kilometers driven. By examining these variables, we can uncover valuable insights into consumer behavior, market trends, and the competitive landscape in Mumbai's automotive sector.

Objective

The primary objective of this analysis is to understand the market dynamics and consumer preferences for these four car brands in Mumbai. Specifically, we aim to:

1. **Compare Brand Performance**: Evaluate how each brand performs in terms of price, fuel type, transmission, and other factors.
2. **Understand Consumer Preferences**: Identify patterns in consumer choices based on the available data.

Tools and Libraries Used

- **Selenium**: Automates web browser interactions.
- **BeautifulSoup**: Parses HTML content to extract necessary data.
- **Pandas**: Manipulates and analyzes data.
- **Chrome WebDriver**: Interacts with the Chrome browser to facilitate data scraping.

Methodology

To achieve the objectives of this analysis, we employ the following methodology:

Data Extraction and Preparation

3. **Data Scraping**:
    - Scrape car listing data from Cars24.com, focusing on the Mumbai location.

- Clean and structure the extracted data to ensure accuracy and consistency.
- Save the processed data to a CSV file for further analysis.

Step-by-Step Approach

4. **Import Necessary Libraries**:
    - Import Selenium for web automation.
    - Import BeautifulSoup from the bs4 library for HTML parsing.
    - Import Pandas for data handling and analysis.
5. **Configure Chrome WebDriver**
    - Set up the Chrome WebDriver to run in headless mode, enabling background data scraping without launching a browser window.
6. **Data Cleaning and Storage**
    - Clean the raw data to remove any inconsistencies or irrelevant information.
    - Store the cleaned data in a structured format, such as a CSV file, for easy access during analysis.

By following this methodology, we aim to deliver a comprehensive analysis of Mumbai's car market, providing insights that can inform future decisions in the automotive sector.

Method

To achieve the objectives of this analysis, we will employ the following methodology:

Data Extraction and Preparation

1. Data Scraping

- To scrape car listing data from Cars24.com.
- To clean and structure the extracted data.
- To save the processed data to a CSV file for further of service.

First, we have to import the necessary libraries for extraction and scraping the data and should configure the Chrome WebDriver to run in headless mode.

We need Chrome Driver and also needs to Import libraries like :

- Selenium
- BS4
- Pandas

The code is given below:

Importing the necessary libraries

```
# Selenium: A tool for automating web browser interaction
from selenium import webdriver # For controlling the web browser
from selenium.webdriver.chrome.options import Options # For configuring browser
# For managing the ChromeDriver service
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By # For locating elements on the page
# For implementing explicit waits
from selenium.webdriver.support.ui import WebDriverWait
# For defining wait conditions
from selenium.webdriver.support import expected_conditions as EC

# BeautifulSoup: A library for parsing HTML and XML documents
from bs4 import BeautifulSoup
import time # Time: A library for handling time-related tasks
import pandas as pd # For creating and manipulating data frames
import warnings # For managing warning messages
# Suppress warnings to avoid clutter in the notebook output
import numpy as np
warnings.filterwarnings("ignore", module="some_library")
import pandas as pd
# Data Visualisation
import matplotlib.pyplot as plt
```

Chrome Driver

```
# Configure Chrome options
chrome_options = Options()
chrome_options.add_argument("--headless") # Run Chrome in headless mode (without

# Initialize Chrome driver
driver = webdriver.Chrome(options=chrome_options)

# Define website URL
website = 'https://www.cars24.com/buy-used-car?sort=bestmatch&serveWarrantyCount

# Open the website
driver.get(website)
```

2. Navigating to the Website

We navigated to the Cars24 website and loaded the page content.

```
website = "https://www.cars24.com/buy-used-car?f=make%3A%3D%3Amaruti&sort=bestma
driver.get(website)
```

3. Scrolling Through the Page

We implemented a function to scroll through the page to load more content dynamically.

```
def scroll_page():
    driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
    time.sleep(2)
for _ in range(5):
    scroll_page()
```

4. Extracting Data

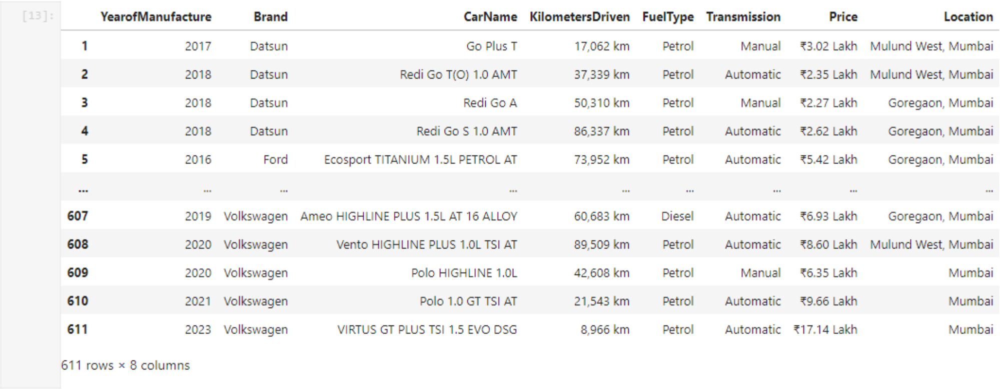
We used BeautifulSoup to parse the HTML content and extract relevant data.

```
wait = WebDriverWait(driver, 10)
wait.until(EC.presence_of_element_located((By.CLASS_NAME, "_2YB7p")))
soup = BeautifulSoup(driver.page_source, 'html.parser')
driver.quit()

results = soup.find_all('div', {'class': '_2YB7p'})
data = []

for car in results:
    name_elem = car.find('h3', {'class': '_11dVb'})
    full_name = name_elem.get_text() if name_elem else ''
    year = full_name[:4] if full_name else ''
    name = full_name[5:] if full_name else ''
    details = car.find('ul', {'class': '_3J2G-'})
    mileage = details.find_all('li')[0].get_text() if details else ''
    fuel_type = details.find_all('li')[2].get_text() if details else ''
    transmission = details.find_all('li')[4].get_text() if details else ''
    price_elem = car.find('div', {'class': '_7udZZ'})
    price = price_elem.get_text() if price_elem else ''
    location_elem = car.find('div', {'class': '_1TzL8'})
    location = location_elem.get_text() if location_elem else ''
    data.append([year, name, kilometers, fuel_type, transmission, price, locatio
```

5. Creating a DataFrame

We converted the extracted data into a Pandas DataFrame for further analysis.

```
columns = ['Year', 'CarName', 'KilometersDriven', 'FuelType',
 'Transmission', 'Price', 'Location']
df = pd.DataFrame(data, columns=columns)
```



	YearofManufacture	Brand	CarName	KilometersDriven	FuelType	Transmission	Price	Location
1	2017	Datsun	Go Plus T	17,062 km	Petrol	Manual	₹3.02 Lakh	Mulund West, Mumbai
2	2018	Datsun	Redi Go T(O) 1.0 AMT	37,339 km	Petrol	Automatic	₹2.35 Lakh	Mulund West, Mumbai
3	2018	Datsun	Redi Go A	50,310 km	Petrol	Manual	₹2.27 Lakh	Goregaon, Mumbai
4	2018	Datsun	Redi Go S 1.0 AMT	86,337 km	Petrol	Automatic	₹2.62 Lakh	Goregaon, Mumbai
5	2016	Ford	Ecosport TITANIUM 1.5L PETROL AT	73,952 km	Petrol	Automatic	₹5.42 Lakh	Goregaon, Mumbai
...	...	...	...	...	...	...	...	...
607	2019	Volkswagen	Ameo HIGHLINE PLUS 1.5L AT 16 ALLOY	60,683 km	Diesel	Automatic	₹6.93 Lakh	Goregaon, Mumbai
608	2020	Volkswagen	Vento HIGHLINE PLUS 1.0L TSI AT	89,509 km	Petrol	Automatic	₹8.60 Lakh	Mulund West, Mumbai
609	2020	Volkswagen	Polo HIGHLINE 1.0L	42,608 km	Petrol	Manual	₹6.35 Lakh	Mumbai
610	2021	Volkswagen	Polo 1.0 GT TSI AT	21,543 km	Petrol	Automatic	₹9.66 Lakh	Mumbai
611	2023	Volkswagen	VIRTUS GT PLUS TSI 1.5 EVO DSG	8,966 km	Petrol	Automatic	₹17.14 Lakh	Mumbai

611 rows × 8 columns

6. Data Cleaning and Processing

We cleaned and processed the data to make it suitable for analysis.

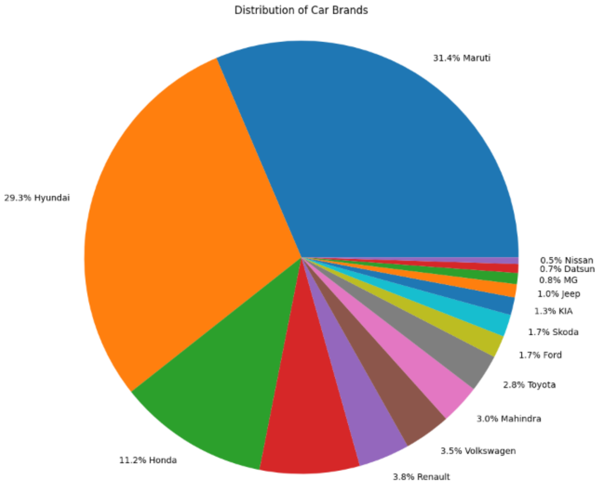
```
df['Price'] = df['Price'].str.replace('₹', '').str.replace(' Lakh', '').astype(f
 * 1e5
df['KilometersDriven'] = df['KilometersDriven'].str.replace(',', '').str.
replace(' km', '').astype(int)
df['Year'] = df['Year'].astype(int)
df['CarName'] = df['CarName'].str.strip()
```

7. Data Analysis

```
# to check any null values and to group the car with respect to year , brand anc
df.sort_values(by=['Brand', 'YearofManufacture'], inplace=True)
df.isnull().values.any()
df.reset_index(drop=True, inplace=True)
df.index = df.index + 1
df
# to obtain cars having low price-
lowest_prices = df.groupby(['Brand','CarName'])['Price'].min().reset_index(0)
print(lowest_prices)
```

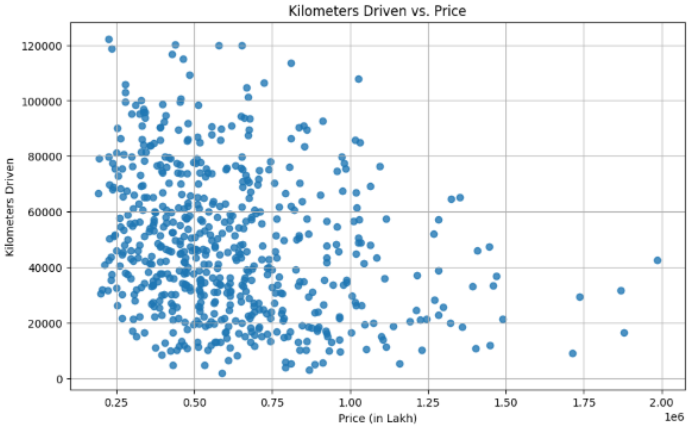
8. Visualizing the Distribution of Car Brands

```
brand_counts = df['Brand'].value_counts()
brand_percentages = brand_counts / brand_counts.sum() * 100
labels = [f'{percentage:.1f}% {brand}' for brand, percentage in
brand_percentages.items()]
plt.figure(figsize=(12, 10))
plt.pie(brand_counts.values, labels=labels)
plt.title('Distribution of Car Brands')
plt.axis('equal')
```

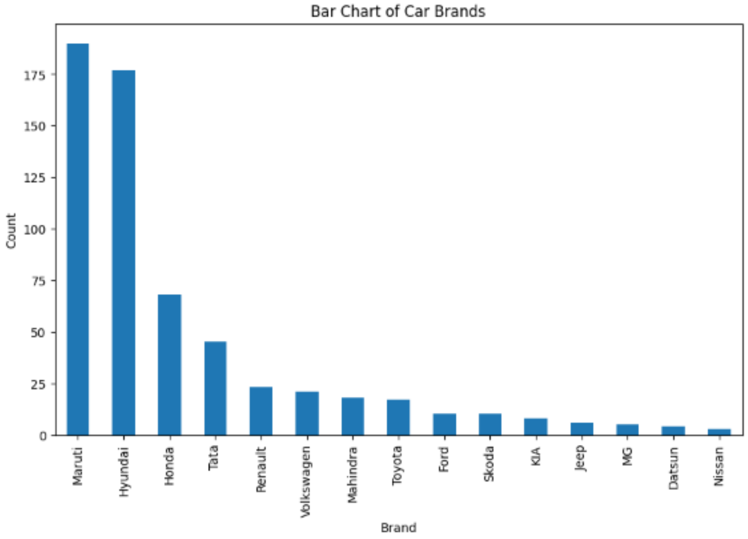


Some extra plots, charts and graphs:

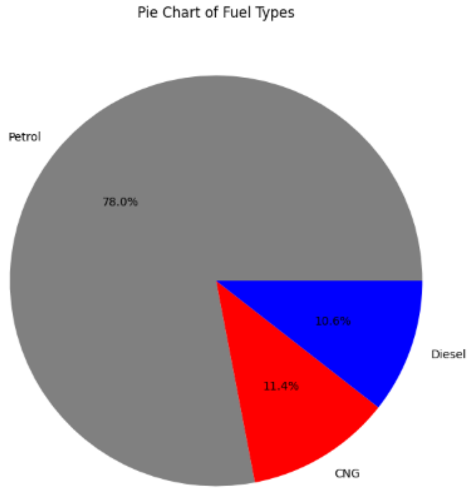
Scatter Plot of Kilometers Driven vs. Price:



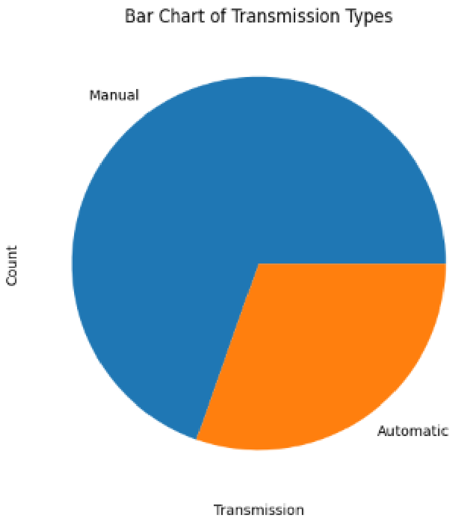
Bar Chart of Car Brands:



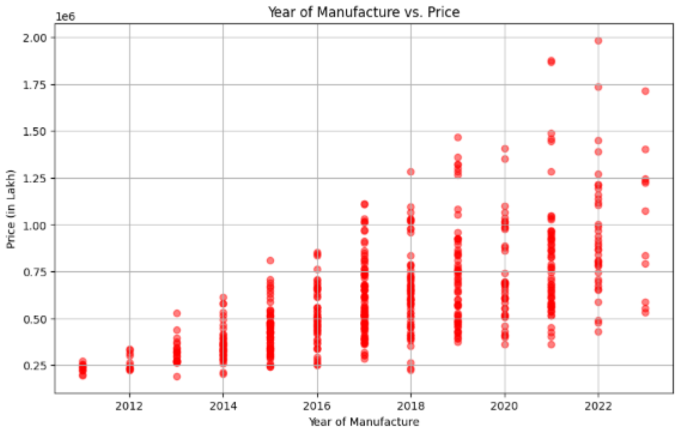
Pie Chart of Fuel Types:



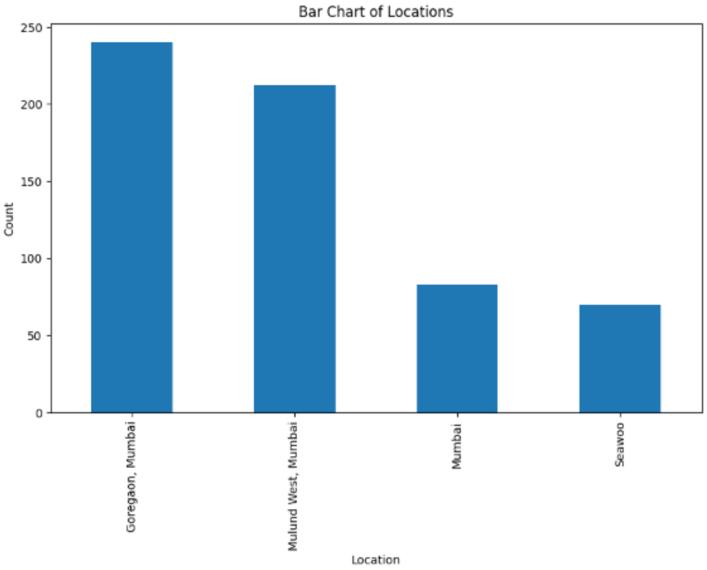
Pie Chart of Transmission Types:



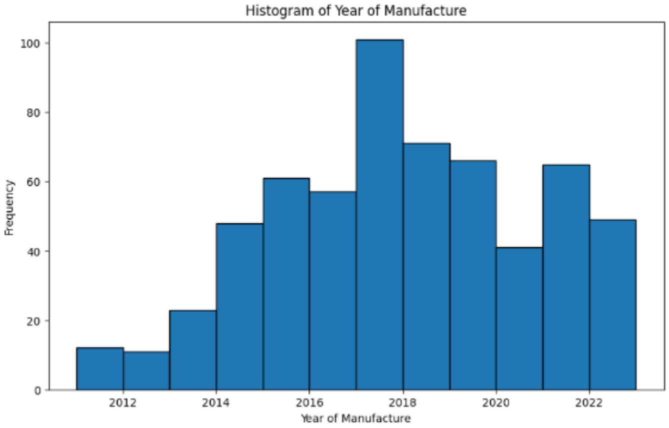
Scatter Plot of Year of Manufacture vs. Price:



Bar Chart of Locations:



Histogram of Year of Manufacture



Challenges

Dynamic Content Loading: The Cars24 website uses infinite scrolling to load more car listings, which required implementing a scroll function to load additional content.

Data Extraction: Extracting specific elements from the HTML content required careful inspection of the webpage structure and handling missing elements gracefully.

Data Cleaning: Converting data types and cleaning the extracted data to remove unwanted characters was necessary for accurate analysis.

Solutions Implemented

- Implemented a scroll function to load more content.

- Used BeautifulSoup to parse the HTML and extract relevant data.
- Cleaned the data by removing unwanted characters and converting data types appropriately.

Insights Gained

- From the data extracted and analyzed, we gained several insights into the car market:
- Price Analysis: We identified the lowest and average prices for different car models.
- Kilometers Driven: We analyzed the total kilometers driven for cars based on their fuel type.