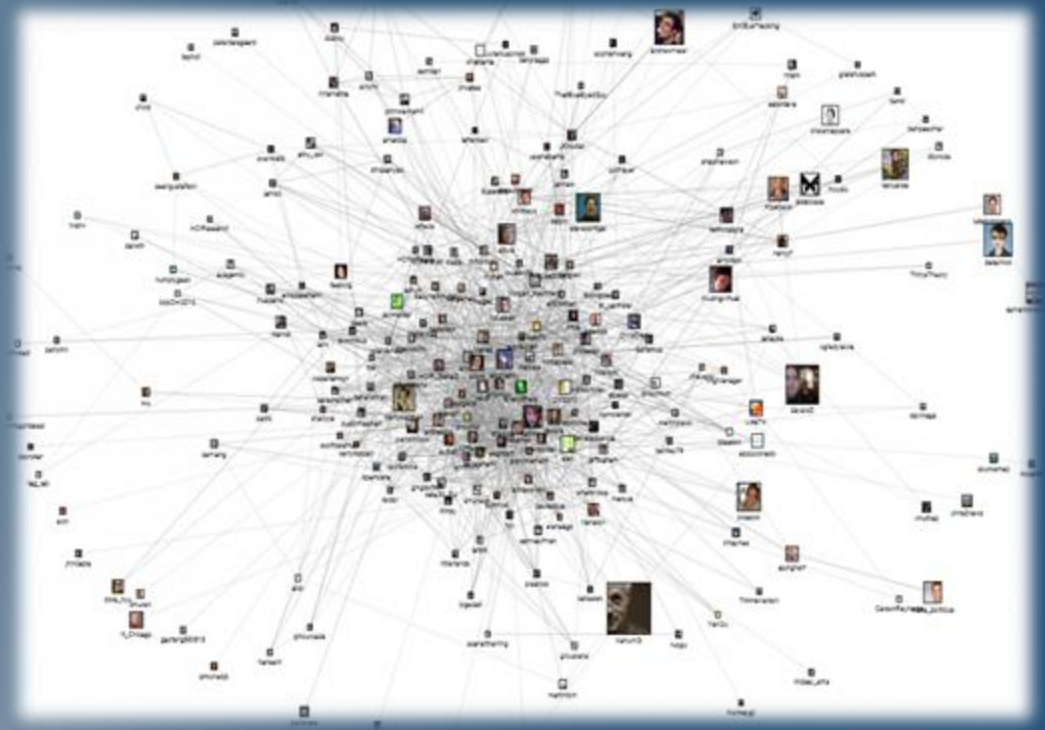
EVOASTRA VENTURES
DIGITAL SOLUTIONS

MINI PROJECT PRESENTATION

TEAM D

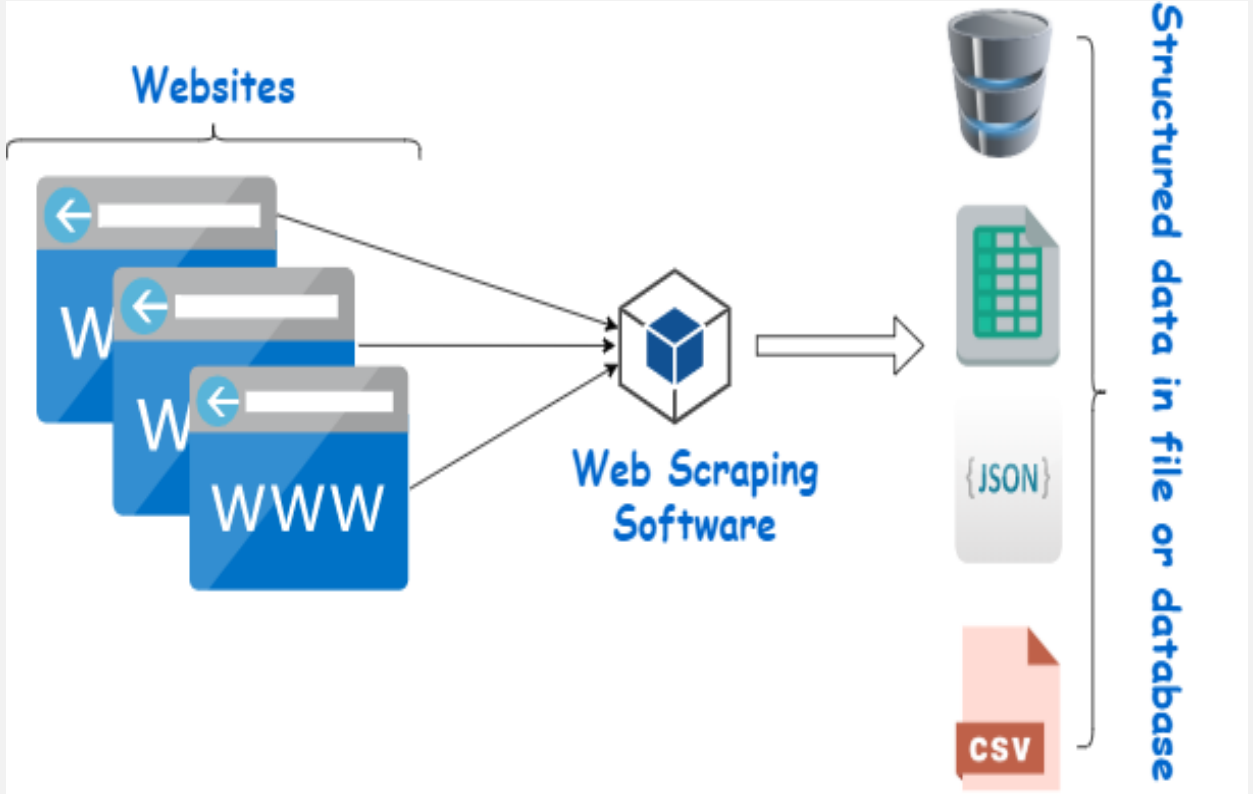
MARUTI SUZUKI



MINI PROJECT TOPIC-

WEB SCRAPING - CARS24.COM

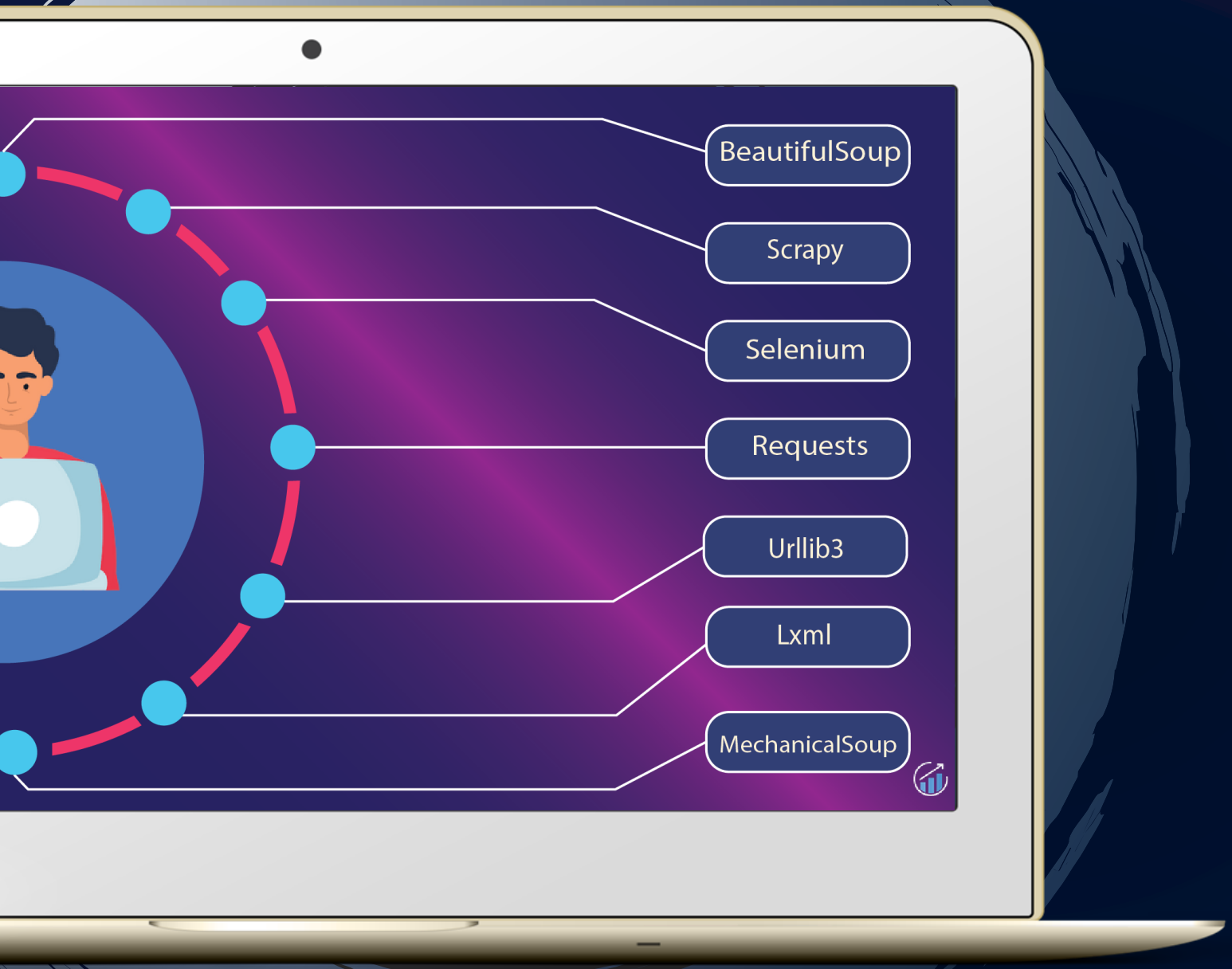
BRAND - MARUTI SUZUKI



What is Web Scraping?

Web scraping, the process of extracting data from websites, has emerged as a powerful technique to gather information from the vast expanse of the internet.

It involves using software, known as a scraper, to make HTTP requests to a website and then parse the HTML content to retrieve specific information.



TOOLS REQUIRED FOR WEB SCRAPING

1. BeautifulSoup (Python)
2. Scrapy (Python)
3. Selenium (Python)
4. Requests (Python)
5. Pandas

SOURCE CODE

MODULES USED -

1. **SELENIUM -**
   - **Purpose :** Selenium is used to automate web browser interaction. In this Script, It is employed to open URL, scroll through the page to load all car listings and retrieve the page source.
2. **TIME -**
   - **Purpose :** Provides various time-related functions. Here it is used to pause the execution for specified intervals, allowing the webpage to load and scroll actions to complete.
3. **BeautifulSoup (from bs4) -**
   - **Purpose :** BeautifulSoup is a library for parsing HTML and XML documents. It creates parse trees that are helpful to extract data from HTML.
4. **REQUESTS -**
   - **Purpose :** while not used directly in the provided script, the 'requests' library is commonly used for sending HTTP requests. It is listed here likely for completeness, as it can be useful in web scraping tasks.
5. **Urllib.parse -**
   - **Purpose :** Provides functions for manipulating URLs. Again, not directly used in the script but useful for resolving relative URLs.
6. **PANDAS -**
   - **Purpose :** Pandas is a powerful data manipulation and analysis library. It is used to create and manipulate DataFrames.
7. **WARNINGS -**
   - **Purpose :** Allows control over warning messages. Here, it is used to ignore all warnings for a cleaner output.
8. **Webdriver_manager.chrome -**
   - **Purpose :** Manages ChromeDriver binaries for selenium.

Function for getting the details

- **Purpose :** Extracts car details from the parsed HTML page source.
- **Parameters :** ‘soup’ - BeautifulSoup object containing the parsed HTML.
- **Returns :** Seven lists containing year, make, model, transmission type, kilometers driver, fuel type, and prices of the cars.

```
def get_details(soup):
    name_elements = soup.find_all('h3', {'class': '_11dVb'})
    transmission_elements = soup.find_all('ul', {'class': '_3J2G-'})
    km_elements = soup.find_all('ul', {'class': '_3J2G-'})
    price_elements = soup.find_all('div', {'class': '_2KyOK'})

    # creating empty lists
    year = []
    make = []
    model = []
    transmissions = []
    kilometers_driven = []
    fuel = []
    prices = []
```

Function for getting the details #2

- **Functionality :**
  - Finds all car name elements and extracts year, make, and model.
  - Finds all transmission elements and extracts transmission type.
  - Finds all kilometers elements and extracts kilometers driven and fuel type.
  - Finds all price elements and extracts the price, converting it to an integer.

```
#Extracting Names of the cars
for elem in name_elements:
    text = elem.get_text()
    year.append(int(text[:4]))
    make.append(text.split(" ")[1])
    model.append(" ".join(text.split(" ")[2:]))

#Extracting Transmission type of the cars
for ul in transmission_elements:
    transmissions.append(ul.find_all('li')[-1].get_text())

#Extracting Kilometer_Driven of the cars
for ul in km_elements:
    km_text = ul.find_all('li')[0].get_text()[:-3].replace(',', '')
    kilometers_driven.append(int(km_text))
    fuel.append(ul.find_all('li')[2].get_text())

#Extracting Price of the cars
for div in price_elements:
    price_text = div.find_all('strong')[0].get_text()[1:].replace(',', '')
    if 'Lakh' in price_text:
        price_value = float(price_text.replace(' Lakh', '')) * 100000
    else:
        price_value = float(price_text)
    prices.append(int(price_value))
return year, make, model, transmissions, kilometers_driven, fuel, prices
```

Function for extracting the data

- **Purpose :** Automates the process of opening the webpage, scrolling through it to load all car listings and extracting data using ‘get_details’ .
- **Parameters :** ‘url’ - The url of the Cars24 used car listing page.
- **Returns :** A pandas DataFrame containing the extracted car details.
- **Functionality :**
  - Initializes the webdriver with headless options to run in the background.
  - Open the specified URL.
  - Scrolls through the page incrementally until all listings are loaded.
  - Parses the loaded page source with BeautifulSoup.
  - Extracts car details using the ‘get_details’ function.
  - Creates and Returns a DataFrame with the extracted details.

```python
def extracting_data(url):
    driver = webdriver.Chrome(options=chrome_options)
    driver.get(url)
    time.sleep(2)  #allow 2 seconds for the webpage to open
    scroll_pause_time = 0.5
    screen_height = driver.execute_script("return window.screen.height;")  #get the screen height of the web
    i = 1

    while True:
        #Scroll one screen heigth each time
        driver.execute_script(f"window.scrollTo(0, {screen_height}*{i});".format(screen_height = screen_height, i=i))
        i += 1
        time.sleep(scroll_pause_time)
        # update scroll height each time after scrolled as the scroll height can change after we scroll the page
        scroll_height = driver.execute_script("return document.body.scrollHeight;")
        # break the loop when the height we need to scroll to is larger the the screen height
        if (screen_height) * i > scroll_height:
            break

    soup = BeautifulSoup(driver.page_source, 'html.parser')
    driver.quit()

    year, make, model, transmissions, kilometers_driven, fuel, prices = get_details(soup)


    # creating data frame
    df = pd.DataFrame({
        'Year of manufacture': year,
        'Make': make,
        'Model': model,
        'Transmission type': transmissions,
        'Km driven': kilometers_driven,
        'Fuel type': fuel,
        'Price (in Rs)': prices
    })
    return df
```

DataFrame and CSV File -

Converted the dataframe into csv file using ".to_csv" function.

```
[ ]  df = extracting_data(url)
```

```
df
```

	Year of manufacture	Make	Model	Transmission type	Km driven	Fuel type	Price (in Rs)
0	2017	Maruti	Swift Dzire ZXI	Manual	36953	Petrol	490000
1	2014	Maruti	Ertiga VXI CNG	Manual	70348	CNG	620000
2	2018	Maruti	Dzire VXI AMT	Automatic	51852	Petrol	574000
3	2014	Maruti	Swift Dzire ZDI	Manual	11144	Diesel	613000
4	2012	Maruti	Wagon R 1.0 VXI	Manual	42425	Petrol	238000
...	...	...	...	...	...	...	...
201	2014	Maruti	Ertiga VXI	Manual	65000	Petrol	535000
202	2021	Maruti	Celerio ZXI (O)	Manual	23012	Petrol	501999
203	2023	Maruti	New Wagon-R LXI 1.0	Manual	1958	Petrol	589000
204	2020	Maruti	New Wagon-R ZXI 1.2 AMT	Automatic	24495	Petrol	549000
205	2014	Maruti	Wagon R 1.0 LXI CNG	Manual	94053	CNG	301000

206 rows × 7 columns

```
[ ]  df.to_csv('cars24.csv')
```

Exploratory Data Analysis (EDA)

```
[5]  data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 206 entries, 0 to 205
Data columns (total 8 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Unnamed: 0            206 non-null    int64
 1   Year of manufacture   206 non-null    int64
 2   Make                  206 non-null    object
 3   Model                 206 non-null    object
 4   Transmission type     206 non-null    object
 5   Km driven             206 non-null    int64
 6   Fuel type             206 non-null    object
 7   Price (in Rs)         206 non-null    int64
dtypes: int64(4), object(4)
memory usage: 13.0+ KB
```

```
# Unique Values in each column
data.nunique()
```

```
Unnamed: 0              206
Year of manufacture      14
Make                      1
Model                    83
Transmission type         2
Km driven               206
Fuel type                 3
Price (in Rs)           174
dtype: int64
```

Fig. Getting Information from the dataset & Finding unique values in each column

EXPLORATORY DATA ANALYSIS (EDA) #2

```
[7] # Null Values in each column
    data.isnull().sum()
```

```
Unnamed: 0            0
Year of manufacture   0
Make                  0
Model                 0
Transmission type     0
Km driven             0
Fuel type             0
Price (in Rs)         0
dtype: int64
```

```
# Drop Unwanted Columns
data = data.drop([1])
data
```

	Unnamed: 0	Year of manufacture	Make	Model	Transmission type	Km driven	Fuel type	Price (in Rs)
0	0	2017	Maruti	Swift Dzire ZXI	Manual	36953	Petrol	490000
2	2	2018	Maruti	Dzire VXI AMT	Automatic	51852	Petrol	574000
3	3	2014	Maruti	Swift Dzire ZDI	Manual	11144	Diesel	613000
4	4	2012	Maruti	Wagon R 1.0 VXI	Manual	42425	Petrol	238000
5	5	2015	Maruti	Wagon R 1.0 LXI CNG	Manual	79837	CNG	364000
...	...	...	...	...	...	...	...	...
201	201	2014	Maruti	Ertiga VXI	Manual	65000	Petrol	535000
202	202	2021	Maruti	Celerio ZXI (O)	Manual	23012	Petrol	501999
203	203	2023	Maruti	New Wagon-R LXI 1.0	Manual	1958	Petrol	589000
204	204	2020	Maruti	New Wagon-R ZXI 1.2 AMT	Automatic	24495	Petrol	549000
205	205	2014	Maruti	Wagon R 1.0 LXI CNG	Manual	94053	CNG	301000

✓ 0s completed at 12:11

Fig. Checking for the Null values and Dropping the unwanted columns.

```
#Age of the car
from datetime import date
date.today().year
data['Car_Age']=date.today().year-data['Year of manufacture']
data
```

	Unnamed: 0	Year of manufacture	Make	Model	Transmission type	Km driven	Fuel type	Price (in Rs)	Car_Age
0	0	2017	Maruti	Swift Dzire ZXI	Manual	36953	Petrol	490000	7
2	2	2018	Maruti	Dzire VXI AMT	Automatic	51852	Petrol	574000	6
3	3	2014	Maruti	Swift Dzire ZDI	Manual	11144	Diesel	613000	10
4	4	2012	Maruti	Wagon R 1.0 VXI	Manual	42425	Petrol	238000	12
5	5	2015	Maruti	Wagon R 1.0 LXI CNG	Manual	79837	CNG	364000	9
...	...	...	...	...	...	...	...	...	...
201	201	2014	Maruti	Ertiga VXI	Manual	65000	Petrol	535000	10
202	202	2021	Maruti	Celerio ZXI (O)	Manual	23012	Petrol	501999	3
203	203	2023	Maruti	New Wagon-R LXI 1.0	Manual	1958	Petrol	589000	1
204	204	2020	Maruti	New Wagon-R ZXI 1.2 AMT	Automatic	24495	Petrol	549000	4
205	205	2014	Maruti	Wagon R 1.0 LXI CNG	Manual	94053	CNG	301000	10

205 rows x 9 columns

Fig. Finding the Age of the Car.

VISUALIZATION

```python
import matplotlib.pyplot as plt
import seaborn as sns

# Distribution plots
plt.figure(figsize=(12, 6))

plt.subplot(2, 2, 1)
sns.histplot(data['Year of manufacture'], bins=20, kde=True)
plt.title('Distribution of Year of Manufacture')

plt.subplot(2, 2, 2)
sns.histplot(data['Km driven'], bins=20, kde=True)
plt.title('Distribution of Km driven')

plt.subplot(2, 2, 3)
sns.histplot(data['Price (in Rs)'], bins=20, kde=True)
plt.title('Distribution of Price')

plt.subplot(2, 2, 4)
sns.histplot(data['Car Age'], bins=20, kde=True)
plt.title('Distribution of Car Age')

plt.tight_layout()
plt.show()
```

Fig. Script for visualising the different features using histogram plot

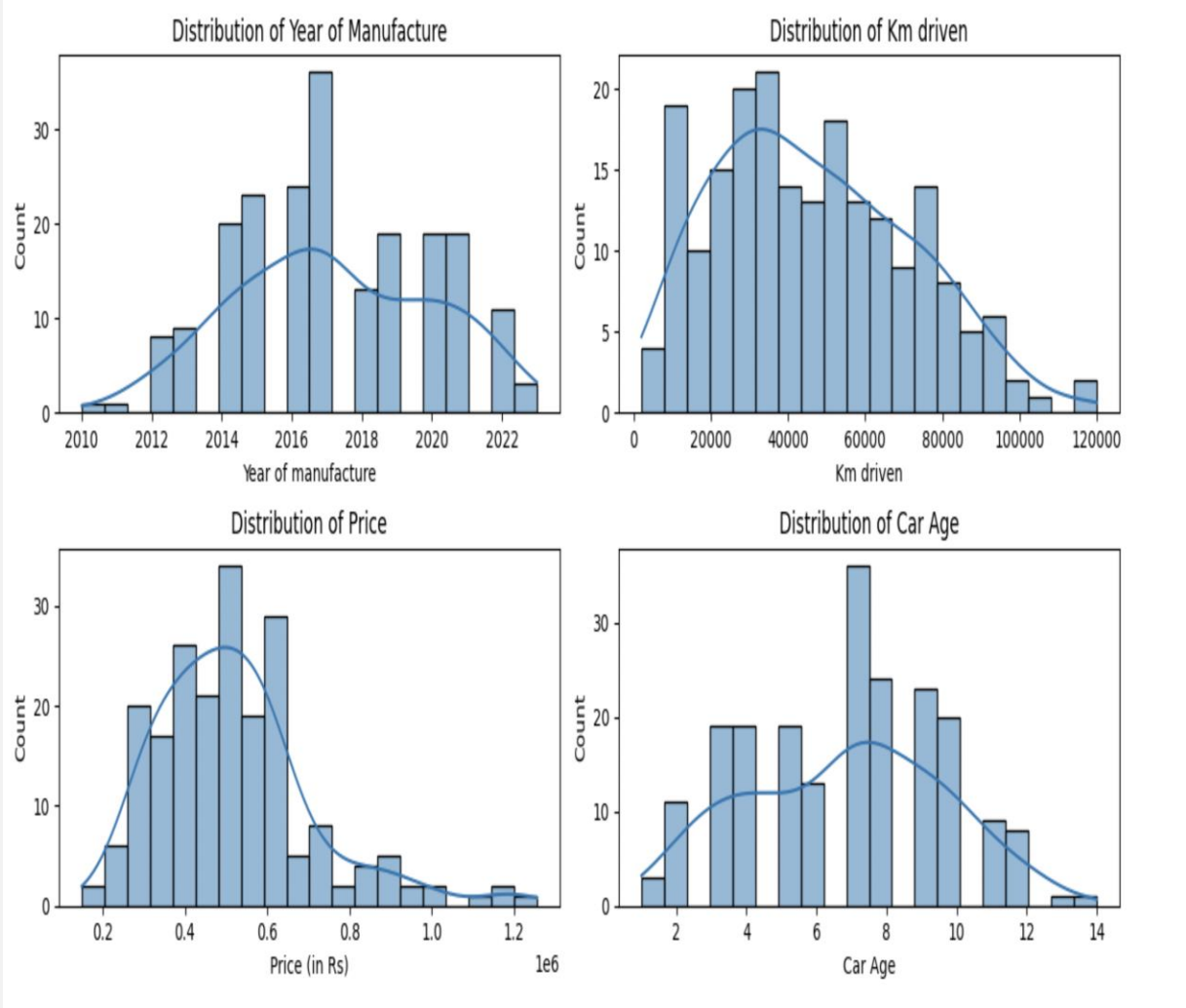


Fig. Output

VISUALIZATION #2

```
plt.figure(figsize=(12, 6))

plt.subplot(2, 2, 1)
sns.countplot(data=data, x='Make')
plt.xticks(rotation=45)
plt.title('Distribution of Makes')

plt.subplot(2, 2, 2)
sns.countplot(data=data, x='Transmission type')
plt.title('Distribution of Transmission types')

plt.subplot(2, 2, 3)
sns.countplot(data=data, x='Fuel type')
plt.title('Distribution of Fuel types')

plt.tight_layout()
plt.show()
```

Fig. Distribution plots for different features

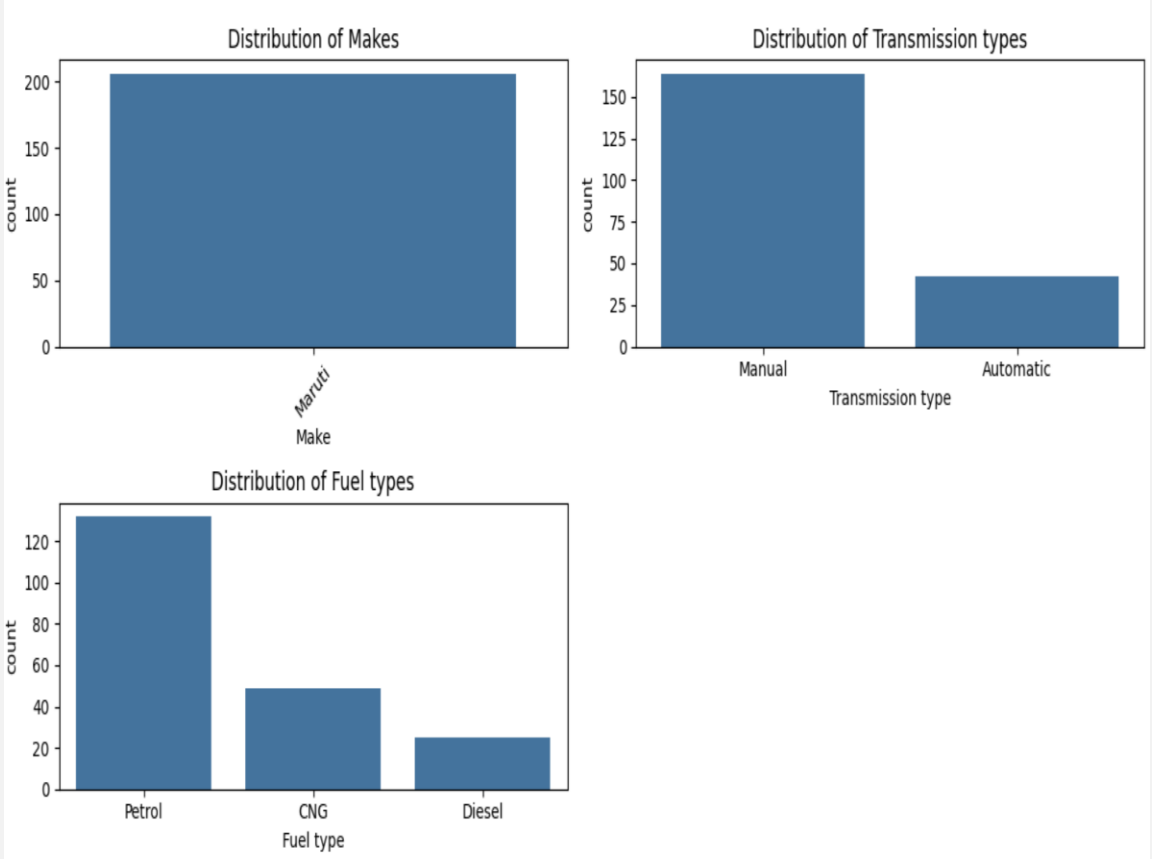
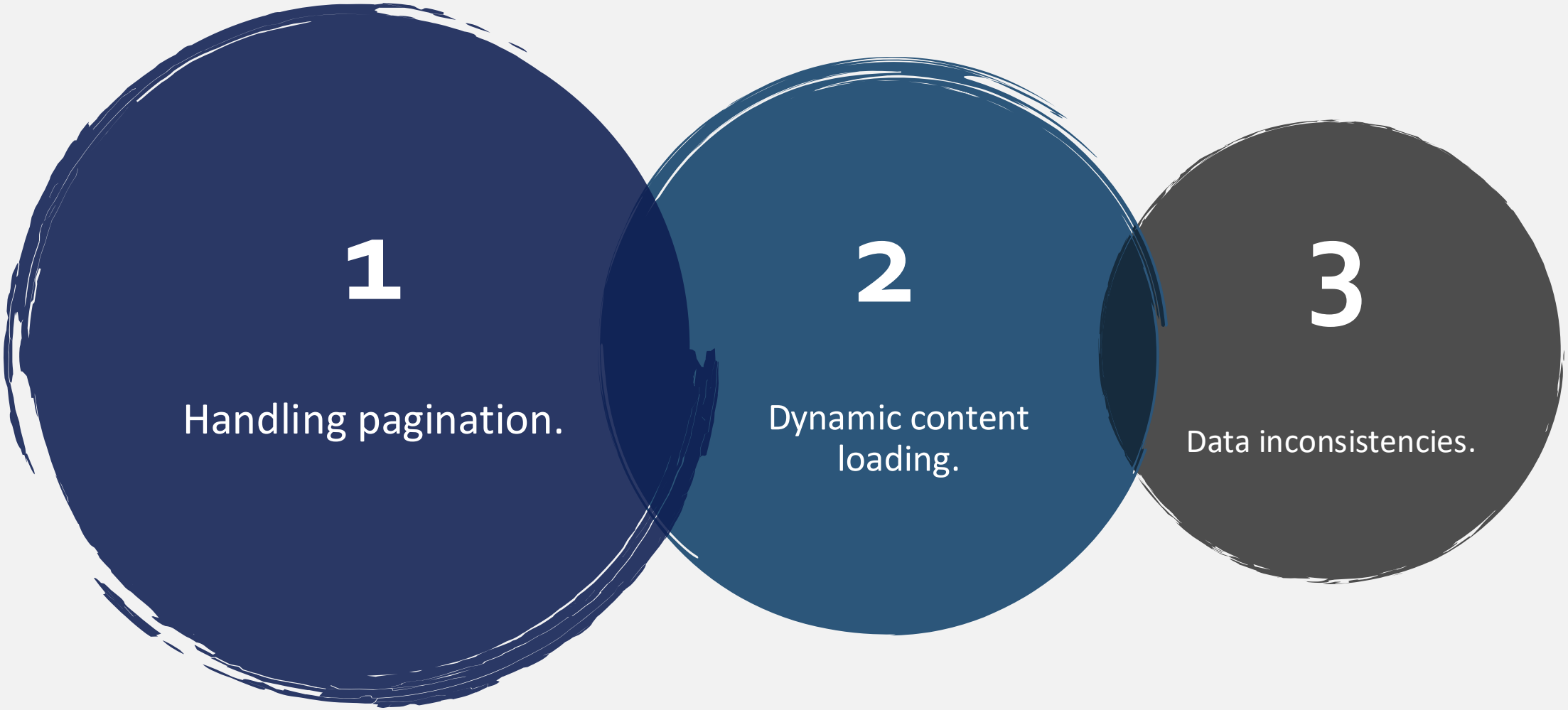


Fig. Output

CHALLENGES FACED -



THANK YOU!!

EVOASTRA VENTURES

DIGITAL SOLUTIONS