Name: Raj Anurag Patil

Project Name: Web Scraping Bickes From Bikewala

Project Guide: Prof.Sameer Warsolkar

1 Aim:

To gather comprehensive information on Bikes from Bikewala website through web scraping, providing valuable insights into the specifications, features, and pricing.

2.Objectives:

1.Data Collection:

- Extract detailed information on bikes models available on bikewala website including specification such as bikes Name, bike rating bike price, bike avg ete.

- Scrape data related to bike feature, such as safety feature entertainment options,

2. pricing analysis :

- Retrieve pricing information for each Bikes model and variant.

- Compare prices across different variants and models to identify trends and variations

3. market Trends :

- Analyze the popularity and demand for bikes by scraping user reviews and ratings.
- Identify key feature and specification that are frequently mentioned in positive and negative reviews.

4.competitor Analysis

- Compare bikes specification and priciping with competitors in the market

5.Data Quality Assurance:

- Address potential challenges such as changes in website structure that might affect the scraping workflow

3. OUTLINE

From this site, we are going to grab the following information:

- bikes Name,
- bike rating
- bike price
- bike avg

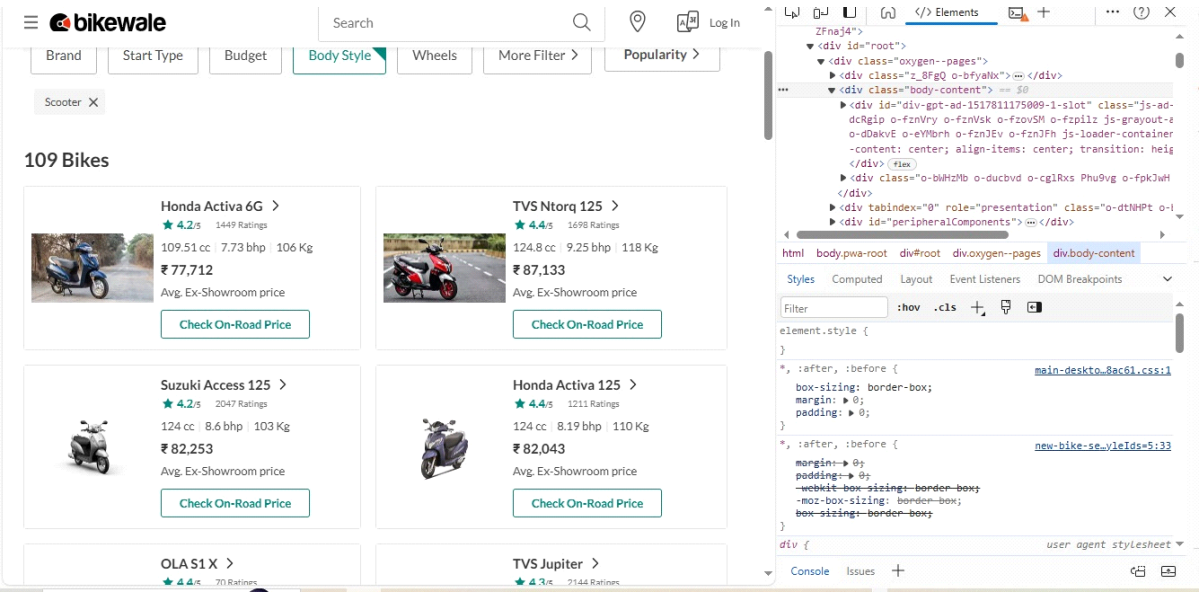
4. STEPS:

➢ The first step is to select the website you want to scrape. We will try to extract data of. Bike for the bike.com

- Inspect the website:

➢ Now the next step is to understand the website structure. Understand what the attributes of the elements that are of your interest are. Right click on the

website to select “Inspect”. This will open HTML code. Use the inspector tool to see the name of all the elements to use in the code.



2. Installing the important libraries:

Python has several web scrapping libraries. We will use the following libraries:

➢ Requests – for making HTTP requests to website

➢ Beautiful Soup – for parsing the HTML code

➢ Pandas – for storing the scraped data in data frame

3. Write the Python source code:

We'll write the main python code. The code will perform the following steps:

- ➢ Using requests to send an HTTP GET requests
- ➢ Using BeautifulSoup to parse the HTML code
- ➢ Extracting the required data from the HTML code
- ➢ Store the information in a pandas DataFrame

4. Exporting the extracted data:

We'll export the data as a CSV file. We will use the panda's library. We'll use the pandas library.

5. Benefits:

- ➢ Access to valuable data for analysis or research.
- ➢ Automation of data collection, saving time and effort.
- ➢ Stay up to date with change changes on the target websites.

6. Risk:

➢ Legal issues related to web scraping.

➢ Technical challenges due to website changes

5. Bikewala web scraping coding command steps

1. Accessing Bike website:

```
[1]: import requests
     from bs4 import BeautifulSoup
     import pandas as pdbbb

[12]: page = requests.get("https://www.bikewale.com/new-bike-search/?bodyStyleIds=5")
      soup = BeautifulSoup(page.content, 'html.parser')
      soup
```

```
<script data-react-helmet="true">window.dataLayer = window.dataLayer || [];
            window.dataLayer.push({"PageCategory":"New","SubCategory":"Search_Page","City":"","Brand":"","Model":"","FuelType":"","ModelId":"","Ma
keId":"","VersionId":""});</script>
<style>
@font-face {
    font-family: 'LatoGoogle';
    font-display: swap;
    src: url('https://stc.aeplcdn.com/fonts/lato-bold.woff2') format('woff2'),
         url('https://stc.aeplcdn.com/fonts/lato-bold.woff') format('woff');
    font-weight: 700;
    font-style: normal;
}

@font-face {
    font-family: 'LatoGoogle';
    font-display: swap;
    src: url('https://stc.aeplcdn.com/fonts/lato-regular.woff2') format('woff2'),
         url('https://stc.aeplcdn.com/fonts/lato-regular.woff') format('woff');
    font-weight: 400;
```

2. Using BeautifulSoup :

```
[1]: import requests
     from bs4 import BeautifulSoup
     import pandas as pdbbb
```

```
[12]: page = requests.get("https://www.bikewale.com/new-bike-search/?bodyStyleIds=5")
      soup = BeautifulSoup(page.content, 'html.parser')
      soup
```

```
<script data-react-helmet="true">window.dataLayer = window.dataLayer || [];
            window.dataLayer.push({"PageCategory":"New","SubCategory":"Search_Page","City":"","Brand":"","Model":"","FuelType":"","ModelId":"","Ma
keId":"","VersionId":""});</script>
<style>
@font-face {
    font-family: 'LatoGoogle';
    font-display: swap;
    src: url('https://stc.aeplcdn.com/fonts/lato-bold.woff2') format('woff2'),
         url('https://stc.aeplcdn.com/fonts/lato-bold.woff') format('woff');
    font-weight: 700;
    font-style: normal;
}

@font-face {
    font-family: 'LatoGoogle';
    font-display: swap;
    src: url('https://stc.aeplcdn.com/fonts/lato-regular.woff2') format('woff2'),
         url('https://stc.aeplcdn.com/fonts/lato-regular.woff') format('woff');
    font-weight: 400;
```

3 check page error :

```
[14]: page.status_code
```

```
[14]: 200
```

4. Accessing the Bike_Names:

```
[16]: Bike_tags=soup.select(".o-eZTujG.o-bkmzIL.o-eqqVmt.o-fzpibr")
      Bike=[]
      for i in Bike_tags:
          Bike.append(i.get_text())
      print(Bike)
```

```
['Honda Activa 6G', 'TVS Ntorq 125', 'Suzuki Access 125', 'Honda Activa 125', 'OLA S1 X', 'TVS Jupiter', 'Honda Dio', 'Suzuki Burgman Street 125', 'TVS
Jupiter 125', 'OLA S1 Pro', 'Bajaj Chetak', 'Yamaha Ray ZR 125', 'Honda Dio 125', 'Hero Xoom', 'TVS iQube', 'Hero Pleasure +', 'Ather 450X', 'Suzuki Ave
nis 125', 'TVS Scooty Pep Plus', 'OLA S1 Air', 'Ather 450S', 'Simple Energy One', 'Aprilia SR 160', 'TVS Scooty Zest 110', 'Hero Pleasure + Xtec', 'Hond
a Grazia', 'Yamaha Fascino 125', 'Vespa VXL 125', 'Aprilia SR 125', 'Hero Maestro Edge 125']
```

```
[17]: len(Bike)
```

```
[17]: 30
```

5. Accessing the Bike Rating:

```
18]: rating_tags=soup.select(".o-eemiLE.o-bdcqVx.o-cKuOoN.o-lIIwF.o-eZTujG")
     rating=[]
     for i in rating_tags:
         rating.append(i.get_text())
     print(rating)
```

```
['4.2/5', '4.4/5', '4.2/5', '4.4/5', '4.4/5', '4.3/5', '4.5/5', '4.4/5', '3.9/5', '4.6/5', '4.2/5', '4.3/5', '4.8/5', '4.4/5', '3.5/5', '4.6/5', '4.2/
5', '4.7/5', '4.2/5', '4.2/5', '4.3/5', '4.6/5', '3.4/5', '4.2/5', '3.6/5', '4.3/5', '4.3/5', '3.5/5', '4.4/5', '4.4/5']
```

```
19]: len(rating)
```

```
19]: 30
```

- The syntax for replace is:
- **str.replace(old, new[, count])**
- old: The substring you want to replace.
- new: The substring you want to replace old with.
- count (optional): The number of times you want to replace old with new. If not provided, all occurrences are replaced.
- replace fuction here use:

cleaned_text = text.replace('/5', '').strip() # Remove the '/5' part and extra whitespace

```
[5]: rating_tags = soup.select(".o-eemiLE.o-bdcqVx.o-cKuOoN.o-lIIwF.o-eZTujG")
     rating = []
     for tag in rating_tags:
         text = tag.get_text()
         cleaned_text = text.replace('/5', '').strip()  # Remove the '/5' part and extra whitespace
         rating.append(cleaned_text)
         print("Ratings:", rating)
```

```
Ratings: ['4.2']
Ratings: ['4.2', '4.4']
Ratings: ['4.2', '4.4', '4.2']
Ratings: ['4.2', '4.4', '4.2', '4.4']
Ratings: ['4.2', '4.4', '4.2', '4.4', '4.4']
Ratings: ['4.2', '4.4', '4.2', '4.4', '4.4', '4.3']
Ratings: ['4.2', '4.4', '4.2', '4.4', '4.4', '4.3', '4.5']
Ratings: ['4.2', '4.4', '4.2', '4.4', '4.4', '4.3', '4.5', '4.4']
Ratings: ['4.2', '4.4', '4.2', '4.4', '4.4', '4.3', '4.5', '4.4', '3.9']
Ratings: ['4.2', '4.4', '4.2', '4.4', '4.4', '4.3', '4.5', '4.4', '3.9', '4.6']
Ratings: ['4.2', '4.4', '4.2', '4.4', '4.4', '4.3', '4.5', '4.4', '3.9', '4.6', '4.2']
Ratings: ['4.2', '4.4', '4.2', '4.4', '4.4', '4.3', '4.5', '4.4', '3.9', '4.6', '4.2', '4.3']
Ratings: ['4.2', '4.4', '4.2', '4.4', '4.4', '4.3', '4.5', '4.4', '3.9', '4.6', '4.2', '4.3', '4.8']
Ratings: ['4.2', '4.4', '4.2', '4.4', '4.4', '4.3', '4.5', '4.4', '3.9', '4.6', '4.2', '4.3', '4.8', '4.4']
Ratings: ['4.2', '4.4', '4.2', '4.4', '4.4', '4.3', '4.5', '4.4', '3.9', '4.6', '4.2', '4.3', '4.8', '4.4', '3.5']
Ratings: ['4.2', '4.4', '4.2', '4.4', '4.4', '4.3', '4.5', '4.4', '3.9', '4.6', '4.2', '4.3', '4.8', '4.4', '3.5', '4.6']
Ratings: ['4.2', '4.4', '4.2', '4.4', '4.4', '4.3', '4.5', '4.4', '3.9', '4.6', '4.2', '4.3', '4.8', '4.4', '3.5', '4.6', '4.2']
Ratings: ['4.2', '4.4', '4.2', '4.4', '4.4', '4.3', '4.5', '4.4', '3.9', '4.6', '4.2', '4.3', '4.8', '4.4', '3.5', '4.6', '4.2', '4.7']
Ratings: ['4.2', '4.4', '4.2', '4.4', '4.4', '4.3', '4.5', '4.4', '3.9', '4.6', '4.2', '4.3', '4.8', '4.4', '3.5', '4.6', '4.2', '4.7', '4.2']
Ratings: ['4.2', '4.4', '4.2', '4.4', '4.4', '4.3', '4.5', '4.4', '3.9', '4.6', '4.2', '4.3', '4.8', '4.4', '3.5', '4.6', '4.2', '4.7', '4.2', '4.2']
Ratings: ['4.2', '4.4', '4.2', '4.4', '4.4', '4.3', '4.5', '4.4', '3.9', '4.6', '4.2', '4.3', '4.8', '4.4', '3.5', '4.6', '4.2', '4.7', '4.2', '4.2',
```

6. Accessing the Bike price:

```
[19]: 30
```

```
[20]: price_tags=soup.select(".o-eZTujG.o-byFsZJ.o-bkmzIL.o-bVSleT")
price=[]
for i in price_tags:
    price.append(i.get_text())
print(price)
```

```
['₹ 77,712', '₹ 87,133', '₹ 82,253', '₹ 82,043', '₹ 79,999', '₹ 76,738', '₹ 74,235', '₹ 96,525', '₹ 85,574', '₹ 1,29,999', '₹ 1,17,285', '₹ 86,779', '₹ 86,147', '₹ 75,513', '₹ 1,55,600', '₹ 70,322', '₹ 1,37,803', '₹ 94,504', '₹ 65,561', '₹ 1,04,999', '₹ 1,27,733', '₹ 1,45,000', '₹ 1,32,734', '₹ 75,208', '₹ 79,117', '₹ 82,408', '₹ 81,979', '₹ 1,32,646', '₹ 1,23,836', '₹ 85,480']
```

```
[21]: len(price)
```

```
[21]: 30
```

The replace function in Python is used to create a new string by replacing all occurrences of a specified substring within the original string with another substring. It does not modify the original string, as strings in Python are immutable, but instead returns a new string with the replacements made.

- The syntax for replace is:
- **str.replace(old, new[, count])**
- old: The substring you want to replace.
- new: The substring you want to replace old with.
- count (optional): The number of times you want to replace old with new. If not provided, all occurrences are replaced.
- replace fuction here use:
- cleaned_text = text.replace('₹', '').strip() # Remove the Rupee symbol and extra whitespace

```
price_tags = soup.select(".o-eZTujG.o-byFsZJ.o-bkmzIL.o-bVSleT")
price = []
for tag in price_tags:
    text = tag.get_text()
    cleaned_text = text.replace('₹', '').strip()  # Remove the Rupee symbol and extra whitespace
    price.append(cleaned_text)
    print("Prices:", price)
```

```
Prices: ['77,712']
Prices: ['77,712', '87,133']
Prices: ['77,712', '87,133', '82,253']
Prices: ['77,712', '87,133', '82,253', '82,043']
Prices: ['77,712', '87,133', '82,253', '82,043', '79,999']
Prices: ['77,712', '87,133', '82,253', '82,043', '79,999', '76,738']
Prices: ['77,712', '87,133', '82,253', '82,043', '79,999', '76,738', '74,235']
Prices: ['77,712', '87,133', '82,253', '82,043', '79,999', '76,738', '74,235', '96,525']
Prices: ['77,712', '87,133', '82,253', '82,043', '79,999', '76,738', '74,235', '96,525', '85,574']
Prices: ['77,712', '87,133', '82,253', '82,043', '79,999', '76,738', '74,235', '96,525', '85,574', '1,29,999']
Prices: ['77,712', '87,133', '82,253', '82,043', '79,999', '76,738', '74,235', '96,525', '85,574', '1,29,999', '1,17,285']
Prices: ['77,712', '87,133', '82,253', '82,043', '79,999', '76,738', '74,235', '96,525', '85,574', '1,29,999', '1,17,285', '86,779']
Prices: ['77,712', '87,133', '82,253', '82,043', '79,999', '76,738', '74,235', '96,525', '85,574', '1,29,999', '1,17,285', '86,779', '86,147']
Prices: ['77,712', '87,133', '82,253', '82,043', '79,999', '76,738', '74,235', '96,525', '85,574', '1,29,999', '1,17,285', '86,779', '86,147', '75,513']
Prices: ['77,712', '87,133', '82,253', '82,043', '79,999', '76,738', '74,235', '96,525', '85,574', '1,29,999', '1,17,285', '86,779', '86,147', '75,513',
'1,55,600']
Prices: ['77,712', '87,133', '82,253', '82,043', '79,999', '76,738', '74,235', '96,525', '85,574', '1,29,999', '1,17,285', '86,779', '86,147', '75,513',
'1,55,600', '70,322']
Prices: ['77,712', '87,133', '82,253', '82,043', '79,999', '76,738', '74,235', '96,525', '85,574', '1,29,999', '1,17,285', '86,779', '86,147', '75,513',
'1,55,600', '70,322', '1,37,803']
Prices: ['77,712', '87,133', '82,253', '82,043', '79,999', '76,738', '74,235', '96,525', '85,574', '1,29,999', '1,17,285', '86,779', '86,147', '75,513',
'1,55,600', '70,322', '1,37,803', '94,504']
Prices: ['77,712', '87,133', '82,253', '82,043', '79,999', '76,738', '74,235', '96,525', '85,574', '1,29,999', '1,17,285', '86,779', '86,147', '75,513',
'1,55,600', '70,322', '1,37,803', '94,504', '65,561']
Prices: ['77,712', '87,133', '82,253', '82,043', '79,999', '76,738', '74,235', '96,525', '85,574', '1,29,999', '1,17,285', '86,779', '86,147', '75,513',
'1,55,600', '70,322', '1,37,803', '94,504', '65,561', '1,04,999']
Prices: ['77,712', '87,133', '82,253', '82,043', '79,999', '76,738', '74,235', '96,525', '85,574', '1,29,999', '1,17,285', '86,779', '86,147', '75,513',
'1,55,600', '70,322', '1,37,803', '94,504', '65,561', '1,04,999', '1,27,733']
```

7. Accessing the Bike Avg:

```
avg_tags=soup.select(".o-bqHweY.o-bVSleT.o-bwCunT.o-cgFpsP.o-bNxxEB.o-cpNAVm.o-eemiLE")
avg=[]
for i in kg_tags:
    avg.append(i.get_text())
print(avg)
```

```
['Avg. Ex-Showroom price', 'Avg. Ex-Showroom price', 'Avg. Ex-Showroom price', 'Avg. Ex-Showroom price', 'Avg. Ex-Showroom price', 'Avg. Ex-Showroom pri
ce', 'Avg. Ex-Showroom price', 'Avg. Ex-Showroom price', 'Avg. Ex-Showroom price', 'Avg. Ex-Showroom price', 'Avg. Ex-Showroom price', 'Avg. Ex-Showroom
price', 'Avg. Ex-Showroom price', 'Avg. Ex-Showroom price', 'Avg. Ex-Showroom price', 'Avg. Ex-Showroom price', 'Avg. Ex-Showroom price', 'Avg. Ex-Showr
oom price', 'Avg. Ex-Showroom price', 'Avg. Ex-Showroom price', 'Avg. Ex-Showroom price', 'Avg. Ex-Showroom price', 'Avg. Ex-Showroom price', 'Avg. Ex-S
howroom price', 'Avg. Ex-Showroom price', 'Avg. Ex-Showroom price', 'Avg. Ex-Showroom price', 'Avg. Ex-Showroom price', 'Avg. Ex-Showroom price', 'Avg.
Ex-Showroom price']
```

```
len(kg)
```

```
30
```

8. Importing the pandas and creating DataFrame

[15]:
```
import pandas as pd
Bikes = pd.DataFrame({
    "Bike Name":Bike,
    "Bike rating":rating,
    "Bike price": price,
    "Bike avg":avg
})
Bikes
```

[15]:

	Bike Name	Bike rating	Bike price	Bike avg
0	Honda Activa 6G	4.2	77,712	Avg. Ex-Showroom price
1	TVS Ntorq 125	4.4	87,133	Avg. Ex-Showroom price
2	Suzuki Access 125	4.2	82,253	Avg. Ex-Showroom price
3	Honda Activa 125	4.4	82,043	Avg. Ex-Showroom price
4	OLA S1 X	4.4	79,999	Avg. Ex-Showroom price
5	TVS Jupiter	4.3	76,738	Avg. Ex-Showroom price
6	Honda Dio	4.5	74,235	Avg. Ex-Showroom price
7	Suzuki Burgman Street 125	4.4	96,525	Avg. Ex-Showroom price
8	TVS Jupiter 125	3.9	85,574	Avg. Ex-Showroom price
9	OLA S1 Pro	4.6	1,29,999	Avg. Ex-Showroom price
10	Bajaj Chetak	4.2	1,17,285	Avg. Ex-Showroom price
11	Yamaha Ray ZR 125	4.3	86,779	Avg. Ex-Showroom price
12	Honda Dio 125	4.8	86,147	Avg. Ex-Showroom price
13	Hero Xoom	4.4	75,513	Avg. Ex-Showroom price
14	TVS iQube	3.5	1,55,600	Avg. Ex-Showroom price
15	Hero Pleasure +	4.6	70,322	Avg. Ex-Showroom price
16	Ather 450X	4.2	1,37,803	Avg. Ex-Showroom price
17	Suzuki Avenis 125	4.7	94,504	Avg. Ex-Showroom price
18	TVS Scooty Pep Plus	4.2	65,561	Avg. Ex-Showroom price

9.converting and Storing DataFrame in the form of a CSV file and opening the file in application:

```
[16]: Bikes.to_csv("Bikes_Report_Scraping.csv",index=False)
```

```
[17]: Read=pd.read_csv("Bikes_Report_Scraping.csv")
      Read
```

[17]:

	Bike Name	Bike rating	Bike price	Bike avg
0	Honda Activa 6G	4.2	77,712	Avg. Ex-Showroom price
1	TVS Ntorq 125	4.4	87,133	Avg. Ex-Showroom price
2	Suzuki Access 125	4.2	82,253	Avg. Ex-Showroom price
3	Honda Activa 125	4.4	82,043	Avg. Ex-Showroom price
4	OLA S1 X	4.4	79,999	Avg. Ex-Showroom price
5	TVS Jupiter	4.3	76,738	Avg. Ex-Showroom price
6	Honda Dio	4.5	74,235	Avg. Ex-Showroom price
7	Suzuki Burgman Street 125	4.4	96,525	Avg. Ex-Showroom price
8	TVS Jupiter 125	3.9	85,574	Avg. Ex-Showroom price
9	OLA S1 Pro	4.6	1,29,999	Avg. Ex-Showroom price
10	Bajaj Chetak	4.2	1,17,285	Avg. Ex-Showroom price
11	Yamaha Ray ZR 125	4.3	86,779	Avg. Ex-Showroom price
12	Honda Dio 125	4.8	86,147	Avg. Ex-Showroom price
13	Hero Xoom	4.4	75,513	Avg. Ex-Showroom price
14	TVS iQube	3.5	1,55,600	Avg. Ex-Showroom price
15	Hero Pleasure +	4.6	70,322	Avg. Ex-Showroom price
16	Ather 450X	4.2	1,37,803	Avg. Ex-Showroom price
17	Suzuki Avenis 125	4.7	94,504	Avg. Ex-Showroom price
18	TVS Scooty Pep Plus	4.2	65,561	Avg. Ex-Showroom price
19	OLA S1 Air	4.2	1,04,999	Avg. Ex-Showroom price
20	Ather 450S	4.3	1,27,733	Avg. Ex-Showroom price

6.Conclusion

In conclusion, the web scraping project focused on bike from bikewala.com has successfully achieved its objectives. The comprehensive data retrieval provided detailed specifications, pricing information, and user sentiments, enriching our understanding of bike offerings This project forms a solid foundation for strategic decision-making and further exploration within the automotive industry


THANK
YOU