# **Market Segmentation Analysis of Electric Vehicles Market in India**

## Date: 8th oct , 2023

~ C.M.RITHIKA

**Fynn Labs: Project 2**



1. IMG

## Problem Statement

Task is to analyse the Electric Vehicles Market in India using *Segmentation* analysis and come up with a feasible strategy to enter the market, targeting the segments most likely to use their product in terms of Geographic, Demographic, Psychographic, and Behavioural.

In this report we analyse the Electric Vehicles Market in India using segments such as region, price, charging facility, type of vehicles (e.g., 2 wheelers, 3 wheelers, 4 wheelers etc.), retail outlets, manufacturers, body type (e.g., Hatchback, Sedan, SUV, Autorickshaw etc.), safety, plug types and much more.

## Fermi Estimation

Wild Guess: Around 8-10% people will have electric vehicles by the end of 2023 in India. Educated Guess:

Employment rate = it is the ratio of number of available labor force to the population of People in the working age.

We think there are about 1.5 billion Indians in the world. Let's assume the only people over18 and under 60 works, assuming that they account for around 60% of the population then that would make 0.9 billion Indians in the working class. Out of the 0.9 billion people not all are employed, assuming only 2023 had 45% employment rate that would bring the number around 405 million.

Since, not everyone can afford an electric vehicle, let’s assume only people above middle class can afford an electric vehicle, that would be 40 million. Not everyone buys an electric vehicle. Let’s assume out of these 40 million only 10 million are willing to buy an electric vehicle.

Variables and Formulas:

Let E(x) be the employment rate of the year x (in %). Let P(x) be the population of the year x.

Let A(x) be the number of available Labor in the year x.

Let r be the ratio of Indians between the age of 18 and 60 to the total population of India.

E(x) = (A(x)\*100)/(P(x)\*r)

This formula will formulate the Employment ratio for the year x.

**Gathering More Information**:

Estimation for the population of the year 2022 can be obtained by the increase in population each year

P (2019) = 1.3676 billion

P (2020) = 1.3786 billion

P (2021) = 1.39199 billion

P (2020)-P (2019) = 11million P (2021)-P (2020) = 13.39 million the mean would be 12.195 million

thus P (2022) = 1.44185 billion assuming A(x) is constant every year= 471,688,990r=0.6 C=0.75

E (2022) = (471,688,990/(1,441,850,000\*0.6))\*0.75 E (2022) = 42%

Conclusion: By this analysis, we conclude that by the end of the year 2024 there would a Employment rate of 42%. That would make 42% of 405 million i.e., 170 million. Out of these 170 million only 10% afford EV'S. So around 17 million people will have EV's by the end of 2024"

## Data Collection

Data was extracted from the various websites mentioned below for EV market segmentation.

Link for data extraction:

* <https://pib.gov.in/PressReleasePage.aspx?PRID=1842704> [https://www.ibef.org/blogs/electricvehicles-market-in-india](https://www.ibef.org/blogs/electric-vehicles-market-in-india)<https://evreporter.com/indias-region-wise-ev-market-jan-may-2022/>
* [https://www.india-briefing.com/news/indias-ev-manufacturing-capacity-and-marketpreferences- progress-25840.html/](https://www.india-briefing.com/news/indias-ev-manufacturing-capacity-and-market-preferences-progress-25840.html/)
* [https://github.com/Marisha18/Market-Segmentation-for-Electric-Vehicles-in-](https://github.com/Marisha18/Market-Segmentation-for-Electric-Vehicles-in-India/blob/main/Market_Segmentation.ipynb)

[India/blob/main/Market\_Segmentation.ipynb](https://github.com/Marisha18/Market-Segmentation-for-Electric-Vehicles-in-India/blob/main/Market_Segmentation.ipynb) [https://github.com/Ashwini3535/EV-MARKETIN-INDIA](https://github.com/Ashwini3535/EV-MARKET-IN-INDIA)

Data from those links are extracted by Google play scraper available on libraries package. There are multiple datasets get extracted from those websites in CSV and Excel formats. There are some pdfs also which contains valuable information regarding the EV market. We have extracted data from those pdfs as well.

Raw data generated:

* https://github.com/ShubhamNavghare/FeyNN\_Labs\_Project\_2[EV\_Market\_Segmentation/tree/main/Dataset](https://github.com/ShubhamNavghare/FeyNN_Labs_Project_2-EV_Market_Segmentation/tree/main/Dataset)
* [https://github.com/ShubhamNavghare/FeyNN\_Labs\_Project\_2-](https://github.com/ShubhamNavghare/FeyNN_Labs_Project_2-EV_Market_Segmentation/tree/main/PDF)

[EV\_Market\_Segmentation/tree/main/PDF](https://github.com/ShubhamNavghare/FeyNN_Labs_Project_2-EV_Market_Segmentation/tree/main/PDF)

Columns explanations:

1. ‘Brand’ and tells the manufacturers of electric vehicles.
2. ‘model’ tells the various of electric vehicles.
3. ‘Accuse’, ‘Top Speed’, ’Power Train’ tells specification about the vehicles.
4. ‘Range’, ‘Fast Charge’, ‘Plug\_type’ and **‘** Bodystyle’ tells us about range of vehicle per full charge,fast charging is provided or not, type of charging plug and body style of vehicle respectively.
5. ‘Seats’ and ‘Price’ tells about the number of seats available on vehicle and their price.
6. ‘Region’ and ‘State/UT’ tells about the states of India.
7. ‘EV Charging Facility’ and ‘Chargers’ tells about the facility of charging in the respective states.
8. ‘2V’, ‘3V’, ‘4V’, ‘Bus’ tells about the type of vehicles in the market.

## Data Preprocessing

Steps taken to preprocess the scraped raw data:

1. Ordinal encoded 'PowerTrain'
2. Label encoded 'RapidCharge’
3. Used Label Encoder and Standard Scaler package for preprocessing of the dataset.

## Exploratory Data Analysis

An Exploratory Data Analysis or EDA is a thorough examination meant to uncover the underlying structure of a data set and is important for a company because it exposes trends, patterns, and relationships that are not readily apparent.

We analyzed our dataset using *univariate* (analyze data over a single variable/column from a dataset), *bivariate* (analyze data by taking two variables/columns into consideration from a dataset) and *multivariate* (analyze data by taking more than two variables/columns into consideration from a dataset) analysis.

The bar graph below shows the diversity of the data geographically. We can see that we have the maximum amount of data of states *Karnataka* and *Maharashtra*; and minimum amount of data for *Sikkim, Meghalaya, Lakshadweep, Ladakh,* and *Dadra and Nagar Haveli and Daman and Diu*. There are a total of 1536 rows of data distributed among the cities shown in the graph.

**IMPLEMENTATION CODE**

**DATASET-1**

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

**from** sklearn.model\_selection **import** train\_test\_split

**from** sklearn.preprocessing **import** StandardScaler

**from** sklearn.decomposition **import** PCA

**from** sklearn.cluster **import** KMeans

In [14]:

df1 **=** pd**.**read\_csv('1\_ev\_charger\_dataset.csv')

df1**.**head()

Out[14]:

|  | **Region** | **2W** | **3W** | **4W** | **Bus** | **Chargers** |
| --- | --- | --- | --- | --- | --- | --- |
| **0** | Uttar Pradesh | 9852 | 42881 | 458 | 197 | 207 |
| **1** | Maharastra | 38558 | 893 | 1895 | 186 | 317 |
| **2** | Karnataka | 32844 | 568 | 589 | 57 | 172 |
| **3** | Tamil Nadu | 25642 | 396 | 426 | 0 | 256 |
| **4** | Gujarat | 22359 | 254 | 423 | 22 | 228 |

In [15]:

print('DF1 Shape: ', df1**.**shape)

DF1 Shape: (24, 6)

In [16]:

print(' <<< DATASET 1 -----------------------------------------------------------')

print(df1**.**info())

<<< DATASET 1 -----------------------------------------------------------

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 24 entries, 0 to 23

Data columns (total 6 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Region 24 non-null object

1 2W 24 non-null int64

2 3W 24 non-null int64

3 4W 24 non-null int64

4 Bus 24 non-null int64

5 Chargers 24 non-null int64

dtypes: int64(5), object(1)

memory usage: 1.2+ KB

None

In [17]:

d1 **=** df1**.**describe()

display('<<< DATASET 1 >>>', d1,)

'<<< DATASET 1 >>>'

|  | **2W** | **3W** | **4W** | **Bus** | **Chargers** |
| --- | --- | --- | --- | --- | --- |
| **count** | 24.000000 | 24.000000 | 24.000000 | 24.000000 | 24.000000 |
| **mean** | 8421.458333 | 3853.166667 | 334.041667 | 28.500000 | 106.791667 |
| **std** | 10942.261145 | 8850.690961 | 476.930628 | 63.771331 | 96.623869 |
| **min** | 187.000000 | 234.000000 | 12.000000 | 0.000000 | 10.000000 |
| **25%** | 848.000000 | 512.750000 | 34.750000 | 0.000000 | 25.000000 |
| **50%** | 2967.500000 | 931.000000 | 129.000000 | 0.000000 | 67.500000 |
| **75%** | 10697.750000 | 2659.250000 | 434.000000 | 5.500000 | 180.250000 |
| **max** | 38558.000000 | 42881.000000 | 1895.000000 | 197.000000 | 317.000000 |

In [18]:

plt**.**figure(figsize**=**(6, 6))

sns**.**barplot(data**=**df1, y**=**df1['Region']**.**sort\_values(ascending**=True**), x**=**'2W', palette**=**'viridis')

plt**.**ylabel('State', fontsize**=**14, family**=**'serif')

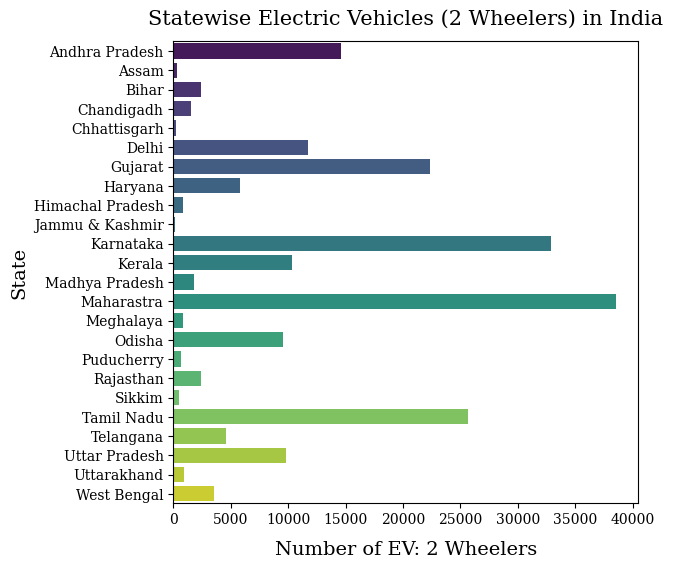
plt**.**xlabel('Number of EV: 2 Wheelers', family**=**'serif', fontsize**=**14, labelpad**=**10)

plt**.**xticks(family**=**'serif')

plt**.**yticks(family**=**'serif')

plt**.**title(label**=**'Statewise Electric Vehicles (2 Wheelers) in India', weight**=**200, family**=**'serif', size**=**15, pad**=**12)

plt**.**show()



In [19]:

plt**.**figure(figsize**=**(6, 6))

sns**.**barplot(data**=**df1, y**=**df1['Region']**.**sort\_values(ascending**=True**), x**=**'3W', palette**=**'viridis')

plt**.**ylabel('State', fontsize**=**14, family**=**'serif')

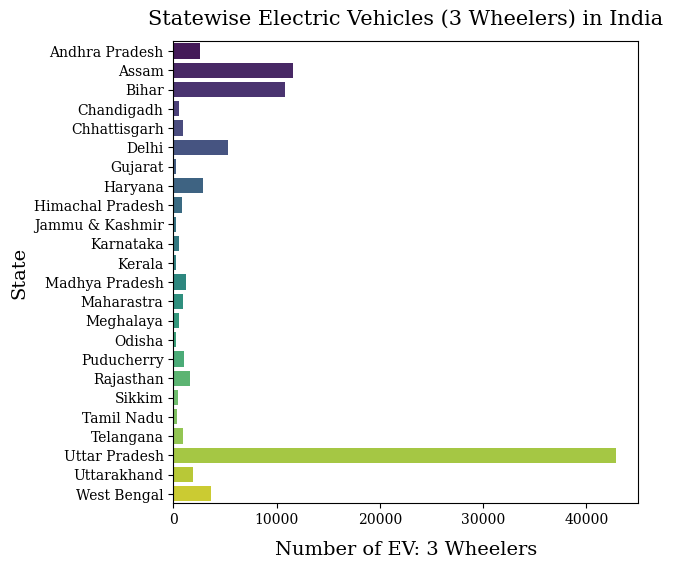
plt**.**xlabel('Number of EV: 3 Wheelers', family**=**'serif', fontsize**=**14, labelpad**=**10)

plt**.**xticks(family**=**'serif')

plt**.**yticks(family**=**'serif')

plt**.**title(label**=**'Statewise Electric Vehicles (3 Wheelers) in India', weight**=**200, family**=**'serif', size**=**15, pad**=**12)

plt**.**show()



In [20]:

plt**.**figure(figsize**=**(6, 6))

sns**.**barplot(data**=**df1, y**=**df1['Region']**.**sort\_values(ascending**=True**), x**=**'4W', palette**=**'viridis')

plt**.**ylabel('State', fontsize**=**14, family**=**'serif')

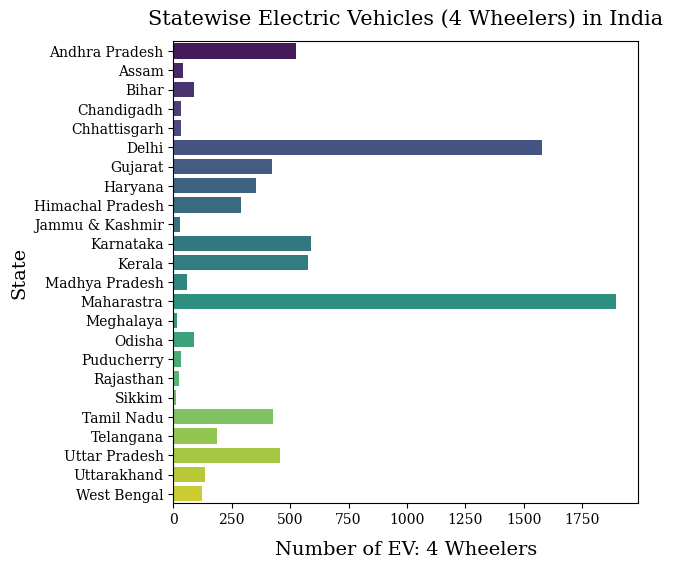
plt**.**xlabel('Number of EV: 4 Wheelers', family**=**'serif', fontsize**=**14, labelpad**=**10)

plt**.**xticks(family**=**'serif')

plt**.**yticks(family**=**'serif')

plt**.**title(label**=**'Statewise Electric Vehicles (4 Wheelers) in India', weight**=**200, family**=**'serif', size**=**15, pad**=**12)

plt**.**show()



In [21]:

plt**.**figure(figsize**=**(6, 6))

sns**.**barplot(data**=**df1, y**=**df1['Region']**.**sort\_values(ascending**=True**), x**=**'Chargers', palette**=**'viridis')

plt**.**ylabel('State', fontsize**=**14, family**=**'serif')

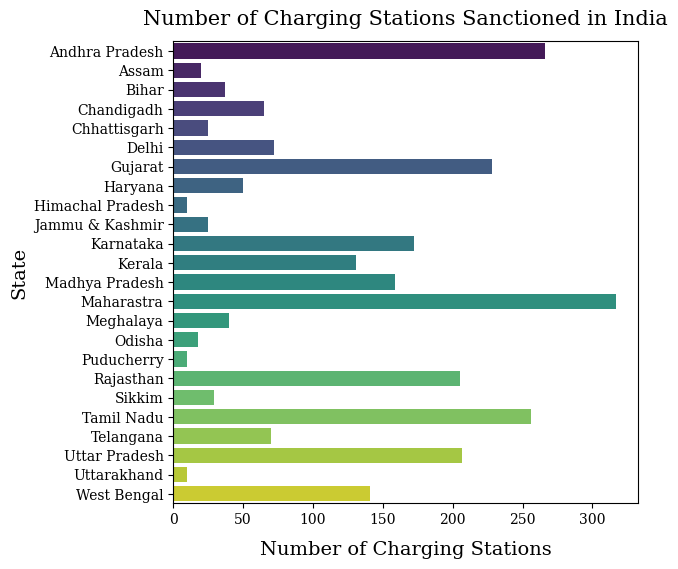
plt**.**xlabel('Number of Charging Stations', family**=**'serif', fontsize**=**14, labelpad**=**10)

plt**.**xticks(family**=**'serif')

plt**.**yticks(family**=**'serif')

plt**.**title(label**=**'Number of Charging Stations Sanctioned in India', weight**=**200, family**=**'serif', size**=**15, pad**=**12)

plt**.**show()



In [23]:

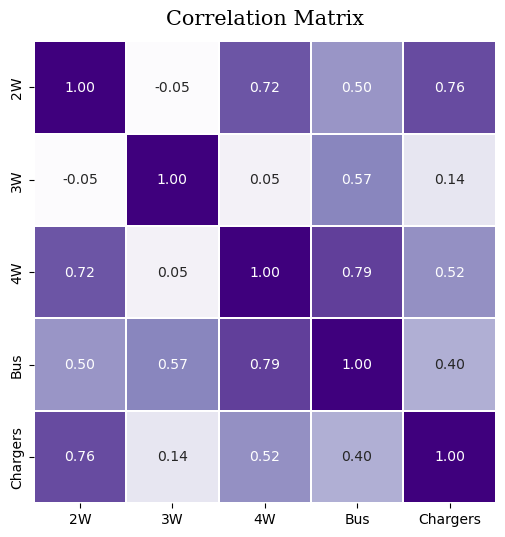
plt**.**figure(figsize**=**(6,6))

sns**.**heatmap(data**=**df1**.**corr(), annot**=True**, cmap**=**'Purples', cbar**=False**, square**=True**, fmt**=**'.2f', linewidths**=**.3)

plt**.**title('Correlation Matrix', family**=**'serif', size**=**15, pad**=**12);

C:\Users\Nethranand PS\AppData\Local\Temp\ipykernel\_7840\1643327926.py:2: FutureWarning: The default value of numeric\_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric\_only to silence this warning.

sns.heatmap(data=df1.corr(), annot=True, cmap='Purples', cbar=False, square=True, fmt='.2f', linewidths=.3)



In [27]:

X **=** df1[['Region','2W','3W','4W', 'Bus', 'Chargers']]

In [30]:

**from** sklearn.preprocessing **import** StandardScaler, LabelEncoder

data**=**X

label\_encoder **=** LabelEncoder()

data['Region'] **=** label\_encoder**.**fit\_transform(data['Region'])

data['2W'] **=** label\_encoder**.**fit\_transform(data['2W'])

data['3W'] **=** label\_encoder**.**fit\_transform(data['3W'])

data['4W'] **=** label\_encoder**.**fit\_transform(data['4W'])

data['Bus'] **=** label\_encoder**.**fit\_transform(data['Bus'])

data['Chargers'] **=** label\_encoder**.**fit\_transform(data['Chargers'])

*# Scale numerical variables*

scaler **=** StandardScaler()

data\_scaled **=** scaler**.**fit\_transform(data)

*# Convert back to DataFrame*

data\_scaled **=** pd**.**DataFrame(data\_scaled, columns**=**data**.**columns)

In [32]:

kmean **=** KMeans(n\_clusters**=**4, init**=**'k-means++', random\_state**=**90)

kmean**.**fit(data)

C:\Users\Nethranand PS\anaconda3\lib\site-packages\sklearn\cluster\\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning

warnings.warn(

C:\Users\Nethranand PS\anaconda3\lib\site-packages\sklearn\cluster\\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP\_NUM\_THREADS=1.

warnings.warn(

Out[32]:

KMeans

KMeans(n\_clusters=4, random\_state=90)

In [33]:

print(kmean**.**labels\_)

[2 3 3 3 3 1 1 1 3 0 1 2 2 2 2 0 1 0 0 0 0 2 0 0]

In [34]:

pd**.**Series(kmean**.**labels\_)**.**value\_counts()

Out[34]:

0 8

2 6

3 5

1 5

dtype: int64

In [35]:

df1['clusters'] **=** kmean**.**labels\_

In [47]:

**from** sklearn.model\_selection **import** train\_test\_split

data **=** data**.**dropna()

x**=**data['2W']

y**=**data['Chargers']

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(x,y, test\_size**=**0.2, random\_state**=**42)

In [51]:

**from** sklearn.cluster **import** KMeans

kmeans\_model **=** KMeans(n\_clusters**=**4)

kmeans\_model**.**fit(data)

kmeans\_labels **=** kmeans\_model**.**predict(data)

C:\Users\Nethranand PS\anaconda3\lib\site-packages\sklearn\cluster\\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning

warnings.warn(

C:\Users\Nethranand PS\anaconda3\lib\site-packages\sklearn\cluster\\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP\_NUM\_THREADS=1.

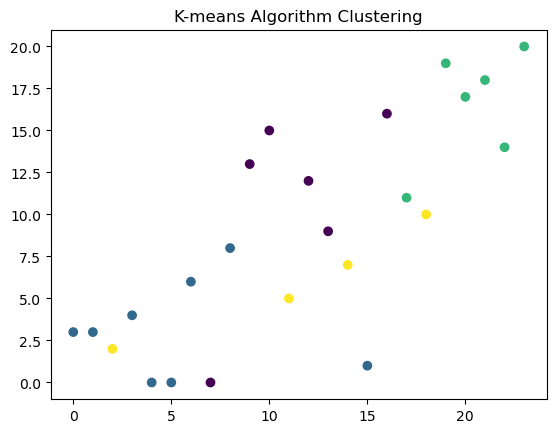
warnings.warn(

In [52]:

plt**.**scatter(data['2W'], data['Chargers'], c**=**kmeans\_labels, cmap**=**'viridis')

plt**.**title('K-means Algorithm Clustering')

plt**.**show()



In [1]:

**Dataset-2**

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

**from** sklearn.model\_selection **import** train\_test\_split

**from** sklearn.preprocessing **import** StandardScaler

**from** sklearn.decomposition **import** PCA

**from** sklearn.cluster **import** KMeans

In [2]:

df2 **=** pd**.**read\_csv('ElectricCarData\_Clean.csv')

df2**.**head()

Out[2]:

|  | **Brand** | **Model** | **AccelSec** | **TopSpeed\_KmH** | **Range\_Km** | **Efficiency\_WhKm** | **FastCharge\_KmH** | **RapidCharge** | **PowerTrain** | **PlugType** | **BodyStyle** | **Segment** | **Seats** | **PriceEuro** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | Tesla | Model 3 Long Range Dual Motor | 4.6 | 233 | 450 | 161 | 940 | Yes | AWD | Type 2 CCS | Sedan | D | 5 | 55480 |
| **1** | Volkswagen | ID.3 Pure | 10.0 | 160 | 270 | 167 | 250 | Yes | RWD | Type 2 CCS | Hatchback | C | 5 | 30000 |
| **2** | Polestar | 2 | 4.7 | 210 | 400 | 181 | 620 | Yes | AWD | Type 2 CCS | Liftback | D | 5 | 56440 |
| **3** | BMW | iX3 | 6.8 | 180 | 360 | 206 | 560 | Yes | RWD | Type 2 CCS | SUV | D | 5 | 68040 |
| **4** | Honda | e | 9.5 | 145 | 170 | 168 | 190 | Yes | RWD | Type 2 CCS | Hatchback | B | 4 | 32997 |

In [3]:

print('DF2 Shape: ', df2**.**shape)

DF2 Shape: (103, 14)

In [5]:

print(' <<< DATASET 1 -----------------------------------------------------------')

print(df2**.**info())

<<< DATASET 1 -----------------------------------------------------------

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 103 entries, 0 to 102

Data columns (total 14 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Brand 103 non-null object

1 Model 103 non-null object

2 AccelSec 103 non-null float64

3 TopSpeed\_KmH 103 non-null int64

4 Range\_Km 103 non-null int64

5 Efficiency\_WhKm 103 non-null int64

6 FastCharge\_KmH 103 non-null object

7 RapidCharge 103 non-null object

8 PowerTrain 103 non-null object

9 PlugType 103 non-null object

10 BodyStyle 103 non-null object

11 Segment 103 non-null object

12 Seats 103 non-null int64

13 PriceEuro 103 non-null int64

dtypes: float64(1), int64(5), object(8)

memory usage: 11.4+ KB

None

In [9]:

d2 **=** df2**.**describe()

display('<<< DATASET 2 >>>', d2)

'<<< DATASET 2 >>>'

|  | **AccelSec** | **TopSpeed\_KmH** | **Range\_Km** | **Efficiency\_WhKm** | **Seats** | **PriceEuro** |
| --- | --- | --- | --- | --- | --- | --- |
| **count** | 103.000000 | 103.000000 | 103.000000 | 103.000000 | 103.000000 | 103.000000 |
| **mean** | 7.396117 | 179.194175 | 338.786408 | 189.165049 | 4.883495 | 55811.563107 |
| **std** | 3.017430 | 43.573030 | 126.014444 | 29.566839 | 0.795834 | 34134.665280 |
| **min** | 2.100000 | 123.000000 | 95.000000 | 104.000000 | 2.000000 | 20129.000000 |
| **25%** | 5.100000 | 150.000000 | 250.000000 | 168.000000 | 5.000000 | 34429.500000 |
| **50%** | 7.300000 | 160.000000 | 340.000000 | 180.000000 | 5.000000 | 45000.000000 |
| **75%** | 9.000000 | 200.000000 | 400.000000 | 203.000000 | 5.000000 | 65000.000000 |
| **max** | 22.400000 | 410.000000 | 970.000000 | 273.000000 | 7.000000 | 215000.000000 |

In [10]:

plt**.**figure(figsize**=**(6, 6))

sns**.**barplot(data**=**df2, y**=**df2['Brand']**.**sort\_values(ascending**=True**), x**=**'PriceEuro', palette**=**'viridis')

plt**.**ylabel('Brand', fontsize**=**14, family**=**'serif')

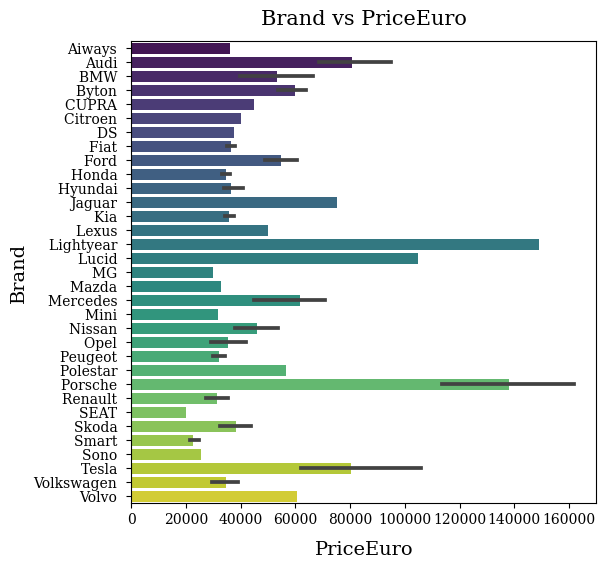
plt**.**xlabel('PriceEuro', family**=**'serif', fontsize**=**14, labelpad**=**10)

plt**.**xticks(family**=**'serif')

plt**.**yticks(family**=**'serif')

plt**.**title(label**=**'Brand vs PriceEuro', weight**=**200, family**=**'serif', size**=**15, pad**=**12)

plt**.**show()



In [12]:

plt**.**figure(figsize**=**(6, 6))

sns**.**barplot(data**=**df2, y**=**df2['Brand']**.**sort\_values(ascending**=True**), x**=**'Efficiency\_WhKm', palette**=**'viridis')

plt**.**ylabel('Brand', fontsize**=**14, family**=**'serif')

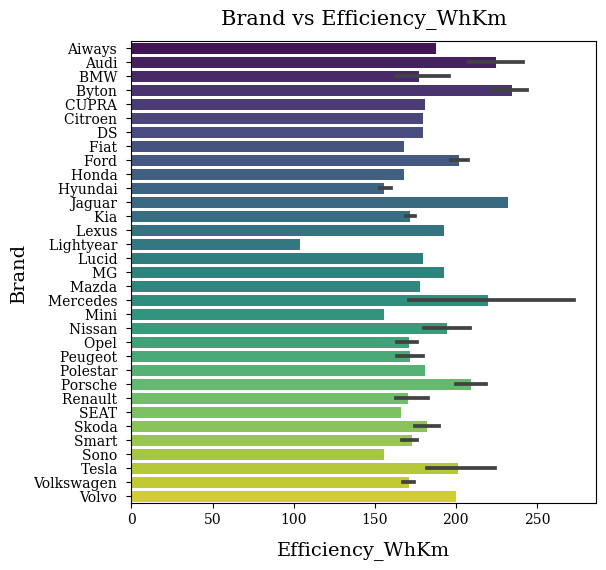
plt**.**xlabel('Efficiency\_WhKm', family**=**'serif', fontsize**=**14, labelpad**=**10)

plt**.**xticks(family**=**'serif')

plt**.**yticks(family**=**'serif')

plt**.**title(label**=**'Brand vs Efficiency\_WhKm', weight**=**200, family**=**'serif', size**=**15, pad**=**12)

plt**.**show()



In [13]:

plt**.**figure(figsize**=**(6, 6))

sns**.**barplot(data**=**df2, y**=**df2['Brand']**.**sort\_values(ascending**=True**), x**=**'Range\_Km', palette**=**'viridis')

plt**.**ylabel('Brand', fontsize**=**14, family**=**'serif')

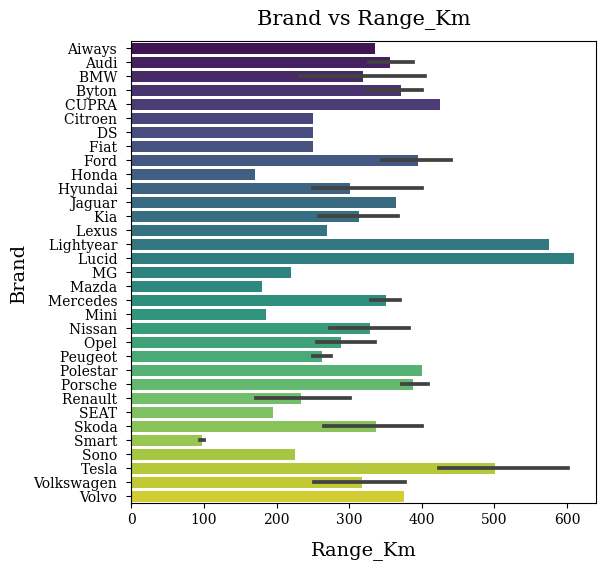
plt**.**xlabel('Range\_Km', family**=**'serif', fontsize**=**14, labelpad**=**10)

plt**.**xticks(family**=**'serif')

plt**.**yticks(family**=**'serif')

plt**.**title(label**=**'Brand vs Range\_Km', weight**=**200, family**=**'serif', size**=**15, pad**=**12)

plt**.**show()



In [14]:

plt**.**figure(figsize**=**(6, 6))

sns**.**barplot(data**=**df2, y**=**df2['Brand']**.**sort\_values(ascending**=True**), x**=**'AccelSec', palette**=**'viridis')

plt**.**ylabel('Brand', fontsize**=**14, family**=**'serif')

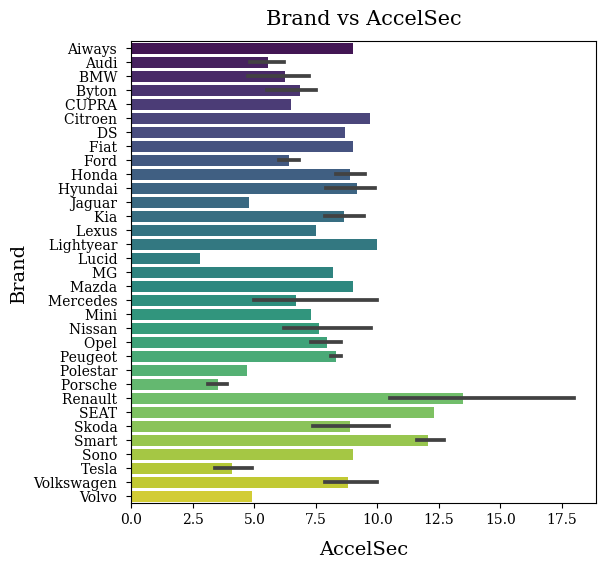
plt**.**xlabel('AccelSec', family**=**'serif', fontsize**=**14, labelpad**=**10)

plt**.**xticks(family**=**'serif')

plt**.**yticks(family**=**'serif')

plt**.**title(label**=**'Brand vs AccelSec', weight**=**200, family**=**'serif', size**=**15, pad**=**12)

plt**.**show()



In [15]:

plt**.**figure(figsize**=**(6,6))

sns**.**heatmap(data**=**df2**.**corr(), annot**=True**, cmap**=**'Purples', cbar**=False**, square**=True**, fmt**=**'.2f', linewidths**=**.3)

plt**.**title('Correlation Matrix', family**=**'serif', size**=**15, pad**=**12);

C:\Users\Nethranand PS\AppData\Local\Temp\ipykernel\_5136\3985166321.py:2: FutureWarning: The default value of numeric\_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric\_only to silence this warning.

sns.heatmap(data=df2.corr(), annot=True, cmap='Purples', cbar=False, square=True, fmt='.2f', linewidths=.3)

