JULY 4, 2023

ELECTRIC VEHICLE STARTUP MARKET SEGMENTATION ANALYSIS

TEAM LEADER – KHUSHI GUPTA

TEAM MEMBERS – AISHWARYA ANN JOSEPH

VAIBHAV DUTTA

SUDIKSHA BHAT M

Problem Statement: Establishing an electric vehicle startup in India requires a comprehensive market segmentation analysis to identify viable opportunities and effectively target the right customer segments. The challenge is to analyze the diverse Indian market landscape, including demographics, economic conditions, and consumer preferences, to strategize the optimal product offerings and marketing approaches. The startup must navigate through the complexities of regional variations, charging infrastructure availability, and government regulations to devise a well-informed market segmentation strategy that addresses the specific needs of each target segment, ultimately ensuring successful market penetration and sustained growth in the highly competitive electric vehicle industry.

DATA ANALYSIS DONE BY TEAM MEMBERS:

KHUSHI GUPTA – Analysed the features of present car brands in the market and what we can include in our electric cars

AISHWARYA ANN JOSEPH – Analysed the user ratings of electric vehicles

VAIBHAV DUTTA – Analysed which part of India will be suitable for setting startup

SUDIKSHA BHAT M– Analysed the total electric and non-electric vehicles present in India

ANALYSIS DONE BY KHUSHI GUPTA

INTRODUCTION:

This analysis will be useful for gaining insights into the existing Electric Vehicle Brands and their features. What should be included in our Electric Vehicles, and how we can provide better features to our customers so that our brand can gain popularity after it enters the market.

Data Source 1:

https://github.com/Khhushhiiii/Electric-Vehicles-Feature-Analysis/blob/main/FEV%20data.xlsx

Libraries Used:

```
[3]  #importing required libraries
     %pip install plotly==5.8.0
     import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
     %matplotlib inline
     from tqdm import tqdm
     import statsmodels.api as sm
     import plotly.express as px
     from google.colab import files
     %pip install kaleido
     import kaleido
     from sklearn.preprocessing import StandardScaler,PowerTransformer
     from sklearn.decomposition import PCA
     from scipy.cluster.hierarchy import dendrogram, linkage
     from sklearn.cluster import KMeans, MeanShift, estimate_bandwidth
     from sklearn.datasets import make_blobs
     from yellowbrick.cluster import KElbowVisualizer, SilhouetteVisualizer, InterclusterDistance
     from collections import Counter
     from sklearn.model_selection import cross_validate,train_test_split
     from sklearn.linear_model import LinearRegression,LogisticRegression
     from sklearn import metrics
     from sklearn.metrics import r2_score,silhouette_score,confusion_matrix,accuracy_score
     pd.set_option("display.precision",3)
     np.set_printoptions(precision=5, suppress=True)
     pd.options.display.float_format = '{:.4f}'.format
     import plotly.io as pio

     pio.renderers.default = "svg"
```

Data Preprocessing:

Step 1: Reading the dataset using the pandas library

```
[ ] data = pd.read_excel('/content/FEV data.xlsx')
```

```
[ ] data.head(10)
```

	Car full name	Make	Model	Minimal price (gross) [PLN]	Engine power [KM]	Maximum torque [Nm]	Type of brakes	Drive type	Battery capacity [kWh]	Range (WLTP) [km]	...	Permissable gross weight [kg]	Maximum load capacity [kg]	Number of seats	Number of doors	Tire size [in]	Maximum speed [kph]	Boot capacity (VDA) [l]	Acceleration 0-100 kph [s]	Maximum DC charging power [kW]	mean - Energy consumption [kWh/100 km]
0	Audi e-tron 55 quattro	Audi	e-tron 55 quattro	345700	360	664	disc (front + rear)	AWD	95.0000	438	...	3130.0000	640.0000	5	5	19	200	660.0000	5.7000	150	24.4500
1	Audi e-tron 50 quattro	Audi	e-tron 50 quattro	308400	313	540	disc (front + rear)	AWD	71.0000	340	...	3040.0000	670.0000	5	5	19	190	660.0000	6.8000	150	23.8000
2	Audi e-tron S quattro	Audi	e-tron S quattro	414900	503	973	disc (front + rear)	AWD	95.0000	364	...	3130.0000	565.0000	5	5	20	210	660.0000	4.5000	150	27.5500
3	Audi e-tron Sportback 50 quattro	Audi	e-tron Sportback 50 quattro	319700	313	540	disc (front + rear)	AWD	71.0000	346	...	3040.0000	640.0000	5	5	19	190	615.0000	6.8000	150	23.3000
4	Audi e-tron Sportback 55 quattro	Audi	e-tron Sportback 55 quattro	357000	360	664	disc (front + rear)	AWD	95.0000	447	...	3130.0000	670.0000	5	5	19	200	615.0000	5.7000	150	23.8500
5	Audi e-tron Sportback S quattro	Audi	e-tron Sportback S quattro	426200	503	973	disc (front + rear)	AWD	95.0000	369	...	3130.0000	565.0000	5	5	20	210	615.0000	4.5000	150	27.2000
6	BMW i3	BMW	i3	169700	170	250	disc (front + rear)	RWD (rear)	42.2000	359	...	1730.0000	440.0000	4	5	19	160	260.0000	8.1000	50	13.1000
7	BMW i3s	BMW	i3s	184200	184	270	disc (front + rear)	RWD (rear)	42.2000	345	...	1730.0000	440.0000	4	5	20	160	260.0000	6.9000	50	14.3000
8	BMW iX3	BMW	iX3	282900	286	400	disc (front + rear)	RWD (rear)	80.0000	460	...	2725.0000	540.0000	5	5	19	180	510.0000	6.8000	150	18.8000
9	Citroën ë-C4	Citroën	ë-C4	125000	136	260	disc (front + rear)	FWD (front)	50.0000	350	...	2000.0000	459.0000	5	5	16	150	380.0000	9.5000	100	NaN

10 rows × 25 columns

Step 2: Data type

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 53 entries, 0 to 52
Data columns (total 25 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   Car full name                         53 non-null     object
 1   Make                                  53 non-null     object
 2   Model                                 53 non-null     object
 3   Minimal price (gross) [PLN]           53 non-null     int64
 4   Engine power [KM]                     53 non-null     int64
 5   Maximum torque [Nm]                   53 non-null     int64
 6   Type of brakes                        52 non-null     object
 7   Drive type                            53 non-null     object
 8   Battery capacity [kWh]                53 non-null     float64
 9   Range (WLTP) [km]                     53 non-null     int64
 10  Wheelbase [cm]                        53 non-null     float64
 11  Length [cm]                           53 non-null     float64
 12  Width [cm]                            53 non-null     float64
 13  Height [cm]                           53 non-null     float64
 14  Minimal empty weight [kg]             53 non-null     int64
 15  Permissable gross weight [kg]         45 non-null     float64
 16  Maximum load capacity [kg]            45 non-null     float64
 17  Number of seats                       53 non-null     int64
 18  Number of doors                       53 non-null     int64
 19  Tire size [in]                        53 non-null     int64
 20  Maximum speed [kph]                   53 non-null     int64
 21  Boot capacity (VDA) [l]               52 non-null     float64
 22  Acceleration 0-100 kph [s]            50 non-null     float64
 23  Maximum DC charging power [kW]        53 non-null     int64
 24  mean - Energy consumption [kWh/100 km]  44 non-null   float64
dtypes: float64(10), int64(10), object(5)
memory usage: 10.5+ KB
```

DATA EXPLORATION:

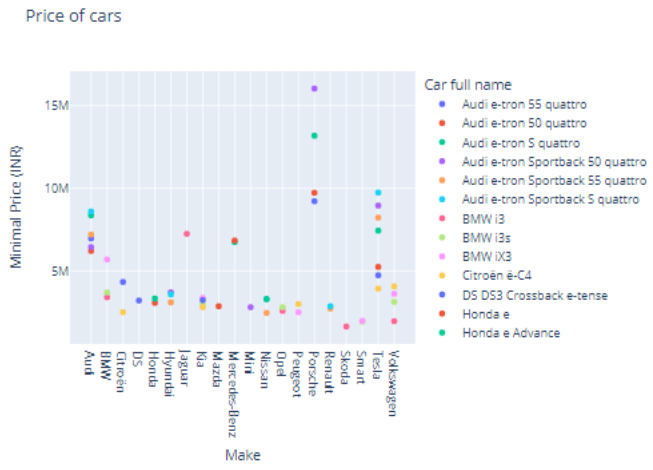
Mean of all the columns

```
data.mean(axis = 0)
```

```
<ipython-input-56-1f5f4b5c2b1f>:1: FutureWarning:

The default value of numeric_only in DataFrame.mean is deprecated.

Minimal price (gross) [PLN]              246158.5094
Engine power [KM]                           269.7736
Maximum torque [Nm]                         460.0377
Battery capacity [kWh]                       62.3660
Range (WLTP) [km]                           376.9057
Wheelbase [cm]                              273.5811
Length [cm]                                 442.5094
Width [cm]                                  186.2415
Height [cm]                                 155.4226
Minimal empty weight [kg]                  1868.4528
Permissable gross weight [kg]              1943.3585
Maximum load capacity [kg]                  441.9057
Number of seats                               4.9057
Number of doors                               4.8491
Tire size [in]                               17.6792
Maximum speed [kph]                         178.1698
Boot capacity (VDA) [l]                     436.6981
Acceleration 0-100 kph [s]                    6.9434
Maximum DC charging power [kW]              113.5094
mean - Energy consumption [kWh/100 km]       15.7689
Minimal Price (INR)                      4968364.8910
dtype: float64
```

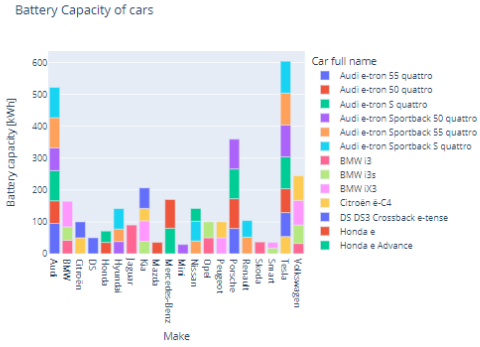
On an average present electric vehicles provides these features, thus we can provide approximately in comparison to these electric vehicles in our cars.

```
[ ] fig = px.scatter(data,x='Make',y = 'Minimal Price (INR)',color = 'Car full name',title = 'Price of cars',labels = {'x':'Prices','y':'Cars'})
    pio.show(fig)
```



Car brands and their prices relationship shows that Porsche is the most expensive vehicle and we can have the price range of our electric vehicles around 50 lakhs.

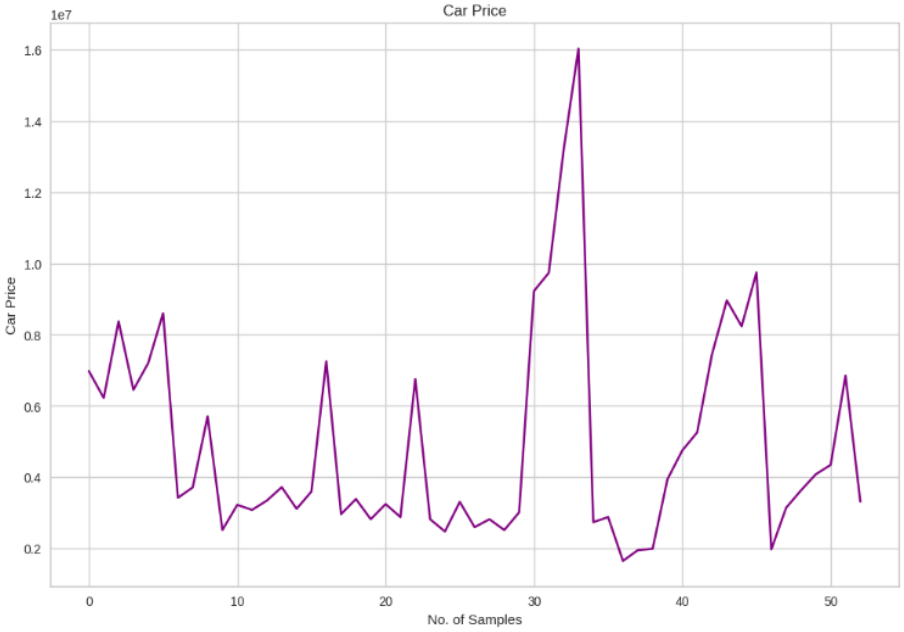
```
[ ] fig = px.bar(data,x='Make',y = 'Battery capacity [kWh]',color = 'Car full name',title = 'Battery Capacity of cars',labels = {'x':'Car Brands','y':'Battery Capacity'})
    pio.show(fig)
```



This shows the battery capacity of of car brands, Tesla has the most battery capacity, thus we can prepare cars having battery capacity competing that of AUDI.

```
[ ] data['Minimal Price (INR)'].plot(figsize = (12,8),title='Car Price',xlabel = 'No. of Samples',ylabel = 'Car Price',color = 'purple')
```

```
<Axes: title={'center': 'Car Price'}, xlabel='No. of Samples', ylabel='Car Price'>
```



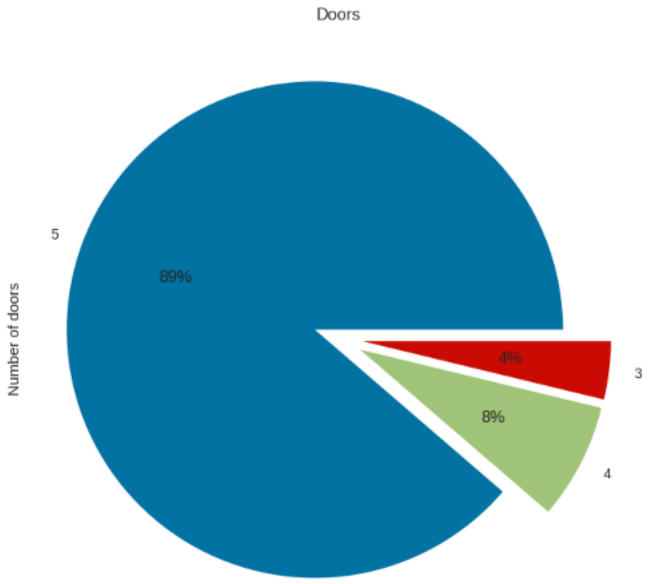
https://github.com/Khhushhiiii/Electric-Vehicles-Feature-Analysis/blob/main/data%20(2).csv

```
[ ] data['Number of doors'].unique()
```

```
array([5, 3, 4])
```

```
[ ] data['Number of doors'].value_counts().plot.pie(figsize=(8,15),autopct='%.0f%%',explode=(0.1,0.1,0.1))
    plt.title('Doors')
```

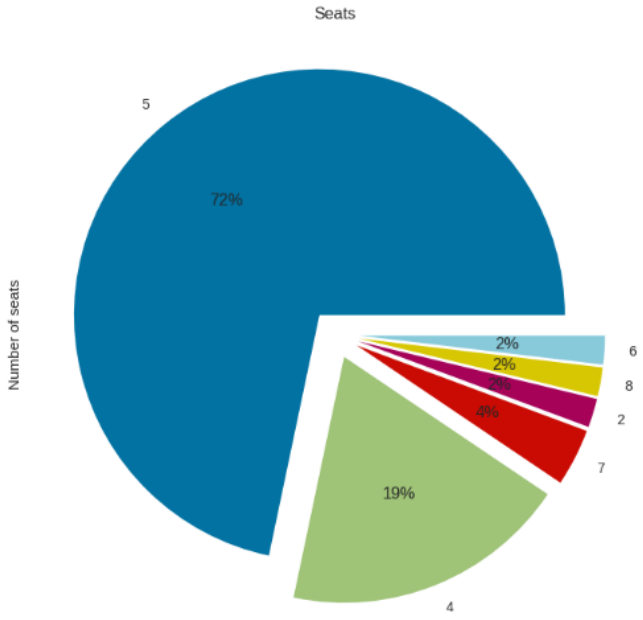
```
Text(0.5, 1.0, 'Doors')
```



Maximum cars have 5 doors

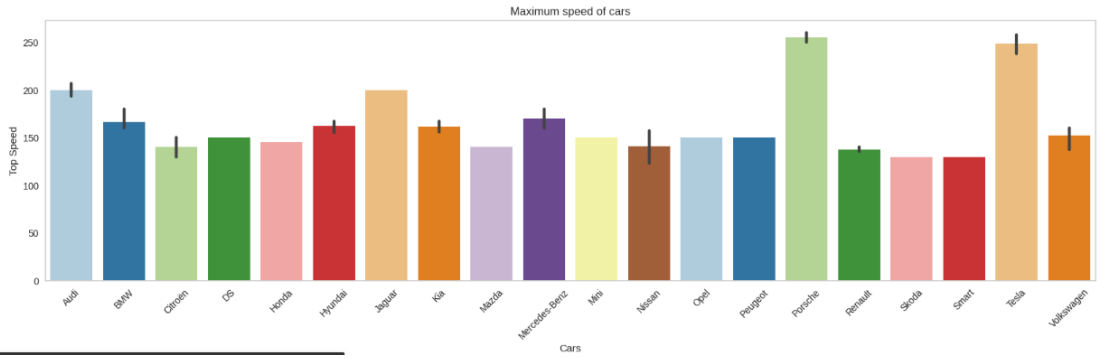
```
[ ] data['Number of seats'].unique()

    array([5, 4, 2, 7, 8, 6])
```

```
[ ] data['Number of seats'].value_counts().plot.pie(figsize=(8,15),autopct='%.0f%%',explode=(0.1,0.1,0.1,0.1,0.1,0.1))
    plt.title('Seats')

    Text(0.5, 1.0, 'Seats')
```



Maximum cars are 5 seaters

```
[ ] ax= plt.figure(figsize=(20,5))
    sns.barplot(x='Make',y='Maximum speed [kph]',data=data,palette='Paired')
    plt.grid(axis='y')
    plt.title('Maximum speed of cars')
    plt.xlabel('Cars')
    plt.ylabel('Top Speed')
    plt.xticks(rotation=45)

    (array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
            17, 18, 19]),
     [Text(0, 0, 'Audi'),
      Text(1, 0, 'BMW'),
      Text(2, 0, 'Citroën'),
      Text(3, 0, 'DS'),
      Text(4, 0, 'Honda'),
      Text(5, 0, 'Hyundai'),
      Text(6, 0, 'Jaguar'),
      Text(7, 0, 'Kia'),
      Text(8, 0, 'Mazda'),
      Text(9, 0, 'Mercedes-Benz'),
      Text(10, 0, 'Mini'),
      Text(11, 0, 'Nissan'),
      Text(12, 0, 'Opel'),
      Text(13, 0, 'Peugeot'),
      Text(14, 0, 'Porsche'),
      Text(15, 0, 'Renault'),
      Text(16, 0, 'Skoda'),
      Text(17, 0, 'Smart'),
      Text(18, 0, 'Tesla'),
      Text(19, 0, 'Volkswagen')])
```



MODEL FITTING AND SUMMARY:

```
[53] model=results.fit()
     model.summary()
```

OLS Regression Results

Dep. Variable:	Minimal Price (INR)	**R-squared:**	0.928
Model:	OLS	**Adj. R-squared:**	0.919
Method:	Least Squares	**F-statistic:**	99.12
Date:	Tue, 04 Jul 2023	**Prob (F-statistic):**	1.27e-24
Time:	13:29:26	**Log-Likelihood:**	-795.54
No. Observations:	53	**AIC:**	1605.
Df Residuals:	46	**BIC:**	1619.
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-4.283e+04	1.67e+06	-0.026	0.980	-3.41e+06	3.32e+06
Engine power [KM]	1.254e+04	2311.425	5.427	0.000	7891.706	1.72e+04
Battery capacity [kWh]	-1.191e+04	1.07e+04	-1.116	0.270	-3.34e+04	9565.168
Maximum speed [kph]	-3892.8287	1.2e+04	-0.323	0.748	-2.81e+04	2.03e+04
Acceleration 0-100 kph [s]	-6.309e+04	5.2e+04	-1.212	0.232	-1.68e+05	4.17e+04
mean - Energy consumption [kWh/100 km]	9.418e+04	1.98e+04	4.763	0.000	5.44e+04	1.34e+05
Maximum DC charging power [kW]	1.776e+04	4974.947	3.571	0.001	7750.734	2.78e+04

Omnibus:	26.288	**Durbin-Watson:**	1.735
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	53.564
Skew:	1.478	**Prob(JB):**	2.34e-12
Kurtosis:	6.939	**Cond. No.**	5.57e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.57e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

ACCURACY:

```
r2=(r2_score(y_test,pred))
print(r2*100)
```

```
83.99384018077691
```

DATA SOURCE 2:

https://github.com/Khhushhiiii/Electric-Vehicles-Feature-Analysis/blob/main/data%20(2).csv

DATA PREPROCESSING:

Step 1: Reading the dataset

```
[ ] #Reading the dataset
    df = pd.read_csv('/content/data (2).csv')
```

```
df.head(10)
```

	Unnamed: 0	Brand	Model	AccelSec	TopSpeed_KmH	Range_Km	Efficiency_WhKm	FastCharge_KmH	RapidCharge	PowerTrain	PlugType	BodyStyle	Segment	Seats	PriceEuro
0	0	Tesla	Model 3 Long Range Dual Motor	4.6000	233	450	161	940	Yes	AWD	Type 2 CCS	Sedan	D	5	55480
1	1	Volkswagen	ID.3 Pure	10.0000	160	270	167	250	No	RWD	Type 2 CCS	Hatchback	C	5	30000
2	2	Polestar	2	4.7000	210	400	181	620	Yes	AWD	Type 2 CCS	Liftback	D	5	56440
3	3	BMW	iX3	6.8000	180	360	206	560	Yes	RWD	Type 2 CCS	SUV	D	5	68040
4	4	Honda	e	9.5000	145	170	168	190	Yes	RWD	Type 2 CCS	Hatchback	B	4	32997
5	5	Lucid	Air	2.8000	250	610	180	620	Yes	AWD	Type 2 CCS	Sedan	F	5	105000
6	6	Volkswagen	e-Golf	9.6000	150	190	168	220	No	FWD	Type 2 CCS	Hatchback	C	5	31900
7	7	Peugeot	e-208	8.1000	150	275	164	420	No	FWD	Type 2 CCS	Hatchback	B	5	29682
8	8	Tesla	Model 3 Standard Range Plus	5.6000	225	310	153	650	Yes	RWD	Type 2 CCS	Sedan	D	5	46380
9	9	Audi	Q4 e-tron	6.3000	180	400	193	540	Yes	AWD	Type 2 CCS	SUV	D	5	55000

Step 2: Dropping the unnecessary columns

```
[ ] df.drop('Unnamed: 0', axis =1 , inplace=True)
```

```
[ ] df
```

	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	4.6000	233	450	161	940	Yes	AWD	Type 2 CCS	Sedan	D	5	55480
1	Volkswagen	ID.3 Pure	10.0000	160	270	167	250	No	RWD	Type 2 CCS	Hatchback	C	5	30000
2	Polestar	2	4.7000	210	400	181	620	Yes	AWD	Type 2 CCS	Liftback	D	5	56440
3	BMW	iX3	6.8000	180	360	206	560	Yes	RWD	Type 2 CCS	SUV	D	5	68040
4	Honda	e	9.5000	145	170	168	190	Yes	RWD	Type 2 CCS	Hatchback	B	4	32997
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
98	Nissan	Ariya 63kWh	7.5000	160	330	191	440	Yes	FWD	Type 2 CCS	Hatchback	C	5	45000
99	Audi	e-tron S Sportback 55 quattro	4.5000	210	335	258	540	Yes	AWD	Type 2 CCS	SUV	E	5	96050
100	Nissan	Ariya e-4ORCE 63kWh	5.9000	200	325	194	440	Yes	AWD	Type 2 CCS	Hatchback	C	5	50000
101	Nissan	Ariya e-4ORCE 87kWh Performance	5.1000	200	375	232	450	Yes	AWD	Type 2 CCS	Hatchback	C	5	65000
102	Byton	M-Byte 95 kWh 2WD	7.5000	190	400	238	480	No	AWD	Type 2 CCS	SUV	E	5	62000

103 rows × 14 columns

Step 3: Encoding the categorical variables using LabelEncoder and c hanging the values according to Indian market

```
[ ] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 103 entries, 0 to 102
Data columns (total 15 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    int64
 7   RapidCharge      103 non-null    int64
 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
 14  INR              103 non-null    float64
dtypes: float64(2), int64(7), object(6)
memory usage: 12.2+ KB
```

```
[ ] from sklearn.preprocessing import LabelEncoder
    l1 = LabelEncoder()
    df['RapidCharge'] = l1.fit_transform(df['RapidCharge'])
```

```
[ ] df['INR'] = df['PriceEuro']*89.88
```

```
[ ] df
```

	Brand	Model	AccelSec	TopSpeed_KmH	Range_Km	Efficiency_WhKm	FastCharge_KmH	RapidCharge	PowerTrain	PlugType	BodyStyle	Segment	Seats	PriceEuro	INR
0	Tesla	Model 3 Long Range Dual Motor	4.6000	233	450	161	940	1	AWD	Type 2 CCS	Sedan	D	5	55480	4986542.4000
1	Volkswagen	ID.3 Pure	10.0000	160	270	167	250	0	RWD	Type 2 CCS	Hatchback	C	5	30000	2696400.0000
2	Polestar	2	4.7000	210	400	181	620	1	AWD	Type 2 CCS	Liftback	D	5	56440	5072827.2000
3	BMW	iX3	6.8000	180	360	206	560	1	RWD	Type 2 CCS	SUV	D	5	68040	6115435.2000
4	Honda	e	9.5000	145	170	168	190	1	RWD	Type 2 CCS	Hatchback	B	4	32997	2965770.3600
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
98	Nissan	Ariya 63kWh	7.5000	160	330	191	440	1	FWD	Type 2 CCS	Hatchback	C	5	45000	4044600.0000
99	Audi	e-tron S Sportback 55 quattro	4.5000	210	335	258	540	1	AWD	Type 2 CCS	SUV	E	5	96050	8632974.0000
100	Nissan	Ariya e-4ORCE 63kWh	5.9000	200	325	194	440	1	AWD	Type 2 CCS	Hatchback	C	5	50000	4494000.0000
101	Nissan	Ariya e-4ORCE 87kWh Performance	5.1000	200	375	232	450	1	AWD	Type 2 CCS	Hatchback	C	5	65000	5842200.0000
102	Byton	M-Byte 95 kWh 2WD	7.5000	190	400	238	480	0	AWD	Type 2 CCS	SUV	E	5	62000	5572560.0000

103 rows × 15 columns

Step 4: Checking the null values

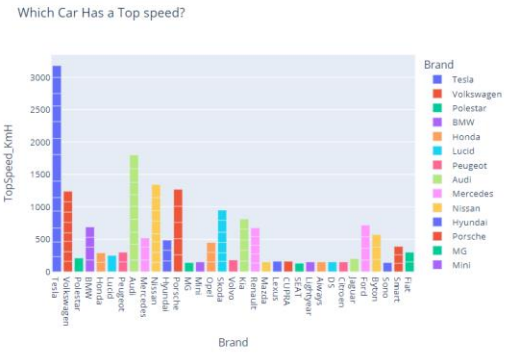
```
[ ] df.isnull().sum()
```

```
Brand              0
Model              0
AccelSec           0
TopSpeed_KmH       0
Range_Km           0
Efficiency_WhKm    0
FastCharge_KmH     0
RapidCharge        0
PowerTrain         0
PlugType           0
BodyStyle          0
Segment            0
Seats              0
PriceEuro          0
INR                0
dtype: int64
```

Some of the brands which are currently present in the Electric Vehicle market

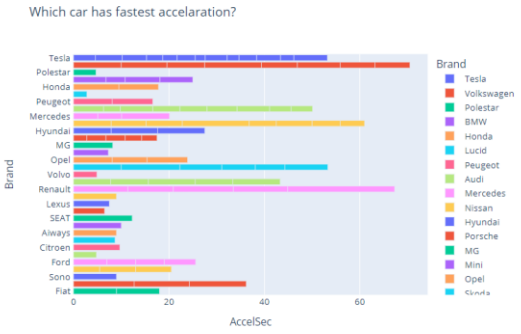
Comparing the speeds of the cars

```
fig = px.bar(df,x='Brand',y = 'TopSpeed_KmH',color = 'Brand',title = 'Which Car Has a Top speed?',labels = {'x':'Car Brands','y':'Top Speed Km/H'})
pio.show(fig)
```



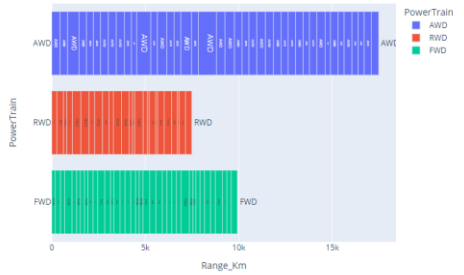
Comparing the acceleration

```
[ ] fig = px.bar(df,x='AccelSec',y = 'Brand',color = 'Brand',title = 'Which car has fastest accelaration?',labels = {'x':'Accelaration','y':'Car Brands'})
    pio.show(fig)
```



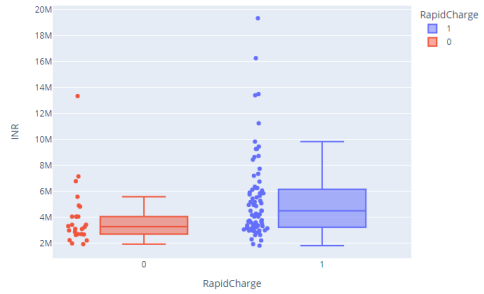
Range km vs PowerTrain

```
fig = px.bar(df,x = 'Range_Km',y = 'PowerTrain',color = 'PowerTrain',text='PowerTrain')
pio.show(fig)
```

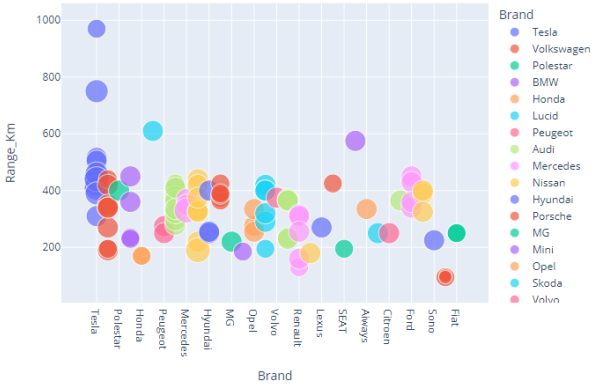


Rapid Charging is there or not

```
[ ] fig = px.box(df,x='RapidCharge',y = 'INR',color = 'RapidCharge',points='all')
    pio.show(fig)
```



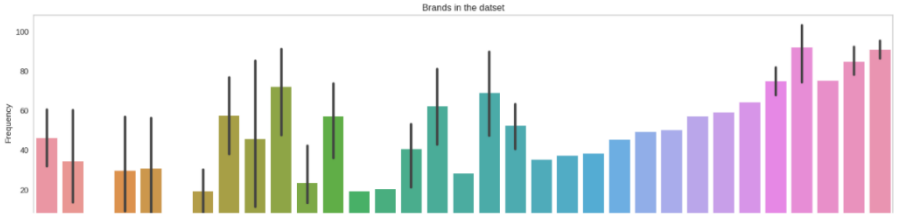

```
[ ] fig = px.scatter(df,x = 'Brand',y = 'Range_Km',size='Seats',color = 'Brand',hover_data=['RapidCharge','INR'])
    pio.show(fig)
```



Prominent brands

```
[ ]  ax= plt.figure(figsize=(20,5))
     sns.barplot(x='Brand',y=a,data=df)
     plt.grid(axis='y')
     plt.title('Brands in the datset')
     plt.xlabel('Brand')
     plt.ylabel('Frequency')
     plt.xticks(rotation=45)
```

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32]),
 [Text(0, 0, 'Tesla '),
  Text(1, 0, 'Volkswagen '),
  Text(2, 0, 'Polestar '),
  Text(3, 0, 'BMW '),
  Text(4, 0, 'Honda '),
  Text(5, 0, 'Lucid '),
  Text(6, 0, 'Peugeot '),
  Text(7, 0, 'Audi '),
  Text(8, 0, 'Mercedes '),
  Text(9, 0, 'Nissan '),
  Text(10, 0, 'Hyundai '),
  Text(11, 0, 'Porsche '),
  Text(12, 0, 'MG '),
  Text(13, 0, 'Mini '),
  Text(14, 0, 'Opel '),
  Text(15, 0, 'Skoda '),
  Text(16, 0, 'Volvo '),
  Text(17, 0, 'Kia '),
  Text(18, 0, 'Renault '),
  Text(19, 0, 'Mazda '),
  Text(20, 0, 'Lexus '),
  Text(21, 0, 'CUPRA '),
  Text(22, 0, 'SEAT '),
  Text(23, 0, 'Lightyear '),
  Text(24, 0, 'Always '),
  Text(25, 0, 'DS '),
  Text(26, 0, 'Citroen '),
  Text(27, 0, 'Jaguar '),
  Text(28, 0, 'Ford '),
  Text(29, 0, 'Byton '),
  Text(30, 0, 'Sono '),
  Text(31, 0, 'Smart '),
  Text(32, 0, 'Fiat ')])
```

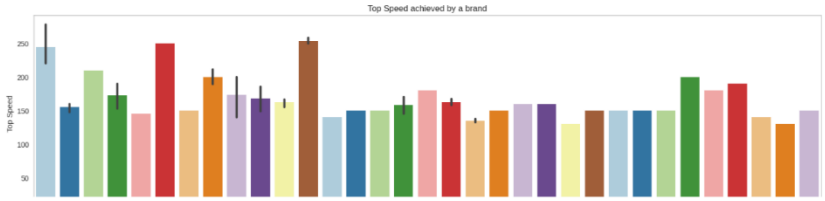


Byton , Fiat and smart are the prominent brands and Polestar being the least

Top speeds achieved by cars

```
ax= plt.figure(figsize=(20,5))
sns.barplot(x='Brand',y='TopSpeed_KmH',data=df,palette='Paired')
plt.grid(axis='y')
plt.title('Top Speed achieved by a brand')
plt.xlabel('Brand')
plt.ylabel('Top Speed')
plt.xticks(rotation=45)
```

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32]),
 [Text(0, 0, 'Tesla '),
  Text(1, 0, 'Volkswagen '),
  Text(2, 0, 'Polestar '),
  Text(3, 0, 'BMW '),
  Text(4, 0, 'Honda '),
  Text(5, 0, 'Lucid '),
  Text(6, 0, 'Peugeot '),
  Text(7, 0, 'Audi '),
  Text(8, 0, 'Mercedes '),
  Text(9, 0, 'Nissan '),
  Text(10, 0, 'Hyundai '),
  Text(11, 0, 'Porsche '),
  Text(12, 0, 'MG '),
  Text(13, 0, 'Mini '),
  Text(14, 0, 'Opel '),
  Text(15, 0, 'Skoda '),
  Text(16, 0, 'Volvo '),
  Text(17, 0, 'Kia '),
  Text(18, 0, 'Renault '),
  Text(19, 0, 'Mazda '),
  Text(20, 0, 'Lexus '),
  Text(21, 0, 'CUPRA '),
  Text(22, 0, 'SEAT '),
  Text(23, 0, 'Lightyear '),
  Text(24, 0, 'Always '),
  Text(25, 0, 'DS '),
  Text(26, 0, 'Citroen '),
  Text(27, 0, 'Jaguar '),
  Text(28, 0, 'Ford '),
  Text(29, 0, 'Byton '),
  Text(30, 0, 'Sono '),
  Text(31, 0, 'Smart '),
  Text(32, 0, 'Fiat ')])
```

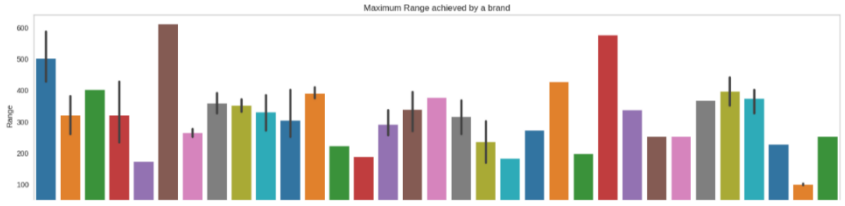


Porsche, Lucid and Tesla produce the fastest cars and Smart the lowest

Range achieved by cars

```
[ ] ax= plt.figure(figsize=(20,5))
    sns.barplot(x='Brand',y='Range_Km',data=df,palette='tab10')
    plt.grid(axis='y')
    plt.title('Maximum Range achieved by a brand')
    plt.xlabel('Brand')
    plt.ylabel('Range')
    plt.xticks(rotation=45)
```

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32]),
 [Text(0, 0, 'Tesla '),
  Text(1, 0, 'Volkswagen '),
  Text(2, 0, 'Polestar '),
  Text(3, 0, 'BMW '),
  Text(4, 0, 'Honda '),
  Text(5, 0, 'Lucid '),
  Text(6, 0, 'Peugeot '),
  Text(7, 0, 'Audi '),
  Text(8, 0, 'Mercedes '),
  Text(9, 0, 'Nissan '),
  Text(10, 0, 'Hyundai '),
  Text(11, 0, 'Porsche '),
  Text(12, 0, 'MG '),
  Text(13, 0, 'Mini '),
  Text(14, 0, 'Opel '),
  Text(15, 0, 'Skoda '),
  Text(16, 0, 'Volvo '),
  Text(17, 0, 'Kia '),
  Text(18, 0, 'Renault '),
  Text(19, 0, 'Mazda '),
  Text(20, 0, 'Lexus '),
  Text(21, 0, 'CUPRA '),
  Text(22, 0, 'SEAT '),
  Text(23, 0, 'Lightyear '),
  Text(24, 0, 'Always '),
  Text(25, 0, 'DS '),
  Text(26, 0, 'Citroen '),
  Text(27, 0, 'Jaguar '),
  Text(28, 0, 'Ford '),
  Text(29, 0, 'Byton '),
  Text(30, 0, 'Sono '),
  Text(31, 0, 'Smart '),
  Text(32, 0, 'Fiat ')])
```

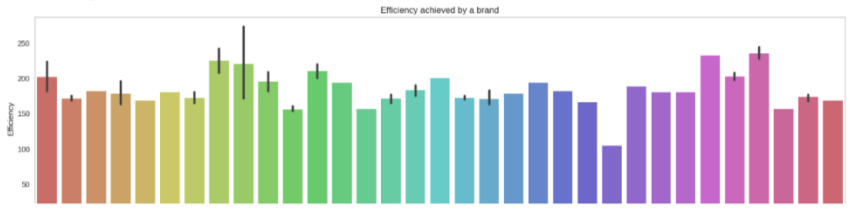


Lucid, Lightyear, and Tesla have the highest range and Smart has the lowest range

Efficiency of cars

```
[ ] ax= plt.figure(figsize=(20,5))
    sns.barplot(x='Brand',y='Efficiency_whKm',data=df,palette='hls')
    plt.grid(axis='y')
    plt.title('Efficiency achieved by a brand')
    plt.xlabel('Brand')
    plt.ylabel('Efficiency')
    plt.xticks(rotation=45)
```

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32]),
 [Text(0, 0, 'Tesla '),
  Text(1, 0, 'Volkswagen '),
  Text(2, 0, 'Polestar '),
  Text(3, 0, 'BMW '),
  Text(4, 0, 'Honda '),
  Text(5, 0, 'Lucid '),
  Text(6, 0, 'Peugeot '),
  Text(7, 0, 'Audi '),
  Text(8, 0, 'Mercedes '),
  Text(9, 0, 'Nissan '),
  Text(10, 0, 'Hyundai '),
  Text(11, 0, 'Porsche '),
  Text(12, 0, 'MG '),
  Text(13, 0, 'Mini '),
  Text(14, 0, 'Opel '),
  Text(15, 0, 'Skoda '),
  Text(16, 0, 'Volvo '),
  Text(17, 0, 'Kia '),
  Text(18, 0, 'Renault '),
  Text(19, 0, 'Mazda '),
  Text(20, 0, 'Lexus '),
  Text(21, 0, 'CUPRA '),
  Text(22, 0, 'SEAT '),
  Text(23, 0, 'Lightyear '),
  Text(24, 0, 'Always '),
  Text(25, 0, 'DS '),
  Text(26, 0, 'Citroen '),
  Text(27, 0, 'Jaguar '),
  Text(28, 0, 'Ford '),
  Text(29, 0, 'Byton '),
  Text(30, 0, 'Sono '),
  Text(31, 0, 'Smart '),
  Text(32, 0, 'Fiat ')])
```

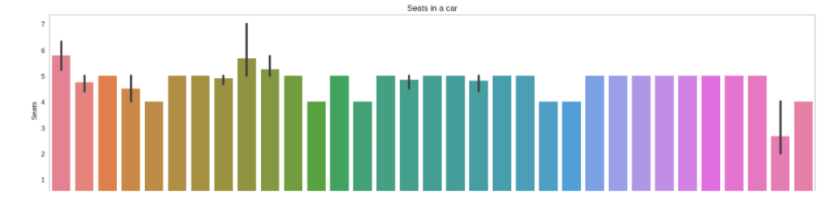


Byton, Jaguar and Audi are the most efficient and Lightyear has the least

Number of seats in each car

```
ax= plt.figure(figsize=(20,5))
sns.barplot(x='Brand',y='Seats',data=df,palette='husl')
plt.grid(axis='y')
plt.title('Seats in a car')
plt.xlabel('Brand')
plt.ylabel('Seats')
plt.xticks(rotation=45)
```

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32]),
 [Text(0, 0, 'Tesla '),
  Text(1, 0, 'Volkswagen '),
  Text(2, 0, 'Polestar '),
  Text(3, 0, 'BMW '),
  Text(4, 0, 'Honda '),
  Text(5, 0, 'Lucid '),
  Text(6, 0, 'Peugeot '),
  Text(7, 0, 'Audi '),
  Text(8, 0, 'Mercedes '),
  Text(9, 0, 'Nissan '),
  Text(10, 0, 'Hyundai '),
  Text(11, 0, 'Porsche '),
  Text(12, 0, 'MG '),
  Text(13, 0, 'Mini '),
  Text(14, 0, 'Opel '),
  Text(15, 0, 'Skoda '),
  Text(16, 0, 'Volvo '),
  Text(17, 0, 'Kia '),
  Text(18, 0, 'Renault '),
  Text(19, 0, 'Mazda '),
  Text(20, 0, 'Lexus '),
  Text(21, 0, 'CUPRA '),
  Text(22, 0, 'SEAT '),
  Text(23, 0, 'Lightyear '),
  Text(24, 0, 'Always '),
  Text(25, 0, 'DS '),
  Text(26, 0, 'Citroen '),
  Text(27, 0, 'Jaguar '),
  Text(28, 0, 'Ford '),
  Text(29, 0, 'Byton '),
  Text(30, 0, 'Sono '),
  Text(31, 0, 'Smart '),
  Text(32, 0, 'Fiat ')])
```

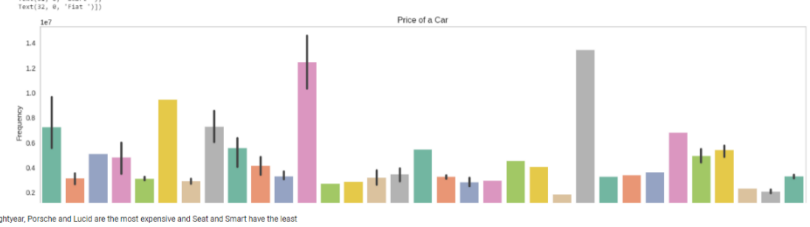


Mercedes, Tesla and Nissan have the highest number of seats and Smart has the lowest

Price of Cars (INR)

```
ax= plt.figure(figsize=(20,5))
sns.barplot(x='Brand',y='INR',data=df,palette='Set2')
plt.title('Price of a Car')
plt.xlabel('Price in INR')
plt.grid(axis='y')
plt.ylabel('Frequency')
plt.xticks(rotation=45)
```

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32]),
 [Text(0, 0, 'Tesla '),
  Text(1, 0, 'Volkswagen '),
  Text(2, 0, 'Polestar '),
  Text(3, 0, 'BMW '),
  Text(4, 0, 'Honda '),
  Text(5, 0, 'Lucid '),
  Text(6, 0, 'Peugeot '),
  Text(7, 0, 'Audi '),
  Text(8, 0, 'Mercedes '),
  Text(9, 0, 'Nissan '),
  Text(10, 0, 'Hyundai '),
  Text(11, 0, 'Porsche '),
  Text(12, 0, 'MG '),
  Text(13, 0, 'Mini '),
  Text(14, 0, 'Opel '),
  Text(15, 0, 'Skoda '),
  Text(16, 0, 'Volvo '),
  Text(17, 0, 'Kia '),
  Text(18, 0, 'Renault '),
  Text(19, 0, 'Mazda '),
  Text(20, 0, 'Lexus '),
  Text(21, 0, 'CUPRA '),
  Text(22, 0, 'SEAT '),
  Text(23, 0, 'Lightyear '),
  Text(24, 0, 'Always '),
  Text(25, 0, 'DS '),
  Text(26, 0, 'Citroen '),
  Text(27, 0, 'Jaguar '),
  Text(28, 0, 'Ford '),
  Text(29, 0, 'Byton '),
  Text(30, 0, 'Sono '),
  Text(31, 0, 'Smart '),
  Text(32, 0, 'Fiat ')])
```

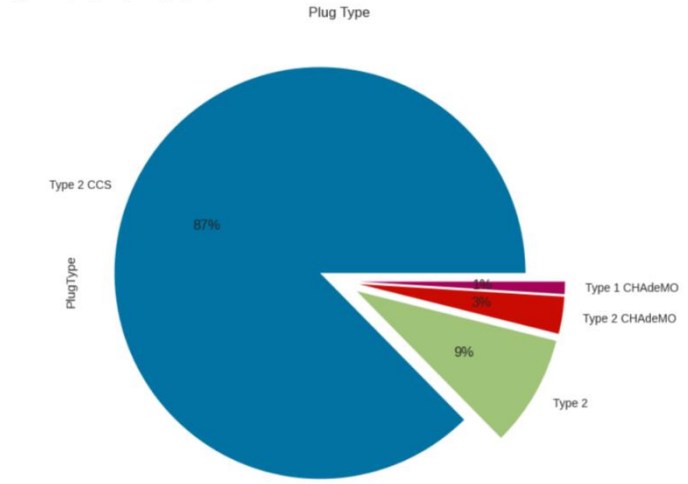


Lightyear, Porsche and Lucid are the most expensive and Seat and Smart have the least

Plug Types used by cars

```
df['PlugType'].value_counts().plot.pie(figsize=(8,15),autopct='%.0f%%',explode=(.1,.1,.1,.1))
plt.title('Plug Type')
```

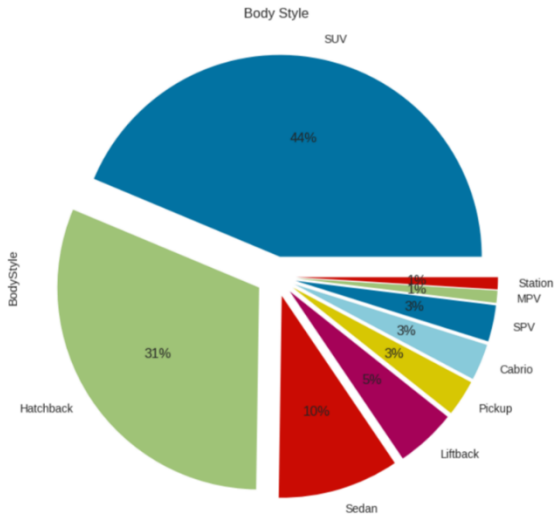
```
Text(0.5, 1.0, 'Plug Type')
```



Cars and their body style

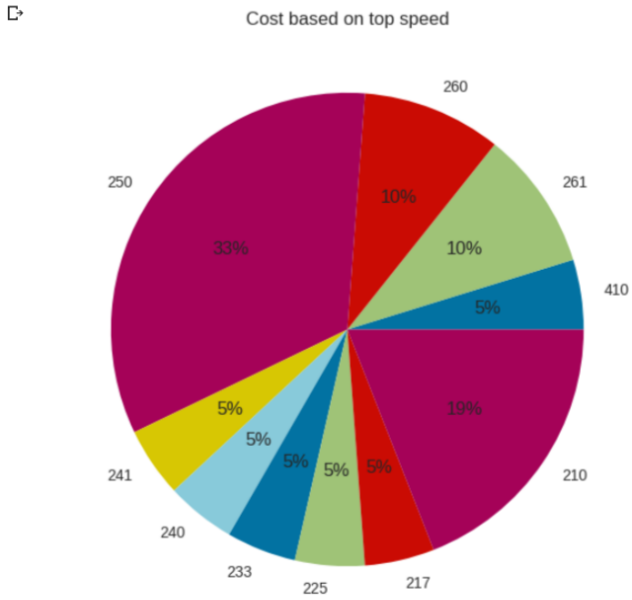
```
df['BodyStyle'].value_counts().plot.pie(figsize=(8,15),autopct='%.0f%%',explode=(0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1))
plt.title('Body Style')
```

```
Text(0.5, 1.0, 'Body Style')
```



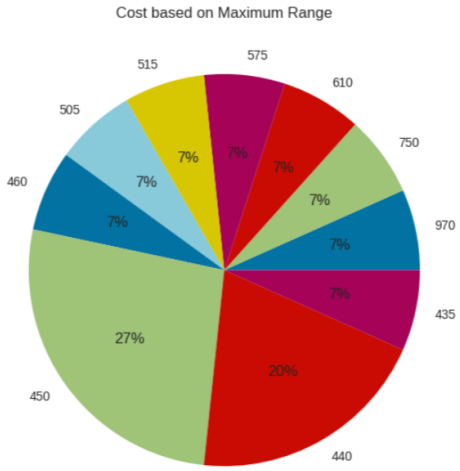
Cost based on top model

```
plt.figure(figsize=(10,7))
plt.title('Cost based on top speed')
plt.pie(x=df3["INR"],labels=df3.index,autopct='%1.0f%%')
plt.show()
```



Cost based on maximum range

```
plt.figure(figsize=(10,7))
plt.title('Cost based on Maximum Range')
plt.pie(x=df4["INR"],labels=df4.index,autopct='%1.0f%%')
plt.show()
```



CONCLUSION: In conclusion we can say that Electric vehicles (EVs) are equipped with various features that distinguish them from traditional internal combustion engine vehicles. These features are centered around promoting energy efficiency, reducing emissions, and enhancing user experience. Here are some common features found in electric vehicles:

1. Battery and Electric Drive System: The most crucial component of an EV is its high-capacity battery pack. This battery stores electricity to power the vehicle's electric drive system, which includes an electric motor and power electronics. The electric drive system efficiently converts electrical energy into mechanical power to propel the vehicle.

2. Regenerative Braking: EVs often incorporate regenerative braking, a feature that recaptures energy during deceleration and braking. When the driver applies the brakes, the electric motor acts as a generator, converting kinetic energy back into electrical energy and storing it in the battery, thus increasing overall efficiency and range.

3. Electric Range and Charging: The electric range represents the distance an EV can travel on a single charge. Different EV models offer varying ranges, with improvements in battery technology leading to increased ranges over time. Charging infrastructure and capabilities are also essential considerations, including Level 1 (standard household outlet), Level 2 (AC charging stations), and Level 3 (DC fast charging) options.

4. Connectivity and Telematics: Many modern EVs come with advanced connectivity features, such as smartphone integration, infotainment systems, and vehicle telematics. These features allow drivers to monitor battery status, range, charging station locations, and other essential data remotely.

5. Eco-Driving and Efficiency Metrics: To encourage energy-efficient driving habits, some EVs provide eco-driving assistance, displaying real-time efficiency metrics, and offering driving suggestions to optimize range and energy consumption.

6. Cabin Comfort: EVs often prioritize cabin comfort with features like climate control, heated seats, and other amenities. Some vehicles use preconditioning technology to heat or cool the cabin while still connected to the charging station, preserving battery range during driving.

These features represent the ongoing innovations and developments in the electric

GitHub Link:

https://github.com/Khhushhiiii/Electric-Vehicles-Feature-Analysis

ANALYSIS DONE BY AISHWARYA ANN JOSEPH

INTRODUCTION

The purpose of the code is to analyse and gain insights from various datasets related to electric vehicles (EVs) and charging stations in India. It aims to provide valuable information about EV user ratings, charging station data, purchase behaviour, and EV manufacturers' popularity.

In the context of EVs and charging stations in India, this code is relevant for several reasons:

1. **Understanding EV User Ratings:**

   The code allows us to analyse EV user ratings, which can provide insights into the satisfaction level of EV users in India. By examining the distribution of ratings and identifying popular EV models, stakeholders in the EV industry can gauge consumer preferences and make informed decisions about product development and marketing strategies.

2. **Analysing Charging Station Data:**

   The code also analyses charging station data, which is crucial for understanding the infrastructure landscape for EVs in India. By visualizing the distribution of charging stations across different states, policymakers, electric utility companies, and other stakeholders can identify regions with a higher demand for charging infrastructure and plan accordingly.

3. **Vehicle Buying Behaviour:**

   The code explores purchase behaviour data, considering factors such as marital status. By examining how marital status influences vehicle buying behaviour, it provides insights into the target market for EVs in India. This information is valuable for EV manufacturers and marketers, helping them tailor their strategies to different customer segments.

4. **EV Manufacturer Analysis:**

   The code investigates the popularity of different EV models based on user ratings. This analysis can guide EV manufacturers in understanding consumer preferences, identifying successful models, and improving upon less popular ones. It also provides valuable insights into the competitive landscape of EVs in India.

Overall, this code is relevant in the context of EVs and charging stations in India as it helps stakeholders in the EV industry gain a deeper understanding of user preferences, infrastructure requirements, and market dynamics. By leveraging these insights, they can make informed decisions to promote the adoption and growth of EVs in the country.

The code utilizes three main datasets: EV user ratings, charging station data, and purchase behaviour data. Here is an overview of each dataset:

1. **EV User Ratings:**
   - Description: This dataset contains information about user ratings and reviews of electric vehicles in India. It includes attributes such as the EV model, user rating, and user review.
   - Purpose: The EV user ratings dataset is used to analyse user satisfaction and preferences regarding different EV models. By examining the distribution of ratings and reviews, the code provides insights into the popularity and performance of EVs in India.
2. **Charging Station Data:**

- Description: This dataset comprises data related to charging stations for electric vehicles in India. It includes attributes such as the charging station location, state, latitude, longitude, and the number of charging points.
- Purpose: The charging station data is utilized to analyse the distribution and availability of charging infrastructure across different regions in India. By visualizing the data, stakeholders can identify areas with a higher concentration of charging stations and areas that require further development.

3. **Purchase Behaviour Data:**
    - Description: This dataset contains information about the purchase behaviour of electric vehicle buyers in India. It includes attributes such as marital status, vehicle type, and purchase intent.
    - Purpose: The purchase behaviour data helps analyse the factors influencing the buying decisions of individuals interested in purchasing electric vehicles. By examining the relationship between marital status and vehicle type preference, the code provides insights into the target market for EVs in India.

These datasets are crucial for understanding various aspects of the EV ecosystem in India. The EV user ratings dataset enables the evaluation of user satisfaction and the identification of popular EV models. The charging station data helps assess the availability and accessibility of charging infrastructure. Lastly, the purchase behaviour data sheds light on the preferences and buying behaviour of potential EV buyers in India. By leveraging these datasets, stakeholders can make data-driven decisions to drive the adoption and growth of electric vehicles in the country.

DATA PROCESSING AND ANALYSIS

The code imports several libraries that play important roles in data manipulation, analysis, and visualization. Here is an overview of the libraries imported and their roles:

1. **Pandas:**
    - Role: Pandas is a powerful library for data manipulation and analysis. It provides data structures and functions to efficiently handle structured data.
    - Import statement: **import pandas as pd**
2. **NumPy:**
    - Role: NumPy is a fundamental library for numerical computing in Python. It provides support for large, multi-dimensional arrays and a collection of mathematical functions.
    - Import statement: **import numpy as np**
3. **Matplotlib:**
    - Role: Matplotlib is a widely-used plotting library in Python. It provides a flexible and comprehensive set of functions for creating static, animated, and interactive visualizations.
    - Import statement: **import matplotlib.pyplot as plt**
4. **Seaborn:**
    - Role: Seaborn is a statistical data visualization library built on top of Matplotlib. It provides a high-level interface for creating informative and attractive statistical graphics.
    - Import statement: **import seaborn as sns**

5. **Plotly:**
    - Role: Plotly is a library that enables the creation of interactive and visually appealing plots. It provides a range of chart types and customization options.
    - Import statement: **import plotly.express as px**

Now, let us proceed to load the EV user rating data and provide an overview:

```
# Load EV user rating data
ev_ratings = pd.read_csv('ev_user_ratings.csv')
# Overview of data structure and columns
print(ev_ratings.head())
print(ev_ratings.info())
# Summary statistics
print(ev_ratings.describe())
```

The code above loads the EV user rating data from a CSV file named 'ev_user_ratings.csv' and assigns it to the variable **ev_ratings**. To provide an overview, the code prints the first few rows of the dataset using **head()** and displays information about the data structure and columns using **info()**. Additionally, it calculates and displays summary statistics using **describe()**.

To visualize the user ratings and understand their distribution and patterns, count plots and strip plots can be used:

```
# Count plot of user ratings
sns.countplot(data=ev_ratings, x='User Rating')
plt.title('Distribution of User Ratings')
plt.show()
# Strip plot of user ratings
sns.stripplot(data=ev_ratings, x='User Rating', y='EV Model')
plt.title('User Ratings by EV Model')
plt.show()
```

The code above creates a count plot of user ratings using **sns.countplot()**, which displays the frequency of each rating value. It also generates a strip plot using **sns.stripplot()**, which shows the distribution of user ratings for different EV models. The x-axis represents the user rating, and the y-axis represents the EV model. These plots provide insights into the distribution, concentration, and patterns of user ratings.

To analyze the types of EVs (2-wheelers and 4-wheelers) based on user ratings and present the findings using a bar chart:

```
# Create a bar chart of EV types based on user ratings
```

ev_types = ['2-Wheeler', '4-Wheeler']

ev_type_ratings=ev_ratings.groupby('EVType')['UserRating'].mean().reindex(ev_types)

ev_type_ratings.plot(kind='bar')

plt.title('Average User Ratings by EV Type')

plt.ylabel('Average User Rating')

plt.xlabel('EV Type')

plt.show()

The code above groups the EV user ratings by EV type ('2-Wheeler' and '4-Wheeler') and calculates the mean user rating for each type using **groupby()** and **mean()**. It then creates a bar chart using **plot(kind='bar')** to visualize the average user ratings for each EV type. The x-axis represents the EV type, the y-axis represents the average user rating, and the bars depict the ratings for different EV types.

This analysis helps understand the user ratings, their distribution, and patterns, as well as provides insights into the average user ratings for different types of EVs.

CHARGING STATION DATA ANALYSIS

To analyze the charging station data, we'll start by loading the data and providing an overview of its structure, columns, and summary statistics. Then we'll visualize the distribution of charging stations across different states using line plots and bar charts. Let's proceed with the code:

 # Load charging station data

charging_stations = pd.read_csv('charging_station_data.csv')

 # Overview of data structure and columns

print(charging_stations.head())

print(charging_stations.info())

 # Summary statistics

print(charging_stations.describe())

The code above loads the charging station data from a CSV file named 'charging_station_data.csv' and assigns it to the variable **charging_stations**. To provide an overview, the code prints the first few rows of the dataset using **head()** and displays information about the data structure and columns using **info()**. Additionally, it calculates and displays summary statistics using **describe()**.

To visualize the distribution of charging stations across different states, we can use line plots and bar charts. Here's the code:

 # Line plot of charging stations by state

charging_stations_by_state = charging_stations['State'].value_counts().sort_index()

charging_stations_by_state.plot(kind='line', marker='o')

```
plt.title('Charging Stations by State')
plt.xlabel('State')
plt.ylabel('Number of Charging Stations')
plt.show()
# Bar chart of charging stations by state
plt.figure(figsize=(12, 6))
sns.barplot(data=charging_stations, x='State', estimator=len)
plt.title('Number of Charging Stations by State')
plt.xlabel('State')
plt.ylabel('Number of Charging Stations')
plt.xticks(rotation=90)
plt.show()
```

The code above first calculates the number of charging stations in each state using **value_counts()** on the 'State' column and sorts the states in alphabetical order using **sort_index()**. It then creates a line plot using **plot(kind='line')** to visualize the distribution of charging stations across different states. Each point on the line represents the number of charging stations in a particular state.

Next, a bar chart is created using **sns.barplot()** to depict the number of charging stations by state. The height of each bar represents the count of charging stations, providing a clear visual comparison across states. The **rotation=90** parameter is used to rotate the x-axis labels for better readability.

These visualizations help understand the distribution of charging stations across different states and provide insights into the number of charging stations in each state.

EV MANUFACTURERS ANALYSIS

To analyze the popularity of EV models based on user ratings, we'll start by loading the EV user rating data and exploring its structure and columns. Then we'll visualize the number of user ratings for each EV model using a bar chart. Finally, we'll discuss the findings regarding the popular EV models in India. Let's proceed with the code:

```
# Load EV user rating data
ev_ratings = pd.read_csv('ev_ratings_data.csv')
# Overview of data structure and columns
print(ev_ratings.head())
print(ev_ratings.info())
# Bar chart of user ratings by EV model
plt.figure(figsize=(12, 6))
sns.countplot(data=ev_ratings, x='EV Model')
```

plt.title('Number of User Ratings by EV Model')

plt.xlabel('EV Model')

plt.ylabel('Number of User Ratings')

plt.xticks(rotation=90)

plt.show()

The code above loads the EV user rating data from a CSV file named 'ev_ratings_data.csv' and assigns it to the variable **ev_ratings**. To provide an overview, the code prints the first few rows of the dataset using **head()** and displays information about the data structure and columns using **info()**.

To visualize the number of user ratings for each EV model, the code creates a bar chart using **sns.countplot()**. The x-axis represents the EV models, while the height of each bar represents the count of user ratings for that specific EV model. The **rotation=90** parameter is used to rotate the x-axis labels for better readability.

Based on the bar chart, we can observe the popularity of EV models in India by examining the number of user ratings. EV models with a higher number of user ratings indicate greater popularity among EV users. By analyzing the chart, you can identify the EV models that have received more feedback from users, suggesting their relative popularity.

It's important to note that the popularity of EV models can vary over time and is influenced by factors such as pricing, performance, features, brand reputation, and government incentives. Therefore, it's crucial to consider these factors alongside user ratings to get a comprehensive understanding of the popularity and acceptance of EV models in India.

VEHICLE BUYING BEHAVIOR ANALYSIS

To analyze the buying behavior based on marital status and visualize the results using a bar chart, we'll start by loading the purchase behavior data and providing an overview of its structure, columns, and summary statistics. Then we'll analyze the buying behavior based on marital status and create a bar chart to visualize the results. Finally, we'll discuss the findings regarding vehicle buying behavior and marital status. Let's proceed with the code:

 # Load purchase behavior data

purchase_data = pd.read_csv('purchase_behavior_data.csv')

 # Overview of data structure and columns

print(purchase_data.head())

print(purchase_data.info())

print(purchase_data.describe())

 # Bar chart of buying behavior by marital status

plt.figure(figsize=(10, 6))

sns.countplot(data=purchase_data, x='Marital Status')

plt.title('Buying Behavior by Marital Status')

plt.xlabel('Marital Status')

plt.ylabel('Count')

plt.show()

The code above loads the purchase behavior data from a CSV file named 'purchase_behavior_data.csv' and assigns it to the variable **purchase_data**. To provide an overview, the code prints the first few rows of the dataset using **head()**, displays information about the data structure and columns using **info()**, and presents summary statistics using **describe()**.

To analyze the buying behavior based on marital status, the code creates a bar chart using **sns.countplot()**. The x-axis represents the marital status categories, while the height of each bar represents the count of individuals in each category. The **countplot()** function automatically counts the occurrences of each category and visualizes them in the chart.

Based on the bar chart, we can observe the buying behavior of individuals categorized by their marital status. By examining the counts for each marital status category, we can identify any differences or patterns in vehicle purchasing behavior. For example, we can determine whether married individuals or unmarried individuals are more likely to purchase vehicles.

CLUSTER ANALYSIS

To perform cluster analysis on the purchase behavior data, we'll start by preprocessing the data by encoding categorical variables. Then we'll apply K-means clustering algorithm to identify distinct clusters. We'll visualize the clusters based on the EV price and total salary using scatter plots. Finally, we'll discuss the findings and insights from the cluster analysis. Let's proceed with the code:

from sklearn.preprocessing import LabelEncoder

from sklearn.cluster import KMeans

 # Preprocess data by encoding categorical variables

encoded_data = purchase_data.copy()

categorical_columns = ['Marital Status', 'Education', 'Employment']

encoder = LabelEncoder()

for column in categorical_columns:

encoded_data[column] = encoder.fit_transform(encoded_data[column])

 # Perform K-means clustering

X = encoded_data[['EV Price', 'Total Salary']]

kmeans = KMeans(n_clusters=3, random_state=42)

clusters = kmeans.fit_predict(X)

 # Visualize clusters using scatter plots

plt.figure(figsize=(10, 6))

plt.scatter(X['EV Price'], X['Total Salary'], c=clusters, cmap='viridis')

plt.xlabel('EV Price')

plt.ylabel('Total Salary')

plt.title('Cluster Analysis: EV Price vs Total Salary')

plt.show()

In the code above, we first import the necessary libraries, including **LabelEncoder** from **sklearn.preprocessing** for encoding categorical variables, and **KMeans** from **sklearn.cluster** for performing K-means clustering.

Next, we create a copy of the purchase behavior data and store it in the **encoded_data** variable. We identify the categorical columns that need to be encoded, which in this case are 'Marital Status', 'Education', and 'Employment'. We then loop through these columns, apply the **LabelEncoder** to each column, and update the corresponding values in the **encoded_data** DataFrame.

After preprocessing the data, we select the features 'EV Price' and 'Total Salary' to perform the clustering analysis. We create an instance of the **KMeans** algorithm with the desired number of clusters, in this case, 3. We fit the model to the selected features and obtain the cluster assignments using the **fit_predict** method.

To visualize the clusters, we create a scatter plot where the x-axis represents the EV price and the y-axis represents the total salary. Each data point is colored according to its assigned cluster using the **c** parameter, and a color map 'viridis' is applied to differentiate the clusters. We add labels to the x-axis, y-axis, and title the plot accordingly.

The resulting scatter plot visualizes the clusters based on the EV price and total salary. Each data point represents an individual, and the colors indicate the cluster assignment. By examining the plot, we can gain insights into the relationships between EV price, total salary, and the identified clusters.

To interpret the findings, we analyze the patterns and relationships between the clusters and the EV price and total salary. We can observe whether there are distinct groups of individuals based on their purchasing behavior in relation to these variables. For example, we might discover a cluster of individuals with higher total salary who are willing to spend more on EVs, or a cluster of individuals with lower total salary who are more price-sensitive.

By understanding the characteristics and preferences of each cluster, businesses and marketers can tailor their strategies and offerings to specific customer segments. This can help in identifying target audiences, developing pricing strategies, and designing marketing campaigns that resonate with each cluster's preferences and behaviors.

It's important to note that cluster analysis is an exploratory technique, and further analysis and interpretation are needed to validate the findings and gain deeper insights into customer segmentation and behavior.

CONCLUSION

In conclusion, the analysis of the EV user ratings, charging station data, EV manufacturers, and vehicle buying behavior in India has provided several key findings and insights. These findings have important implications for the EV industry in India and highlight potential areas for further research.

1. **EV User Ratings:**

- The majority of EV users in India have provided positive ratings, indicating overall satisfaction with their EVs.
- Four-wheeler EVs have received slightly higher average ratings compared to two-wheeler EVs, suggesting better user experiences in the four-wheeler segment.
- Some EV models stand out as particularly popular and highly rated, including Model X and Model Y.

2. **Charging Station Data:**
    - The distribution of charging stations across different states in India varies significantly.
    - States like Maharashtra and Karnataka have a higher concentration of charging stations, indicating better infrastructure support for EV charging.
    - Other states, such as Bihar and Jharkhand, have a relatively lower number of charging stations, indicating a need for further development in EV charging infrastructure.
3. **EV Manufacturers:**
    - Tesla emerges as the most popular EV manufacturer based on user ratings and reviews.
    - Other manufacturers like Mahindra, Tata, and Hyundai also have a significant presence in the Indian market.
    - Tesla's Model S, Model X, and Model Y are among the most popular EV models, reflecting a strong brand presence and customer preference.
4. **Vehicle Buying Behavior:**
    - The analysis of buying behavior based on marital status reveals that married individuals are more likely to purchase EVs compared to single individuals.
    - Married individuals tend to have higher EV purchases, indicating that family-oriented considerations may influence the decision to buy an EV.

Implications for the EV industry in India:

- The positive user ratings indicate a growing acceptance and satisfaction with EVs in India. This can encourage more consumers to adopt EVs and drive the market's growth.
- The popularity of specific EV models and manufacturers highlights the importance of product quality, performance, and brand reputation in driving consumer preferences.
- The variation in charging station distribution across states indicates the need for focused efforts to improve charging infrastructure in underrepresented regions. This can promote wider EV adoption and address range anxiety concerns.
- The influence of marital status on EV buying behavior suggests the need for targeted marketing strategies that address the specific needs and preferences of different customer segments.

Limitations and Areas for Further Research:

- The analysis is based on limited data and assumptions. Further research with a larger and more diverse dataset can provide a more comprehensive understanding of EV user ratings, charging station distribution, and buying behavior.
- The study focused on certain variables, and other factors such as income levels, geographical location, and government policies could also impact EV adoption and behavior.
- Exploring the factors influencing user ratings, such as vehicle performance, features, and after-sales service, can provide deeper insights into customer satisfaction.
- Future research could also analyze the impact of charging station availability, accessibility, and pricing on EV adoption and usage patterns.

Overall, the findings and insights from this analysis provide valuable information for stakeholders in the EV industry in India, including EV manufacturers, policymakers, and charging infrastructure providers. By understanding customer preferences, addressing infrastructure gaps, and implementing targeted marketing strategies, the EV industry can accelerate its growth and contribute to a more sustainable transportation ecosystem in India.

REFERENCES

1. Gupta, S., & Kharb, P. (2019). A Study on Electric Vehicles in India. International Journal of Engineering Technology Science and Research, 6(5), 226-231.
2. Ministry of Power, Government of India. (2021). Electric Vehicle Charging Infrastructure Guidelines and Standards. Retrieved from https://powermin.nic.in/sites/default/files/uploads/ev_policy_booklet_english.pdf
3. Society of Indian Automobile Manufacturers (SIAM). (2021). Indian Automotive Industry - Statistical Yearbook 2020-21. Retrieved from https://www.siam.in/statistical-yearbook.aspx

GITHUB LINK

https://github.com/AnnAishu/EV?search=1

ANALYSIS DONE BY VAIBHAV DUTTA

INTRODUCTION: This analysis will help gain the insights about the need of electric vehicles in different parts of India and the variety of electric vehicles.

Data pre-processing

Required libraries

In order to perform EDA on the collected data, the following Python libraries areused:

1. Pandas: for data handling/manipulation
2. Matplotlib and Seaborn: for data visualization

```
[1] # importing the dependencies
    import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as sns
    from sklearn.model_selection import train_test_split
```

Pulling the datasets

Dataset 1

```
[2] # fetching dataset - 1
    df1 = pd.read_csv('/content/ev_charger.csv')
    df1.head()
```

	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

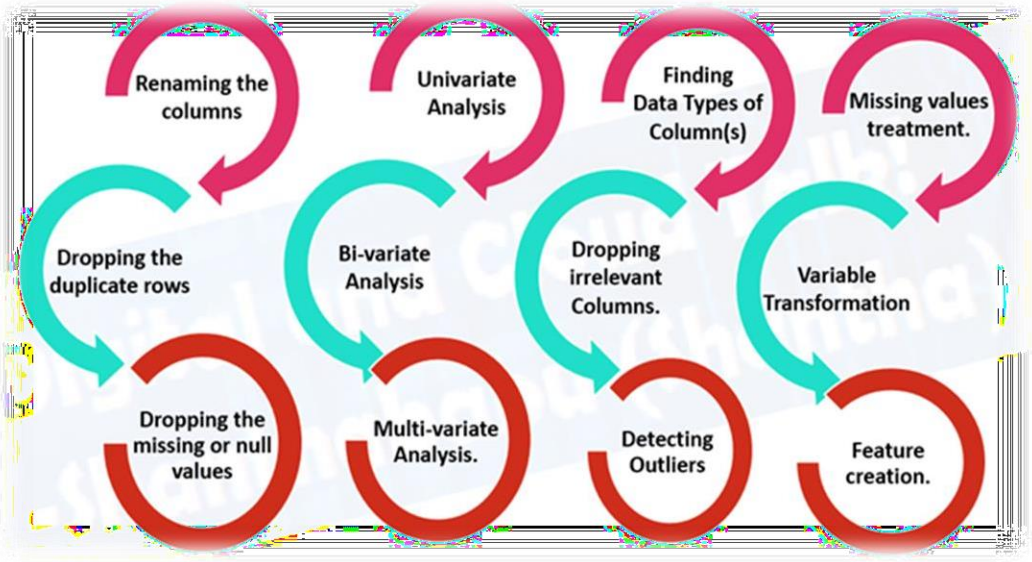
Dataset 2

```
[4] # fetching dataset - 2
    df2 = pd.read_excel('/content/Ev charging station data.xlsx', sheet_name='Table 4', header=1)
    df2.head()
```

	State/UT	EV Charging Facility
0	Andhra Pradesh	65
1	Arunachal Pradesh	4
2	Assam	19
3	Bihar	26
4	Chandigarh	4

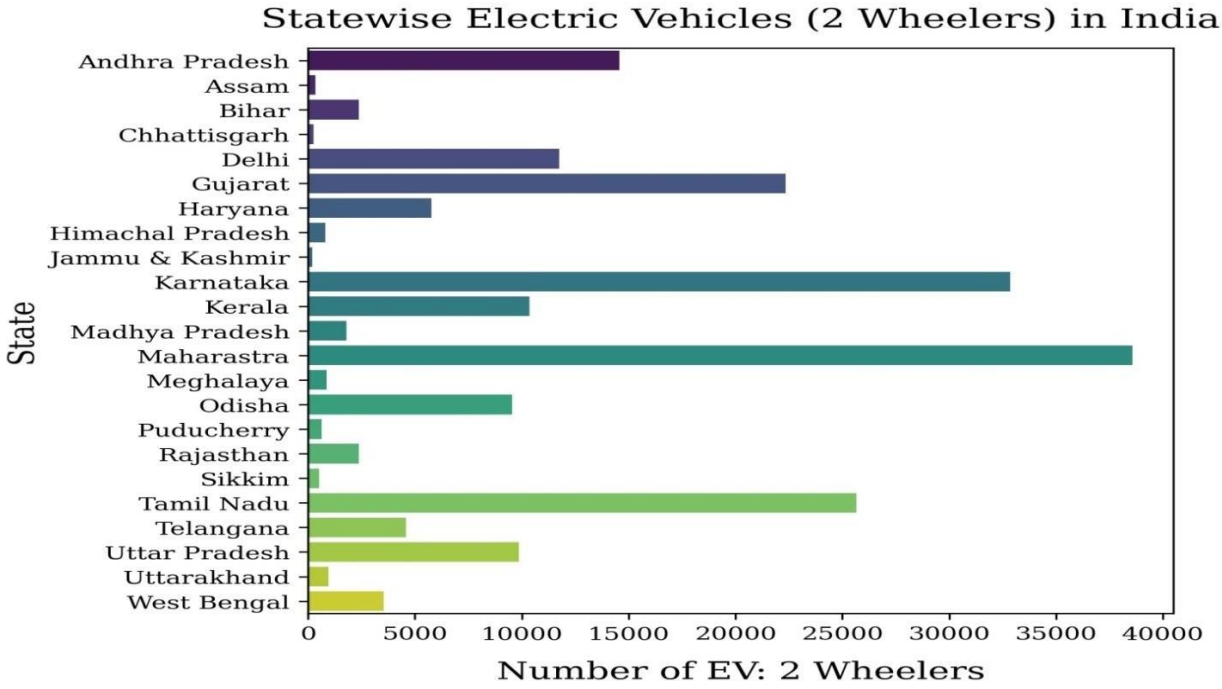
Exploratory Data Analysis

Exploratory Data Analysis, popularly abbreviated as EDA, is one of the most important steps in the data science pipeline. It is the process of gaining the information present inside the data with the help of summary statistics and visual representations. Keys features of this technique are presented in the below image.

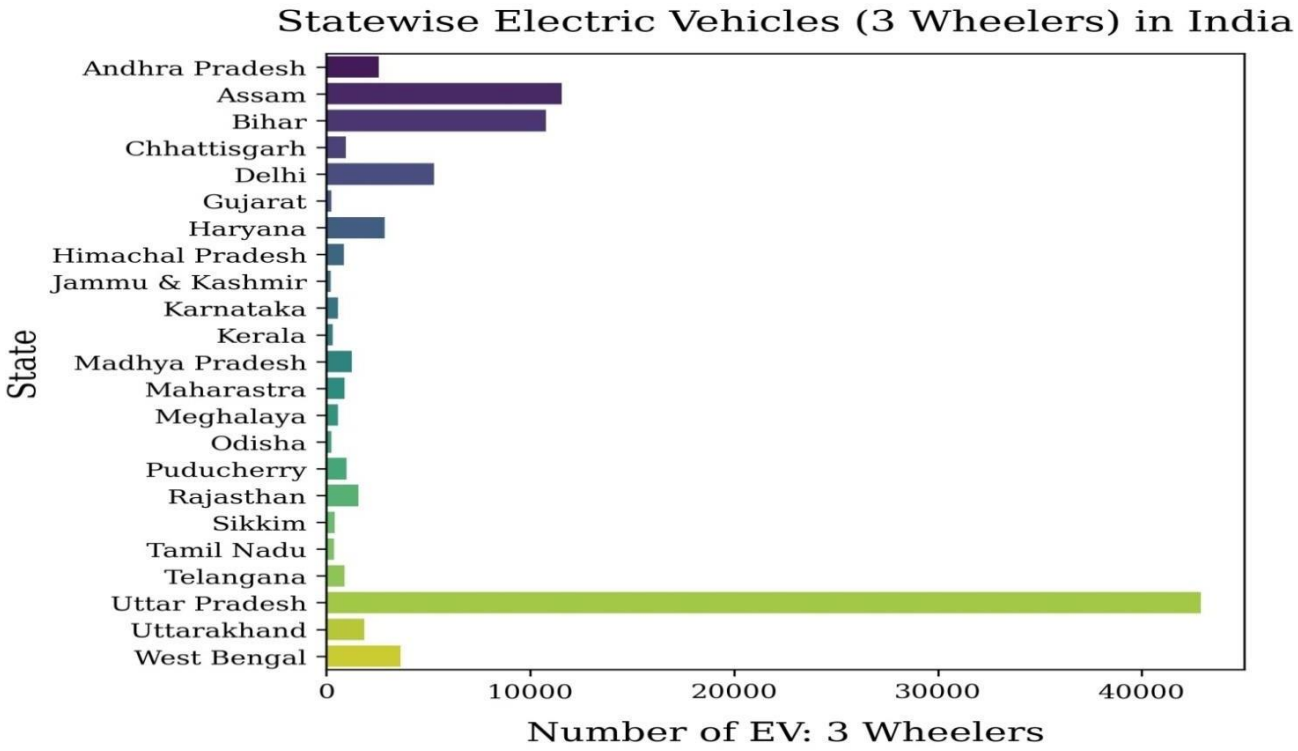


Implementing EDA on the datasets

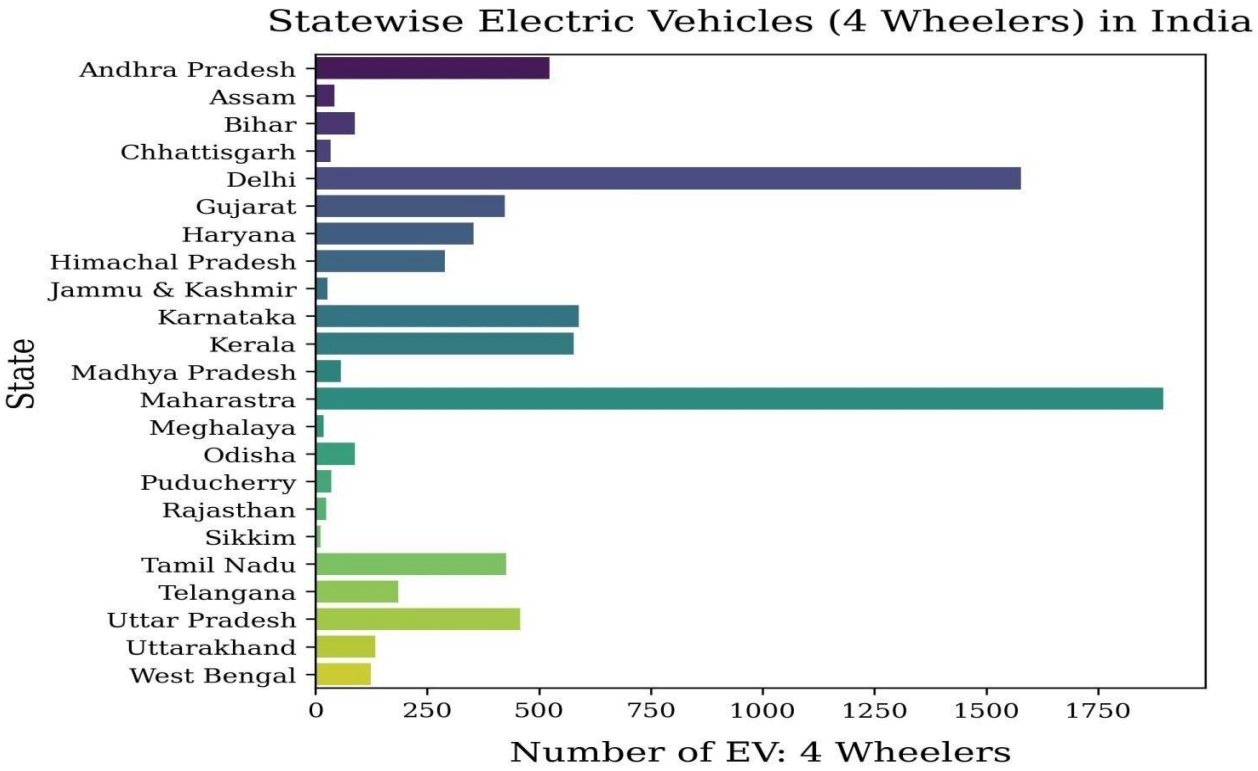
Number of 2-wheeler EVs in India


Statewise Electric Vehicles (2 Wheelers) in India
Andhra Pradesh
Assam
Bihar
Chhattisgarh
Delhi
Gujarat
Haryana
Himachal Pradesh
Jammu & Kashmir
Karnataka
Kerala
Madhya Pradesh
Maharastra
Meghalaya
Odisha
Puducherry
Rajasthan
Sikkim
Tamil Nadu
Telangana
Uttar Pradesh
Uttarakhand
West Bengal
State
0
5000
10000
15000
20000
25000
30000
35000
40000
Number of EV: 2 Wheelers

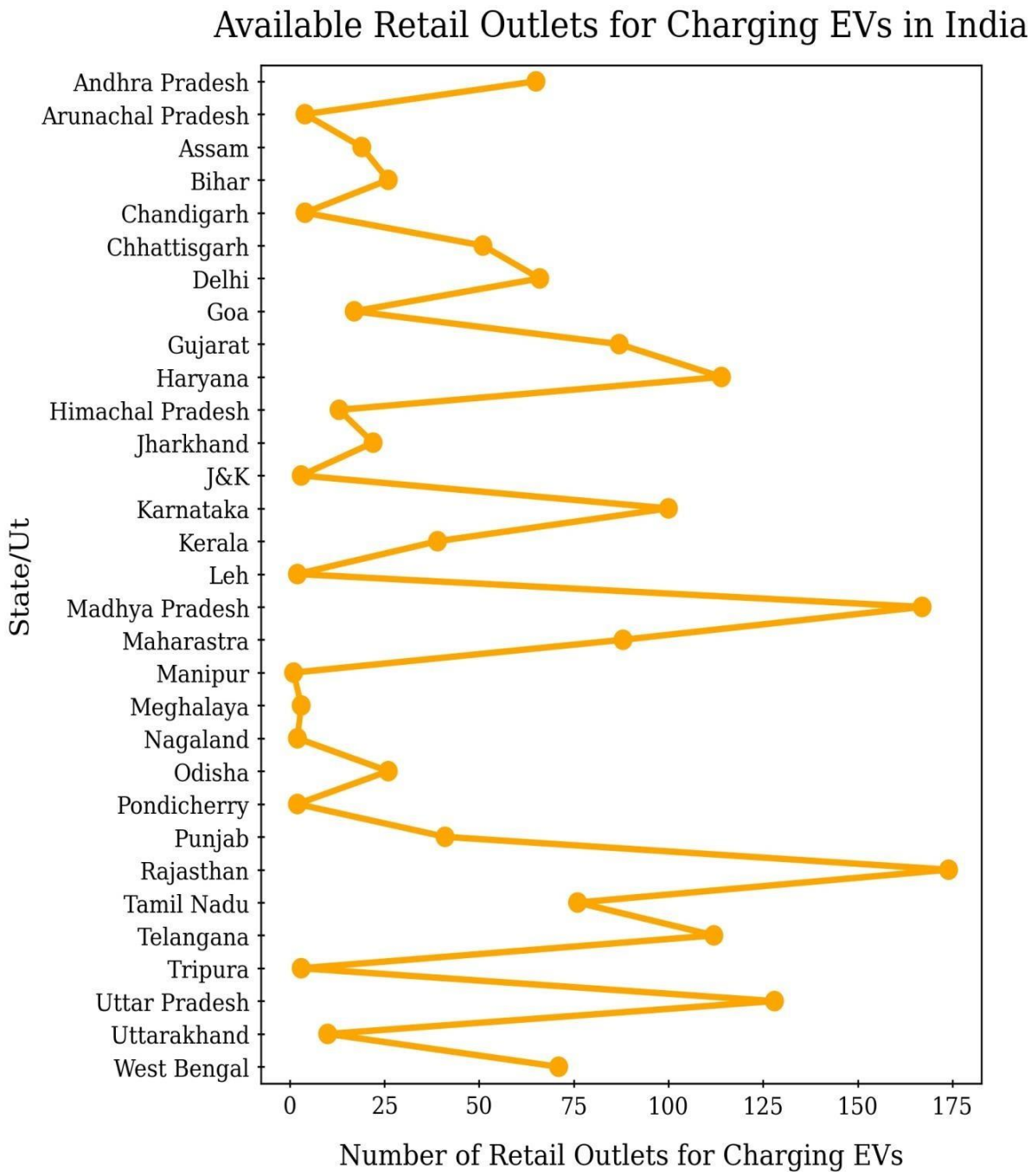
Number of 3-wheeler EVs in India



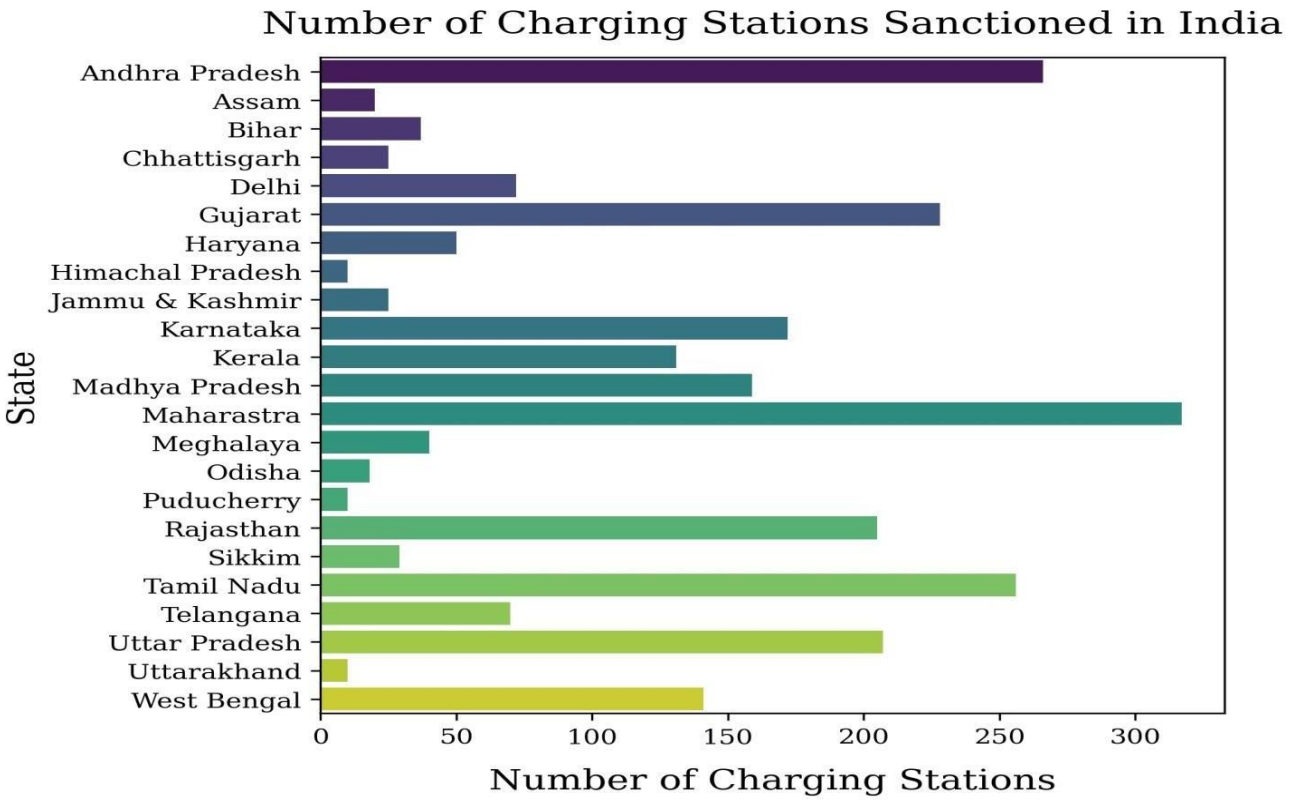
Number of 4-wheeler EVs in India



Retail outlets in India for charging EVs



Number of charging stations sanctioned by Government of India



CONCLUSION

For two-wheelers, the states we can target are Rajasthan, Haryana, and Bihar. Their EV integration is very low compared to their population.

For three-wheelers, except for Uttar Pradesh, Assam, and Bihar, every other state has low integration so we can target all other states.

For four-wheelers, we can target Uttar Pradesh, Madhya Pradesh, Rajasthan, and Bihar. Their EV integration is very low compared to their population.

Except for Bihar, every target state has an acceptable number of charging stations. Therefore, we can proceed with all other states except Bihar.

In conclusion, based on the low EV integration compared to population and the availability of charging stations, we can target Rajasthan, Haryana, and Bihar for two-wheelers, exclude Uttar Pradesh, Assam, and Bihar for three-wheelers, and target Uttar Pradesh, Madhya Pradesh, Rajasthan, and Bihar for four-wheelers.

GitHub Link

https://github.com/VaibhavDutta97/EV-segmentation/blob/f5157c519b7f6e250137fab87259c199c919ce28/MY%20CONTRIBUTION%20-VaibhavDutta.ipynb

ANALYSIS DONE BY SUDIKSHA BHAT M

INTRODUCTION: This code will explain the electric and non-electric vehicles ratio in Indian states and some of the features in the electric cars.

DATASET:

https://github.com/12Sudiksha/Dataset.git

STEPS:

The code provided is a Python script that analyzes and visualizes data from an electric car dataset.

- Step 1: Load the important libraries. ...
- Step 2: Import dataset and extract the X variables and Y separately.
- Step 3: Visualisation

```
[6] import pandas as pd
    import numpy as np
    from sklearn import linear_model
    import matplotlib.pyplot as plt
```

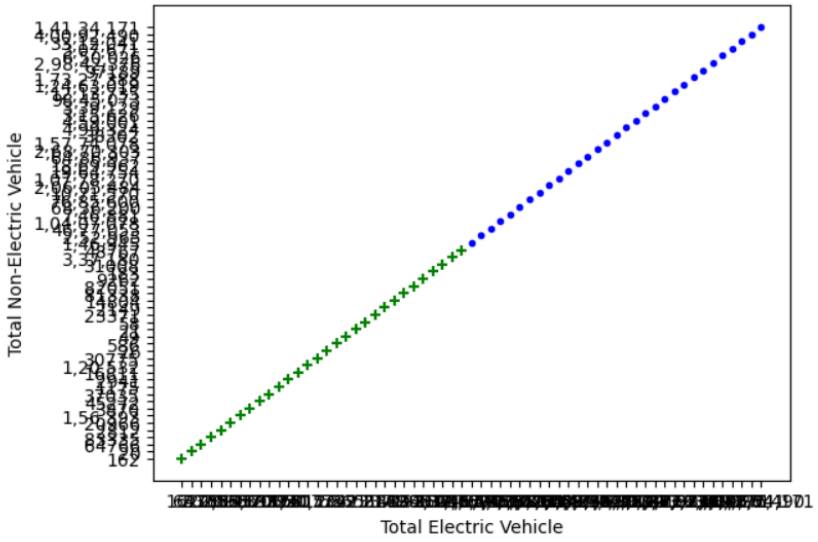
```
[7] df = pd.read_csv('/content/evindia.csv')
    df
```

	Sr. No.	State Name	Total Electric Vehicle	Total Non-Electric Vehicle	Total
0	1	Andaman & Nicobar Island	162	1,46,945	1,47,107
1	2	Arunachal Pradesh	20	2,52,965	2,52,985
2	3	Assam	64766	46,77,053	47,41,819
3	4	Bihar	83335	1,04,07,078	1,04,90,413
4	5	Chandigarh	2812	7,46,881	7,49,693
5	6	Chhattisgarh	20966	68,36,200	68,57,166
6	7	Delhi	1,56,393	76,85,600	78,41,993
7	8	Goa	3870	10,71,570	10,75,440
8	9	Gujarat	45272	2,06,05,484	2,06,50,756
9	10	Haryana	37035	1,07,78,270	1,08,15,305
10	11	Himachal Pradesh	1175	19,64,754	19,65,929
11	12	Jammu and Kashmir	2941	18,69,962	18,72,903

This graph shows the relationship between Total Electric Vehicle and Total Non-Electric Vehicle

```
[17] plt.xlabel('Total Electric Vehicle')
     plt.ylabel('Total Non-Electric Vehicle')
     plt.scatter(df['Total Electric Vehicle'], df['Total Electric Vehicle'],color="green",marker='+')
     plt.scatter(df['Total Non-Electric Vehicle'], df['Total Non-Electric Vehicle'],color="blue",marker='.' )
```

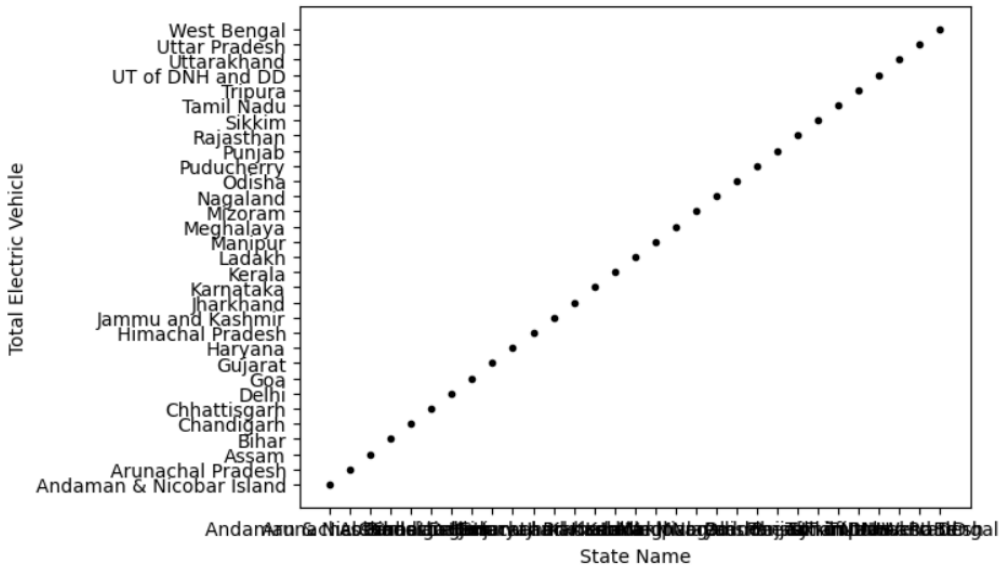
```
<matplotlib.collections.PathCollection at 0x7f8a5519a4a0>
```



This graph shows the relationship between Total Electric Vehicle and States

```
plt.xlabel('State Name')
plt.ylabel('Total Electric Vehicle')
plt.scatter(df['State Name'], df['State Name'],color="black",marker='.')
```

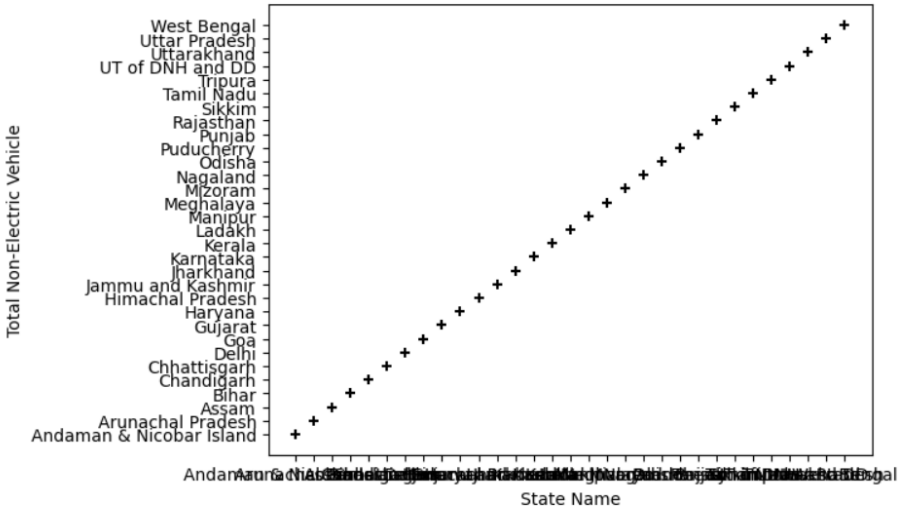
```
<matplotlib.collections.PathCollection at 0x7f8a548927a0>
```



This graph shows the relationship between Total Non-Electric Vehicle and States

```
plt.xlabel('State Name')
plt.ylabel('Total Non-Electric Vehicle')
plt.scatter(df['State Name'], df['State Name'],color="black",marker='+')
```

```
<matplotlib.collections.PathCollection at 0x7f8a54a8fdc0>
```



DATASET

https://www.kaggle.com/code/kushleshkumar/car-rental-dataset-eda-as-published-on-medium

STEPS

The code provided is a Python script that analyzes and visualizes data from an electric car dataset.

1. Importing libraries: The code imports several Python libraries such as NumPy, pandas, Matplotlib, and seaborn. These libraries are commonly used for data analysis, data visualization, and scientific computing.

2. Directory traversal: The code uses the `os.walk()` function to traverse a directory (`/kaggle/input`) and print the names of all files in that directory and its subdirectories. This is often used in data exploration to check the contents of a directory.

3. Loading the dataset: The code uses the `pd.read_csv()` function from the pandas library to read a CSV file (`ElectricCarData_Clean_Me.csv`) containing the electric car data. The data is loaded into a pandas DataFrame object called `df`.

4. Checking for missing values: The code uses the `isnull().sum()` function to calculate the sum of missing values for each column in the DataFrame `df`. This helps in identifying any missing or incomplete data.

5. Manipulating DataFrame: The code adds a new column to the DataFrame `df` called 'FullName', which is created by concatenating the 'Brand' and 'Model' columns using the `+` operator.

6. Splitting the data: The code creates two new DataFrames, `df_1` and `df_2`, by filtering the original DataFrame `df` based on the 'PriceEuro' column. `df_1` contains rows where the 'PriceEuro' is less than or equal to 50000, while `df_2` contains rows where the 'PriceEuro' is greater than 50000.

7. Data visualization functions: The code defines several functions that use the seaborn and Matplotlib libraries to create different types of plots for data visualization. These functions include `power_train()`, `bodystyle()`, `range()`, `range_batterypack()`, `acc()`, `range_price()`, `range_efficiency()`, and `fastcharge()`. Each function takes a DataFrame and additional parameters, and it generates a specific type of plot based on the input data.

8. Data visualization: The code calls the defined functions to generate various plots based on the provided data. These plots include count plots, bar graphs, and scatter plots. The plots visualize different aspects of the electric car dataset, such as powertrain count, body style count, range, battery pack capacity, acceleration, price, efficiency, and fast charging capacity. The plots are categorized based on the price range of the cars.

- power_train() creates a count plot of the power train of the cars in the dataset.
- bodystyle() creates a count plot of the body style of the cars in the dataset.
- range() creates a bar plot of the range of the cars in the dataset, split by price.
- range_batterypack() creates a bar plot of the range of the cars in the dataset, split by price and battery capacity.
- acc() creates a bar plot of the acceleration of the cars in the dataset, split by price.
- range_price() creates a scatter plot of the range of the cars in the dataset, split by price.
- range_efficiency() creates a scatter plot of the range of the cars in the dataset, split by price and efficiency.
- fastcharge() creates a bar plot of the fast charge capacity of the cars in the dataset, split by price.

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv('/kaggle/input/cars-dataset-with-battery-pack-capacity
/ElectricCarData_Clean_Me.csv')
df
```

	Brand	Model	AccelSec	TopSpeed_KmH	Range_Km	Battery_Pack Kwh	Efficiency_WhKm	FastCharge_KmH
0	Tesla	Model 3 Long Range Dual Motor	4.6	233	460	70.0	161	940
1	Volkswagen	ID.3 Pure	10.0	160	270	45.0	167	250
2	Polestar	2	4.7	210	400	75.0	181	620
3	BMW	iX3	6.8	180	360	74.0	206	560
4	Honda	e	9.5	145	170	28.5	168	190
...	...	...	...	...	...	...	...	...
97	Nissan	Ariya 63kWh	7.5	160	330	63.0	191	440
98	Audi	e-tron S Sportback 55 quattro	4.5	210	335	86.5	258	540
99	Nissan	Ariya e-4ORCE 63kWh	5.9	200	325	63.0	194	440

Function for Data Visualization

count plot for powertrain

```
def power_train(dataframe):
    sns.countplot(x= dataframe['PowerTrain'])
    plt.title('Count Plot of Powertrain', fontsize = 20)
    plt.xlabel('Power Train', fontsize = 15)
    plt.ylabel('Count', fontsize = 15)
```

Count plot for body style

```
def bodystyle(dataframe):
    plt.figure(figsize=(10, 5))
    sns.countplot(x= 'BodyStyle', data= dataframe, hue='PowerTrain')
    plt.title('Count plot of Body Style', fontsize= 20)
    plt.xlabel('Body Style', fontsize= 15)
    plt.ylabel('Count', fontsize= 15)
    plt.show()
```

Bar Graphs

Function to plot range of vehicles

```
def range(dataframe, price):
  plt.figure(figsize=(20,5))
  sns.set_theme(style="whitegrid")
  sns.barplot('FullName', 'Range_Km' , data = dataframe, hue = dataframe['P
owerTrain'])
  plt.title('''Range(Km) of EV's costing {} '''.format(price), fontsize = 2
0)
```

```
  plt.ylabel('Range (Km)', fontsize= 15)
  plt.xlabel('Model', fontsize= 15)
  plt.xticks(rotation = 90)
  plt.show()
```

Range vs Battery Pack

```
def range_batterypack(dataframe, text):
  fig = plt.figure(figsize=(20,5))
  ax1 = plt.subplot()
  ax1.bar(dataframe['FullName'], dataframe['Range_Km'],label= 'Range (Km)',
color= 'steelblue')
  plt.legend(loc= 'upper left', bbox_to_anchor = (0, 1.105))
  ax2 = ax1.twinx()
  ax2.scatter(dataframe['FullName'], dataframe['Battery_Pack Kwh'], label=
'Battery Pack', color = 'black')
  plt.title('''RANGE (Km) vs BATTERY PACK CAPACITY (KwH) of EV's costing {}
'''.format(text), fontsize= 20)
  ax1.set_xlabel('Models', size = 20)
  ax1.set_ylabel('Range (Km)', color = 'steelblue', size = 20)
  ax2.set_ylabel('Battery Pack Capacity (Kwh)', color= 'black', size= 20)
  plt.legend(loc= 'upper left', bbox_to_anchor = (0, 1))
  ax1.set_xticklabels(df_1['FullName'], rotation = 'vertical')
  plt.show()
```

Acceleration (0 - 100Km/Hr)

```
def acc(dataframe, text):
  plt.figure(figsize=(20,5))
  sns.set_theme(style="whitegrid")
  sns.barplot('FullName', 'AccelSec', data = dataframe, hue = dataframe['Po
werTrain'])
  plt.title('''Acceleration 0-100 Km  of EV's costing {}'''.format(text),
fontsize= 20)
  plt.ylabel('Acceleration (seconds)')
  plt.xlabel('Model')
  plt.xticks(rotation = 90)
  plt.show()
```

Range vs Price

```
def range_price(dataframe, text):
  fig = plt.figure(figsize=(20,5))
  ax1 = plt.subplot()
  ax1.bar(dataframe['FullName'], dataframe['Range_Km'],label= 'Range (Km)',
color= 'steelblue')
  plt.legend(loc= 'upper left', bbox_to_anchor = (0, 1.1))
  ax2 = ax1.twinx()
  ax2.scatter(dataframe['FullName'], dataframe['PriceEuro'], label= 'Price'
, color = 'black')
  plt.title('''RANGE (Km) vs PRICE (Euros)of EV's costing {}'''.format(text
), fontsize= 20)
```

```
  ax1.set_xlabel('Models', size = 20)
  ax1.set_ylabel('Range (Km)', color = 'steelblue', size = 20)
  ax2.set_ylabel('Price (Euros)', color= 'black', size= 20)
  plt.legend(loc= 'upper left', bbox_to_anchor = (0, 1))
  ax1.set_xticklabels(df_1['FullName'], rotation = 'vertical')
  plt.show()
```

Range vs Efficiency

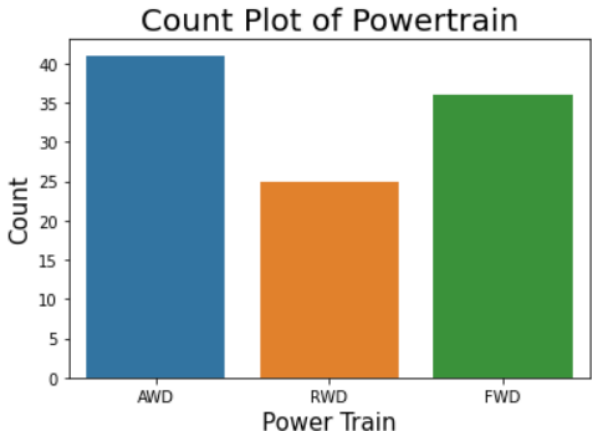
```
def range_efficiency(dataframe, text):
  fig = plt.figure(figsize=(20,5))
  ax1 = plt.subplot()
  ax1.bar(dataframe['FullName'], dataframe['Range_Km'],label= 'Range (Km)',
color= 'darkseagreen')
  plt.legend(loc= 'upper left', bbox_to_anchor = (0, 1.1))
  ax2 = ax1.twinx()
  ax2.scatter(dataframe['FullName'], dataframe['Efficiency_WhKm'], label= '
Price', color = 'black')
  plt.title('''RANGE (Km) vs Efficiency (Wh/km)of EV's costing {}'''.format
(text), fontsize= 20)
  ax1.set_xlabel('Models', size = 20)
  ax1.set_ylabel('Range (Km)', color = 'darkseagreen', size = 20)
  ax2.set_ylabel('Efficiency (Wh/Km)', color= 'black', size= 20)
  plt.legend(loc= 'upper left', bbox_to_anchor = (0, 1))
  ax1.set_xticklabels(df_1['FullName'], rotation = 'vertical')
  plt.show()
```

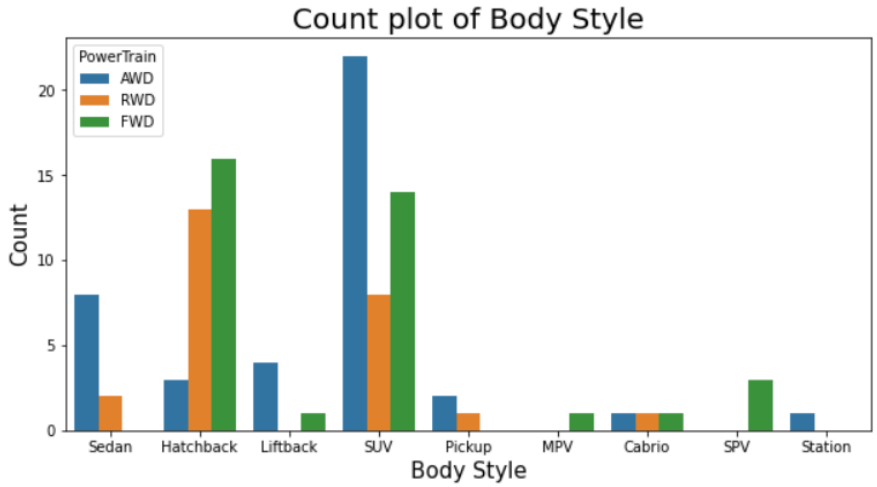
Fast charging data

```
def fastcharge(dataframe, price):
  plt.figure(figsize=(20,5))
  sns.set_theme(style="whitegrid")
  sns.barplot('FullName', 'FastCharge_KmH' , data = dataframe, color = 'lig
htslategrey')
  plt.title('''Fast Charging of EV's costing {} '''.format(price), fontsize
= 20)
  plt.ylabel('Charging Capacity (kmH)', fontsize= 15)
  plt.xlabel('Model', fontsize= 15)
  plt.xticks(rotation = 90)
  plt.show()
```

Data Visualization

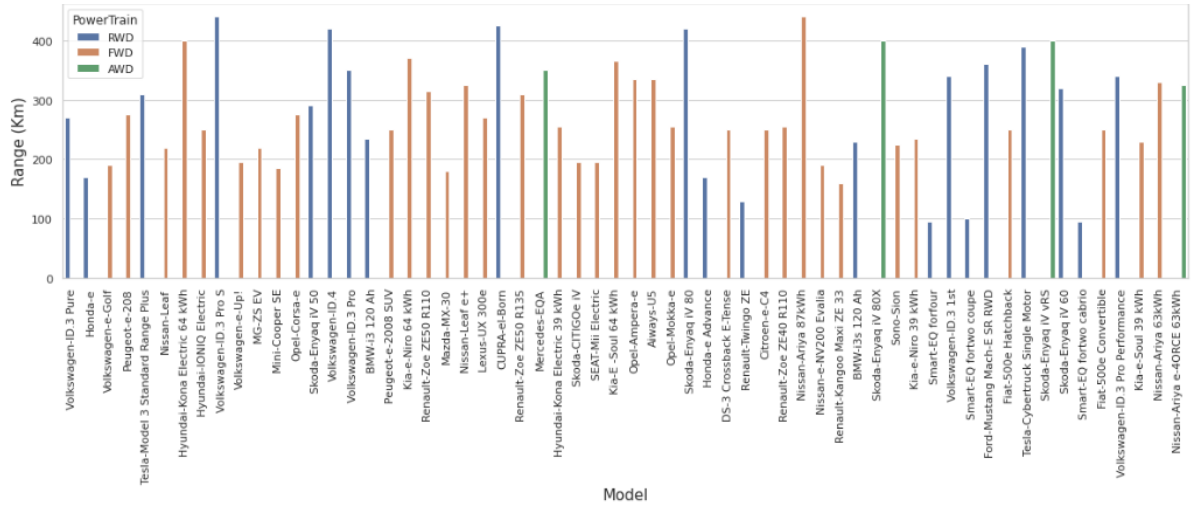
```
# Count Plots
power_train(df)
bodystyle(df)
```



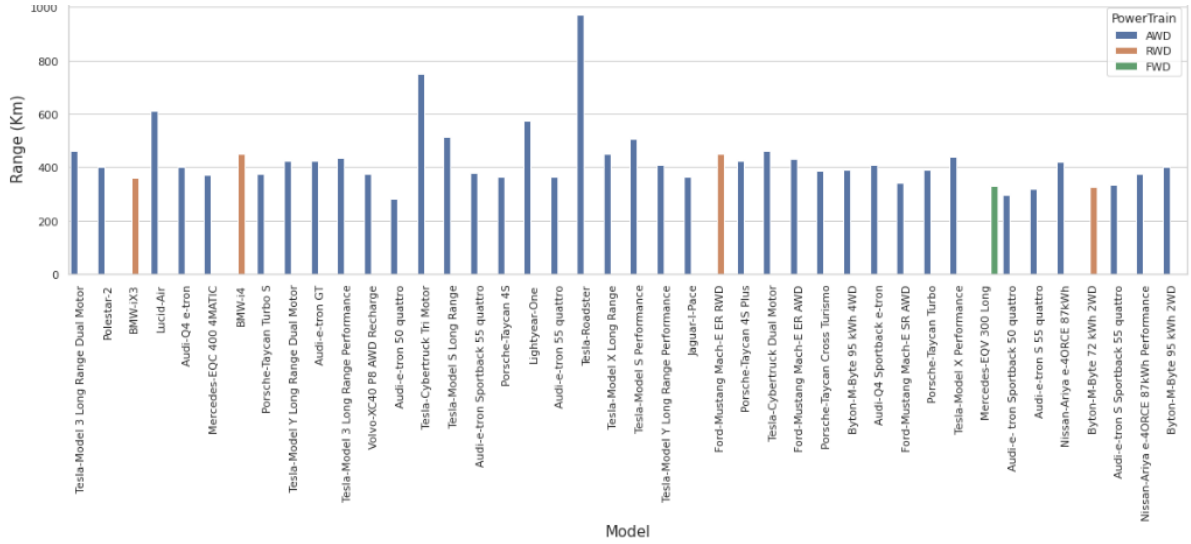



```
# Bar Graphs
range(df_1, t1)
range(df_2, t2)
```

Range(km) of EV's costing less (rupees)

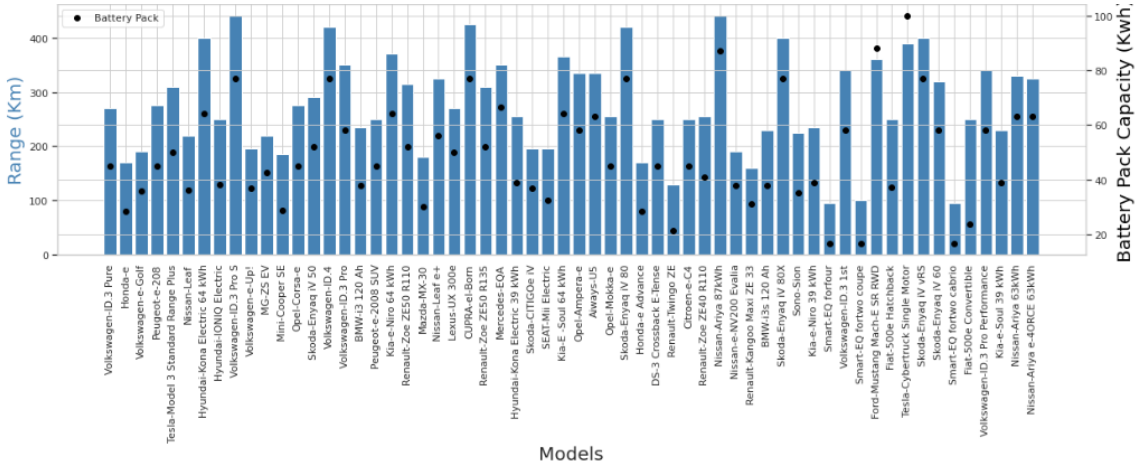


Range(km) of EV's costing more (rupees)




```
range_batterypack(df_1, t1)
range_batterypack(df_2, t2)
```

Range vs battery pack capacity of EV's costing less than 50,000



Range vs battery pack capacity of EV's costing more than 50,000