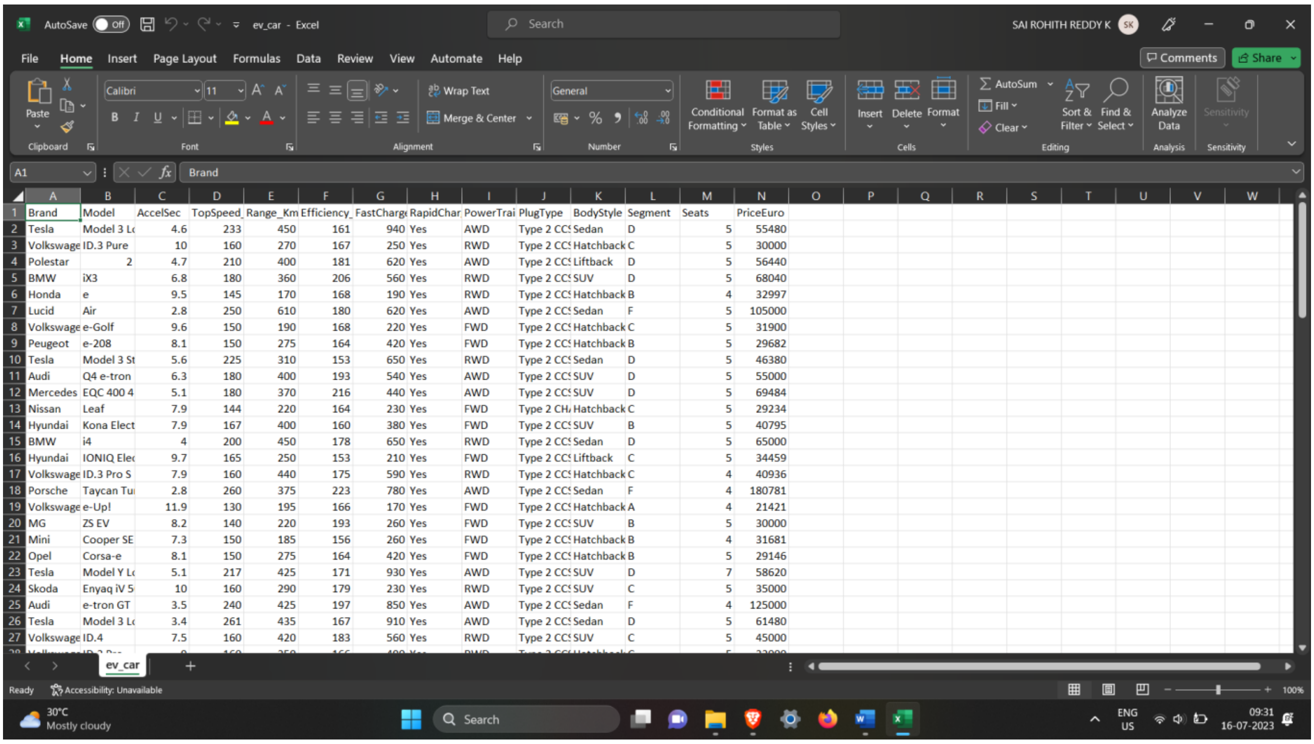
K.SAI ROHITH REDDY

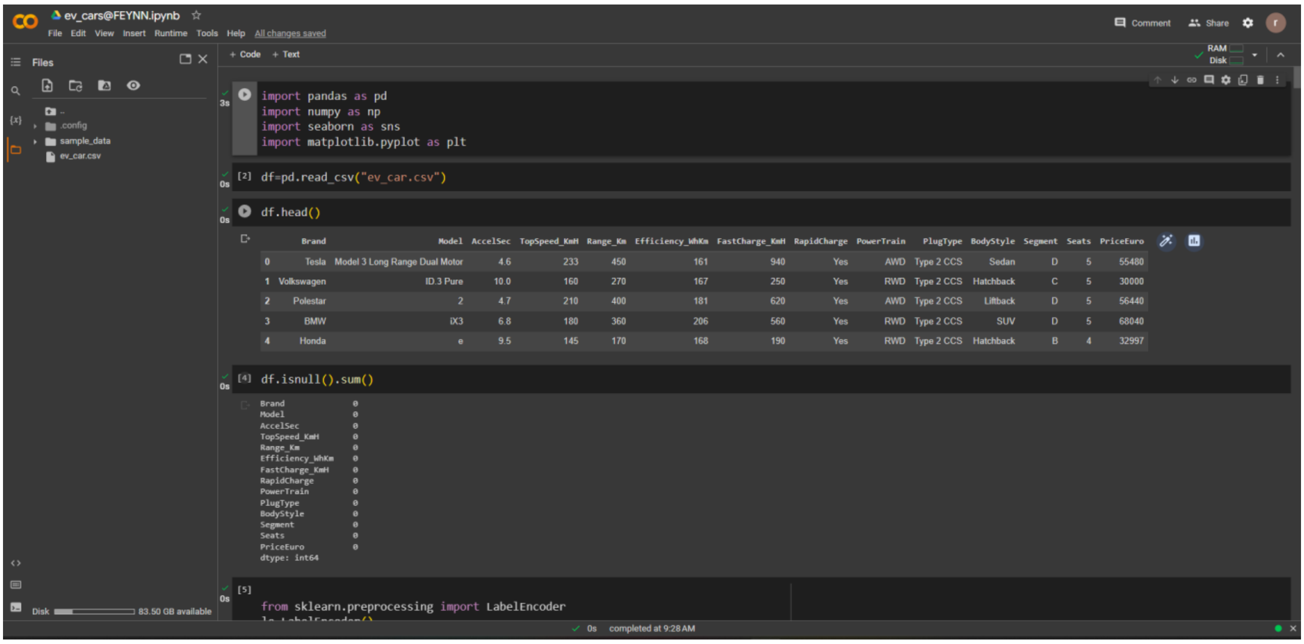
PROJECT -2

MARKET SEGMENTATION ON ELECTRIC VEHICLE CARS

DATASET SOURCE: https://www.kaggle.com/datasets/geoffnel/evs-one-electric-vehicle-dataset



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Brand	Model	AccelSec	TopSpeed_	Range_Km	Efficiency_	FastCharge	RapidChar	PowerTrai	PlugType	BodyStyle	Segment	Seats	PriceEuro
2	Tesla	Model 3 Lo	4.6	233	450	161	940	Yes	AWD	Type 2 CCS	Sedan	D	5	55480
3	Volkswage	ID.3 Pure	10	160	270	167	250	Yes	RWD	Type 2 CCS	Hatchback	C	5	30000
4	Polestar	2	4.7	210	400	181	620	Yes	AWD	Type 2 CCS	Liftback	D	5	56440
5	BMW	iX3	6.8	180	360	206	560	Yes	RWD	Type 2 CCS	SUV	D	5	68040
6	Honda	e	9.5	145	170	168	190	Yes	RWD	Type 2 CCS	Hatchback	B	4	32997
7	Lucid	Air	2.8	250	610	180	620	Yes	AWD	Type 2 CCS	Sedan	F	5	105000
8	Volkswage	e-Golf	9.6	150	190	168	220	Yes	FWD	Type 2 CCS	Hatchback	C	5	31900
9	Peugeot	e-208	8.1	150	275	164	420	Yes	FWD	Type 2 CCS	Hatchback	B	5	29682
10	Tesla	Model 3 St	5.6	225	310	153	650	Yes	RWD	Type 2 CCS	Sedan	D	5	46380
11	Audi	Q4 e-tron	6.3	180	400	193	540	Yes	AWD	Type 2 CCS	SUV	D	5	55000
12	Mercedes	EQC 400 4	5.1	180	370	216	440	Yes	AWD	Type 2 CCS	SUV	D	5	69484
13	Nissan	Leaf	7.9	144	220	164	230	Yes	FWD	Type 2 CHA	Hatchback	C	5	29234
14	Hyundai	Kona Elect	7.9	167	400	160	380	Yes	FWD	Type 2 CCS	SUV	B	5	40795
15	BMW	i4	4	200	450	178	650	Yes	RWD	Type 2 CCS	Sedan	D	5	65000
16	Hyundai	IONIQ Elec	9.7	165	250	153	210	Yes	FWD	Type 2 CCS	Liftback	C	5	34459
17	Volkswage	ID.3 Pro S	7.9	160	440	175	590	Yes	RWD	Type 2 CCS	Hatchback	C	4	40936
18	Porsche	Taycan Tu	2.8	260	375	223	780	Yes	AWD	Type 2 CCS	Sedan	F	4	180781
19	Volkswage	e-Up!	11.9	130	195	166	170	Yes	FWD	Type 2 CCS	Hatchback	A	4	21421
20	MG	ZS EV	8.2	140	220	193	260	Yes	FWD	Type 2 CCS	SUV	B	5	30000
21	Mini	Cooper SE	7.3	150	185	156	260	Yes	FWD	Type 2 CCS	Hatchback	B	4	31681
22	Opel	Corsa-e	8.1	150	275	164	420	Yes	FWD	Type 2 CCS	Hatchback	B	5	29146
23	Tesla	Model Y Lo	5.1	217	425	171	930	Yes	AWD	Type 2 CCS	SUV	D	7	58620
24	Skoda	Enyaq iV 5	10	160	290	179	230	Yes	RWD	Type 2 CCS	SUV	C	5	35000
25	Audi	e-tron GT	3.5	240	425	197	850	Yes	AWD	Type 2 CCS	Sedan	F	4	125000
26	Tesla	Model 3 Lo	3.4	261	435	167	910	Yes	AWD	Type 2 CCS	Sedan	D	5	61480
27	Volkswage	ID.4	7.5	160	420	183	560	Yes	RWD	Type 2 CCS	SUV	C	5	45000



```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

```python
df=pd.read_csv("ev_car.csv")
```

```python
df.head()
```

	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

```python
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
dtype: int64
```

```python
from sklearn.preprocessing import LabelEncoder
```

FEYNN LABS
Experiment with your Knowledge

K.SAI ROHITH REDDY

IMPORTING REQUIRED LIBRARIES

```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

READING THE DATASET AND CHECKING THE NULL VALUES

```python
[2] df=pd.read_csv("ev_car.csv")
[3] df.head()
```

	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

```python
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
dtype: int64
```

CONVERTING CATEGORICAL VARIABLES INTO NUMERICAL VARIABLES USING LABELENCODING

```python
[5]
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
categorical_cols=["RapidCharge","PowerTrain","PlugType","BodyStyle","Segment"]
for col in categorical_cols:
    df[col] = le.fit_transform(df[col])
```

```python
[6] df.head(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	1	0	2	7	3	5	55480
1	Volkswagen	ID.3 Pure	10.0	160	270	167	250	1	2	2	1	2	5	30000

```python
df.FastCharge_KmH.unique()
```

```
array(['940', '250', '620', '560', '190', '220', '420', '650', '540',
       '440', '230', '380', '210', '590', '780', '170', '260', '930',
       '850', '910', '490', '470', '270', '450', '350', '710', '240',
       '390', '570', '610', '340', '730', '920', '-', '550', '900', '520',
       '430', '890', '410', '770', '460', '360', '810', '480', '290',
       '330', '740', '510', '320', '500'], dtype=object)
```

CHOOSING THE ATTRIBUTES FOR MODEL-1

Selecting Top speed and Efficiency attributes

Choosing the TopSpeed_KmH and Efficiency_WhKm Attributes

```python
[8] X = df.iloc[:,[3,5]].values
print(X)
```

```
[140 168]
[200 171]
[250 184]
[155 154]
[200 228]
[130 166]
[130 166]
[167 175]
[150 173]
[250 195]
[150 104]
[150 188]
[200 237]
[410 206]
[150 176]
[160 183]
[250 211]
[145 168]
[150 180]
[135 164]
[150 180]
[261 188]
[135 161]
[241 177]
[160 198]
[200 232]
[180 200]
[250 197]
[123 200]
```

FEYNN LABS
Experiment with your Knowledge

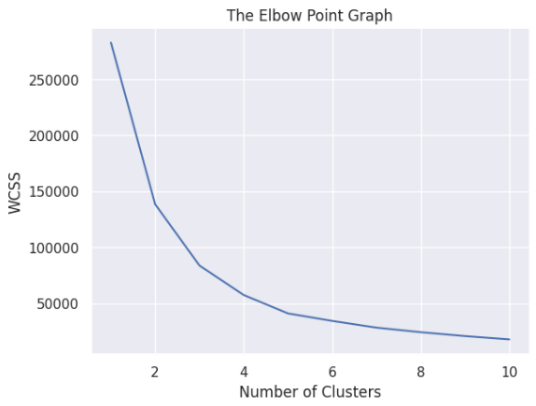
K.SAI ROHITH REDDY

APPLYING K-MEANS ALGORITHM

MODEL FITTING

```
# finding wcss value for different number of clusters
from sklearn.cluster import KMeans
wcss = []
for i in range(1,11):
  kmeans = KMeans(n_clusters=i, init='k-means++', random_state=0)
  kmeans.fit(X)
  wcss.append(kmeans.inertia_)
```

```
sns.set()
plt.plot(range(1,11), wcss)
plt.title('The Elbow Point Graph')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.show()
```




```
kmeans = KMeans(n_clusters=3, init='k-means++', random_state=0)

# return a label for each data point based on their cluster
Y = kmeans.fit_predict(X)

print(Y)
```

```
[2 0 2 1 0 2 0 0 2 0 1 0 0 0 0 0 2 0 0 0 0 2 0 2 2 0 0 1 0 0 1 0 0 1 0 0 0
 0 0 0 2 0 1 0 0 0 0 2 0 0 1 2 0 0 2 0 0 0 0 2 0 2 0 1 1 2 0 1 0 1 0 0 2 1
 0 0 0 0 1 2 0 2 0 0 1 0 1 1 0 0 1 0 0 1 0 0 0 1 0 1 1 1 1]
/usr/local/lib/python3.10/dist-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 war
  warnings.warn(
```

```
plt.figure(figsize=(8,8))
plt.scatter(X[Y==0,0], X[Y==0,1], s=50, c='green', label='Cluster 1')
plt.scatter(X[Y==1,0], X[Y==1,1], s=50, c='red', label='Cluster 2')
plt.scatter(X[Y==2,0], X[Y==2,1], s=50, c='yellow', label='Cluster 3')
# plot the centroids
plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], s=100, c='cyan', label='Centroids')
plt.title('EV MARKET FOR CARS')
plt.xlabel('TopSpeed_KmH')
plt.ylabel('Efficiency_WhKm')
plt.show()
```

```
[14] kmeans.inertia_

83664.6347681623
```

APPLYING PRINCIPAL COMPONENT ANALYSIS

Considering n_components as 2(PC1,PC2)

APPLYING PRINCIPLE COMPONENT ANALYSIS

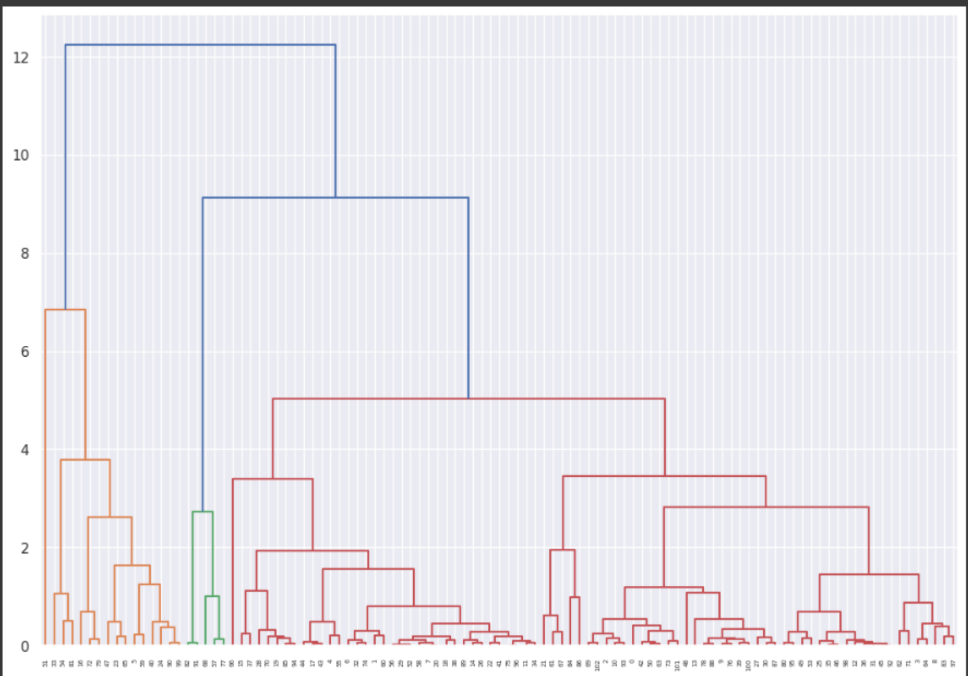
```
[16] features = ['AccelSec','TopSpeed_KmH','Efficiency_WhKm', 'RapidCharge','Range_Km', 'Seats','PowerTrain',"PriceEuro"]
from sklearn.preprocessing import StandardScaler
# Separating out the features
scale = df.loc[:, features].values
scale= StandardScaler().fit_transform(scale)
```

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
t = pca.fit_transform(scale)
data2 = pd.DataFrame(t, columns=['PC1', 'PC2'])
data2
```

	PC1	PC2
0	1.511285	0.211975
1	-1.740644	-0.582778
2	1.292997	0.020945
3	0.021276	-0.115449
4	-2.327960	0.244931
...	...	...
98	-0.338325	-0.462721
99	2.279502	0.230222
100	0.815126	-0.164293
101	1.617635	-0.089687
102	1.277082	-0.221299

103 rows × 2 columns

```
from scipy.cluster.hierarchy import dendrogram, linkage
linked = linkage(data2, 'complete')
plt.figure(figsize=(13, 9))
dendrogram(linked, orientation='top')
plt.show()
```



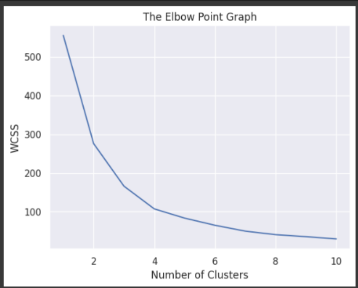
APPLYING K-MEANS ALGORITHM FOR MODEL-2

```
wcss1 = []

for i in range(1,11):
  kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
  kmeans.fit(t)
  wcss1.append(kmeans.inertia_)
```

MODEL FITTING

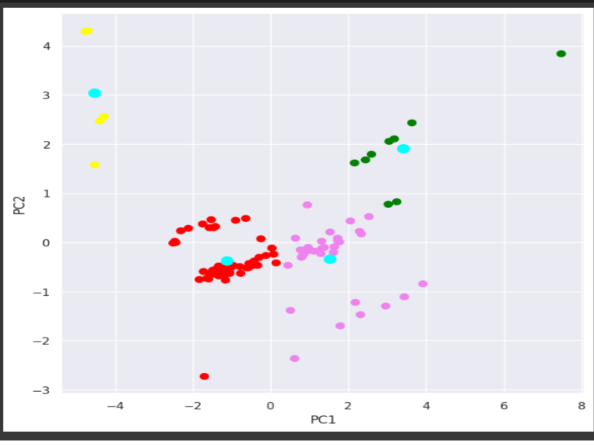
```
# plot an elbow graph
sns.set()
plt.plot(range(1,11), wcss1)
plt.title('The Elbow Point Graph')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.show()
```




```
kmeans = KMeans(n_clusters=4, init='k-means++', random_state=0)
# return a label for each data point based on their cluster
Y1= kmeans.fit_predict(t)
print(Y1)
```

```
[3 1 3 1 1 0 1 1 1 3 3 1 1 3 1 1 0 1 1 1 1 3 1 0 3 1 1 3 1 1 3 1 1 3 1 1 1
 1 1 3 3 1 3 1 1 1 1 0 3 1 3 0 1 1 3 1 1 2 1 0 1 3 1 3 1 0 1 3 2 3 1 3 0 3
 1 1 3 2 3 0 1 3 2 1 3 1 3 3 3 1 3 2 1 3 1 1 1 1 1 1 3 3 3 3]
/usr/local/lib/python3.10/dist-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
```

```
plt.figure(figsize=(8,8))
plt.scatter(t[Y1==0,0], t[Y1==0,1], s=50, c='green', label='Cluster 1')
plt.scatter(t[Y1==1,0], t[Y1==1,1], s=50, c='red', label='Cluster 2')
plt.scatter(t[Y1==2,0], t[Y1==2,1], s=50, c='yellow', label='Cluster 3')
plt.scatter(t[Y1==3,0], t[Y1==3,1], s=50, c='violet', label='Cluster 4')
# plot the centroids
plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], s=100, c='cyan', label='Centroids')
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.show()
```




```
kmeans.inertia_
```

```
107.08971916002179
```

CHECKING THE K-MEANS ALGORITHM FOR DIFFERENT ATTRIBUTES

Considering the Segment and Price Euro attributes

CHECKING THE KMEANS ACCURACY BY CONSIDERING DIFFERENT ATTRIBUTES

```
df.sample(2)
```

	Brand	Model	AccelSec	TopSpeed_KmH	Range_Km	Efficiency_WhKm	FastCharge_KmH	RapidCharge	PowerTrain	PlugType	BodyStyle	Segment	Seats	PriceEuro
1	Volkswagen	ID.3 Pure	10.0	160	270	167	250	1	2	2	1	2	5	30000
61	Tesla	Model Y Long Range Performance	3.7	241	410	177	900	1	0	2	6	3	7	65620

```
x3=df.iloc[:,[11,13]].values
x3
```

```
array([[     3,  55480],
       [     2,  30000],
       [     3,  56440],
       [     3,  68040],
       [     1,  32997],
       [     5, 105000],
       [     2,  31900],
       [     1,  29682],
       [     3,  46380],
       [     3,  55000],
```

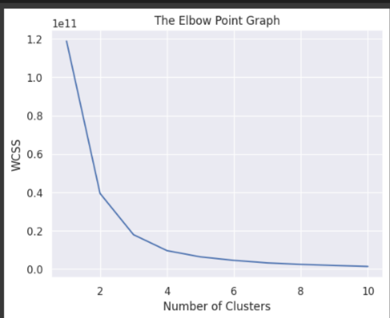
APPLYING KMEANS ALGORITHM FOR MODEL-3

CHOOSING THE NUMBER OF CLUSTERS

```
wcss3 = []
for i in range(1,11):
  kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
  kmeans.fit(x3)
  wcss3.append(kmeans.inertia_)
```

MODEL FITTING

```
sns.set()
plt.plot(range(1,11), wcss3)
plt.title('The Elbow Point Graph')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.show()
```



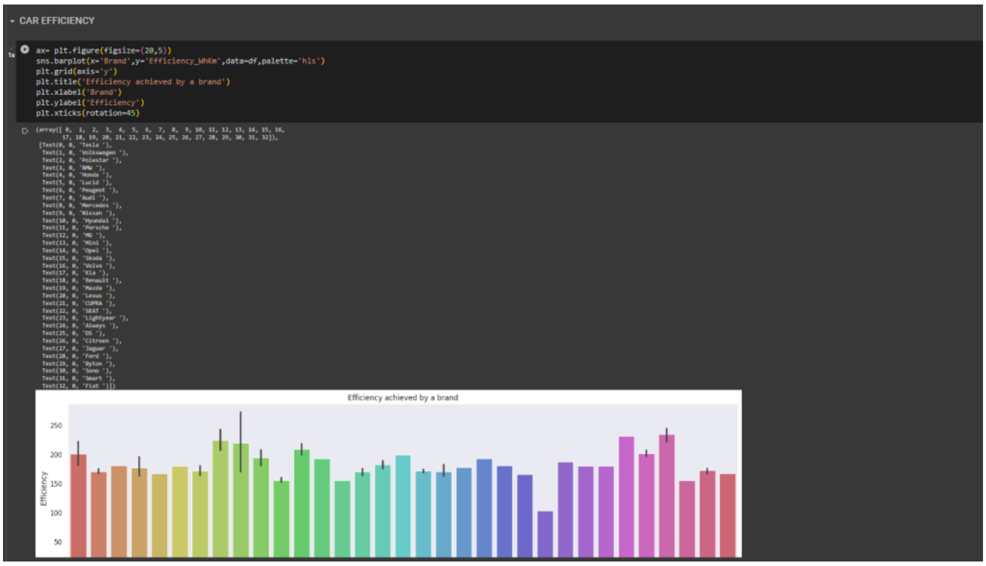

```
kmeans = KMeans(n_clusters=4, init='k-means++', random_state=0)
# return a label for each data point based on their cluster
Y3 = kmeans.fit_predict(x3)

print(Y3)
```

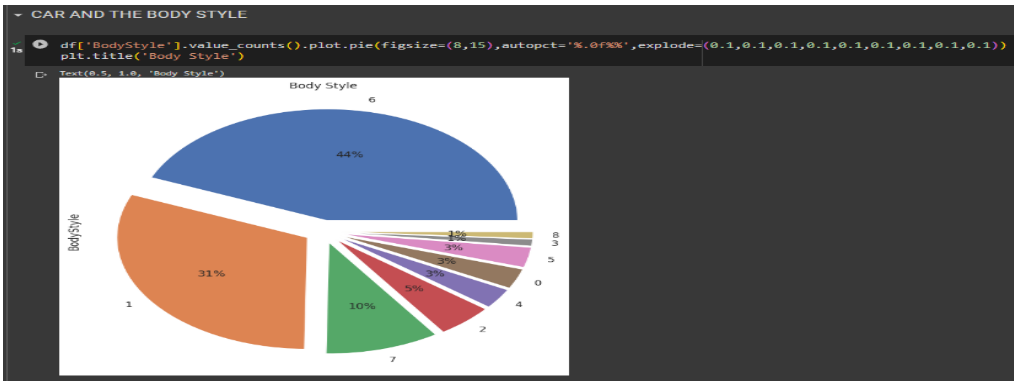
```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
[0 2 0 0 2 3 2 2 2 0 0 2 2 0 2 2 1 2 2 2 2 0 2 3 0 2 2 0 2 2 0 2 2 0 2 2 0
 2 2 2 0 2 0 2 2 2 2 3 1 2 0 1 2 2 3 2 2 2 2 3 2 0 0 0 0 3 2 0 2 0 2 2 1 0
 2 2 0 2 0 1 2 3 2 2 0 2 2 0 2 2 3 2 2 0 2 2 2 0 2 3 0 0 0]
```

VISUALIZATIONS USING SEABORN LIBRARY

Car Efficiency



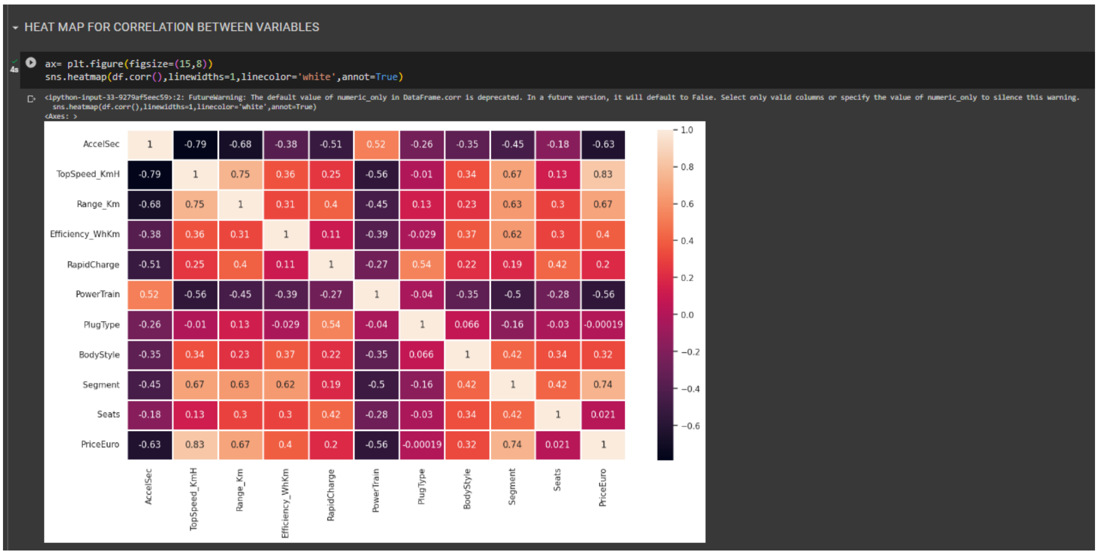
Car and the BodyStyle



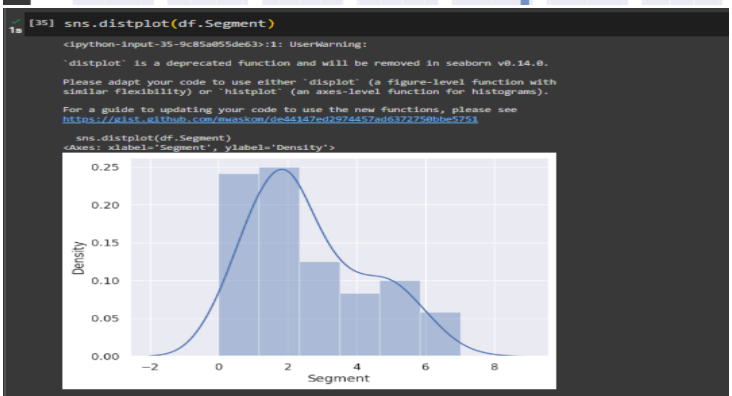
FEYNN LABS
Experiment with your Knowledge

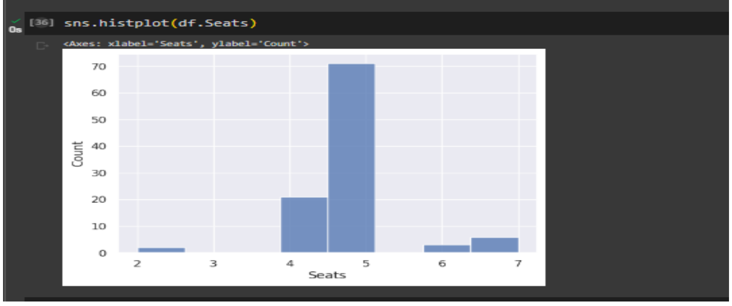
K.SAI ROHITH REDDY

Heat map for corr between variables



Pair plot





QUESTIONS:

Which vehicle has max range (km) under 50,000 Euros?

+ Code + Text

QUESTIONS:

Which vehicle has max range (km) under 50,000 Euros?

```
[38] df['FullName'] = df['Brand'] + '-' + df['Model']
     df_1 = df.loc[df['PriceEuro'] <= 50000]
     df_2 = df.loc[df['PriceEuro'] > 50000]
     t1 = 'Less than 50,000 Euros'
     t2 = 'More than 50,000 Euros'
```

```
pd.set_option('display.max_columns', None)
top_range_1 = df.sort_values(by= 'Range_Km', ascending= False)
print(top_range_1[['FullName', 'Range_Km' ,'PriceEuro', 'RapidCharge']])
```

```
                      FullName  Range_Km  PriceEuro  RapidCharge
51             Tesla -Roadster       970     215000            1
33  Tesla -Cybertruck Tri Motor      750      75000            1
5                   Lucid -Air       610     105000            1
48             Lightyear -One        575     149000            1
40   Tesla -Model S Long Range       515      79990            1
..                         ...       ...        ...          ...
68   Renault -Kangoo Maxi ZE 33      160      38000            0
57          Renault -Twingo ZE       130      24790            0
82      Smart -EQ fortwo coupe       100      21387            0
77          Smart -EQ forfour         95      22030            0
91      Smart -EQ fortwo cabrio       95      24565            0

[103 rows x 4 columns]
```

Which vehicle has max range (Km) costing more than 50,000 Euros?

Which vehicle has max range(Km) costing more than 50,000 Euros?

```
pd.set_option('display.max_columns', None)
top_range_2 = df_2.sort_values(by= 'Range_Km', ascending= False)
print(top_range_2[['FullName', 'Range_Km' ,'PriceEuro', 'RapidCharge']])
```

```
                                  FullName  Range_Km  PriceEuro  RapidCharge
51                         Tesla -Roadster       970     215000            1
33             Tesla -Cybertruck Tri Motor       750      75000            1
5                               Lucid -Air       610     105000            1
48                         Lightyear -One        575     149000            1
40               Tesla -Model S Long Range       515      79990            1
59              Tesla -Model S Performance       505      96990            1
67            Tesla -Cybertruck Dual Motor       460      55000            1
64              Ford -Mustang Mach-E ER RWD      450      54475            1
54               Tesla -Model X Long Range       450      85990            1
0      Tesla -Model 3 Long Range Dual Motor      450      55480            1
13                                BMW -i4        450      65000            1
81              Tesla -Model X Performance       440     102990            1
24     Tesla -Model 3 Long Range Performance     435      61480            1
69              Ford -Mustang Mach-E ER AWD      430      62900            1
23                          Audi -e-tron GT      425     125000            1
21     Tesla -Model Y Long Range Dual Motor      425      58620            1
65                  Porsche -Taycan 4S Plus      425     109302            1
93              Nissan -Ariya e-4ORCE 87kWh      420      57500            1
61     Tesla -Model Y Long Range Performance     410      65620            1
76                Audi -Q4 Sportback e-tron      410      57500            1
9                          Audi -Q4 e-tron       400      55000            1
102                Byton -M-Byte 95 kWh 2WD      400      62000            1
2                            Polestar -2         400      56440            1
73                 Byton -M-Byte 95 kWh 4WD      390      64000            1
79                    Porsche -Taycan Turbo      390     148301            1
72             Porsche -Taycan Cross Turismo     385     150000            1
42             Audi -e-tron Sportback 55 quattro     380      81639            1
16                  Porsche -Taycan Turbo S      375     180781            1
101  Nissan -Ariya e-4ORCE 87kWh Performance     375      65000            1
27               Volvo -XC40 P8 AWD Recharge     375      60437            1
10                Mercedes -EQC 400 4MATIC      370      69484            1
63                           Jaguar -I-Pace      365      75351            1
50                  Audi -e-tron 55 quattro      365      79445            1
47                        Porsche -Taycan 4S     365     102945            1
3                                 BMW -iX3       360      68040            1
78              Ford -Mustang Mach-E SR AWD      340      54000            1
99        Audi -e-tron S Sportback 55 quattro    335      96050            1
84                    Mercedes -EQV 300 Long     330      70631            1
97                 Byton -M-Byte 72 kWh 2WD      325      53500            1
90                  Audi -e-tron S 55 quattro    320      93800            1
87            Audi -e-tron Sportback 50 quattro  295      69551            1
30                  Audi -e-tron 50 quattro      280      67358            1
```

Vehicles with best acceleration under 50,000 Euros?

Vehicles with best acceleration under 50,000 Euros?

```
[41] pd.set_option('display.max_columns', None)
     acceleration_1 = df_1.sort_values(by= 'AccelSec')
     print(acceleration_1[['FullName','AccelSec', 'Range_Km', 'PowerTrain' ,'PriceEuro']])
```

```
                          FullName  AccelSec  Range_Km  PowerTrain  \
39                   Mercedes -EQA       5.0       350           0
8    Tesla -Model 3 Standard Range Plus   5.6       310           2
100        Nissan -Ariya e-4ORCE 63kWh    5.9       325           0
88                 Skoda -Enyaq iV vRS     6.2       400           0
37                     CUPRA -el-Born      6.5       425           2
..                                 ...       ...       ...         ...
43                  Skoda -CITIGOe iV     12.3       195           1
57                 Renault -Twingo ZE     12.6       130           2
77                  Smart -EQ forfour     12.7        95           2
66            Nissan -e-NV200 Evalia      14.0       190           1
68         Renault -Kangoo Maxi ZE 33     22.4       160           1

     PriceEuro
39       45000
8        46380
100      50000
88       47500
37       45000
..         ...
43       24534
57       24790
77       22030
66       33246
68       38000

[61 rows x 5 columns]
```

GITHUB LINK: https://github.com/rohithreddy999/Feynn-lab/blob/main/ev_cars_FEYNN.ipynb