ELECTRONIC VEHICLE MARKET

NEHA DWIVEDI

Abstract

Market segmentation becomes a crucial tool for evolving transportation technology such as electric vehicles (EVs) in emerging markets to explore and implement for extensive adoption. EVs adoption is expected to grow phenomenally in near future as low emission and low operating cost vehicle, and thus, it drives a considerable amount of forthcoming academic research curiosity. The main aim of this study is to explore and identify distinct sets of potential buyer segments for EVs based on psychographic, behavioral, and socio-economic characterization by employing an integrated research framework of 'perceived benefits-attitude-intention'. The study applied robust analytical procedures including cluster analysis, multiple discriminant analysis and Chi-square test to operationalize and validate segments from the data collected of 563 respondents using a cross-sectional online survey. The findings posit that the three distinct sets of young consumer groups have been identified and labelled as 'Conservatives', 'Indifferents', and 'Enthusiasts' which are deemed to be buddying EV buyers The implications are recommended, which may offer some pertinent guidance for scholars and policymakers to encourage EVs adoption in the backdrop of emerging sustainable transport market.

Keywords : Electric vehicles, Market segmentation, Cluster analysis, Attitude towards electric vehicles

INTRODUCTION:

Segmenting for Electric Vehicle Market

The market segmentation approach aims at defining actionable, manageable, homogenous subgroups of individual customers to whom the marketers can target with a similar set of marketing strategies. In practice, there are two ways of segmenting the market-a-priori and post-hoc. An a-priori approach utilizes predefined characteristics such as age, gender, income, education, etc. to predefine the segments followed by profiling based on a host of measured variables (behavioral, psychographic or benefit). In the post-hoc approach to segmentation on other hand, the segments are identified based on the relationship among the multiple measured variables. The commonality between both approaches lies in the fact that the measured variables determine the 'segmentation theme'. The present study utilizes an a-priori approach to segmentation so as to divide the potential EV customers into sub-groups.

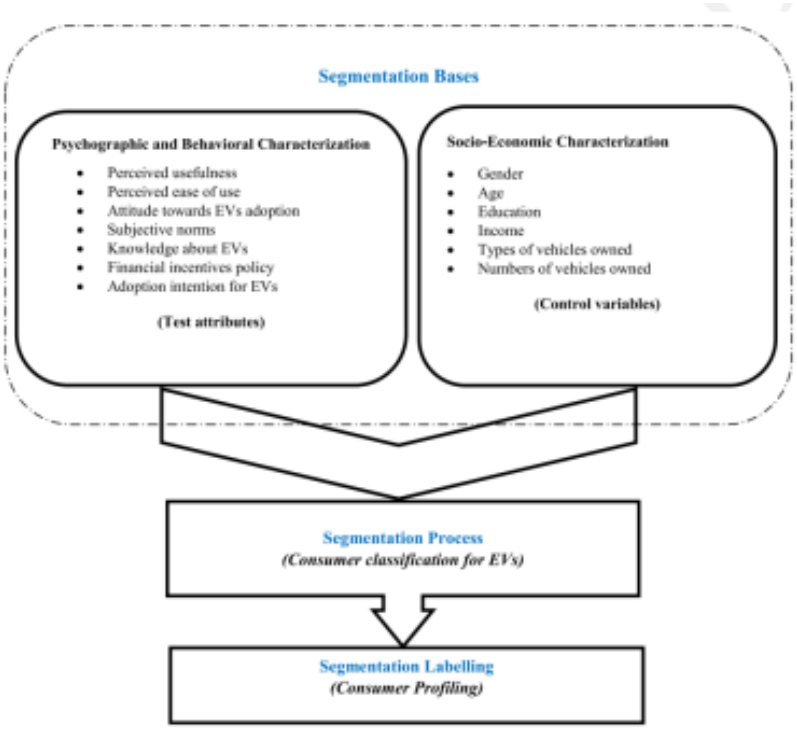


Fig: Market Segmentation Electric Vehicles

It is argued that the blended approach of psychographic and socioeconomic attributes for market segmentation enables the formulation of sub-market strategies which in turn satisfy the specific tastes and preferences of the consumer groups.

Straughan and Roberts presented a comparison between the usefulness of psychographic, demographic, and economic characteristics based on consumer evaluation for eco-friendly products.They pinpointed the perceived superiority of the psychographic characteristics over the socio-demographic and economic ones in explaining the environmentally-conscious consumer behavior and thus, the study recommended the use of psychographic characteristics in profiling the consumer segments in the market for eco-friendly products. The present study adds perceived-benefit characteristics guided by blended psychographic and socio-economic aspects for segmenting the consumer market

PROBLEM STATEMENT:

Electronic Vehical Market

Data Sources:

Importing the Dataset: We will import the dataset that we need to use. So here, we are using the Car Rental Data. It can be imported using the below code
df=pd.read_csv("ElectricCarData_Clean.csv")

Data Pre-processing: (steps and libraries used)

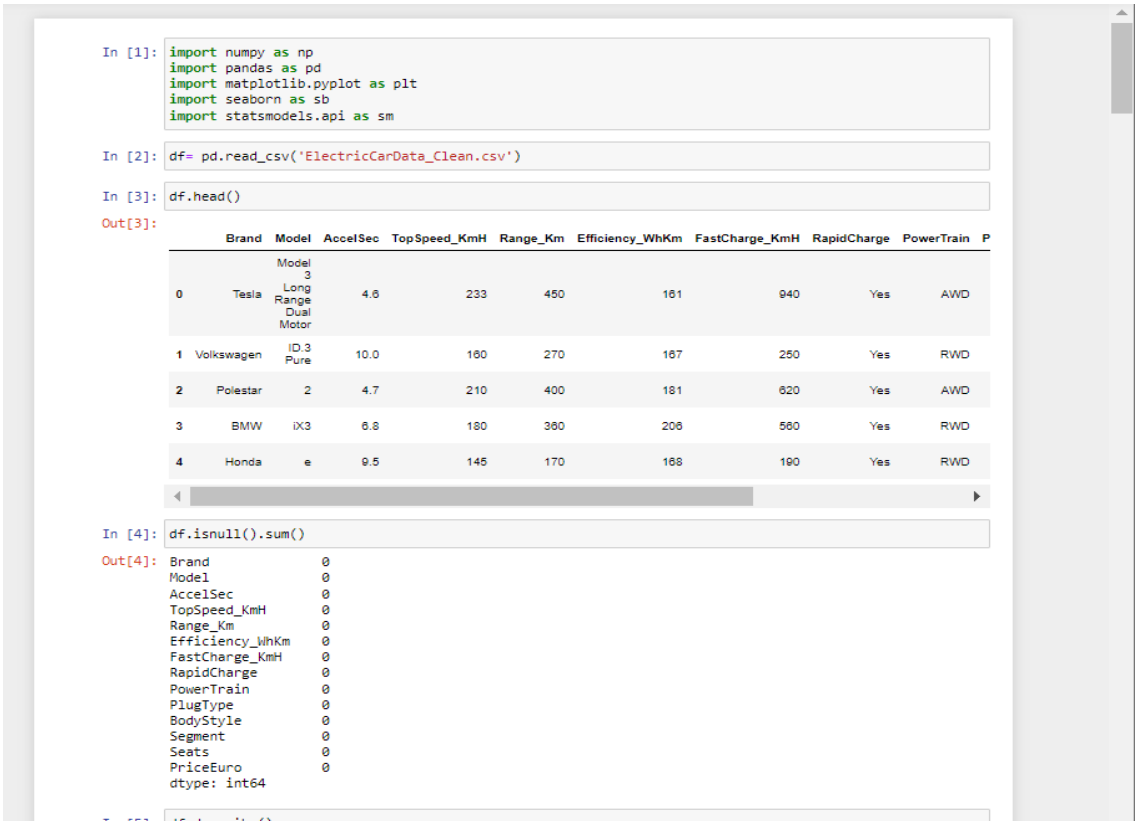
Importing Libraries:
We will import the libraries for our model, which is part of data pre-processing. The code is given below:

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

- Numpy we have imported for the performing mathematics calculation.
- Matplotlib is for plotting the graph, and pandas are for managing the dataset.
- Seaborn is for data visualization library, it is based on matplotlib.

Exploratory Data Analysis:

Exploratory data analysis, or EDA, is a detailed analysis intended to reveal a data set's underlying structure. It is significant for a business because it reveals trends, patterns, and linkages that are not immediately obvious.



```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sb
        import statsmodels.api as sm

In [2]: df= pd.read_csv('ElectricCarData_Clean.csv')

In [3]: df.head()

Out[3]:
```

	Brand	Model	AccelSec	TopSpeed_KmH	Range_Km	Efficiency_WhKm	FastCharge_KmH	RapidCharge	PowerTrain	P
0	Tesla	Model 3 Long Range Dual Motor	4.6	233	450	161	940	Yes	AWD	
1	Volkswagen	ID.3 Pure	10.0	160	270	167	250	Yes	RWD	
2	Polestar	2	4.7	210	400	181	620	Yes	AWD	
3	BMW	iX3	6.8	180	360	206	560	Yes	RWD	
4	Honda	e	9.5	145	170	168	190	Yes	RWD	

```
In [4]: df.isnull().sum()

Out[4]: 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
```

```
In [5]: df.describe()
```

Out[5]:

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

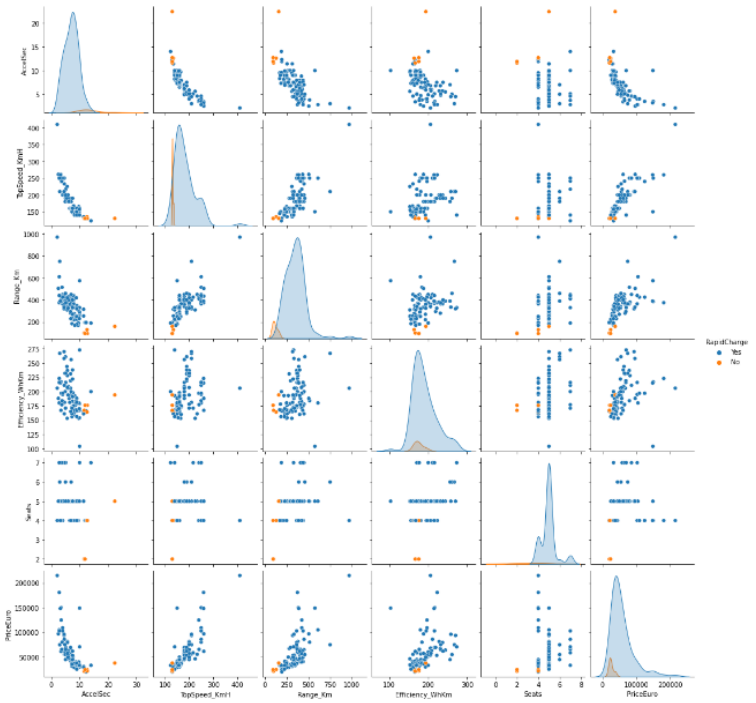
```
In [6]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 103 entries, 0 to 102
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Brand            103 non-null    object
 1   Model            103 non-null    object
 2   AccelSec         103 non-null    float64
 3   TopSpeed_KmH     103 non-null    int64
 4   Range_Km         103 non-null    int64
 5   Efficiency_WhKm  103 non-null    int64
 6   FastCharge_KmH   103 non-null    object
 7   RapidCharge      103 non-null    object
 8   PowerTrain       103 non-null    object
 9   PlugType         103 non-null    object
 10  BodyStyle        103 non-null    object
 11  Segment          103 non-null    object
 12  Seats            103 non-null    int64
 13  PriceEuro        103 non-null    int64
dtypes: float64(1), int64(5), object(8)
memory usage: 11.4+ KB
```

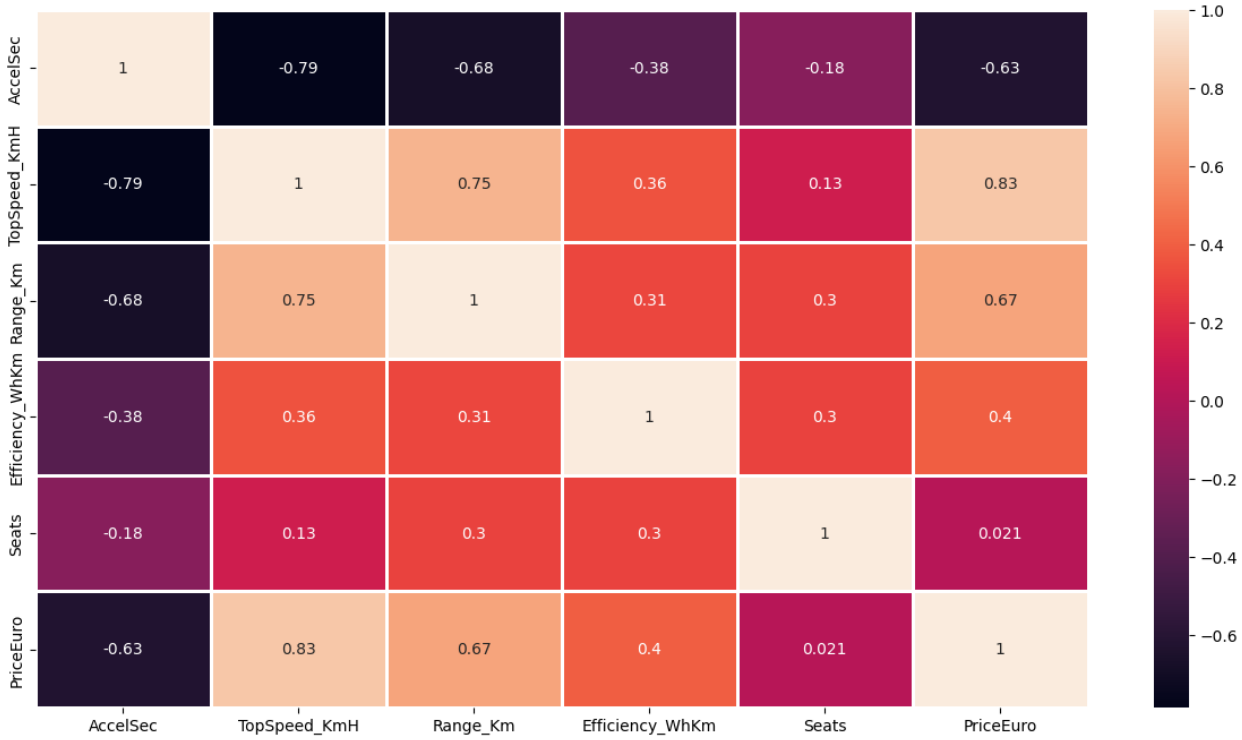
Pairplot of all the columns based on Rapid Charger presence

```
In [7]: sb.pairplot(df,hue='RapidCharge')
```

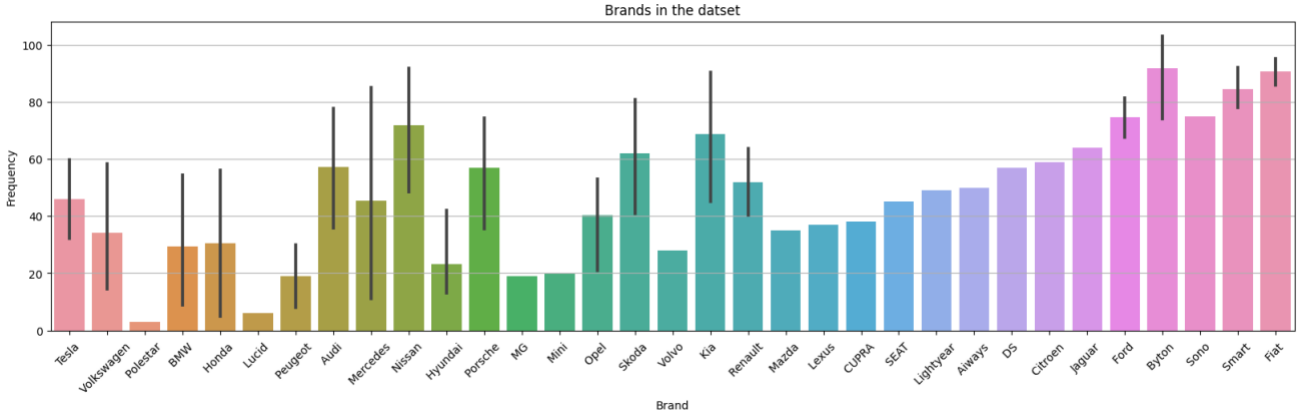
Out[7]: <seaborn.axisgrid.PairGrid at 0x1607f2879a0>



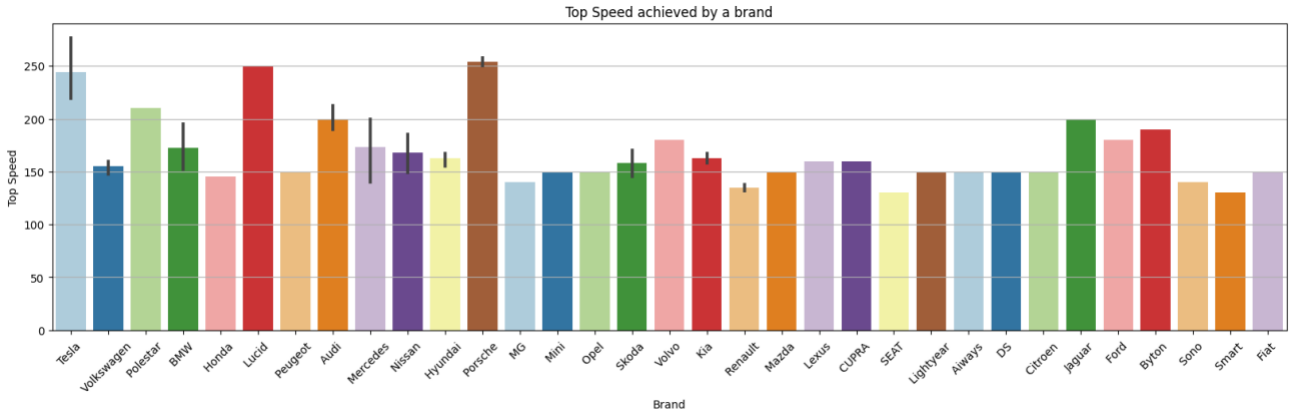
Correlation Matrix: A correlation matrix is simply a table that displays the correlation. It is best used in variables that demonstrate a linear relationship between
each other. Coefficients for different variables. The matrix depicts the correlation between all the possible pairs of values through the heatmap in the below figure. The
relationship between two variables is usually considered strong when their correlation coefficient value is larger than 0.7



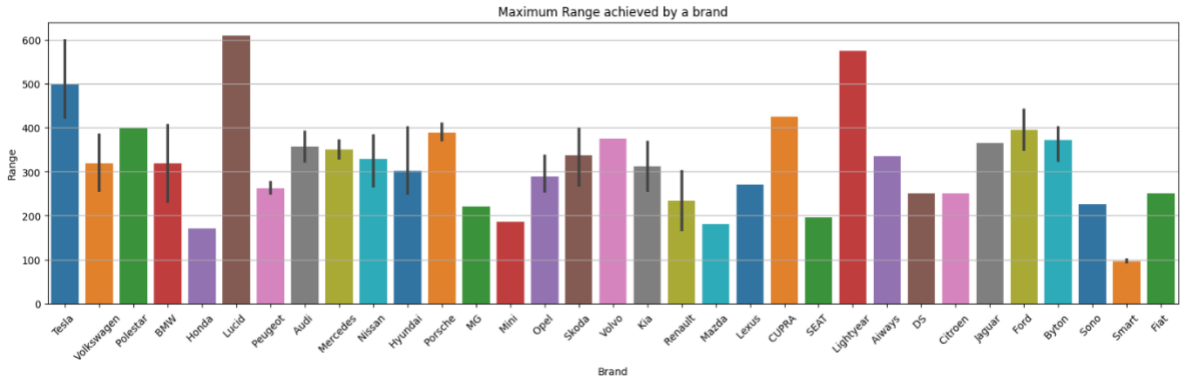
Frequency of the Brands in the dataset



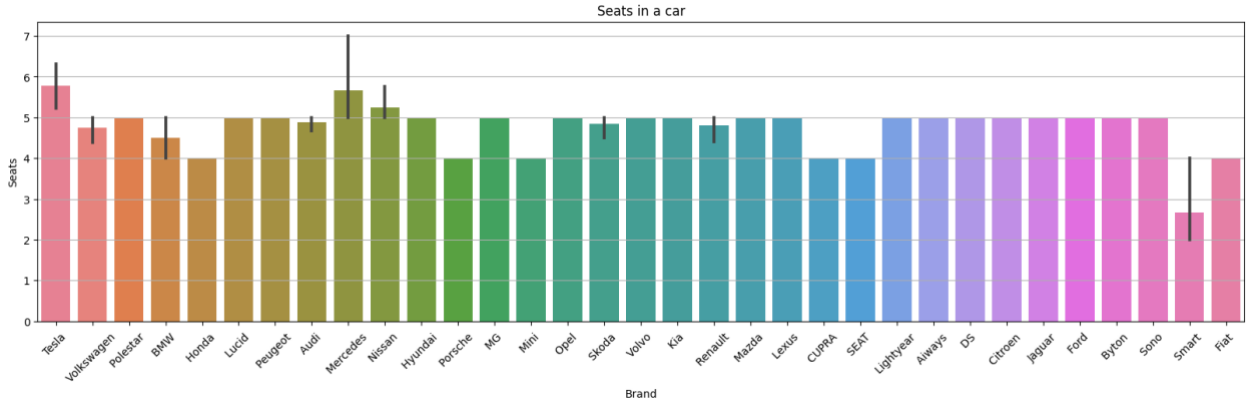
Top speeds achieved by the cars of a brand



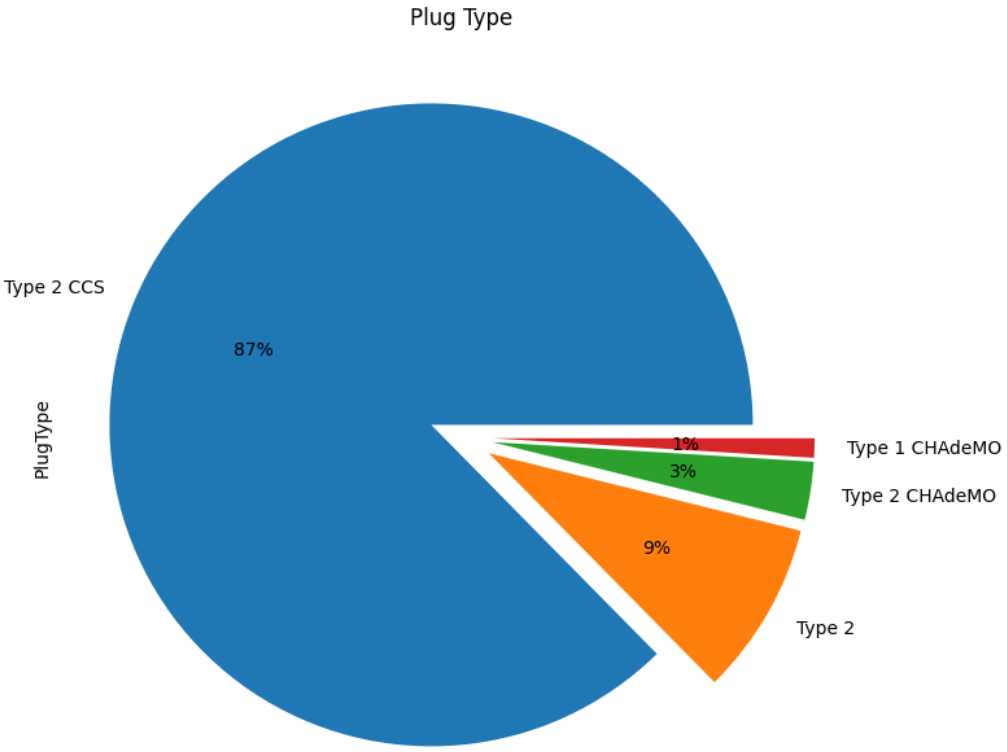
Range a car can achieve



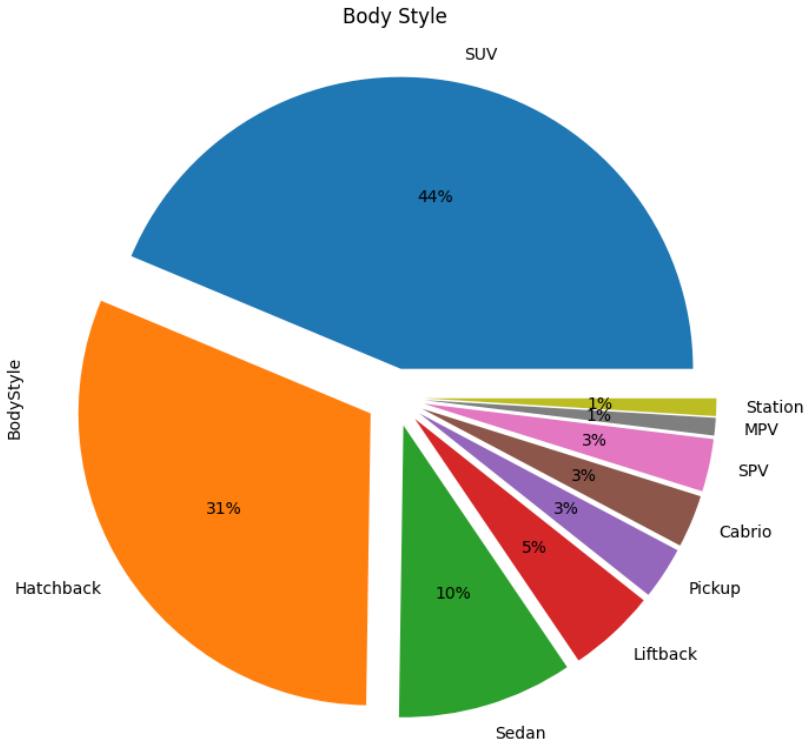
Number of seats in each car



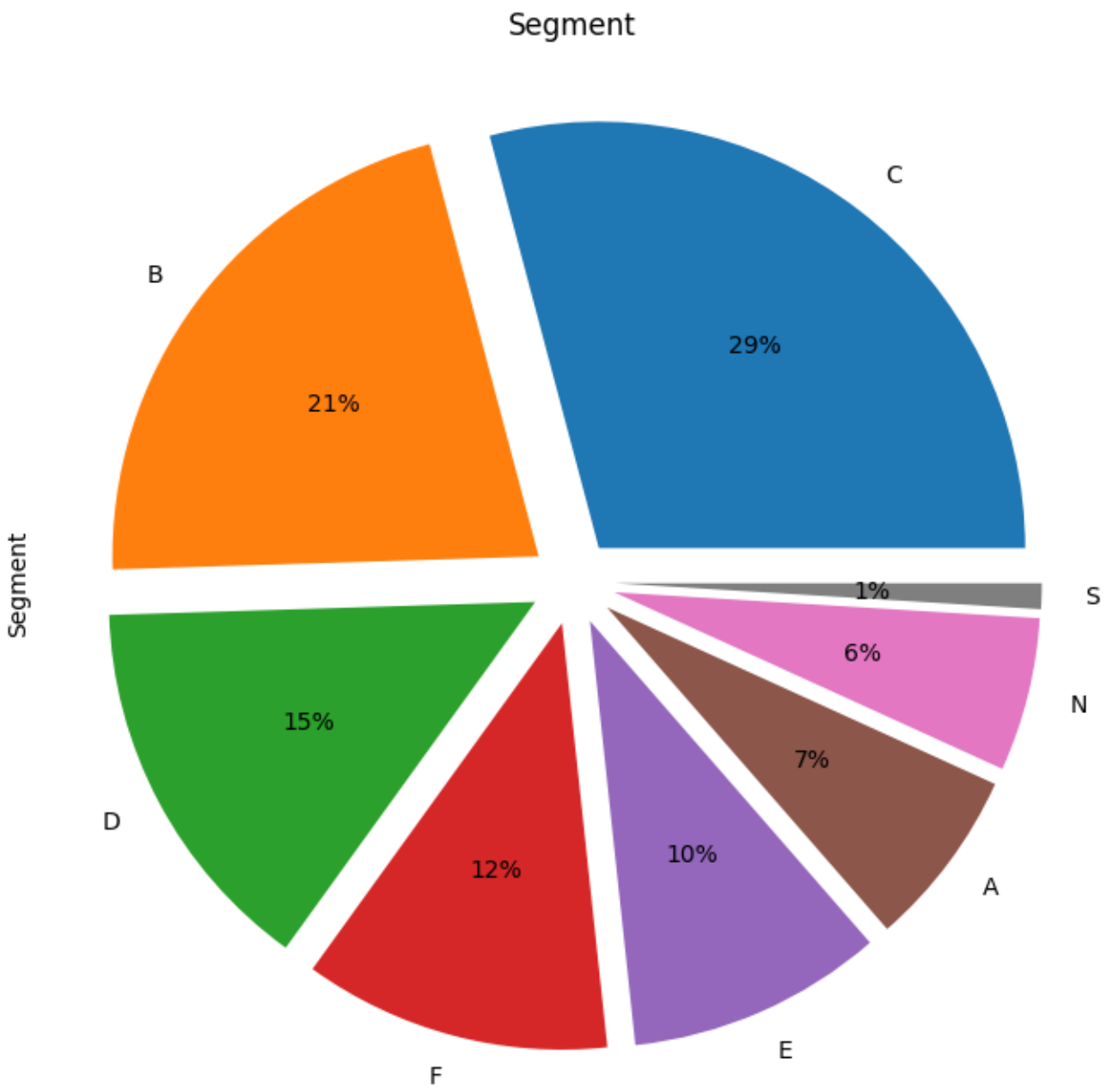
Type of Plug used for charging



Cars and their body style



Segment in which the cars fall under



Regression

In [17]:
```
x=df[['AccelSec','Range_Km','TopSpeed_KmH','Efficiency_WhKm']]
y=df['PriceEuro']
```

In [18]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
lr= LinearRegression()
```

In [19]:
```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3,random_state=365)
```

In [20]:
```
lr.fit(X_train, y_train)
pred = lr.predict(X_test)
```

In [21]:
```
#Finding out the R-squared value

from sklearn.metrics import r2_score
r2=(r2_score(y_test,pred))
print(r2*100)
```

```
78.35225979903608
```

In [22]:
```
#Putting Yes value as 1 and No value as 0 for Logistic Regression

df['RapidCharge'].replace(to_replace=['No','Yes'],value=[0, 1],inplace=True)
```

In [23]:
```
y1=df[['RapidCharge']]
x1=df[['PriceEuro']]
```

In [24]:
```
from sklearn.model_selection import train_test_split
X1_train, X1_test, y1_train, y1_test = train_test_split(x1, y1, test_size=0.2,random_state=365)
```

In [25]:
```
#Importing Logistic Regression

from sklearn.linear_model import LogisticRegression
```

In [26]:
```
log= LogisticRegression()
```

In [27]:
```
log.fit(X1_train, y1_train)
pred1 = log.predict(X1_test)
pred1
```

```
C:\Users\rajen\anaconda3\lib\site-packages\sklearn\utils\validation.py:993: DataConversionWarning: A
column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,
), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[27]:
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
      dtype=int64)
```

Linear Regression:

The first part of the code selects specific columns from a DataFrame df and assigns them to the variable x. It selects columns 'AccelSec', 'Range_Km', 'TopSpeed_KmH', and 'Efficiency_WhKm'.
The target variable 'PriceEuro' is assigned to the variable y.
The code then imports necessary libraries for linear regression and creates a LinearRegression object named lr.

The data is split into training and testing sets using the train_test_split function from sklearn.
The split is done with a test size of 30% and a random state of 365.
The lr model is fitted using the training data (X_train, y_train).
Predictions are made on the test data (X_test) using the predict method of lr.
The code calculates the R-squared value, which is a measure of how well the linear regression model fits the data. It uses the r2_score function from sklearn.metrics.
Finally, the R-squared value is printed.

Logistic Regression:

The second part of the code focuses on logistic regression.
The column 'RapidCharge' in the DataFrame df is modified to replace 'No' with 0 and 'Yes' with 1, and the changes are made in-place.
The feature variable 'PriceEuro' is assigned to x1, and the target variable 'RapidCharge' is assigned to y1.
The data is split into training and testing sets using the train_test_split function with a test size of 20% and a random state of 365.
Logistic regression is imported from sklearn, and a LogisticRegression object named log is created.
The log model is fitted using the training data (X1_train, y1_train).
Predictions are made on the test data (X1_test) using the predict method of log.
The code prints the predictions (pred1) from logistic regression.

All the elements of the marketing mix influence each other. They make up the business plan for a company and handle it right, and can give it great success. The marketing mix needs a lot of understanding, market research and consultation with several people, from users to trade to manufacturing and several others.

GITHUB:

https://github.com/NehaDwivedi842/Feynnlab_internship