**Analysis of EVs Market and Predicting EVs car Price**

**Student Name:**

**Institution:**

**Course:**

**Instructor’s Name:**

**Assignment Due Date**

**Table of contents**

[**Executive Summary 3**](#_Toc130665441)

[**Case Study Overview 4**](#_Toc130665442)

[**Summary of key Points 5**](#_Toc130665443)

[**Demonstration of Analytics 5**](#_Toc130665444) **to 38**

[**Recommendation 36**](#_Toc130665445)

# 

# Executive Summary

1. The growing demand for electric vehicles is gasoline by the desire to shift to renewable energy sources for transportation and reduce greenhouse gas emissions, which are a major contributor to climate change

1. It is crucial for companies to conduct extensive research and analysis to gain a deep understanding of the market dynamics and consumer needs as electric vehicle market is relatively new
2. pricing of electric cars is directly influenced by the level of features they offer, with higher quality features commanding higher prices in the market
3. As the market for electric vehicles continues to grow, there will be an increasing demand for charging stations to support the charging of these vehicles. Companies like Audi can take advantage of this opportunity by investing in the development and installation of charging infrastructure, which can help to promote the widespread adoption of electric vehicles
4. In addition to the environmental benefits of reducing greenhouse gas emissions, companies like Audi can also focus on implementing sustainable and environmentally-friendly manufacturing processes in the production of their electric vehicles
5. Collaboration and partnerships with other players in the industry can also be beneficial for companies like Audi. By working together with battery manufacturers, charging infrastructure providers, and renewable energy companies, companies can leverage each other's strengths and create synergies that can drive innovation and growth in the industry

# 

# Case Study Overview

* Climate change is a significant issue in society. The UK government has mandated that all new vehicles produced by 2035 should be electric, with a further target of having all cars on UK roads be electric by 2050
* Regulations have been introduced to reduce the impact of transportation on the environment
* The transition to electric vehicles has created new demands in market
* Audi AG has assigned task to conduct an analysis of data related to the electric vehicle industry to inform the company's decision-making and reduce market risks associated with a new production line
* Transport sector is a major contributor to greenhouse gas (GHG) emissions, responsible for 25% of total GHG emissions.
* This study will provide insights into the industry, and based on these insights, recommendations will be made to Audi AG.
* Climate can be also indirectly affected by the EV, when we use energy from Renewable resources, as they produce significantly less greenhouse gas emissions compared to non-renewable sources
* We will review the given dataset of 103 Electric Vehicles with 14 features and build an ML model to estimate the EV’s Price in Euro

# 

# Summary of key Points

The given dataset has 103 Electric Vehicle features of different electric vehicle models.

* **Brand** - Manufacturer of the vehicle
* **Model** - Model name
* **AccelSec** - Acceleration, in seconds, to go from 0 to 100 km/h
* **TopSpeed\_KmH** - The top speed in km/h
* **Range\_Km** - How far the vehicle can travel in a single, full charge, in km
* **Efficiency\_WhKm** - How efficient the vehicle is at using energy, specifically how many watt-hours (Wh) of battery charge it takes to travel a kilometre (km). Units are Wh/km.
* **FastCharge\_KmH** - If the vehicle has RapidCharge, how quickly the battery charges (in terms of distance that the battery will power the vehicle), in km/h.
* **RapidCharge** - Whether or not the vehicle has RapidCharge.
* **PowerTrain** - Front, rear, or all wheel drive.
* **PlugType** - Type of plug.
* **BodyStyle** - Basic size or style of vehicle.
* **Segment** - Market segment (in other words, the sorts of people that the vehicle is designed to be sold to)
* **Seats** - Number of seats
* **PriceEuro** - Price of the vehicle in Germany before tax incentives.

**Key Insights:-**

1. We have 103 electrical vehicles comprises of different model and brand
2. 95% vehicle support Rapid Charge
3. Vehicle with 5 seating is more preferred – 69% vehicles are of 5 seating
4. Average price of Audi is slightly on higher side when compared with Tesla
5. Rapid Charge vehicles are 50% higher on price when compared with Non rapid charge Vehicles
6. Average price for AWD vehicles are on higher side
7. We have 5 seating vehicles are more in percentage because of its lower price when compared with 4 seating
8. Key Feature that drives the Electric Vehicle Price
   1. Acceleration Per Sec
   2. Segment
   3. Top Speed

**Demonstration of Analytics**

# Import Required Packages

In [453…

**import** warnings

warnings**.**filterwarnings("ignore")

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

**import** numpy **as** np

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

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

pd**.**set\_option('display.max\_columns',**None**)

**%matplotlib** inline

# Load Data

In [454…

ev\_car\_df **=** pd**.**read\_csv("eleccarsfin.csv")

print(f"Total Number of Records = {ev\_car\_df**.**shape[0]} and Columns = {ev\_car\_df**.**sh print("First 5 records of the dataframe:")

ev\_car\_df**.**head(5)

Out[454]:

Total Number of Records = 103 and Columns = 14 First 5 records of the dataframe:

**Brand Model AccelSec TopSpeed\_KmH Range\_Km Efficiency\_WhKm FastCharge\_KmH**

Model

3

Long

**0** Tesla

Range Dual Motor

4.6 233 450 161 940

**1** Volkswagen

ID.3

Pure

10.0

160

270

167

250

**2** Polestar 2 4.7 210 400 181 620

**3**

BMW

iX3

6.8

180

360

206

560

**4** Honda e 9.5 145 170 168 190

# Info()

In [455…

ev\_car\_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 |  | object |
| 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 |  | object |
| 13 |  | PriceEuro | 103 | non-null |  | int64 |

dtypes: float64(1), int64(3), object(10) memory usage: 11.4+ KB

#### Observations

* 1. Data Types of TopSpeed\_KmH,FastCharge\_KmH and Seats columns supposed to be int64 but info() showed it up as Object

# Data Pre-Processing

In [456… Out[456]:

In [457…

array(['Tesla ', 'Volkswagen ', 'Polestar ', 'BMW ', 'Honda ', 'Lucid ',

ev\_car\_df**.**Brand**.**unique()

'Peugeot ', 'Audi ', 'Mercedes ', 'Nissan ', 'Hyundai ',

'Porsche ', 'MG ', 'Mini ', 'Opel ', 'Skoda ', 'Volvo ', 'Kia ',

'Renault ', 'Mazda ', 'Lexus ', 'CUPRA ', 'SEAT ', 'Lightyear ',

'Aiways ', 'DS ', 'Citroen ', 'Jaguar ', 'Ford ', 'Byton ', 'Sono ', 'Smart ', 'Fiat '], dtype=object)

*#- Remove Leading and trailing spaces*

ev\_car\_df['Brand'] **=** ev\_car\_df['Brand']**.**str**.**strip() ev\_car\_df['Model'] **=** ev\_car\_df['Model']**.**str**.**strip()

### TopSpeed\_KmH

In [458… Out[458]:

In [459…

array(['233', '160', '210', '180', '145', '250', '150', '225', '144',

ev\_car\_df['TopSpeed\_KmH']**.**unique()

'167', '200', '165', '260', '130', '140', '217', '240', '261',

'190', '135', '157', '155', '410', '241', 'two hundred', '123'],

dtype=object)

*#- we have a textual representation of the value - replace 'two hundred' with 200* ev\_car\_df['TopSpeed\_KmH'] **=** ev\_car\_df['TopSpeed\_KmH']**.**replace('two hundred', '200' *#- convert the entire column into int64 data type*

ev\_car\_df['TopSpeed\_KmH'] **=** ev\_car\_df['TopSpeed\_KmH']**.**astype('int64') *#- cast to i*

ev\_car\_df['TopSpeed\_KmH']**.**unique()

Out[459]:

array([233, 160, 210, 180, 145, 250, 150, 225, 144, 167, 200, 165, 260,

130, 140, 217, 240, 261, 190, 135, 157, 155, 410, 241, 123],

dtype=int64)

### FastCharge\_KmH

In [460… Out[460]:

In [461…

array(['940', '250', '620', '560', '190', '220', '420', '650', '540',

ev\_car\_df['FastCharge\_KmH']**.**unique()

'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)

*#- we see "-" value which is not meaningful - we either can replace with Average v #- Lets print how many such records we have*

ev\_car\_df[ev\_car\_df**.**FastCharge\_KmH **==** "-"]

Out[461]:

**Brand Model AccelSec TopSpeed\_KmH Range\_Km Efficiency\_WhKm FastCharge\_KmH R**

**57** Renault Twingo

ZE

12.6 135 130 164 -

**68** Renault

Kangoo

Maxi ZE 33

22.4

130

160

194

-

**77** Smart EQ forfour

12.7 130 95 176 -

**82** Smart

EQ

fortwo coupe

11.6

130

100

167

-

**91** Smart

EQ

fortwo cabrio

11.9 130 95 176 -

In [462…

fast\_charge\_df **=** ev\_car\_df[ev\_car\_df**.**FastCharge\_KmH **!=** "-"][['FastCharge\_KmH']]

fast\_charge\_df['FastCharge\_KmH'] **=** fast\_charge\_df['FastCharge\_KmH']**.**astype('int64' fast\_charge\_df['FastCharge\_KmH']**.**mean()

Out[462]: In [463…

*#- Lets replace "-" with np.nan*

ev\_car\_df['FastCharge\_KmH'] **=** np**.**where(ev\_car\_df['FastCharge\_KmH'] **==** "-", "456", ev\_car\_df['FastCharge\_KmH'] **=** ev\_car\_df['FastCharge\_KmH']**.**astype('int64')

456.734693877551

In [464… Out[464]:

In [465… Out[465]:

ev\_car\_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, | 456, | 550, | 900, | 520, | 430, | 890, |
| 410, 770, 460,  dtype=int64) | 360, | 810, | 480, | 290, | 330, | 740, | 510, | 320, | 500], |  |

### Seats

ev\_car\_df['Seats']**.**unique()

array(['5', '4', '7', '6', 'five', '2'], dtype=object)

In [466…

*#- we have a textual representation of the value - replace 'five' with 5* ev\_car\_df['Seats'] **=** ev\_car\_df['Seats']**.**replace('five', '5') *#- replace #- convert the entire column into int64 data type*

ev\_car\_df['Seats'] **=** ev\_car\_df['Seats']**.**astype('int64') *#- cast to int64*

ev\_car\_df['Seats']**.**unique()

Out[466]:

In [467…

array([5, 4, 7, 6, 2], dtype=int64)

info()

<class 'pandas.core.frame.DataFrame'> RangeIndex: 103 entries, 0 to 102

ev\_car\_df**.**info()

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 |  | int64 |
| 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(6), object(7) memory usage: 11.4+ KB

# Descriptive Stats - Describe()

In [468…

ev\_car\_df**.**describe(include**=**'all')

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Out[468]: | **Brand** | **Model** | **AccelSec** | **TopSpeed\_KmH** | **Range\_Km** | **Efficiency\_WhKm** | **FastCharge\_Km** |
| **count** | 103 | 103 | 103.000000 | 103.000000 | 103.000000 | 103.000000 | 103.00000 |
| **unique** | 33 | 102 | NaN | NaN | NaN | NaN | Na |
| **top** | Tesla | e-Soul  64 | NaN | NaN | NaN | NaN | Na |
|  |  | kWh |  |  |  |  |  |
| **freq** | 13 | 2 | NaN | NaN | NaN | NaN | Na |
| **mean** | NaN | NaN | 7.396117 | 179.194175 | 338.786408 | 189.165049 | 456.69902 |
| **std** | NaN | NaN | 3.017430 | 43.573030 | 126.014444 | 29.566839 | 196.26806 |
| **min** | NaN | NaN | 2.100000 | 123.000000 | 95.000000 | 104.000000 | 170.00000 |
| **25%** | NaN | NaN | 5.100000 | 150.000000 | 250.000000 | 168.000000 | 305.00000 |
| **50%** | NaN | NaN | 7.300000 | 160.000000 | 340.000000 | 180.000000 | 450.00000 |
| **75%** | NaN | NaN | 9.000000 | 200.000000 | 400.000000 | 203.000000 | 555.00000 |
| **max** | NaN | NaN | 22.400000 | 410.000000 | 970.000000 | 273.000000 | 940.00000 |

#### Observations

##### There are 33 Unique brands, More Cars of brand Tesla

* 1. Average Accelration per Sec for all vehciles = 7.3 while Min & Max are 2.10, 22.4

##### Average TopSpeed = 17.19 KMH

* 1. 95% vehcile support Rapid Charge

##### 39.8% vehicles are AWD Power Train

* 1. Plug type is Type 2 CCS for 87.37% vehicles

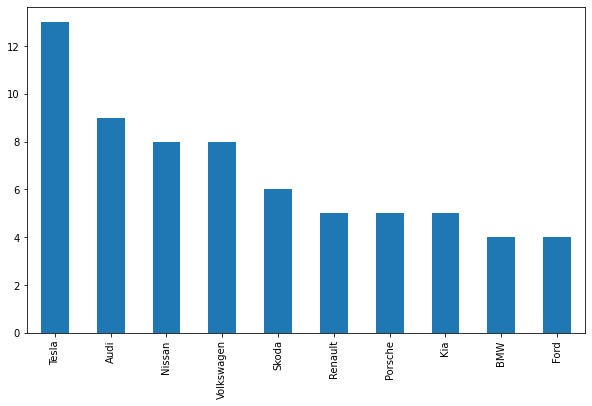
# UniVariate Analysis

In [469…

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

ev\_car\_df['Brand']**.**value\_counts()**.**head(10)**.**plot(kind**=**'bar') plt**.**show()





#### Observations

##### Number of Models in EV : Tesla followed by Audi

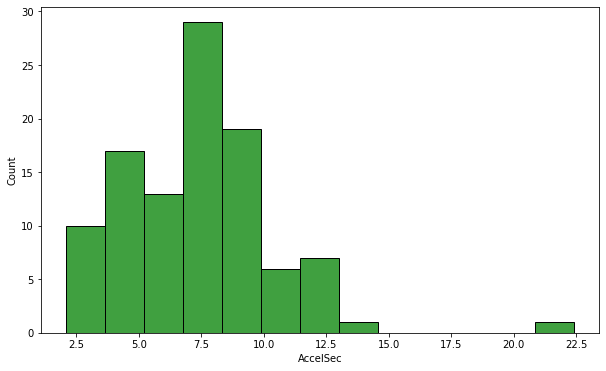
In [470…

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

sns**.**histplot(x**=**'AccelSec', data**=**ev\_car\_df, color**=**'green')

Out[470]:

<AxesSubplot:xlabel='AccelSec', ylabel='Count'>



#### Observation

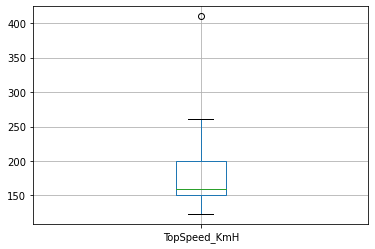
##### 1. These are some higher values of AccelSec which can be considered as anomolies

In [471… Out[471]:

In [473…

<AxesSubplot:>

ev\_car\_df**.**boxplot(column**=**'TopSpeed\_KmH')



#### Observation

##### we definitely see a outlier in TopSpeed\_KmH which make distribution Right Skewed

Out[473]:

In [474…

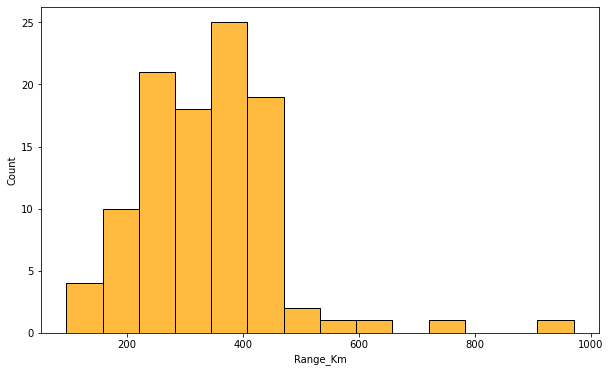
plt**.**figure(figsize**=**(10,6))

sns**.**histplot(x**=**'Efficiency\_WhKm', data**=**ev\_car\_df, color**=**'black')

<AxesSubplot:xlabel='Range\_Km', ylabel='Count'>

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

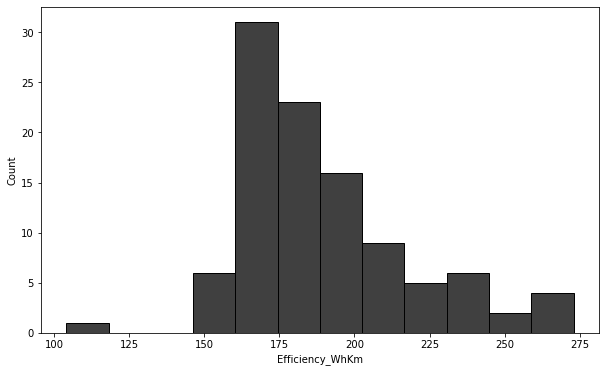
sns**.**histplot(x**=**'Range\_Km', data**=**ev\_car\_df, color**=**'orange')



Out[474]:

In [475…

<AxesSubplot:xlabel='Efficiency\_WhKm', ylabel='Count'>



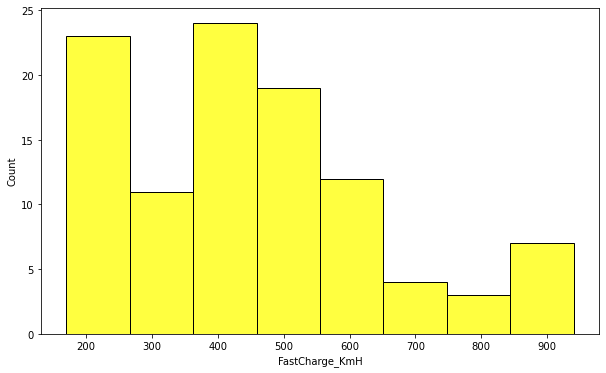
Out[475]:

In [476…

<AxesSubplot:xlabel='FastCharge\_KmH', ylabel='Count'>

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

sns**.**histplot(x**=**'FastCharge\_KmH', data**=**ev\_car\_df, color**=**'yellow')



Out[476]:

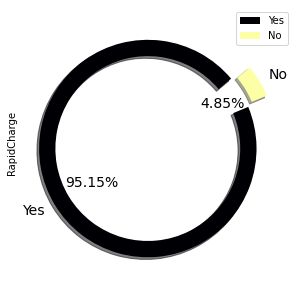
<AxesSubplot:ylabel='RapidCharge'>

ev\_car\_df['RapidCharge']**.**value\_counts()**.**plot(kind**=**'pie',figsize**=**(5,5),

autopct**=**'%.2f%%', explode**=**np**.**ones(2)**/** fontsize**=**14,legend**=True**,shadow**=True**,

wedgeprops**=**{'width':0.15}, cmap**=**'infer





#### Observation

##### 95% vehicles are RapidCharge Supported

In [477…

ev\_car\_df['PlugType']**.**value\_counts()**.**plot(kind**=**'pie',figsize**=**(5,5),

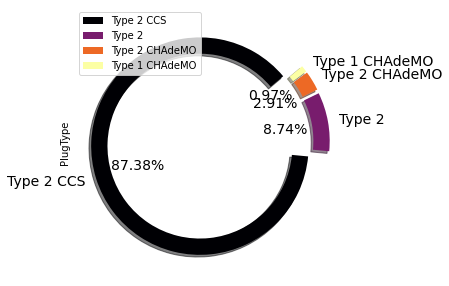
autopct**=**'%.2f%%', explode**=**np**.**ones(4)**/** fontsize**=**14,legend**=True**,shadow**=True**,

wedgeprops**=**{'width':0.15}, cmap**=**'infer

Out[477]:

In [478…

<AxesSubplot:ylabel='PlugType'>



#### Observation

##### 1. 87% vehicles are of plug Type "Type2CCS"

Out[478]:

<AxesSubplot:ylabel='PowerTrain'>

ev\_car\_df['PowerTrain']**.**value\_counts()**.**plot(kind**=**'pie',figsize**=**(5,5),

autopct**=**'%.2f%%', explode**=**np**.**ones(3)**/** fontsize**=**14,legend**=True**,shadow**=True**,

wedgeprops**=**{'width':0.15}, cmap**=**'infer





In [479…

ev\_car\_df['BodyStyle']**.**value\_counts()**.**plot(kind**=**'pie',figsize**=**(5,5),

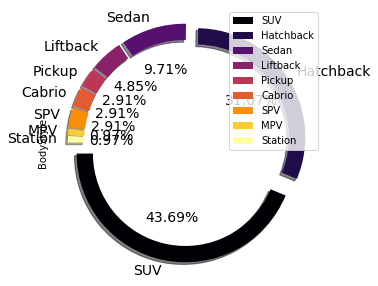
autopct**=**'%.2f%%', explode**=**np**.**ones(ev\_ fontsize**=**14,legend**=True**,shadow**=True**,

wedgeprops**=**{'width':0.15}, cmap**=**'infer

Out[479]:

In [480…

<AxesSubplot:ylabel='BodyStyle'>



#### Observation

##### 1. Vehicle are more of Body type SUV followed by Hatchback

Out[480]:

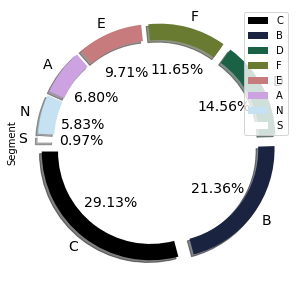
<AxesSubplot:ylabel='Segment'>

ev\_car\_df['Segment']**.**value\_counts()**.**plot(kind**=**'pie',figsize**=**(5,5),

autopct**=**'%.2f%%', explode**=**np**.**ones(ev\_ fontsize**=**14,legend**=True**,shadow**=True**,

wedgeprops**=**{'width':0.15}, cmap**=**'cubeh





In [481…

ev\_car\_df['Seats']**.**value\_counts()**.**plot(kind**=**'pie',figsize**=**(5,5),

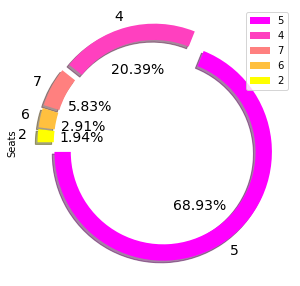
autopct**=**'%.2f%%', explode**=**np**.**ones(ev\_ fontsize**=**14,legend**=True**,shadow**=True**,

wedgeprops**=**{'width':0.15}, cmap**=**'sprin

Out[481]:

In [482… Out[482]:

<AxesSubplot:ylabel='Seats'>



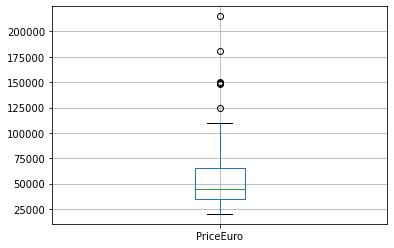
#### Observation

##### Vehicle with 5 seating is more preferred

ev\_car\_df**.**boxplot(column**=**'PriceEuro')

<AxesSubplot:>





#### Observation

* 1. There are definitely vehicles with higher prices

# Bi-Variate Analysis

In [483…

*#- Brand Min, Max and Average price*

brand\_price **=** ev\_car\_df**.**groupby('Brand')**.**agg({"PriceEuro" : ['min','mean','max']}) brand\_price**.**columns **=** [ "\_"**.**join(col) **for** col **in** brand\_price**.**columns]

brand\_price**.**sort\_values('PriceEuro\_mean',ascending**=False**)**.**head(10)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Out[483]: |  | **Brand\_** | **PriceEuro\_min** | **PriceEuro\_mean** | **PriceEuro\_max** |
|  | **14** | Lightyear | 149000 | 149000.000000 | 149000 |
|  | **24** | Porsche | 102945 | 138265.800000 | 180781 |
|  | **15** | Lucid | 105000 | 105000.000000 | 105000 |
|  | **1** | Audi | 55000 | 80593.666667 | 125000 |
|  | **30** | Tesla | 45000 | 80272.307692 | 215000 |
|  | **11** | Jaguar | 75351 | 75351.000000 | 75351 |
|  | **18** | Mercedes | 45000 | 61705.000000 | 70631 |
|  | **32** | Volvo | 60437 | 60437.000000 | 60437 |
|  | **3** | Byton | 53500 | 59833.333333 | 64000 |
|  | **23** | Polestar | 56440 | 56440.000000 | 56440 |

#### Observations

##### Average price of Audi is slighly on higer side when compared with tesla

* 1. Max price for Audi is 125000 Euros while for Tesla it is 215000 Euros

In [484…

rapidcharge\_price **=** ev\_car\_df**.**groupby('RapidCharge')**.**agg({"PriceEuro" :'mean'})**.**re rapidcharge\_price**.**columns **=** ['RapidCharge', 'PriceEuro']

rapidcharge\_price**.**sort\_values('PriceEuro',ascending**=False**)**.**head(10)

Out[484]:

**RapidCharge PriceEuro 1** Yes 57324.683673

#### Observations

**0** No 26154.400000

##### 1. Rapid Charge Vehicle are 50% higher on price when compared with Non Rapid Charge Vehicles

In [485… Out[485]:

In [486…

Index(['Brand', 'Model', 'AccelSec', 'TopSpeed\_KmH', 'Range\_Km',

ev\_car\_df**.**columns

'Efficiency\_WhKm', 'FastCharge\_KmH', 'RapidCharge', 'PowerTrain', 'PlugType', 'BodyStyle', 'Segment', 'Seats', 'PriceEuro'],

dtype='object')

pt\_price **=** ev\_car\_df**.**groupby('PowerTrain')**.**agg({"PriceEuro" :'mean'})**.**reset\_index( pt\_price**.**columns **=** ['PowerTrain', 'PriceEuro']

pt\_price**.**sort\_values('PriceEuro',ascending**=False**)**.**head(10)

|  |  |  |  |
| --- | --- | --- | --- |
| Out[486]: |  | **PowerTrain** | **PriceEuro** |
|  | **0** | AWD | 83840.097561 |
|  | **2** | RWD | 40061.040000 |
|  | **1** | FWD | 35395.162162 |

#### Observations

##### 1. Averge Price for AWD Vehicles are on higher side

In [487…

seats\_price **=** ev\_car\_df**.**groupby('Seats')**.**agg({"PriceEuro" :'mean'})**.**reset\_index() seats\_price**.**columns **=** ['Seats', 'PriceEuro']

seats\_price**.**sort\_values('PriceEuro',ascending**=False**)**.**head(10)

|  |  |  |  |
| --- | --- | --- | --- |
| Out[487]: |  | **Seats** | **PriceEuro** |
|  | **1** | 4 | 70624.333333 |
|  | **4** | 7 | 69516.166667 |
|  | **3** | 6 | 58333.333333 |
|  | **2** | 5 | 51090.577465 |
|  | **0** | 2 | 22976.000000 |

#### Observation

1. We have 5 seating vehicles are more in percentage because of its price is low when compared with 4 seating

# Multi-Variate Analysis

In [488…

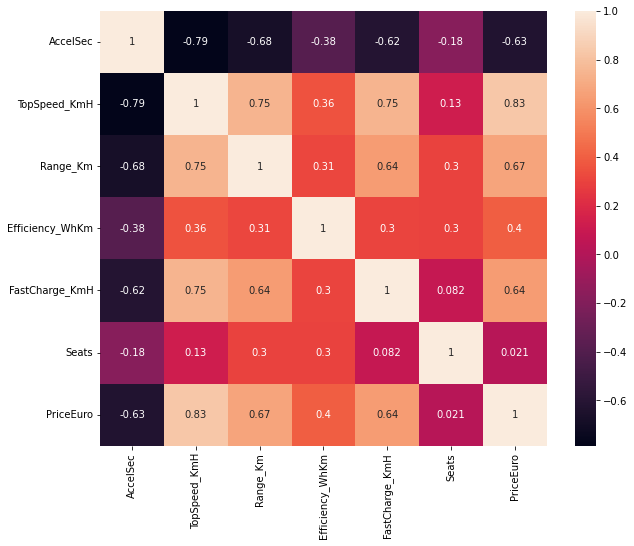
plt**.**figure(figsize**=**(10,8))

sns**.**heatmap(ev\_car\_df**.**corr(),annot**=True**)

Out[488]:

In [489… Out[489]:

<AxesSubplot:>



#### Observation:

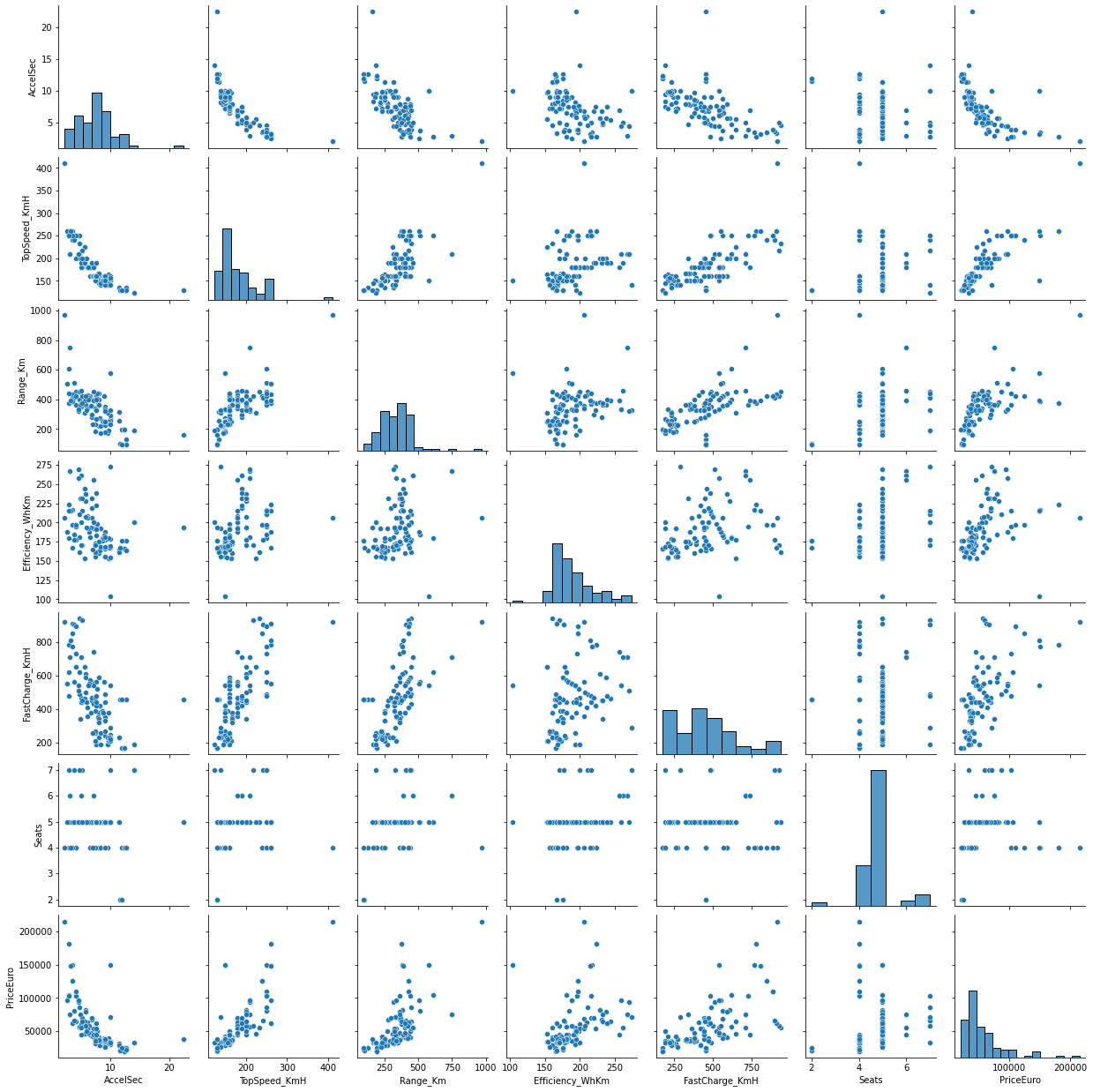
##### Considering 0.8 correlation value

1. Strong Negative corelation between TopSpeed\_KmH and AccelSec

sns**.**pairplot(ev\_car\_df)

<seaborn.axisgrid.PairGrid at 0x28fc0989a90>





# Building Regression Model

In [490…

**for** col **in** ev\_car\_df**.**columns:

print(f"{col} = {ev\_car\_df[col]**.**nunique()}")

Brand = 33

Model = 102

AccelSec = 55

TopSpeed\_KmH = 25

Range\_Km = 50

Efficiency\_WhKm = 54

FastCharge\_KmH = 51

RapidCharge = 2

PowerTrain = 3

PlugType = 4

BodyStyle = 9

Segment = 8

Seats = 5

PriceEuro = 87

In [491…

*#- Model is so unique - we can consider this as an id column - so dropping this co*

In [492…

ev\_car\_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 |  | int64 |
| 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(6), object(7) memory usage: 11.4+ KB

ev\_car\_df**.**columns

In [493… Out[493]:

In [494…

Index(['Brand', 'Model', 'AccelSec', 'TopSpeed\_KmH', 'Range\_Km',

'Efficiency\_WhKm', 'FastCharge\_KmH', 'RapidCharge', 'PowerTrain', 'PlugType', 'BodyStyle', 'Segment', 'Seats', 'PriceEuro'],

dtype='object')

categorical\_col **=** ['Brand','RapidCharge', 'PowerTrain','PlugType', 'BodyStyle', 'S numerical\_col **=** ['AccelSec', 'TopSpeed\_KmH', 'Range\_Km','Efficiency\_WhKm', 'FastCh

In [495…

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

**from** sklearn.preprocessing **import** StandardScaler **from** sklearn.linear\_model **import** LinearRegression **from** sklearn.ensemble **import** RandomForestRegressor

**from** sklearn.metrics **import** mean\_squared\_error, r2\_score

In [496…

*#- Convert Categorical Features to Numeric using pandas get\_dummies*

categorical\_df **=** pd**.**get\_dummies(ev\_car\_df[categorical\_col], columns**=**categorical\_co

In [497…

numerical\_df **=** ev\_car\_df[numerical\_col]

In [498…

ev\_model\_df **=** pd**.**concat([categorical\_df, numerical\_df],axis**=**1) ev\_model\_df**.**head(2)

Out[498]:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Brand\_Audi** | **Brand\_BMW** | **Brand\_Byton** | **Brand\_CUPRA** | **Brand\_Citroen** | **Brand\_DS** | **Brand\_Fiat B** |
| **0** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **1** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

In [ ]:

## Train-Test Split

In [499…

X **=** ev\_model\_df**.**drop('PriceEuro',axis**=**1) y **=** ev\_model\_df['PriceEuro']

In [500…

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X,y,test\_size**=**0.2, random\_stat

In [501…

print("Number of Training Records = ", X\_train**.**shape[0]) print("Number of Test Records = ", X\_test**.**shape[0])

Number of Training Records = 82 Number of Test Records = 21

In [502…

std\_scaler **=** StandardScaler() std\_scaler**.**fit(X\_train)

X\_train\_scaled **=** std\_scaler**.**transform(X\_train) X\_test\_scaled **=** std\_scaler**.**transform(X\_test)

## Linear Regression

In [503…

linear\_model **=** LinearRegression()

linear\_model**.**fit(X\_train\_scaled,y\_train)

Out[503]: In [504…

y\_pred **=** linear\_model**.**predict(X\_test\_scaled)

LinearRegression()

In [505…

print(linear\_model**.**intercept\_)

59096.65853658536

In [506…

print(linear\_model**.**coef\_)

[ 2.89840851e+03 4.19106562e+03 -1.14053846e+03 1.00557418e+03 1.41876933e+03 1.20676078e+03 1.45362574e+03 2.31465567e+02

2.19955901e+03 2.24180104e+03 1.02772901e-10 2.18787952e+03

-2.50079742e+02 1.24964668e+04 -5.56901825e+02 -5.83264927e+02 5.78707948e+02 9.02330387e+02 1.06410880e-10 -2.28249378e+02

1.98605353e+03 1.41206677e+03 1.17324817e-10 2.64426320e+03

3.98503008e+03 2.42380338e-10 2.63517065e+03 1.01761573e+03

1.38116120e+03 8.52460809e+03 3.24123604e+03 1.42487358e+03

-1.08758470e+04 5.51164036e+03 2.10080230e+03 -2.45449027e+04

-1.09590274e+04 -4.94285307e+03 6.50504970e+02 1.10961856e+03 3.06639318e+02 -5.53159922e+03 2.98944540e+00 -2.31334690e+02

-1.99961255e+03 2.95838374e+02 5.43623056e+02 2.02827599e+03

3.35527307e+03 1.08679084e+01 2.33234684e+04 -3.95911969e+03

9.55178040e+03 -6.11860842e+03 8.18041754e+03 2.30908684e+03

1.99195538e+04 -1.85273780e+03 -1.31373377e+03]

## Model Evaluation

In [507…

rmse\_error **=** np**.**sqrt(mean\_squared\_error(y\_test,y\_pred)) print("RMSE = ", rmse\_error)

RMSE = 9231.680614936346

In [508…

rscore **=** r2\_score(y\_test,y\_pred) print("R2\_SCORE =", rscore)

R2\_SCORE = 0.7582699134423788

In [509…

comp\_df\_lr **=** pd**.**concat([pd**.**DataFrame(pd**.**Series(y\_test))**.**reset\_index(drop**=True**),

pd**.**DataFrame(pd**.**Series(y\_pred))**.**reset\_index(drop**=True**)],ax comp\_df\_lr**.**columns **=** ['ACTUAL','PREDICTED']

comp\_df\_lr

Out[509]:

**ACTUAL PREDICTED 0** 36837 40551.293405

**1** 34459 39584.620708

**2** 75351 69914.481173

**3** 24565 26187.471774

**4** 45000 36544.611213

**5** 68040 78846.665892

**6** 31681 16908.067575

**7** 33000 29590.531441

**8** 33971 28714.006194

**9** 21387 22226.073986

**10** 20129 8026.103517

**11** 35921 35842.632443

**12** 56440 51164.718667

**13** 30000 28659.255456

**14** 57500 51658.728586

**15** 46900 51657.004926

**16** 38000 6642.912594

**17** 55000 54354.171678

**18** 37900 33486.951594

**19** 96050 98228.545679

**20** 24534 19038.763568

|  |  |  |
| --- | --- | --- |
|  | | Random Forest |
| Train-Test Split |
| In | [510… | X **=** ev\_model\_df**.**drop('PriceEuro',axis**=**1) y **=** ev\_model\_df['PriceEuro'] |
|  |  |  |
| In | [511… | X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X,y,test\_size**=**0.2, random\_stat |
|  |  |  |
| In | [512… | print("Number of Training Records = ", X\_train**.**shape[0]) print("Number of Test Records = ", X\_test**.**shape[0]) |
|  |  | Number of Training Records = 82 |
|  |  | Number of Test Records = 21 |
| In | [513… | rf\_model **=** RandomForestRegressor(criterion**=**'mse') rf\_model**.**fit(X\_train,y\_train) |

Out[513]:

RandomForestRegressor(criterion='mse')

In [514…

y\_rf\_pred **=** rf\_model**.**predict(X\_test)

In [ ]:

## Model Evaluation

In [515…

rmse\_error **=** np**.**sqrt(mean\_squared\_error(y\_test,y\_rf\_pred)) print("RMSE = ", rmse\_error)

RMSE = 6849.604619415293

In [516…

rscore **=** r2\_score(y\_test,y\_rf\_pred) print("R2\_SCORE =", rscore)

R2\_SCORE = 0.8669238669944139

In [517…

comp\_df\_rf **=** pd**.**concat([pd**.**DataFrame(pd**.**Series(y\_test))**.**reset\_index(drop**=True**),

pd**.**DataFrame(pd**.**Series(y\_rf\_pred))**.**reset\_index(drop**=True**)] comp\_df\_rf**.**columns **=** ['ACTUAL','PREDICTED']

comp\_df\_rf

Out[517]:

In [ ]:

**ACTUAL PREDICTED 0** 36837 37648.853333

**1** 34459 38229.113333

**2** 75351 69470.770000

**3** 24565 25944.020000

**4** 45000 58051.740000

**5** 68040 53905.200000

**6** 31681 35374.280000

**7** 33000 35580.510000

**8** 33971 32495.380000

**9** 21387 25899.220000

**10** 20129 25496.690000

**11** 35921 30973.790000

**12** 56440 63853.520000

**13** 30000 34785.630000

**14** 57500 54920.290000

**15** 46900 52657.520000

**16** 38000 32324.220000

**17** 55000 67792.880000

**18** 37900 34970.110000

**19** 96050 84025.320000

**20** 24534 25496.690000

In [524… Out[524]:

In [518… Out[518]:

In [525…

rf\_model**.**feature\_names\_in\_

array(['Brand\_Audi', 'Brand\_BMW', 'Brand\_Byton', 'Brand\_CUPRA', 'Brand\_Citroen', 'Brand\_DS', 'Brand\_Fiat', 'Brand\_Ford',

'Brand\_Honda', 'Brand\_Hyundai', 'Brand\_Jaguar', 'Brand\_Kia', 'Brand\_Lexus', 'Brand\_Lightyear', 'Brand\_Lucid', 'Brand\_MG',

'Brand\_Mazda', 'Brand\_Mercedes', 'Brand\_Mini', 'Brand\_Nissan',

'Brand\_Opel', 'Brand\_Peugeot', 'Brand\_Polestar', 'Brand\_Porsche', 'Brand\_Renault', 'Brand\_SEAT', 'Brand\_Skoda', 'Brand\_Smart',

'Brand\_Sono', 'Brand\_Tesla', 'Brand\_Volkswagen', 'Brand\_Volvo',

'RapidCharge\_Yes', 'PowerTrain\_FWD', 'PowerTrain\_RWD', 'PlugType\_Type 2', 'PlugType\_Type 2 CCS',

'PlugType\_Type 2 CHAdeMO', 'BodyStyle\_Hatchback',

'BodyStyle\_Liftback', 'BodyStyle\_MPV', 'BodyStyle\_Pickup', 'BodyStyle\_SPV', 'BodyStyle\_SUV', 'BodyStyle\_Sedan',

'BodyStyle\_Station', 'Segment\_B', 'Segment\_C', 'Segment\_D',

'Segment\_E', 'Segment\_F', 'Segment\_N', 'Segment\_S', 'AccelSec', 'TopSpeed\_KmH', 'Range\_Km', 'Efficiency\_WhKm', 'FastCharge\_KmH', 'Seats'], dtype=object)

rf\_model**.**feature\_importances\_

array([4.14638642e-03, 6.75377864e-05, 1.39580622e-04, 4.06563833e-05, 9.99143275e-05, 1.23669871e-05, 1.11871538e-06, 5.88698177e-05,

4.71306836e-07, 1.37225696e-06, 0.00000000e+00, 8.03539969e-06,

1.56124049e-05, 6.06508107e-03, 8.14982621e-05, 7.04721800e-06,

9.13615200e-06, 2.84964373e-03, 0.00000000e+00, 7.13263757e-05,

7.05278340e-05, 5.00438466e-05, 0.00000000e+00, 1.12502545e-02,

1.25311087e-05, 0.00000000e+00, 1.16102121e-04, 1.33690918e-05,

2.27895003e-05, 5.03462835e-03, 1.05877501e-04, 4.95031127e-05,

8.30630360e-05, 2.46816510e-05, 1.94914249e-03, 3.97883064e-03,

4.09341179e-03, 4.12762494e-05, 1.98009891e-04, 5.25340785e-03,

3.54703460e-05, 2.09624675e-04, 1.40412976e-04, 6.98580472e-04,

1.64245459e-03, 3.99983407e-04, 6.33187191e-05, 1.54337564e-04,

1.42318657e-03, 9.09569530e-04, 2.15469264e-01, 2.47298385e-04,

1.17651031e-02, 2.51443161e-01, 2.89039189e-01, 3.82967461e-02,

7.13713710e-02, 3.03161793e-02, 4.03516438e-02])

**from** decimal **import** Decimal

**for** f\_name, fval **in** zip(rf\_model**.**feature\_names\_in\_,rf\_model**.**feature\_importances\_): print("%-30s = %-10f"**%**(f\_name,Decimal(fval)))

|  |  |  |  |
| --- | --- | --- | --- |
| Brand\_Audi |  | = | 0.004146 |
| Brand\_BMW |  | = | 0.000068 |
| Brand\_Byton |  | = | 0.000140 |
| Brand\_CUPRA |  | = | 0.000041 |
| Brand\_Citroen |  | = | 0.000100 |
| Brand\_DS |  | = | 0.000012 |
| Brand\_Fiat |  | = | 0.000001 |
| Brand\_Ford |  | = | 0.000059 |
| Brand\_Honda |  | = | 0.000000 |
| Brand\_Hyundai |  | = | 0.000001 |
| Brand\_Jaguar |  | = | 0.000000 |
| Brand\_Kia |  | = | 0.000008 |
| Brand\_Lexus |  | = | 0.000016 |
| Brand\_Lightyear |  | = | 0.006065 |
| Brand\_Lucid |  | = | 0.000081 |
| Brand\_MG |  | = | 0.000007 |
| Brand\_Mazda |  | = | 0.000009 |
| Brand\_Mercedes |  | = | 0.002850 |
| Brand\_Mini |  | = | 0.000000 |
| Brand\_Nissan |  | = | 0.000071 |
| Brand\_Opel |  | = | 0.000071 |
| Brand\_Peugeot |  | = | 0.000050 |
| Brand\_Polestar |  | = | 0.000000 |
| Brand\_Porsche |  | = | 0.011250 |
| Brand\_Renault |  | = | 0.000013 |
| Brand\_SEAT |  | = | 0.000000 |
| Brand\_Skoda |  | = | 0.000116 |
| Brand\_Smart |  | = | 0.000013 |
| Brand\_Sono |  | = | 0.000023 |
| Brand\_Tesla |  | = | 0.005035 |
| Brand\_Volkswagen |  | = | 0.000106 |
| Brand\_Volvo |  | = | 0.000050 |
| RapidCharge\_Yes |  | = | 0.000083 |
| PowerTrain\_FWD |  | = | 0.000025 |
| PowerTrain\_RWD |  | = | 0.001949 |
| PlugType\_Type 2 |  | = | 0.003979 |
| PlugType\_Type 2 | CCS | = | 0.004093 |
| PlugType\_Type 2 | CHAdeMO | = | 0.000041 |
| BodyStyle\_Hatchback | | = | 0.000198 |
| BodyStyle\_Liftback | | = | 0.005253 |
| BodyStyle\_MPV | | = | 0.000035 |
| BodyStyle\_Pickup | | = | 0.000210 |
| BodyStyle\_SPV | | = | 0.000140 |
| BodyStyle\_SUV | | = | 0.000699 |
| BodyStyle\_Sedan | | = | 0.001642 |
| BodyStyle\_Station | | = | 0.000400 |
| Segment\_B | | = | 0.000063 |
| Segment\_C | | = | 0.000154 |
| Segment\_D | | = | 0.001423 |
| Segment\_E | | = | 0.000910 |
| Segment\_F | | = | 0.215469 |
| Segment\_N | | = | 0.000247 |
| Segment\_S | | = | 0.011765 |
| AccelSec | | = | 0.251443 |
| TopSpeed\_KmH | | = | 0.289039 |
| Range\_Km | | = | 0.038297 |
| Efficiency\_WhKm | | = | 0.071371 |
| FastCharge\_KmH | | = | 0.030316 |
| Seats | | = | 0.040352 |

# Model Recommendation

##### The key drivers for predicting PriceEuro area following features

* + 1. AccelSec

##### Segment\_F

* + 1. TopSpeed\_KmH

In [ ]:

# Recommendation:-

1. Electric Vehicles of **Plug Type as "Type 2 CCS"** is more preferred though it is expensive
2. Electric vehicles with more features are preferred even though the price is on higher side
3. Electric vehicles with 5 seating is more preferred because of its price on lower side when compared with 4 seater
4. Audi should manufacture Electric Vehicles with 4/5 seater of body type SUV/hatchback with Rapid Charging/Level2

**Regression Model Recommendations**

Linear Regression Model :

R-Squared Score : 0.75

Random Forest Model:

R-Squared Score : 0.86

We can go with Random Forest Model which has better R-squared when compared with Linear Regression