EV Market Analysis Report

- Mohammed Abdul Akram

Introduction

This report aims to analyse the Electric Vehicle (EV) market in India using segmentation analysis and develop a feasible strategy for an EV startup to enter the market. The analysis includes geographic, demographic, psychographic, and behavioural segments.

An EV is defined as a vehicle that can be powered by an electric motor that draws electricity from a battery and is capable of being charged from an external source

Data Analysis

Loading the data, removing unnecessary column, converting euros to Indian rupees and replacing no and yes values of rapid charge with 0 and 1

```
[2] df = pd.read_csv('data.csv')
    df.drop('Unnamed: 0', axis=1, inplace=True)
    df['inr(10e3)'] = df['PriceEuro']*0.09122
    df['RapidCharge'].replace(to_replace=['No','Yes'],value=[0, 1],inplace=True)
    df.head()
```

:elSec	TopSpeed_KmH	Range_Km	Efficiency_WhKm	FastCharge_KmH	RapidCharge	PowerTrain	PlugType	BodyStyle	Segment	Seats	PriceEuro	inr(10e3)
4.6000	233	450	161	940	1	AWD	Type 2 CCS	Sedan	D	5	55480	5060.8856
0.0000	160	270	167	250	0	RWD	Type 2 CCS	Hatchback	C	5	30000	2736.6000
4.7000	210	400	181	620	1	AWD	Type 2 CCS	Liftback	D	5	56440	5148.4568

Information about type of data in each column

```
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(10e3)        103 non-null    float64
dtypes: float64(2), int64(7), object(6)
```

Descriptive statistics:

```
df.describe()
```

	AccelSec	TopSpeed_KmH	Range_Km	Efficiency_WhKm	FastCharge_KmH	RapidCharge	Seats	PriceEuro	inr(10e3)
count	103.0000	103.0000	103.0000	103.0000	103.0000	103.0000	103.0000	103.0000	103.0000
mean	7.3961	179.1942	338.7864	189.1650	444.2718	0.7476	4.8835	55811.5631	5091.1308
std	3.0174	43.5730	126.0144	29.5668	203.9493	0.4365	0.7958	34134.6653	3113.7642
min	2.1000	123.0000	95.0000	104.0000	170.0000	0.0000	2.0000	20129.0000	1836.1674
25%	5.1000	150.0000	250.0000	168.0000	260.0000	0.5000	5.0000	34429.5000	3140.6590
50%	7.3000	160.0000	340.0000	180.0000	440.0000	1.0000	5.0000	45000.0000	4104.9000
75%	9.0000	200.0000	400.0000	203.0000	555.0000	1.0000	5.0000	65000.0000	5929.3000
max	22.4000	410.0000	970.0000	273.0000	940.0000	1.0000	7.0000	215000.0000	19612.3000

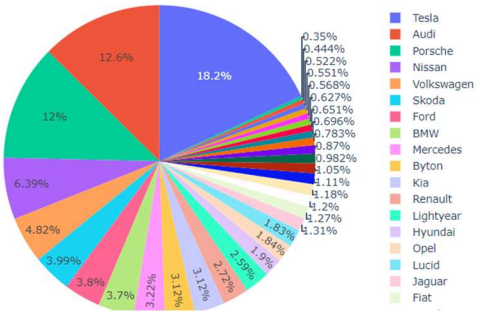
Analysing presence of Null values in the columns

```
df.isnull().sum()
```

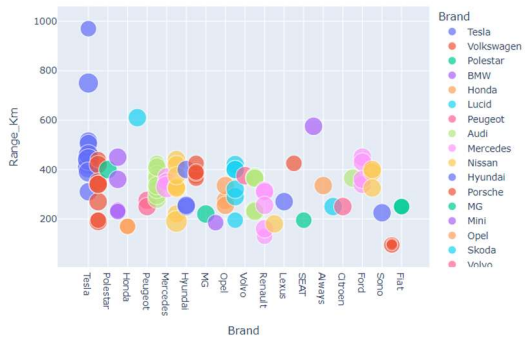
```
Brand              0
Model              0
AccelSec           0
TopSpeed_KmH       0
Range_Km           0
Efficiency_WhKm    0
FastCharge_KmH     0
RapidCharge        0
PowerTrain         0
PlugType           0
BodyStyle          0
Segment            0
Seats              0
PriceEuro          0
inr(10e3)          0
dtype: int64
```

Pie chart of brands

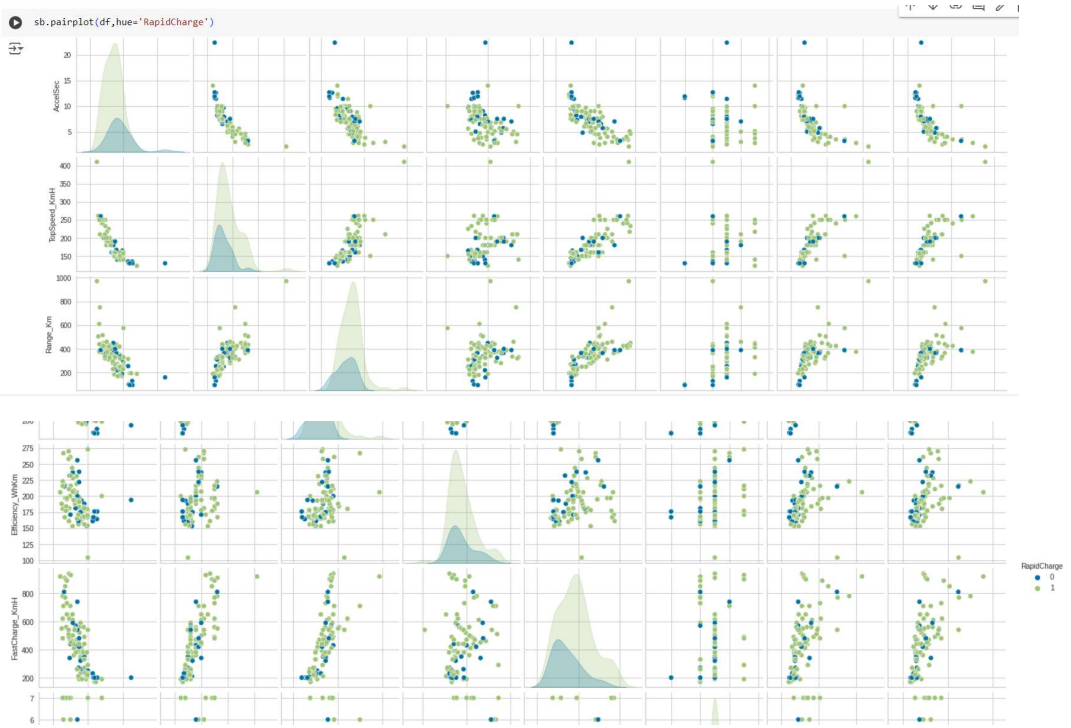
```
fig = px.pie(df,names = 'Brand',values = 'inr(10e3)')
pio.show(fig)
```



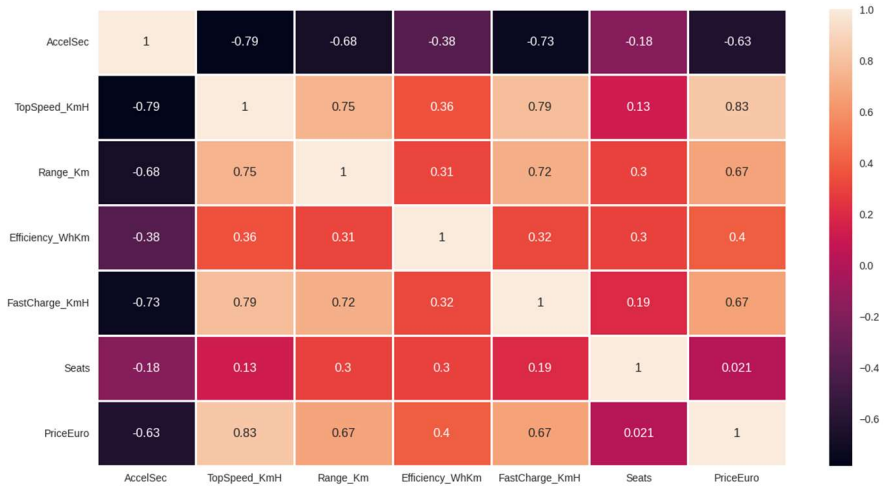
```
fig = px.scatter(df,x = 'Brand',y = 'Range_Km',size='Seats',color = 'Brand',hover_data=['RapidCharge','inr(10e3)'])
pio.show(fig)
```




```
sb.pairplot(df,hue='RapidCharge')
```

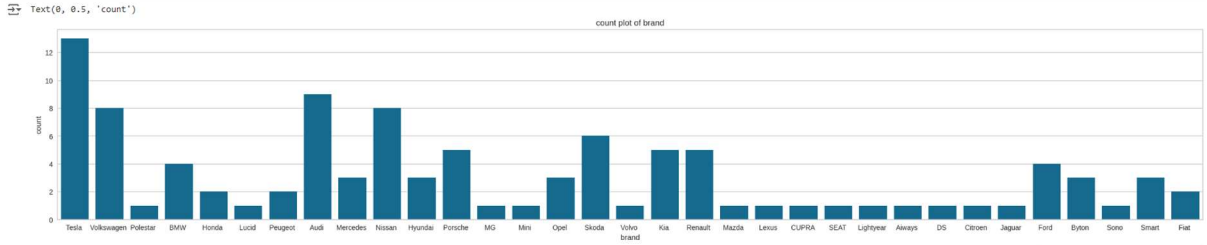


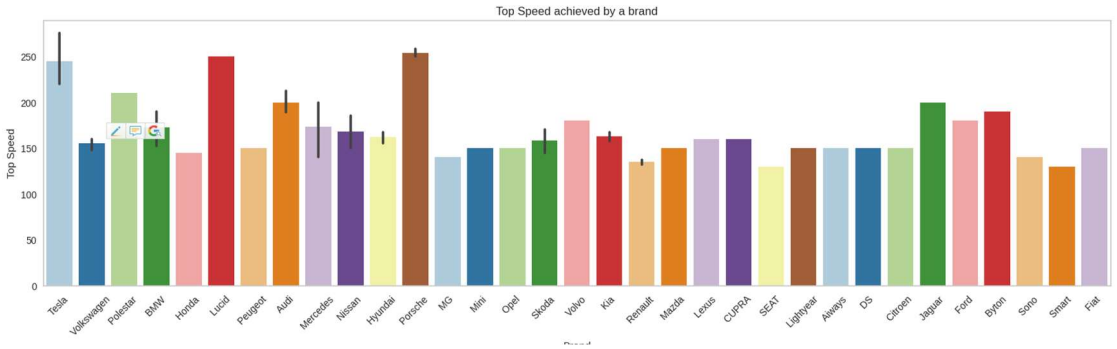
Correlation matrix



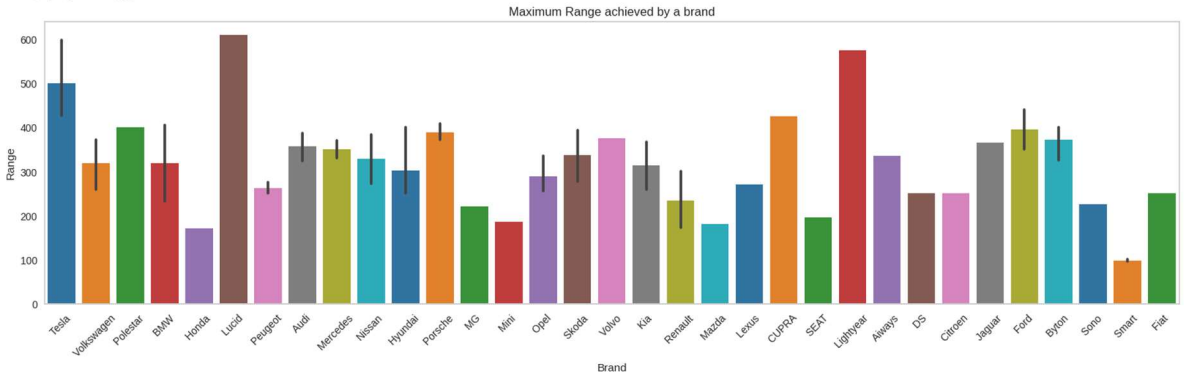
```
ax= plt.figure(figsize=(30,5))
sb.countplot(x=df['Brand'])
plt.title('count plot of brand')
plt.xlabel('brand')
plt.ylabel('count')
```

```
Text(0, 0.5, 'count')
```

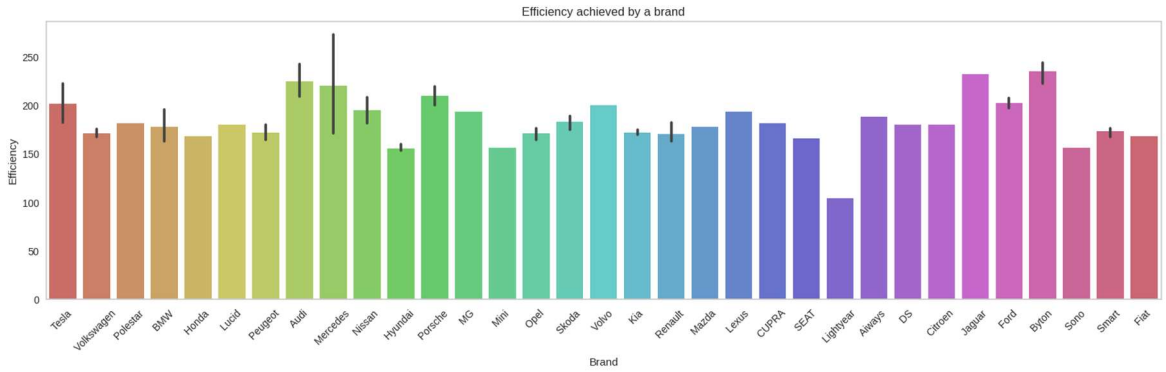




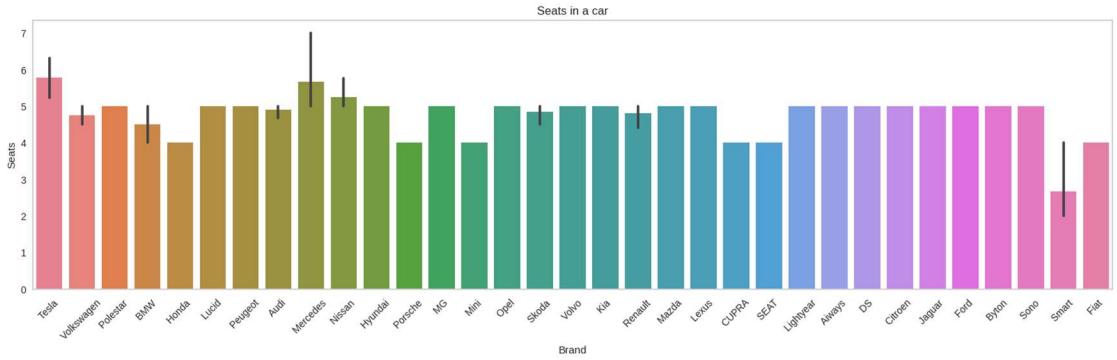
Porsche, Lucid and Tesla produce the fastest cars and Smart the lowest



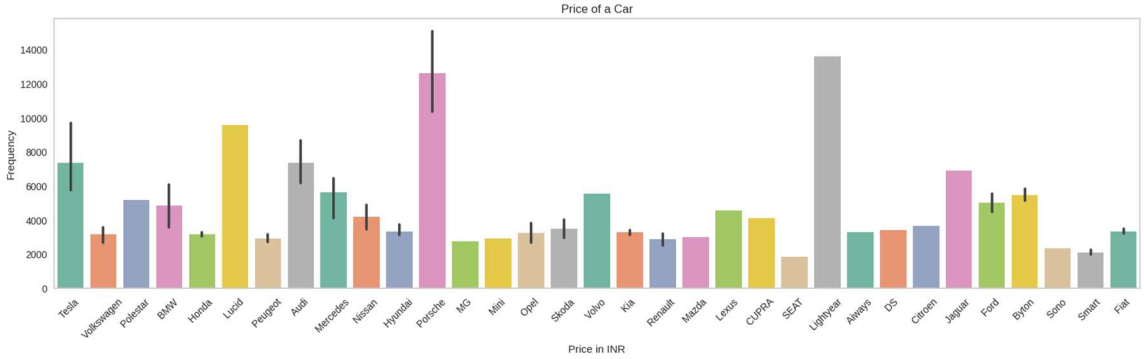
Lucid, Lightyear and Tesla have the highest range and Smart the lowest



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



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

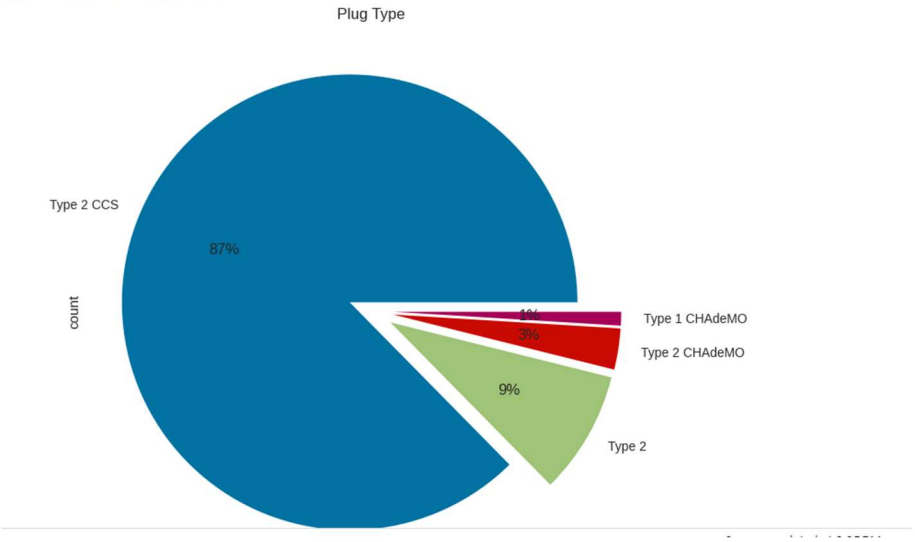


Lightyear, Porsche and Lucid are the most expensive and SEAT and Smart the least

Types of plug used for charging

```
df['PlugType'].value_counts().plot.pie(figsize=(8,10),autopct='%.0f%%',explode=(.1,.1,.1,.1))
plt.title('Plug Type')
```

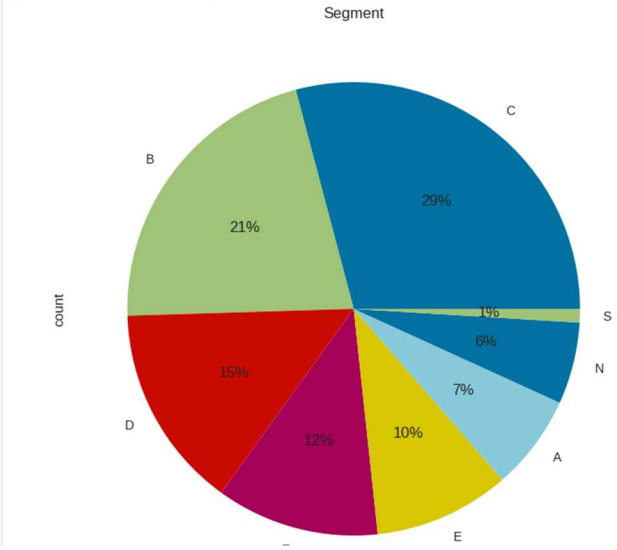
```
Text(0.5, 1.0, 'Plug Type')
```



Segments in which cars fall under

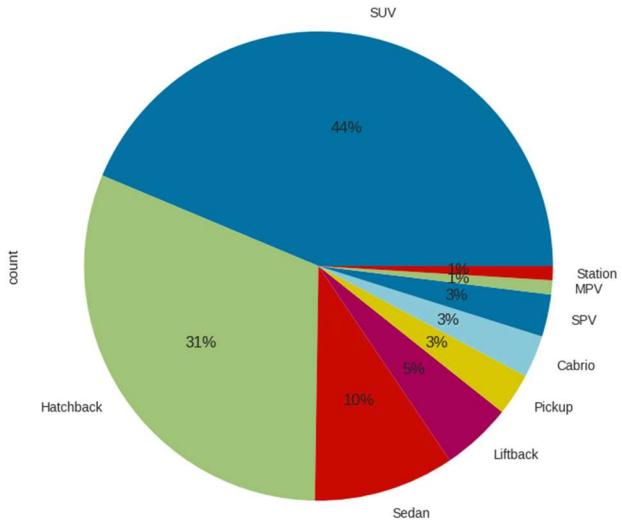
```
df['Segment'].value_counts().plot.pie(figsize=(8,15),autopct='%.0f%%')
plt.title('Segment')
```

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




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

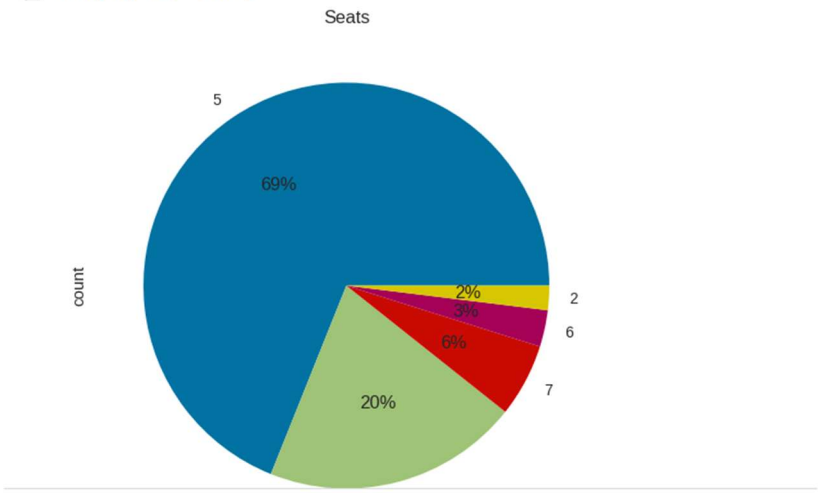
```
Text(0.5, 1.0, 'Body Style')
```



Number of Seats

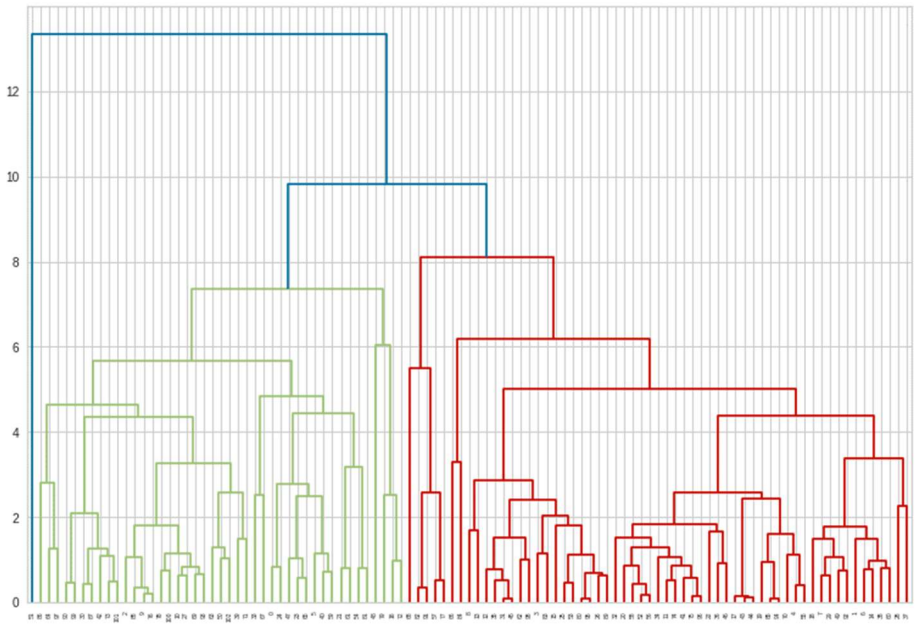
```
[64] df['Seats'].value_counts().plot.pie(figsize=(6,20),autopct='%.0f%%')
     plt.title('Seats')
```

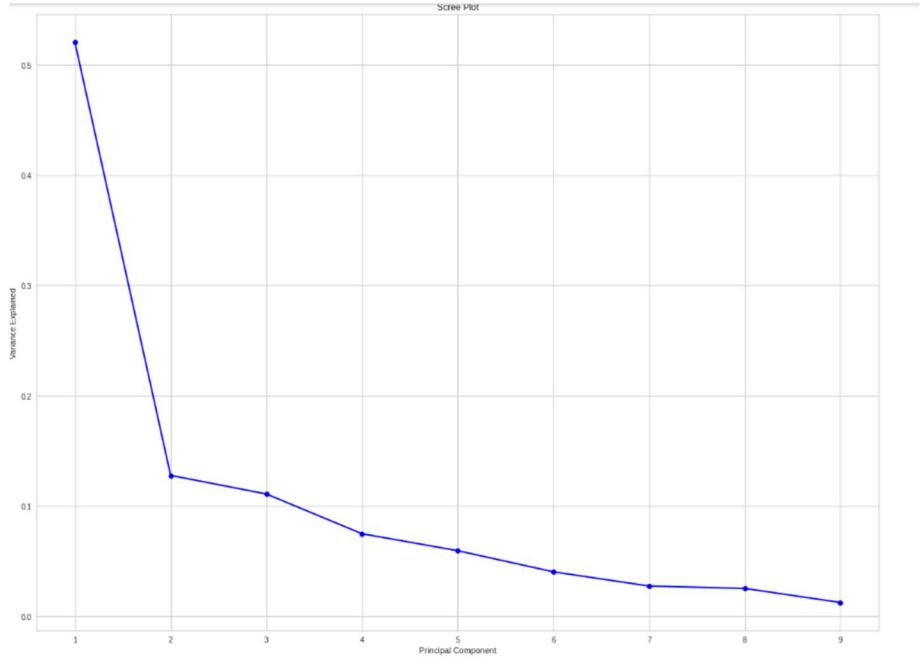
```
Text(0.5, 1.0, 'Seats')
```



Dendrogram:

This technique is specific to the agglomerative hierarchical method of clustering. The agglomerative hierarchical method of clustering starts by considering each point as a separate cluster and starts joining points to clusters in a hierarchical fashion based on their distances. To get the optimal number of clusters for hierarchical clustering, we make use of a dendrogram which is a tree-like chart that shows the sequences of merges or splits of clusters. If two clusters are merged, the dendrogram will join them in a graph and the height of the join will be the distance between those clusters. As shown in Figure, we can choose the optimal number of clusters based on hierarchical structure of the dendrogram. As highlighted by other cluster validation metrics, four to five clusters can be considered for the agglomerative hierarchical as well.





Clustering:

Clustering is one of the most common exploratory data analysis techniques used to get an intuition about the structure of the data. It can be defined as the task of identifying subgroups in the data such that data points in the same subgroup (cluster) are very similar while data points in different clusters are very different. In other words, we try to find homogeneous subgroups within the data such that data points in each cluster are as similar as possible according to a similarity measure such as Euclidean based distance or correlation-based distance. The decision of which similarity measure to use is application-specific. Clustering analysis can be done on the basis of features where we try to find

subgroups of samples based on features or on the basis of samples where we try to find subgroups of features based on samples

K-means algorithm

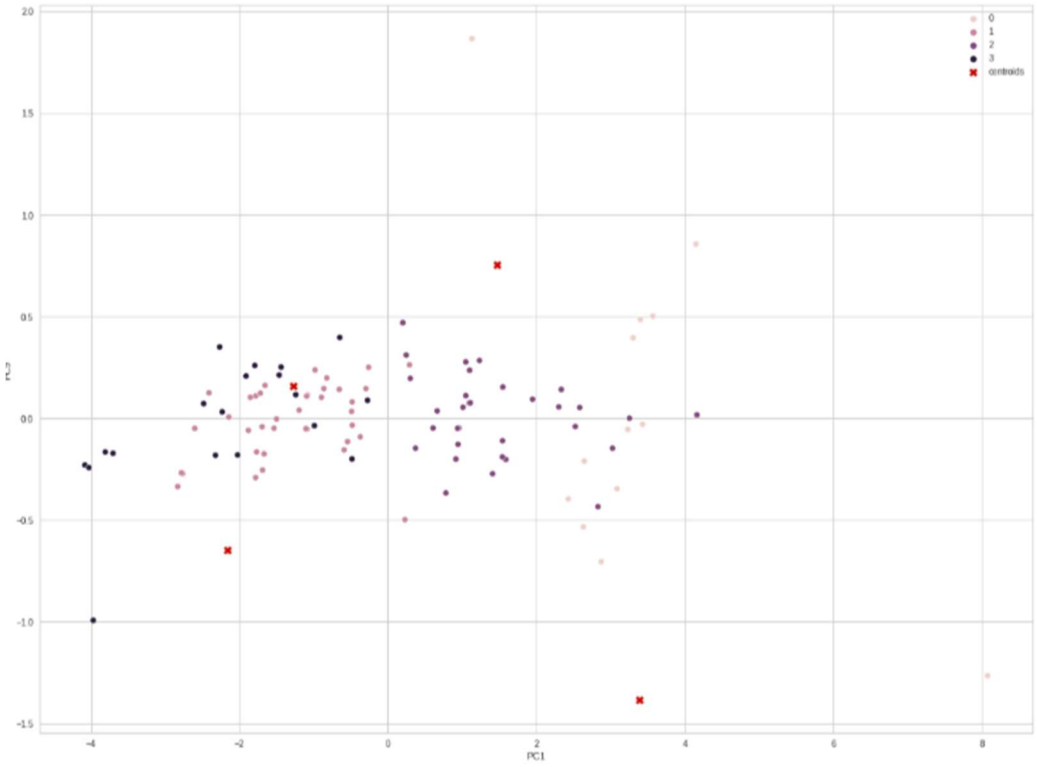
K Means algorithm is an iterative algorithm that tries to partition the dataset into pre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to only one group. It tries to make the intra-cluster data points as similar as possible while also keeping the clusters as different (far) as possible. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum. The less variation we have within clusters, the more homogeneous (similar) the data points are within the same cluster.

The way k means algorithm works is as follows:

- Specify number of clusters K.
- Initialize centroids by first shuffling the dataset and then randomly selecting K data points for the centroids without replacement.
- Keep iterating until there is no change to the centroids. i.e assignment of data points to clusters isn't changing.

```
#K-means clustering

kmeans = KMeans(n_clusters=4, init='k-means++', random_state=0).fit(t)
df['cluster_num'] = kmeans.labels_ #adding to df
print (kmeans.labels_) #Label assigned for each data point
print (kmeans.inertia_) #gives within-cluster sum of squares.
print(kmeans.n_iter_) #number of iterations that k-means algorithm runs to get a minimum within-cluster sum of squares
print(kmeans.cluster_centers_) #Location of the centroids on each cluster.
```



Prediction of prices most used cars

Linear regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models targets prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Here we use a linear regression model to predict the prices of different Electric cars in different companies. X contains the independent variables and y is the dependent Prices that is to be predicted. We train our model with a splitting of data into a 4:6 ratio, i.e. 40% of the data is used to train the model.

```
X=data2[['PC1', 'PC2','PC3','PC4','Pc5','PC6', 'PC7','PC8','PC9']]
y=df['inr(10e3)']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.4, random_state=101)
lm=LinearRegression().fit(X_train,y_train)
```

```
print(lm.intercept_)
```

```
4643.522050485438
```

```
lm.coef_
```

```
array([ 1101.58721,  -741.20904,   208.53617,   508.32246,   122.3533 ,
        1579.00686,   333.61147, -1079.99512,  1461.72269])
```

```
X_train.columns
```

```
Index(['PC1', 'PC2', 'PC3', 'PC4', 'Pc5', 'PC6', 'PC7', 'PC8', 'PC9'], dtype='object')
```

```
cdf=pd.DataFrame(lm.coef_, X.columns, columns=['Coeff'])
cdf
```

	Coeff
PC1	1101.5872
PC2	-741.2090
PC3	208.5362
PC4	508.3225
Pc5	122.3533
PC6	1579.0069
PC7	333.6115
PC8	-1079.9951
PC9	1461.7227

```
print('MAE:',metrics.mean_absolute_error(y_test,predictions))
print('MSE:',metrics.mean_squared_error(y_test,predictions))
print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test,predictions)))
```

```
MAE: 1.6674069532503684e-12
MSE: 4.854762404626698e-24
RMSE: 2.2033525375270063e-12
```

```
metrics.mean_absolute_error(y_test,predictions)
```

```
1.6674069532503684e-12
```

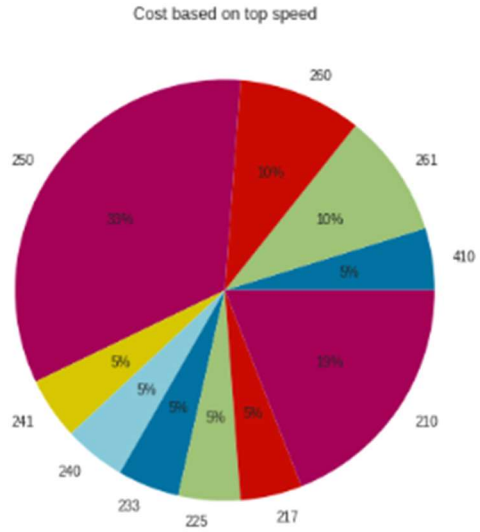
```
metrics.mean_squared_error(y_test,predictions)
```

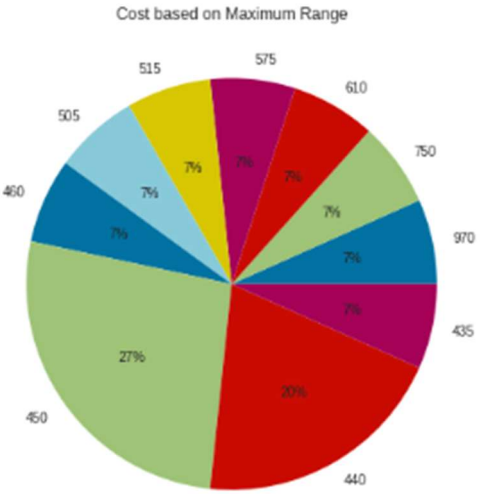
```
4.854762404626698e-24
```

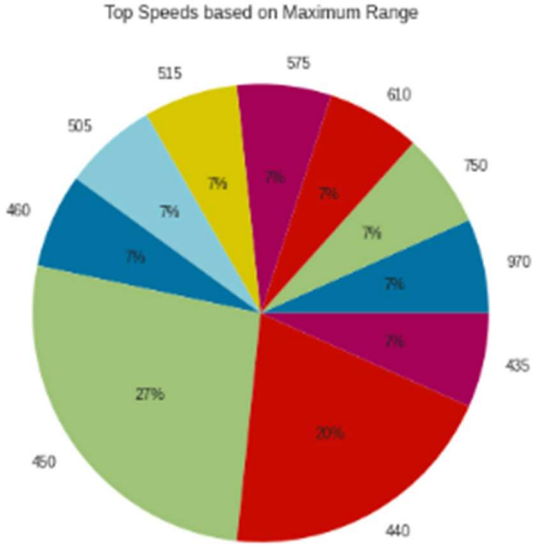
```
np.sqrt(metrics.mean_squared_error(y_test,predictions))
```

```
2.2033525375270063e-12
```

Profiling and Describing the Segments







Target Segments

From the analysis we can see that optimum targeted segment should belong to the following categories

Behavioural: most of the cars are having 5 seats

Demographic:

Top speed and range – with a large area of market the cost is dependent on top speed and maximum range of cars

Efficiency – mostly the segments are with most efficiency

Psychographic:

Price: from above analysis the price range between 16,00,000 to 1,80,00,000

Finally, our target segment should contain cars with most Efficiency, contains Top Speed and price between 16 to 180 lakhs with mostly with 5 seats.