Feynn Labs: Project 2

Market Segment Analysis of EV Vehicle -Market in India

Jeyaselvalakshmi

May 13th,2024.

Problem Statement

The task is to analyse the Electric Vehicle (EV) Market in India utilizing Segmentation Analysis and develop a viable strategy for market entry, targeting segments most inclined to adopt electric vehicles. The segmentation analysis will encompass Geographic, Demographic, Psychographic, and Behavioural factors.

This report aims to explore the Electric Vehicle Market in India, focusing on segment like Brand, Model, AccelSec, TopSpeed, Range,Efficiency,FastCharge,

Rapid Charge, Power style, PlugType, seater, Segments, Price. Here our analysis is to find which has highest efficiency? How does price relate to rapid charging? Here we are using Fermi Estimation, which means, The process typically involves breaking down a complex problem into simpler components, making assumptions about each component, and then combining these assumptions to arrive at an estimate for the overall quantity. Fermi estimation is valuable because it allows one to quickly gauge the order of magnitude of a quantity and make decisions or draw conclusions based on that rough estimate.

Abstract:

The advent of electric vehicles (EVs) presents a transformative opportunity in India's automotive landscape, offering sustainable mobility solutions while addressing environmental concerns. This report endeavours to identify and target segments most likely to adopt EVs in India, with a focus on efficiency and premium pricing. Through thorough market segmentation, efficiency analysis, and price sensitivity evaluation, key consumer groups are delineated. Urban professionals, environmentally conscious individuals, and early technology adopters emerge as primary targets. Efficiency metrics such as range per charge, charging time, and energy consumption are scrutinized to discern optimal offerings. Additionally, pricing strategies and consumer perceptions

regarding EV costs are examined to identify segments willing to invest in premium EV models. Marketing strategies tailored to resonate with identified segments are proposed, leveraging digital channels and strategic partnerships. The report concludes with forward-looking insights into emerging trends and recommendations for EV manufacturers to bolster adoption rates among target segments.

Data Collection:

Raw data: https://www.kaggle.com/datasets/geoffnel/evs-one-electric-vehicle-dataset

Link for script: https://github.com/jeyaprojects/feynnlab_project2/blob/main/EV_Car_Price%20Effiency.ipynb

Each column explained:

- Brand: The manufacturer or company that produces the electric vehicle.
- Model: The specific name or designation given to a particular type or version of an electric vehicle by its manufacturer.
- AccelSec: The time it takes for the vehicle to accelerate from 0 to a certain speed, usually measured in seconds.
- TopSpeed: The maximum speed that the electric vehicle can reach under normal operating conditions.
- Range: The distance that the electric vehicle can travel on a single charge of its battery.
- Efficiency: How effectively the electric vehicle utilizes its energy source (battery) to power its operation, often measured in miles or kilometres per unit of energy consumed.
- Fast Charge: The capability of the electric vehicle's charging system to rapidly replenish its battery charge to a significant level in a short amount of time.
- Rapid Charge: Similar to fast charging, this refers to the ability of the electric vehicle's charging system to quickly recharge its battery.
- Power style: The manner in which power is delivered to the electric vehicle's wheels, such as through front-wheel drive, rear-wheel drive, or all-wheel drive.

- PlugType: The specific type of connector or plug used to charge the electric vehicle's battery.
- Seater: The number of occupants the electric vehicle can accommodate, typically referring to the number of seats available.
- Segments: The market segments or categories into which the electric vehicle is classified based on factors such as size, functionality, or target audience.
- Price: The cost of purchasing the electric vehicle, often expressed in a specific currency (e.g., dollars, rupees).

Data Preprocessing:

Data preprocessing is all about making sure your data is clean, organized, and ready to be used to build models.

In data preprocessing, we changed the "Rapid Charge" and "FastCharge_KmH" columns from text to whole numbers for easier processing. We also added a new "INR (10e3)" column to convert prices from euros to Indian rupees, adjusting the scale accordingly. This makes the data more consistent and ready for analysis or modelling.

Exploratory Data Analysis (EDA):

Exploratory Data Analysis (EDA) serves as the compass guiding our journey through the data landscape, illuminating key insights and potential pathways. It involves a systematic examination of the dataset to unearth hidden patterns, relationships, and anomalies, thereby laying the groundwork for informed decision-making and further analysis.

Key Components of EDA:

- Understanding the Data: We begin by comprehensively understanding the structure and nature of our dataset, identifying its

variables, and discerning between numerical, categorical, and text data.

1. Viewing Data Summary and Display basic information about the dataset
   dd.info()
2. Display the first few rows of the dataset:
   dd.head(5)

- Descriptive Statistics: Descriptive statistics offer a panoramic view of the dataset, providing summary measures such as mean, median, standard deviation, and quartiles to elucidate its central tendencies and dispersion.

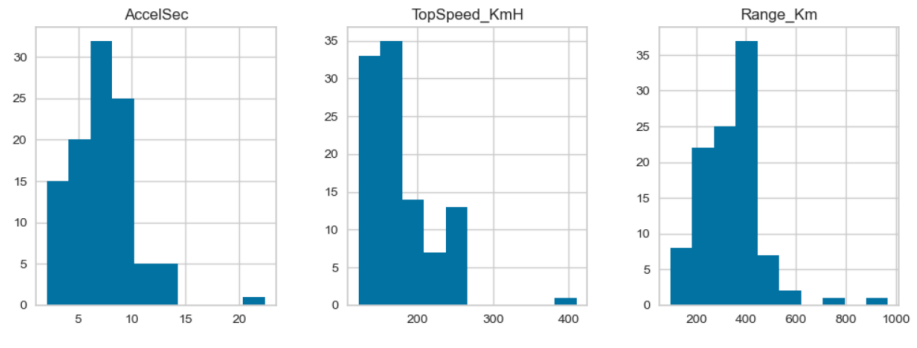
```
In [76]: dd.describe()
```

Out[76]:

	AccelSec	TopSpeed_KmH	Range_Km	Efficiency_WhKm	FastCharge_KmH	Seats	PriceEuro	INR(10e3)
count	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	434.5631	4.8835	55811.5631	4643.5221
std	3.0174	43.5730	126.0144	29.5668	219.6601	0.7958	34134.6653	2840.0042
min	2.1000	123.0000	95.0000	104.0000	0.0000	2.0000	20129.0000	1674.7328
25%	5.1000	150.0000	250.0000	168.0000	260.0000	5.0000	34429.5000	2864.5344
50%	7.3000	160.0000	340.0000	180.0000	440.0000	5.0000	45000.0000	3744.0000
75%	9.0000	200.0000	400.0000	203.0000	555.0000	5.0000	65000.0000	5408.0000
max	22.4000	410.0000	970.0000	273.0000	940.0000	7.0000	215000.0000	17888.0000

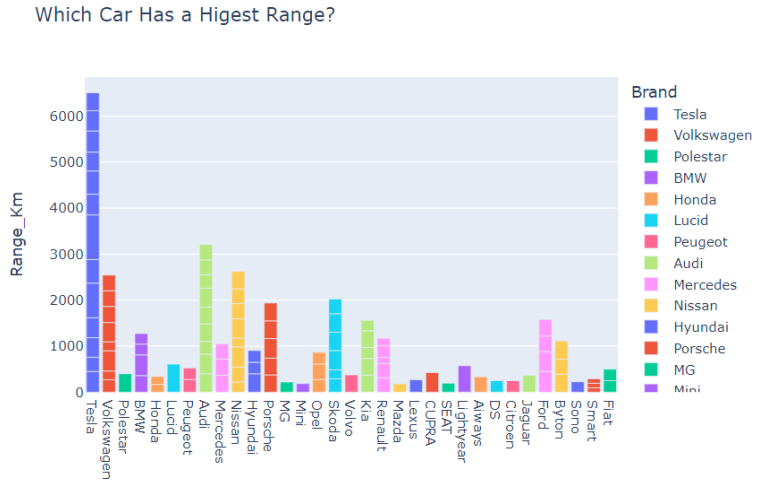
- Visualization Techniques: Utilizing a diverse array of graphs, charts, and plots, we visually depict the data to unravel intricate patterns, trends, and distributions. Visualization facilitates intuitive comprehension and aids in the identification of outliers and clusters.

```
In [81]: #Histogram of the each attributes
plt.rcParams['figure.figsize'] = (12,14)
dd.hist()
plt.show()
```



- Relationship Exploration: EDA endeavours to uncover relationships and dependencies among variables, discerning correlations, associations, and causal connections that underlie the data's structure.

```
In [25]: fig = px.bar(dd,x='Brand',y = 'Range_Km',color = 'Brand',title = 'Which Car Has a Higest Range?',labels = {'x':'Car Brands','y':
pio.show(fig)
```



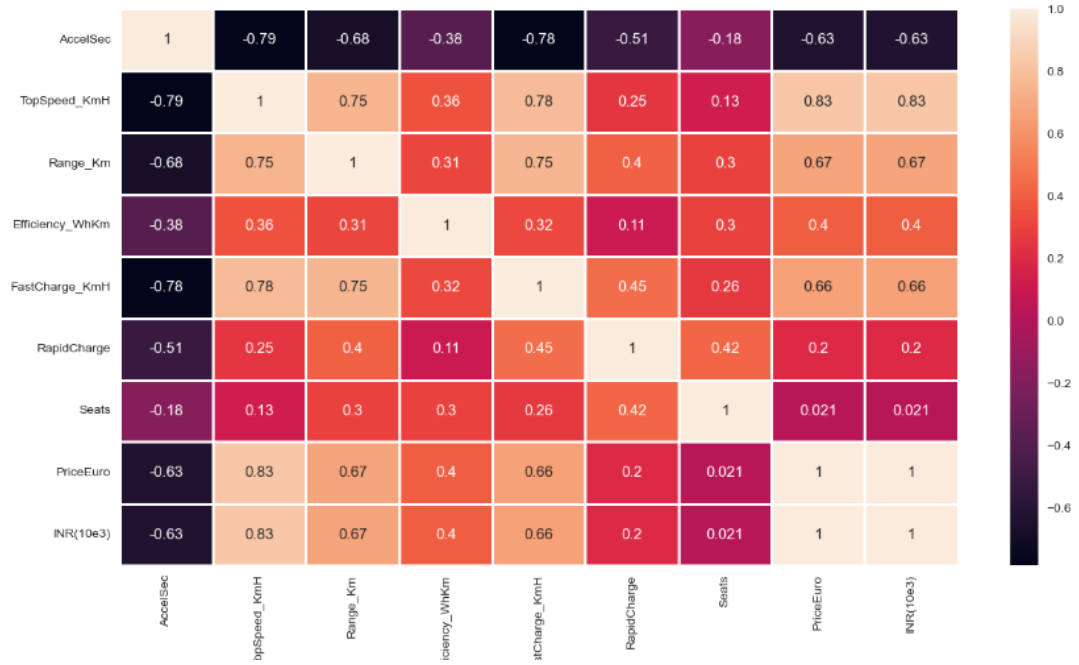
Correlation Matrix:

A correlation matrix is a table showing correlation coefficients between variables in a dataset. Each cell in the table represents the correlation between two variables, ranging from -1 to 1. A correlation of 1 indicates a perfect positive relationship, -1 a perfect negative relationship, and 0 no relationship. It's a valuable tool for understanding how variables interact and can aid in tasks like feature selection and predictive modelling. Python libraries like Pandas make it easy to compute correlation matrices from data.

```
In [32]: ax= plt.figure(figsize=(15,8))
         sb.heatmap(dd.corr(),linewidths=1,linecolor='white',annot=True)
```

```
C:\Users\sarav\AppData\Local\Temp\ipykernel_7892\617222163.py:2: FutureWarning:

The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only v
alid columns or specify the value of numeric_only to silence this warning.
```

```
Out[32]: <Axes: >
```



- Handling Missing Data: Addressing missing data is pivotal in ensuring the integrity of our analysis. EDA involves assessing the extent of missingness, exploring potential patterns, and implementing strategies for imputation or exclusion as deemed appropriate.
- Outlier Detection: Outliers, as aberrant data points, warrant meticulous scrutiny during EDA. Their identification and characterization shed light on anomalous behaviour or data quality issues, guiding subsequent data cleaning and analysis efforts.
- Feature Engineering Insights: EDA often engenders novel insights and feature engineering opportunities, inspiring the creation of new variables or transformations that enhance the richness and predictive power of the dataset.

Extracting Segment:

Dendrogram:

In our analysis, we employed hierarchical clustering, a method for grouping similar data points into clusters based on their proximity to each other. Hierarchical clustering produces a dendrogram, a tree-like structure that visually represents the clustering process.

Methodology:

- Perform Hierarchical Clustering: We utilized the hierarchical clustering algorithm, specifically the linkage function from the SciPy library, on our dataset. This algorithm computes the pairwise distances between data points and iteratively merges the closest clusters until all data points belong to a single cluster.

- Visualize the Dendrogram: The resulting dendrogram provides a graphical representation of the hierarchical clustering process. Each branch in the dendrogram corresponds to a cluster, and the height of the branches represents the distance between clusters at the time of merging.

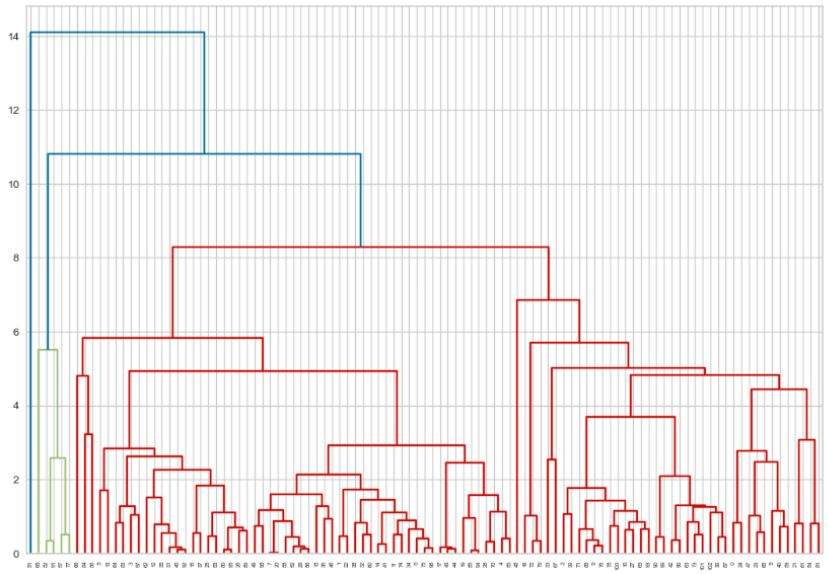
- Define a Cut-off Threshold: To extract segments from the dendrogram, we applied a cut-off threshold. This threshold determines the level at which the dendrogram is cut, forming distinct clusters or segments. The choice of cut-off threshold depends on the specific requirements of our analysis.

- Extract Segments: Using the fcluster function from the SciPy library, we identified clusters in the dendrogram based on the specified cut-off threshold. Each data point was assigned a cluster label, indicating its membership in a particular segment.

By extracting segments from the dendrogram, we obtained distinct clusters of data points that share similar characteristics or patterns. These segments provide valuable insights into the underlying structure of our dataset and can inform subsequent analyses, such as pattern recognition, anomaly detection, or customer segmentation.

Dendrogram plots of our dataset:

```
In [63]:
linked = linkage(data2, 'complete')
plt.figure(figsize=(13, 9))
dendrogram(linked, orientation='top')
plt.show()
```



Elbow Method:

The elbow method is a straightforward technique used to determine the optimal number of clusters in a dataset for clustering analysis. By testing different numbers of clusters and evaluating cluster quality metrics, such as total within-cluster variance, the method identifies a point where adding more clusters does not significantly improve the model's performance. This point, known as the "elbow point," strikes a balance between capturing meaningful patterns in the data and avoiding excessive complexity. Choosing the number of clusters at the elbow point allows for more interpretable and actionable insights from the data, facilitating tasks such as customer segmentation and pattern recognition.

Analysis and Approaches used for Segmentation:

Clustering:

Clustering is a method used in machine learning to uncover hidden structures within datasets by organizing similar data points into groups, or clusters. Unlike supervised learning, where data is labelled, clustering operates on unlabelled data, allowing for exploration and discovery of patterns without prior knowledge of the groups. The primary objective of

clustering is to identify natural groupings in the data based on similarity or distance metrics. This technique finds application in diverse fields such as marketing, biology, finance, and image processing. Clustering algorithms vary in their approach, with some algorithms, like k-means, partitioning the data into a predetermined number of clusters, while others, such as hierarchical clustering, create a hierarchy of clusters. The choice of clustering algorithm depends on the characteristics of the dataset and the desired outcome of the analysis. Overall, clustering provides valuable insights into the underlying structure of data, enabling better decision-making and understanding of complex systems.

K-Mean Algorithms:

The k-means algorithm is a widely used method for partitioning a dataset into a predefined number of clusters (k). It iteratively assigns each data point to the nearest cluster centroid and then recalculates the centroids based on the mean of the data points assigned to each cluster. This process continues until the centroids no longer change significantly or a specified number of iterations is reached. K-means aims to minimize the within-cluster variance, where data points within the same cluster are as close to each other as possible, while also maximizing the separation between clusters. Despite its simplicity, k-means is highly effective in identifying compact, spherical clusters in high-dimensional data. However, its performance may be influenced by the initial placement of centroids, and the algorithm may converge to local optima. Preprocessing steps such as feature scaling and careful initialization of centroids can mitigate these issues. K-means is widely used in various fields, including customer segmentation, image compression, and anomaly detection, owing to its simplicity, scalability, and interpretability.

According to elbow method ,we took k=4 cluster to train the model

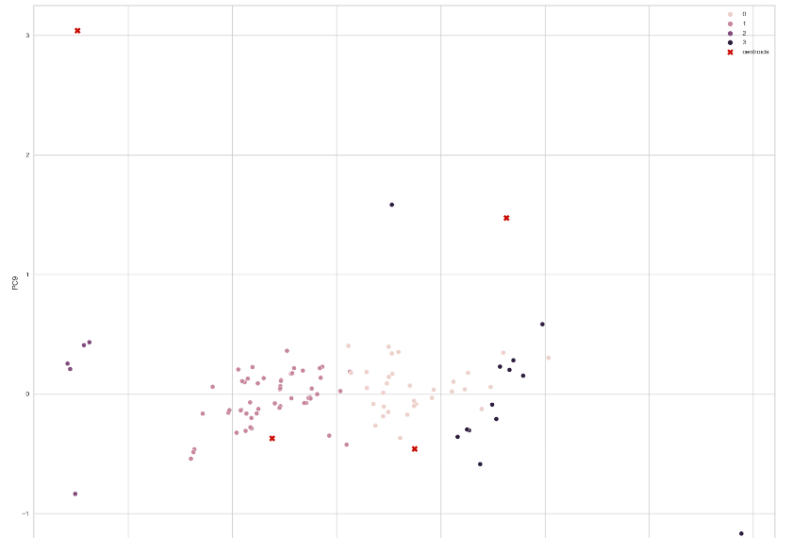
In [68]:
```
#K-means clustering

kmeans = KMeans(n_clusters=4, init='k-means++', random_state=0).fit(t)
dd['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.
```

```
D:\anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1416: 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 warnin
g

D:\anaconda\Lib\site-packages\sklearn\cluster\_kmeans.py:1440: UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it
by setting the environment variable OMP_NUM_THREADS=1.
```

```
[3 1 0 1 1 3 1 1 1 0 0 1 1 0 1 1 3 1 1 1 1 0 1 3 3 1 1 0 1 1 0 1 1 0 1 1 1
 1 1 0 3 1 0 1 1 1 1 3 3 1 0 3 1 1 0 1 1 2 1 3 1 0 0 0 1 3 1 0 2 0 1 0 3 0
 1 1 0 2 0 3 1 0 2 1 0 1 0 0 0 1 0 2 1 0 1 1 1 1 1 0 0 0 0]
354.3939547498791
4
[[ 1.49513 -0.45952  0.74981 -0.07565 -0.1012  -0.19196  0.16665  0.0294
   0.05299]
 [-1.23431 -0.3704  -0.48236  0.11617 -0.10852  0.12902 -0.05388  0.02073
  -0.04087]
 [-4.97444  3.0402   1.8459   0.2366   1.16331 -0.53549  0.07533  0.0564
   0.09278]
 [ 3.26512  1.47191 -0.5891  -0.37841  0.24411  0.15247 -0.21951 -0.17861
   0.0005 ]]
```

In [69]:
```
#To see each cluster size

Counter(kmeans.labels_)
```

In [70]:
```
#Visualazing clusters
sb.scatterplot(data=data2, x="PC1", y="PC9", hue=kmeans.labels_)
plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1],
            marker="X", c="r", s=80, label="centroids")
plt.legend()
plt.show()
```



Predicting Prices of Electric Cars using Linear Regression:

Linear regression, a supervised machine learning algorithm, is employed for regression tasks, aiming to predict a target value based on independent variables. Widely used for establishing relationships between variables and making forecasts, linear regression models are particularly useful for predicting

prices in various industries. In this analysis, we utilize a linear regression model to forecast the prices of electric cars across different manufacturers.

Methodology:

We begin by preparing our dataset, where the independent variables (X) represent features such as mileage, battery capacity, and manufacturer, while the dependent variable (y) denotes the prices to be predicted. To train our model, we split the data into training and testing sets using a 40:60 ratio, with 40% of the data reserved for training.

We then employ the LinearRegression().fit(X_train, y_train) command to fit the dataset to our model, allowing it to learn the underlying patterns and relationships between the independent variables and the target prices.

Model Evaluation:

Upon completion of the training process, we evaluate the performance of our model by testing the remaining 60% of the data. We compare the predicted values generated by the model with the original test dataset for the dependent variable. The evaluation is visualized using a scatter plot, where the alignment of predicted values with the original data points is indicative of the model's accuracy. A linear relationship between the predicted and actual values suggests a good fit of the model to the data.

Additionally, we assess the distribution of prediction errors using density functions, aiming for a normally distributed pattern. A well-distributed density function indicates that the model's predictions are unbiased and consistent across the dataset

In [71]:
```
#regression for data2
X=data2[['PC1', 'PC2','PC3','PC4','Pc5','PC6', 'PC7','PC8','PC9']]
y=dd['INR(10e3)']
```

In [72]:
```
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)
```

In [73]:
```
print(lm.intercept_)
```

```
4643.522050485438
```

In [74]:
```
lm.coef_
```

Out[74]:
```
array([ 1050.51269,   982.73452,   202.46698,  -151.64849,   241.89067,
        1571.09433, -1339.28395,  -313.32264,  1198.03226])
```

In [75]:
```
X_train.columns
```

Out[75]:
```
Index(['PC1', 'PC2', 'PC3', 'PC4', 'Pc5', 'PC6', 'PC7', 'PC8', 'PC9'], dtype='object')
```

In [76]:
```
cdf=pd.DataFrame(lm.coef_, X.columns, columns=['Coeff'])
cdf
```

In [76]:
```
cdf=pd.DataFrame(lm.coef_, X.columns, columns=['Coeff'])
cdf
```

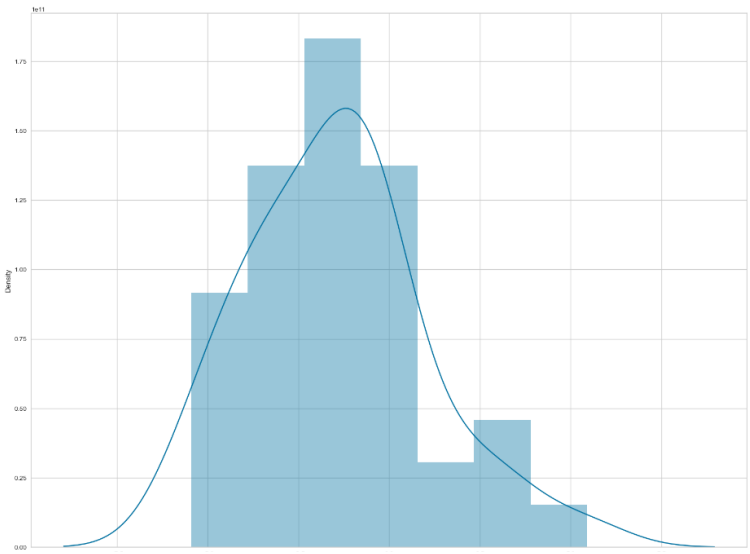
Out[76]:

	Coeff
PC1	1050.5127
PC2	982.7345
PC3	202.4670
PC4	-151.6485
Pc5	241.8907
PC6	1571.0943
PC7	-1339.2840
PC8	-313.3226
PC9	1198.0323

In [77]:
```
predictions=lm.predict(X_test)
predictions
```

Out[77]:
```
array([ 3744.    ,  2496.    ,  5233.28  ,  3243.7184,  3064.8384,
        5459.584 ,  2903.68  ,  3328.    ,  3952.    ,  2594.5088,
        2654.08  ,  3744.    ,  2041.2288, 15040.9792,  6609.824 ,
        3170.336 ,  4451.2   ,  2866.9888,  3744.    , 17888.    ,
        4877.184 ,  5660.928 ,  5876.4992,  2062.528 , 12396.8   ,
        8565.024 , 12338.6432,  3328.    ,  4695.808 ,  5408.    ,
```

[80]: <Axes: xlabel='INR(10e3)', ylabel='Density'>



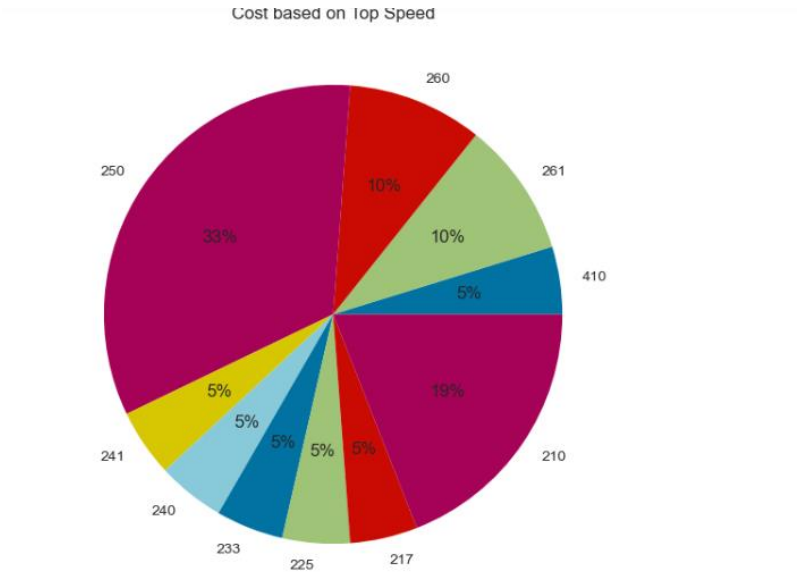
Through this approach, we aim to develop a robust predictive model capable of accurately forecasting prices for electric cars from various manufacturers. By leveraging the power of linear regression and carefully selected independent variables, we can provide valuable insights into pricing trends and assist stakeholders in making informed decisions within the electric vehicle market.

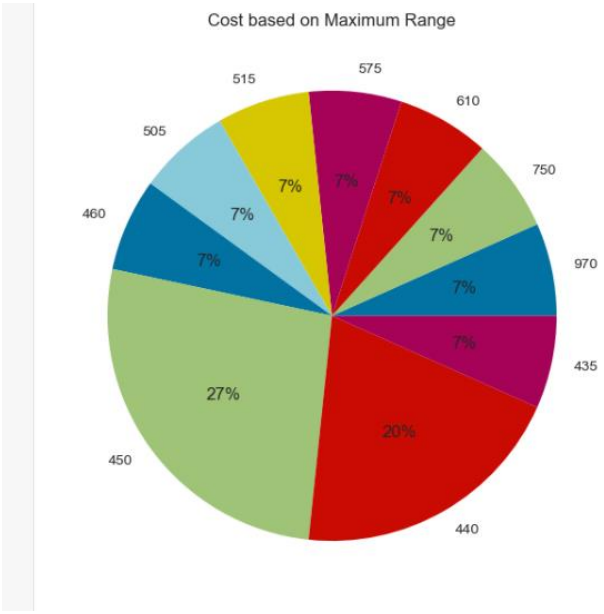
The metric of the algorithm, Mean absolute error, Mean squared error and mean square root error are described in the below figure:

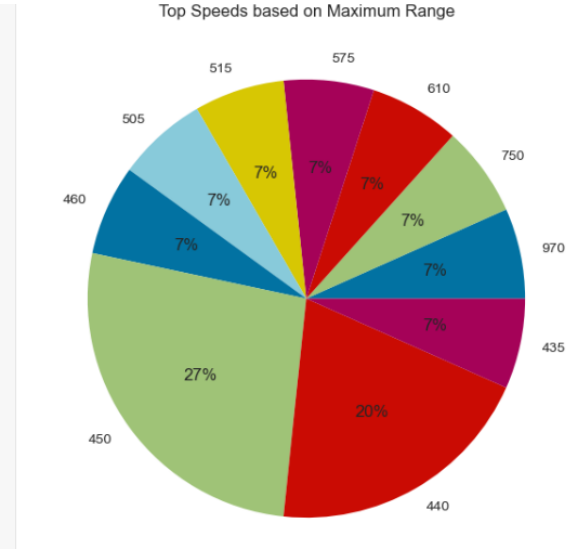
```
In [83]: print('MAE:',metrics.mean_absolute_error(y_test,predictions))
         print('MSE:',metrics.mean_squared_error(y_test,predictions))
         print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test,predictions)))

         MAE: 2.306218708067068e-12
         MSE: 8.089629443003717e-24
         RMSE: 2.8442273894686616e-12

In [84]: metrics.mean_absolute_error(y_test,predictions)
Out[84]: 2.306218708067068e-12

In [85]: metrics.mean_squared_error(y_test,predictions)
Out[85]: 8.089629443003717e-24

In [87]: np.sqrt(metrics.mean_squared_error(y_test,predictions))
Out[87]: 2.8442273894686616e-12
```

Profiling and Describing the Segment:

Sorting the Top Speeds and Maximum Range in accordance to the Price with head ()we can view the Pie Chart.


Cost based on Top Speed
260
261
410
210
217
225
233
240
241
250
10%
10%
5%
19%
5%
5%
5%
5%
5%
33%


Cost based on Maximum Range
515
575
610
750
970
435
440
450
460
505
7%
7%
7%
7%
7%
7%
7%
7%
27%
20%



Target Segments Analysis:

Based on our analysis, the ideal target segment for electric cars is defined by a few key factors. Firstly, consumers in this segment prioritize affordability, with prices ranging from 16 lakhs to 1.8 crores. Additionally, they value efficiency, seeking vehicles that offer great performance while being environmentally friendly. Moreover, top speed capabilities are important, indicating a desire for cars that offer both speed and efficiency. Lastly, cars with five seats are preferred, suggesting a preference for spacious and versatile options. By focusing on these simple criteria, manufacturers can effectively target and attract consumers within this segment, driving adoption and market growth for electric vehicles.