Electric Vehicle Market

1. Data Cleaning

The data we collected is straightforward and serves two main purposes: creating visual representations and grouping similar items together. We use Python tools like NumPy, Pandas, Scikit-Learn, and SciPy to organize our work, making sure that our results are reliable and can be easily recreated.

```
+ Code  + Text                                                    Connecting   ✦ Gemini

[ ] import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    import seaborn as sns
    import plotly.express as px
    import plotly.io as pio

[ ] df=pd.read_csv("/content/data.csv")

    df.head(4)
```

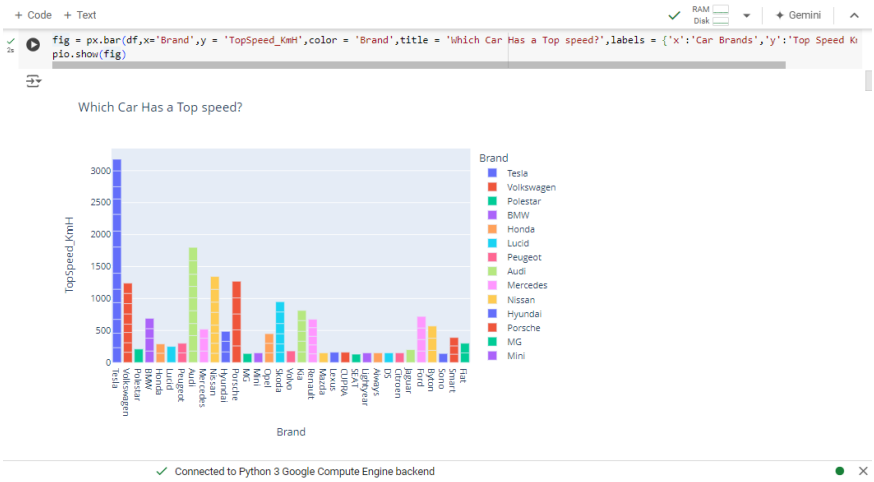
	Unnamed: 0	Brand	Model	AccelSec	TopSpeed_KmH	Range_Km	Efficiency_WhKm	FastCharge_KmH	RapidCharge	PowerTrain	PlugType	BodyStyle
0	0	Tesla	Model 3 Long Range Dual Motor	4.6	233	450	161	940	Yes	AWD	Type 2 CCS	Sedan
1	1	Volkswagen	ID.3 Pure	10.0	160	270	167	250	No	RWD	Type 2 CCS	Hatchback
2	2	Polestar	2	4.7	210	400	181	620	Yes	AWD	Type 2 CCS	Liftback
3	3	BMW	iX3	6.8	180	360	206	560	Yes	RWD	Type 2 CCS	SUV

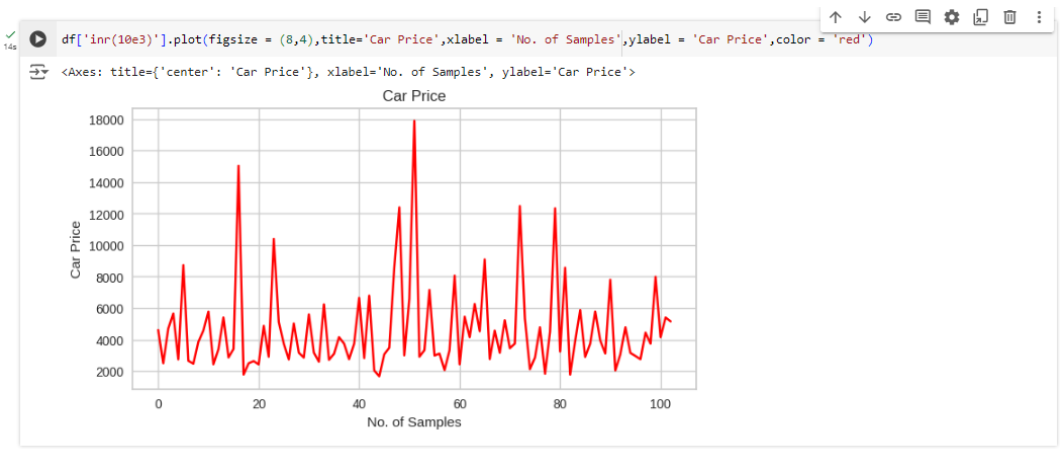
```
Connecting to Python 3 Google Compute Engine backend
```

2. EDA

We begin our Exploratory Data Analysis (EDA) by examining initial insights from our dataset both with and without Principal Component Analysis (PCA). PCA is a statistical technique that transforms correlated features into a new set of linearly uncorrelated ones through orthogonal transformation. These new features, known as Principal Components, reduce the dimensionality of the data. This reduction simplifies tasks such as classification, regression, and other forms of machine learning, making them more efficient and cost-effective.

Comparision of cars in our data

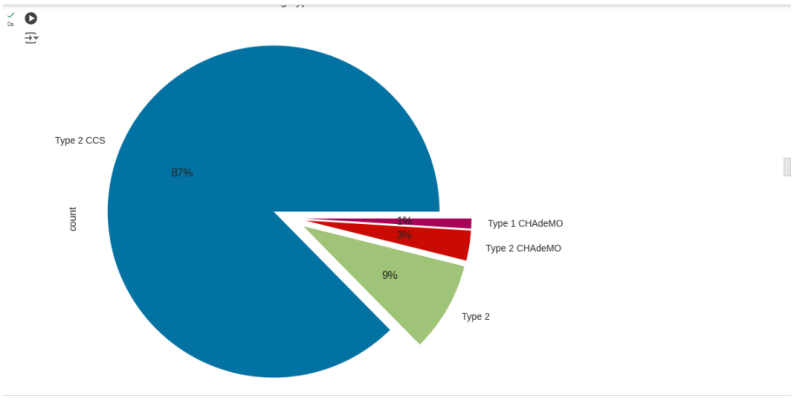


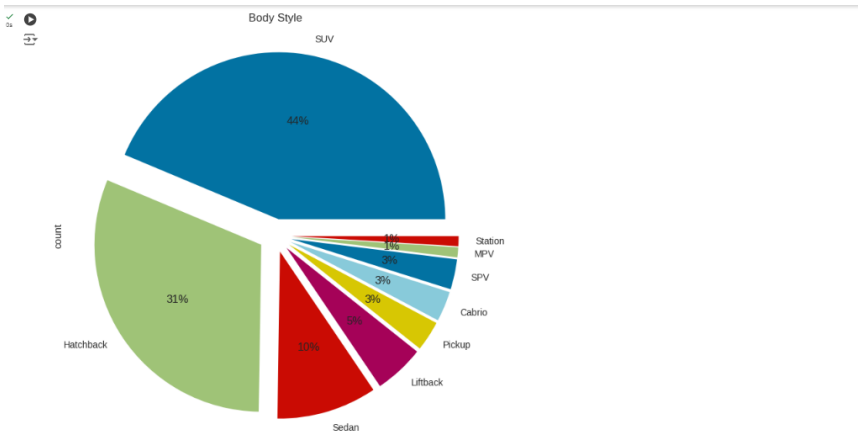


For Electric Vehicle Market one of the most important key is Charging

3. Correlation Matrix

A correlation matrix is simply a table that displays the cor relation. It is best used in variables that demonstrate a linear relationship between each other. Coefficients for different variables. The matrix depicts the correlation be tween 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.



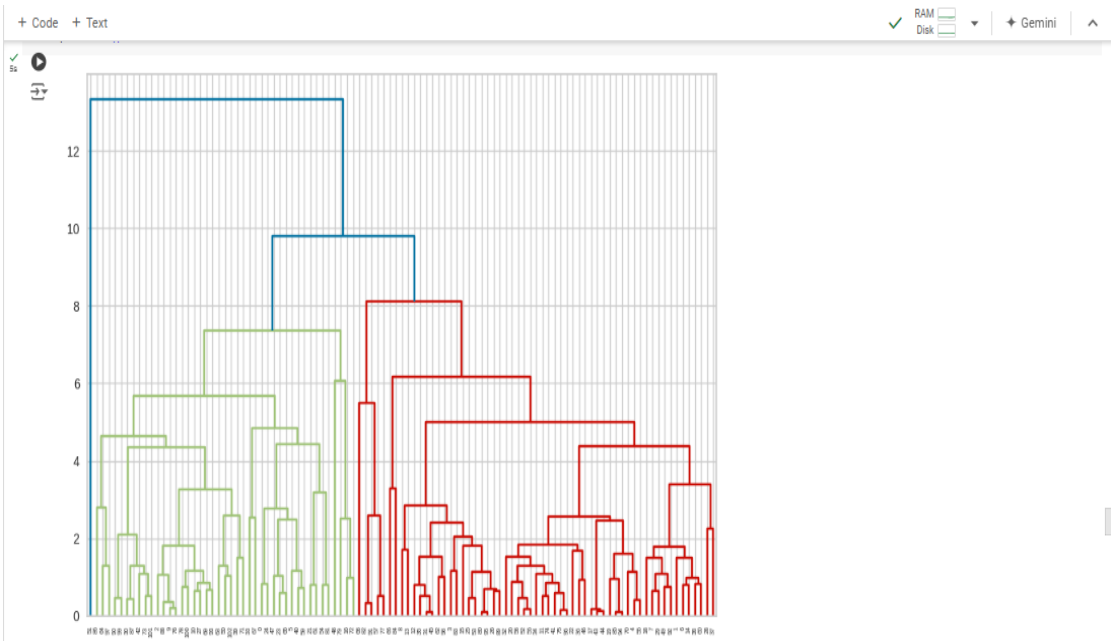


4. Extracting Segments

Dendrogram

This method is specifically tailored for agglomerative hierarchical clustering, which initially treats each data point as its own cluster. Clusters are then sequentially merged based on their distances, forming a hierarchical structure. To determine the optimal number of clusters for this method, we rely on a dendrogram—a tree-like diagram that illustrates the sequence of cluster merges or splits. In a dendrogram, clusters are joined at varying heights, with the height representing the distance between clusters at each merge.

In the accompanying figure, we can visually identify the optimal number of clusters based on the hierarchical structure of the dendrogram. Additionally, other cluster validation metrics suggest that agglomerative hierarchical clustering typically performs well with four to five clusters.



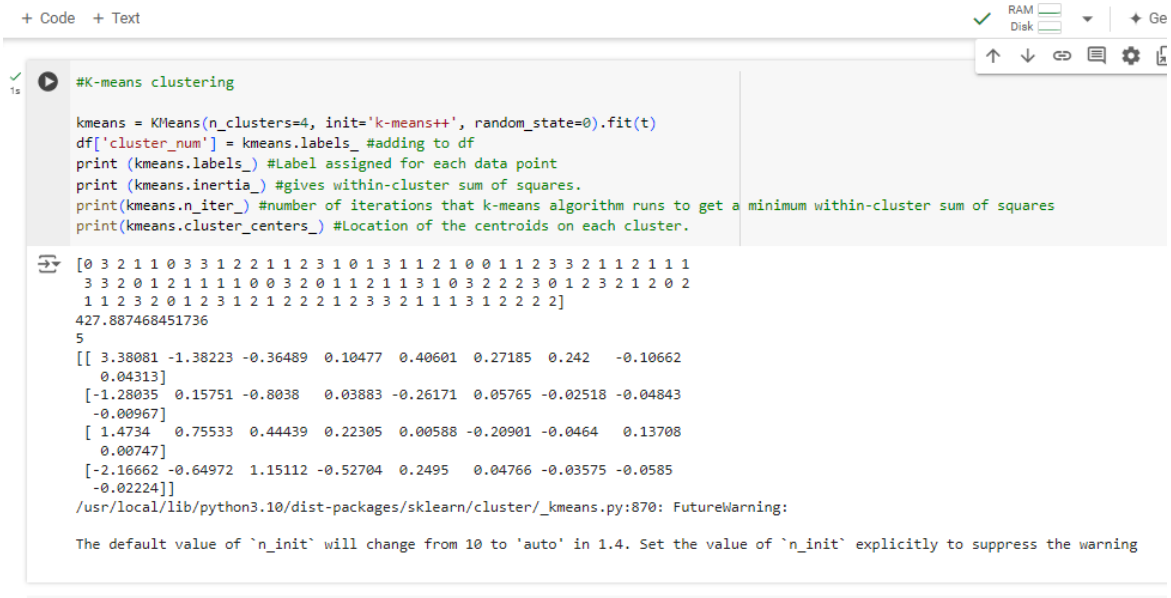
5. Analysis and Approaches used for Segmentation

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 iden tifying 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 be longs 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 homo geneous (similar) the data points are within the same cluster.

The k-means clustering algorithm performs the following tasks:
• Specify number of clusters K
• Initialize centroids by first shuffling the dataset and then randomly selecting K data points for the centroids without replacement.

• Compute the sum of the squared distance between data points and all centroids.
• Assign each data point to the closest cluster (centroid).
• Compute the centroids for the clusters by taking the average of the all data points that belong to each cluster.
• 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.
```

```
[0 3 2 1 1 0 3 3 1 2 2 1 1 2 3 1 0 1 3 1 1 2 1 0 0 1 1 2 3 3 2 1 1 2 1 1 1
 3 3 2 0 1 2 1 1 1 1 0 0 3 2 0 1 1 2 1 1 3 1 0 3 2 2 2 3 0 1 2 3 2 1 2 0 2
 1 1 2 3 2 0 1 2 3 1 2 1 2 2 2 1 2 3 3 2 1 1 1 3 1 2 2 2 2]
427.887468451736
5
[[ 3.38081 -1.38223 -0.36489  0.10477  0.40601  0.27185  0.242   -0.10662
   0.04313]
 [-1.28035  0.15751 -0.8038   0.03883 -0.26171  0.05765 -0.02518 -0.04843
  -0.00967]
 [ 1.4734   0.75533  0.44439  0.22305  0.00588 -0.20901 -0.0464   0.13708
   0.00747]
 [-2.16662 -0.64972  1.15112 -0.52704  0.2495   0.04766 -0.03575 -0.0585
  -0.02224]]
/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
```

6. Prediction of Prices most used cars

Linear regression is a machine learning algorithm based on supervised learning. It per forms a regression task. Regression models targets prediction value based on indepen dent 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. LinearRegression().fit(Xtrain,ytrain) command is used to fit the data set into model. The values of intercept, coefficient, and cumulative distribution function (CDF) are described in the figure.

Regression for data2

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

```
[65] 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)
```

```
[66] print(lm.intercept_)
```

```
4643.522050485438
```

```
[67] lm.coef_
```

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

```
[68] X_train.columns
```

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

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

+ Code + Text

```
[68] 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

Next steps: Generate code with cdf View recommended plots

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