Electrical Vehicle Market in India

Market segmentation

Balla Rakesh, Shubhra Saxena

Electrical Vehicle Market Analysis in India

Introduction

They often say, "Electric vehicles are the future." With the right amount of functionality, they are eco-friendly, and hence it is an excellent choice for people. Today, Tesla, Mercedes Benz, Audi, Hyundai, Mahindra, Chevrolet, BMW, and Renault are manufacturing ground-breaking electric vehicles.Electric vehicles came into existence in the 19th century. Earlier, they did not do that well in the market because of its high cost, low speed, and short-range. So initially, the demand declined worldwide. However, they have been used for transportation and public transport, especially as rail vehicles.

As the concern for the environment increased in the 21st century, gas-powered vehicles emit a lot of smoke and are incredibly harmful to the atmosphere.Therefore, the interest in electric vehicles increased too. Electric cars were popular among those who used them in the city where their short-range did not prove a disadvantage.The significant popularity of electric rickshaws dominated the entire market. In 2016-17 about 500000 e-rickshaws were sold in India. It served as an excellent help for the population to commute daily.

Fast forward to 2021, where companies are working aggressively to develop affordable electric vehicles, with a vision to dispel the notion that electric cars are expensive. For example, E-Trio, an EV company, is working on providing EVs for all in an affordable manner. They focus on cars and other products such as Electric LCV, Electric Bikes, and Electric three-wheelers.

The government in India aims to make arrangements and incentives so that the Battery Electric Vehicles will encourage more people to contribute to 25% of all new vehicle registrations by 2024. Currently, electric vehicles contribute to only 0.29% of the overall registrations in India. For this to happen, a two-prong approach must be set in motion. Firstly, India needs to be genuinely EV-ready by setting up the proper infrastructure and the necessary technology to support electric vehicle manufacturing. Secondly, provisions must be put in place so that the old vehicle can be converted into hybrid electric vehicles through retrofitting to curb the rising pollution levels.

In this report we are going to analyse the data and solve the problem using **Fermi Estimation** by breaking down the problem.

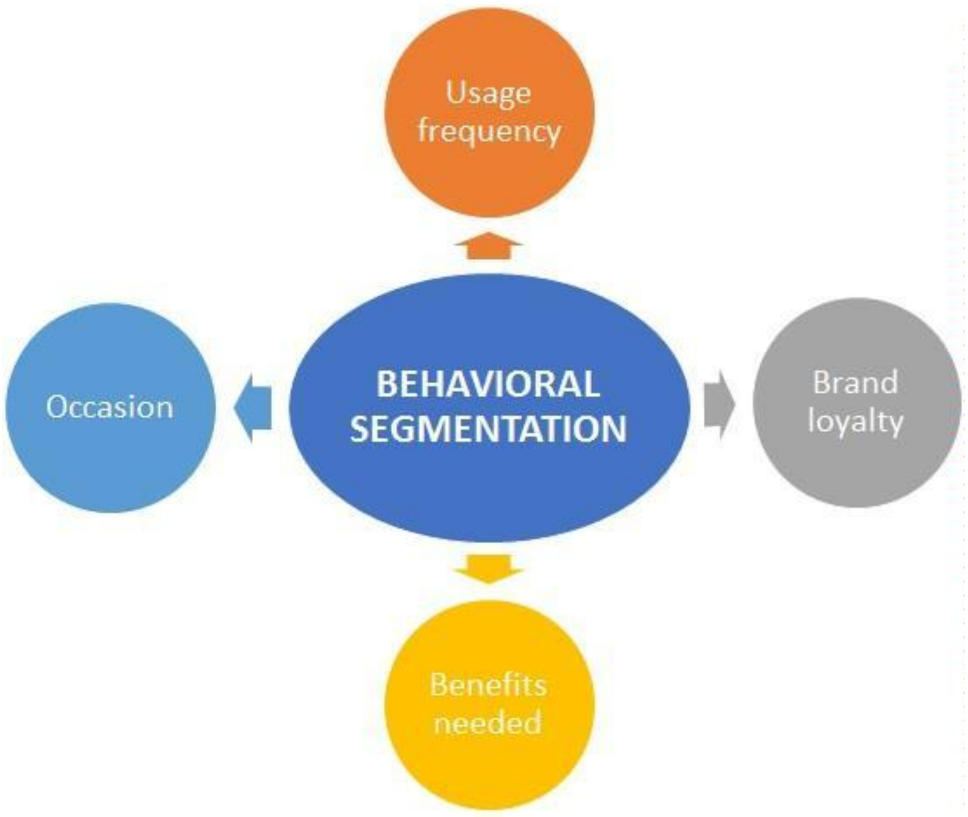
Data Collection

The data has been collected manually, and the sources used for this process are listed below :

- https://www.kaggle.com/datasets
- https://data.gov.in/
- https://data.worldbank.org/
- https://datasetsearch.research.google.com/

Market Segmentation Target Market:

The target market of Electric Vehicle Market Segmentation can be categorised into Geographic, SocioDemographic, Behavioral, and Psychographic Segmentation.



Behavioural Segmentation: searches directly for similarities in behavior or reported behavior.

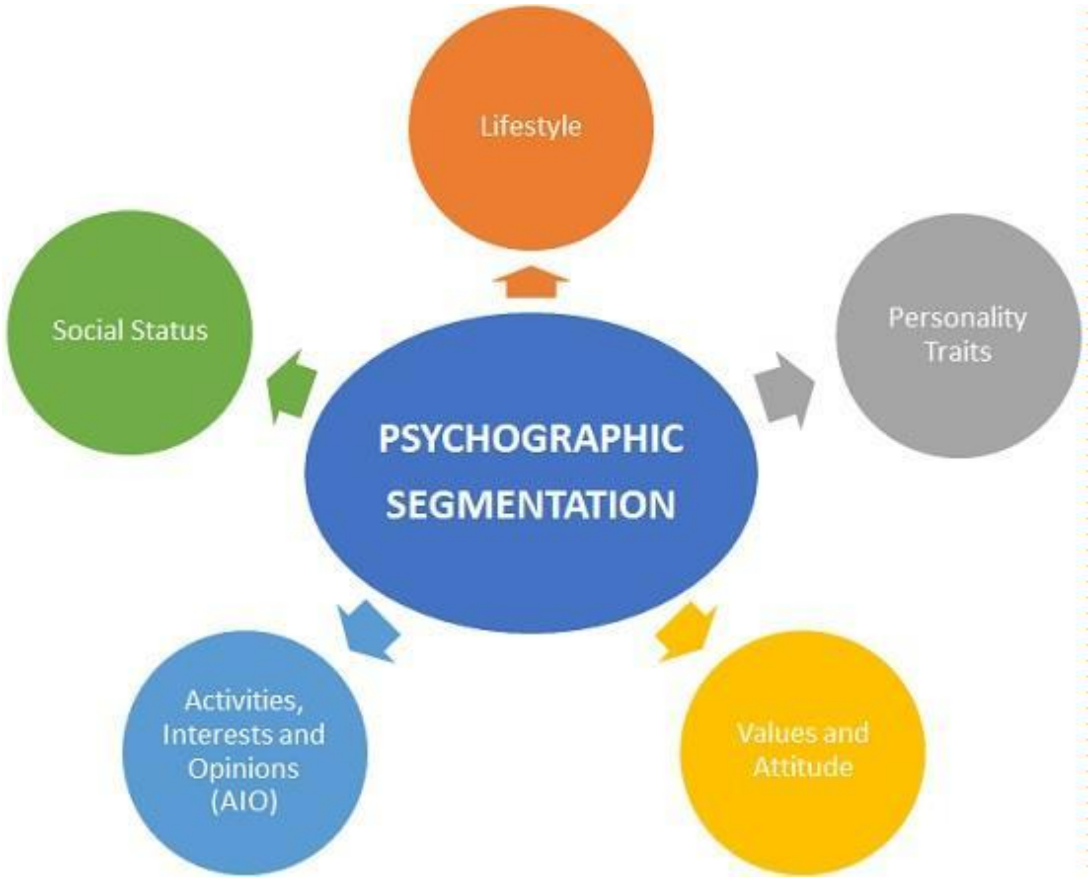
Example: prior experience with the product, amount spent on the purchase, etc.

Advantage: uses the very behavior of interest is used as the basis of segment extraction.
Disadvantage: not always readily available.

Psychographic Segmentation: grouped based on beliefs, interests, preferences, aspirations, or benefits sought when purchasing a product. Suitable for lifestyle segmentation. Involves many segmentation variables.

Advantage: generally more reflective of the underlying reasons for differences in consumer behavior.

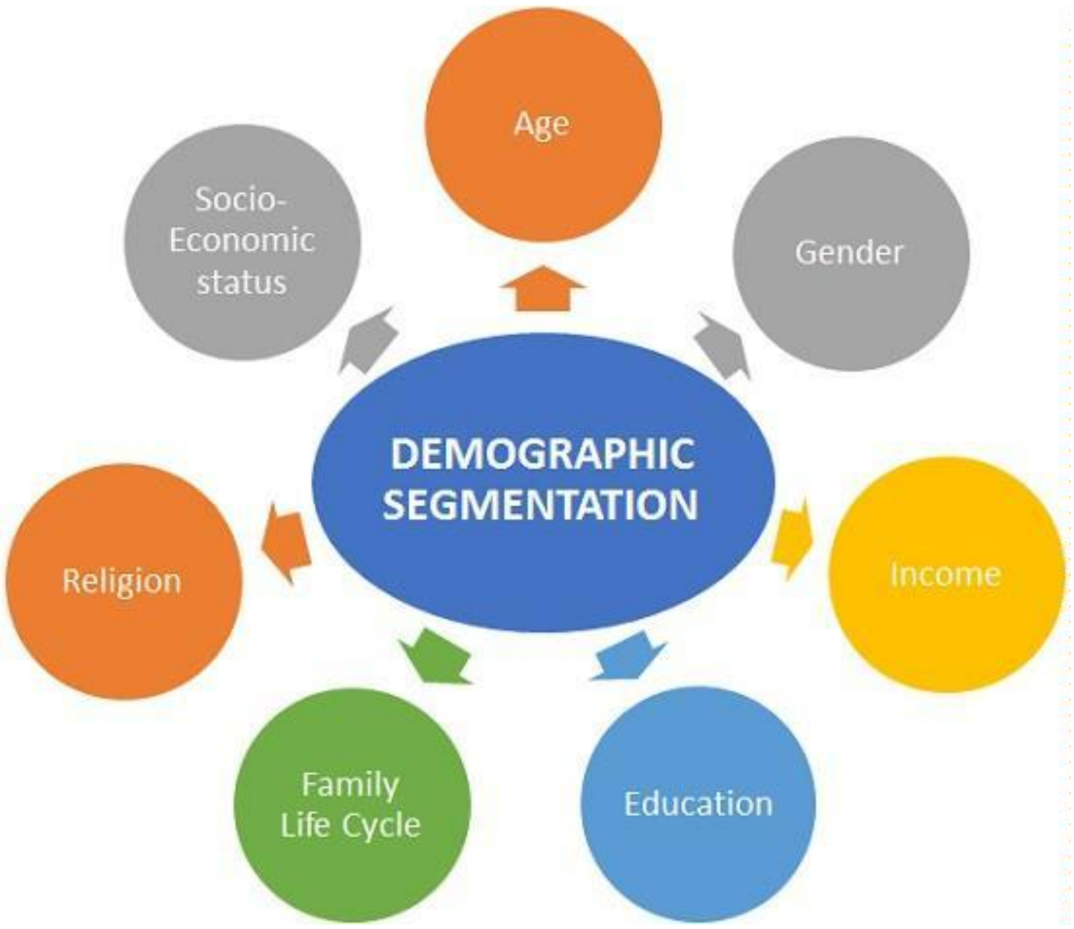
Disadvantage: increased complexity of determining segment memberships for consumers.


Lifestyle
Social Status
Personality Traits
PSYCHOGRAPHIC SEGMENTATION
Activities, Interests and Opinions (AIO)
Values and Attitude

Demographic Segmentation: includes age, gender, income and education. Useful in industries.

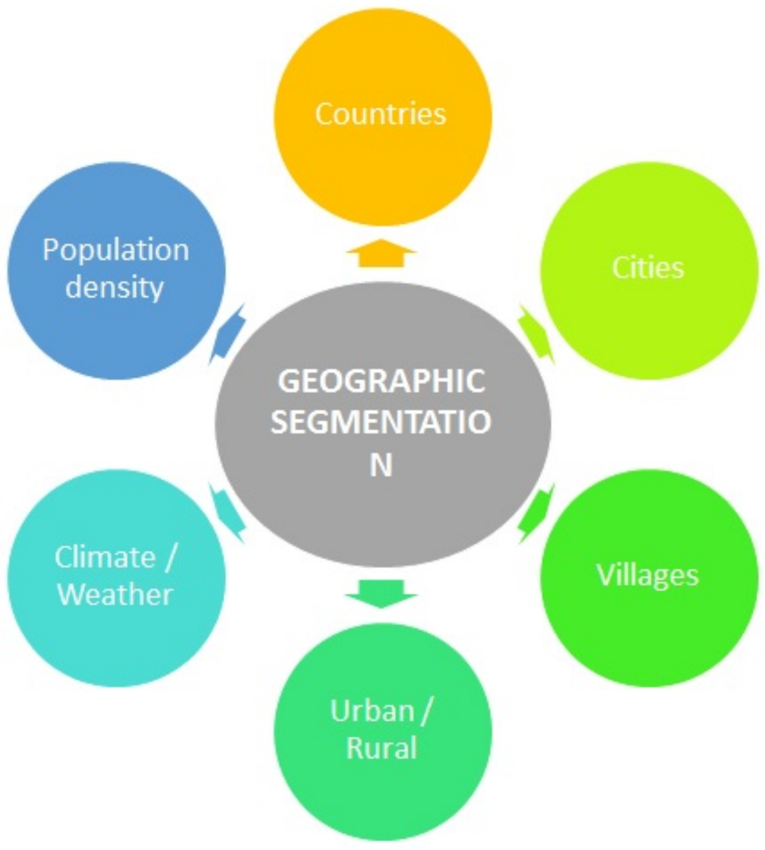
Advantage: segment membership can easily be determined for every customer.

Disadvantage: if this criteria is not the cause for customers product preferences then it does not provide sufficient market insight for optimal segmentation decisions.



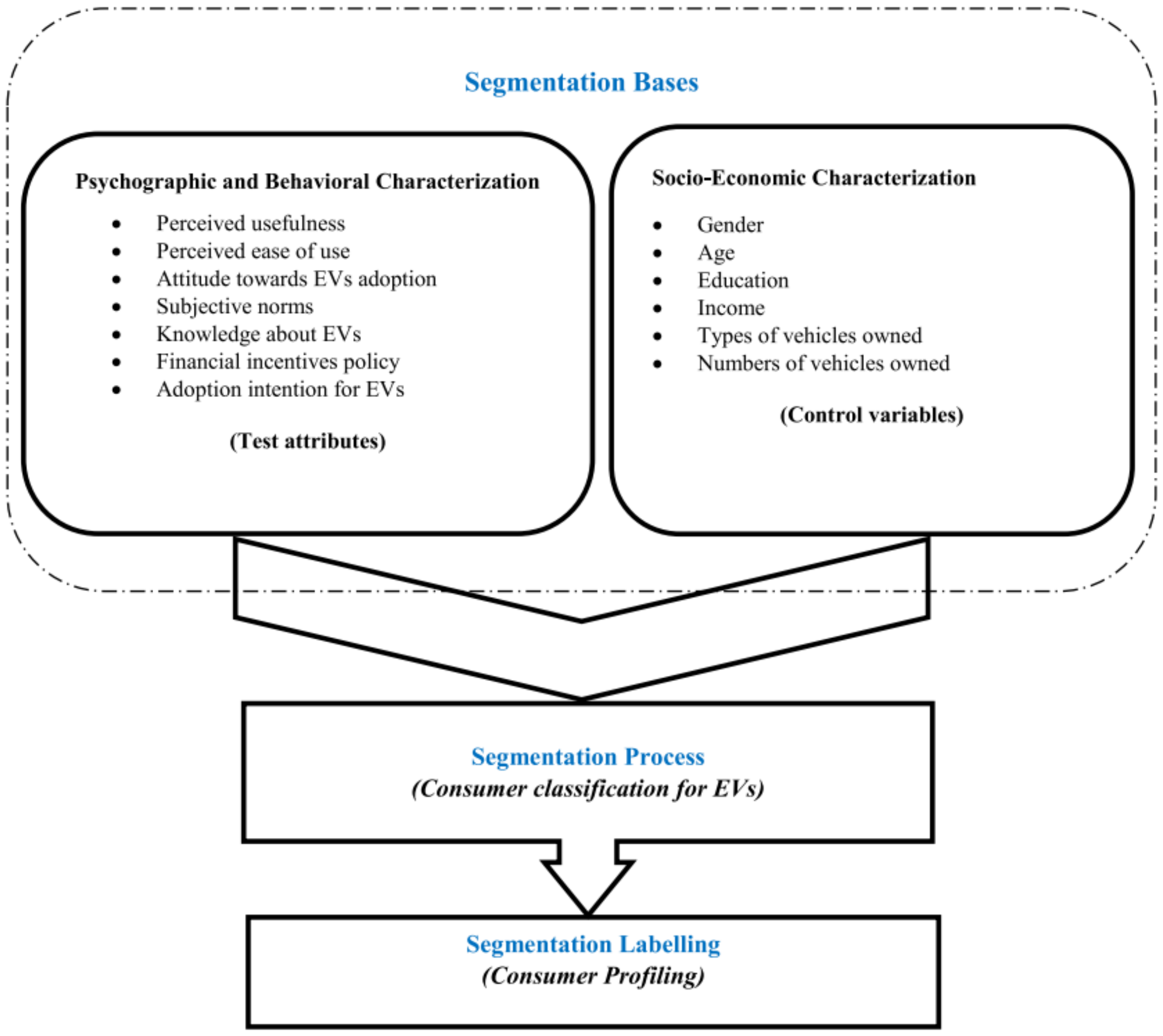
Geographic Segmentation: includes location based on population density, location(far/near, city/village), climate. Useful in industries.
Advantage: Location information can easily be determined and analyse.

Disadvantage: Companies often do not rely solely on geographic segments to determine their target market.



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. The segments are identified based on the relationship among the multiple measured variables. 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. We are going to look into different aspects of segmentation for EVs.



Implementation Packages/Tools used:

1. **Numpy:** To calculate various calculations related to arrays.

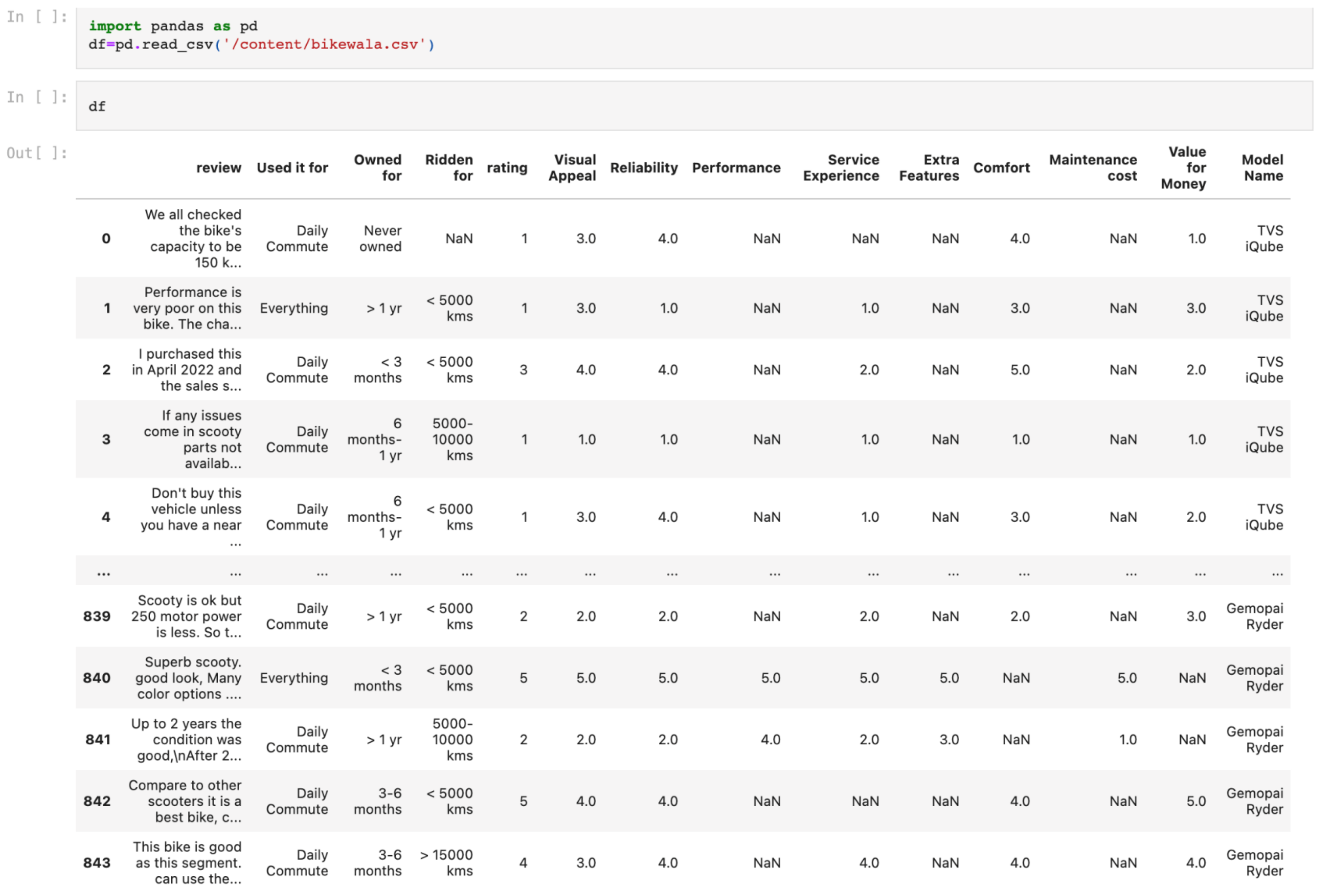
2. **Pandas:** To read or load the datasets.

Data-Preprocessing

Data Cleaning

The data collected is compact and is partly used for visualisation purposes and partly for clustering. Python libraries such as NumPy, Pandas, Scikit-Learn, and SciPy are used for the workflow, and the results obtained are ensured to be reproducible.

Two datasets are used and achieved the results from both the sets.

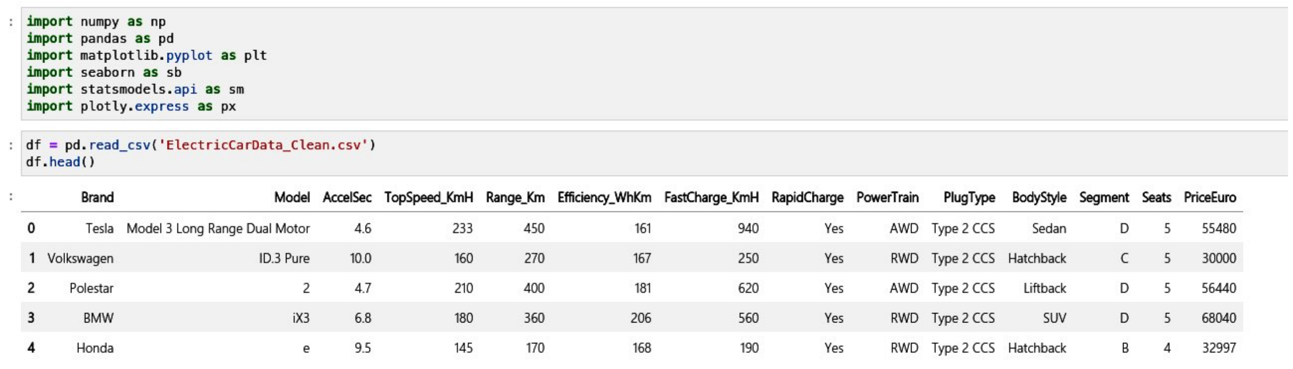


```
In [ ]: import pandas as pd
        df=pd.read_csv('/content/bikewala.csv')
```

```
In [ ]: df
```

Out[ ]:

	review	Used it for	Owned for	Ridden for	rating	Visual Appeal	Reliability	Performance	Service Experience	Extra Features	Comfort	Maintenance cost	Value for Money	Model Name
0	We all checked the bike's capacity to be 150 k...	Daily Commute	Never owned	NaN	1	3.0	4.0	NaN	NaN	NaN	4.0	NaN	1.0	TVS iQube
1	Performance is very poor on this bike. The cha...	Everything	> 1 yr	< 5000 kms	1	3.0	1.0	NaN	1.0	NaN	3.0	NaN	3.0	TVS iQube
2	I purchased this in April 2022 and the sales s...	Daily Commute	< 3 months	< 5000 kms	3	4.0	4.0	NaN	2.0	NaN	5.0	NaN	2.0	TVS iQube
3	If any issues come in scooty parts not availab...	Daily Commute	6 months-1 yr	5000-10000 kms	1	1.0	1.0	NaN	1.0	NaN	1.0	NaN	1.0	TVS iQube
4	Don't buy this vehicle unless you have a near ...	Daily Commute	6 months-1 yr	< 5000 kms	1	3.0	4.0	NaN	1.0	NaN	3.0	NaN	2.0	TVS iQube
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
839	Scooty is ok but 250 motor power is less. So t...	Daily Commute	> 1 yr	< 5000 kms	2	2.0	2.0	NaN	2.0	NaN	2.0	NaN	3.0	Gemopai Ryder
840	Superb scooty. good look, Many color options	Everything	< 3 months	< 5000 kms	5	5.0	5.0	5.0	5.0	5.0	NaN	5.0	NaN	Gemopai Ryder
841	Up to 2 years the condition was good,\nAfter 2...	Daily Commute	> 1 yr	5000-10000 kms	2	2.0	2.0	4.0	2.0	3.0	NaN	1.0	NaN	Gemopai Ryder
842	Compare to other scooters it is a best bike, c...	Daily Commute	3-6 months	< 5000 kms	5	4.0	4.0	NaN	NaN	NaN	4.0	NaN	5.0	Gemopai Ryder
843	This bike is good as this segment. can use the...	Daily Commute	3-6 months	> 15000 kms	4	3.0	4.0	NaN	4.0	NaN	4.0	NaN	4.0	Gemopai Ryder



```
: import numpy as np
  import pandas as pd
  import matplotlib.pyplot as plt
  import seaborn as sb
  import statsmodels.api as sm
  import plotly.express as px
```

```
: df = pd.read_csv('ElectricCarData_Clean.csv')
  df.head()
```

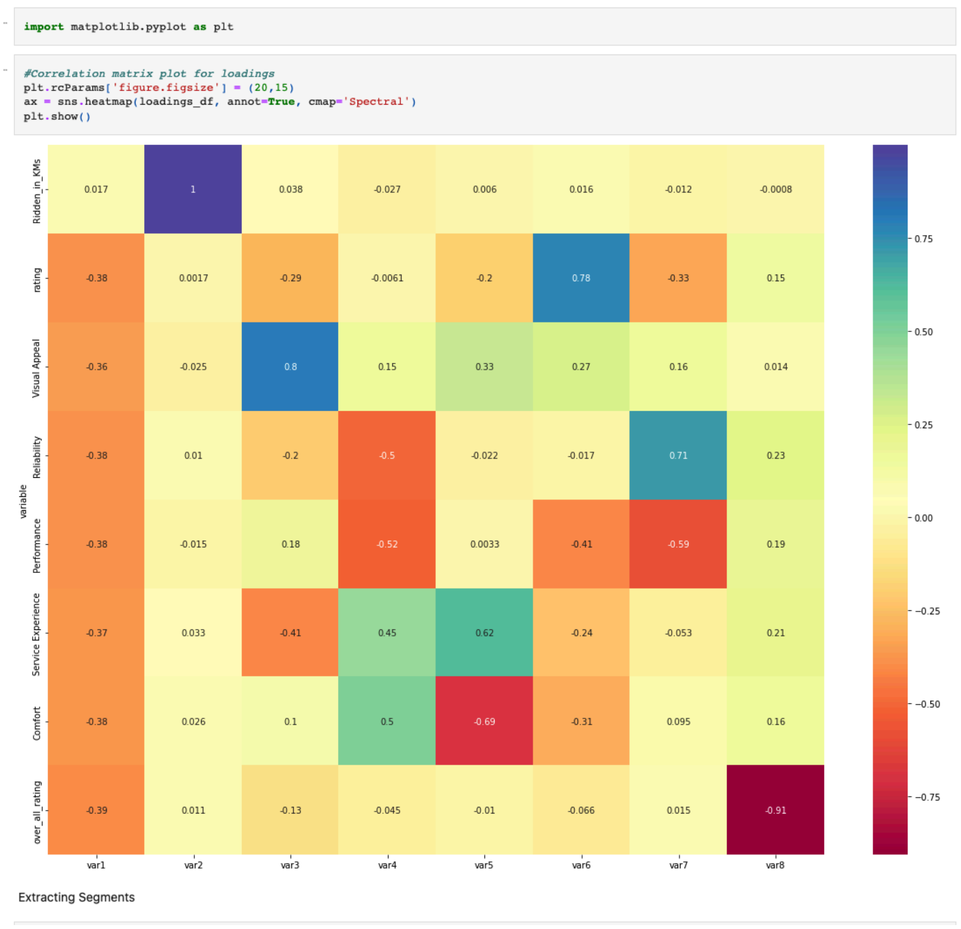
:

	Brand	Model	AccelSec	TopSpeed_KmH	Range_Km	Efficiency_WhKm	FastCharge_KmH	RapidCharge	PowerTrain	PlugType	BodyStyle	Segment	Seats	PriceEuro
0	Tesla	Model 3 Long Range Dual Motor	4.6	233	450	161	940	Yes	AWD	Type 2 CCS	Sedan	D	5	55480
1	Volkswagen	ID.3 Pure	10.0	160	270	167	250	Yes	RWD	Type 2 CCS	Hatchback	C	5	30000
2	Polestar	2	4.7	210	400	181	620	Yes	AWD	Type 2 CCS	Liftback	D	5	56440
3	BMW	iX3	6.8	180	360	206	560	Yes	RWD	Type 2 CCS	SUV	D	5	68040
4	Honda	e	9.5	145	170	168	190	Yes	RWD	Type 2 CCS	Hatchback	B	4	32997

EDA

We start the Exploratory Data Analysis with some data Analysis drawn from the data without Principal Component Analysis and with some Principal Component Analysis in the dataset obtained from the combination of all the data we have. PCA is a statistical process that converts the observations of correlated features into a set of linearly uncorrelated features with the help of orthogonal transformation. These new transformed features are called the Principal Components. The process helps in reducing dimensions of the data to make the process of classification/regression or any form of machine learning, cost-effective.

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 between all the possible pairs of values through the heat-map in the below figure. The relationship between two variables is usually considered strong when their correlation coefficient value is larger than 0.7.

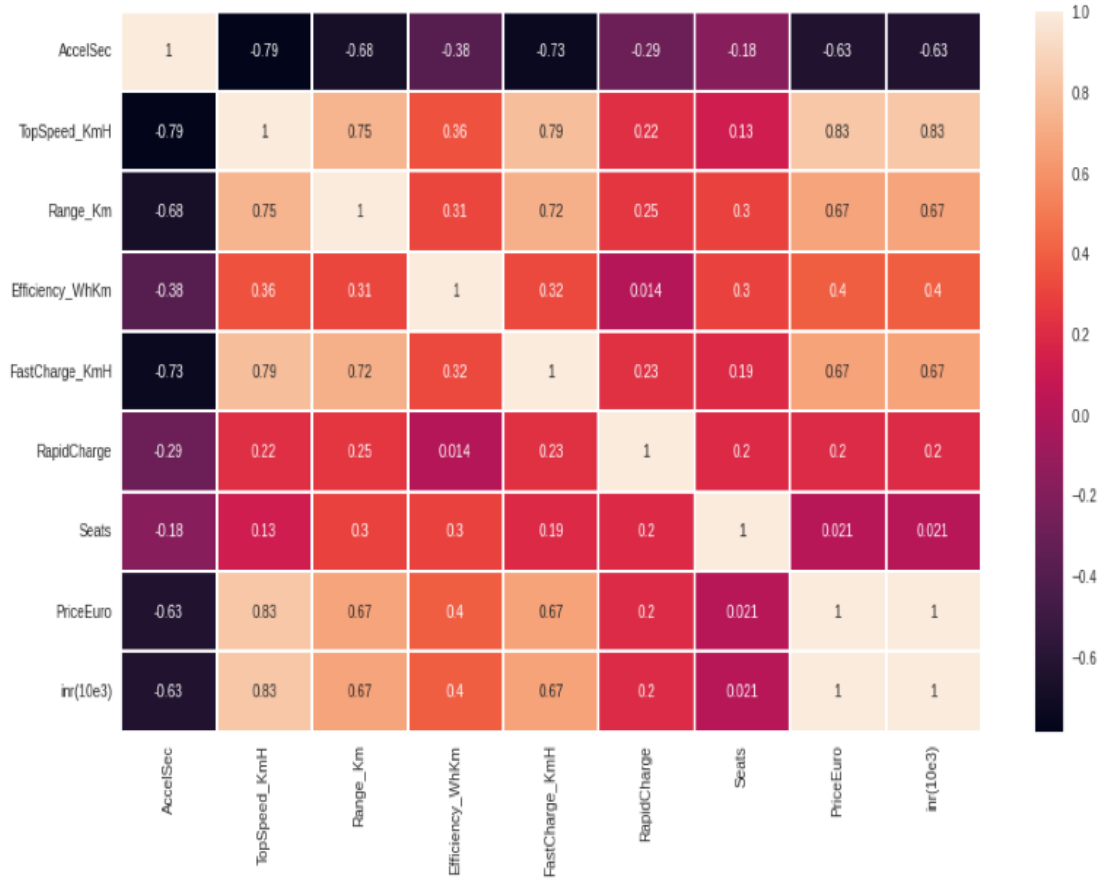


```
import matplotlib.pyplot as plt
```

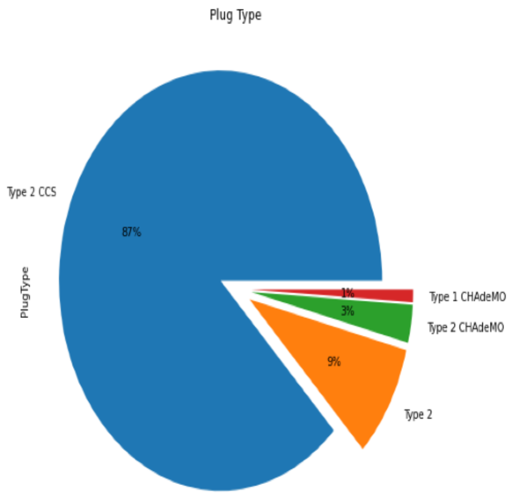
```
#Correlation matrix plot for loadings
plt.rcParams['figure.figsize'] = (20,15)
ax = sns.heatmap(loadings_df, annot=True, cmap='Spectral')
plt.show()
```

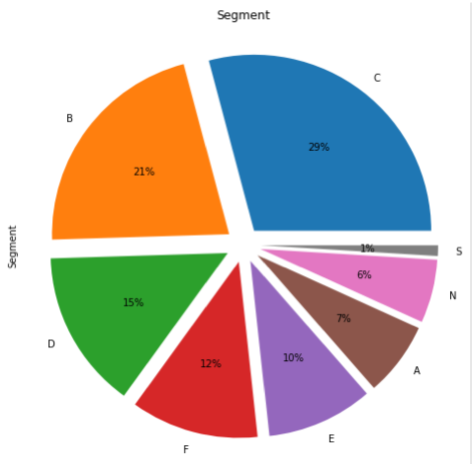
variable	var1	var2	var3	var4	var5	var6	var7	var8
Ridden_in_KMs	0.017	1	0.038	-0.027	0.006	0.016	-0.012	-0.0008
rating	-0.38	0.0017	-0.29	-0.0061	-0.2	0.78	-0.33	0.15
Visual Appeal	-0.36	-0.025	0.8	0.15	0.33	0.27	0.16	0.014
Reliability	-0.38	0.01	-0.2	-0.5	-0.022	-0.017	0.71	0.23
Performance	-0.38	-0.015	0.18	-0.52	0.0033	-0.41	-0.59	0.19
Service Experience	-0.37	0.033	-0.41	0.45	0.62	-0.24	-0.053	0.21
Comfort	-0.38	0.026	0.1	0.5	-0.69	-0.31	0.095	0.16
over_all_rating	-0.39	0.011	-0.13	-0.045	-0.01	-0.066	0.015	-0.91

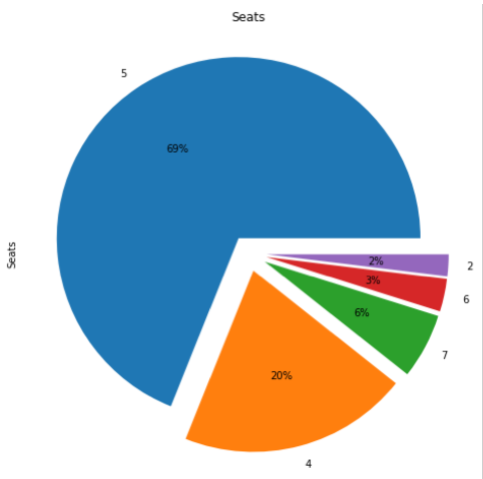
Extracting Segments

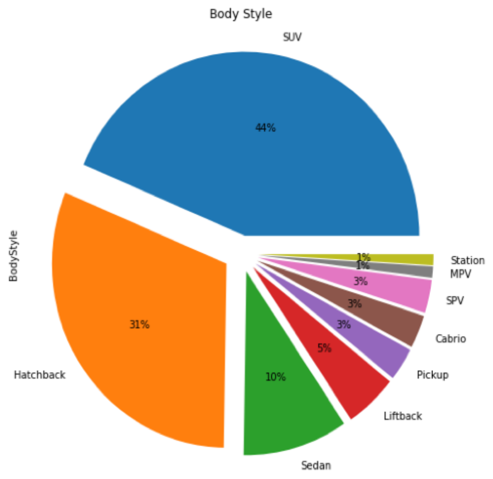


Comparison of cars:



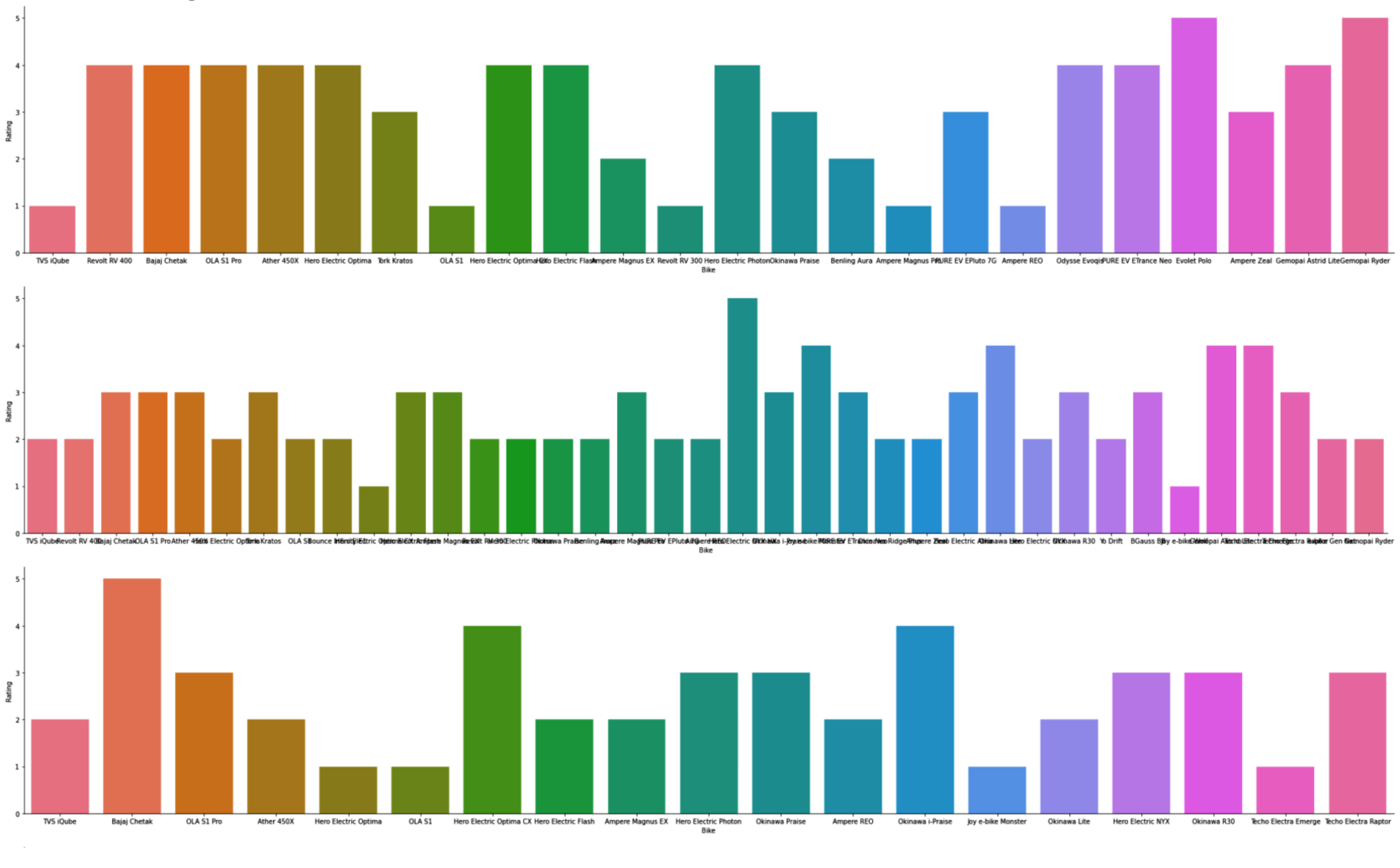


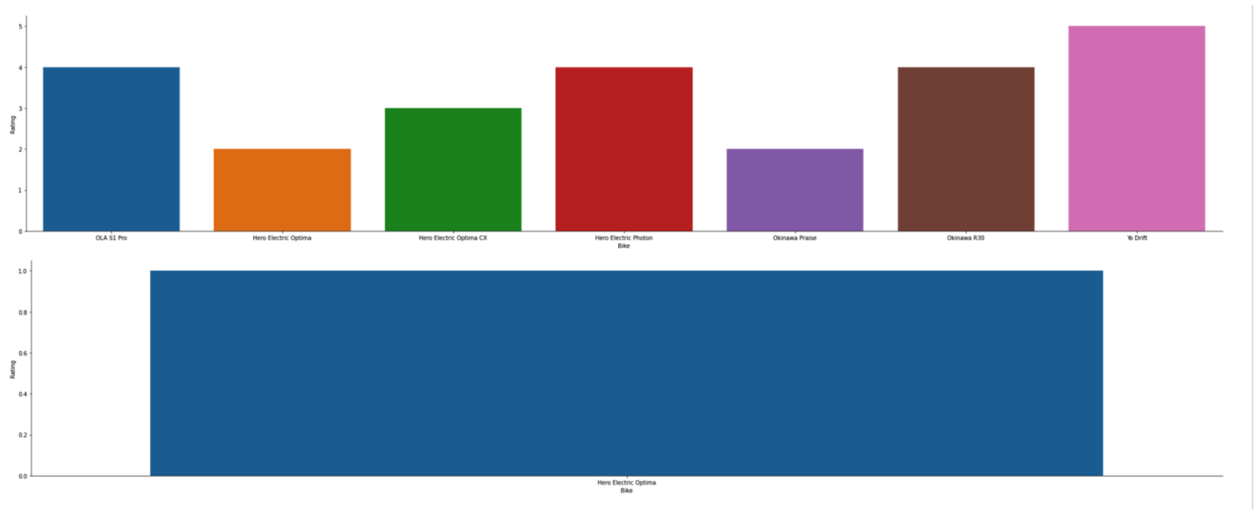




Comparison of Bikes:

```
0  Hero Electric Optima     1.0
```

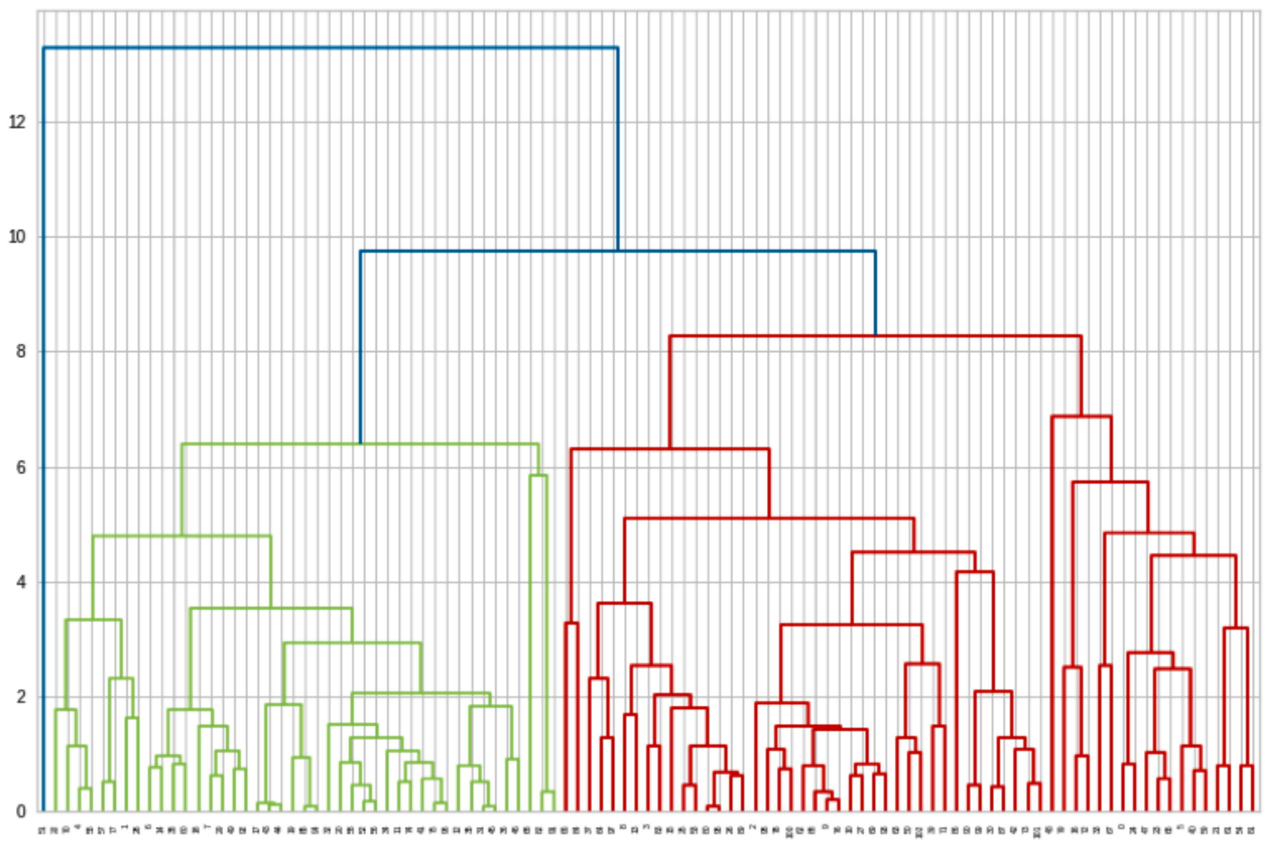




Extracting Segments

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 chose 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.

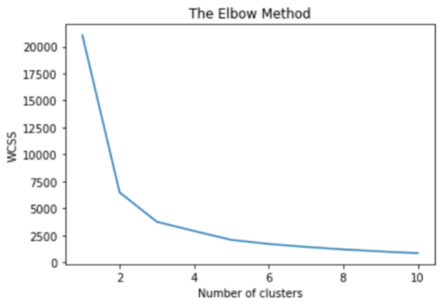


Elbow Method

The Elbow method is a popular method for determining the optimal number of clusters. The method is based on calculating the Within-Cluster-Sum of Squared Errors (WSS) for a different number of clusters (k) and selecting the k for which change in WSS first starts to diminish. The idea behind the elbow method is that the explained variation changes rapidly for a small number of clusters and then it slows down leading to an elbow formation in the curve. The elbow point is the number of clusters we can use for our clustering algorithm.

The KElbowVisualizer function fits the KMeans model for a range of clusters values between 2 to 8. As shown in Figure, the elbow point is achieved which is highlighted by the function itself. The function also informs us about how much time was needed to plot models for various numbers of clusters through the green line.

In [68]:
```
# Using the elbow method to find the optimal number of clusters
from sklearn.cluster import KMeans
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



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 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.

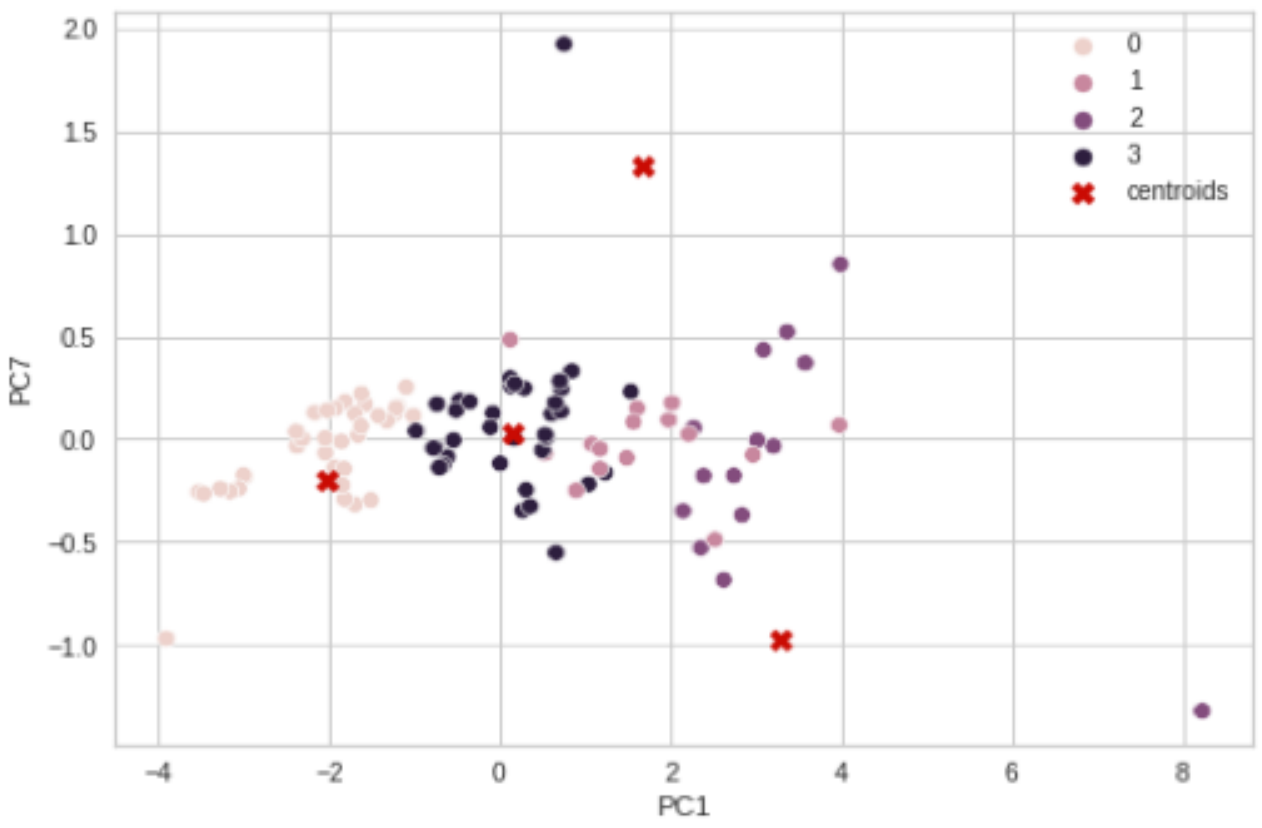
K-Means Algorithm

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 **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.
  According to the Elbow method, here we take K=4 clusters to train KMeans model. The derived clusters are shown in the following figure

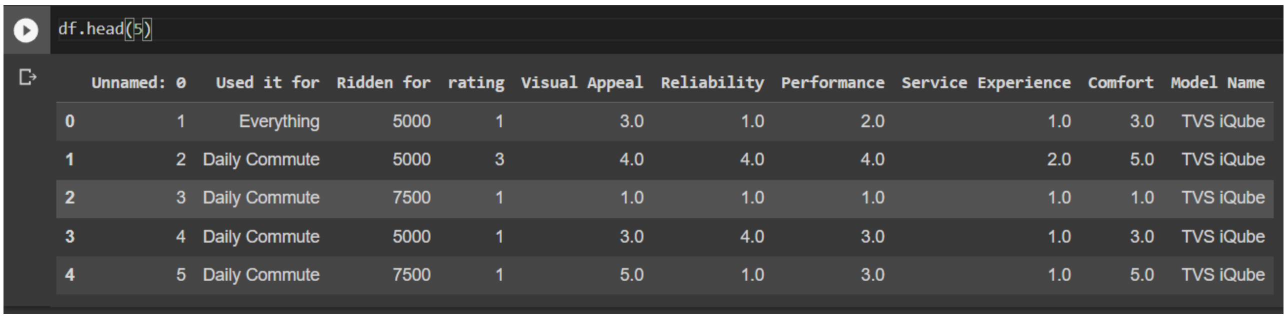
```python
#K-means clustering

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.
```



The new data set taken to understand the behavior of different customers based on their usage

The data contain all the ratings for the bikes

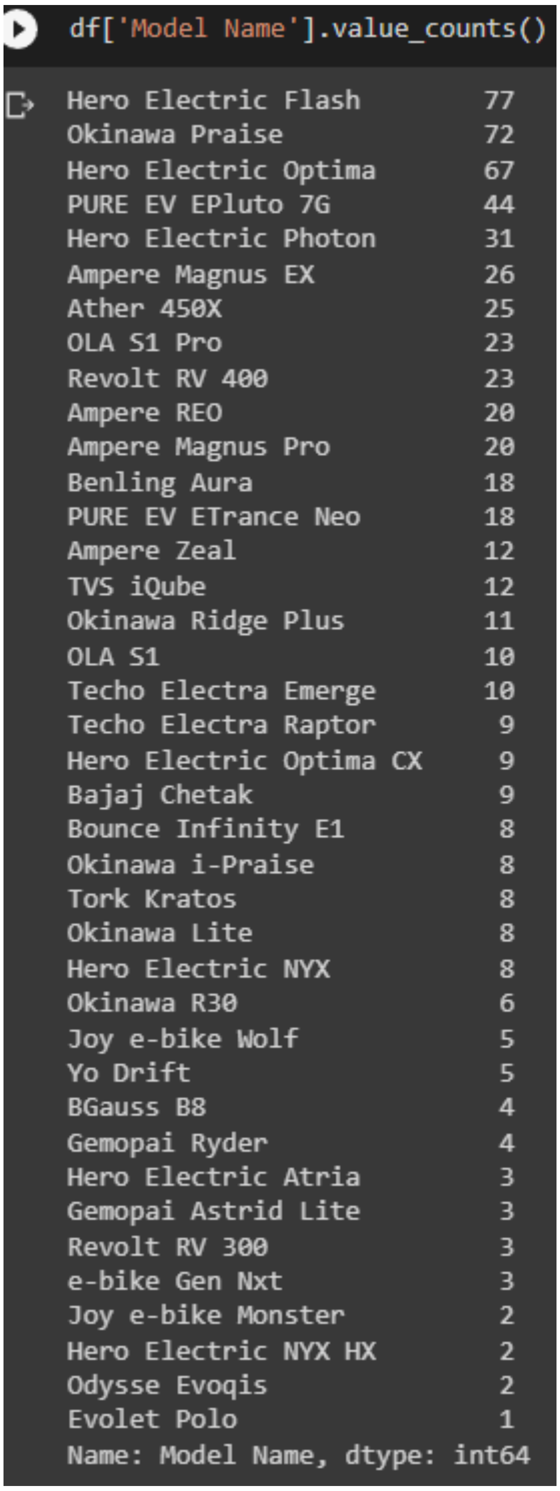
Figure shows the data taken for extracting segments



```
df.head(5)
```

	Unnamed: 0	Used it for	Ridden for	rating	Visual Appeal	Reliability	Performance	Service Experience	Comfort	Model Name
0	1	Everything	5000	1	3.0	1.0	2.0	1.0	3.0	TVS iQube
1	2	Daily Commute	5000	3	4.0	4.0	4.0	2.0	5.0	TVS iQube
2	3	Daily Commute	7500	1	1.0	1.0	1.0	1.0	1.0	TVS iQube
3	4	Daily Commute	5000	1	3.0	4.0	3.0	1.0	3.0	TVS iQube
4	5	Daily Commute	7500	1	5.0	1.0	3.0	1.0	5.0	TVS iQube

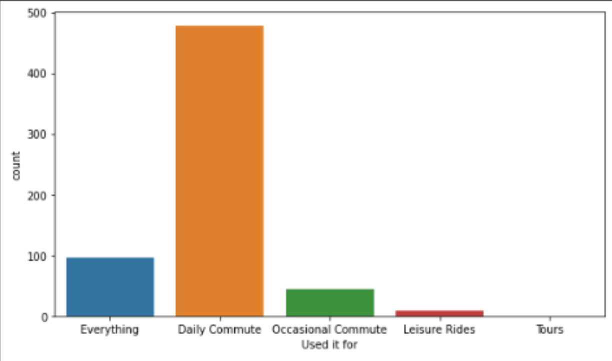
Different models of Electric Bikes in our data set

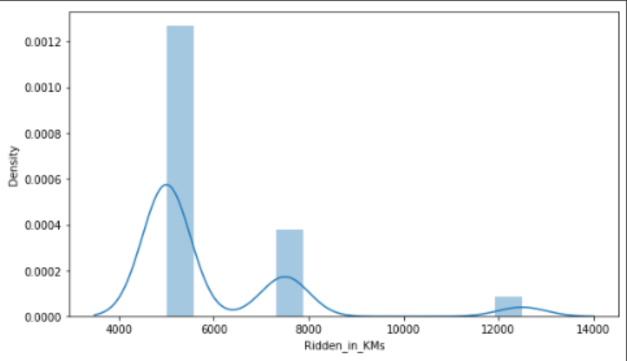


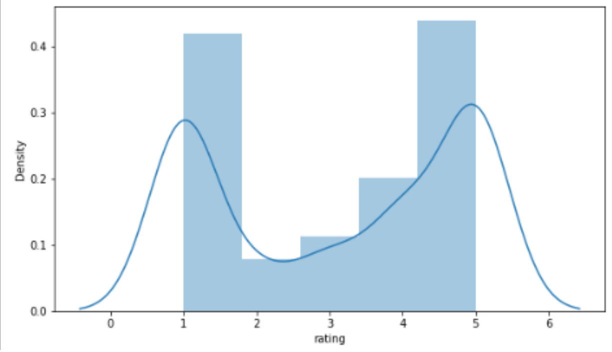
```
df['Model Name'].value_counts()

Hero Electric Flash         77
Okinawa Praise              72
Hero Electric Optima        67
PURE EV EPluto 7G           44
Hero Electric Photon        31
Ampere Magnus EX            26
Ather 450X                  25
OLA S1 Pro                  23
Revolt RV 400               23
Ampere REO                  20
Ampere Magnus Pro           20
Benling Aura                18
PURE EV ETrance Neo         18
Ampere Zeal                 12
TVS iQube                   12
Okinawa Ridge Plus          11
OLA S1                      10
Techo Electra Emerge        10
Techo Electra Raptor         9
Hero Electric Optima CX      9
Bajaj Chetak                 9
Bounce Infinity E1           8
Okinawa i-Praise             8
Tork Kratos                  8
Okinawa Lite                 8
Hero Electric NYX            8
Okinawa R30                  6
Joy e-bike Wolf              5
Yo Drift                     5
BGauss B8                    4
Gemopai Ryder                4
Hero Electric Atria          3
Gemopai Astrid Lite          3
Revolt RV 300                3
e-bike Gen Nxt               3
Joy e-bike Monster           2
Hero Electric NYX HX         2
Odysse Evoqis                2
Evolet Polo                  1
Name: Model Name, dtype: int64
```

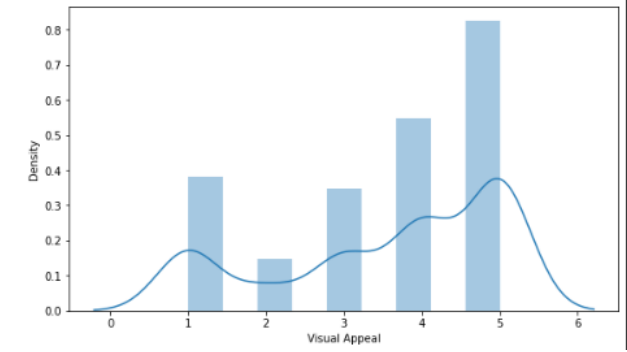
EDA of Data

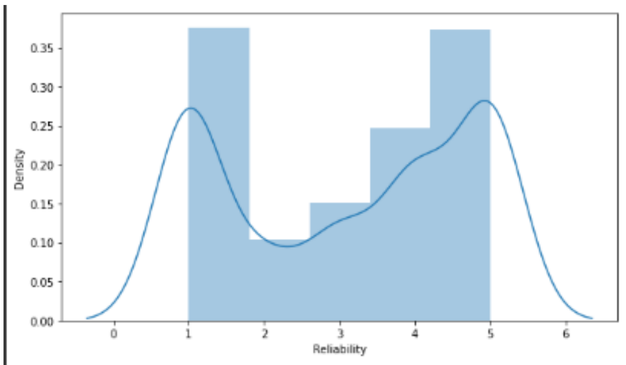
Analyzing each variable in the data set i.e. the different variables taken into consideration. Where users are given their individual ratings for each variables

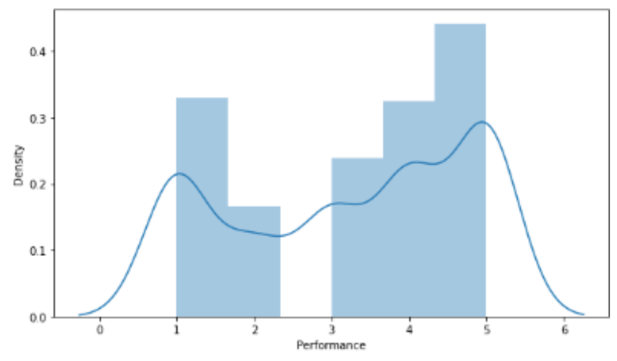

count
Everything
Daily Commute
Occasional Commute
Leisure Rides
Tours
Used it for

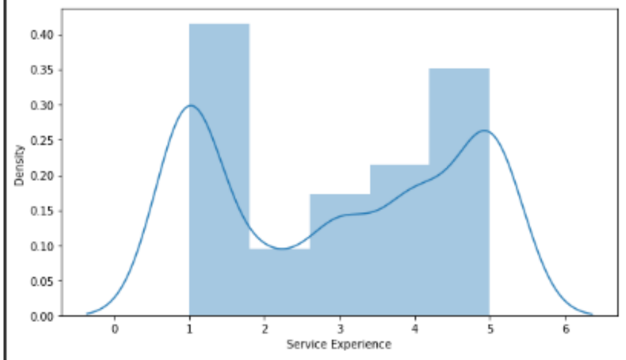

Density
Ridden_in_KMs

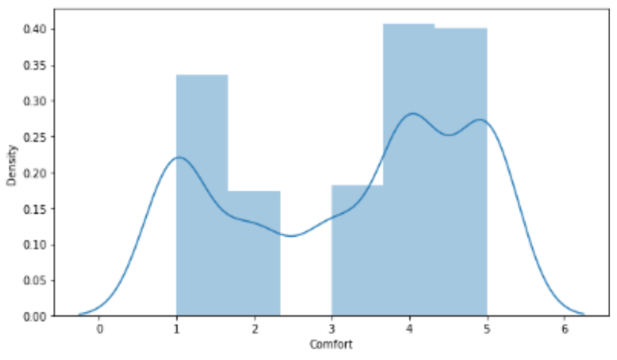

Density
rating

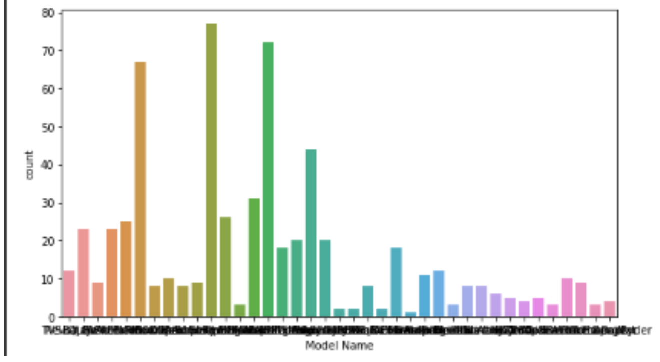

Density
Visual Appeal

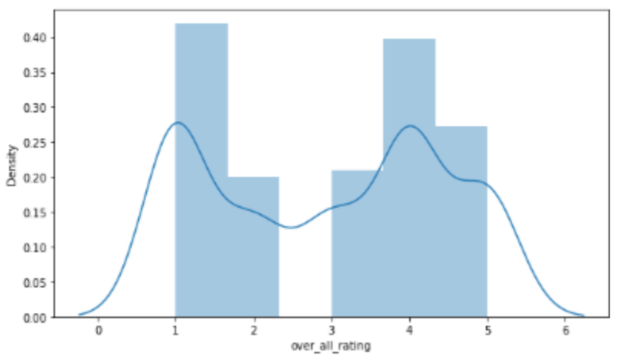

Density
Reliability


Density
Performance


Density
Service Experience

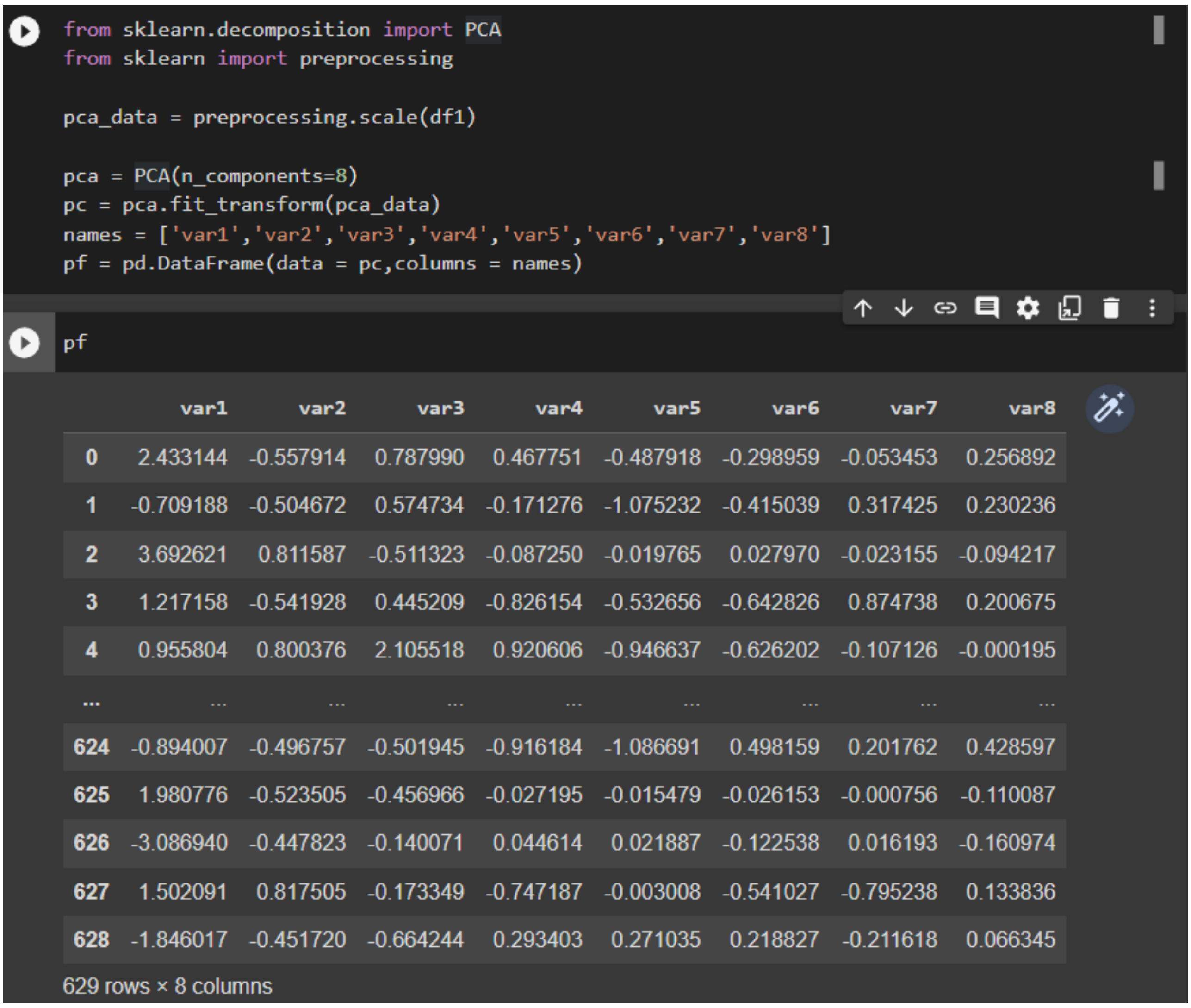

Density
Comfort


count
Model Name


Density
over_all_rating

Principal component analysis (PCA):

Principal component analysis (PCA) is a popular technique for analyzing large datasets containing a high number of dimensions/features per observation, increasing the interpretability of data while preserving the maximum amount of information, and enabling the visualization of multidimensional data. Formally, PCA is a statistical technique for reducing the dimensionality of a dataset. This is accomplished by linearly transforming the data into a new coordinate system where (most of) the variation in the data can be described with fewer dimensions than the initial data. Many studies use the first two principal components in order to plot the data in two dimensions and to visually identify clusters of closely related data points. Principal component analysis has applications in many fields such as population genetics, micro biome studies, and atmospheric science.



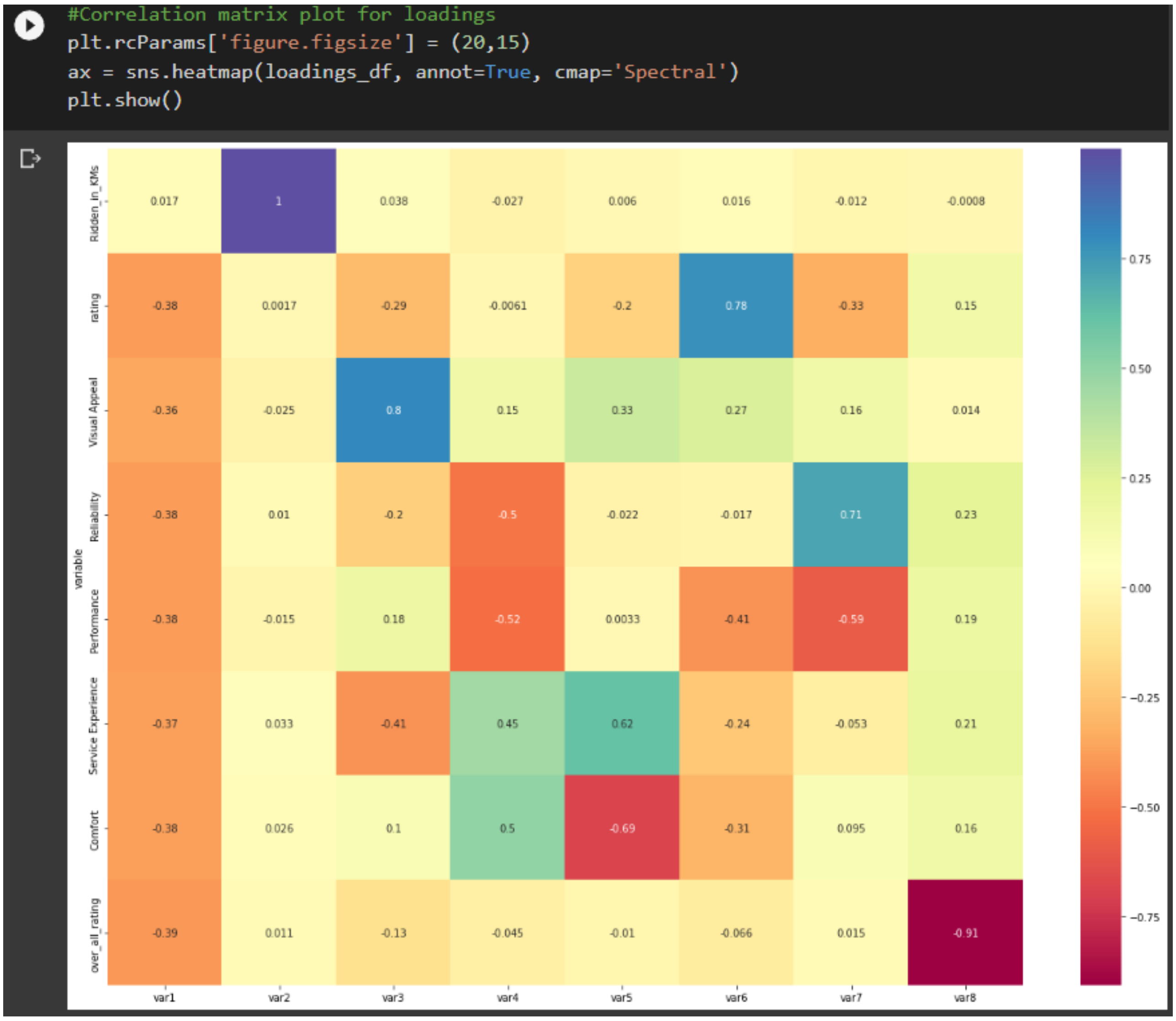
```
from sklearn.decomposition import PCA
from sklearn import preprocessing

pca_data = preprocessing.scale(df1)

pca = PCA(n_components=8)
pc = pca.fit_transform(pca_data)
names = ['var1','var2','var3','var4','var5','var6','var7','var8']
pf = pd.DataFrame(data = pc,columns = names)
```

```
pf
```

	var1	var2	var3	var4	var5	var6	var7	var8
0	2.433144	-0.557914	0.787990	0.467751	-0.487918	-0.298959	-0.053453	0.256892
1	-0.709188	-0.504672	0.574734	-0.171276	-1.075232	-0.415039	0.317425	0.230236
2	3.692621	0.811587	-0.511323	-0.087250	-0.019765	0.027970	-0.023155	-0.094217
3	1.217158	-0.541928	0.445209	-0.826154	-0.532656	-0.642826	0.874738	0.200675
4	0.955804	0.800376	2.105518	0.920606	-0.946637	-0.626202	-0.107126	-0.000195
...	...	...	...	...	...	...	...	...
624	-0.894007	-0.496757	-0.501945	-0.916184	-1.086691	0.498159	0.201762	0.428597
625	1.980776	-0.523505	-0.456966	-0.027195	-0.015479	-0.026153	-0.000756	-0.110087
626	-3.086940	-0.447823	-0.140071	0.044614	0.021887	-0.122538	0.016193	-0.160974
627	1.502091	0.817505	-0.173349	-0.747187	-0.003008	-0.541027	-0.795238	0.133836
628	-1.846017	-0.451720	-0.664244	0.293403	0.271035	0.218827	-0.211618	0.066345

629 rows × 8 columns

CORRELATION MATRIX

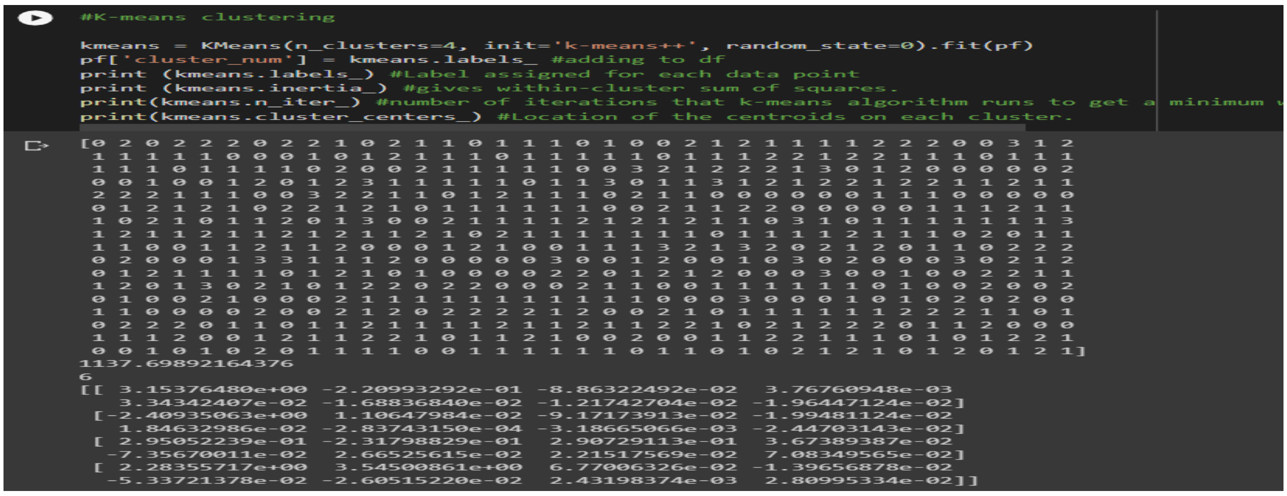


```
#Correlation matrix plot for loadings
plt.rcParams['figure.figsize'] = (20,15)
ax = sns.heatmap(loadings_df, annot=True, cmap='Spectral')
plt.show()
```

variable	var1	var2	var3	var4	var5	var6	var7	var8
Ridden_in_KMs	0.017	1	0.038	-0.027	0.006	0.016	-0.012	-0.0008
rating	-0.38	0.0017	-0.29	-0.0061	-0.2	0.78	-0.33	0.15
Visual Appeal	-0.36	-0.025	0.8	0.15	0.33	0.27	0.16	0.014
Reliability	-0.38	0.01	-0.2	-0.5	-0.022	-0.017	0.71	0.23
Performance	-0.38	-0.015	0.18	-0.52	0.0033	-0.41	-0.59	0.19
Service Experience	-0.37	0.033	-0.41	0.45	0.62	-0.24	-0.053	0.21
Comfort	-0.38	0.026	0.1	0.5	-0.69	-0.31	0.095	0.16
over_all_rating	-0.39	0.011	-0.13	-0.045	-0.01	-0.066	0.015	-0.91

USING K-MEAN CLUSTERING TO FROM CLUSTERS

We have chosen the no.of segments to “4”



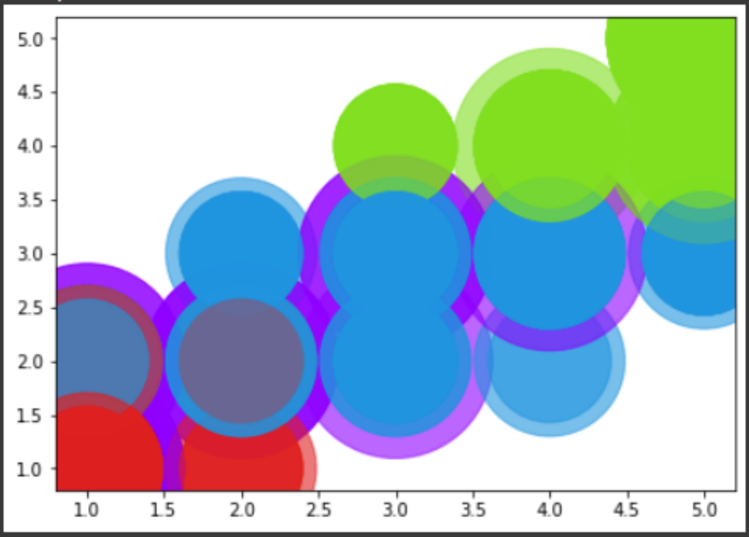
```
#K-means clustering

kmeans = KMeans(n_clusters=4, init='k-means++', random_state=0).fit(pf)
pf['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 w
print(kmeans.cluster_centers_) #Location of the centroids on each cluster.
```

```
[0 2 0 2 2 2 0 2 2 1 0 2 1 1 0 1 1 1 0 1 0 0 2 1 2 1 1 1 1 2 2 2 0 0 3 1 2
 1 1 1 1 1 0 0 0 1 0 1 2 1 1 1 0 1 1 1 1 1 1 0 1 1 1 2 2 1 2 2 1 1 1 0 1 1 1
 1 1 1 0 1 1 1 1 0 2 0 0 2 1 1 1 1 1 0 0 3 2 1 2 2 2 1 3 0 1 2 0 0 0 0 0 2
 0 0 1 0 0 1 2 0 1 2 3 1 1 1 1 1 0 1 1 3 0 1 1 3 1 2 1 2 2 1 2 2 1 1 2 1 1
 2 2 2 1 1 1 0 0 3 2 2 1 1 0 1 2 1 1 1 0 2 1 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0
 0 1 2 1 2 1 0 2 2 1 2 1 0 1 1 1 1 1 1 1 0 0 2 1 1 2 2 0 0 0 0 0 1 1 1 2 1 1
 1 0 2 1 0 1 1 2 0 1 3 0 0 2 1 1 1 1 1 2 1 2 1 2 1 1 0 3 1 0 1 1 1 1 1 1 1 3
 1 2 1 1 2 1 1 2 1 2 1 1 2 1 0 2 1 1 1 1 1 1 1 1 0 1 1 1 1 1 2 1 1 1 0 2 0 1 1
 1 1 0 0 1 1 2 1 1 2 0 0 0 1 2 1 0 0 1 1 1 3 2 1 3 2 0 2 1 2 0 1 1 0 2 2 2
 0 2 0 0 0 1 3 3 1 1 1 2 0 0 0 0 0 3 0 0 1 2 0 0 1 0 3 0 2 0 0 0 3 0 2 1 2
 0 1 2 1 1 1 1 0 1 2 1 0 1 0 0 0 0 2 2 0 1 2 1 2 0 0 0 3 0 0 1 0 0 2 2 1 1
 1 2 0 1 3 0 2 1 0 1 2 2 0 2 2 0 0 0 2 1 1 0 0 1 1 1 1 1 1 0 1 0 0 2 0 0 2
 0 1 0 0 2 1 0 0 0 2 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 3 0 0 0 1 0 1 0 2 0 2 0 0
 1 1 0 0 0 0 2 0 0 2 1 2 0 2 2 2 2 1 2 0 0 2 1 0 1 1 1 1 1 1 2 2 2 1 1 0 1
 0 2 2 2 0 1 1 0 1 1 2 1 1 1 1 1 2 1 1 2 1 1 2 2 1 0 2 1 2 2 2 0 1 1 2 0 0 0
 1 1 1 2 0 0 1 2 1 1 2 2 1 0 1 1 2 1 0 0 2 0 0 1 1 2 2 1 1 1 0 1 0 1 2 2 1
 0 0 1 0 1 0 2 0 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 0 1 0 2 1 2 1 0 1 2 0 1 2 1]
1137.69892164376
6
[[ 3.15376480e+00 -2.20993292e-01 -8.86322492e-02  3.76760948e-03
   3.34342407e-02 -1.68836840e-02 -1.21742704e-02 -1.96447124e-02]
 [-2.40935063e+00  1.10647984e-02 -9.17173913e-02 -1.99481124e-02
   1.84632986e-02 -2.83743150e-04 -3.18665066e-03 -2.44703143e-02]
 [ 2.95052239e-01 -2.31798829e-01  2.90729113e-01  3.67389387e-02
  -7.35670011e-02  2.66525615e-02  2.21517569e-02  7.08349565e-02]
 [ 2.28355717e+00  3.54500861e+00  6.77006326e-02 -1.39656878e-02
  -5.33721378e-02 -2.60515220e-02  2.43198374e-03  2.80995334e-02]]
```

Analyzing the points under each cluster

```
[109] import matplotlib.pyplot as plt
      plt.scatter(df.rating, df.over_all_rating,c=df['c'],s=df['Ridden_in_KMs'], alpha = 0.6)
```

```
<matplotlib.collections.PathCollection at 0x7f35fa004a50>
```



DESCRIBING THE SEGMENTS

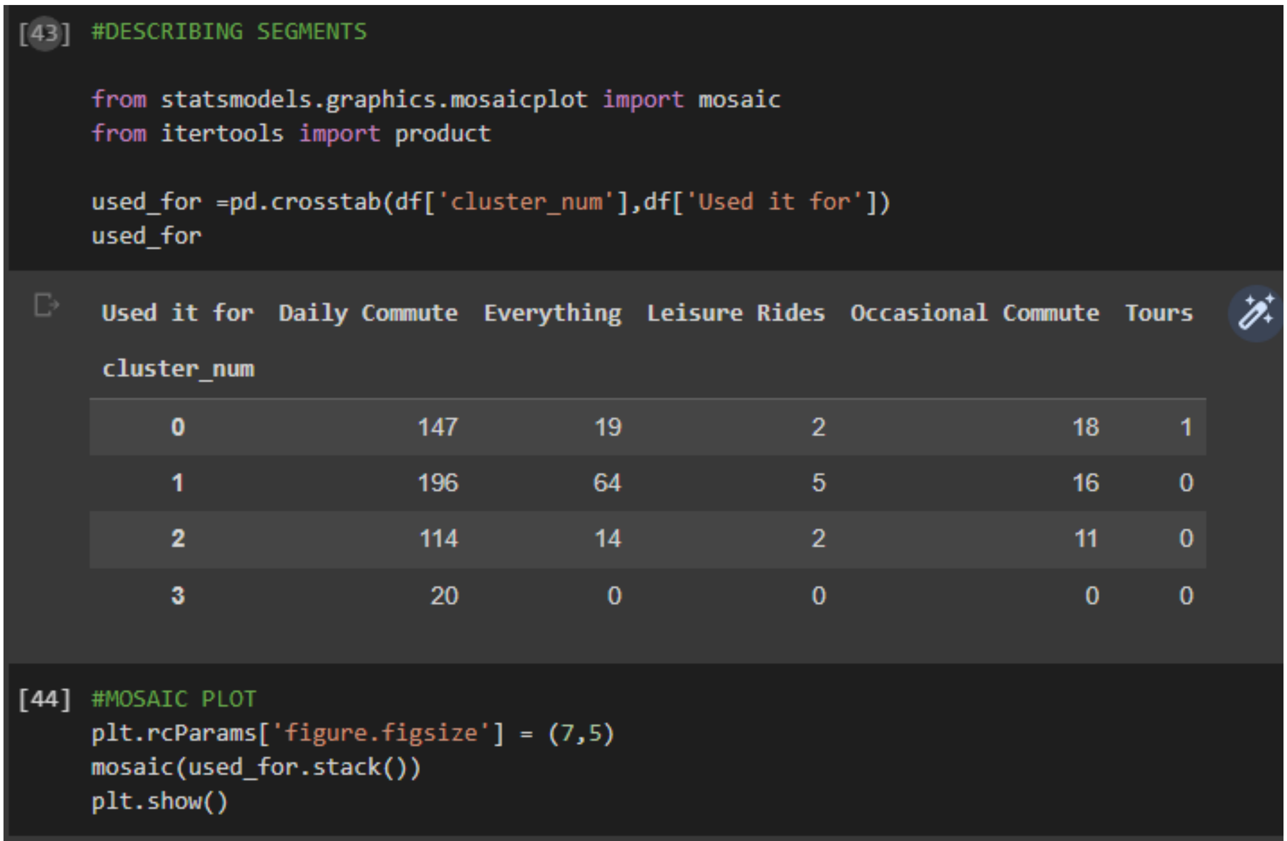
Here we describe our segments with respect to two key variables in our data set i.e.

1) Used For It

2) Model Name

In the data we have different usages like {Daily Commute, Everything, occasional Commute, Leisure Rides, Tours} under the “Used For It” column.

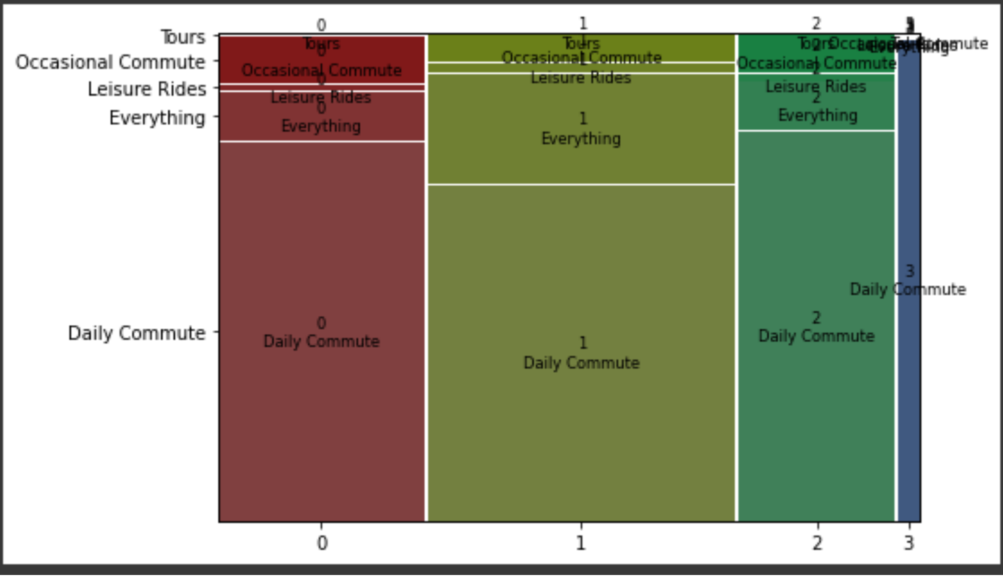
The below figure shows how the segments are formed for each usage



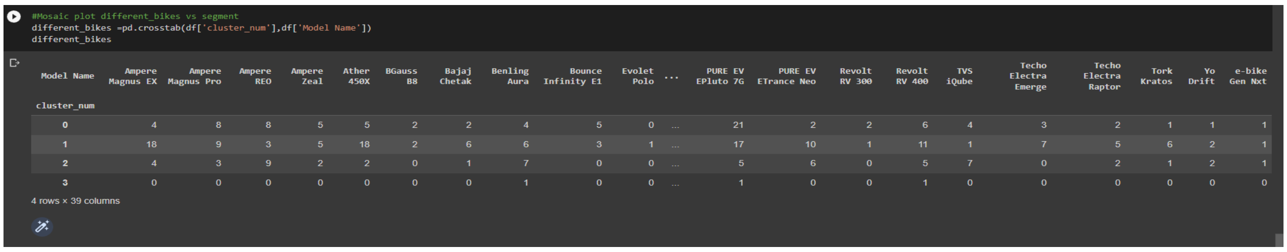
```
[43] #DESCRIBING SEGMENTS

     from statsmodels.graphics.mosaicplot import mosaic
     from itertools import product

     used_for =pd.crosstab(df['cluster_num'],df['Used it for'])
     used_for
```

Used it for	Daily Commute	Everything	Leisure Rides	Occasional Commute	Tours
cluster_num					
0	147	19	2	18	1
1	196	64	5	16	0
2	114	14	2	11	0
3	20	0	0	0	0

```
[44] #MOSAIC PLOT
     plt.rcParams['figure.figsize'] = (7,5)
     mosaic(used_for.stack())
     plt.show()
```



These are the different types of electric bikes as mentioned in the earlier. The below figure depicts the segments based on the different models of electric bikes

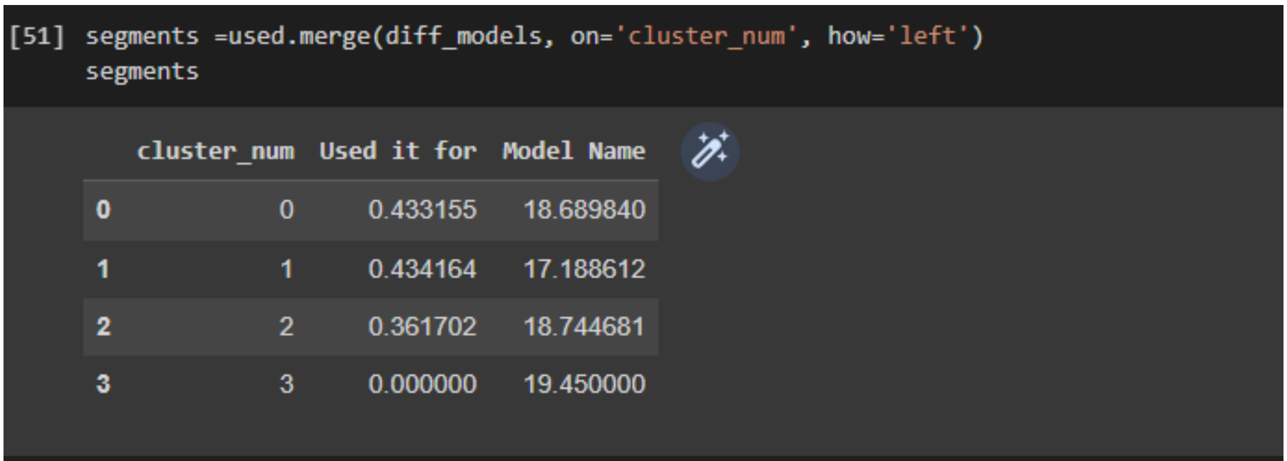


```
#Mosaic plot different_bikes vs segment
different_bikes =pd.crosstab(df['cluster_num'],df['Model Name'])
different_bikes
```

Model Name	Ampere Magnus EX	Ampere Magnus Pro	Ampere REO	Ampere Zeal	Ather 450X	BGauss B8	Bajaj Chetak	Benling Aura	Bounce Infinity E1	Evolet Polo	...	PURE EV EPluto 7G	PURE EV ETrance Neo	Revolt RV 300	Revolt RV 400	TVS iQube	Techo Electra Emerge	Techo Electra Raptor	Tork Kratos	Yo Drift	e-bike Gen Nxt
cluster_num																					
0	4	8	8	5	5	2	2	4	5	0	...	21	2	2	6	4	3	2	1	1	1
1	18	9	3	5	18	2	6	6	3	1	...	17	10	1	11	1	7	5	6	2	1
2	4	3	9	2	2	0	1	7	0	0	...	5	6	0	5	7	0	2	1	2	1
3	0	0	0	0	0	0	0	1	0	0	...	1	0	0	1	0	0	0	0	0	0

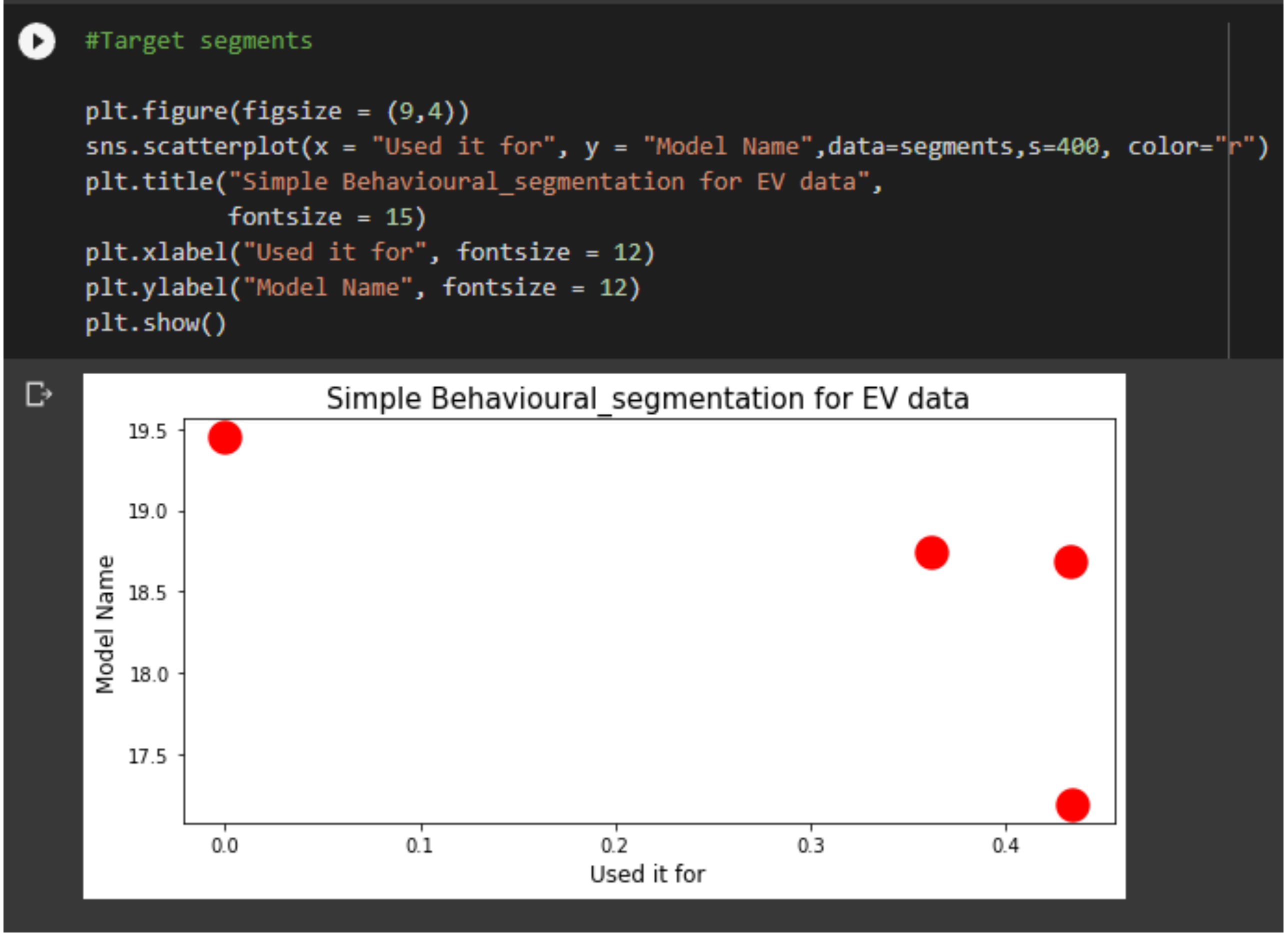
4 rows × 39 columns

Overs all Segment points



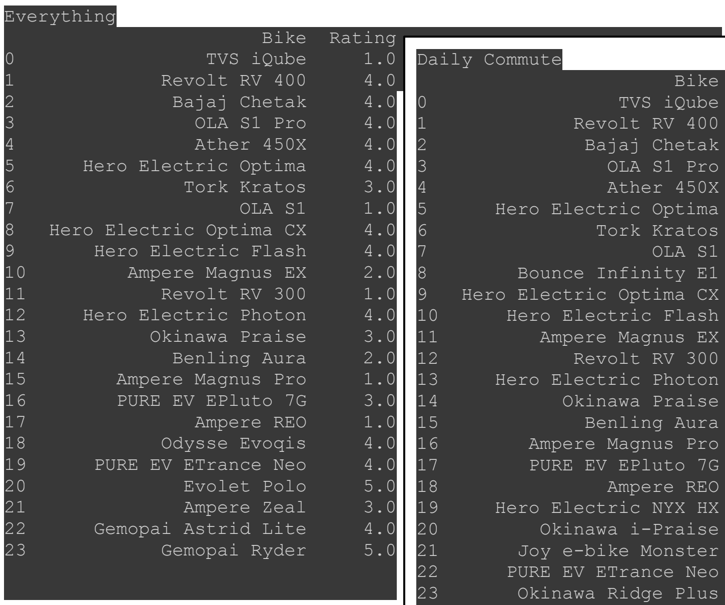
```
[51] segments =used.merge(diff_models, on='cluster_num', how='left')
     segments
```

	cluster_num	Used it for	Model Name
0	0	0.433155	18.689840
1	1	0.434164	17.188612
2	2	0.361702	18.744681
3	3	0.000000	19.450000

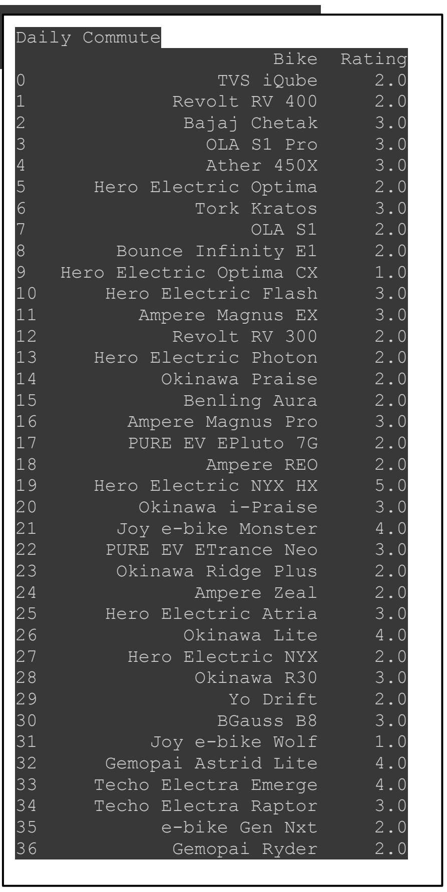


```
#Target segments

plt.figure(figsize = (9,4))
sns.scatterplot(x = "Used it for", y = "Model Name",data=segments,s=400, color="r")
plt.title("Simple Behavioural_segmentation for EV data",
          fontsize = 15)
plt.xlabel("Used it for", fontsize = 12)
plt.ylabel("Model Name", fontsize = 12)
plt.show()
```

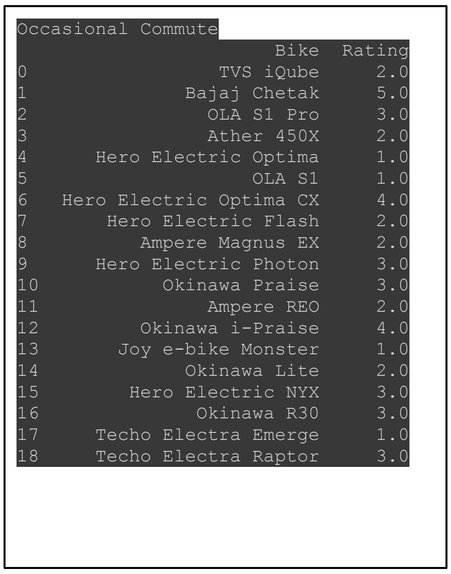
For the better conclusion we have calculated the overall rating for each bike with respect to the usage individually



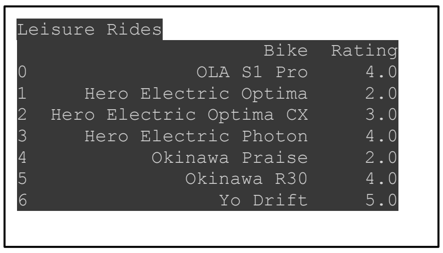
```
Everything
                        Bike  Rating
0                 TVS iQube      1.0
1               Revolt RV 400    4.0
2              Bajaj Chetak      4.0
3                OLA S1 Pro      4.0
4                Ather 450X      4.0
5      Hero Electric Optima      4.0
6               Tork Kratos      3.0
7                    OLA S1      1.0
8   Hero Electric Optima CX      4.0
9       Hero Electric Flash      4.0
10         Ampere Magnus EX      2.0
11            Revolt RV 300      1.0
12     Hero Electric Photon      4.0
13           Okinawa Praise      3.0
14             Benling Aura      2.0
15        Ampere Magnus Pro      1.0
16        PURE EV EPluto 7G      3.0
17               Ampere REO      1.0
18            Odysse Evoqis      4.0
19       PURE EV ETrance Neo     4.0
20              Evolet Polo      5.0
21              Ampere Zeal      3.0
22       Gemopai Astrid Lite     4.0
23            Gemopai Ryder      5.0
```



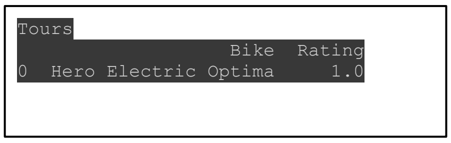
```
Daily Commute
                        Bike  Rating
0                 TVS iQube      2.0
1             Revolt RV 400      2.0
2              Bajaj Chetak      3.0
3                OLA S1 Pro      3.0
4                Ather 450X      3.0
5      Hero Electric Optima      2.0
6               Tork Kratos      3.0
7                    OLA S1      2.0
8        Bounce Infinity E1      2.0
9   Hero Electric Optima CX      1.0
10      Hero Electric Flash      3.0
11         Ampere Magnus EX      3.0
12            Revolt RV 300      2.0
13     Hero Electric Photon      2.0
14           Okinawa Praise      2.0
15             Benling Aura      2.0
16        Ampere Magnus Pro      3.0
17        PURE EV EPluto 7G      2.0
18               Ampere REO      2.0
19     Hero Electric NYX HX      5.0
20         Okinawa i-Praise      3.0
21       Joy e-bike Monster      4.0
22      PURE EV ETrance Neo      3.0
23       Okinawa Ridge Plus      2.0
24              Ampere Zeal      2.0
25      Hero Electric Atria      3.0
26             Okinawa Lite      4.0
27        Hero Electric NYX      2.0
28              Okinawa R30      3.0
29                 Yo Drift      2.0
30                BGauss B8      3.0
31          Joy e-bike Wolf      1.0
32      Gemopai Astrid Lite      4.0
33     Techo Electra Emerge      4.0
34     Techo Electra Raptor      3.0
35           e-bike Gen Nxt      2.0
36            Gemopai Ryder      2.0
```



```
Occasional Commute
                        Bike  Rating
0                 TVS iQube      2.0
1              Bajaj Chetak      5.0
2                OLA S1 Pro      3.0
3                Ather 450X      2.0
4      Hero Electric Optima      1.0
5                    OLA S1      1.0
6   Hero Electric Optima CX      4.0
7       Hero Electric Flash      2.0
8          Ampere Magnus EX      2.0
9      Hero Electric Photon      3.0
10           Okinawa Praise      3.0
11               Ampere REO      2.0
12         Okinawa i-Praise      4.0
13       Joy e-bike Monster      1.0
14             Okinawa Lite      2.0
15        Hero Electric NYX      3.0
16              Okinawa R30      3.0
17     Techo Electra Emerge      1.0
18     Techo Electra Raptor      3.0
```



```
Leisure Rides
                       Bike  Rating
0                OLA S1 Pro     4.0
1      Hero Electric Optima     2.0
2   Hero Electric Optima CX     3.0
3      Hero Electric Photon     4.0
4            Okinawa Praise     2.0
5               Okinawa R30     4.0
6                  Yo Drift     5.0
```



```
Tours
                    Bike  Rating
0  Hero Electric Optima      1.0
```

CONCLUSION :

conclusion for manual process:From the data and above interpretation , it clearly says that

for "Everything" usage	[Gemopai Ryder,Evolet Polo] bikes are more preferred
for 'Daily Commute" usage	[Hero Electric NYX HX] bike is more preferred
for "Occasional Commute" usage	[Bajaj Chetak] bike is more preferred
for "Leisure Rides" usage	[Yo Drift] bike is more preferred
for "Tours" usage	[Hero Electric Optima] bike is more preferred

The above conclusions are made after analysing all the data.i.e based on the ratings given by each customers for their corresponding bike.

GitHub links-
Balla Rakesh : https://github.com/Balla01/EV_SEGMENATAION
Shubhra Saxena : https://github.com/Shubhra2502/EV-Segmentation