Clustering cars on market - Unsupervised machine learning - Nemanja Kostić 1753

On this topic, We will try to devide cars in few clusters that will have potencial meaning that is valuable in commerc. Dataset I have used can be found on this link: https://www.kaggle.com/datasets/manishkr1754/cardekho-used-car-data

Libraries import

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.cluster import KMeans
from sklearn.preprocessing import MinMaxScaler
from sklearn.covariance import EllipticEnvelope
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score, davies_bouldin_score, calinski_haraba
import plotly.graph_objs as go
from scipy.cluster import hierarchy as sch
from sklearn.cluster import DBSCAN
from sklearn.cluster import AgglomerativeClustering
from sklearn.mixture import GaussianMixture
```

Data overview

- car_name - String that cointains brand + model in respective order.
- brand, name - String values of brand and neme values.
- vehicle_age - Integer value of cars age.
- km_driven - Integer value of how many kms have car been driven.
- seller_type - Who sold a car (3 categories)
- fuel_type, transmission_type - Petrol, diesel, automatic, manual...
- mileage - Average fuel consumption.
- engine - cubic cms of engines piston.
- max_power - HP of a car.
- seats - Number of seats.
- selling_price - Price that the car is been sold.

In [ ]:
```
data_frame = pd.read_csv("cardekho_dataset.csv")

# Shuffle dataset
data_frame_shuffled = data_frame#.sample(frac=1)
```

```
data_frame_shuffled.head()
```

Out[ ]:

	Unnamed: 0	car_name	brand	model	vehicle_age	km_driven	seller_type	fuel_type
0	0	Maruti Alto	Maruti	Alto	9	120000	Individual	Petrol
1	1	Hyundai Grand	Hyundai	Grand	5	20000	Individual	Petrol
2	2	Hyundai i20	Hyundai	i20	11	60000	Individual	Petrol
3	3	Maruti Alto	Maruti	Alto	9	37000	Individual	Petrol
4	4	Ford Ecosport	Ford	Ecosport	6	30000	Dealer	Diesel

Data manipulation

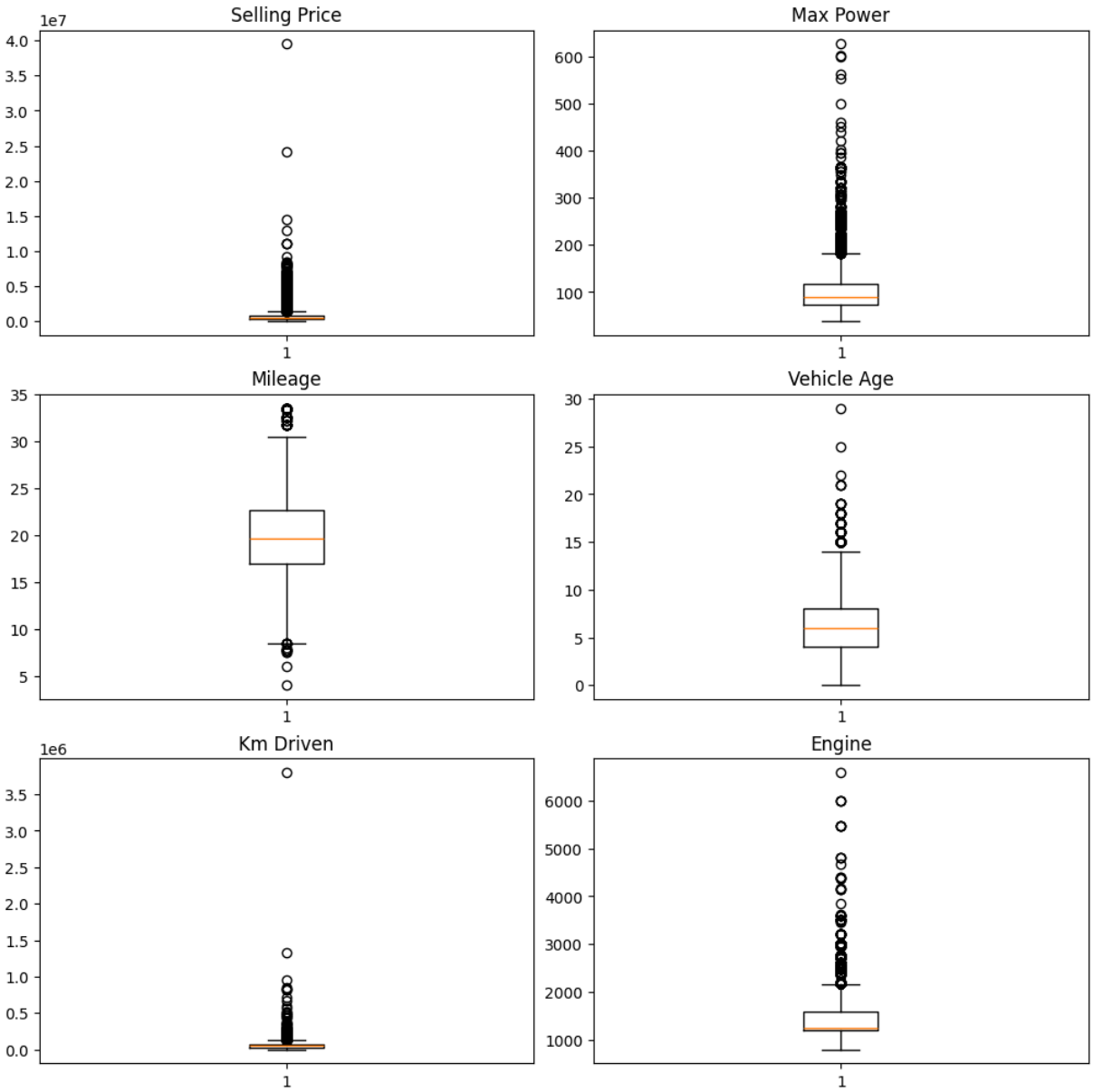
In [ ]:

```
# Deleting unvanted column for indexing
data_frame_shuffled.drop("Unnamed: 0", axis=1, inplace=True)

# Droping instances with null values, there is none
inicial_count = len(data_frame_shuffled)
data_frame_shuffled.dropna(inplace=True)
count_after_missig_values = len(data_frame_shuffled)
print("Droped " + str(inicial_count-count_after_missig_values) + " rows with missin

#Deleting duplicates, 167 duplicates of 14000
inicial_count = len(data_frame_shuffled)
data_frame_shuffled.drop_duplicates()
count_after_missig_values = len(data_frame_shuffled)
print("Droped " + str(inicial_count-count_after_missig_values) + " duplicate rows")


numerical_features = ['vehicle_age', 'km_driven', 'selling_price', 'max_power', 'en
categorical_features = ['brand', 'seats', 'seller_type', 'fuel_type', 'transmission

# Create subplots
fig, axs = plt.subplots(3, 2, figsize=(10, 10), constrained_layout=True)

# Plot each boxplot
axs[0, 0].boxplot(data_frame_shuffled['selling_price'])
axs[0, 0].set_title('Selling Price')

axs[0, 1].boxplot(data_frame_shuffled['max_power'])
axs[0, 1].set_title('Max Power')

axs[1, 0].boxplot(data_frame_shuffled['mileage'])
```

```
axs[1, 0].set_title('Mileage')

axs[1, 1].boxplot(data_frame_shuffled['vehicle_age'])
axs[1, 1].set_title('Vehicle Age')

axs[2, 0].boxplot(data_frame_shuffled['km_driven'])
axs[2, 0].set_title('Km Driven')

axs[2, 1].boxplot(data_frame_shuffled['engine'])
axs[2, 1].set_title('Engine')
```

```
Droped 0 rows with missing values
Droped 0 duplicate rows
```

Out[ ]: Text(0.5, 1.0, 'Engine')



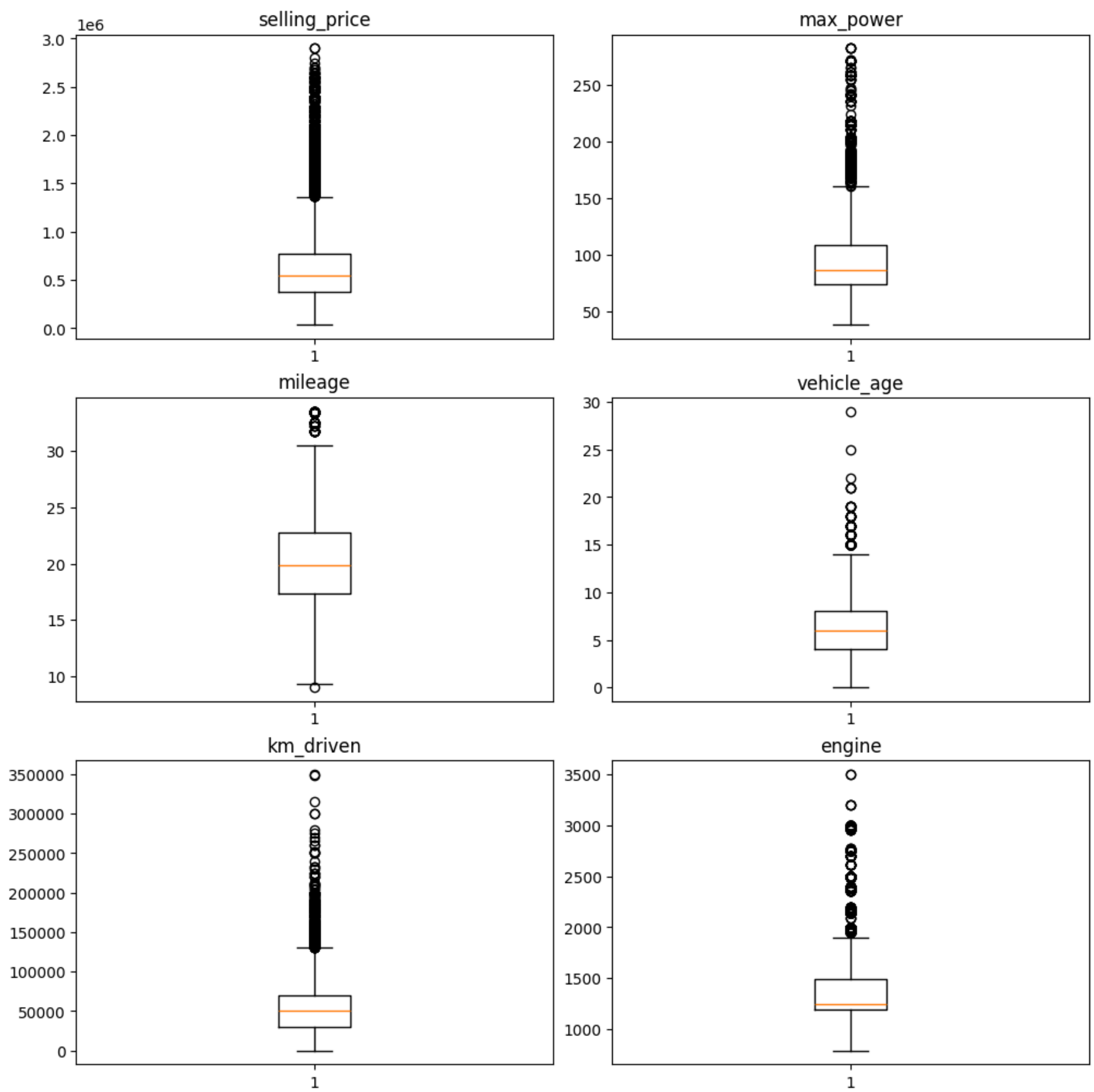
After outliers removal

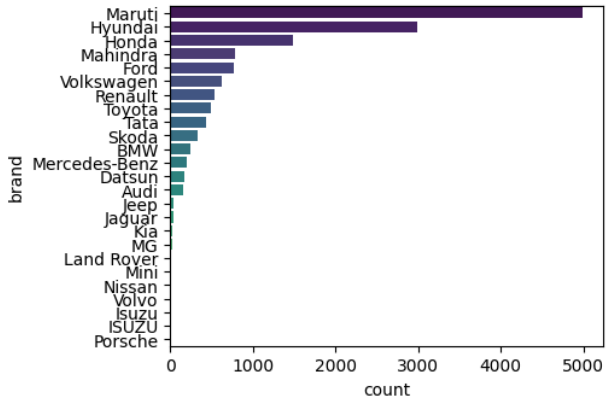
In [ ]:
```
outlier_detector = EllipticEnvelope(contamination=.07)
```

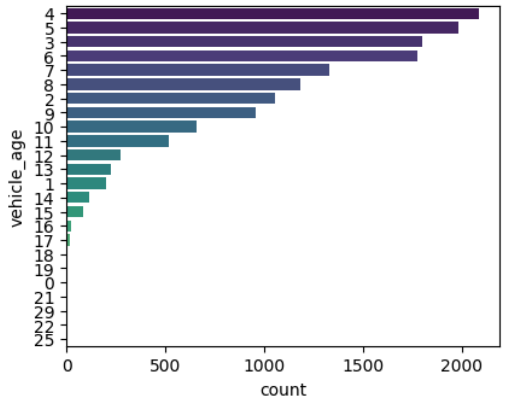
```
# Fit detector
outlier_detector.fit(data_frame_shuffled[numerical_features])

inicial_count = len(data_frame_shuffled)

# Predict outliers
outliers = outlier_detector.predict(data_frame_shuffled[numerical_features])

# Outliers deletation
outliers_indices = outliers == -1
data_frame_shuffled = data_frame_shuffled[~outliers_indices]

# Deletation of cars thath doesn't have seats
data_frame_shuffled = data_frame_shuffled.drop(data_frame_shuffled[data_frame_shuff

data_frame_shuffled_original = pd.DataFrame(data_frame_shuffled)

# Plot each boxplot
fig, axs = plt.subplots(3, 2, figsize=(10,10), constrained_layout=True)
categorical = ['selling_price', 'max_power', 'mileage', 'vehicle_age', 'km_driven',
for i, f in enumerate(categorical):
    axs[i//2][i%2].boxplot(data_frame_shuffled[f])
    axs[i//2][i%2].set_title(f)

count_after_missig_values = len(data_frame_shuffled)
print("Outliers Droped: " + str(inicial_count-count_after_missig_values))

fig, axs = plt.subplots(3, 2, figsize=(10,10), constrained_layout=True)
categorical = ['brand', 'vehicle_age', 'seller_type', 'fuel_type', 'seats', 'transm
for i, f in enumerate(categorical):
    sns.countplot(y=f, data=data_frame_shuffled, ax=axs[i//2][i%2], order=data_fram

fig, axs = plt.subplots(3, 2, figsize=(10,10), constrained_layout=True)
numerical = ['km_driven', 'selling_price', 'max_power', 'engine', 'mileage']
for i, f in enumerate(numerical):
    sns.histplot(x=f, data=data_frame_shuffled, ax=axs[i//2][i%2], bins=100)
```

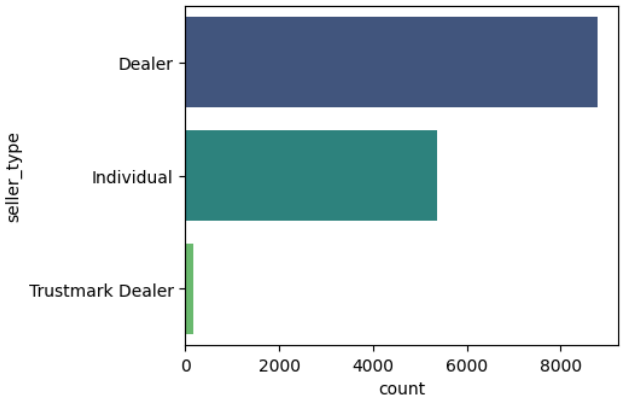
```
Outliers Droped: 1081
```

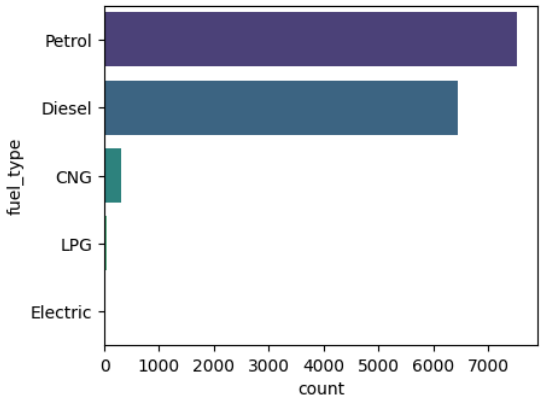
```
c:\Users\Nemanja\anaconda3\envs\mlenv\Lib\site-packages\seaborn\_oldcore.py:1119: Fu
tureWarning: use_inf_as_na option is deprecated and will be removed in a future vers
ion. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Nemanja\anaconda3\envs\mlenv\Lib\site-packages\seaborn\_oldcore.py:1119: Fu
tureWarning: use_inf_as_na option is deprecated and will be removed in a future vers
ion. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Nemanja\anaconda3\envs\mlenv\Lib\site-packages\seaborn\_oldcore.py:1119: Fu
tureWarning: use_inf_as_na option is deprecated and will be removed in a future vers
ion. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Nemanja\anaconda3\envs\mlenv\Lib\site-packages\seaborn\_oldcore.py:1119: Fu
tureWarning: use_inf_as_na option is deprecated and will be removed in a future vers
ion. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Nemanja\anaconda3\envs\mlenv\Lib\site-packages\seaborn\_oldcore.py:1119: Fu
tureWarning: use_inf_as_na option is deprecated and will be removed in a future vers
ion. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

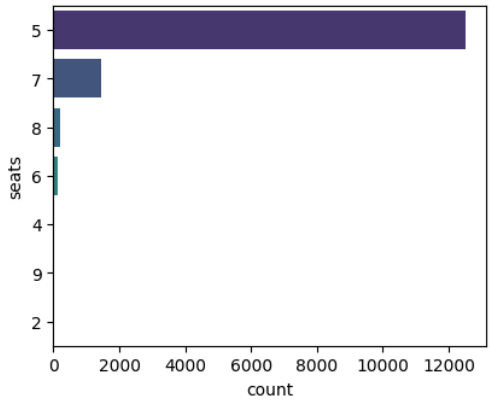


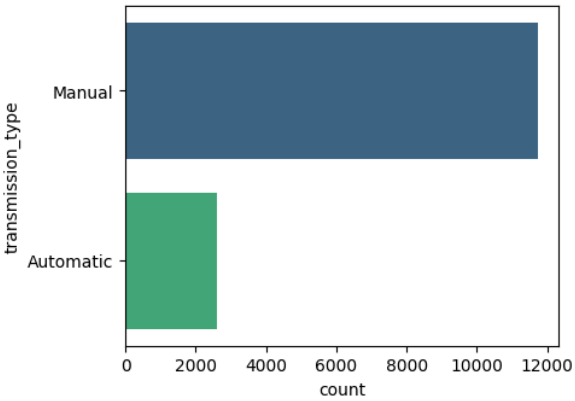

brand
Maruti
Hyundai
Honda
Mahindra
Ford
Volkswagen
Renault
Toyota
Tata
Skoda
BMW
Mercedes-Benz
Datsun
Audi
Jeep
Jaguar
Kia
MG
Land Rover
Mini
Nissan
Volvo
Isuzu
ISUZU
Porsche
0 1000 2000 3000 4000 5000
count

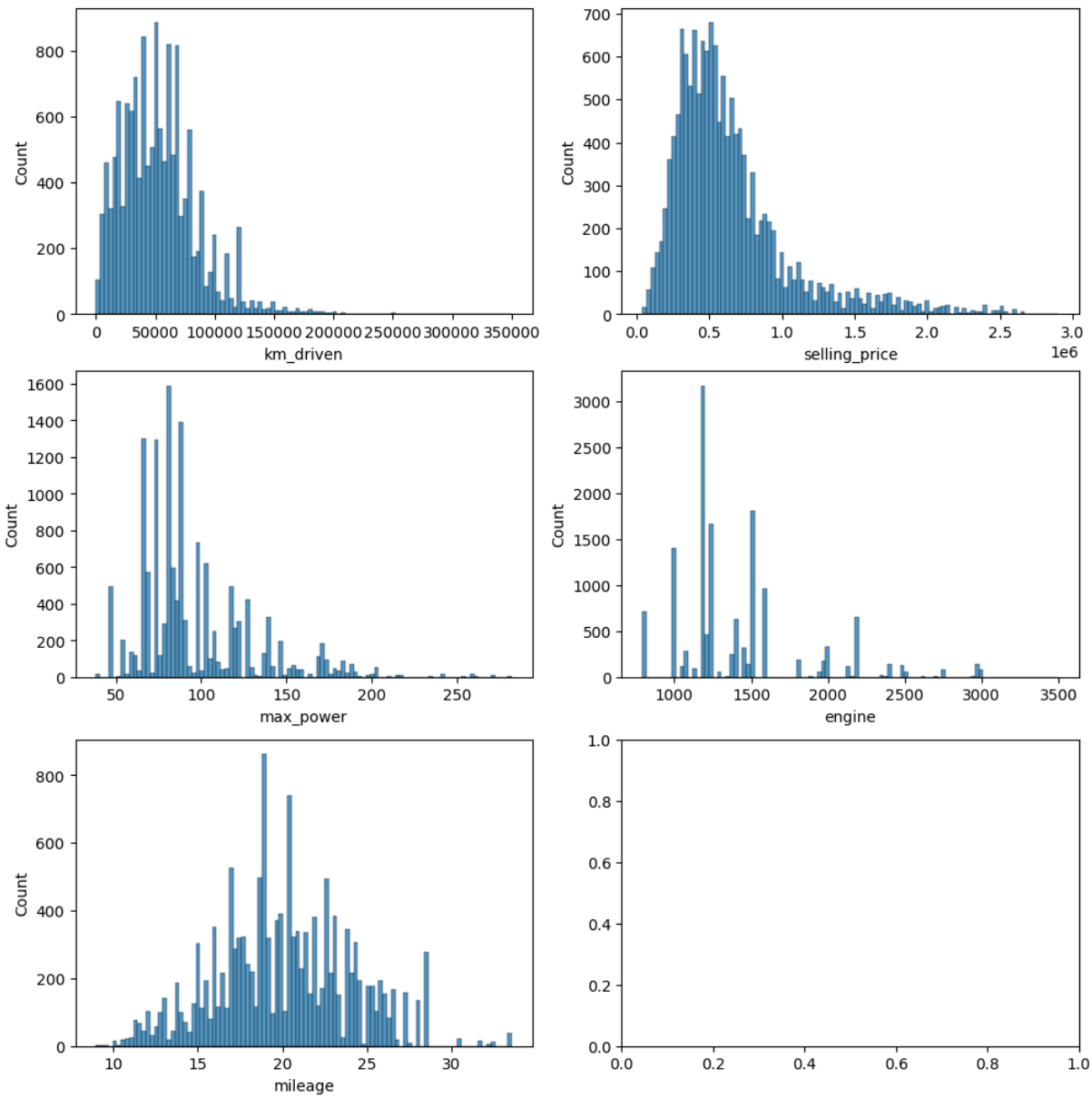

vehicle_age
4
5
3
6
7
8
2
9
10
11
12
13
1
14
15
16
17
18
19
0
21
29
22
25
0 500 1000 1500 2000
count


seller_type
Dealer
Individual
Trustmark Dealer
0 2000 4000 6000 8000
count


fuel_type
Petrol
Diesel
CNG
LPG
Electric
0 1000 2000 3000 4000 5000 6000 7000
count


seats
5
7
8
6
4
9
2
0 2000 4000 6000 8000 10000 12000
count


transmission_type
Manual
Automatic
0 2000 4000 6000 8000 10000 12000
count



In [ ]:
```
data_frame_shuffled.drop("car_name", axis=1, inplace=True)
data_frame_shuffled.drop("brand", axis=1, inplace=True)
data_frame_shuffled.drop("model", axis=1, inplace=True)

# Droping cars without seats
data_frame_shuffled = data_frame_shuffled.drop(data_frame_shuffled[data_frame_shuff

# Trustmark dealers have very low number of instances and will be categorized as De
data_frame_shuffled['seller_type'] = data_frame_shuffled['seller_type'].replace(['T

fuel_type_mapping = {fuel_type: idx for idx, fuel_type in enumerate(data_frame_shuf
data_frame_shuffled['fuel_type'] = data_frame_shuffled['fuel_type'].map(fuel_type_m

transmission_type_mapping = {trans_type: idx for idx, trans_type in enumerate(data_
data_frame_shuffled['transmission_type'] = data_frame_shuffled['transmission_type']

# Integer encoding seller_type feature
seller_type_mapping = {seller_type: idx for idx, seller_type in enumerate(data_fram
data_frame_shuffled['seller_type'] = data_frame_shuffled['seller_type'].map(seller_
```

```
# Integer encoding brand feature
#brand_mapping = {brand: idx for idx, brand in enumerate(data_frame_shuffled['brand
#data_frame_shuffled['brand'] = data_frame_shuffled['brand'].map(brand_mapping)

# Integer encoding seats feature
transmission_type_mapping = {seats: idx for idx, seats in enumerate(data_frame_shuf
data_frame_shuffled['seats'] = data_frame_shuffled['seats'].map(transmission_type_m

## One-hot encoding seller_type feature
# data_frame_shuffled = pd.get_dummies(data_frame_shuffled, columns=['seller_type']

## One-hot encoding fuel_type feature
#data_frame_shuffled = pd.get_dummies(data_frame_shuffled, columns=['fuel_type'])

## One-hot encoding transmission_type feature
#data_frame_shuffled = pd.get_dummies(data_frame_shuffled, columns=['transmission_t

# Discretization of selling_price feature
data_frame_shuffled['selling_price'] = np.digitize(data_frame_shuffled['selling_pri

# Discretization of km_driven feature
data_frame_shuffled['km_driven'] = np.digitize(data_frame_shuffled['km_driven'], bi



# Extract numerical features for clustering
numerical_features = ['vehicle_age', 'max_power', 'engine', 'mileage']
categorical_features = ['brand', 'seats', 'seller_type', 'fuel_type', 'transmission

scaler = MinMaxScaler(feature_range=(-1, 1))
data_frame_shuffled[numerical_features] = scaler.fit_transform(data_frame_shuffled[

# Display the enumerated data
data_frame_shuffled.head()
```

Out[ ]:

	vehicle_age	km_driven	seller_type	fuel_type	transmission_type	mileage	engine
0	-0.379310	2	0	0	0	-0.127954	-0.997782
1	-0.655172	0	0	0	0	-0.193154	-0.701294
2	-0.241379	1	0	0	0	-0.348003	-0.701294
3	-0.379310	1	0	0	0	-0.028525	-0.848429
4	-0.586207	0	1	1	0	0.122249	-0.478743

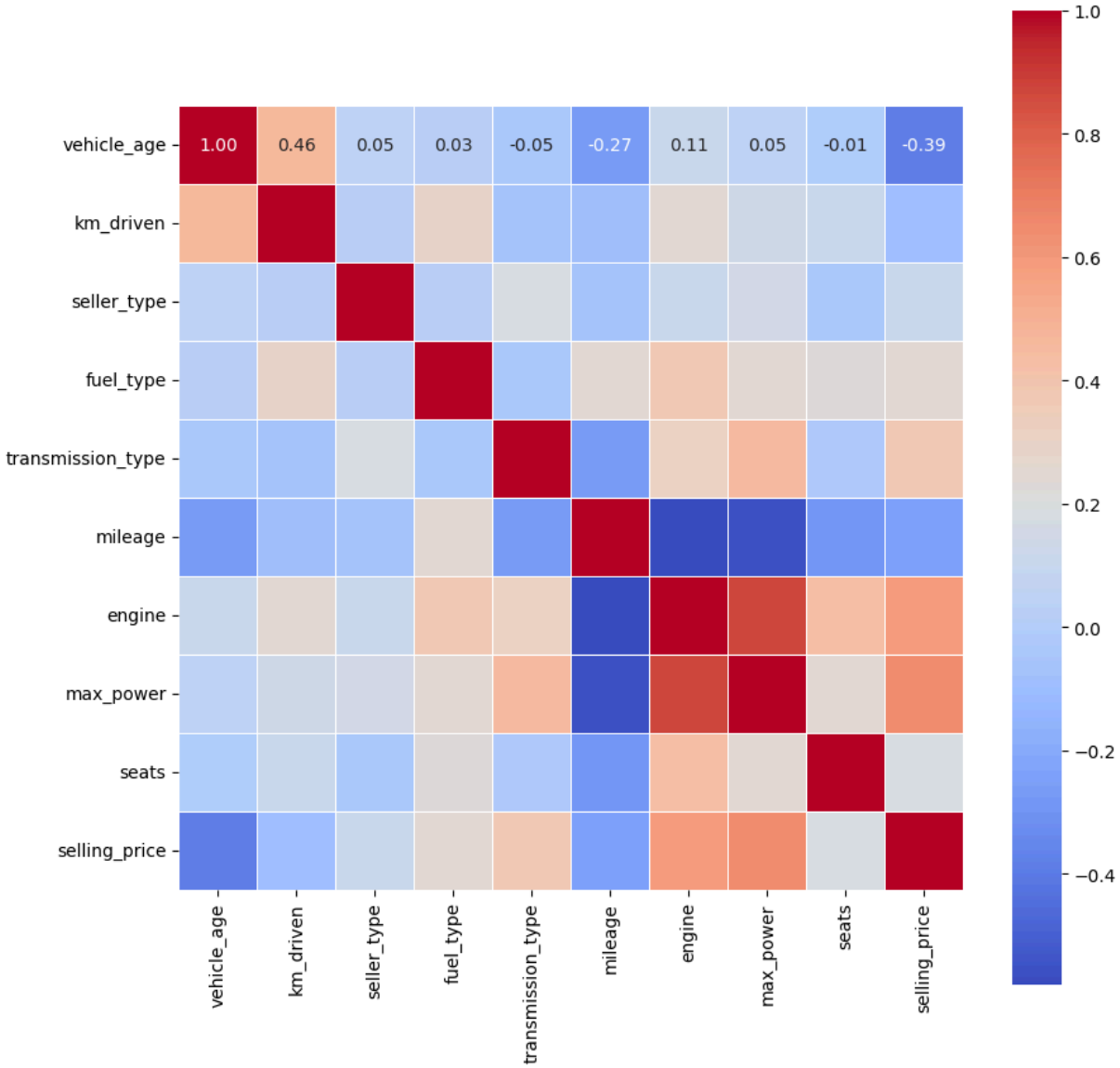
Correlation matrix

In [ ]:

```
corr_matrix = data_frame_shuffled.corr()
print(corr_matrix)
plt.figure(figsize=(10,10))
```

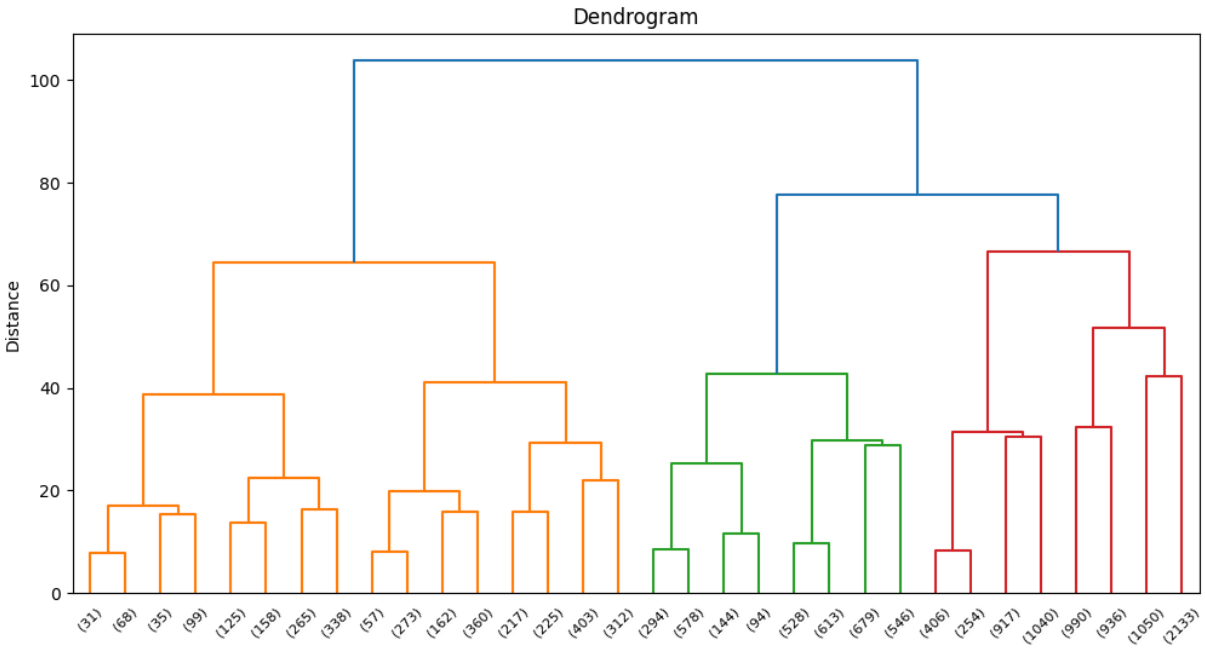
```
sns.heatmap(corr_matrix, annot=True, fmt='.2f', square=True, cmap='coolwarm', linew
plt.show()
```

```
                  vehicle_age  km_driven  seller_type  fuel_type  \
vehicle_age          1.000000   0.458249     0.053116   0.028541
km_driven            0.458249   1.000000     0.013387   0.300888
seller_type          0.053116   0.013387     1.000000   0.017563
fuel_type            0.028541   0.300888     0.017563   1.000000
transmission_type   -0.047159  -0.063008     0.178929  -0.039335
mileage             -0.268048  -0.085036    -0.070125   0.245665
engine               0.111430   0.242393     0.105832   0.381795
max_power            0.050232   0.136535     0.151149   0.237623
seats               -0.007538   0.099738    -0.034174   0.234377
selling_price       -0.385601  -0.090873     0.114709   0.246275

                  transmission_type   mileage    engine  max_power     seats  \
vehicle_age               -0.047159 -0.268048  0.111430   0.050232 -0.007538
km_driven                 -0.063008 -0.085036  0.242393   0.136535  0.099738
seller_type                0.178929 -0.070125  0.105832   0.151149 -0.034174
fuel_type                 -0.039335  0.245665  0.381795   0.237623  0.234377
transmission_type          1.000000 -0.260527  0.303460   0.455327 -0.023990
mileage                   -0.260527  1.000000 -0.579147  -0.558818 -0.279839
engine                     0.303460 -0.579147  1.000000   0.875178  0.438367
max_power                  0.455327 -0.558818  0.875178   1.000000  0.239499
seats                     -0.023990 -0.279839  0.438367   0.239499  1.000000
selling_price              0.365439 -0.249129  0.585273   0.652083  0.189595

                  selling_price
vehicle_age           -0.385601
km_driven             -0.090873
seller_type            0.114709
fuel_type              0.246275
transmission_type      0.365439
mileage               -0.249129
engine                 0.585273
max_power              0.652083
seats                  0.189595
selling_price          1.000000
```



Dendrogram

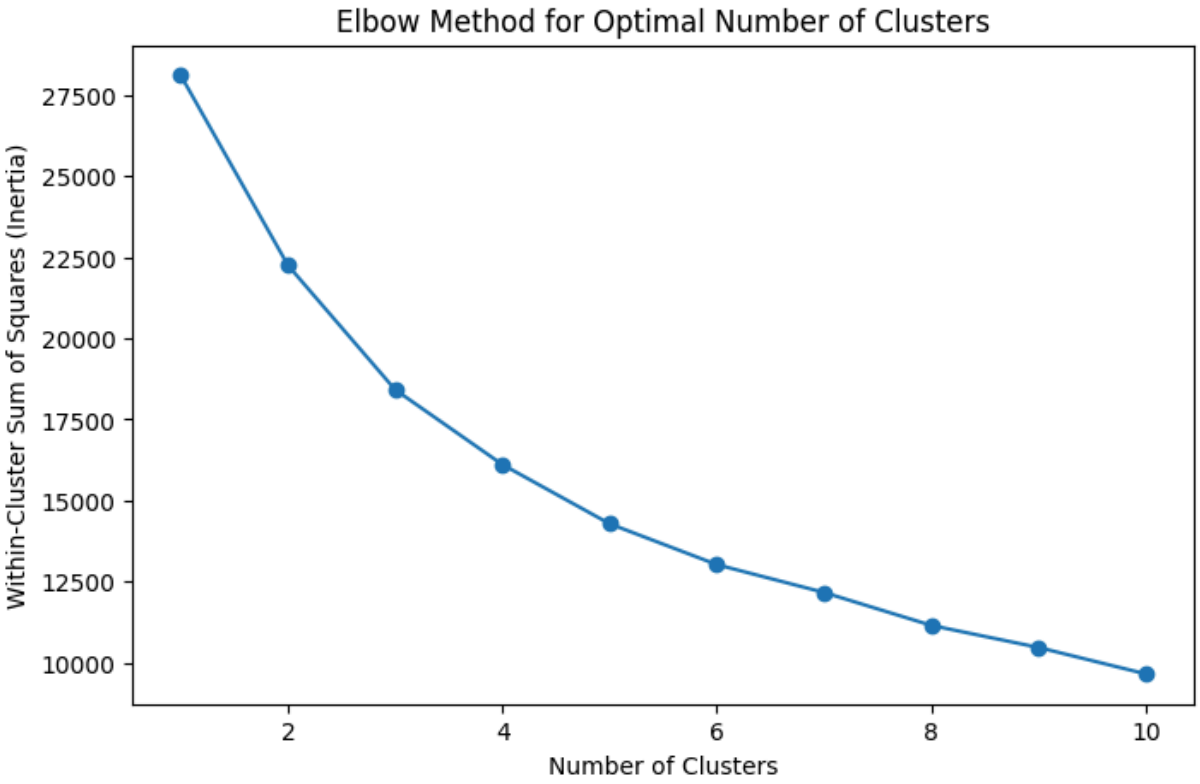
In [ ]:
```
# Plot the dendrogram
plt.figure(figsize=(12, 6))
dendrogram = sch.dendrogram(sch.linkage(data_frame_shuffled.values, method='ward'),

plt.title('Dendrogram')
plt.ylabel('Distance')
plt.show()
```



Elbow Method for Optimal Number of Clusters

In [ ]:
```
inertia_values = []
for i in range(1, 11):  # Trying clusters from 1 to 10
    kmeans = KMeans(n_clusters=i, random_state=42)
    kmeans.fit(data_frame_shuffled)
    inertia_values.append(kmeans.inertia_)

# Plot the elbow method
plt.figure(figsize=(8, 5))
plt.plot(range(1, 11), inertia_values, marker='o')
plt.title('Elbow Method for Optimal Number of Clusters')
plt.xlabel('Number of Clusters')
plt.ylabel('Within-Cluster Sum of Squares (Inertia)')
plt.show()
```

```
C:\Users\Nemanja\AppData\Roaming\Python\Python311\site-packages\sklearn\cluster\_kme
ans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'au
to' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Nemanja\AppData\Roaming\Python\Python311\site-packages\sklearn\cluster\_kme
ans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'au
to' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Nemanja\AppData\Roaming\Python\Python311\site-packages\sklearn\cluster\_kme
ans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'au
to' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Nemanja\AppData\Roaming\Python\Python311\site-packages\sklearn\cluster\_kme
ans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'au
to' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Nemanja\AppData\Roaming\Python\Python311\site-packages\sklearn\cluster\_kme
ans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'au
to' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Nemanja\AppData\Roaming\Python\Python311\site-packages\sklearn\cluster\_kme
ans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'au
to' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Nemanja\AppData\Roaming\Python\Python311\site-packages\sklearn\cluster\_kme
ans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'au
to' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Nemanja\AppData\Roaming\Python\Python311\site-packages\sklearn\cluster\_kme
ans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'au
to' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Nemanja\AppData\Roaming\Python\Python311\site-packages\sklearn\cluster\_kme
ans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'au
to' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Nemanja\AppData\Roaming\Python\Python311\site-packages\sklearn\cluster\_kme
ans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'au
to' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```



PCA

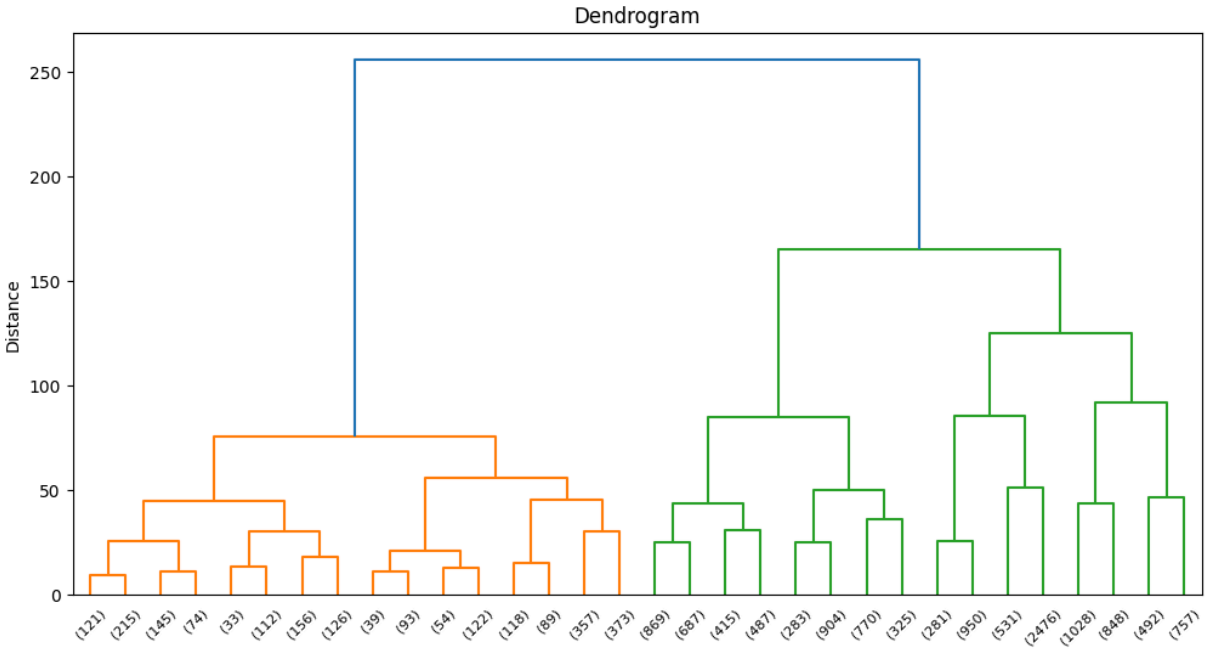
In [ ]:
```
# scaling
from scipy.stats import zscore
from sklearn.decomposition import PCA

data_frame_shuffled_Scaled=data_frame_shuffled.apply(zscore)
data_frame_shuffled_Scaled.head()
# Covariance
covMatrix = np.cov(data_frame_shuffled_Scaled,rowvar=False)
print(covMatrix)
# PCA
pca = PCA(n_components=10)
pca.fit(data_frame_shuffled_Scaled)
#Eigenvalues
pca.explained_variance_ratio_.cumsum()
print("next")

# based on the values chose 3 best components and then use Regression with these co
pca3 = PCA(n_components=3)
pca3.fit(data_frame_shuffled_Scaled)
print(pca3.explained_variance_ratio_.cumsum())

data_frame_pca3 = pd.DataFrame(pca3.transform(data_frame_shuffled_Scaled), columns=
```

```
[[ 1.00006979  0.45828098  0.05311995  0.02854341 -0.04716189 -0.2680672
   0.11143787  0.05023524 -0.00753897 -0.38562799]
 [ 0.45828098  1.00006979  0.01338765  0.30090857 -0.06301269 -0.08504195
   0.24241037  0.13654472  0.09974525 -0.09087887]
 [ 0.05311995  0.01338765  1.00006979  0.01756407  0.17894189 -0.07012981
   0.10583959  0.15115973 -0.03417592  0.11471698]
 [ 0.02854341  0.30090857  0.01756407  1.00006979 -0.03933798  0.24568176
   0.38182139  0.23764006  0.23439317  0.2462926 ]
 [-0.04716189 -0.06301269  0.17894189 -0.03933798  1.00006979 -0.2605447
   0.30348157  0.45535891 -0.0239918   0.36546477]
 [-0.2680672  -0.08504195 -0.07012981  0.24568176 -0.2605447   1.00006979
  -0.57918698 -0.55885676 -0.27985882 -0.2491463 ]
 [ 0.11143787  0.24241037  0.10583959  0.38182139  0.30348157 -0.57918698
   1.00006979  0.87523904  0.43839787  0.58531428]
 [ 0.05023524  0.13654472  0.15115973  0.23764006  0.45535891 -0.55885676
   0.87523904  1.00006979  0.23951548  0.65212861]
 [-0.00753897  0.09974525 -0.03417592  0.23439317 -0.0239918  -0.27985882
   0.43839787  0.23951548  1.00006979  0.18960845]
 [-0.38562799 -0.09087887  0.11471698  0.2462926   0.36546477 -0.2491463
   0.58531428  0.65212861  0.18960845  1.00006979]]
next
[0.32886444 0.50238366 0.64326669]
```

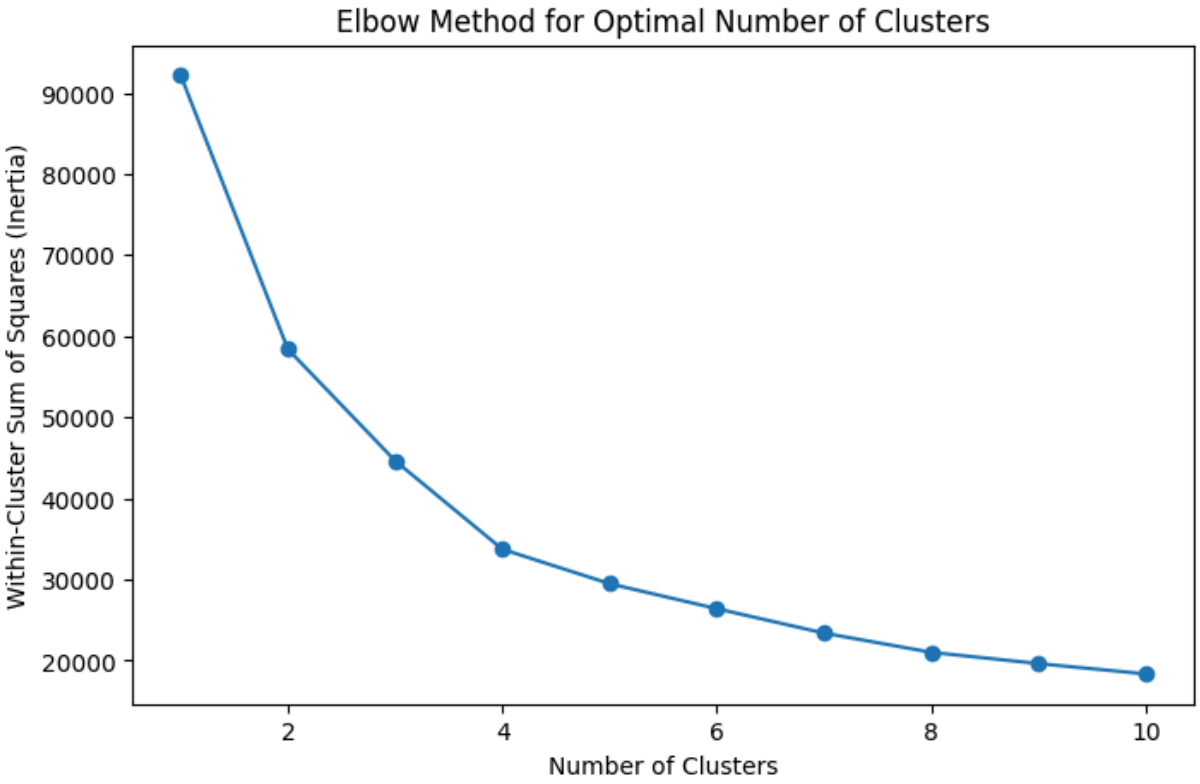
Elbow method and dendogram after PCA

In [ ]:

```
# Plot the dendrogram
plt.figure(figsize=(12, 6))
dendrogram = sch.dendrogram(sch.linkage(data_frame_pca3.values, method='ward'), tru

plt.title('Dendrogram')
plt.ylabel('Distance')
plt.show()


inertia_values = []
for i in range(1, 11):  # Trying clusters from 1 to 10
    kmeans = KMeans(n_clusters=i, random_state=42)
    kmeans.fit(data_frame_pca3)
    inertia_values.append(kmeans.inertia_)

# Plot the elbow method
plt.figure(figsize=(8, 5))
plt.plot(range(1, 11), inertia_values, marker='o')
plt.title('Elbow Method for Optimal Number of Clusters')
plt.xlabel('Number of Clusters')
plt.ylabel('Within-Cluster Sum of Squares (Inertia)')
plt.show()
```


Dendrogram
Distance
250
200
150
100
50
0
(121)
(215)
(145)
(74)
(33)
(112)
(156)
(126)
(39)
(93)
(54)
(122)
(118)
(89)
(357)
(373)
(869)
(687)
(415)
(487)
(283)
(904)
(770)
(325)
(281)
(950)
(531)
(2476)
(1028)
(848)
(492)
(757)

```
C:\Users\Nemanja\AppData\Roaming\Python\Python311\site-packages\sklearn\cluster\_kme
ans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'au
to' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Nemanja\AppData\Roaming\Python\Python311\site-packages\sklearn\cluster\_kme
ans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'au
to' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Nemanja\AppData\Roaming\Python\Python311\site-packages\sklearn\cluster\_kme
ans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'au
to' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Nemanja\AppData\Roaming\Python\Python311\site-packages\sklearn\cluster\_kme
ans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'au
to' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Nemanja\AppData\Roaming\Python\Python311\site-packages\sklearn\cluster\_kme
ans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'au
to' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Nemanja\AppData\Roaming\Python\Python311\site-packages\sklearn\cluster\_kme
ans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'au
to' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Nemanja\AppData\Roaming\Python\Python311\site-packages\sklearn\cluster\_kme
ans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'au
to' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Nemanja\AppData\Roaming\Python\Python311\site-packages\sklearn\cluster\_kme
ans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'au
to' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Nemanja\AppData\Roaming\Python\Python311\site-packages\sklearn\cluster\_kme
ans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'au
to' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\Nemanja\AppData\Roaming\Python\Python311\site-packages\sklearn\cluster\_kme
ans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'au
to' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```



Clustering and visualization

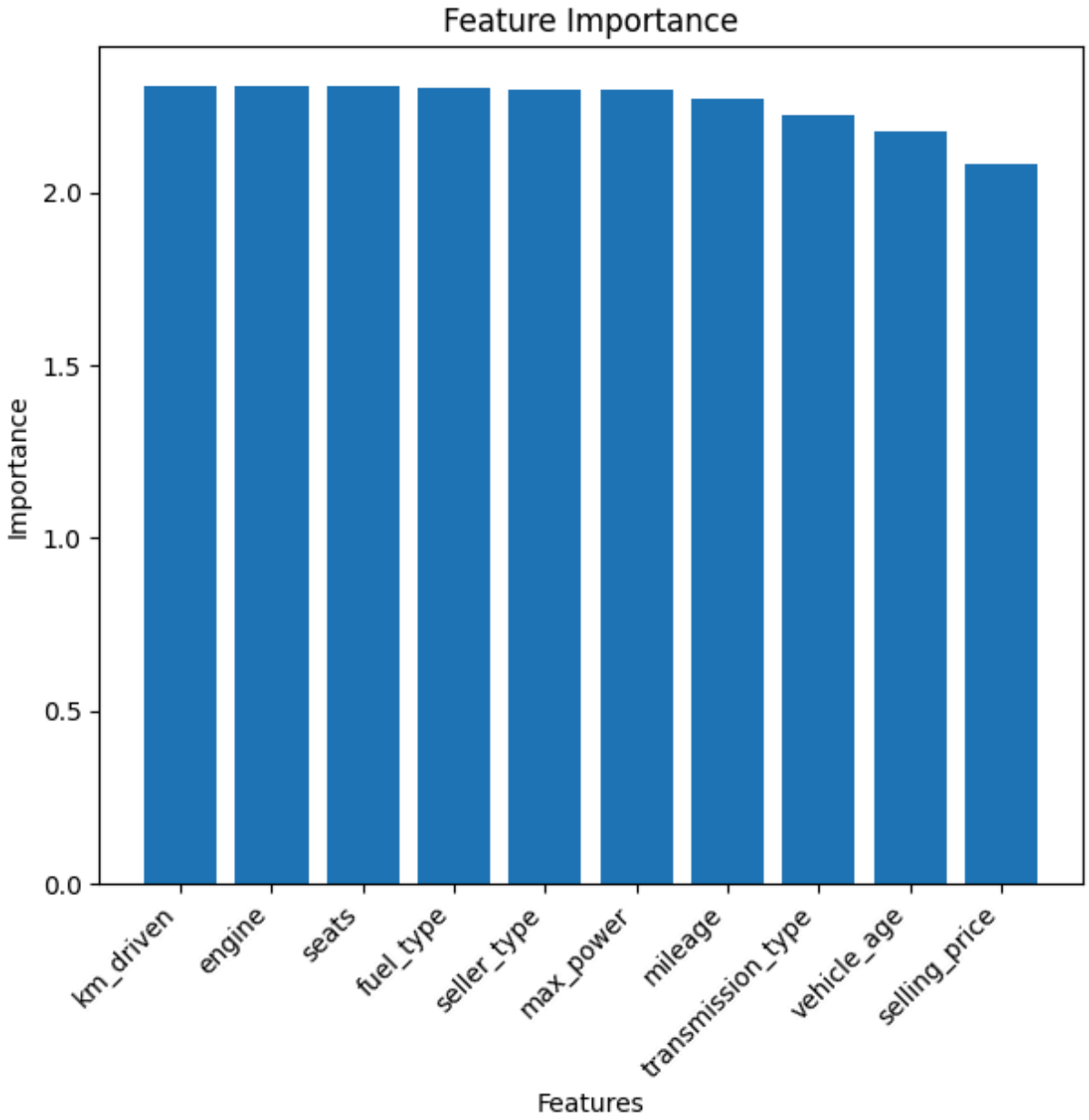
K-Means

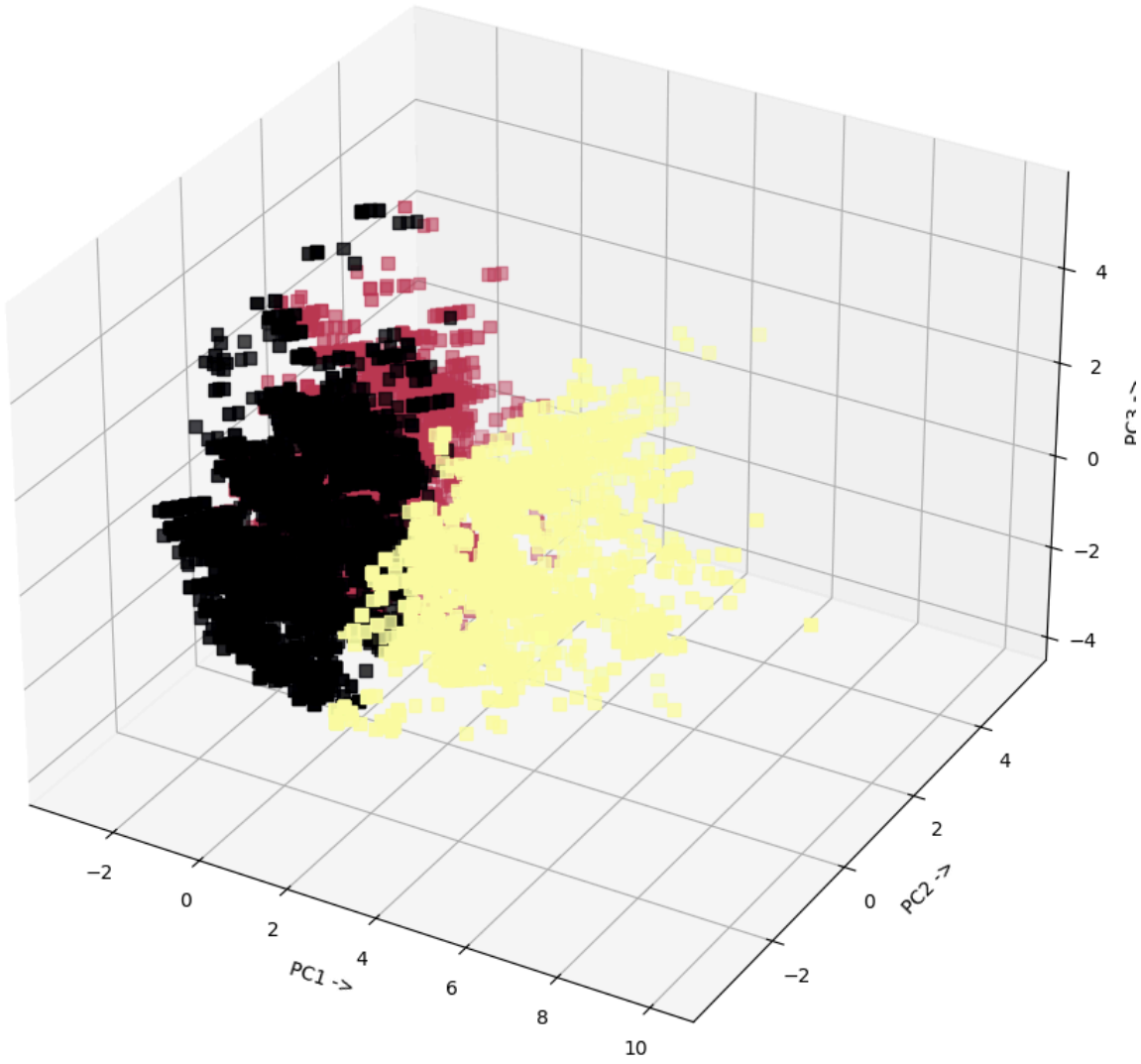
K-Means, 3 clusters

In [ ]:
```
# Apply K-Means clustering
num_clusters = 3
kmeans = KMeans(n_clusters=num_clusters, random_state=42)
data_frame_shuffled['cluster'] = kmeans.fit_predict(data_frame_pca3)

data_frame_shuffled_original['cluster'] = data_frame_shuffled['cluster']

data_frame_shuffled_original.to_csv('result_kmeans_3clusters.csv')

# Compute mean values of features within each cluster
cluster_means = data_frame_shuffled.groupby('cluster').mean()

# Calculate feature importance by comparing mean values across clusters
feature_importance = cluster_means.sub(cluster_means.mean()).abs().sum() / cluster_

# Sort features by importance
feature_importance_sorted = feature_importance.sort_values(ascending=False)

# Plot feature importance
plt.figure(figsize=(7, 6))
plt.bar(feature_importance_sorted.index, feature_importance_sorted)
```

```
plt.xlabel('Features')
plt.ylabel('Importance')
plt.title('Feature Importance')
plt.xticks(rotation=45, ha='right')
plt.show()


fig = plt.figure(figsize = (11,11))
ax = fig.add_subplot(111, projection='3d')
x = np.array(data_frame_pca3['PC1'])
y = np.array(data_frame_pca3['PC2'])
z = np.array(data_frame_pca3['PC3'])

ax.scatter(x,y,z, marker="s", c=data_frame_shuffled["cluster"], s=40, cmap="inferno
ax.set_xlabel('PC1 ->')
ax.set_ylabel('PC2 ->')
ax.set_zlabel('PC3 ->')

plt.show()


# 3d scatterplot using plotly
Scene = dict(xaxis = dict(title  = 'PC1'),yaxis = dict(title  = 'PC2'),zaxis = dict
labels = kmeans.labels_
trace = go.Scatter3d(x=data_frame_pca3['PC1'], y=data_frame_pca3['PC2'], z=data_fra
layout = go.Layout(margin=dict(l=0,r=0),scene = Scene,height = 800,width = 800)
data = [trace]
fig = go.Figure(data = data, layout = layout)
fig.show()


# Calculate clustering metrics
silhouette_kmeans_3 = silhouette_score(data_frame_pca3, kmeans.labels_)
db_index_kmeans_3 = davies_bouldin_score(data_frame_pca3, kmeans.labels_)
ch_index_kmeans_3 = calinski_harabasz_score(data_frame_pca3, kmeans.labels_)


# Print the metric scores
print(f"Silhouette Score: {silhouette_kmeans_3:.2f}")
print(f"Davies-Bouldin Index: {db_index_kmeans_3:.2f}")
print(f"Calinski-Harabasz Index: {ch_index_kmeans_3:.2f}")
```

```
C:\Users\Nemanja\AppData\Roaming\Python\Python311\site-packages\sklearn\cluster\_kme
ans.py:1416: FutureWarning: The default value of `n_init` will change from 10 to 'au
to' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```


Feature Importance
Importance
0.0
0.5
1.0
1.5
2.0
km_driven
engine
seats
fuel_type
seller_type
max_power
mileage
transmission_type
vehicle_age
selling_price
Features



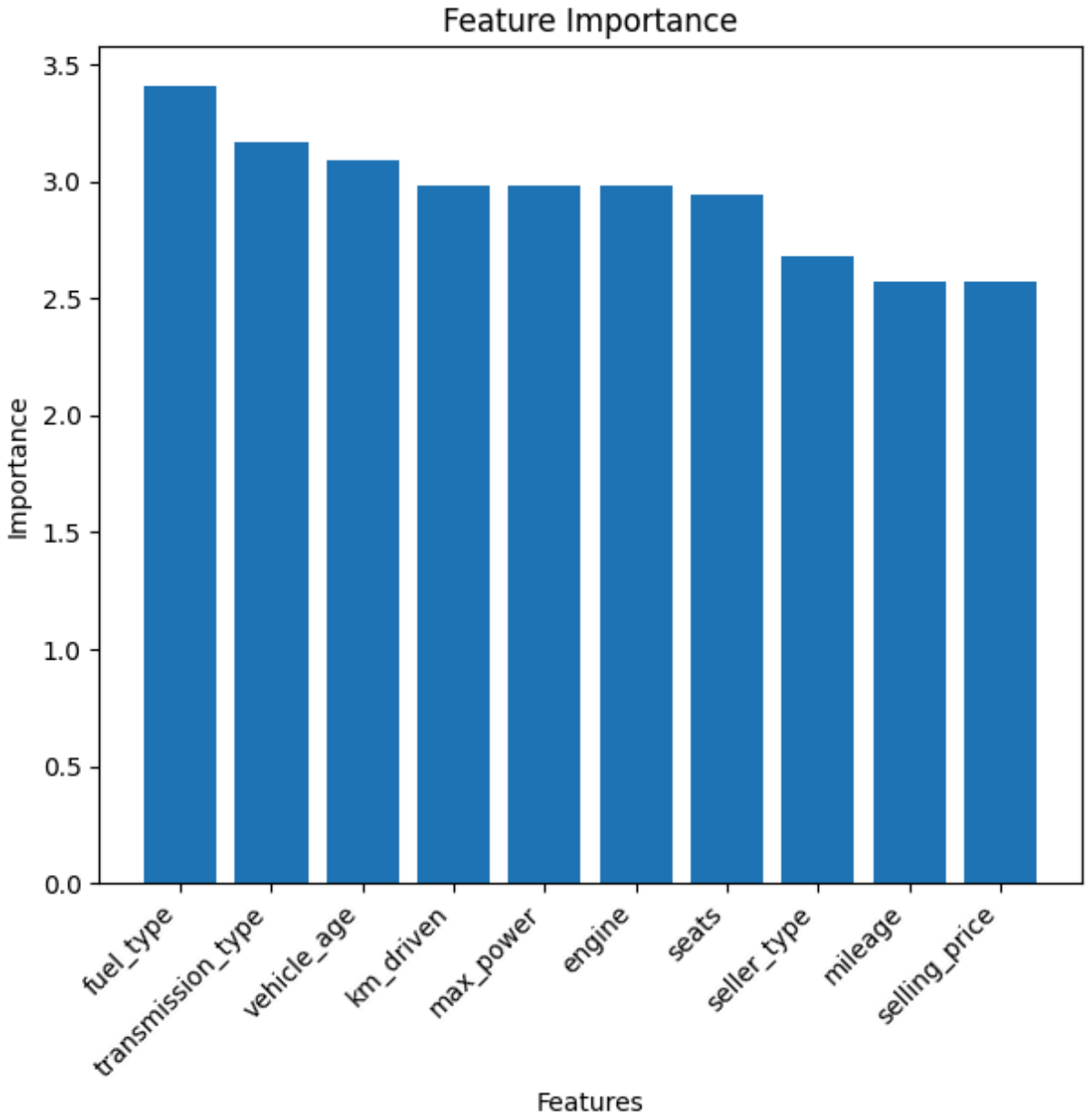

```
Silhouette Score: 0.29
Davies-Bouldin Index: 1.25
Calinski-Harabasz Index: 7628.68
```

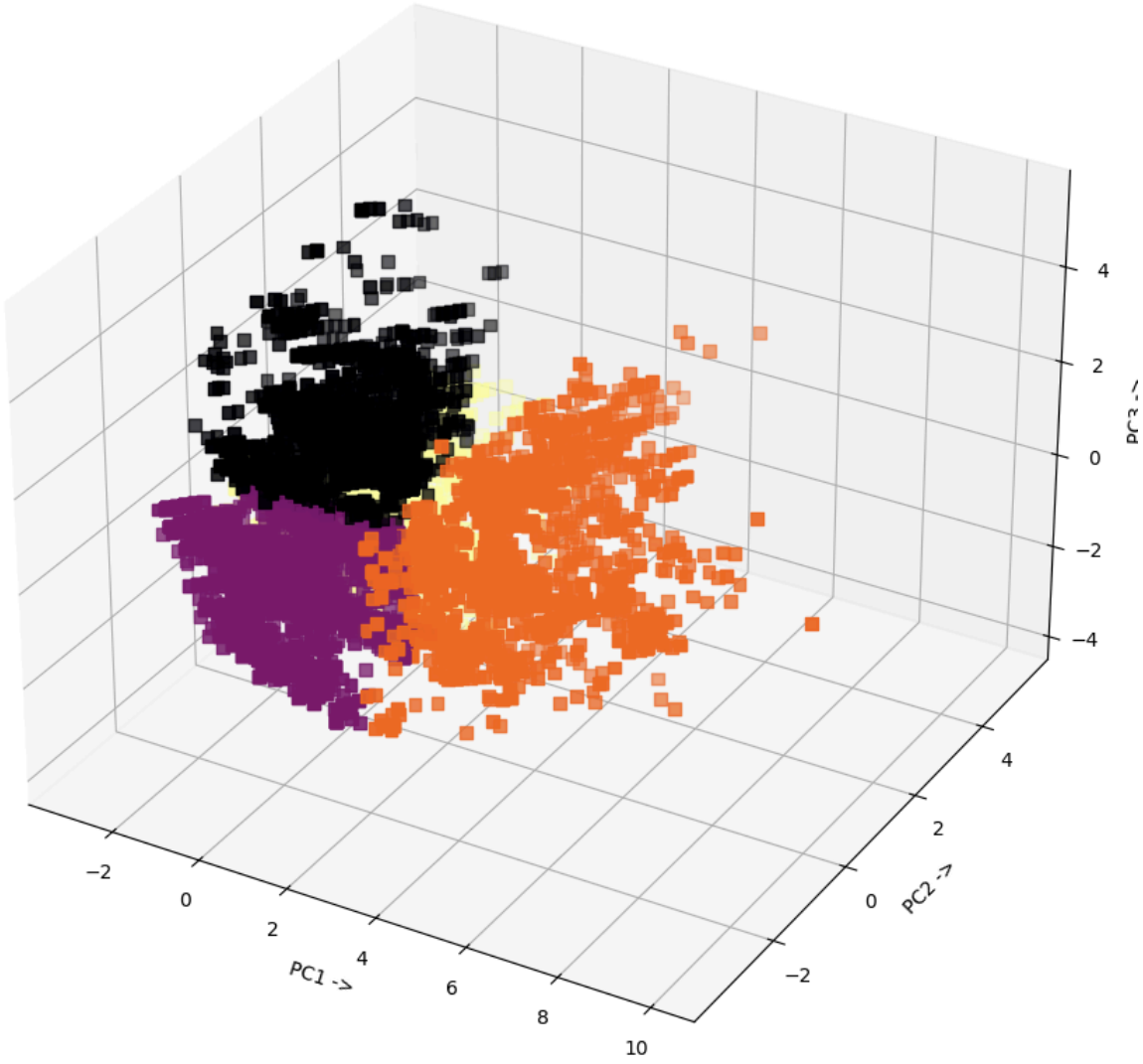
K-Means, 4 clusters

In [ ]:
```
# Apply K-Means clustering
num_clusters = 4
kmeans = KMeans(n_clusters=num_clusters, random_state=42)
data_frame_shuffled['cluster'] = kmeans.fit_predict(data_frame_pca3)

data_frame_shuffled_original['cluster'] = data_frame_shuffled['cluster']

df_kmeans_4clusters = pd.DataFrame(data_frame_shuffled)

data_frame_shuffled_original.to_csv('result_kmeans_4clusters.csv')

# Compute mean values of features within each cluster
cluster_means = data_frame_shuffled.groupby('cluster').mean()
```

```
# Calculate feature importance by comparing mean values across clusters
feature_importance = cluster_means.sub(cluster_means.mean()).abs().sum() / cluster_

# Sort features by importance
feature_importance_sorted = feature_importance.sort_values(ascending=False)

# Plot feature importance
plt.figure(figsize=(7, 6))
plt.bar(feature_importance_sorted.index, feature_importance_sorted)
plt.xlabel('Features')
plt.ylabel('Importance')
plt.title('Feature Importance')
plt.xticks(rotation=45, ha='right')
plt.show()


fig = plt.figure(figsize = (11,11))
ax = fig.add_subplot(111, projection='3d')
x = np.array(data_frame_pca3['PC1'])
y = np.array(data_frame_pca3['PC2'])
z = np.array(data_frame_pca3['PC3'])

ax.scatter(x,y,z, marker="s", c=data_frame_shuffled["cluster"], s=40, cmap="inferno
ax.set_xlabel('PC1 ->')
ax.set_ylabel('PC2 ->')
ax.set_zlabel('PC3 ->')

plt.show()


# 3d scatterplot using plotly
Scene = dict(xaxis = dict(title  = 'PC1'),yaxis = dict(title  = 'PC2'),zaxis = dict
labels = kmeans.labels_
trace = go.Scatter3d(x=data_frame_pca3['PC1'], y=data_frame_pca3['PC2'], z=data_fra
layout = go.Layout(margin=dict(l=0,r=0),scene = Scene,height = 800,width = 800)
data = [trace]
fig = go.Figure(data = data, layout = layout)
fig.show()


# Calculate clustering metrics
silhouette_kmeans_4 = silhouette_score(data_frame_pca3, kmeans.labels_)
db_index_kmeans_4 = davies_bouldin_score(data_frame_pca3, kmeans.labels_)
ch_index_kmeans_4 = calinski_harabasz_score(data_frame_pca3, kmeans.labels_)


# Print the metric scores
print(f"Silhouette Score: {silhouette_kmeans_4:.2f}")
print(f"Davies-Bouldin Index: {db_index_kmeans_4:.2f}")
print(f"Calinski-Harabasz Index: {ch_index_kmeans_4:.2f}")
```

```
C:\Users\Nemanja\AppData\Roaming\Python\Python311\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 warning
```



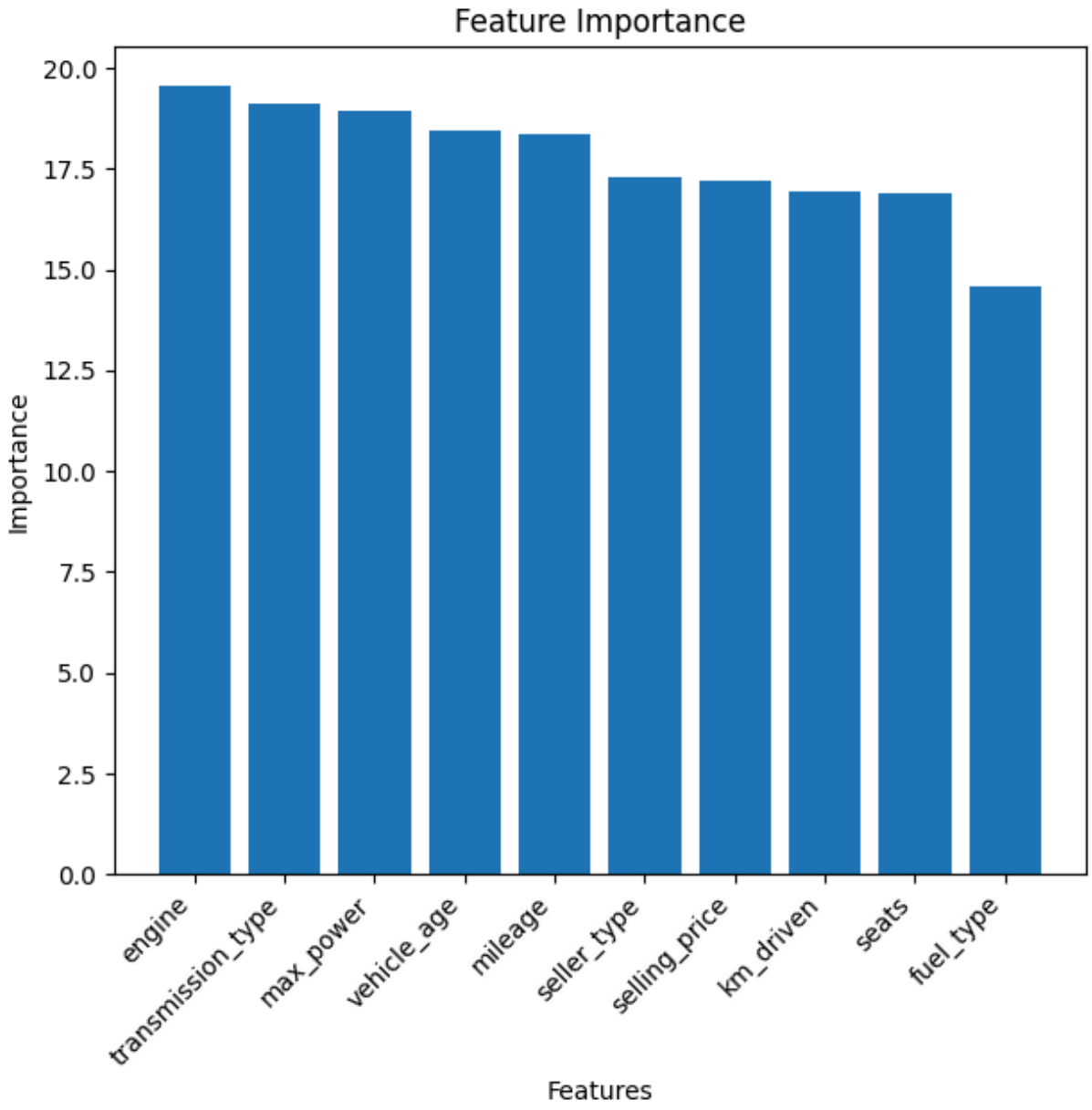


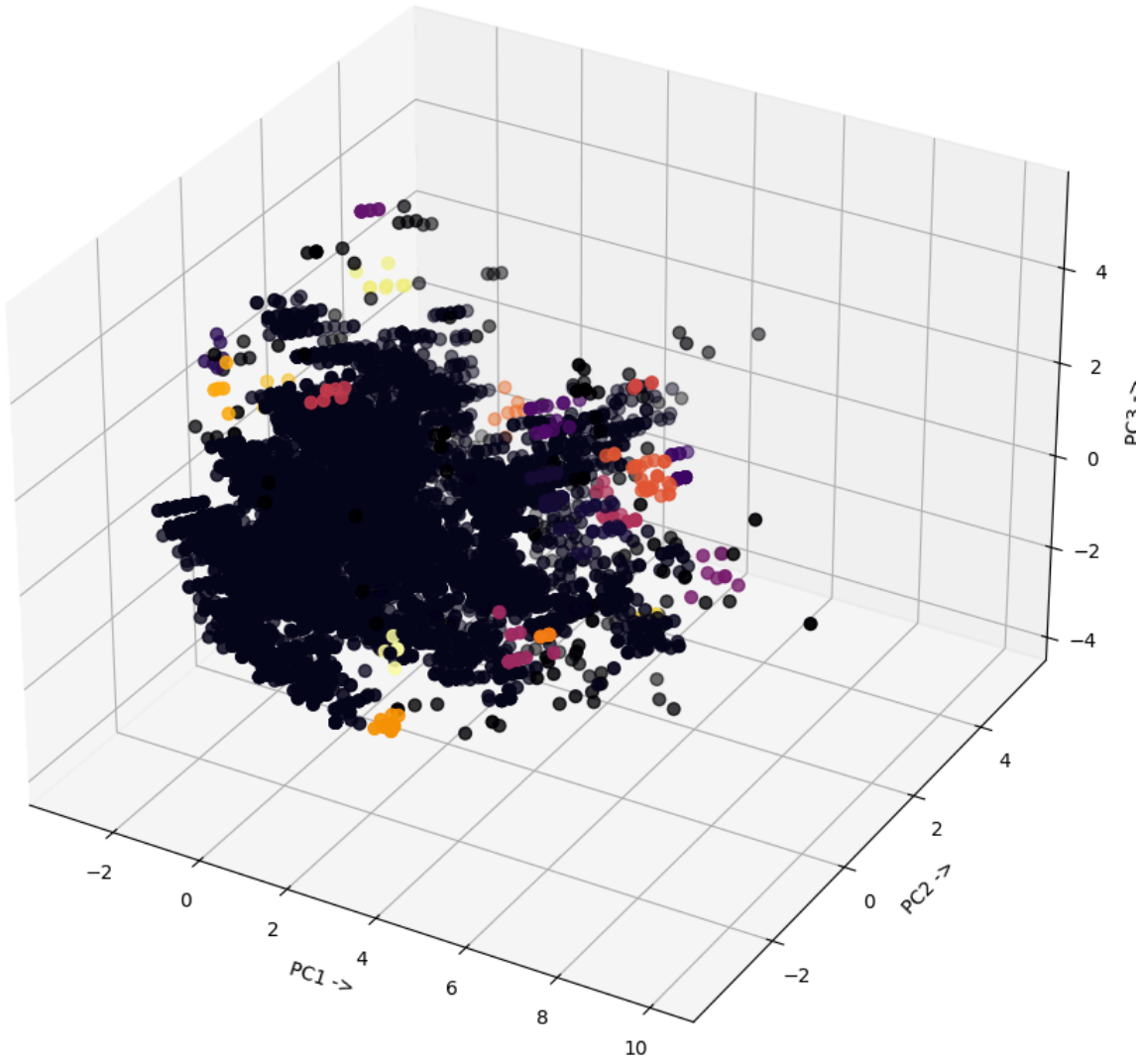

```
Silhouette Score: 0.35
Davies-Bouldin Index: 0.96
Calinski-Harabasz Index: 8276.70
```

DBSCAN

In [ ]:
```
# Apply DBSCAN clustering
dbscan = DBSCAN(eps=0.5, min_samples=5)
data_frame_shuffled['cluster'] = dbscan.fit_predict(data_frame_pca3)

# Compute mean values of features within each cluster
cluster_means = data_frame_shuffled.groupby('cluster').mean()

# Calculate feature importance by comparing mean values across clusters
feature_importance = cluster_means.sub(cluster_means.mean()).abs().sum() / cluster_

# Sort features by importance
feature_importance_sorted = feature_importance.sort_values(ascending=False)
```

```
# Plot feature importance
plt.figure(figsize=(7, 6))
plt.bar(feature_importance_sorted.index, feature_importance_sorted)
plt.xlabel('Features')
plt.ylabel('Importance')
plt.title('Feature Importance')
plt.xticks(rotation=45, ha='right')
plt.show()

data_frame_shuffled_original['cluster'] = data_frame_shuffled['cluster']
data_frame_shuffled_original.to_csv('result_dbscan.csv')

# Plot the clusters
fig = plt.figure(figsize=(11, 11))
ax = fig.add_subplot(111, projection='3d')
x = np.array(data_frame_pca3['PC1'])
y = np.array(data_frame_pca3['PC2'])
z = np.array(data_frame_pca3['PC3'])
ax.scatter(x, y, z, c=data_frame_shuffled['cluster'], cmap='inferno', s=40, label=T
ax.set_xlabel('PC1 ->')
ax.set_ylabel('PC2 ->')
ax.set_zlabel('PC3 ->')
plt.show()

# Plot 3D scatterplot using Plotly
scene = dict(xaxis=dict(title='PC1'), yaxis=dict(title='PC2'), zaxis=dict(title='PC
labels = data_frame_shuffled['cluster']
trace = go.Scatter3d(x=data_frame_pca3['PC1'], y=data_frame_pca3['PC2'], z=data_fra
                     mode='markers', marker=dict(color=labels, size=10, line=dict(c
layout = go.Layout(margin=dict(l=0, r=0), scene=scene, height=800, width=800)
data = [trace]
fig = go.Figure(data=data, layout=layout)
fig.show()

# Calculate clustering metrics
silhouette_dbscan = silhouette_score(data_frame_pca3[['PC1', 'PC2', 'PC3']], data_f
db_index_dbscan = davies_bouldin_score(data_frame_pca3[['PC1', 'PC2', 'PC3']], data
ch_index_dbscan = calinski_harabasz_score(data_frame_pca3[['PC1', 'PC2', 'PC3']], d

# Print the metric scores
print(f"Silhouette Score: {silhouette_dbscan:.2f}")
print(f"Davies-Bouldin Index: {db_index_dbscan:.2f}")
print(f"Calinski-Harabasz Index: {ch_index_dbscan:.2f}")
```


Feature Importance
Importance
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
engine
transmission_type
max_power
vehicle_age
mileage
seller_type
selling_price
km_driven
seats
fuel_type
Features



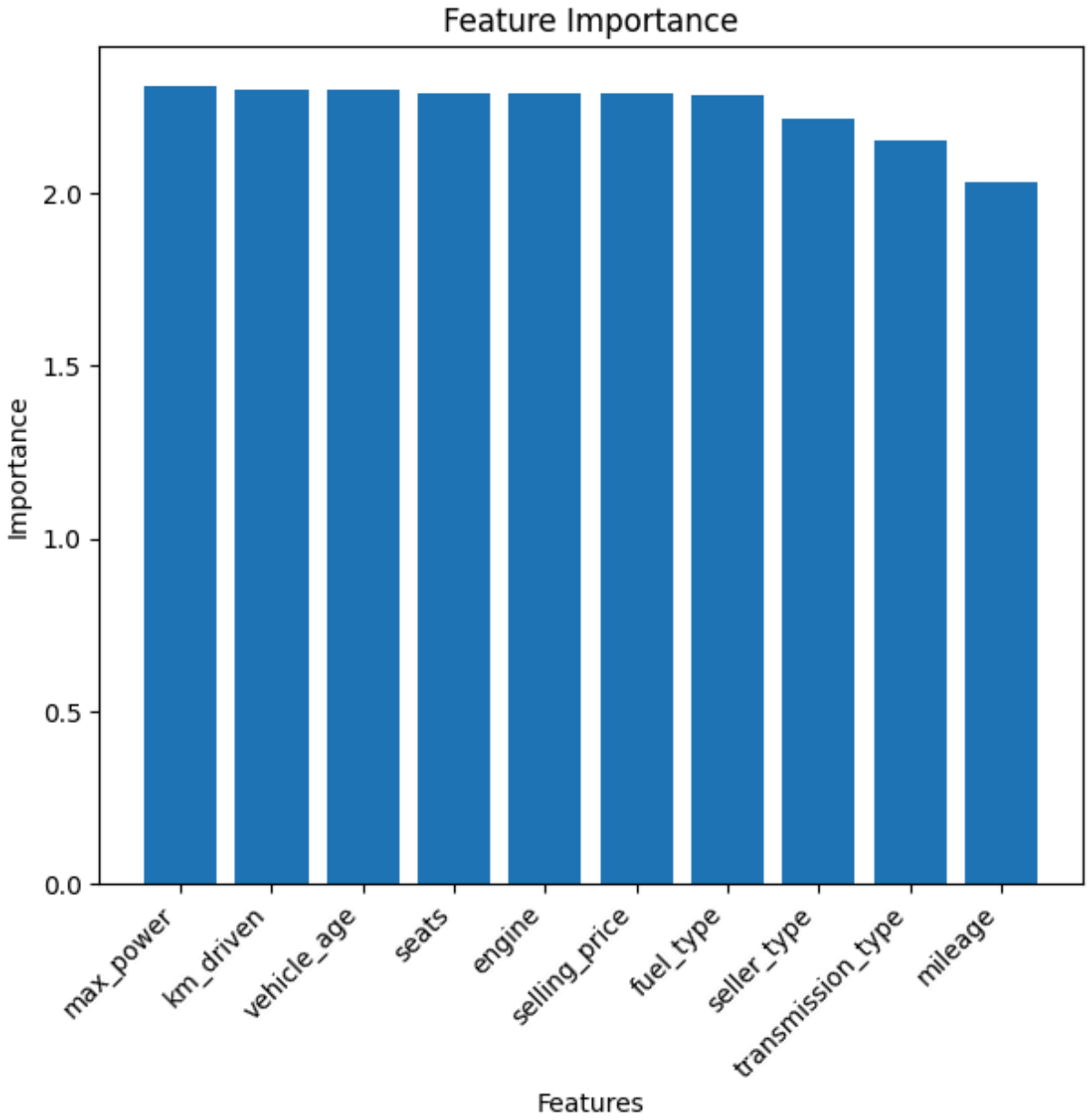

```
Silhouette Score: -0.23
Davies-Bouldin Index: 1.17
Calinski-Harabasz Index: 63.70
```

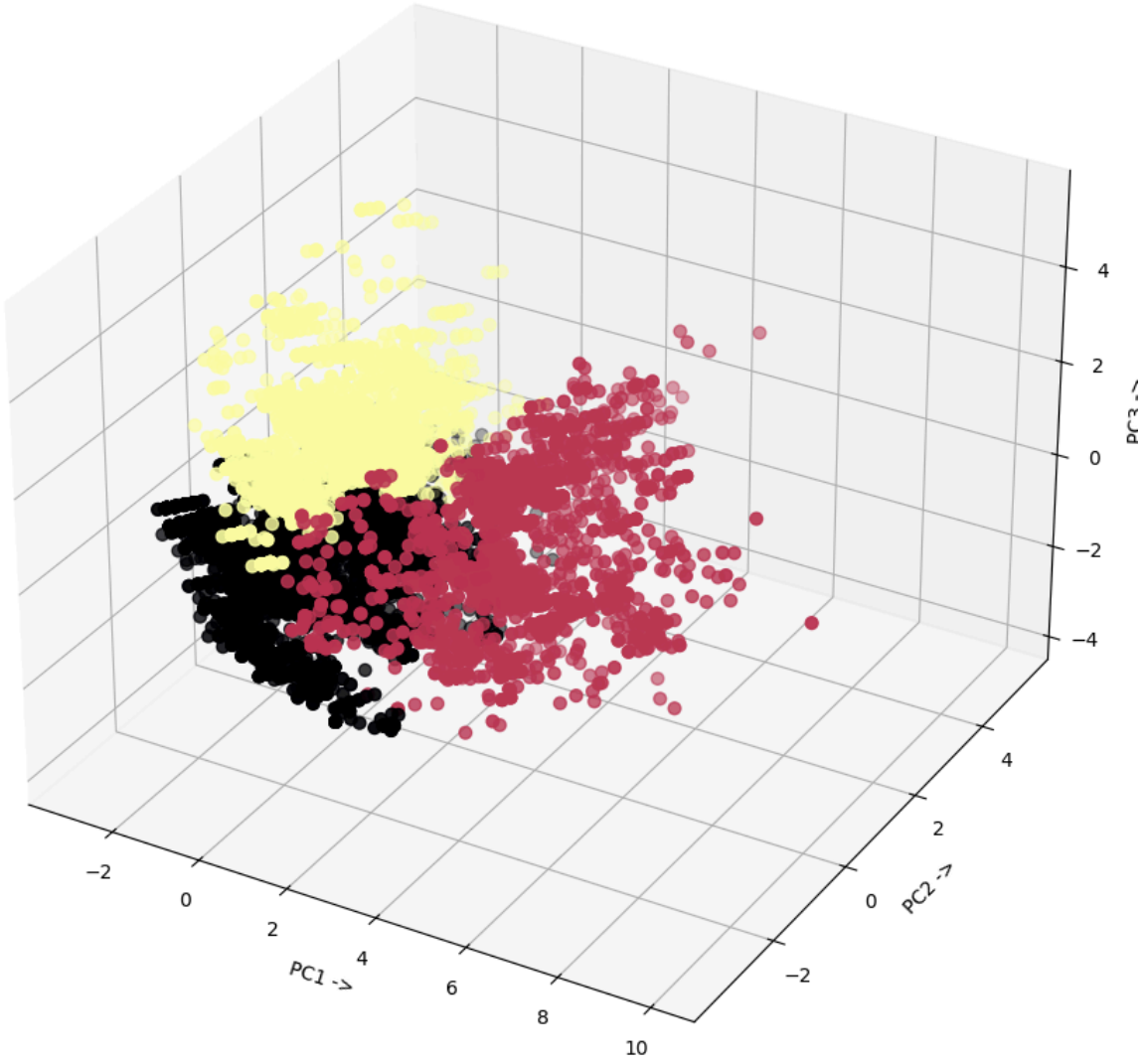
Agglomerative

Agglomerative, 3 clusters

In [ ]:
```
# Apply Agglomerative clustering
agg_cluster = AgglomerativeClustering(n_clusters=3)
data_frame_shuffled['cluster'] = agg_cluster.fit_predict(data_frame_pca3)

# Compute mean values of features within each cluster
cluster_means = data_frame_shuffled.groupby('cluster').mean()

# Calculate feature importance by comparing mean values across clusters
feature_importance = cluster_means.sub(cluster_means.mean()).abs().sum() / cluster_
```

```
# Sort features by importance
feature_importance_sorted = feature_importance.sort_values(ascending=False)

# Plot feature importance
plt.figure(figsize=(7, 6))
plt.bar(feature_importance_sorted.index, feature_importance_sorted)
plt.xlabel('Features')
plt.ylabel('Importance')
plt.title('Feature Importance')
plt.xticks(rotation=45, ha='right')
plt.show()

data_frame_shuffled_original['cluster'] = data_frame_shuffled['cluster']
data_frame_shuffled_original.to_csv('result_agg_3.csv')

# Plot the clusters
fig = plt.figure(figsize=(11, 11))
ax = fig.add_subplot(111, projection='3d')
x = np.array(data_frame_pca3['PC1'])
y = np.array(data_frame_pca3['PC2'])
z = np.array(data_frame_pca3['PC3'])
ax.scatter(x, y, z, c=data_frame_shuffled['cluster'], cmap='inferno', s=40, label=T
ax.set_xlabel('PC1 ->')
ax.set_ylabel('PC2 ->')
ax.set_zlabel('PC3 ->')
plt.show()

# Plot 3D scatterplot using Plotly
scene = dict(xaxis=dict(title='PC1'), yaxis=dict(title='PC2'), zaxis=dict(title='PC
labels = data_frame_shuffled_original['cluster']
trace = go.Scatter3d(x=data_frame_pca3['PC1'], y=data_frame_pca3['PC2'], z=data_fra
                     mode='markers', marker=dict(color=labels, size=10, line=dict(c
layout = go.Layout(margin=dict(l=0, r=0), scene=scene, height=800, width=800)
data = [trace]
fig = go.Figure(data=data, layout=layout)
fig.show()

# Calculate clustering metrics
silhouette_agg_3 = silhouette_score(data_frame_pca3[['PC1', 'PC2', 'PC3']], data_fr
db_index_agg_3 = davies_bouldin_score(data_frame_pca3[['PC1', 'PC2', 'PC3']], data_
ch_index_agg_3 = calinski_harabasz_score(data_frame_pca3[['PC1', 'PC2', 'PC3']], da

# Print the metric scores
print(f"Silhouette Score: {silhouette_agg_3:.2f}")
print(f"Davies-Bouldin Index: {db_index_agg_3:.2f}")
print(f"Calinski-Harabasz Index: {ch_index_agg_3:.2f}")
```


Feature Importance
Importance
0.0
0.5
1.0
1.5
2.0
max_power
km_driven
vehicle_age
seats
engine
selling_price
fuel_type
seller_type
transmission_type
mileage
Features

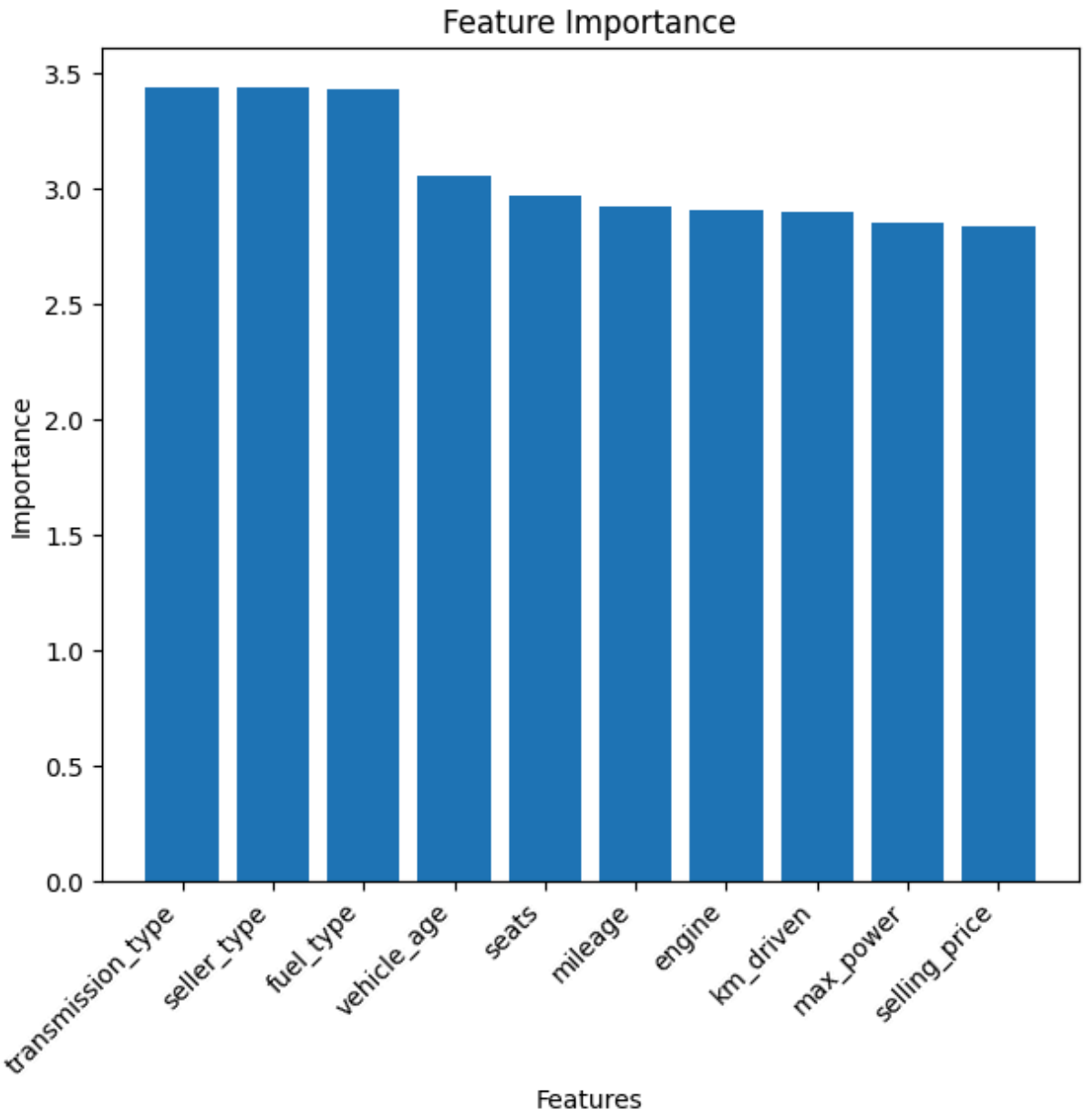


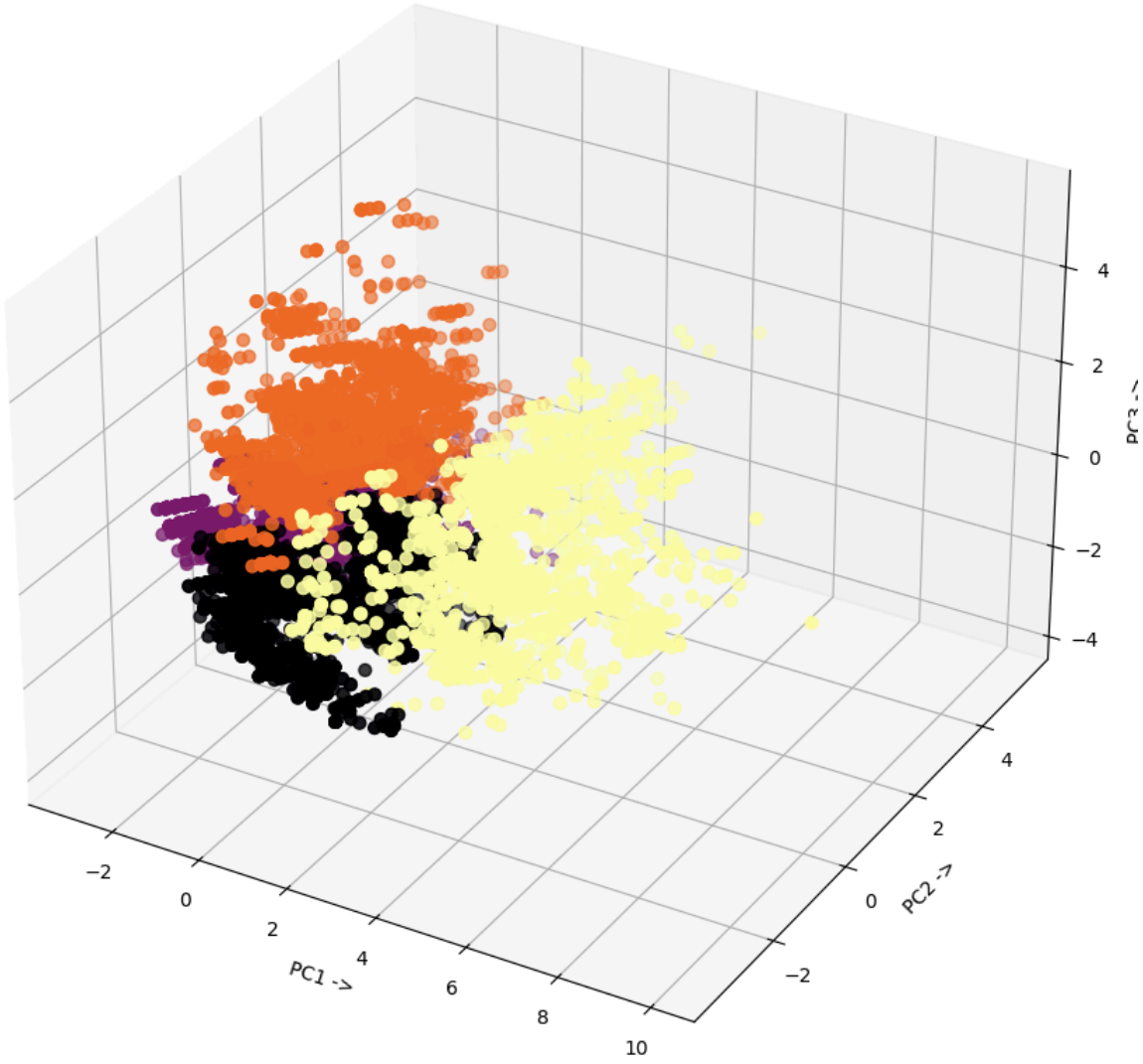

```
Silhouette Score: 0.30
Davies-Bouldin Index: 1.21
Calinski-Harabasz Index: 7234.80
```

Agglomerative, 4 clusters

In [ ]:
```
# Apply Agglomerative clustering
agg_cluster = AgglomerativeClustering(n_clusters=4)
data_frame_shuffled['cluster'] = agg_cluster.fit_predict(data_frame_pca3)

# Compute mean values of features within each cluster
cluster_means = data_frame_shuffled.groupby('cluster').mean()

# Calculate feature importance by comparing mean values across clusters
feature_importance = cluster_means.sub(cluster_means.mean()).abs().sum() / cluster_

# Sort features by importance
feature_importance_sorted = feature_importance.sort_values(ascending=False)
```

```
# Plot feature importance
plt.figure(figsize=(7, 6))
plt.bar(feature_importance_sorted.index, feature_importance_sorted)
plt.xlabel('Features')
plt.ylabel('Importance')
plt.title('Feature Importance')
plt.xticks(rotation=45, ha='right')
plt.show()

data_frame_shuffled_original['cluster'] = data_frame_shuffled['cluster']
data_frame_shuffled_original.to_csv('result_agg_4.csv')

# Plot the clusters
fig = plt.figure(figsize=(11, 11))
ax = fig.add_subplot(111, projection='3d')
x = np.array(data_frame_pca3['PC1'])
y = np.array(data_frame_pca3['PC2'])
z = np.array(data_frame_pca3['PC3'])
ax.scatter(x, y, z, c=data_frame_shuffled['cluster'], cmap='inferno', s=40, label=T
ax.set_xlabel('PC1 ->')
ax.set_ylabel('PC2 ->')
ax.set_zlabel('PC3 ->')
plt.show()

# Plot 3D scatterplot using Plotly
scene = dict(xaxis=dict(title='PC1'), yaxis=dict(title='PC2'), zaxis=dict(title='PC
labels = data_frame_shuffled_original['cluster']
trace = go.Scatter3d(x=data_frame_pca3['PC1'], y=data_frame_pca3['PC2'], z=data_fra
                     mode='markers', marker=dict(color=labels, size=10, line=dict(c
layout = go.Layout(margin=dict(l=0, r=0), scene=scene, height=800, width=800)
data = [trace]
fig = go.Figure(data=data, layout=layout)
fig.show()

# Calculate clustering metrics
silhouette_agg_4 = silhouette_score(data_frame_pca3[['PC1', 'PC2', 'PC3']], data_fr
db_index_agg_4 = davies_bouldin_score(data_frame_pca3[['PC1', 'PC2', 'PC3']], data_
ch_index_agg_4 = calinski_harabasz_score(data_frame_pca3[['PC1', 'PC2', 'PC3']], da

# Print the metric scores
print(f"Silhouette Score: {silhouette_agg_4:.2f}")
print(f"Davies-Bouldin Index: {db_index_agg_4:.2f}")
print(f"Calinski-Harabasz Index: {ch_index_agg_4:.2f}")
```


Feature Importance
Importance
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
transmission_type
seller_type
fuel_type
vehicle_age
seats
mileage
engine
km_driven
max_power
selling_price
Features

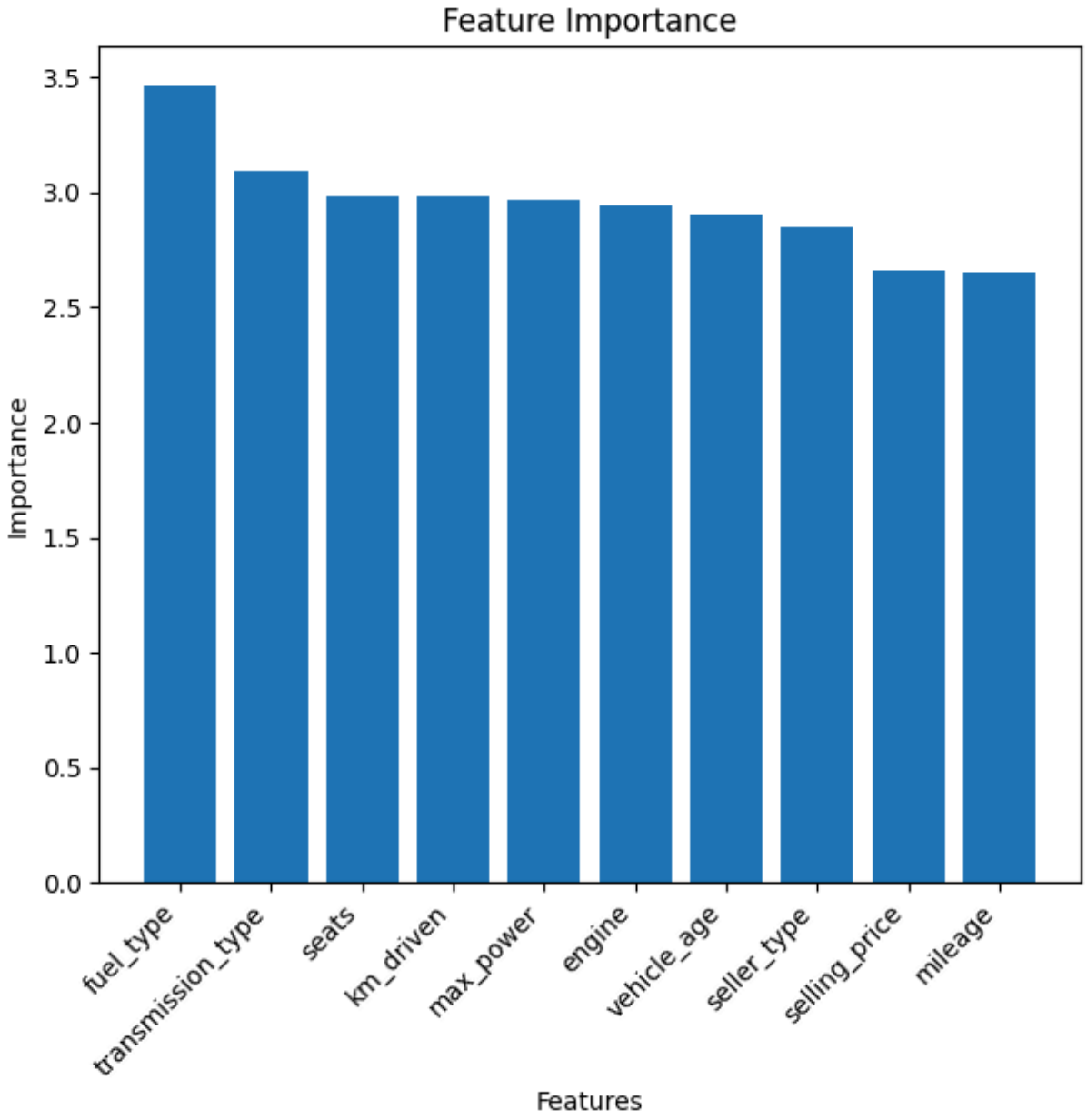


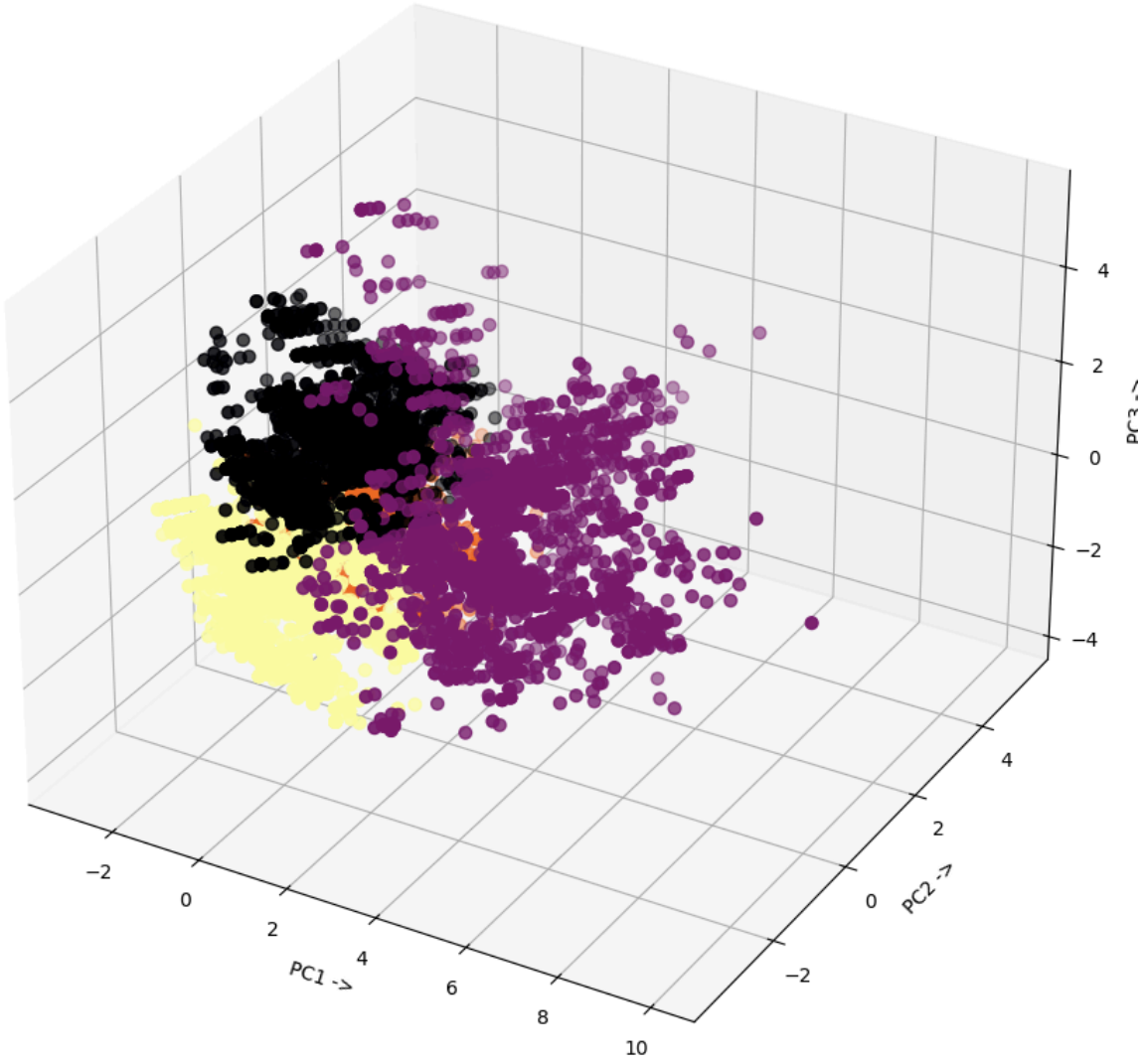

```
Silhouette Score: 0.30
Davies-Bouldin Index: 1.20
Calinski-Harabasz Index: 6783.61
```

GMM

In [ ]:
```
# Apply Gaussian Mixture Model clustering

gmm = GaussianMixture(n_components=4, random_state=42)
data_frame_shuffled['cluster'] = gmm.fit_predict(data_frame_pca3)

# Compute mean values of features within each cluster
cluster_means = data_frame_shuffled.groupby('cluster').mean()

# Calculate feature importance by comparing mean values across clusters
feature_importance = cluster_means.sub(cluster_means.mean()).abs().sum() / cluster_

# Sort features by importance
```

```
feature_importance_sorted = feature_importance.sort_values(ascending=False)

# Plot feature importance
plt.figure(figsize=(7, 6))
plt.bar(feature_importance_sorted.index, feature_importance_sorted)
plt.xlabel('Features')
plt.ylabel('Importance')
plt.title('Feature Importance')
plt.xticks(rotation=45, ha='right')
plt.show()

data_frame_shuffled_original['cluster'] = data_frame_shuffled['cluster']
data_frame_shuffled_original.to_csv('result_gmm.csv')

# Plot the clusters
fig = plt.figure(figsize=(11, 11))
ax = fig.add_subplot(111, projection='3d')
x = np.array(data_frame_pca3['PC1'])
y = np.array(data_frame_pca3['PC2'])
z = np.array(data_frame_pca3['PC3'])
ax.scatter(x, y, z, c=data_frame_shuffled['cluster'], cmap='inferno', s=40, label=T
ax.set_xlabel('PC1 ->')
ax.set_ylabel('PC2 ->')
ax.set_zlabel('PC3 ->')
plt.show()

# Plot 3D scatterplot using Plotly
scene = dict(xaxis=dict(title='PC1'), yaxis=dict(title='PC2'), zaxis=dict(title='PC
labels = data_frame_shuffled['cluster']
trace = go.Scatter3d(x=data_frame_pca3['PC1'], y=data_frame_pca3['PC2'], z=data_fra
                     mode='markers', marker=dict(color=labels, size=10, line=dict(c
layout = go.Layout(margin=dict(l=0, r=0), scene=scene, height=800, width=800)
data = [trace]
fig = go.Figure(data=data, layout=layout)
fig.show()

# Calculate clustering metrics
silhouette_gmm = silhouette_score(data_frame_pca3[['PC1', 'PC2', 'PC3']], data_fram
db_index_gmm = davies_bouldin_score(data_frame_pca3[['PC1', 'PC2', 'PC3']], data_fr
ch_index_gmm = calinski_harabasz_score(data_frame_pca3[['PC1', 'PC2', 'PC3']], data

# Print the metric scores
print(f"Silhouette Score: {silhouette_gmm:.2f}")
print(f"Davies-Bouldin Index: {db_index_gmm:.2f}")
print(f"Calinski-Harabasz Index: {ch_index_gmm:.2f}")
```


Feature Importance
Importance
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
fuel_type
transmission_type
seats
km_driven
max_power
engine
vehicle_age
seller_type
selling_price
mileage
Features

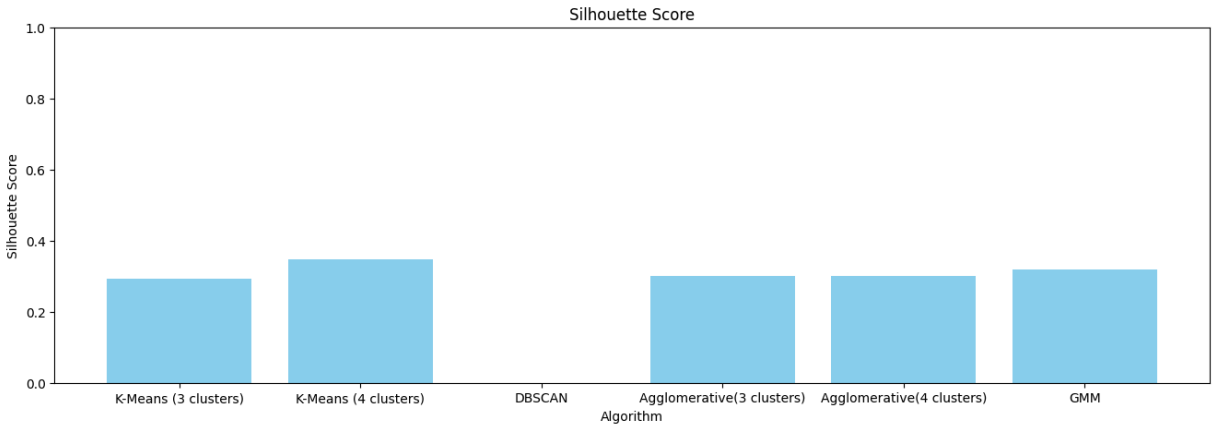


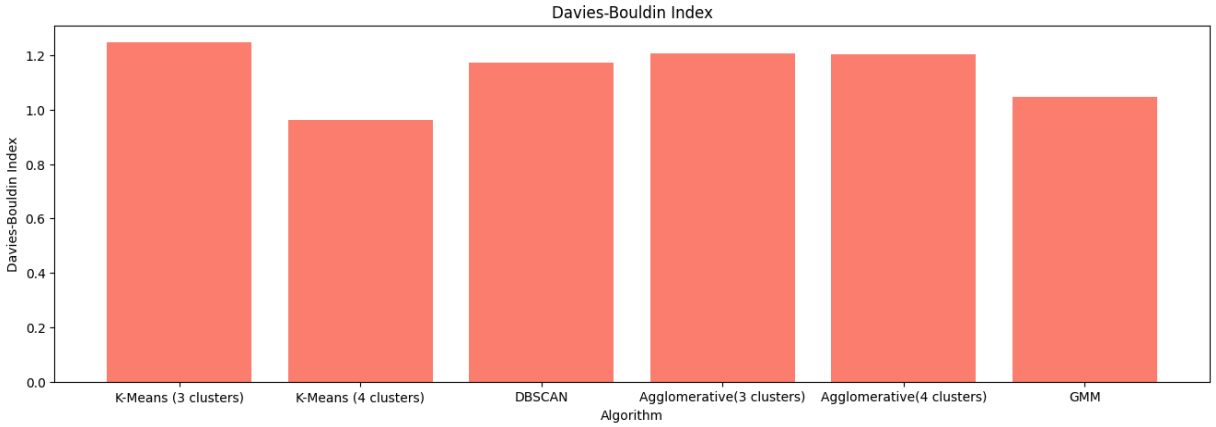

```
Silhouette Score: 0.32
Davies-Bouldin Index: 1.05
Calinski-Harabasz Index: 6805.18
```

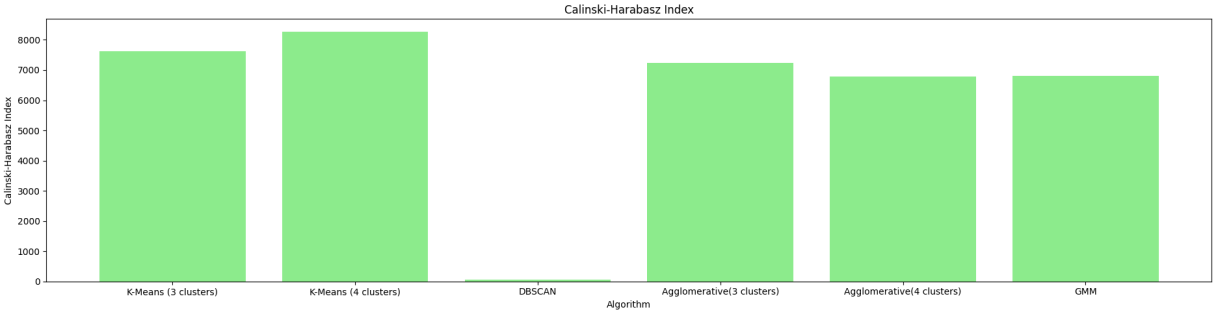
Comparison

In [ ]:
```
# Metrics scores
silhouette_scores = [silhouette_kmeans_3, silhouette_kmeans_4, silhouette_dbscan, s
davies_bouldin_scores = [db_index_kmeans_3, db_index_kmeans_4, db_index_dbscan, db_
calinski_harabasz_scores = [ch_index_kmeans_3, ch_index_kmeans_4, ch_index_dbscan,

# Algorithms names
algorithms = ['K-Means (3 clusters)',
              'K-Means (4 clusters)',
              'DBSCAN',
              'Agglomerative(3 clusters)',
              'Agglomerative(4 clusters)',
              'GMM']
```

```
# Plot silhouette score
plt.figure(figsize=(55, 5))
plt.subplot(1, 3, 1)
plt.bar(algorithms, silhouette_scores, color='skyblue')
plt.title('Silhouette Score')
plt.xlabel('Algorithm')
plt.ylabel('Silhouette Score')
plt.ylim(0, 1)

# Plot Davies-Bouldin index
plt.figure(figsize=(55, 5))
plt.subplot(1, 3, 1)
plt.bar(algorithms, davies_bouldin_scores, color='salmon')
plt.title('Davies-Bouldin Index')
plt.xlabel('Algorithm')
plt.ylabel('Davies-Bouldin Index')

# Plot Calinski-Harabasz index
plt.figure(figsize=(55, 5))
plt.subplot(1, 3, 1)
plt.bar(algorithms, calinski_harabasz_scores, color='lightgreen')
plt.title('Calinski-Harabasz Index')
plt.xlabel('Algorithm')
plt.ylabel('Calinski-Harabasz Index')


plt.tight_layout()
plt.show()
```





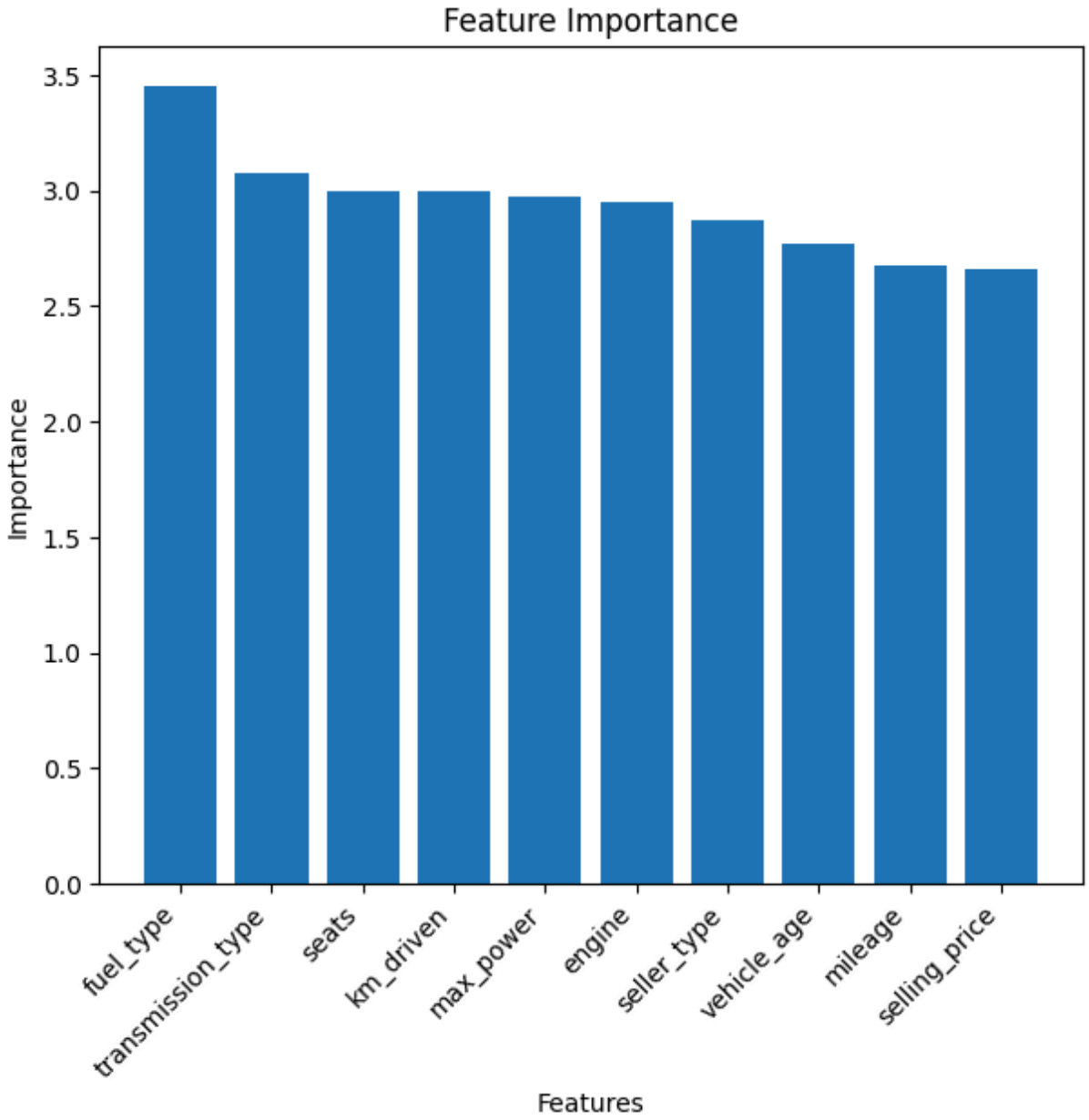


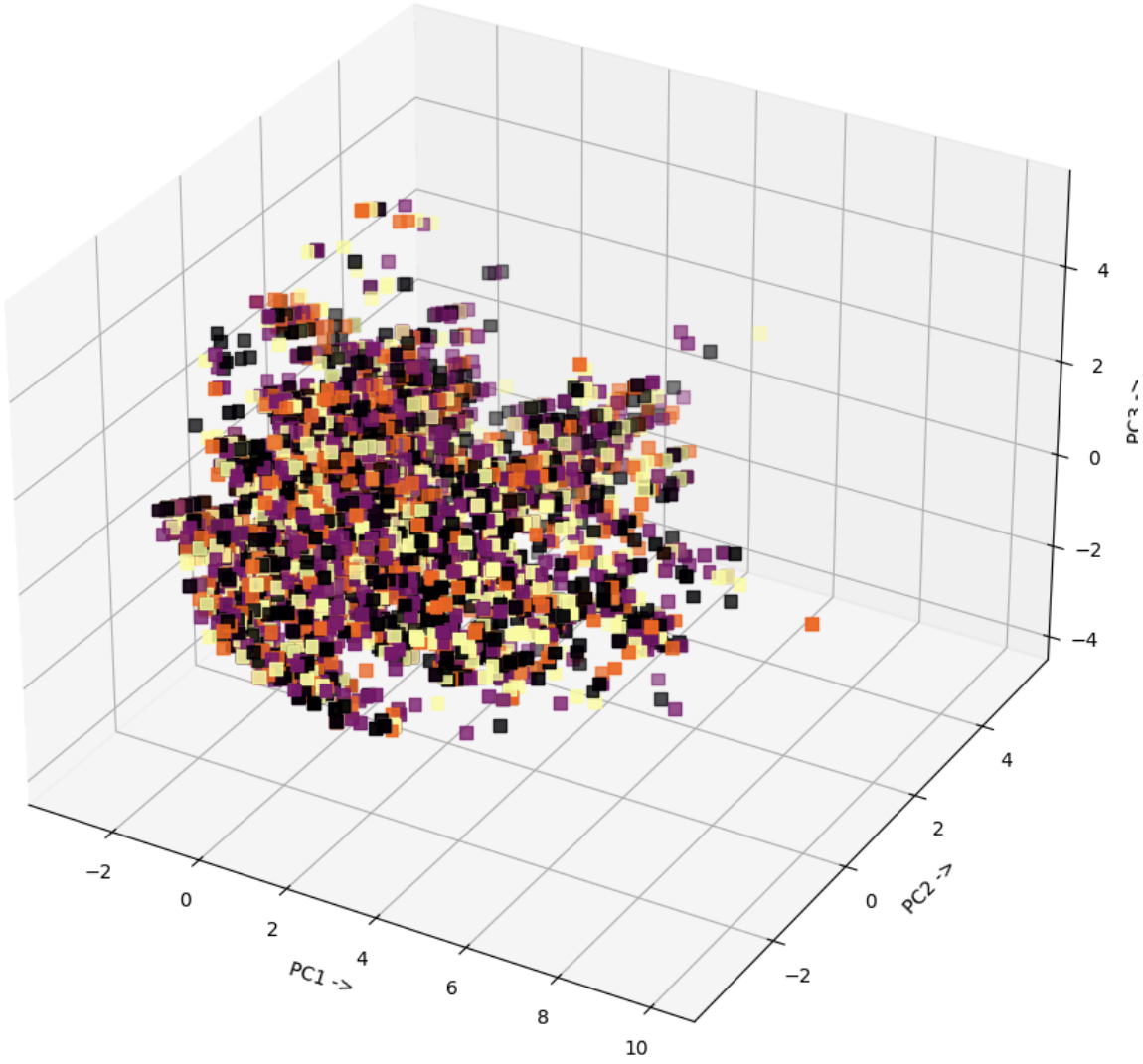
K-Means, 4 cluster, no PCA

In [ ]:
```
# Apply K-Means clustering
num_clusters = 4
kmeans = KMeans(n_clusters=num_clusters, random_state=42)
data_frame_shuffled['cluster'] = kmeans.fit_predict(data_frame_shuffled)

data_frame_shuffled_original['cluster'] = data_frame_shuffled['cluster']

data_frame_shuffled_original.to_csv('result_kmeans_4clusters_noPCA.csv')

# Compute mean values of features within each cluster
cluster_means = data_frame_shuffled.groupby('cluster').mean()

# Calculate feature importance by comparing mean values across clusters
feature_importance = cluster_means.sub(cluster_means.mean()).abs().sum() / cluster_

# Sort features by importance
feature_importance_sorted = feature_importance.sort_values(ascending=False)

# Plot feature importance
plt.figure(figsize=(7, 6))
plt.bar(feature_importance_sorted.index, feature_importance_sorted)
plt.xlabel('Features')
plt.ylabel('Importance')
plt.title('Feature Importance')
plt.xticks(rotation=45, ha='right')
plt.show()

data_frame_pca3['cluster'] = data_frame_shuffled['cluster']
```

```
fig = plt.figure(figsize = (11,11))
ax = fig.add_subplot(111, projection='3d')
x = np.array(data_frame_pca3['PC1'])
y = np.array(data_frame_pca3['PC2'])
z = np.array(data_frame_pca3['PC3'])

ax.scatter(x,y,z, marker="s", c=data_frame_pca3["cluster"], s=40, cmap="inferno", l
ax.set_xlabel('PC1 ->')
ax.set_ylabel('PC2 ->')
ax.set_zlabel('PC3 ->')

plt.show()


# 3d scatterplot using plotly
Scene = dict(xaxis = dict(title  = 'PC1'),yaxis = dict(title  = 'PC2'),zaxis = dict
labels = kmeans.labels_
trace = go.Scatter3d(x=data_frame_pca3['PC1'], y=data_frame_pca3['PC2'], z=data_fra
layout = go.Layout(margin=dict(l=0,r=0),scene = Scene,height = 800,width = 800)
data = [trace]
fig = go.Figure(data = data, layout = layout)
fig.show()


# Calculate clustering metrics
silhouette_kmeans_4_noPCA = silhouette_score(data_frame_shuffled, kmeans.labels_)
db_index_kmeans_4_noPCA = davies_bouldin_score(data_frame_shuffled, kmeans.labels_)
ch_index_kmeans_4_noPCA = calinski_harabasz_score(data_frame_shuffled, kmeans.label


# Print the metric scores
print(f"Silhouette Score: {silhouette_kmeans_4_noPCA:.2f}")
print(f"Davies-Bouldin Index: {db_index_kmeans_4_noPCA:.2f}")
print(f"Calinski-Harabasz Index: {ch_index_kmeans_4_noPCA:.2f}")
```

```
C:\Users\Nemanja\AppData\Roaming\Python\Python311\site-packages\sklearn\cluster\_kme
ans.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 warning
```

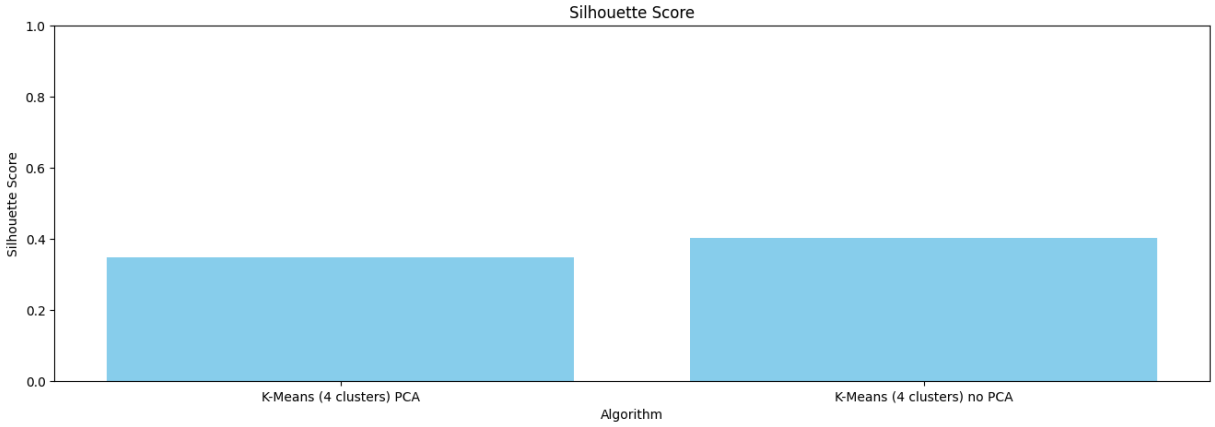

Feature Importance
Importance
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
fuel_type
transmission_type
seats
km_driven
max_power
engine
seller_type
vehicle_age
mileage
selling_price
Features

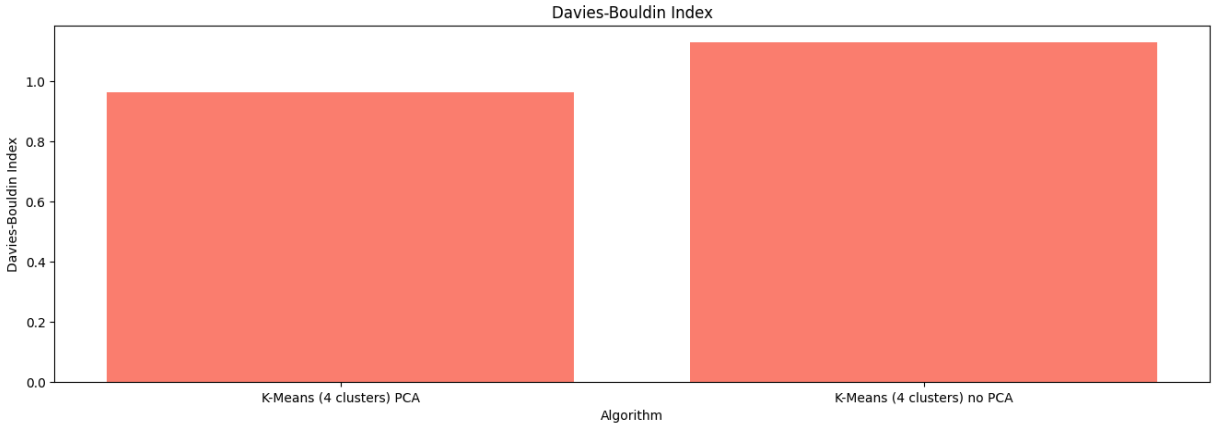


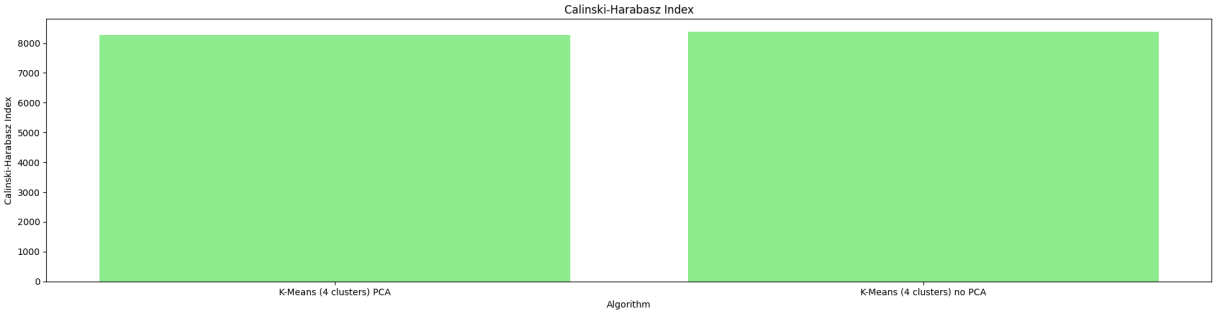

```
Silhouette Score: 0.40
Davies-Bouldin Index: 1.13
Calinski-Harabasz Index: 8391.73
```

In [ ]:
```python
# Metrics scores
silhouette_scores = [silhouette_kmeans_4, silhouette_kmeans_4_noPCA]
davies_bouldin_scores = [db_index_kmeans_4, db_index_kmeans_4_noPCA]
calinski_harabasz_scores = [ch_index_kmeans_4, ch_index_kmeans_4_noPCA]

# Algorithms names
algorithms = ['K-Means (4 clusters) PCA',
              'K-Means (4 clusters) no PCA']


# Plot silhouette score
plt.figure(figsize=(55, 5))
plt.subplot(1, 3, 1)
plt.bar(algorithms, silhouette_scores, color='skyblue')
plt.title('Silhouette Score')
plt.xlabel('Algorithm')
```

```
plt.ylabel('Silhouette Score')
plt.ylim(0, 1)

# Plot Davies-Bouldin index
plt.figure(figsize=(55, 5))
plt.subplot(1, 3, 1)
plt.bar(algorithms, davies_bouldin_scores, color='salmon')
plt.title('Davies-Bouldin Index')
plt.xlabel('Algorithm')
plt.ylabel('Davies-Bouldin Index')

# Plot Calinski-Harabasz index
plt.figure(figsize=(55, 5))
plt.subplot(1, 3, 1)
plt.bar(algorithms, calinski_harabasz_scores, color='lightgreen')
plt.title('Calinski-Harabasz Index')
plt.xlabel('Algorithm')
plt.ylabel('Calinski-Harabasz Index')


plt.tight_layout()
plt.show()

# Print the metric scores
print(f"Silhouette Score: {silhouette_kmeans_4:.2f}")
print(f"Davies-Bouldin Index: {db_index_kmeans_4:.2f}")
print(f"Calinski-Harabasz Index: {ch_index_kmeans_4:.2f}")

# Print the metric scores
print(f"Silhouette Score (no PCA): {silhouette_kmeans_4_noPCA:.2f}")
print(f"Davies-Bouldin Index (no PCA): {db_index_kmeans_4_noPCA:.2f}")
print(f"Calinski-Harabasz Index (no PCA): {ch_index_kmeans_4_noPCA:.2f}")
```








```
Silhouette Score: 0.35
Davies-Bouldin Index: 0.96
Calinski-Harabasz Index: 8276.70
Silhouette Score (no PCA): 0.40
Davies-Bouldin Index (no PCA): 1.13
Calinski-Harabasz Index (no PCA): 8391.73
```

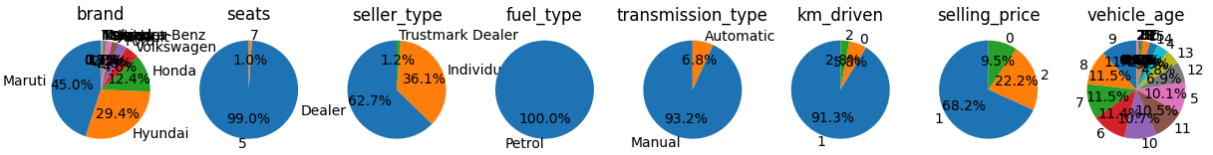
In [ ]:
```
categorical_features = ['brand', 'seats', 'seller_type', 'fuel_type', 'transmission

results = {}

data_frame_shuffled_original_modified =  pd.read_csv('result_kmeans_4clusters_noPCA


# Discretization of selling_price feature
data_frame_shuffled_original_modified['selling_price'] = np.digitize(data_frame_shu

# Discretization of km_driven feature
data_frame_shuffled_original_modified['km_driven'] = np.digitize(data_frame_shuffle


for cluster in data_frame_shuffled_original_modified['cluster'].unique():
    cluster_results = {}

    for feature in categorical_features:
        feature_counts = data_frame_shuffled_original_modified[data_frame_shuffled_

        cluster_results[feature] = feature_counts.to_dict()

    results[f'cluster {cluster}'] = cluster_results

df_results = pd.DataFrame(results)
```

```
pd.set_option('display.max_colwidth', None)
display(df_results.T)

columns = results
for cluster in columns:
    cluster_data = df_results[cluster]
    fig, axes = plt.subplots(1, len(categorical_features), figsize=(15, 5))
    fig.suptitle(f'cluster {cluster}', fontsize=16)

    for i, feature in enumerate(categorical_features):
        ax = axes[i]
        feature_data = cluster_data[feature]
        ax.pie(feature_data.values(), labels=feature_data.keys(), autopct='%1.1f%%'
        ax.set_title(feature)

    plt.show()

# Calculate the count of each cluster
cluster_counts = data_frame_shuffled_original_modified['cluster'].value_counts()

# Create a pie plot for the cluster counts
plt.figure(figsize=(8, 8))
plt.pie(cluster_counts, labels=cluster_counts.index, autopct='%1.1f%%', startangle=
plt.title('Cluster Counts')
plt.show()
```

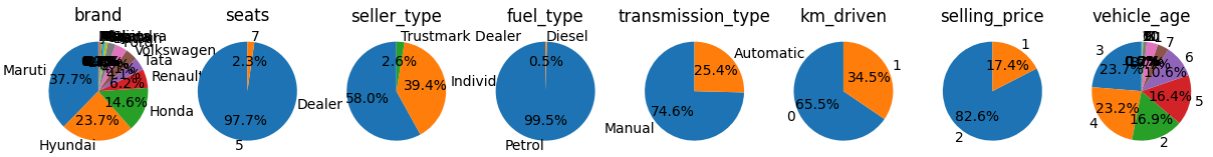
	brand	seats	seller_type	fuel_type	transmission_type	km_driven	selling_pric
cluster 3	{'Maruti': 1358, 'Hyundai': 887, 'Honda': 374, 'Volkswagen': 140, 'Ford': 84, 'Renault': 67, 'Skoda': 53, 'Datsun': 43, 'Tata': 5, 'Toyota': 3, 'Mercedes-Benz': 3, 'Mahindra': 3}	{5: 2990, 7: 30}	{'Dealer': 1893, 'Individual': 1090, 'Trustmark Dealer': 37}	{'Petrol': 3020}	{'Manual': 2814, 'Automatic': 206}	{1: 2758, 0: 176, 2: 86}	{1: 2061, 2 671, 0: 288
cluster 1	{'Maruti': 1565, 'Hyundai': 984, 'Honda': 605, 'Renault': 258, 'Tata': 169, 'Volkswagen': 168, 'Ford': 165, 'Datsun': 105, 'Skoda': 34, 'Toyota': 25, 'Kia': 18, 'Mahindra': 16, 'Jeep': 13, 'MG': 12, 'Audi': 4, 'Nissan': 4, 'Mini': 2}	{5: 4053, 7: 94}	{'Dealer': 2406, 'Individual': 1633, 'Trustmark Dealer': 108}	{'Petrol': 4127, 'Diesel': 20}	{'Manual': 3093, 'Automatic': 1054}	{0: 2715, 1: 1432}	{2: 3424, 1 723
cluster 0	{'Maruti': 1973, 'Hyundai': 947, 'Ford': 492, 'Honda': 476, 'Volkswagen': 295, 'Renault': 203, 'Skoda': 132, 'Tata': 95, 'Mahindra': 79, 'Datsun': 21, 'Kia': 7, 'Nissan': 3}	{5: 4430, 7: 285, 6: 8}	{'Dealer': 2756, 'Individual': 1947, 'Trustmark Dealer': 20}	{'Diesel': 4341, 'CNG': 232, 'Petrol': 106, 'LPG': 44}	{'Manual': 4553, 'Automatic': 170}	{1: 3777, 0: 583, 2: 363}	{2: 3587, 1 1103, 0: 33

	brand	seats	seller_type	fuel_type	transmission_type	km_driven	selling_price
cluster 2	{'Mahindra': 681, 'Toyota': 466, 'BMW': 236, 'Mercedes-Benz': 201, 'Hyundai': 163, 'Tata': 159, 'Audi': 150, 'Skoda': 109, 'Maruti': 91, 'Jaguar': 34, 'Honda': 26, 'Jeep': 26, 'Ford': 21, 'Volkswagen': 15, 'Land Rover': 12, 'Mini': 9, 'Volvo': 9, 'Isuzu': 8, 'MG': 7, 'Renault': 7, 'Kia': 3, 'Nissan': 3, 'ISUZU': 2, 'Porsche': 2}	{7: 1050, 5: 1035, 8: 198, 6: 105, 4: 36, 9: 15, 2: 1}	{'Dealer': 1728, 'Individual': 704, 'Trustmark Dealer': 8}	{'Diesel': 2100, 'Petrol': 267, 'CNG': 69, 'Electric': 4}	{'Manual': 1259, 'Automatic': 1181}	{1: 1808, 2: 320, 0: 312}	{2: 2315, 1 120, 0: 5

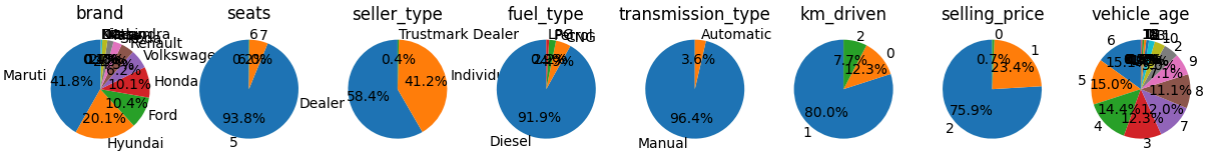
cluster cluster 3



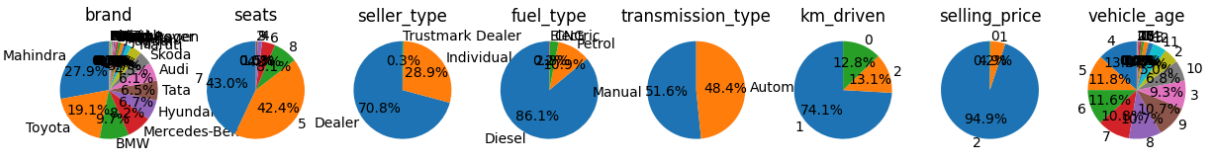
cluster cluster 1

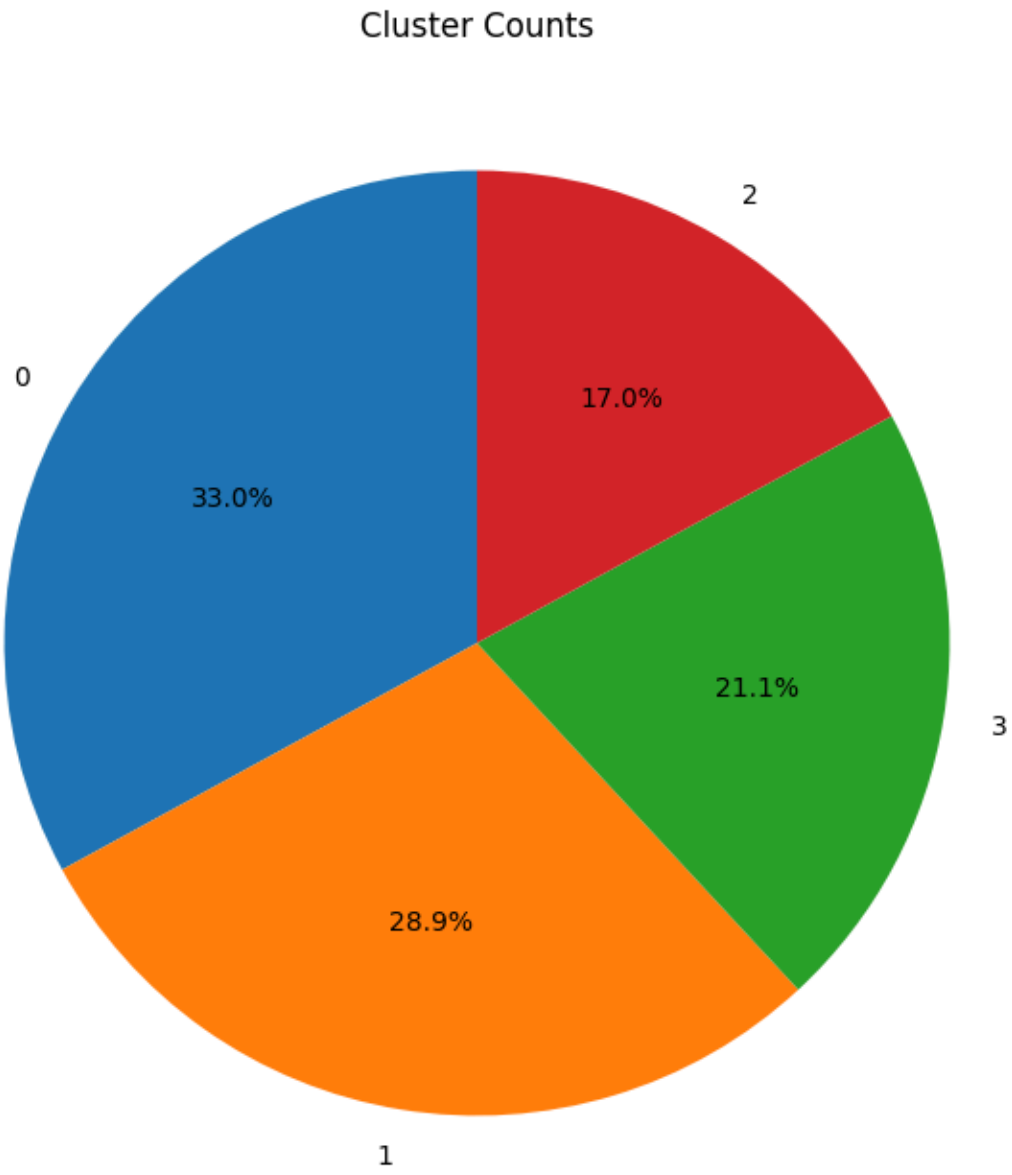


cluster cluster 0



cluster cluster 2





In [ ]:
```
data_frame_shuffled_original_modified[data_frame_shuffled_original_modified['cluste
```

Out[ ]:

	Unnamed: 0	car_name	brand	model	vehicle_age	km_driven	seller_type	fue
4	4	Ford Ecosport	Ford	Ecosport	6	0	Dealer	
10	10	Hyundai Verna	Hyundai	Verna	8	1	Dealer	
11	11	Renault Duster	Renault	Duster	5	1	Individual	
12	13	Maruti Ciaz	Maruti	Ciaz	5	1	Dealer	
13	14	Maruti Swift	Maruti	Swift	5	0	Individual	
15	16	Maruti Swift	Maruti	Swift	7	1	Dealer	
20	22	Hyundai Creta	Hyundai	Creta	2	0	Individual	
22	25	Maruti Swift Dzire	Maruti	Swift Dzire	6	1	Individual	
23	26	Maruti Swift Dzire	Maruti	Swift Dzire	8	1	Dealer	
31	34	Maruti Swift Dzire	Maruti	Swift Dzire	8	1	Individual	
32	35	Hyundai i20	Hyundai	i20	3	0	Individual	
35	38	Maruti Swift Dzire	Maruti	Swift Dzire	7	1	Dealer	
37	40	Mahindra Bolero	Mahindra	Bolero	1	1	Individual	
43	46	Maruti Wagon R	Maruti	Wagon R	3	1	Trustmark Dealer	
45	48	Hyundai Verna	Hyundai	Verna	7	1	Individual	
46	49	Hyundai i20	Hyundai	i20	5	1	Individual	
47	50	Maruti Swift Dzire	Maruti	Swift Dzire	4	1	Individual	
52	55	Volkswagen Vento	Volkswagen	Vento	9	1	Individual	
54	57	Hyundai Verna	Hyundai	Verna	8	1	Dealer	

	Unnamed: 0	car_name	brand	model	vehicle_age	km_driven	seller_type	fue
56	59	Hyundai Grand	Hyundai	Grand	6	1	Dealer	
59	62	Hyundai Grand	Hyundai	Grand	3	1	Individual	
63	66	Maruti Swift Dzire	Maruti	Swift Dzire	5	1	Dealer	
66	69	Maruti Vitara	Maruti	Vitara	3	0	Dealer	
69	72	Maruti Swift	Maruti	Swift	10	2	Individual	
76	79	Honda City	Honda	City	7	1	Dealer	
80	84	Hyundai i20	Hyundai	i20	10	2	Individual	
84	88	Volkswagen Polo	Volkswagen	Polo	8	1	Dealer	
89	93	Ford Ecosport	Ford	Ecosport	4	1	Dealer	
90	94	Hyundai Verna	Hyundai	Verna	9	1	Dealer	
96	100	Hyundai Verna	Hyundai	Verna	6	2	Individual	
98	102	Hyundai i20	Hyundai	i20	8	1	Dealer	
101	105	Maruti Swift	Maruti	Swift	6	1	Dealer	
105	109	Datsun GO	Datsun	GO	6	1	Individual	
109	113	Maruti Swift Dzire	Maruti	Swift Dzire	4	0	Individual	
113	117	Ford Ecosport	Ford	Ecosport	3	1	Individual	
116	120	Maruti Swift Dzire	Maruti	Swift Dzire	7	1	Individual	
118	122	Ford Aspire	Ford	Aspire	4	1	Dealer	
125	130	Honda Amaze	Honda	Amaze	3	1	Individual	
126	131	Mahindra KUV	Mahindra	KUV	5	1	Dealer	

	Unnamed: 0	car_name	brand	model	vehicle_age	km_driven	seller_type	fue
127	132	Datsun GO	Datsun	GO	6	1	Individual	

In [ ]:
```
data_frame_shuffled_original_modified[data_frame_shuffled_original_modified['cluste
```

Out[ ]:

	Unnamed: 0	car_name	brand	model	vehicle_age	km_driven	seller_type	fuel_type
1	1	Hyundai Grand	Hyundai	Grand	5	0	Individual	Petro
7	7	Maruti Wagon R	Maruti	Wagon R	3	0	Dealer	Petro
8	8	Hyundai Venue	Hyundai	Venue	2	0	Individual	Petro
9	9	Maruti Swift	Maruti	Swift	4	0	Dealer	Petro
16	18	Maruti Baleno	Maruti	Baleno	6	0	Individual	Petro
17	19	Maruti Swift Dzire	Maruti	Swift Dzire	5	1	Individual	Petro
19	21	Maruti Alto	Maruti	Alto	8	0	Dealer	Petro
21	23	Honda City	Honda	City	6	1	Individual	Petro
25	28	Hyundai i20	Hyundai	i20	3	0	Individual	Petro
27	30	Renault KWID	Renault	KWID	4	0	Individual	Petro
33	36	Hyundai i20	Hyundai	i20	3	1	Individual	Petro
34	37	Honda City	Honda	City	6	1	Dealer	Petro
36	39	Maruti Baleno	Maruti	Baleno	5	1	Dealer	Petro
38	41	Mahindra KUV100	Mahindra	KUV100	3	0	Individual	Petro
39	42	Maruti Baleno	Maruti	Baleno	5	0	Dealer	Petro
40	43	Maruti Swift Dzire	Maruti	Swift Dzire	5	1	Individual	Petro
41	44	Maruti Wagon R	Maruti	Wagon R	2	0	Individual	Petro
42	45	Maruti Ignis	Maruti	Ignis	2	0	Individual	Petro

	Unnamed: 0	car_name	brand	model	vehicle_age	km_driven	seller_type	fuel_type
44	47	Datsun RediGO	Datsun	RediGO	3	0	Individual	Petro
48	51	Hyundai i20	Hyundai	i20	2	1	Dealer	Petro
50	53	Hyundai i20	Hyundai	i20	3	1	Individual	Petro
58	61	Maruti Swift Dzire	Maruti	Swift Dzire	3	0	Individual	Petro
60	63	Ford Figo	Ford	Figo	4	0	Trustmark Dealer	Petro
64	67	Honda City	Honda	City	8	1	Individual	Petro
65	68	Renault KWID	Renault	KWID	5	0	Individual	Petro
67	70	Ford Aspire	Ford	Aspire	4	0	Individual	Petro
70	73	Tata Tiago	Tata	Tiago	4	1	Individual	Petro
71	74	Hyundai i20	Hyundai	i20	5	0	Dealer	Petro
77	80	Kia Seltos	Kia	Seltos	2	0	Individual	Petro
78	81	Maruti Baleno	Maruti	Baleno	5	1	Dealer	Petro

In [ ]:
```
data_frame_shuffled_original_modified[data_frame_shuffled_original_modified['cluste
```

Out[ ]:

	Unnamed: 0	car_name	brand	model	vehicle_age	km_driven	seller_type	fuel_t
14	15	Mercedes-Benz C-Class	Mercedes-Benz	C-Class	7	1	Dealer	Di
26	29	Toyota Fortuner	Toyota	Fortuner	8	1	Individual	Di
30	33	Mahindra XUV500	Mahindra	XUV500	9	2	Individual	Di
51	54	Mahindra Scorpio	Mahindra	Scorpio	4	1	Individual	Di
53	56	Mahindra Marazzo	Mahindra	Marazzo	2	1	Individual	Di
55	58	Mahindra XUV500	Mahindra	XUV500	3	0	Individual	Di
72	75	Mahindra Scorpio	Mahindra	Scorpio	5	1	Dealer	Di
74	77	Mercedes-Benz C-Class	Mercedes-Benz	C-Class	9	1	Dealer	Pe
86	90	Hyundai Verna	Hyundai	Verna	3	0	Dealer	Di
99	103	Hyundai Verna	Hyundai	Verna	2	0	Dealer	Di
103	107	Mahindra Scorpio	Mahindra	Scorpio	5	1	Dealer	Di
106	110	Mahindra Scorpio	Mahindra	Scorpio	4	0	Dealer	Di
107	111	BMW 5	BMW	5	5	1	Dealer	Di
110	114	Hyundai Creta	Hyundai	Creta	4	1	Dealer	Di
117	121	Mahindra Scorpio	Mahindra	Scorpio	8	1	Individual	Di
121	125	Honda CR-V	Honda	CR-V	6	1	Dealer	Pe
134	139	BMW 3	BMW	3	5	1	Dealer	Di
142	147	Audi A4	Audi	A4	8	1	Dealer	Di
147	153	Mahindra Scorpio	Mahindra	Scorpio	9	2	Individual	Di
161	168	Tata Safari	Tata	Safari	5	1	Dealer	Di

	Unnamed: 0	car_name	brand	model	vehicle_age	km_driven	seller_type	fuel_t
173	181	Tata Safari	Tata	Safari	8	1	Dealer	Di
176	184	Toyota Fortuner	Toyota	Fortuner	6	1	Individual	Di
192	200	Mahindra Scorpio	Mahindra	Scorpio	4	1	Dealer	Di
201	210	Mahindra Thar	Mahindra	Thar	3	0	Individual	Di
208	217	Mahindra Scorpio	Mahindra	Scorpio	6	1	Individual	Di
209	218	Mahindra XUV500	Mahindra	XUV500	8	1	Individual	Di
211	220	Mahindra XUV500	Mahindra	XUV500	4	1	Dealer	Di
216	225	Tata Hexa	Tata	Hexa	3	0	Individual	Di
240	251	Mahindra KUV	Mahindra	KUV	5	1	Individual	Di
241	252	Mahindra Scorpio	Mahindra	Scorpio	4	2	Individual	Di

In [ ]:
```
data_frame_shuffled_original_modified[data_frame_shuffled_original_modified['cluste
```

Out[ ]:

	Unnamed: 0	car_name	brand	model	vehicle_age	km_driven	seller_type	fuel_
0	0	Maruti Alto	Maruti	Alto	9	2	Individual	P
2	2	Hyundai i20	Hyundai	i20	11	1	Individual	P
3	3	Maruti Alto	Maruti	Alto	9	1	Individual	P
5	5	Maruti Wagon R	Maruti	Wagon R	8	1	Individual	P
6	6	Hyundai i10	Hyundai	i10	8	1	Dealer	P
18	20	Volkswagen Vento	Volkswagen	Vento	8	1	Dealer	P
24	27	Honda City	Honda	City	14	1	Dealer	P
28	31	Honda Amaze	Honda	Amaze	5	1	Dealer	P
29	32	Hyundai Santro	Hyundai	Santro	12	1	Individual	P
49	52	Hyundai Santro	Hyundai	Santro	14	1	Dealer	P
57	60	Ford Aspire	Ford	Aspire	4	1	Dealer	P
61	64	Honda Amaze	Honda	Amaze	7	1	Individual	P
62	65	Maruti Wagon R	Maruti	Wagon R	15	1	Dealer	P
68	71	Hyundai i20	Hyundai	i20	8	0	Individual	P
73	76	Volkswagen Polo	Volkswagen	Polo	5	1	Dealer	P
75	78	Maruti Wagon R	Maruti	Wagon R	7	1	Individual	P
79	83	Maruti Alto	Maruti	Alto	8	1	Individual	P
81	85	Ford Figo	Ford	Figo	10	1	Individual	P
82	86	Maruti Swift Dzire	Maruti	Swift Dzire	9	1	Dealer	P
85	89	Hyundai i20	Hyundai	i20	9	1	Dealer	P
93	97	Maruti Wagon R	Maruti	Wagon R	5	1	Individual	P

	Unnamed: 0	car_name	brand	model	vehicle_age	km_driven	seller_type	fuel_
100	104	Maruti Wagon R	Maruti	Wagon R	12	1	Dealer	P
102	106	Maruti Alto	Maruti	Alto	10	0	Individual	P
104	108	Hyundai i10	Hyundai	i10	7	1	Trustmark Dealer	P
112	116	Maruti Swift Dzire	Maruti	Swift Dzire	9	1	Dealer	P
114	118	Hyundai Grand	Hyundai	Grand	6	1	Dealer	P
122	126	Volkswagen Polo	Volkswagen	Polo	6	1	Dealer	P
141	146	Honda City	Honda	City	11	1	Dealer	P
156	163	Maruti Wagon R	Maruti	Wagon R	10	1	Individual	P
164	172	Hyundai i20	Hyundai	i20	6	1	Individual	P

Result Analysis

Klaster "0" (oko 33% entiteta, najveci klaster)

Gradski automobili i SUV automobili, karakterise ih sve vrte goriva sa dominantnim dizel motorima, najvise manuelnih menjaca, takodje dolaze u svim kilometrazama

Klaster "1" (oko 29% entiteta)

Klaster 1 je veoma slican klasteru 0, s tim da za razliku od klastera 0, klaster 1 ima vecinski benzinske motore i vise automatskih menjaca sto ih i cini malo skupljim. U ovom klasteru nemamo automobile sa malom kilometrazom. Automobili su skoro uvek mladji od 7 godina

Klaster "2" (oko 21% eniteta)

Off-road i veliki automobili, karaterise ih razlicit broj sedista (ne samo 5), poznati brendovi ovog tipa vozila, sve vrste predjene kilometraze, dominantni dizel motori i automatski menjaci, visoka cena. Za ovakve automobile, prodavac je najcesce diler automobila u poredjenju sa ostalim klasterima. Sve starosne kategorije.

Klaster "3" (oko 17% eniteta, najmanji klaster)

Gradski automobili sa malom cenom i srednjom kilometrazom, iskljucivo benzinski motori, skoro uvek manuelni menjac koji utice na cenu. Uglavnom stariji automobili (preko 9 godina)