July 11, 2022

[1]:
```
from bs4 import BeautifulSoup
import urllib as ul
import requests
import pandas as pd
import time as t
```

[3]:
```
feature = []
value = []
dictionary = []
price = []
carname= []
c=0
temp2=''
empty = True
for i in range(2,1500):
    str_i = str(i)
    url = "https://www.kcar.com/car/info/car_info_detail.do?
 ↪i_sCarCd=EC6068"+str_i
    response = requests.get(url)
    html = response.text
    soup = BeautifulSoup(html, 'html.parser')
    if soup.select_one('.car_line_list') != None :
        title = soup.select_one('.car_line_list').get_text().split()
        if title[1] == "  ":
            empty = True
        else :
            empty = False
            if soup.select_one('.calc_detail_list>li:nth-child(1)>div') != None␣
 ↪:
                price.append(soup.select_one('.calc_detail_list>li:
 ↪nth-child(1)>div').get_text())
                cn=soup.select_one('h2').get_text().split()
                for i in range(len(cn)):
                    temp2 = temp2+cn[i]
                carname.append(temp2)
                temp2=''
        if empty == False :
```

```
            for i in range(3):
                title.pop()
            ad = [9,8,5,4]
            for i in ad:
                temp = title[i]
                title.remove(title[i])
                if i == 8 :
                    a = temp[0:3]
                    b = temp[3:]
                else:
                    a=temp[0:2]
                    b=temp[2:]
                title.append(a)
                title.append(b)

            c = c+1
            value.append([])
            for j in range(len(title)):
                if j % 2 == 0:
                    if c == 1:
                        feature.append(title[j])
                else :
                    value[c-1].append(title[j])


for p in range(len(price)):
    value[p].append(price[p])
    value[p].append(carname[p])
feature.append(" ")
feature.append("  ")
df = pd.DataFrame(value, columns = feature)
```

[86]:
```
df.to_csv("  1500.csv")
```

[6]:
```
df = pd.read_csv('  1500.csv')
df.head()
```

[6]:

	Unnamed: 0				/		\
0	0	20 3870	998cc	16,465Km		5	
1	1	53 5162	998cc	42,082Km		5	
2	2	67 0998	3,342cc	62,265Km	4	5	
3	3	106 5607	1,995cc	38,233Km	SUV	5	
4	4	63 4826	1,998cc	56,155Km	SUV	7	

0	2018	9,800,000	(JA)
1	2015	6,500,000	

```
2  2016  26,700,000         DHG330  AWD
3  2019  25,800,000      TL 2.02WD
4  2017  24,200,000        2.0 2WD
```

[7]:
```
data = df[['  ',' ',' ']]
data.head()
```

[7]:
```
0  16,465Km  2018   9,800,000
1  42,082Km  2015   6,500,000
2  62,265Km  2016  26,700,000
3  38,233Km  2019  25,800,000
4  56,155Km  2017  24,200,000
```

[8]:
```
temp=data['  '].str.replace('Km','')
temp=temp.str.replace(',','')
temp=temp.astype(int)
temp
```

[8]:
```
0      16465
1      42082
2      62265
3      38233
4      56155
       …
479    92211
480    10040
481    34013
482    12291
483    72590
Name:    , Length: 484, dtype: int32
```

[26]:
```
temp2 = data[' '].str.replace(' ','2018')
temp2 = temp2.astype(int)
```

[27]:
```
temp3=data[' '].str.replace(' ','')
temp3=temp3.str.replace(',','')
temp3=temp3.astype(int)
temp3
```

[27]:
```
0       9800000
1       6500000
2      26700000
3      25800000
4      24200000
         …
479     9500000
```

```
480    78500000
481     9500000
482    51000000
483     9900000
Name:  , Length: 484, dtype: int32
```

[28]:
```
data2 = pd.DataFrame([temp,temp2,temp3])
data2=data2.transpose()
```

[29]:
```
data2
```

[29]:
```
0    16465  2018   9800000
1    42082  2015   6500000
2    62265  2016  26700000
3    38233  2019  25800000
4    56155  2017  24200000
..     …    …        …
479  92211  2012   9500000
480  10040  2021  78500000
481  34013  2015   9500000
482  12291  2022  51000000
483  72590  2016   9900000

[484 rows x 3 columns]
```

[30]:
```
data2.describe()
```

[30]:
```
count     484.000000   484.000000  4.840000e+02
mean    58378.074380  2017.113636  1.955645e+07
std     33106.338112     2.676773  1.232710e+07
min         5.000000  2009.000000  4.200000e+06
25%     31289.000000  2015.000000  1.080000e+07
50%     55681.000000  2018.000000  1.685000e+07
75%     82749.250000  2019.000000  2.420000e+07
max    169329.000000  2023.000000  7.850000e+07
```

[31]:
```
df[df[' '] == " "]
```

[31]:
```
     Unnamed: 0                                /               \
91           91  83 5305  3,649cc   13,706Km   SUV        4    5
266         266  51 8276  1,591cc  100,633Km                5
459         459  63 6544  1,995cc   45,630Km   SUV            5


91         34,200,000    (GM )    4WD3.6  -X
```

```
266          6,400,000                          1.6
459         15,920,000                 ( )QM6 2WDSE
```

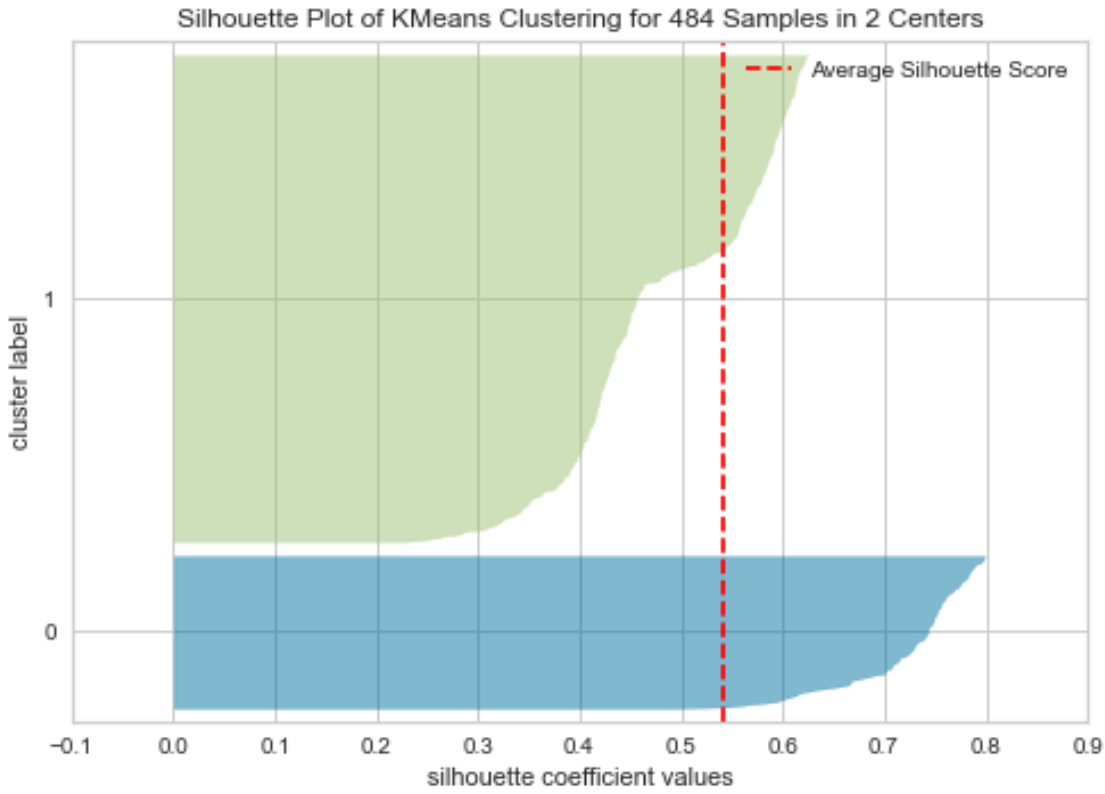
[32]:
```
df[df['  '] == "  1.6  "]
```

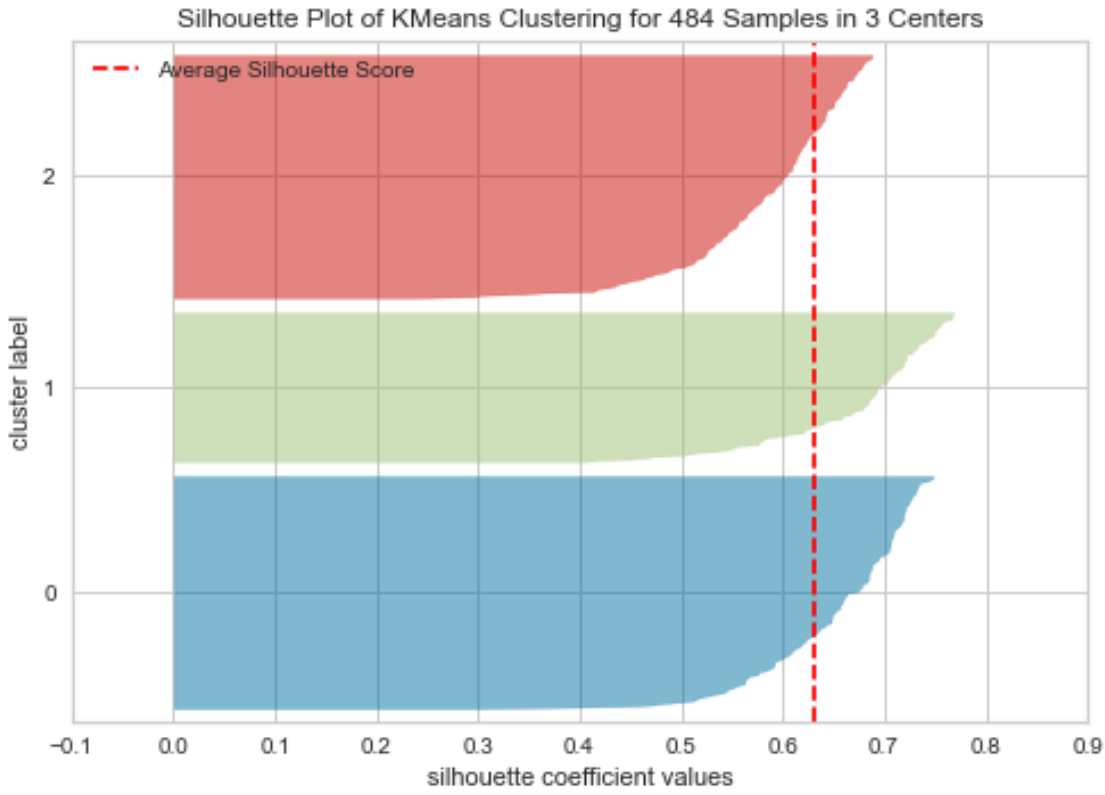
[32]:
```
     Unnamed: 0                                  /                  \
155         155   66 6904   1,591cc   79,306Km                    5
266         266   51 8276   1,591cc  100,633Km                    5


     155      2011  6,900,000     1.6
     266            6,400,000     1.6
```

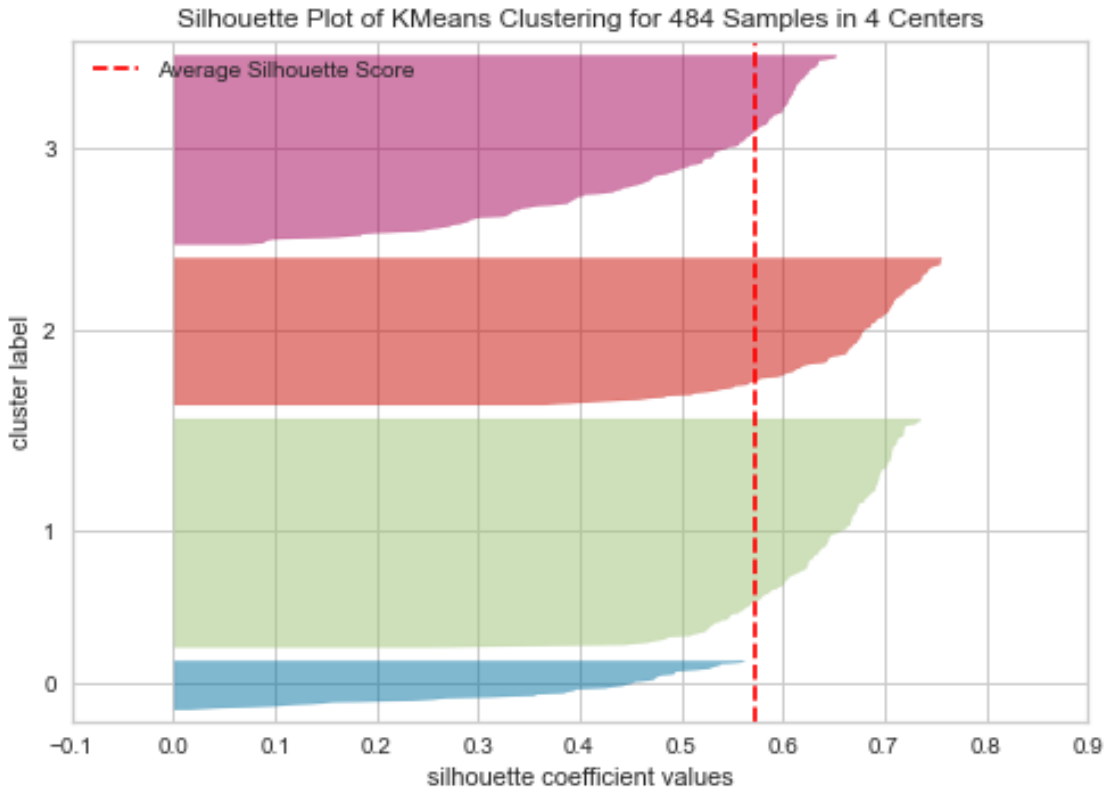
[33]:
```
data4=data2[['  ',' ']]
from sklearn.cluster import KMeans
from sklearn.preprocessing import MinMaxScaler

SSE=[]
scaler = MinMaxScaler()
data3 = scaler.fit_transform(data4)
```

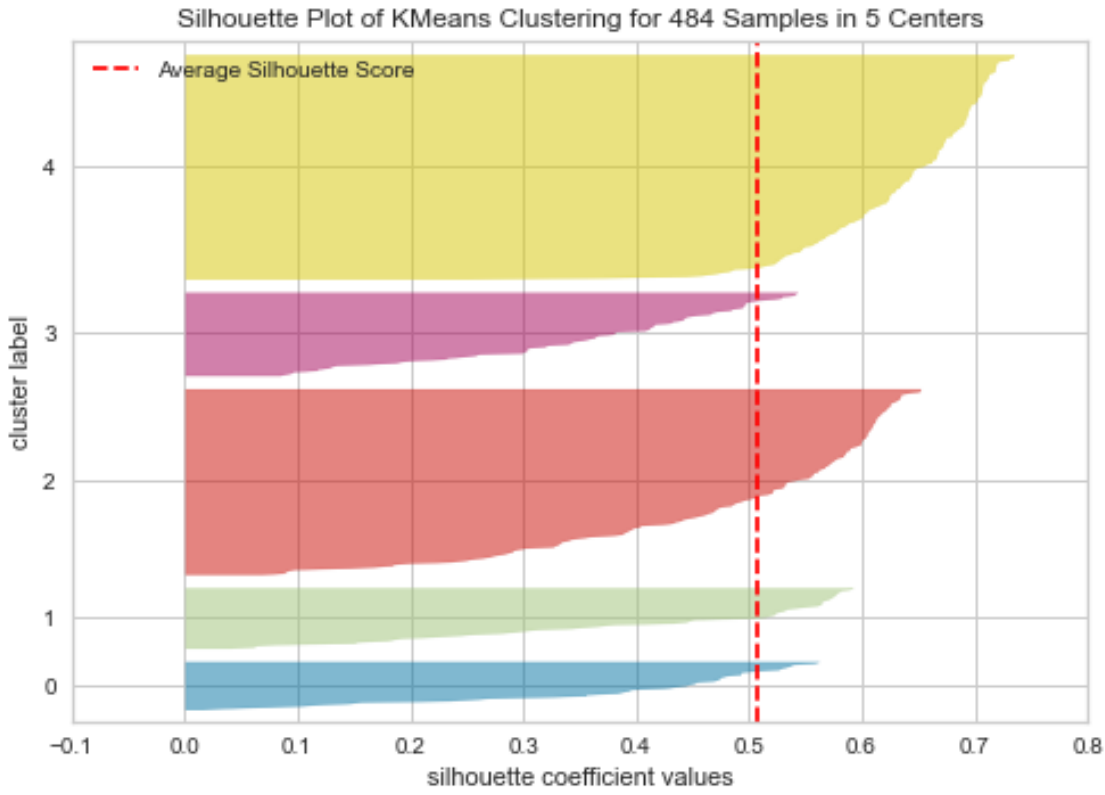
[39]:
```
import numpy as np
from sklearn.cluster import KMeans
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt
from sklearn.metrics import silhouette_samples, silhouette_score
from yellowbrick.cluster import SilhouetteVisualizer

kmeanslabel = []
silhouette_score_average = []
SSE=[]
scaler = MinMaxScaler()
data3 = scaler.fit_transform(data2)
for k in range(2,10):
        kmeans=KMeans(n_clusters = k, random_state = 42, max_iter = 300,n_init␣
 ↪= 10, init = 'random')
        kmeans.fit(data3)
        SSE.append(kmeans.inertia_)
        visualizer = SilhouetteVisualizer(kmeans, colors='yellowbrick')
        #  visualizer
        visualizer.fit(data3)
        visualizer.show()
        data2['Cluster'] = kmeans.labels_


print(silhouette_score_average)
f, ax = plt.subplots(1,1,figsize=(13,7))
```

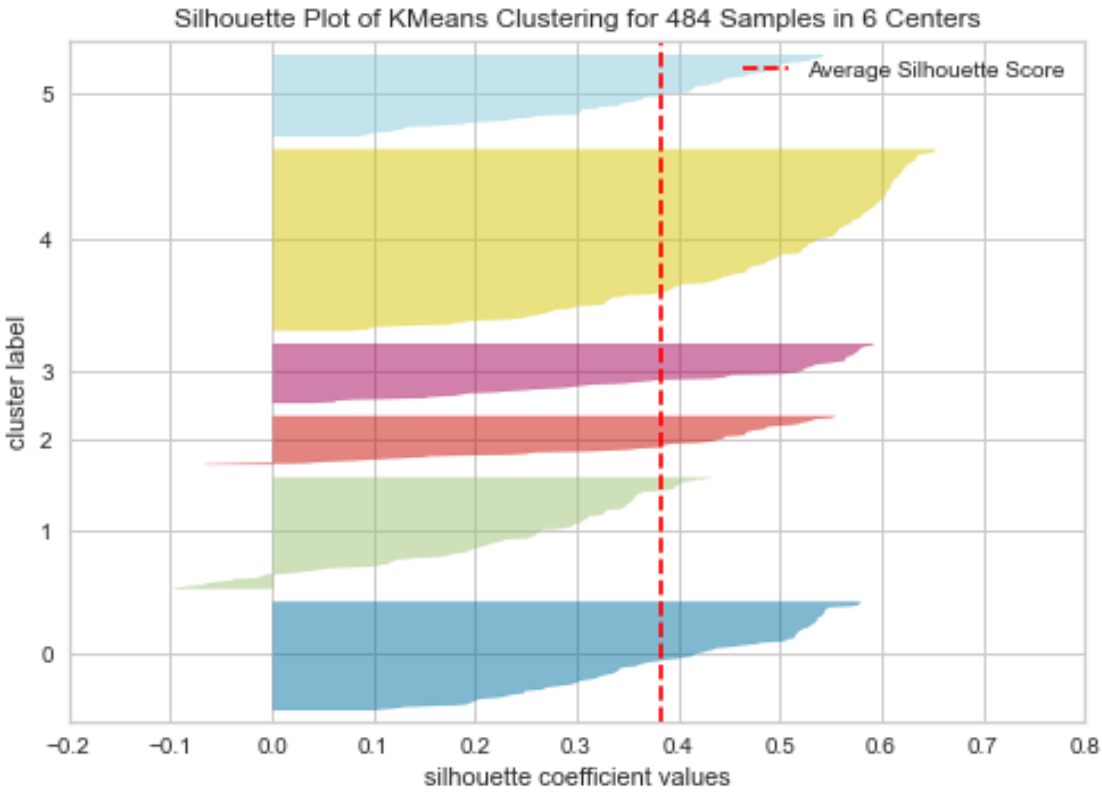
```
plt.plot(range(2, 10), SSE)
plt.xticks(range(2, 10))
plt.xlabel("Number of Clusters")
plt.ylabel("SSE")
plt.title('SSE for different number of clusters', fontsize = 20, c='black')
plt.show()
```

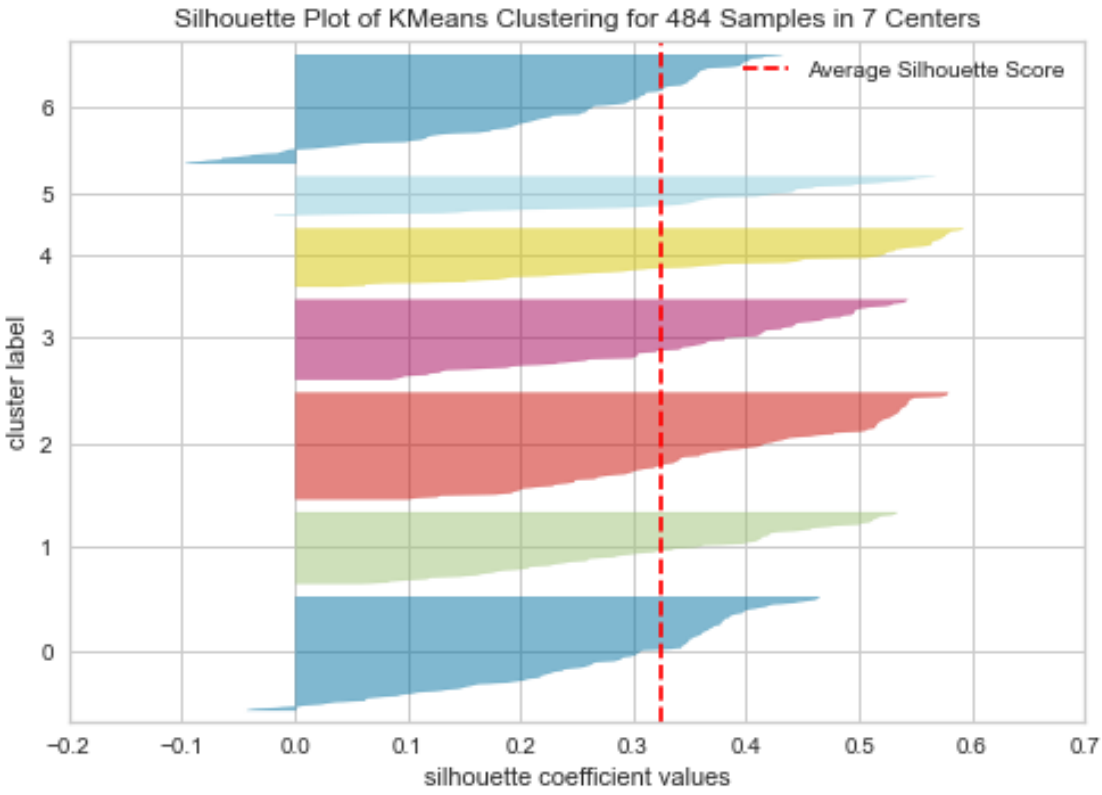


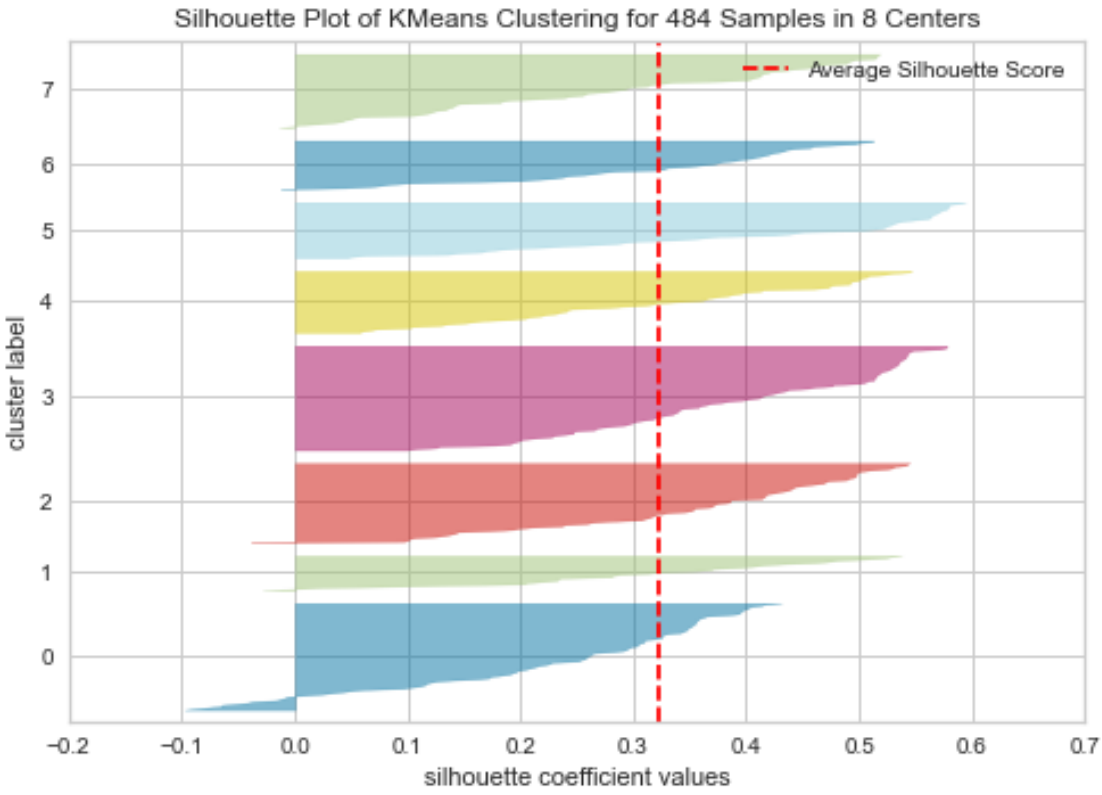

Silhouette Plot of KMeans Clustering for 484 Samples in 3 Centers
Average Silhouette Score
cluster label
2
1
0
−0.1
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
silhouette coefficient values

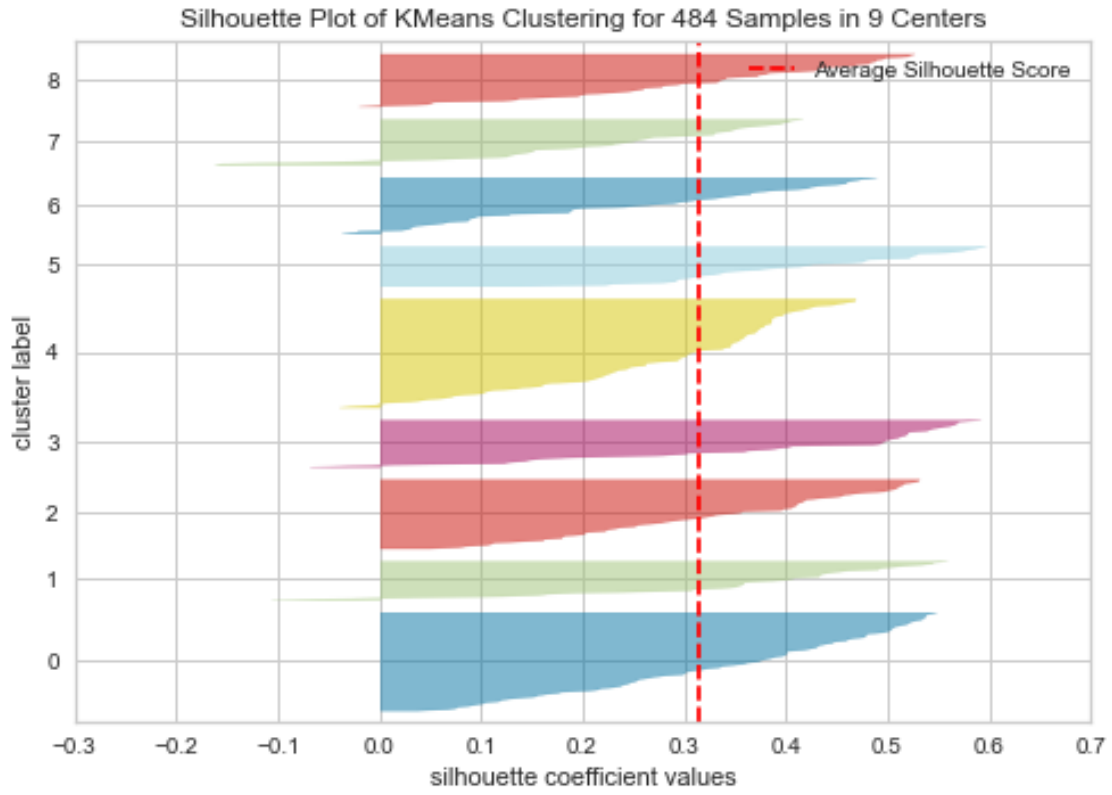

Silhouette Plot of KMeans Clustering for 484 Samples in 4 Centers
Average Silhouette Score
cluster label
3
2
1
0
−0.1
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
silhouette coefficient values


Silhouette Plot of KMeans Clustering for 484 Samples in 5 Centers
Average Silhouette Score
cluster label
4
3
2
1
0
−0.1
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
silhouette coefficient values


Silhouette Plot of KMeans Clustering for 484 Samples in 6 Centers
Average Silhouette Score
cluster label
0
1
2
3
4
5
−0.2
−0.1
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
silhouette coefficient values

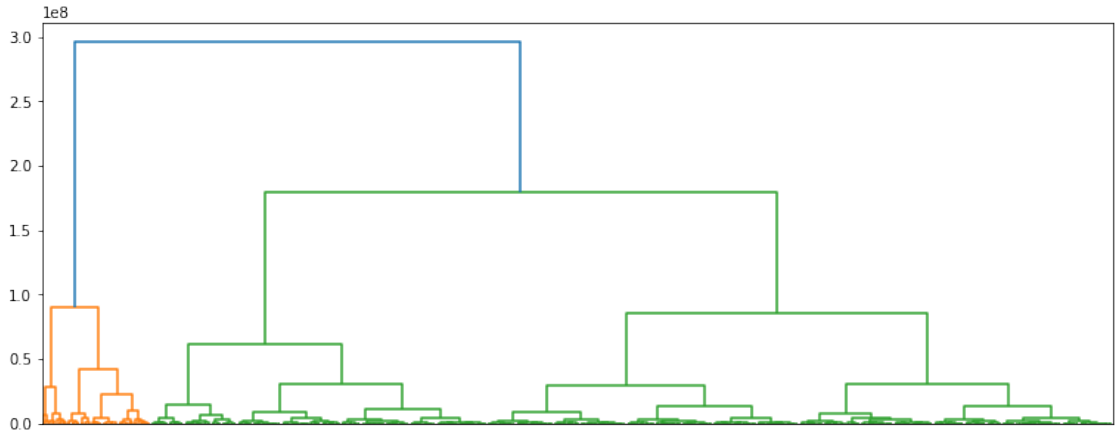

Silhouette Plot of KMeans Clustering for 484 Samples in 7 Centers
Average Silhouette Score
cluster label
0
1
2
3
4
5
6
−0.2
−0.1
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
silhouette coefficient values


Silhouette Plot of KMeans Clustering for 484 Samples in 8 Centers
Average Silhouette Score
cluster label
0
1
2
3
4
5
6
7
−0.2
−0.1
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
silhouette coefficient values



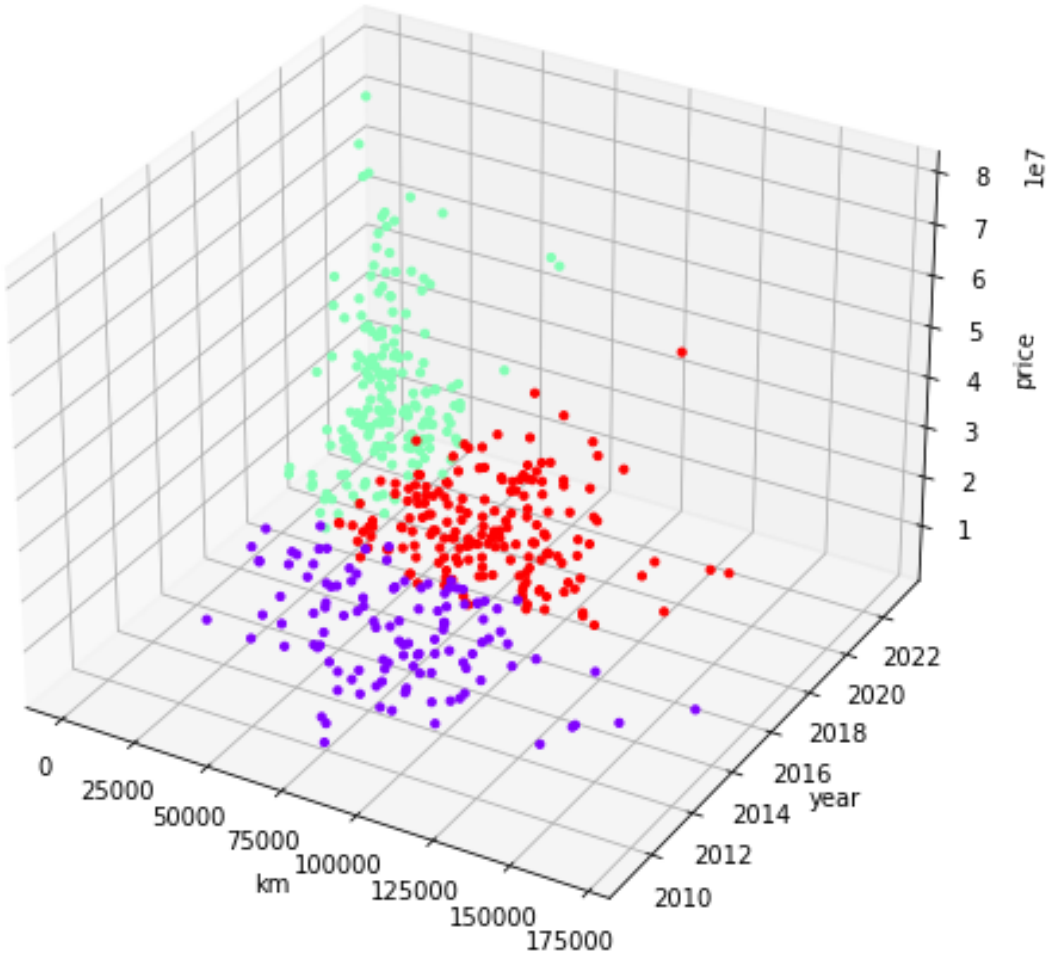

```
[0.15403389273688878, 0.056226754936875144, 0.10582678619871165,
-0.023231260508227363, -0.05940150847712007, -0.11285864254087809,
-0.13864179620762687, -0.1505421417242551]
```

[35]:
```
import scipy.cluster.hierarchy as shh

ax,fig = plt.subplots(1,1, figsize=(13,5))
dendro = shh.dendrogram(shh.linkage(data2, method = 'ward'))
plt.tick_params(
    axis='x',          # changes apply to the x-axis
    which='both',      # both major and minor ticks are affected
    bottom=False,      # ticks along the bottom edge are off
    top=False,         # ticks along the top edge are off
    labelbottom=False) # labels along the bottom edge are off
```



[38]:
```
import seaborn as sns
kmeans = KMeans(n_clusters = 3)
kmeans.fit(data3)

fig = plt.figure(figsize=(8, 8))
ax = fig.add_subplot(111, projection='3d')

X = data2
X['Cluster'] = kmeans.labels_
#    scatterplot
ax.scatter(  X['  ']
           , X[' ']
           , X[' ']
            , c = X['Cluster']
           , s = 10
           , cmap = "rainbow"
           , alpha = 1
          )
```

```
ax.set_zlabel('price')
ax.set_xlabel('km')
ax.set_ylabel('year')
plt.show()
```



[ ]: