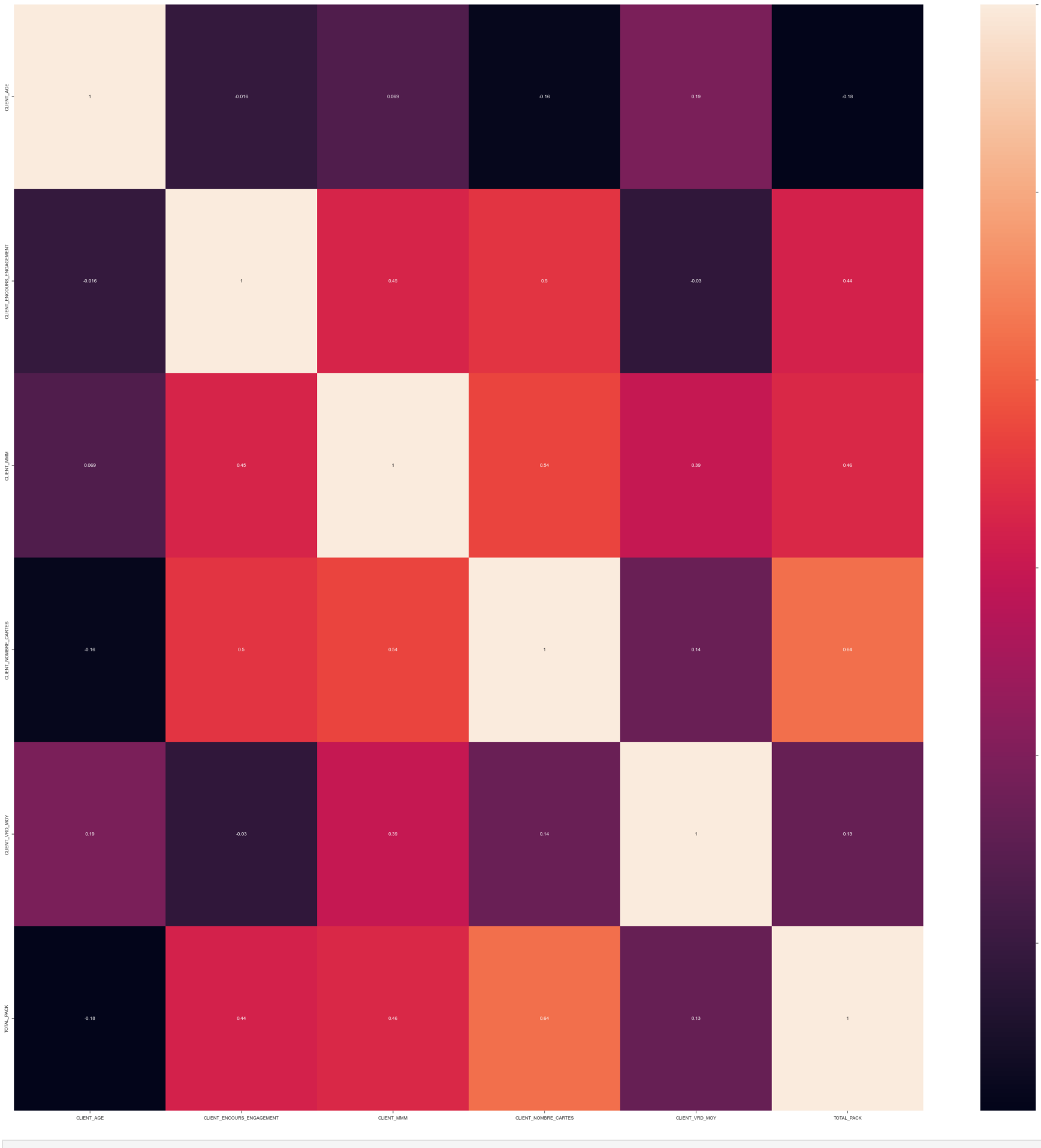
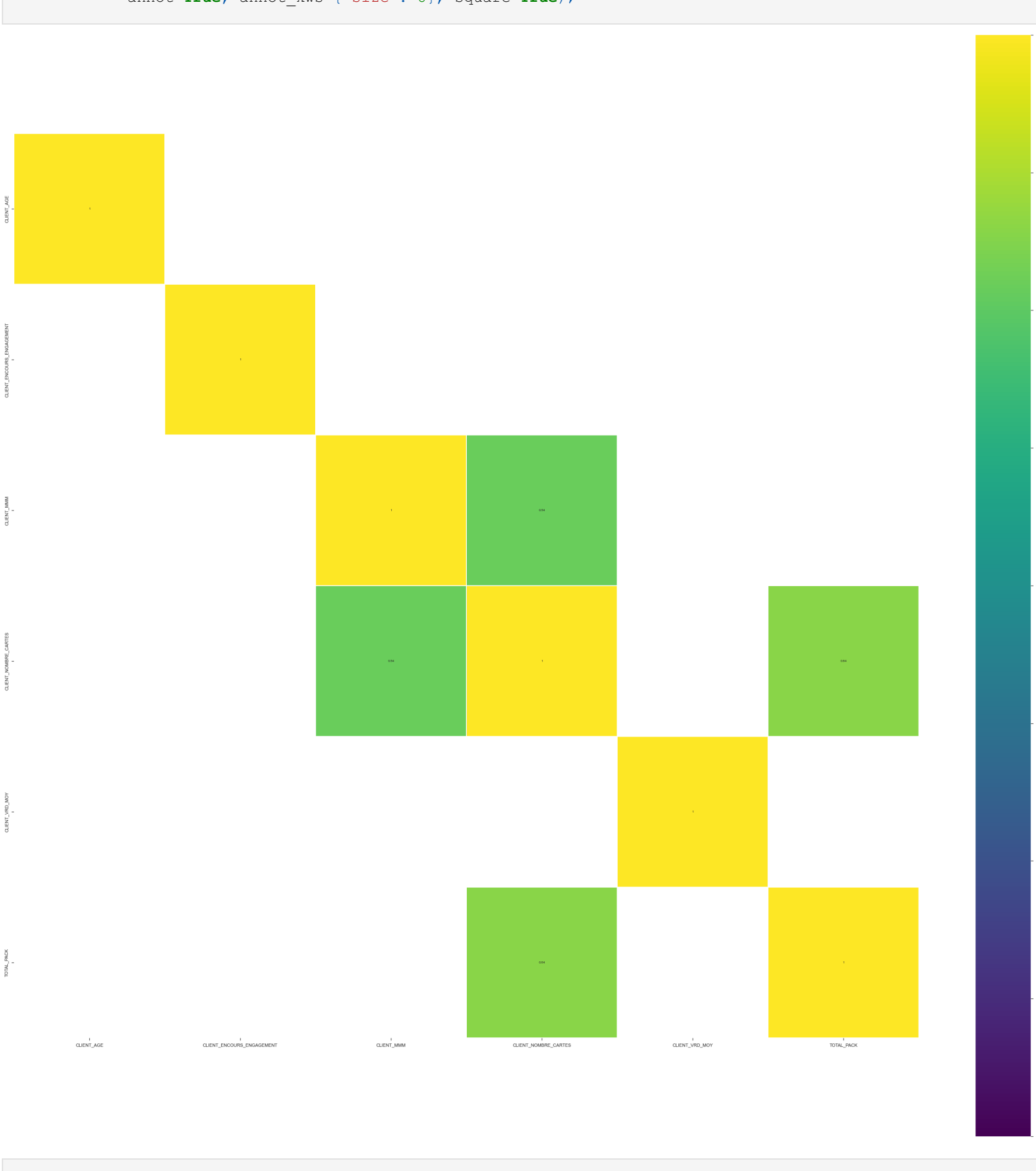


In [102]:
```python
plt.figure(figsize=(50, (50)))

sns.heatmap(correlation[(correlation >= 0.5) | (correlation <= -0.4)],
            cmap='viridis', vmax=1.0, vmin=-1.0, linewidths=0.1,
            annot=True, annot_kws={"size": 8}, square=True);
```



In [103]:
```python
data.corr()
```

Out[103]:

	CLIENT_AGE	CLIENT_ENCOURS_ENGAGEMENT	CLIENT_MMM	CLIENT_NOMBRE_CARTES	CLIENT_VRD_MOY	TOT
CLIENT_AGE	1.000000	-0.016378	0.069281	-0.161171	0.186538	-
CLIENT_ENCOURS_ENGAGEMENT	-0.016378	1.000000	0.447925	0.498627	-0.029685	
CLIENT_MMM	0.069281	0.447925	1.000000	0.543113	0.393993	
CLIENT_NOMBRE_CARTES	-0.161171	0.498627	0.543113	1.000000	0.142612	
CLIENT_VRD_MOY	0.186538	-0.029685	0.393993	0.142612	1.000000	
TOTAL_PACK	-0.178431	0.440892	0.463931	0.644047	0.132922	

Feature Scaling

Label Encoder

In [104]:
```python
encoder = LabelEncoder()
data_encoded = data.copy()
data_encoded.CLIENT_PROFESSION = encoder.fit_transform(data_encoded.CLIENT_PROFESSION)
data_encoded.CLIENT_TYPE_DEPOSANT = encoder.fit_transform(data_encoded.CLIENT_TYPE_DEPOSANT)
```

In [105]:
```python
data_encoded
```

Out[105]:

	CLIENT_AGE	CLIENT_ENCOURS_ENGAGEMENT	CLIENT_MMM	CLIENT_NOMBRE_CARTES	CLIENT_PROFESSION	CLIENT_TYPE_DEPOSANT
0	51	176.48125	916.666	0	14	0
1	58	84.39600	0.000	0	14	0
2	57	176.48125	1252.316	1	0	0
3	52	176.48125	1748.310	3	1	0
4	58	0.00000	216.331	0	1	0
...	...	...	...	...	...	...
834748	39	0.00000	0.000	0	11	1
834749	51	0.00000	0.000	1	10	4
834750	45	0.00000	0.000	0	6	1
834751	38	176.48125	0.000	1	11	0
834752	42	176.48125	0.000	1	11	0

743339 rows × 8 columns

MinMaxScaler

In [106]:
```python
scaler = MinMaxScaler()
model=scaler.fit(data_encoded)
scaled=model.transform(data_encoded)
scaled_df = pd.DataFrame(scaled,columns=data_encoded.columns)
scaled_df.head()
```

Out[106]:

	CLIENT_AGE	CLIENT_ENCOURS_ENGAGEMENT	CLIENT_MMM	CLIENT_NOMBRE_CARTES	CLIENT_PROFESSION	CLIENT_TYPE_DEPOSANT	CLIEN
0	0.533333	1.000000	0.309948	0.0	0.451613	0.0	
1	0.650000	0.478215	0.000000	0.0	0.451613	0.0	
2	0.633333	1.000000	0.423440	0.2	0.000000	0.0	
3	0.550000	1.000000	0.591148	0.6	0.032258	0.0	
4	0.650000	0.000000	0.073147	0.0	0.032258	0.0	

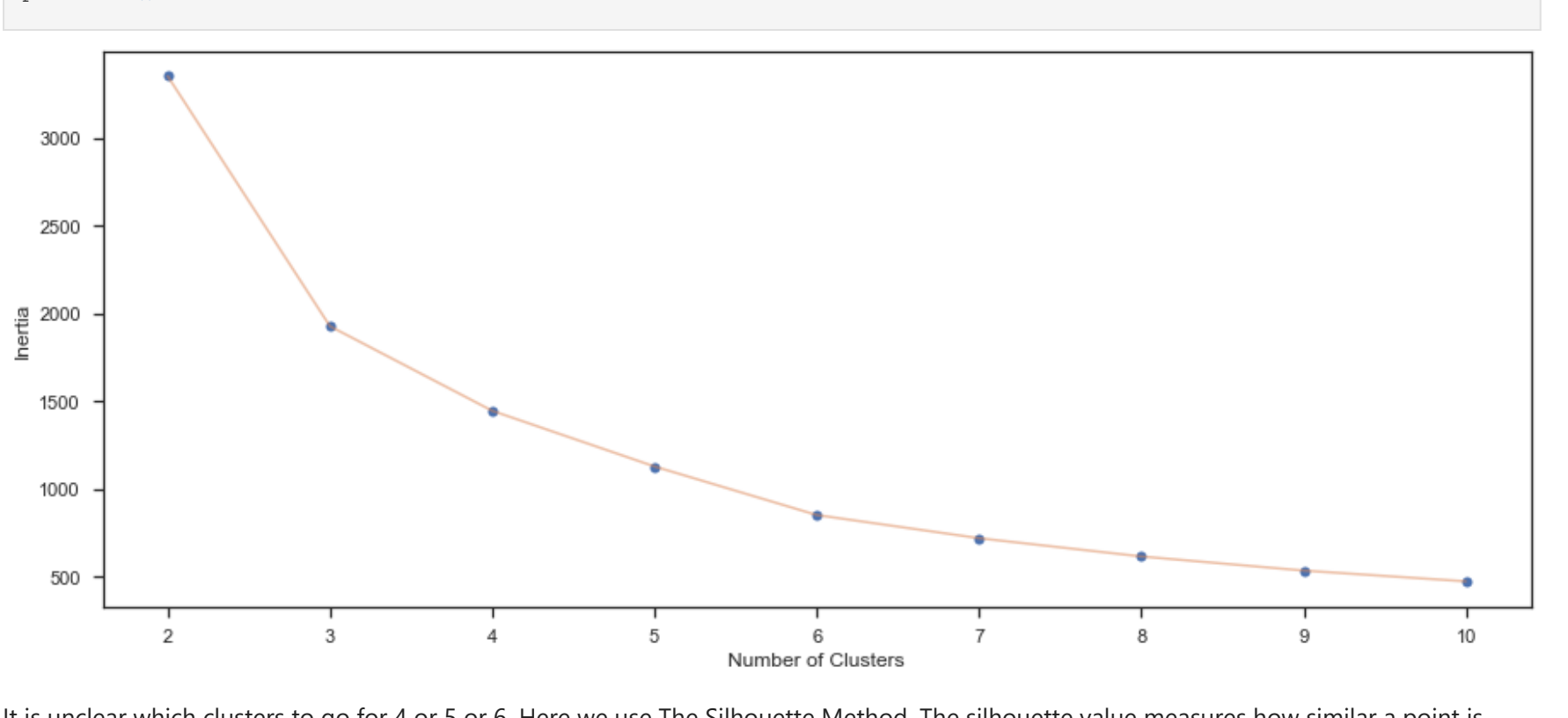
Cluster Analysis

Elbow method

In [107]:
```python
temp = scaled_df.sample(n=50000)
```

In [108]:
```python
X1 = temp[['CLIENT_AGE' , 'CLIENT_VRD_MOY']].iloc[: , :].values
inertia = []
s_scores = []
for n in range(2 , 11):
    algorithm = (KMeans(n_clusters = n ,init='k-means++', n_init = 10 ,max_iter=300,
                        tol=0.0001,  random_state= 111  , algorithm='elkan') )
    algorithm.fit(X1)
    inertia.append(algorithm.inertia_)
    silhouette_avg = silhouette_score(X1, algorithm.labels_)
    s_scores.append(silhouette_avg) # data for the silhouette score method
```

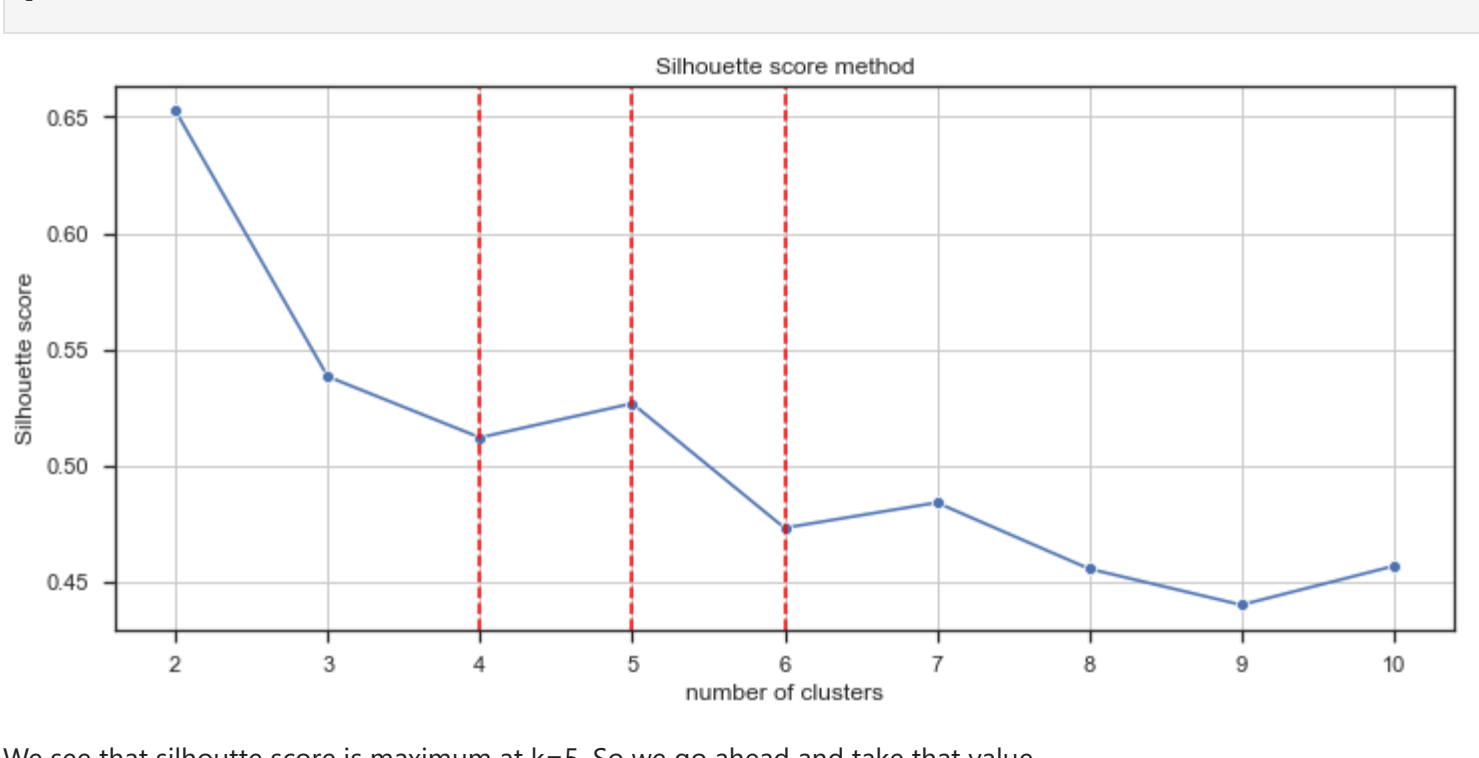
In [109]:
```python
#The Elbow Chart
plt.figure(1 , figsize = (15 ,6))
plt.plot(np.arange(2 , 11) , inertia , 'o')
plt.plot(np.arange(2 , 11) , inertia , '-' , alpha = 0.5)
plt.xlabel('Number of Clusters') , plt.ylabel('Inertia')
plt.show()
```



It is unclear which clusters to go for 4 or 5 or 6. Here we use The Silhouette Method. The silhouette value measures how similar a point is to its own cluster (cohesion) compared to other clusters (separation).The range of the Silhouette value is between +1 and -1. A high value is desirable and indicates that the point is placed in the correct cluster. If many points have a negative Silhouette value, it may indicate that we have created too many or too few clusters.

A high Silhouette Score is desirable. The Silhouette Score reaches its global maximum at the optimal k. This should ideally appear as a peak in the Silhouette Value-versus-k plot.

In [110]:
```python
fig, ax = plt.subplots(figsize=(12,5))
ax = sns.lineplot(np.arange(2 , 11), s_scores, marker='o', ax=ax)
ax.set_title("Silhouette score method")
ax.set_xlabel("number of clusters")
ax.set_ylabel("Silhouette score")
ax.axvline(4, ls="--", c="red")
ax.axvline(5, ls="--", c="red")
ax.axvline(6, ls="--", c="red")
plt.grid()
plt.show()
```

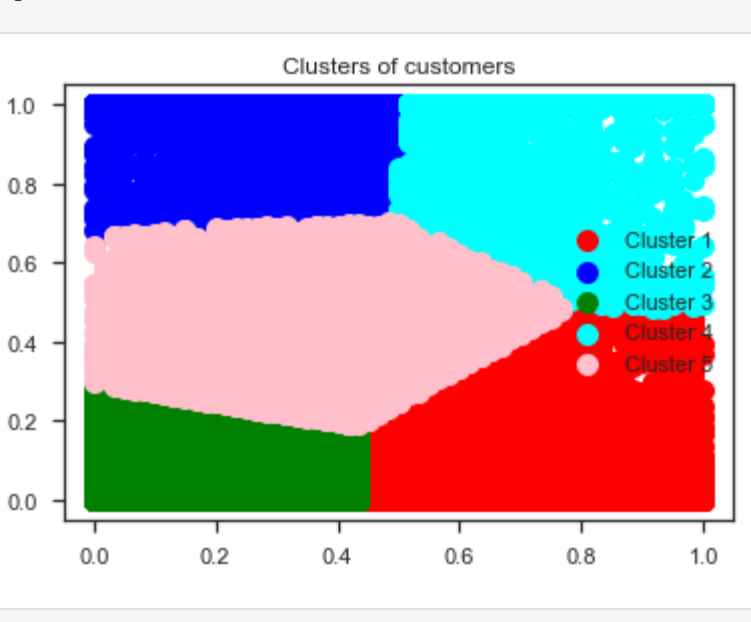


We see that silhoutte score is maximum at k=5. So we go ahead and take that value

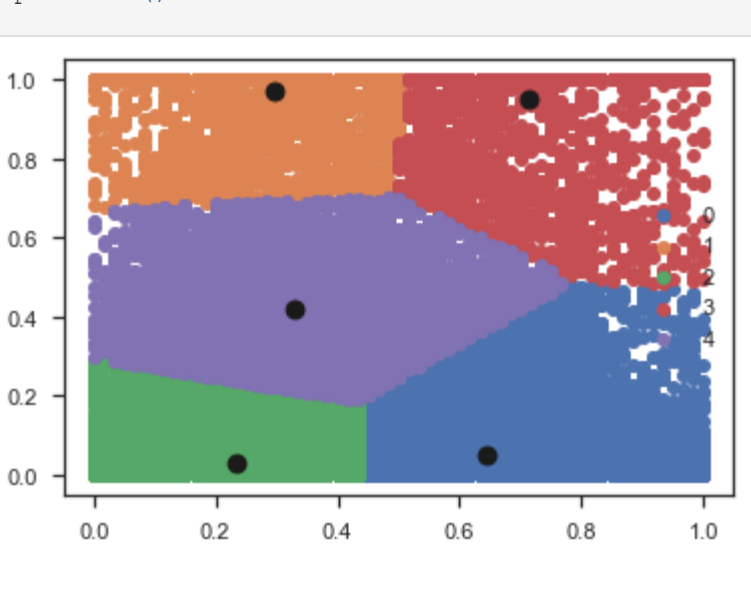
The Elbow Method is more of a decision rule, while the Silhouette is a metric used for validation while clustering. Thus, it can be used in combination with the Elbow Method.

In [111]:
```python
algorithm = (KMeans(n_clusters = 5 ,init='k-means++', n_init = 10 ,max_iter=300,
                    tol=0.0001,  random_state= 111  , algorithm='elkan') )
algorithm.fit(X1)
labels1 = algorithm.labels_
centroids1 = algorithm.cluster_centers_
```

In [112]:
```python
#Plotting the clusters
plt.scatter(X1[labels1 == 0, 0], X1[labels1 == 0, 1], s = 100, c = 'red', label = 'Cluster 1')
plt.scatter(X1[labels1 == 1, 0], X1[labels1 == 1, 1], s = 100, c = 'blue', label = 'Cluster 2')
plt.scatter(X1[labels1 == 2, 0], X1[labels1 == 2, 1], s = 100, c = 'green', label = 'Cluster 3')
plt.scatter(X1[labels1 == 3, 0], X1[labels1 == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')
plt.scatter(X1[labels1 == 4, 0], X1[labels1 == 4, 1], s = 100, c = 'pink', label = 'Cluster 5')
plt.title('Clusters of customers')
plt.legend()
plt.show()
```



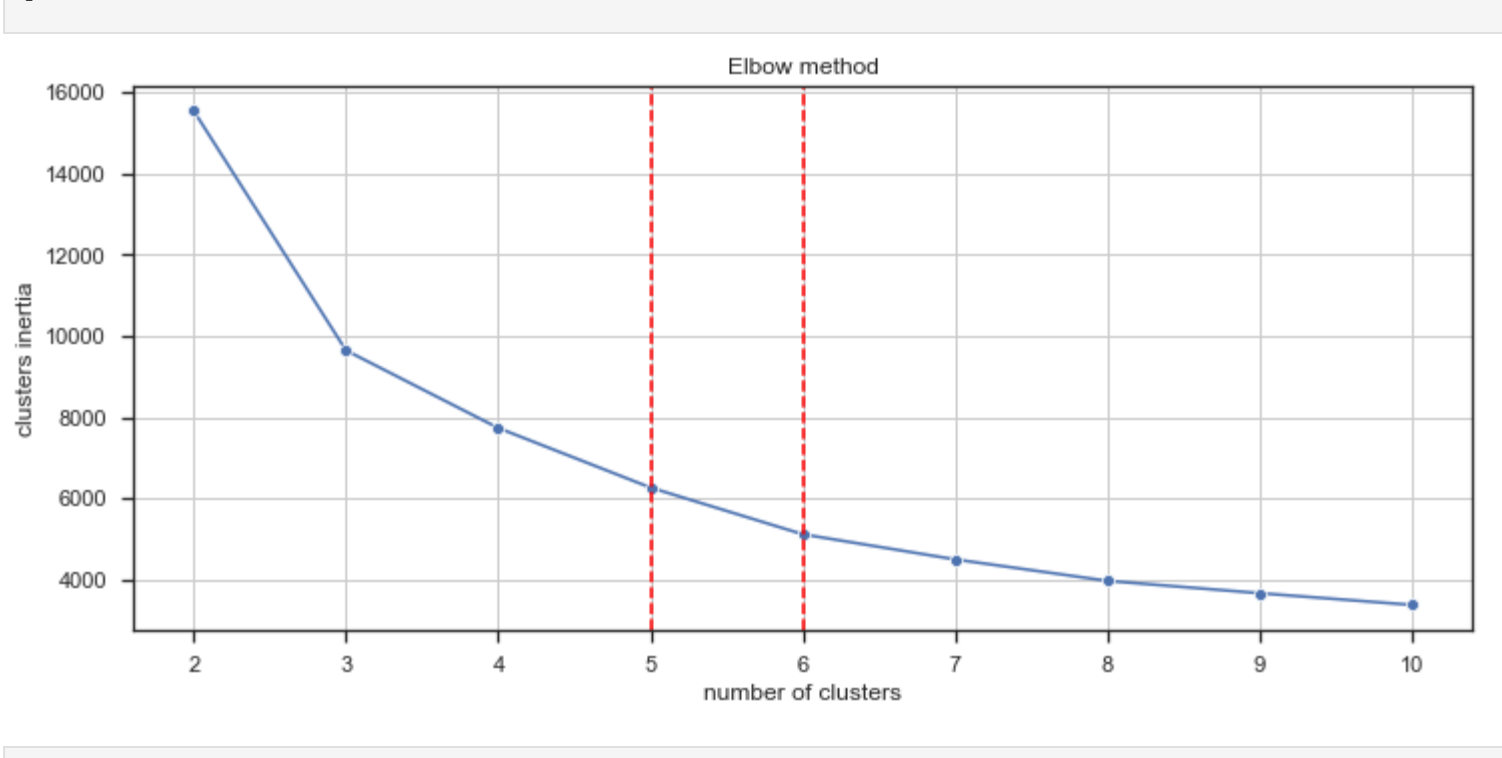
In [113]:
```python
#Getting the Centroids
centroids =algorithm.cluster_centers_
u_labels = np.unique(labels1)

#plotting the results:

for i in u_labels:
    plt.scatter(X1[labels1 == i , 0] , X1[labels1 == i , 1] , label = i)
plt.scatter(centroids[:,0] , centroids[:,1] , s = 80, color = 'k')
plt.legend()
plt.show()
```



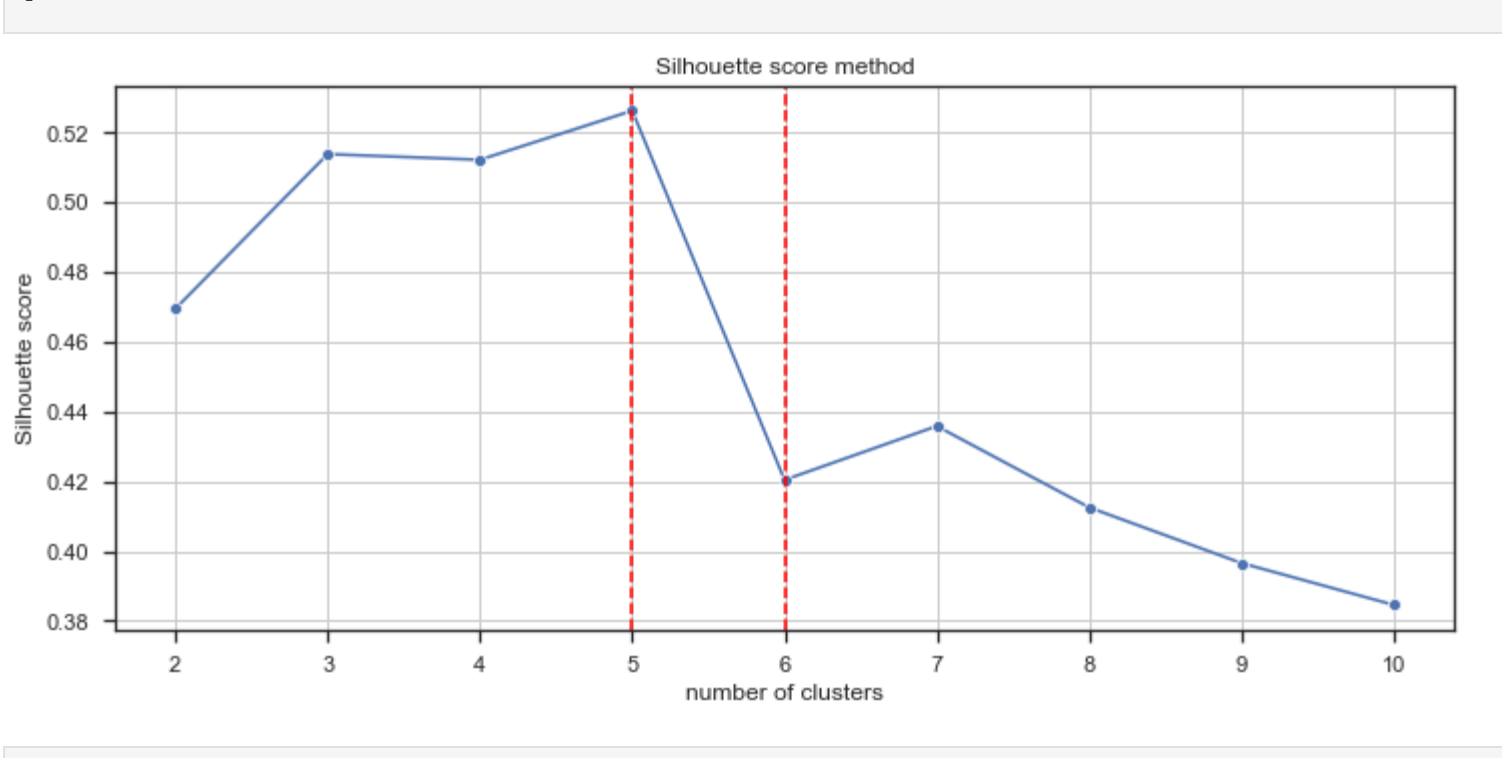
Perfoming segmentation using all the numeric variables

In [114]:
```python
X = temp[['CLIENT_AGE' ,'CLIENT_MMM' ,'CLIENT_ENCOURS_ENGAGEMENT','CLIENT_VRD_MOY','TOTAL_PACK']]
inertia = []
s_scores = []
for n in range(2 , 11):
    algorithm = KMeans(n_clusters = n ,init='k-means++', n_init = 10 ,max_iter=300,
                       tol=0.0001,  random_state= 111  , algorithm='elkan').fit(X)
    inertia.append(algorithm.inertia_)
    silhouette_avg = silhouette_score(X, algorithm.labels_)
    s_scores.append(silhouette_avg) # data for the silhouette score method
```

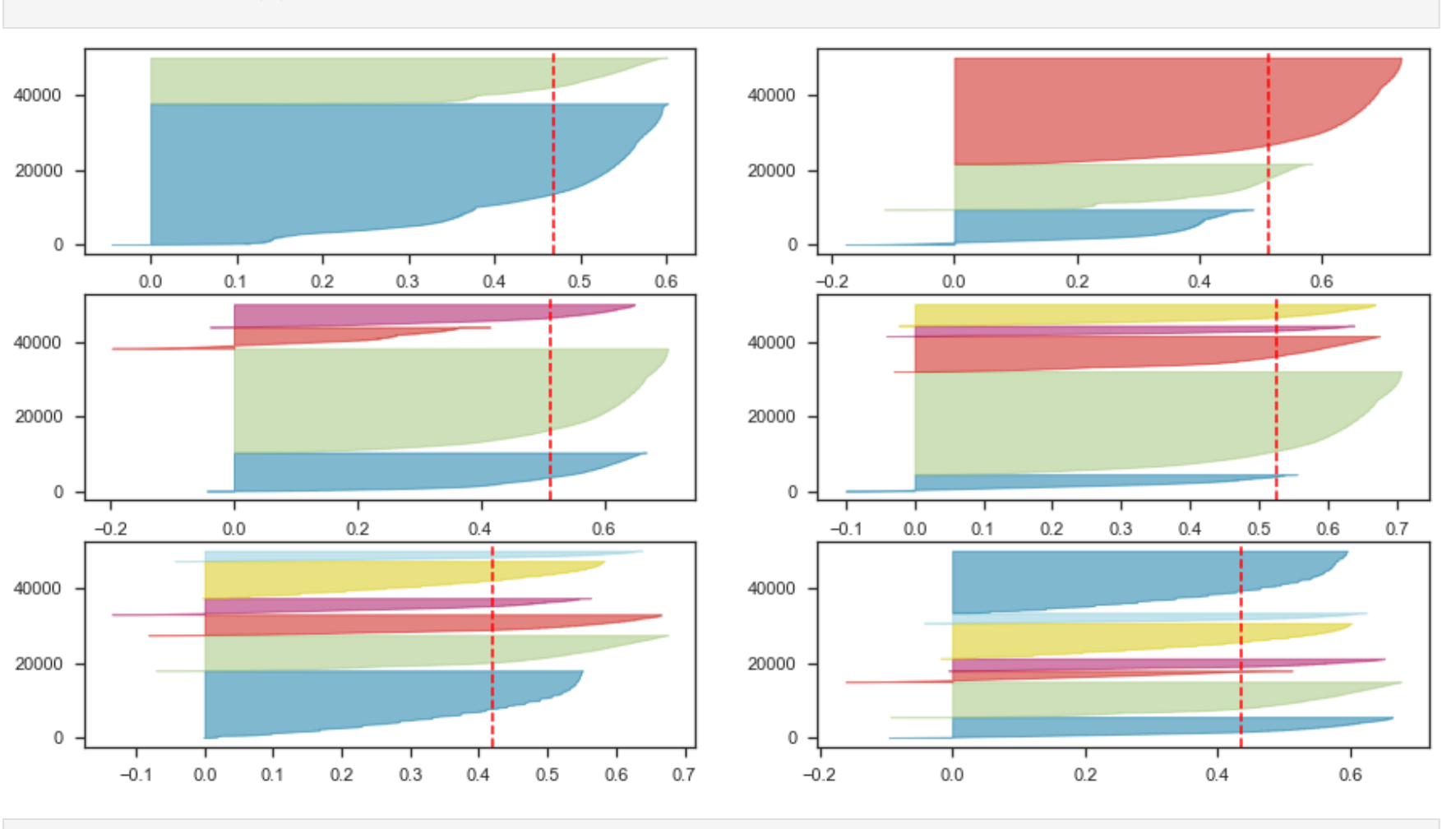
In [115]:
```python
fig, ax = plt.subplots(figsize=(12,5))
ax = sns.lineplot(np.arange(2 , 11), inertia, marker='o', ax=ax)
ax.set_title("Elbow method")
ax.set_xlabel("number of clusters")
ax.set_ylabel("clusters inertia")
ax.axvline(5, ls="--", c="red")
ax.axvline(6, ls="--", c="red")
plt.grid()
plt.show()
```



In [116]:
```python
fig, ax = plt.subplots(figsize=(12,5))
ax = sns.lineplot(np.arange(2 , 11), s_scores, marker='o', ax=ax)
ax.set_title("Silhouette score method")
ax.set_xlabel("number of clusters")
ax.set_ylabel("Silhouette score")
ax.axvline(5, ls="--", c="red")
ax.axvline(6, ls="--", c="red")

plt.grid()
plt.show()
```

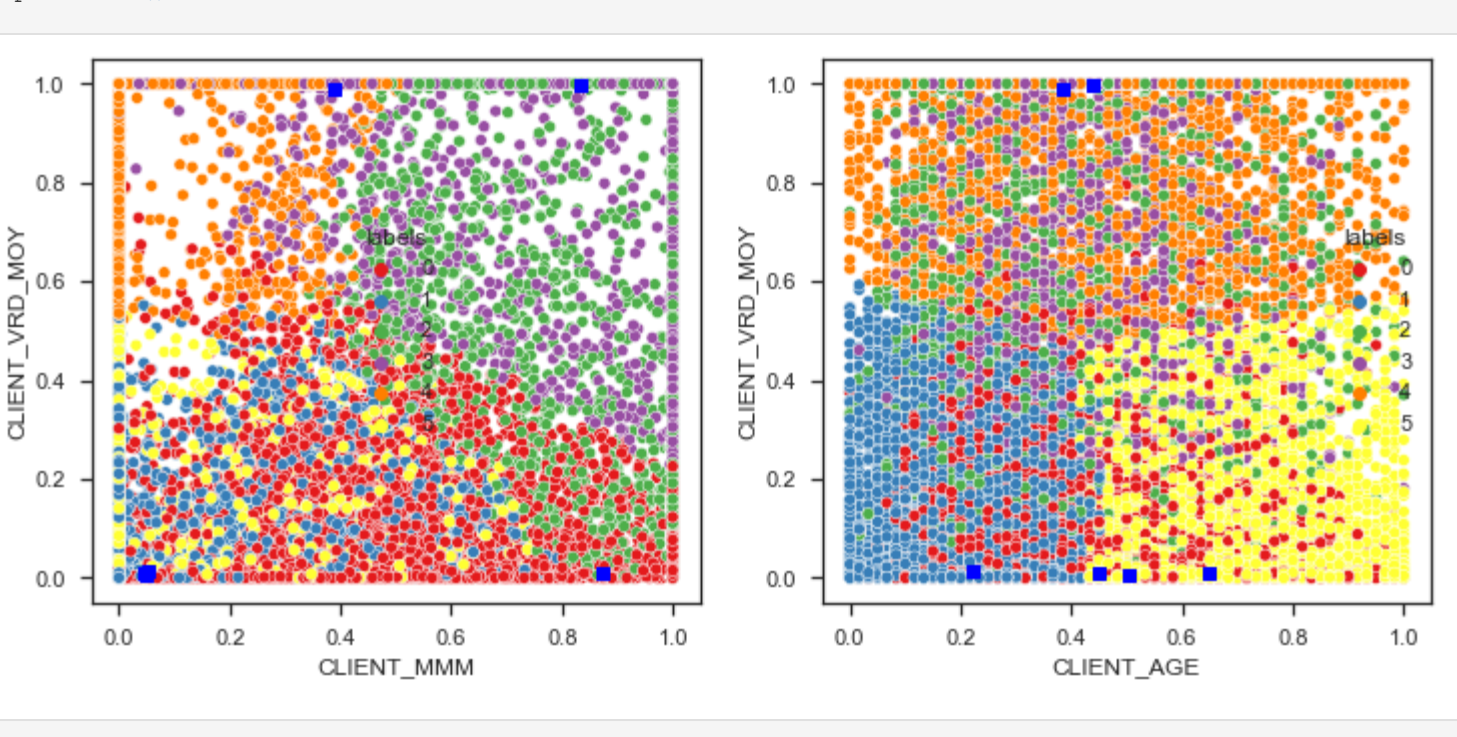


In [117]:
```python
fig, ax = plt.subplots(3, 2, figsize=(15,8))
for i in [2,3, 4, 5,6,7]:
    '''
    Create KMeans instance for different number of clusters
    '''
    km = KMeans(n_clusters=i, init='k-means++', n_init=10, max_iter=100, random_state=42)
    q, mod = divmod(i, 2)
    '''
    Create SilhouetteVisualizer instance with KMeans instance
    Fit the visualizer
    '''
    visualizer = SilhouetteVisualizer(km, colors='yellowbrick', ax=ax[q-1][mod])
    visualizer.fit(X)
```



In [118]:
```python
#For clusters of K=6
KM6 = (KMeans(n_clusters = 6 ,init='k-means++', n_init = 10 ,max_iter=300,
              tol=0.0001,  random_state= 111  , algorithm='elkan') )
KM6.fit(X)
labels6 = KM6.labels_
centroids6 = KM6.cluster_centers_
KM6_df = temp.copy()
KM6_df['labels'] = labels6
```

In [119]:
```python
fig1, (axes) = plt.subplots(1,2,figsize=(12,5))


scat_1 = sns.scatterplot('CLIENT_MMM', 'CLIENT_VRD_MOY', data=KM6_df,
                hue='labels', ax=axes[0], palette='Set1', legend='full')

sns.scatterplot('CLIENT_AGE', 'CLIENT_VRD_MOY', data=KM6_df,
                hue='labels', palette='Set1', ax=axes[1], legend='full')

axes[0].scatter(centroids6[:,1],centroids6[:,2], marker='s', s=40, c="blue")
axes[1].scatter(centroids6[:,0],centroids6[:,2], marker='s', s=40, c="blue")
plt.show()
```

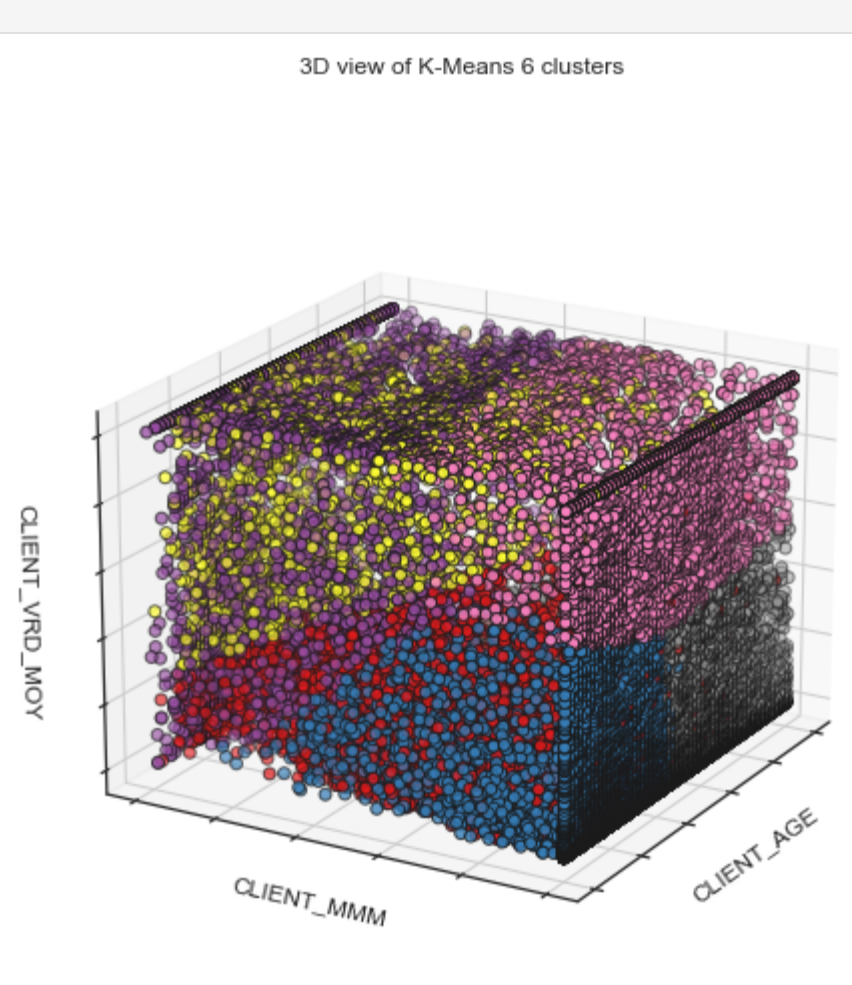


In [120]:
```python
KM_clust_sizes = KM6_df.groupby('labels').size().to_frame()
KM_clust_sizes.columns = ["KM_size"]
KM_clust_sizes
```

Out[120]:

labels	KM_size
0	9469
1	17963
2	4336
3	2765
4	5562
5	9905

In [121]:
```python
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure(figsize=(7, 7))
ax = Axes3D(fig, rect=[0, 0, .99, 1], elev=20, azim=210)
ax.scatter(KM6_df['CLIENT_AGE'],
           KM6_df['CLIENT_MMM'],
           KM6_df['CLIENT_VRD_MOY'],
           c=KM6_df['labels'],
           s=35, edgecolor='k', cmap=plt.cm.Set1)

ax.w_xaxis.set_ticklabels([])
ax.w_yaxis.set_ticklabels([])
ax.w_zaxis.set_ticklabels([])
ax.set_xlabel('CLIENT_AGE')
ax.set_ylabel('CLIENT_MMM')
ax.set_zlabel('CLIENT_VRD_MOY')
ax.set_title('3D view of K-Means 6 clusters')
ax.dist = 12

plt.show()
```



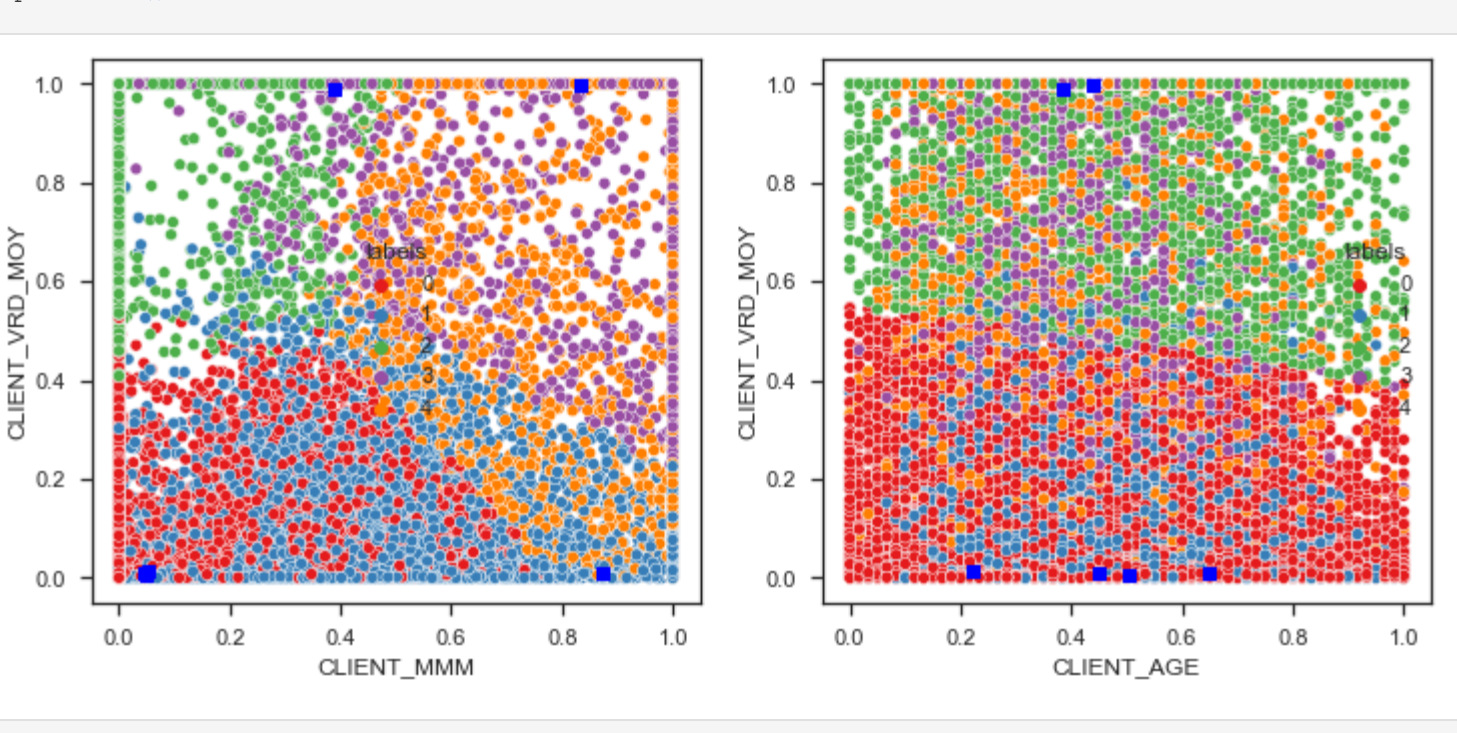
In [122]:
```python
import plotly as py
import plotly.graph_objs as go
def tracer(db, n, name):
    '''
    This function returns trace object for Plotly
    '''
    return go.Scatter3d(
        x = db[db['labels']==n]['CLIENT_AGE'],
        y = db[db['labels']==n]['CLIENT_VRD_MOY'],
        z = db[db['labels']==n]['CLIENT_MMM'],
        mode = 'markers',
        name = name,
        marker = dict(
            color = KM6_df['labels'],
            size = 5,
            line=dict(
            color= KM6_df['labels'],
            width= 12
        ),
        opacity=0.8
        )
     )

trace0 = tracer(KM6_df, 0, 'Cluster 0')
trace1 = tracer(KM6_df, 1, 'Cluster 1')
trace2 = tracer(KM6_df, 2, 'Cluster 2')
trace3 = tracer(KM6_df, 3, 'Cluster 3')
trace4 = tracer(KM6_df, 4, 'Cluster 4')
trace5 = tracer(KM6_df, 5, 'Cluster 5')
trace_data = [trace0, trace1, trace2, trace3, trace4, trace5]

layout = go.Layout(
    title = 'Clusters with k=6 wrt Age, Income and Spending Scores',
    scene = dict(
            xaxis = dict(title = 'Age'),
            yaxis = dict(title = 'Spending Score'),
            zaxis = dict(title = 'Annual Income')
        )
)

fig = go.Figure(data=trace_data, layout=layout)
py.offline.iplot(fig)
```

In [123]:
```python
db_index = davies_bouldin_score(X, labels6)
print("Davies Bouldin Index: %0.2f"%db_index)
```

```
Davies Bouldin Index: 0.92
```

In [124]:
```python
#For clusters of K=5
KM5 = (KMeans(n_clusters = 5 ,init='k-means++', n_init = 10 ,max_iter=300,
              tol=0.0001,  random_state= 111  , algorithm='elkan') )
KM5.fit(X)
labels5 = KM5.labels_
centroids5 = KM5.cluster_centers_
KM5_df = temp.copy()
KM5_df['labels'] = labels5
```

In [125]:
```python
fig1, (axes) = plt.subplots(1,2,figsize=(12,5))


scat_1 = sns.scatterplot('CLIENT_MMM', 'CLIENT_VRD_MOY', data=KM5_df,
                hue='labels', ax=axes[0], palette='Set1', legend='full')

sns.scatterplot('CLIENT_AGE', 'CLIENT_VRD_MOY', data=KM5_df,
                hue='labels', palette='Set1', ax=axes[1], legend='full')

axes[0].scatter(centroids6[:,1],centroids6[:,2], marker='s', s=40, c="blue")
axes[1].scatter(centroids6[:,0],centroids6[:,2], marker='s', s=40, c="blue")
plt.show()
```

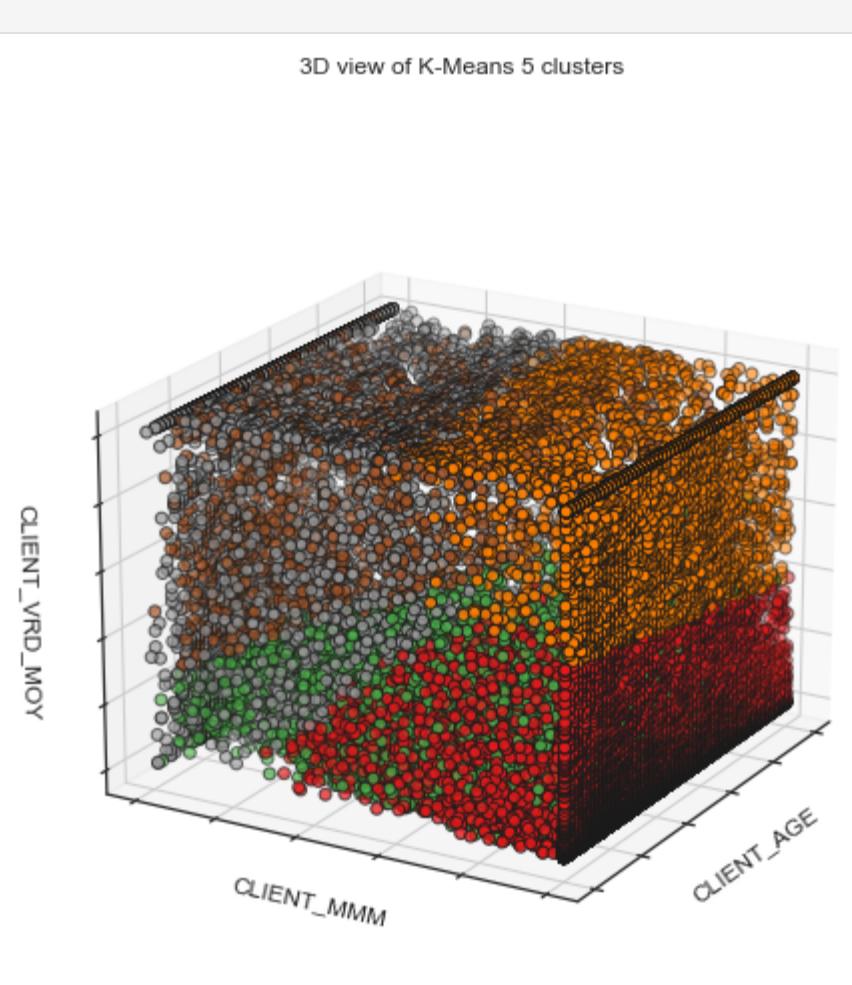


In [126]:
```python
KM_clust_sizes5 = KM5_df.groupby('labels').size().to_frame()
KM_clust_sizes5.columns = ["KM_size"]
KM_clust_sizes5
```

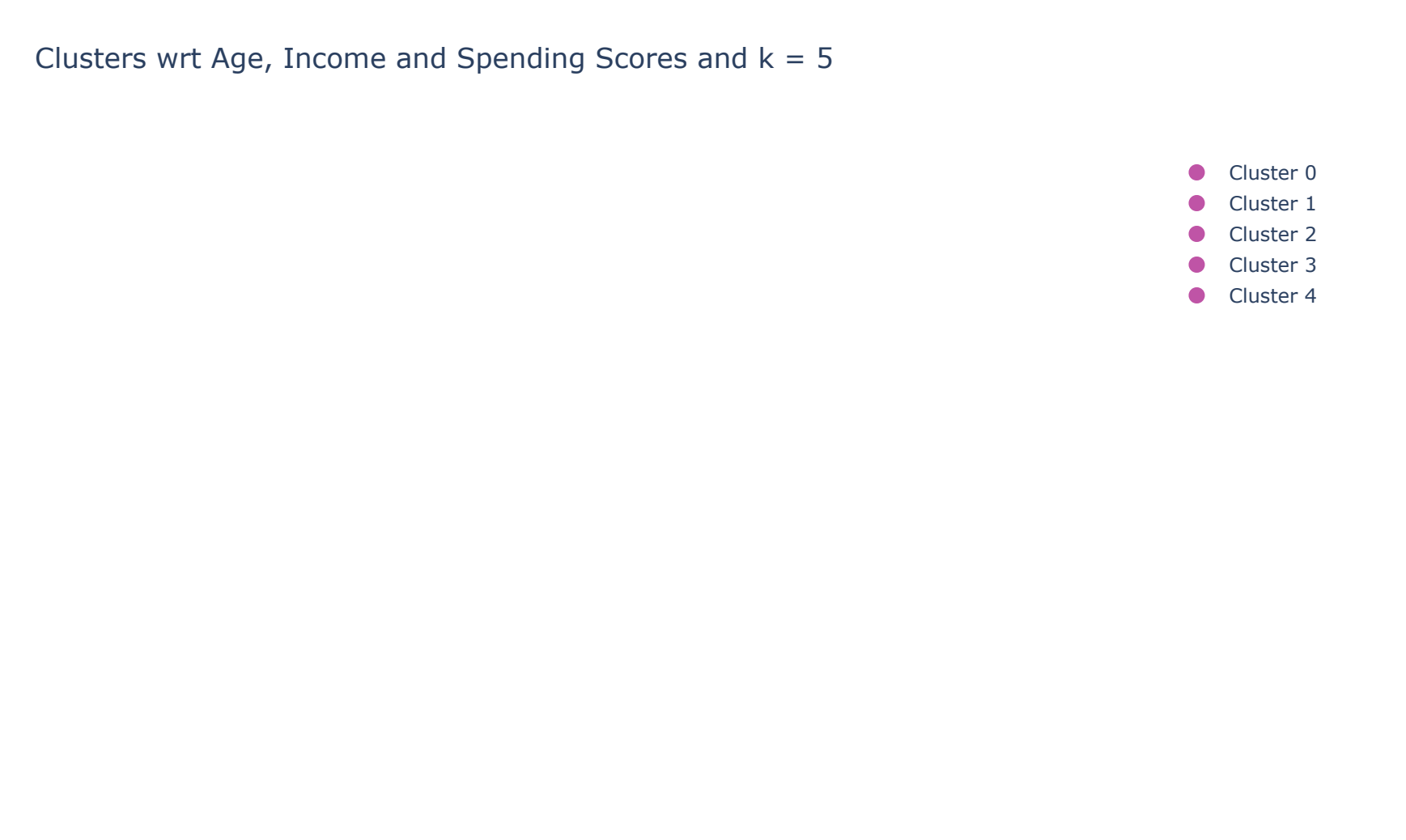
Out[126]:

labels	KM_size
0	27566
1	9484
2	5743
3	2763
4	4444

In [127]:
```python
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure(figsize=(7, 7))
ax = Axes3D(fig, rect=[0, 0, .99, 1], elev=20, azim=210)
ax.scatter(KM5_df['CLIENT_AGE'],
           KM5_df['CLIENT_MMM'],
           KM5_df['CLIENT_VRD_MOY'],
           c=KM5_df['labels'],
           s=35, edgecolor='k', cmap=plt.cm.Set1)

ax.w_xaxis.set_ticklabels([])
ax.w_yaxis.set_ticklabels([])
ax.w_zaxis.set_ticklabels([])
ax.set_xlabel('CLIENT_AGE')
ax.set_ylabel('CLIENT_MMM')
ax.set_zlabel('CLIENT_VRD_MOY')
ax.set_title('3D view of K-Means 5 clusters')
ax.dist = 12

plt.show()
```



In [128]:
```python
trace0 = tracer(KM5_df, 0, 'Cluster 0')
trace1 = tracer(KM5_df, 1, 'Cluster 1')
trace2 = tracer(KM5_df, 2, 'Cluster 2')
trace3 = tracer(KM5_df, 3, 'Cluster 3')
trace4 = tracer(KM5_df, 4, 'Cluster 4')

trace_data5 = [trace0, trace1, trace2, trace3, trace4]

layout = go.Layout(
    title = 'Clusters wrt Age, Income and Spending Scores and k = 5',
    scene = dict(
            xaxis = dict(title = 'Age'),
            yaxis = dict(title = 'Spending Score'),
            zaxis = dict(title = 'Annual Income')
        )
)

fig = go.Figure(data=trace_data5, layout=layout)
py.offline.iplot(fig)
```



Evaluation metrics

In [129]:
```python
db_index = davies_bouldin_score(X, labels6)
print("Davies Bouldin Index: %0.2f"%db_index)
sc = metrics.silhouette_score(X, labels6)
print("Silhouette Coefficient:%0.2f"%sc)
ch_score = calinski_harabasz_score(X, labels6)
print("Calinski harabasz score:%0.2f"%ch_score)
```

```
Davies Bouldin Index: 0.92
Silhouette Coefficient:0.42
Calinski harabasz score:40324.80
```

In [130…

```
db_index = davies_bouldin_score(X, labels5)
print("Davies Bouldin Index: %0.2f"%db_index)
sc = metrics.silhouette_score(X, labels5)
print("Silhouette Coefficient:%0.2f"%sc)
ch_score = calinski_harabasz_score(X, labels5)
print("Calinski harabasz score:%0.2f"%ch_score)
```

```
Davies Bouldin Index: 0.84
Silhouette Coefficient:0.53
Calinski harabasz score:38863.35
```

Clearly, the metrics show that using 5 clusters is more optimal

In [131…

```
# calculate SC for K=2 through K=19
scores = []
for k in range(2,20):
    labels = KMeans(n_clusters=k).fit(X).labels_
    score = metrics.silhouette_score(X, labels)
    scores.append(score)

scores
```

Out[131…

```
[0.46954068092765716,
 0.5138604991582205,
 0.5122099113572987,
 0.5262700225728112,
 0.42040109487662564,
 0.4358438863341281,
 0.41247688852935416,
 0.39600510845887715,
 0.3848489676446466,
 0.36927239836625175,
 0.3607123706180566,
 0.37087367993795056,
 0.3491170194419902,
 0.37117832087908975,
 0.3432624683830541,
 0.3542450335211789,
 0.35624165493993437,
 0.35950497934420367]
```

The highest silhouette score is for k=5 (0.5217648399961166)

Improve kmeans with PCA

In [132…

```
# This time the client's profession feature will be added
x_pca = temp[['CLIENT_AGE' ,'CLIENT_MMM' ,'CLIENT_ENCOURS_ENGAGEMENT','CLIENT_VRD_MOY','TOTAL_PACK','CLIENT_PRO
pcamodel = PCA(n_components=0.85)
pca = pcamodel.fit_transform(x_pca)
pca.shape
```

Out[132…

```
(50000, 4)
```

the algorithm automatically selects the best number of principal components that keep 85% of the variance in the original data.

In [133…

```
# know how many components that the algorithm has selected
pcamodel.n_components_
```

Out[133…

```
4
```

In [134…

```
pcamodel.explained_variance_
```

Out[134…

```
array([0.24137679, 0.16263663, 0.07410932, 0.05622784])
```

pca explains the variance ratio of 85% with n_component: 4

In [135…

```
pcamodel.explained_variance_ratio_
```

Out[135…

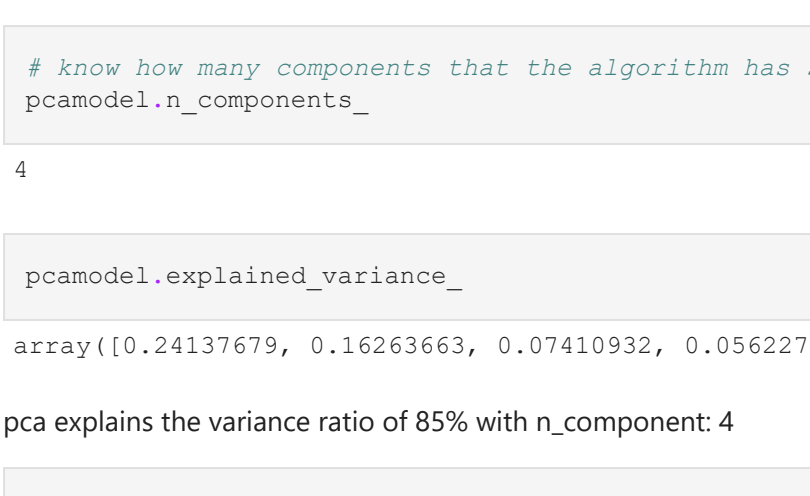
```
array([0.40539324, 0.27314883, 0.12446688, 0.09443486])
```

In [136…

```
# Bar plot of explained_variance
plt.bar(range(1,len(pcamodel.explained_variance_ )+1),pcamodel.explained_variance_ )
plt.ylabel('Explained variance')
plt.xlabel('Components')
plt.plot(range(1,len(pcamodel.explained_variance_ )+1),
         np.cumsum(pcamodel.explained_variance_),
         c='red',
         label="Cumulative Explained Variance")
plt.legend(loc='upper left')
```

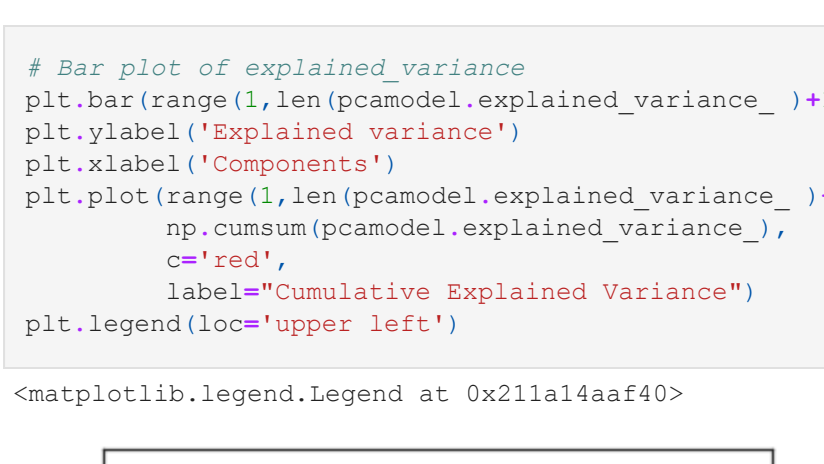
Out[136…

```
<matplotlib.legend.Legend at 0x211a14aaf40>
```



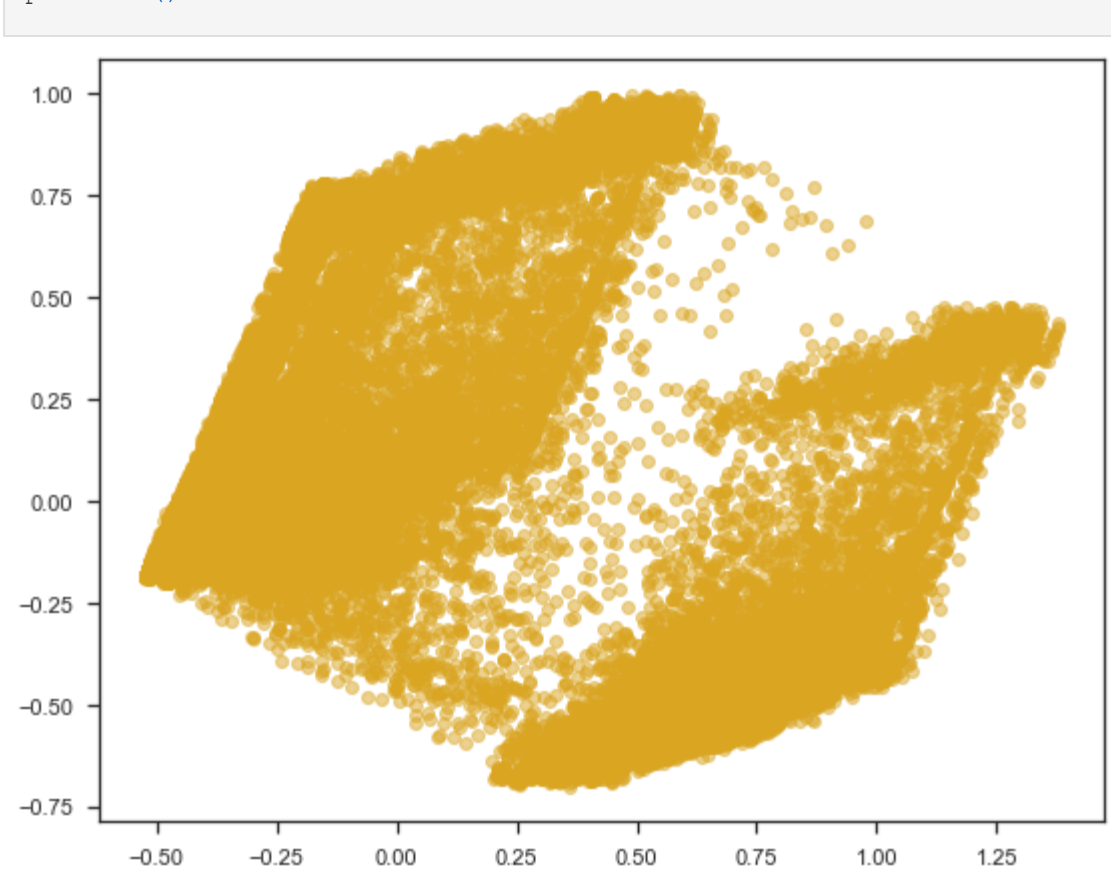
In [137…

```
plt.plot(pcamodel.explained_variance_ratio_)
plt.xlabel('number of components')
plt.ylabel('cumulative explained variance')
plt.show()

#PCA1 is at 0 in xscale
```



In [138…

```
plt.figure(figsize = (9,7))
plt.scatter(pca[:, 0], pca[:, 1], c='goldenrod',alpha=0.5)
plt.show()
```



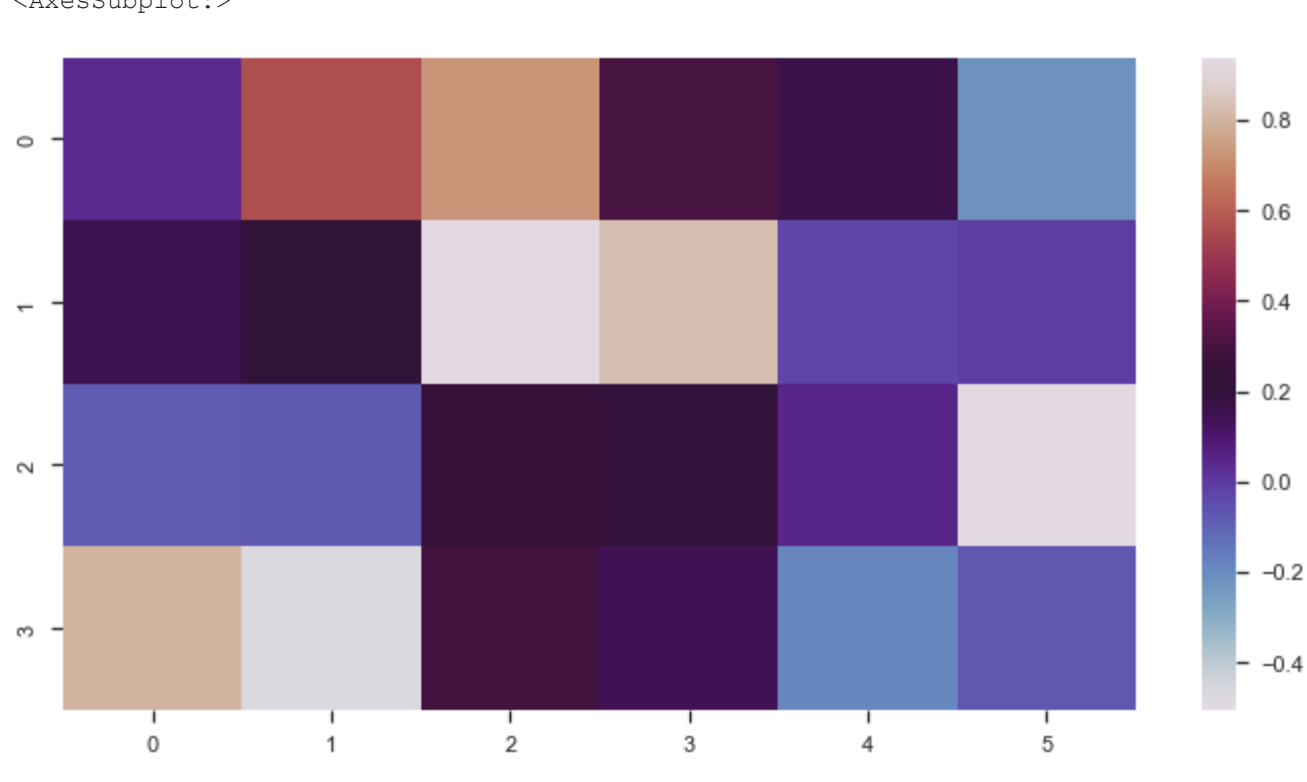
A rule of thumb is to preserve around 80 % of the variance. So, in this instance, we decide to keep 3 components.

In [139…

```
map= pd.DataFrame(pcamodel.components_)
plt.figure(figsize=(12,6))
sns.heatmap(map,cmap='twilight')
```

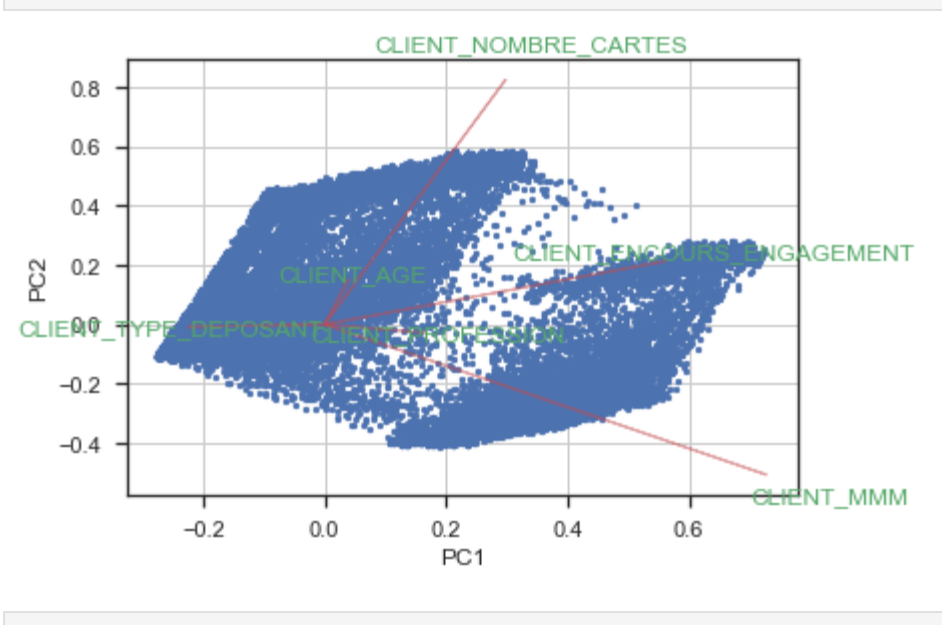
Out[139…

```
<AxesSubplot:>
```



In [140…

```
def myplot(score,coeff,labels=None):
    xs = score[:,0]
    ys = score[:,1]
    n = coeff.shape[0]
    scalex = 1.0/(xs.max() - xs.min())
    scaley = 1.0/(ys.max() - ys.min())
    plt.scatter(xs * scalex,ys * scaley,s=5)
    for i in range(n):
        plt.arrow(0, 0, coeff[i,0], coeff[i,1],color = 'r',alpha = 0.5)
        if labels is None:
            plt.text(coeff[i,0]* 1.15, coeff[i,1] * 1.15, "Var"+str(i+1), color = 'green', ha = 'center', va =
        else:
            plt.text(coeff[i,0]* 1.15, coeff[i,1] * 1.15, labels[i], color = 'g', ha = 'center', va = 'center')

    plt.xlabel("PC{}".format(1))
    plt.ylabel("PC{}".format(2))
    plt.grid()

myplot(pca[:,0:2],np.transpose(pcamodel.components_[0:2, :]),list(data.columns))
plt.show()
```



In [141…

```
pcamodel.get_params()
```

Out[141…

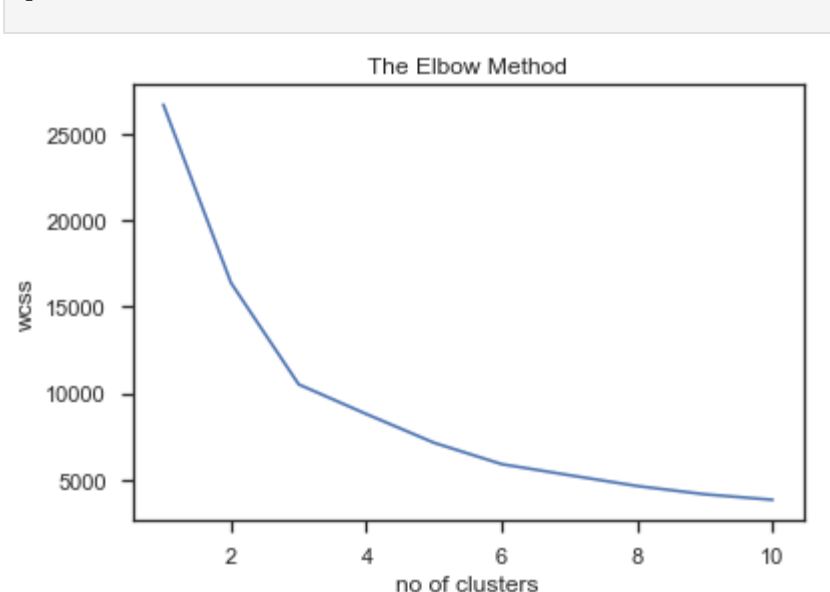
```
{'copy': True,
 'iterated_power': 'auto',
 'n_components': 0.85,
 'n_oversamples': 10,
 'power_iteration_normalizer': 'auto',
 'random_state': None,
 'svd_solver': 'auto',
 'tol': 0.0,
 'whiten': False}
```

In [142…

```
wcss=[]
for i in range(1,11):
    kmeans_pca = KMeans(n_clusters= i, init='k-means++', random_state=0)
    kmeans_pca.fit(pca)
    wcss.append(kmeans_pca.inertia_)
```

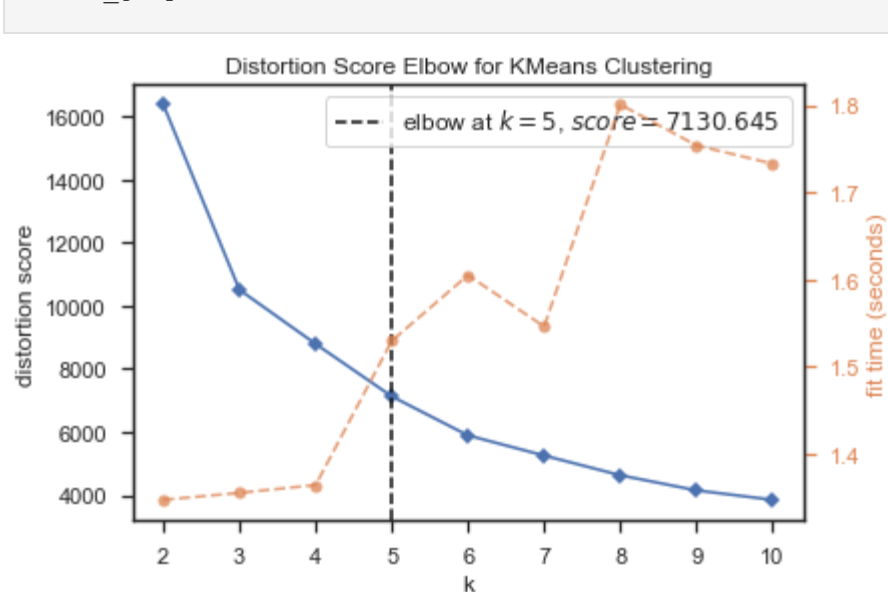
In [143…

```
plt.plot(range(1,11), wcss)
plt.title('The Elbow Method')
plt.xlabel('no of clusters')
plt.ylabel('wcss')
plt.show()
```



In [144…

```
elbow_graph = KElbowVisualizer(KMeans(random_state=43), k=10)
elbow_graph.fit(pca)
elbow_graph.show()
```



Out[144…

```
<AxesSubplot:title={'center':'Distortion Score Elbow for KMeans Clustering'}, xlabel='k', ylabel='distortion score'>
```

In [145…

```
kmeans_pca = KMeans(n_clusters=4, init='k-means++', random_state=42)
```

In [146…

```
label_pca = kmeans_pca.fit_predict(pca)
```

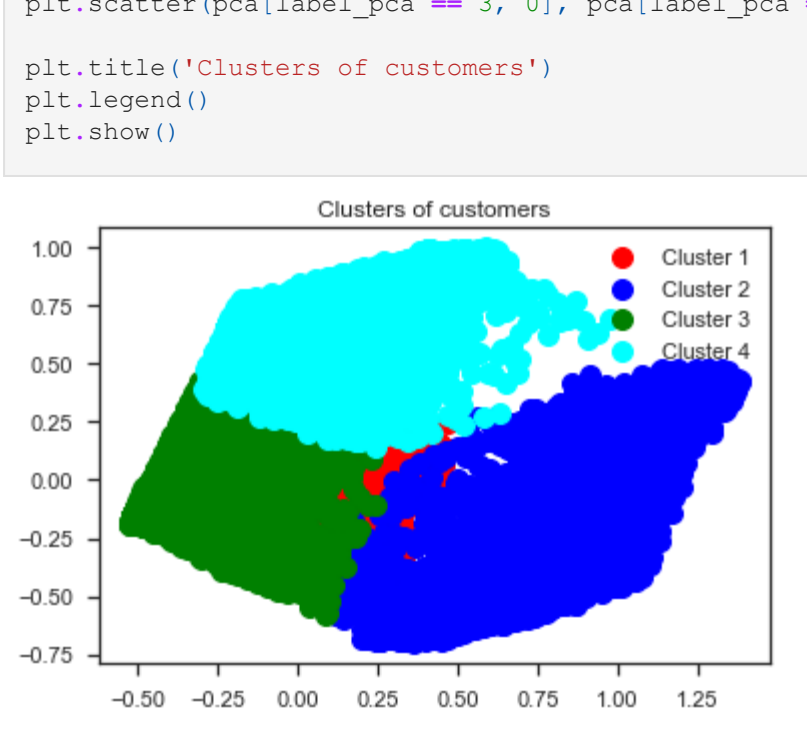
In [147…

```
db_index = davies_bouldin_score(pca, label_pca)
print("Davies Bouldin Index: %0.2f"%db_index)
sc = metrics.silhouette_score(pca, label_pca)
print("Silhouette Coefficient:%0.2f"%sc)
ch_score = calinski_harabasz_score(pca, label_pca)
print("Calinski harabasz score:%0.2f"%ch_score)
```

```
Davies Bouldin Index: 0.96
Silhouette Coefficient:0.38
Calinski harabasz score:33961.70
```

In [148…

```
plt.scatter(pca[label_pca == 0, 0], pca[label_pca == 0, 1], s = 100, c = 'red', label = 'Cluster 1')
plt.scatter(pca[label_pca == 1, 0], pca[label_pca == 1, 1], s = 100, c = 'blue', label = 'Cluster 2')
plt.scatter(pca[label_pca == 2, 0], pca[label_pca == 2, 1], s = 100, c = 'green', label = 'Cluster 3')
plt.scatter(pca[label_pca == 3, 0], pca[label_pca == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')

plt.title('Clusters of customers')
plt.legend()
plt.show()
```



In [149…

```
kmeans_pca = KMeans(n_clusters=3, init='k-means++', random_state=42)
```

In [150…

```
label_pca = kmeans_pca.fit_predict(pca)
```

In [151…

```
db_index = davies_bouldin_score(pca, label_pca)
print("Davies Bouldin Index: %0.2f"%db_index)
sc = metrics.silhouette_score(pca, label_pca)
print("Silhouette Coefficient:%0.2f"%sc)
ch_score = calinski_harabasz_score(pca, label_pca)
print("Calinski harabasz score:%0.2f"%ch_score)
```

```
Davies Bouldin Index: 0.92
Silhouette Coefficient:0.47
Calinski harabasz score:38534.26
```

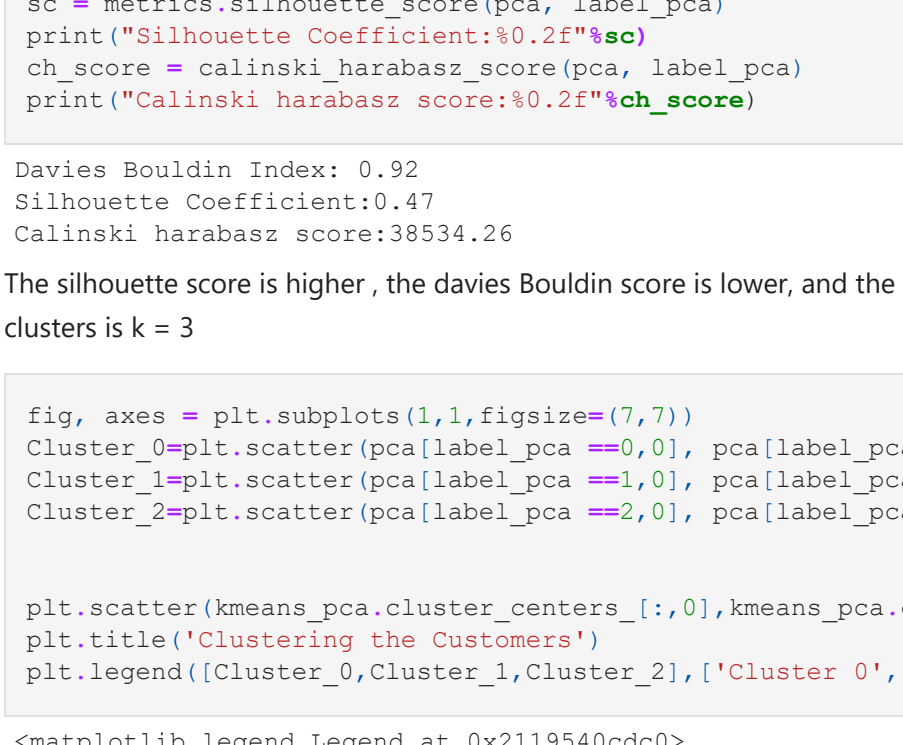
The silhouette score is higher , the davies Bouldin score is lower, and the calinski harabasz is higher so the chosen value for the number of clusters is k = 3

In [155…

```
fig, axes = plt.subplots(1,1,figsize=(7,7))
Cluster_0=plt.scatter(pca[label_pca ==0,0], pca[label_pca == 0,1], s=100, c='#63b3c0')
Cluster_1=plt.scatter(pca[label_pca ==1,0], pca[label_pca == 1,1], s=100, c='#006494')
Cluster_2=plt.scatter(pca[label_pca ==2,0], pca[label_pca == 2,1], s=100, c='#293f4e')


plt.scatter(kmeans_pca.cluster_centers_[:,0],kmeans_pca.cluster_centers_[:,1],color='orange',marker='*',label='
plt.title('Clustering the Customers')
plt.legend([Cluster_0,Cluster_1,Cluster_2],['Cluster 0','Cluster 1','Cluster 2'])
```

Out[155…

```
<matplotlib.legend.Legend at 0x2119540cdc0>
```



In [156…

```
data_pca = pd.concat([temp.reset_index(drop=True),pd.DataFrame(pca)],axis=1)
data_pca.columns.values[-3:]=['Component 1','Component 2','Component 3']
data_pca['SEGMENT_PCA'] = kmeans_pca.labels_
```

In [157…

```
data_pca
```

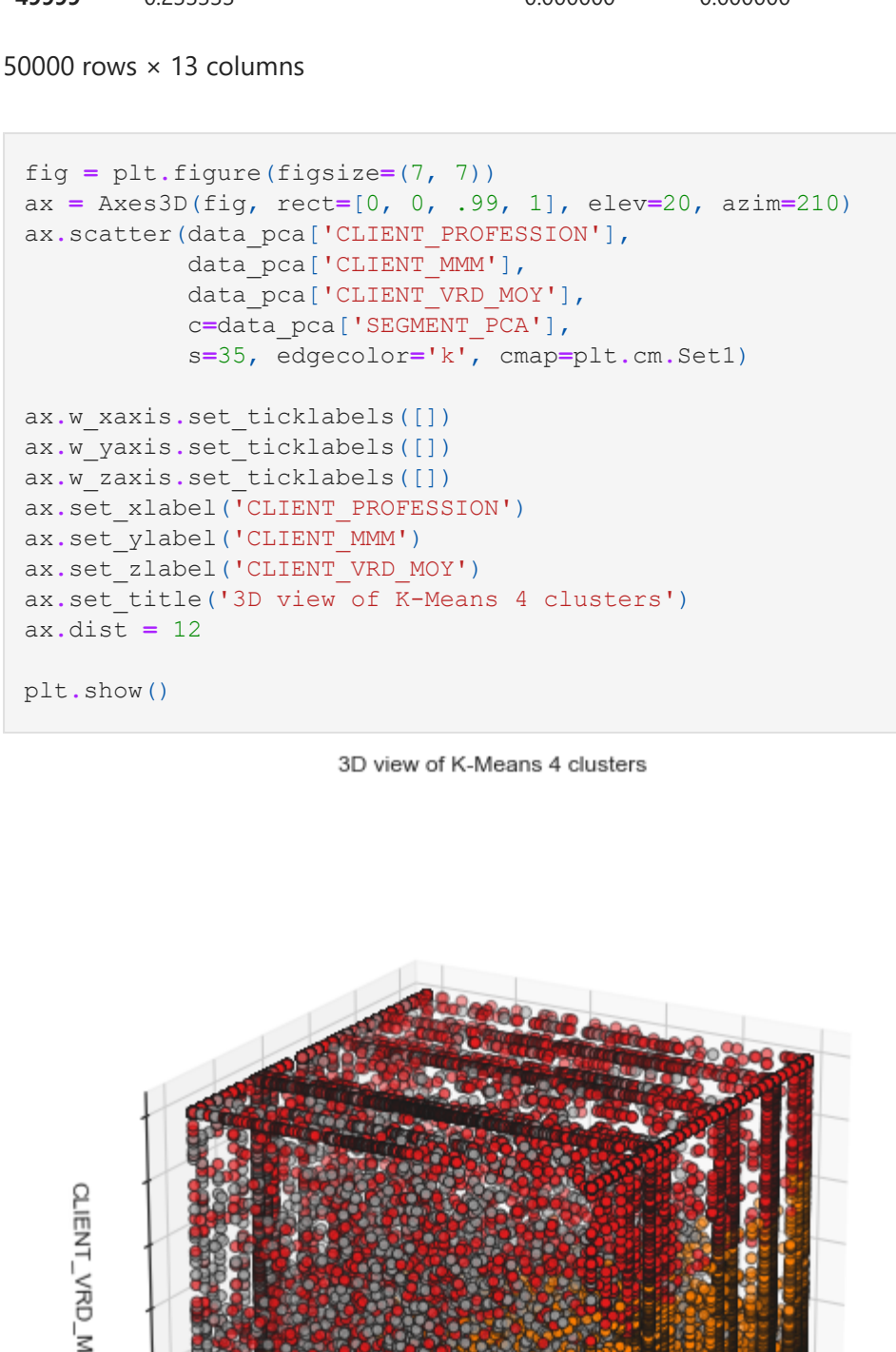
Out[157…

	CLIENT_AGE	CLIENT_ENCOURS_ENGAGEMENT	CLIENT_MMM	CLIENT_NOMBRE_CARTES	CLIENT_PROFESSION	CLIENT_TYPE_DEPOSANT
0	0.633333	0.000000	1.000000	0.6	0.258065	0.00
1	0.433333	0.145041	0.156169	0.2	0.645161	0.00
2	0.933333	0.000000	0.453050	0.2	0.000000	0.00
3	0.500000	0.000000	0.000000	0.0	0.580645	0.25
4	0.500000	1.000000	1.000000	0.4	0.258065	0.00
...	...	...	...	...	...	...
49995	0.466667	0.000000	0.000000	0.0	0.064516	0.25
49996	0.200000	0.000000	0.000000	0.2	0.645161	0.25
49997	0.366667	0.000000	0.000000	0.2	0.645161	0.25
49998	0.133333	0.000000	0.000000	0.2	0.645161	0.25
49999	0.233333	0.000000	0.000000	0.0	0.967742	0.25

50000 rows × 13 columns

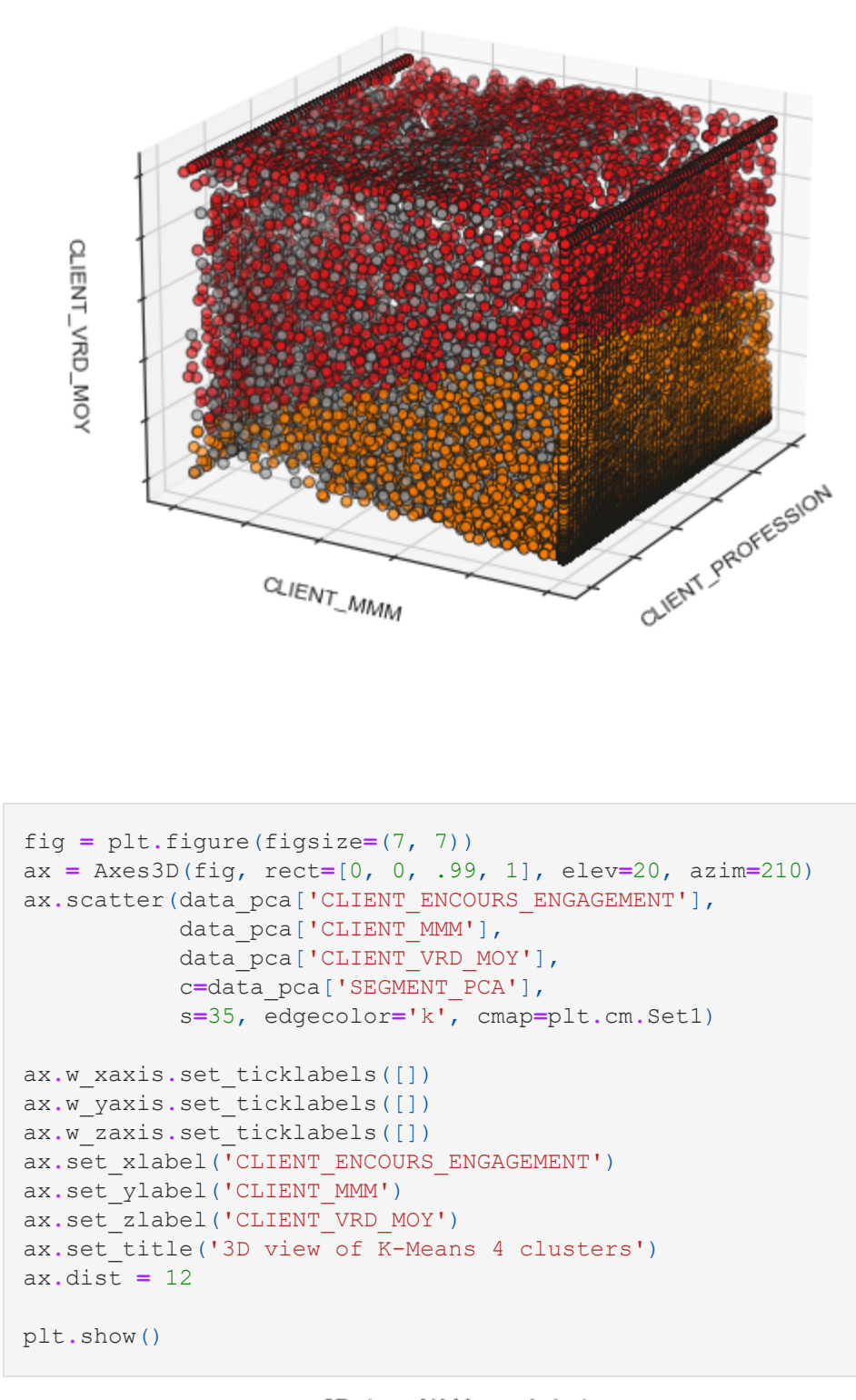
In [158…

```
fig = plt.figure(figsize=(7, 7))
ax = Axes3D(fig, rect=[0, 0, .99, 1], elev=20, azim=210)
ax.scatter(data_pca['CLIENT_PROFESSION'],
           data_pca['CLIENT_MMM'],
           data_pca['CLIENT_VRD_MOY'],
           c=data_pca['SEGMENT_PCA'],
           s=35, edgecolor='k', cmap=plt.cm.Set1)

ax.w_xaxis.set_ticklabels([])
ax.w_yaxis.set_ticklabels([])
ax.w_zaxis.set_ticklabels([])
ax.set_xlabel('CLIENT_PROFESSION')
ax.set_ylabel('CLIENT_MMM')
ax.set_zlabel('CLIENT_VRD_MOY')
ax.set_title('3D view of K-Means 4 clusters')
ax.dist = 12

plt.show()
```



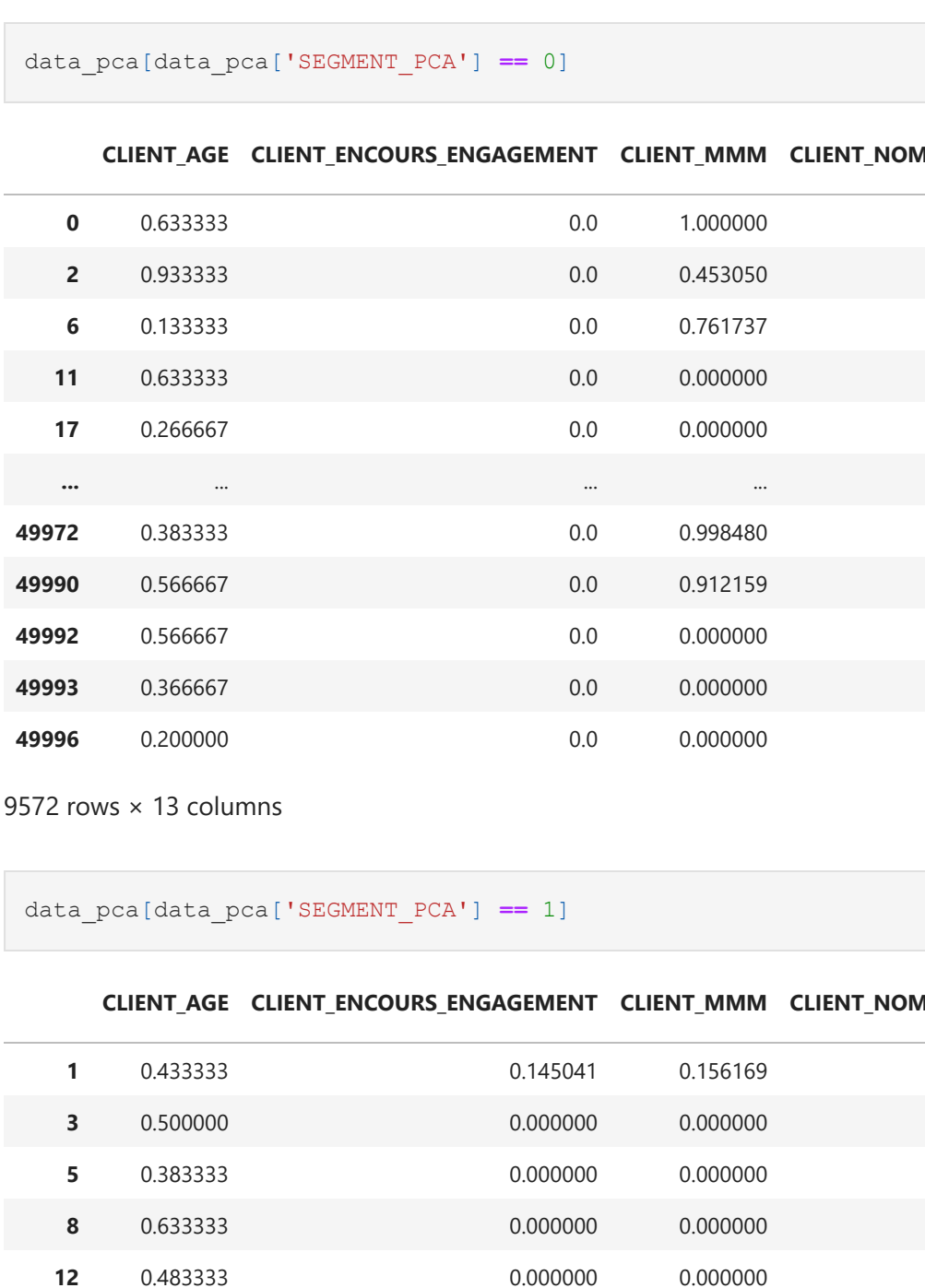
In [159…

```
fig = plt.figure(figsize=(7, 7))
ax = Axes3D(fig, rect=[0, 0, .99, 1], elev=20, azim=210)
ax.scatter(data_pca['CLIENT_AGE'],
           data_pca['CLIENT_MMM'],
           data_pca['CLIENT_VRD_MOY'],
           c=data_pca['SEGMENT_PCA'],
           s=35, edgecolor='k', cmap=plt.cm.Set1)

ax.w_xaxis.set_ticklabels([])
ax.w_yaxis.set_ticklabels([])
ax.w_zaxis.set_ticklabels([])
ax.set_xlabel('CLIENT_PROFESSION')
ax.set_ylabel('CLIENT_MMM')
ax.set_zlabel('CLIENT_VRD_MOY')
ax.set_title('3D view of K-Means 3 clusters')
ax.dist = 12

plt.show()
```



In [160…

```
fig = plt.figure(figsize=(7, 7))
ax = Axes3D(fig, rect=[0, 0, .99, 1], elev=20, azim=210)
ax.scatter(data_pca['CLIENT_ENCOURS_ENGAGEMENT'],
           data_pca['CLIENT_MMM'],
           data_pca['CLIENT_VRD_MOY'],
           c=data_pca['SEGMENT_PCA'],
           s=35, edgecolor='k', cmap=plt.cm.Set1)

ax.w_xaxis.set_ticklabels([])
ax.w_yaxis.set_ticklabels([])
ax.w_zaxis.set_ticklabels([])
ax.set_xlabel('CLIENT_ENCOURS_ENGAGEMENT')
ax.set_ylabel('CLIENT_MMM')
ax.set_zlabel('CLIENT_VRD_MOY')
ax.set_title('3D view of K-Means 4 clusters')
ax.dist = 12

plt.show()
```



In [161…

```
data_pca[data_pca['SEGMENT_PCA'] == 0]
```

Out[161…

	CLIENT_AGE	CLIENT_ENCOURS_ENGAGEMENT	CLIENT_MMM	CLIENT_NOMBRE_CARTES	CLIENT_PROFESSION	CLIENT_TYPE_DEPOSANT
0	0.633333	0.0	1.000000	0.6	0.258065	0.00
2	0.933333	0.0	0.453050	0.2	0.000000	0.00
6	0.133333	0.0	0.761737	0.6	0.258065	0.00
11	0.633333	0.0	0.000000	0.0	0.838710	0.25
17	0.266667	0.0	0.000000	0.0	0.838710	0.25
...	...	...	...	...	...	...
49972	0.383333	0.0	0.998480	0.6	0.258065	0.00
49990	0.566667	0.0	0.912159	0.2	0.000000	0.00
49992	0.566667	0.0	0.000000	0.0	0.838710	0.25
49993	0.366667	0.0	0.000000	0.2	0.645161	0.25
49996	0.200000	0.0	0.000000	0.2	0.645161	0.25

9572 rows × 13 columns

In [162…

```
data_pca[data_pca['SEGMENT_PCA'] == 1]
```

Out[162…

	CLIENT_AGE	CLIENT_ENCOURS_ENGAGEMENT	CLIENT_MMM	CLIENT_NOMBRE_CARTES	CLIENT_PROFESSION	CLIENT_TYPE_DEPOSANT
1	0.433333	0.145041	0.156169	0.2	0.645161	0.00
3	0.500000	0.000000	0.000000	0.0	0.580645	0.25
5	0.383333	0.000000	0.000000	0.0	0.387097	0.25
8	0.633333	0.000000	0.000000	0.0	0.193548	0.25
12	0.483333	0.000000	0.000000	0.0	0.580645	0.25
...	...	...	...	...	...	...
49994	0.116667	0.168329	0.215285	0.4	0.645161	0.00
49995	0.466667	0.000000	0.000000	0.0	0.064516	0.25
49997	0.366667	0.000000	0.000000	0.2	0.645161	0.25
49998	0.133333	0.000000	0.000000	0.2	0.645161	0.25
49999	0.233333	0.000000	0.000000	0.0	0.967742	0.25

28176 rows × 13 columns

In [163…

```
data_pca[data_pca['SEGMENT_PCA'] == 2]
```

Out[163…

	CLIENT_AGE	CLIENT_ENCOURS_ENGAGEMENT	CLIENT_MMM	CLIENT_NOMBRE_CARTES	CLIENT_PROFESSION	CLIENT_TYPE_DEPOSANT
4	0.500000	1.0	1.000000	0.4	0.258065	0.0
7	0.400000	1.0	0.481134	0.4	0.645161	0.0
9	0.266667	1.0	0.262139	0.0	0.000000	0.0
10	0.566667	1.0	0.190365	0.6	0.645161	0.0
13	0.366667	1.0	0.000000	0.6	0.161290	0.0
...	...	...	...	...	...	...
49971	0.100000	1.0	0.363748	0.0	0.645161	0.0
49973	0.383333	1.0	1.000000	0.8	0.258065	0.0
49974	0.416667	1.0	0.288073	0.0	0.580645	0.0
49982	0.366667	1.0	1.000000	0.6	0.290323	0.0
49983	0.433333	1.0	1.000000	0.0	0.451613	0.0

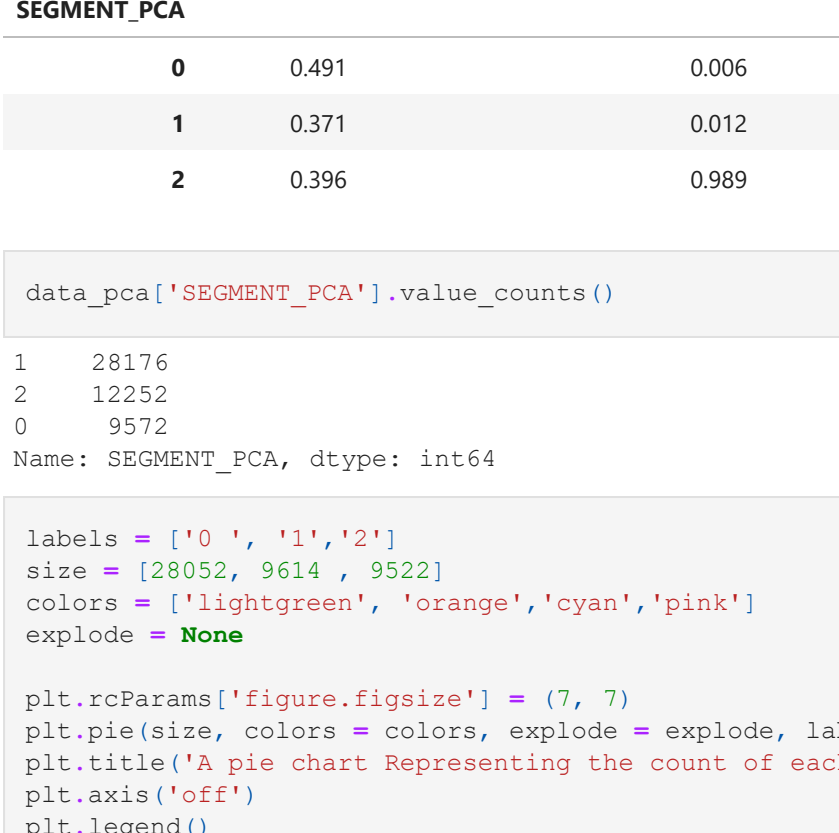
12252 rows × 13 columns

In [164…

```
sns.countplot(label_pca)
```

Out[164…

```
<AxesSubplot:ylabel='count'>
```



In [165…

```
data_pca.groupby('SEGMENT_PCA').mean().round(3)
```

Out[165…

	CLIENT_AGE	CLIENT_ENCOURS_ENGAGEMENT	CLIENT_MMM	CLIENT_NOMBRE_CARTES	CLIENT_PROFESSION	CLIENT_TYPE_DEPO
SEGMENT_PCA						
0	0.491	0.006	0.405	0.183	0.524	
1	0.371	0.012	0.053	0.097	0.596	
2	0.396	0.989	0.492	0.349	0.445	

In [166…

```
data_pca['SEGMENT_PCA'].value_counts()
```

Out[166…

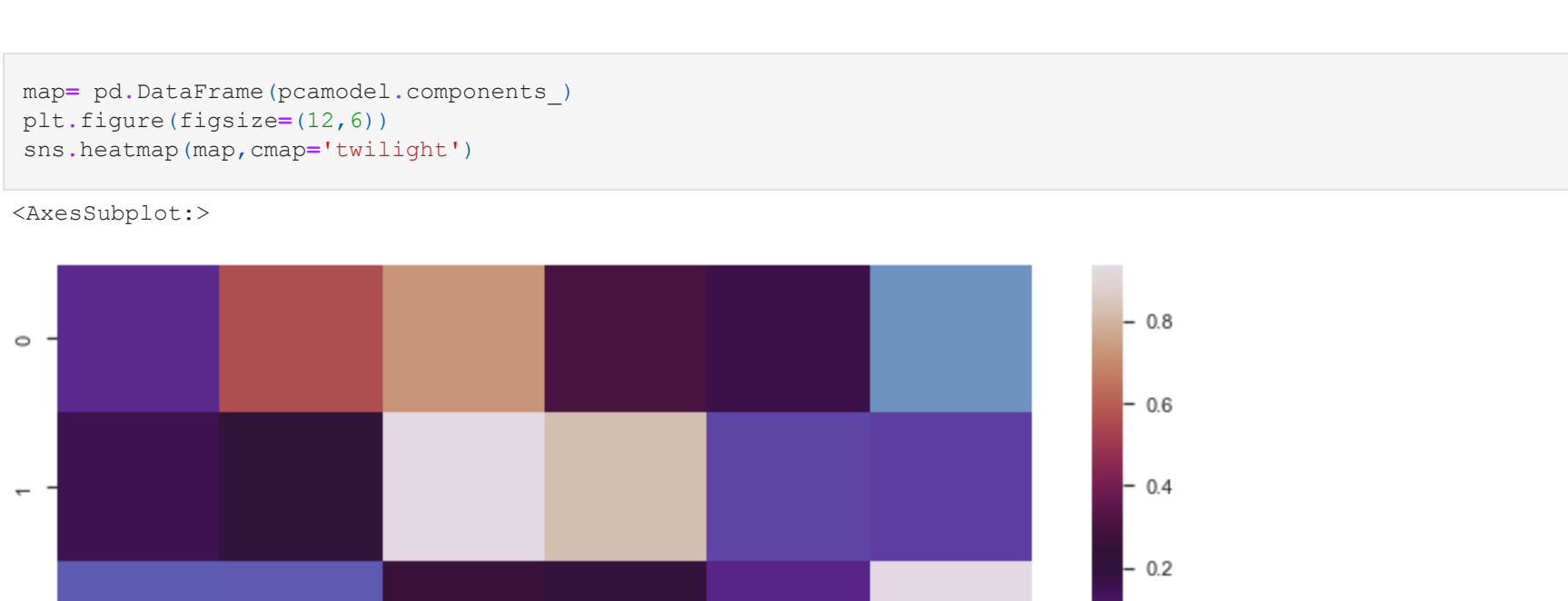
```
1    28176
2    12252
0     9572
Name: SEGMENT_PCA, dtype: int64
```

In [167…

```
labels = ['0 ', '1','2']
size = [28052, 9614 , 9522]
colors = ['lightgreen', 'orange','cyan','pink']
explode = None

plt.rcParams['figure.figsize'] = (7, 7)
plt.pie(size, colors = colors, explode = explode, labels = labels, shadow = True, autopct = '%.2f%%')
plt.title('A pie chart Representing the count of each segment')
plt.axis('off')
plt.legend()
plt.show()
```


In [168…

```
fig=px.scatter_3d(data_frame=x_pca,x='CLIENT_AGE',y='CLIENT_MMM',z='CLIENT_VRD_MOY',color=label_pca, color_cont
fig.update_layout(
    title={
        'text': 'Customer Segmentation Based on Age, Average monthly flow and Actual values of deposits',
        'y':0.95,
        'x':0.5,
        'xanchor': 'center',
        'yanchor': 'top'})
fig.show()
```



In [169…

```
map= pd.DataFrame(pcamodel.components_)
plt.figure(figsize=(12,6))
sns.heatmap(map,cmap='twilight')
```

Out[169…

```
<AxesSubplot:>
```

In [170…

```
bins= [30,40,50,70,80]
labels=['Young Adult','Adult','Middle Aged','Senior Citizen']
data_pca['AgeGroup'] = pd.cut(data['CLIENT_AGE'], bins=bins, labels=labels, right=False)
print (data_pca)
```

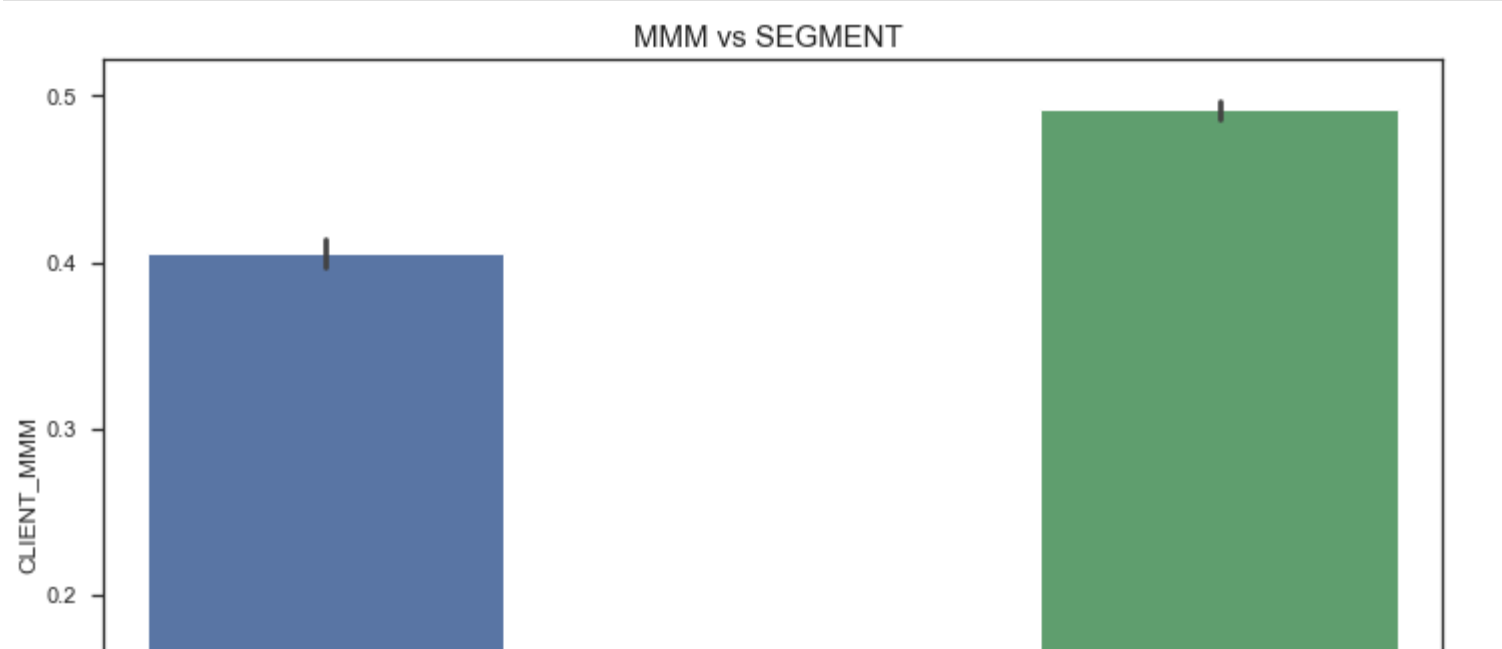
```
       CLIENT_AGE  CLIENT_ENCOURS_ENGAGEMENT  CLIENT_MMM  \
0        0.633333                   0.000000    1.000000
1        0.433333                   0.145041    0.156169
2        0.933333                   0.000000    0.453050
3        0.500000                   0.000000    0.000000
4        0.500000                   1.000000    1.000000
...           ...                        ...         ...
49995    0.466667                   0.000000    0.000000
49996    0.200000                   0.000000    0.000000
49997    0.366667                   0.000000    0.000000
49998    0.133333                   0.000000    0.000000
49999    0.233333                   0.000000    0.000000

       CLIENT_NOMBRE_CARTES  CLIENT_PROFESSION  CLIENT_TYPE_DEPOSANT  \
0                       0.6           0.258065                  0.00
1                       0.2           0.645161                  0.00
2                       0.2           0.000000                  0.00
3                       0.0           0.580645                  0.25
4                       0.4           0.258065                  0.00
...                     ...                ...                   ...
49995                   0.0           0.064516                  0.25
49996                   0.2           0.645161                  0.25
49997                   0.2           0.645161                  0.25
49998                   0.2           0.645161                  0.25
49999                   0.0           0.967742                  0.25

       CLIENT_VRD_MOY  TOTAL_PACK         0  Component 1  Component 2  \
0            0.974288        0.25  0.550564     0.914327    -0.260721
1            0.014642        0.25 -0.193248    -0.160433     0.025819
2            1.000000        0.25  0.321793     0.866200    -0.480355
3            0.003033        0.00 -0.412280    -0.112597    -0.080160
4            0.025342        0.25  0.987932    -0.391208    -0.185449
...               ...         ...       ...          ...          ...
49995        0.029838        0.00 -0.291437    -0.091596    -0.556527
49996        0.905150        0.25 -0.130126     0.577280     0.197155
49997        0.010298        0.00 -0.429549    -0.127069    -0.007747
49998        0.000143        0.00 -0.441580    -0.170346     0.008444
49999        0.000553        0.00 -0.508954    -0.157504     0.303429

       Component 3  SEGMENT_PCA     AgeGroup
0        -0.145008            0  Middle Aged
1        -0.029576            1  Middle Aged
2         0.373832            0  Middle Aged
3         0.105969            1  Middle Aged
4        -0.104551            2  Middle Aged
...            ...          ...          ...
49995     0.118588            1  Middle Aged
49996    -0.055443            0          NaN
49997    -0.003813            1  Middle Aged
49998    -0.191473            1  Middle Aged
49999    -0.133723            1        Adult

[50000 rows x 14 columns]
```

In [171…

```python
plt.figure(figsize=(12, 8))
sns.countplot(x='SEGMENT_PCA', data=data_pca, hue='AgeGroup')
plt.show()
```

In [172…

```python
plt.figure(figsize=(12, 8))
sns.barplot(x=data_pca["SEGMENT_PCA"], y=data_pca["CLIENT_MMM"])
plt.title("MMM vs SEGMENT ", size=15)
plt.show()
```

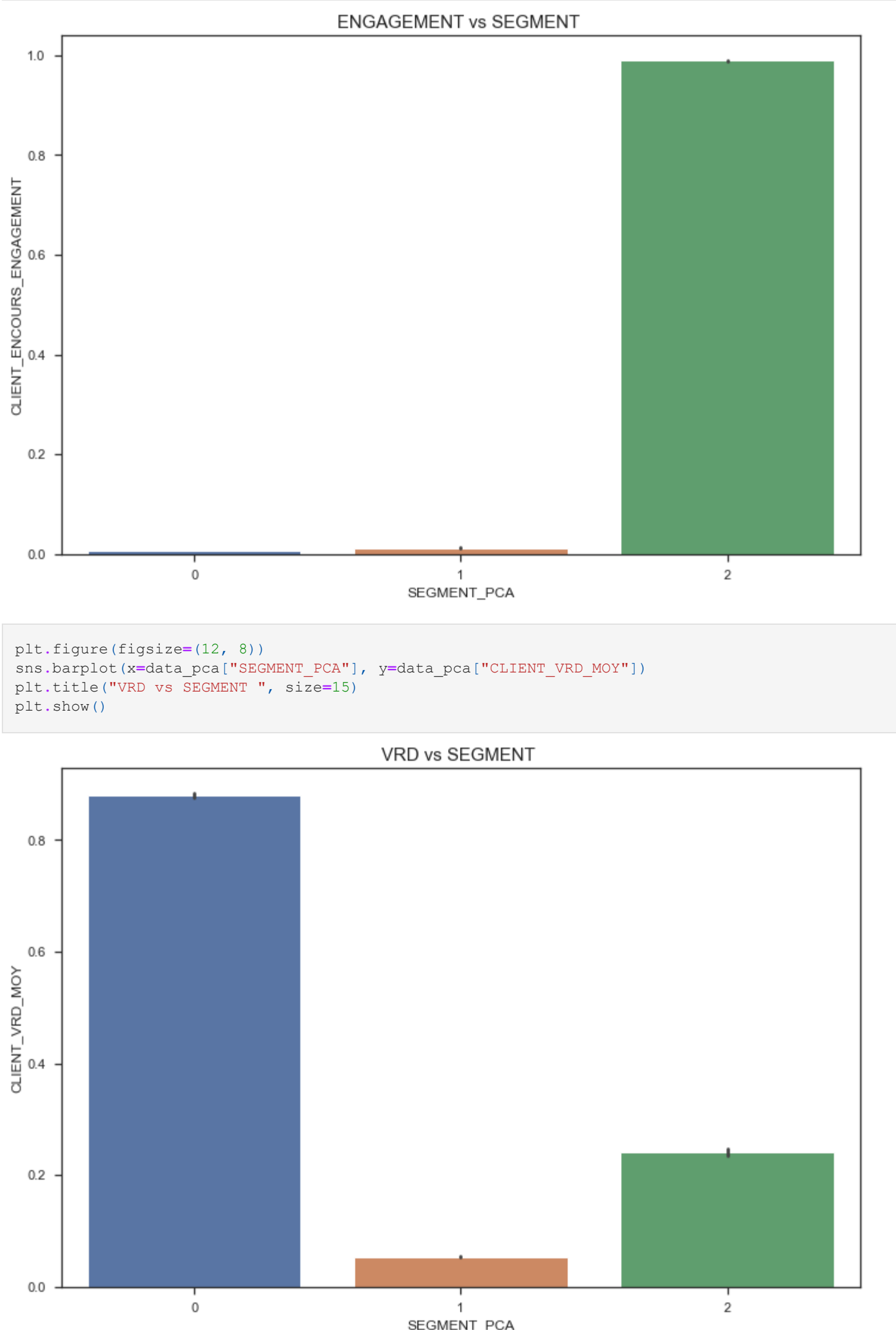


In [173…

```python
plt.figure(figsize=(12, 8))
sns.barplot(x=data_pca["SEGMENT_PCA"], y=data_pca["CLIENT_ENCOURS_ENGAGEMENT"])
plt.title("ENGAGEMENT vs SEGMENT ", size=15)
plt.show()
```

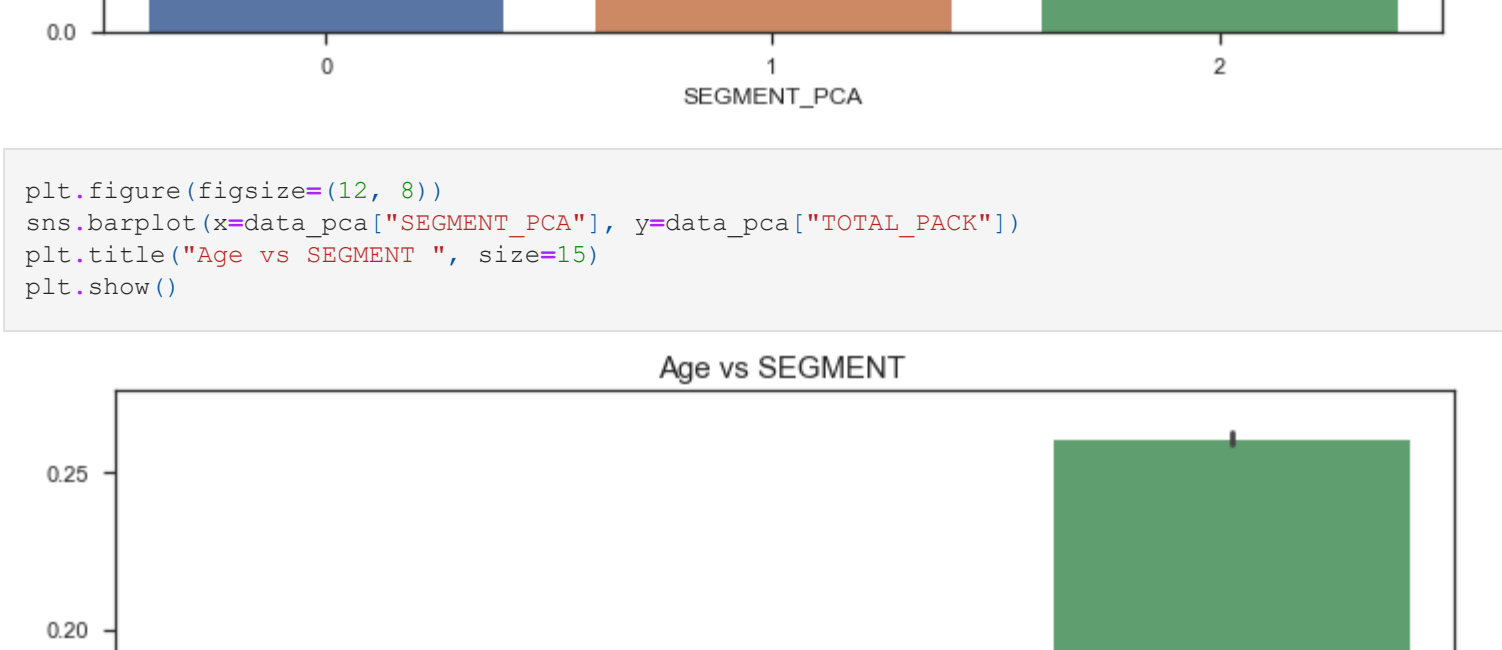
In [174…

```python
plt.figure(figsize=(12, 8))
sns.barplot(x=data_pca["SEGMENT_PCA"], y=data_pca["CLIENT_VRD_MOY"])
plt.title("VRD vs SEGMENT ", size=15)
plt.show()
```



In [175…

```python
plt.figure(figsize=(12, 8))
sns.barplot(x=data_pca["SEGMENT_PCA"], y=data_pca["TOTAL_PACK"])
plt.title("Age vs SEGMENT ", size=15)
plt.show()
```



In [176…

```python
data_pca
```

Out[176…

	CLIENT_AGE	CLIENT_ENCOURS_ENGAGEMENT	CLIENT_MMM	CLIENT_NOMBRE_CARTES	CLIENT_PROFESSION	CLIENT_TYPE_DEPOSANT
0	0.633333	0.000000	1.000000	0.6	0.258065	0.00
1	0.433333	0.145041	0.156169	0.2	0.645161	0.00
2	0.933333	0.000000	0.453050	0.2	0.000000	0.00
3	0.500000	0.000000	0.000000	0.0	0.580645	0.25
4	0.500000	1.000000	1.000000	0.4	0.258065	0.00
...	...	...	...	...	...	...
49995	0.466667	0.000000	0.000000	0.0	0.064516	0.25
49996	0.200000	0.000000	0.000000	0.2	0.645161	0.25
49997	0.366667	0.000000	0.000000	0.2	0.645161	0.25
49998	0.133333	0.000000	0.000000	0.2	0.645161	0.25
49999	0.233333	0.000000	0.000000	0.0	0.967742	0.25

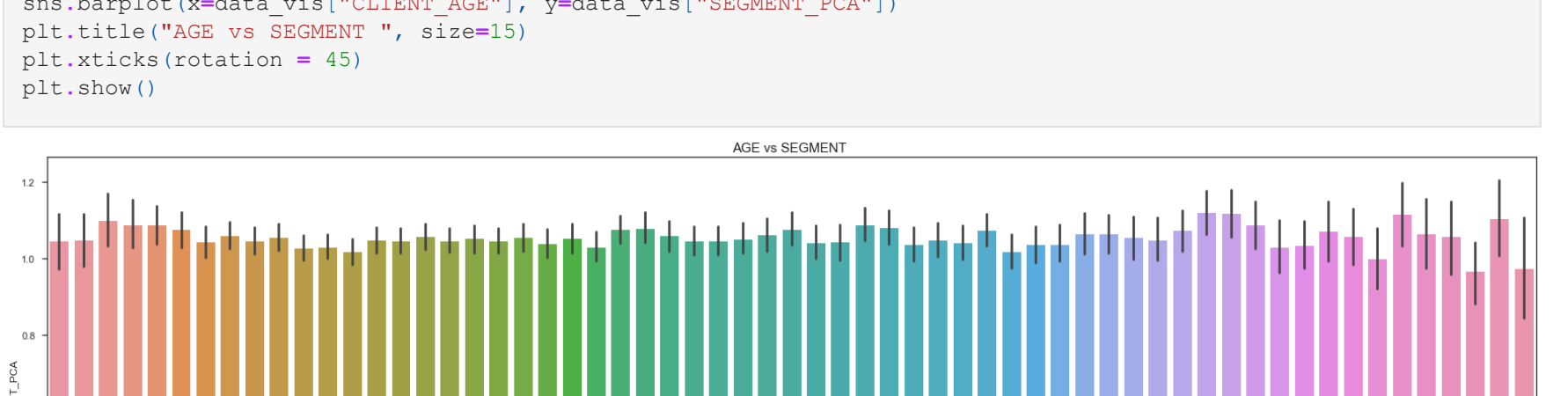
50000 rows × 14 columns

In [177…

```python
vis = data.sample(50000)
data_vis = pd.concat([vis.reset_index(drop=True),pd.DataFrame(pca)],axis=1)
data_vis.columns.values[-3:]=['Component 1','Component 2','Component 3']
data_vis['SEGMENT_PCA'] = kmeans_pca.labels_
```

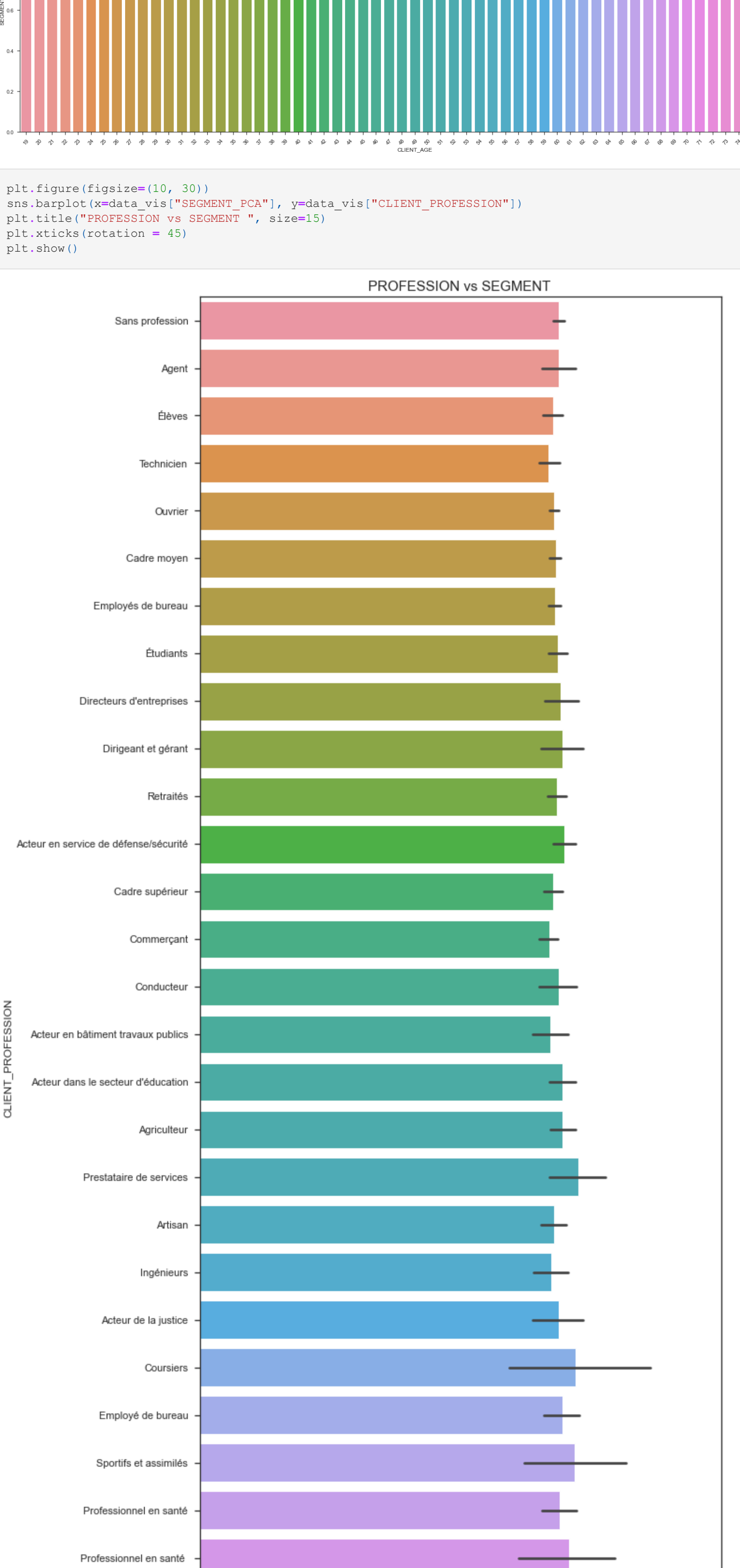
In [178…

```python
plt.figure(figsize=(30, 10))
sns.barplot(x=data_vis["CLIENT_AGE"], y=data_vis["SEGMENT_PCA"])
plt.title("AGE vs SEGMENT ", size=15)
plt.xticks(rotation = 45)
plt.show()
```



In [179…

```python
plt.figure(figsize=(10, 30))
sns.barplot(x=data_vis["SEGMENT_PCA"], y=data_vis["CLIENT_PROFESSION"])
plt.title("PROFESSION vs SEGMENT ", size=15)
plt.xticks(rotation = 45)
plt.show()
```



In [180…

```python
DColumns = data_pca[['CLIENT_AGE' ,'CLIENT_MMM' ,'CLIENT_ENCOURS_ENGAGEMENT','CLIENT_VRD_MOY','TOTAL_PACK','CLI

_, ax1 = plt.subplots(3,2, figsize=(25,22))
for i, col in enumerate(DColumns):
    sns.barplot(x='SEGMENT_PCA', y=col, data=data_pca, ax=ax1[i//2, i%2])

plt.show()
```

In [181…

```python
centers = data_pca.groupby("SEGMENT_PCA").mean()
```

In [182…

```python
centers
```

Out[182…

	CLIENT_AGE	CLIENT_ENCOURS_ENGAGEMENT	CLIENT_MMM	CLIENT_NOMBRE_CARTES	CLIENT_PROFESSION	CLIENT_TYPE_DEPO
SEGMENT_PCA						
0	0.491092	0.006369	0.404966	0.183222	0.524288	0.1
1	0.370837	0.012067	0.053349	0.096536	0.596308	0.1
2	0.395531	0.989155	0.491754	0.349494	0.444844	0.0

In [183…

```python
plt.rcParams['font.size'] = 14
```

In [184…

```python
colors = np.array(['red', 'green', 'blue', 'yellow'])
```

In [185…

```python
plt.scatter(data_pca['CLIENT_MMM'], data_pca['CLIENT_VRD_MOY'],c=colors[data_pca['SEGMENT_PCA']])
# TODO - cluster centers, marked by "+"
# Just make sure you understand this command:
plt.scatter(centers.CLIENT_MMM, centers.CLIENT_VRD_MOY, linewidths=3, marker='+', s=300, c='black')

# TODO - always remember to add labels!
plt.xlabel("MMM")
plt.ylabel("VRD moy")
```

Out[185…

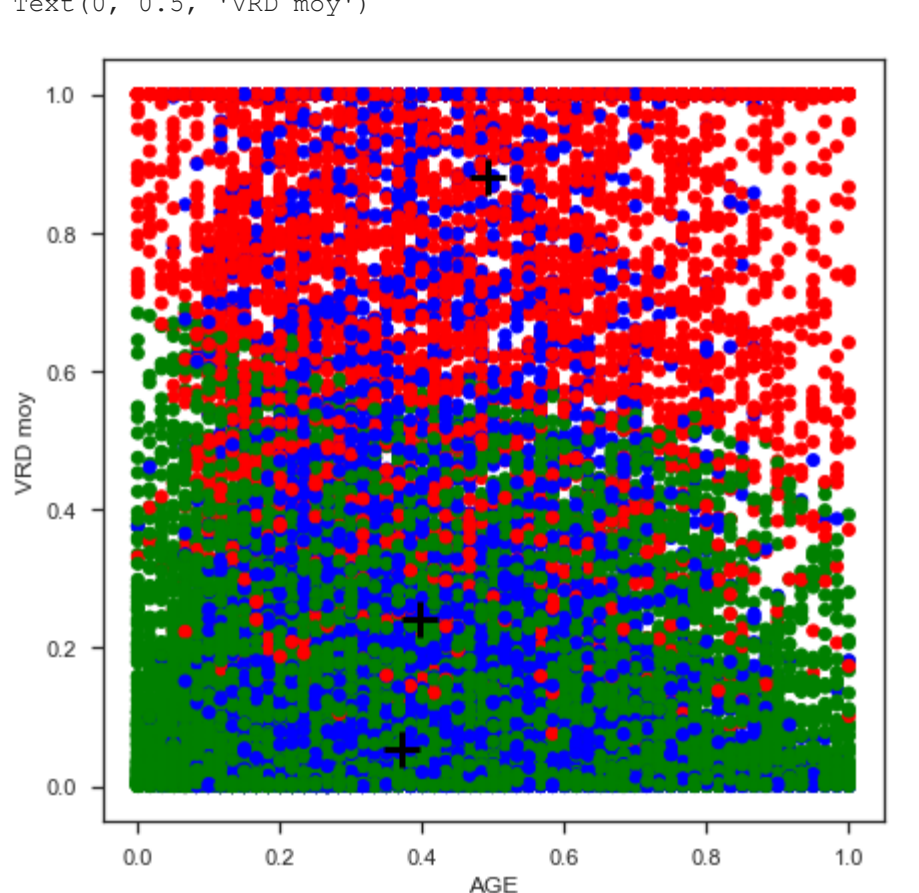
```
Text(0, 0.5, 'VRD moy')
```

In [186…

```python
plt.scatter(data_pca['CLIENT_AGE'], data_pca['CLIENT_VRD_MOY'],c=colors[data_pca['SEGMENT_PCA']])
# TODO - cluster centers, marked by "+"
# Just make sure you understand this command:
plt.scatter(centers.CLIENT_AGE, centers.CLIENT_VRD_MOY, linewidths=3, marker='+', s=300, c='black')

# TODO - always remember to add labels!
plt.xlabel("AGE")
plt.ylabel("VRD moy")
```

Out[186…

```
Text(0, 0.5, 'VRD moy')
```

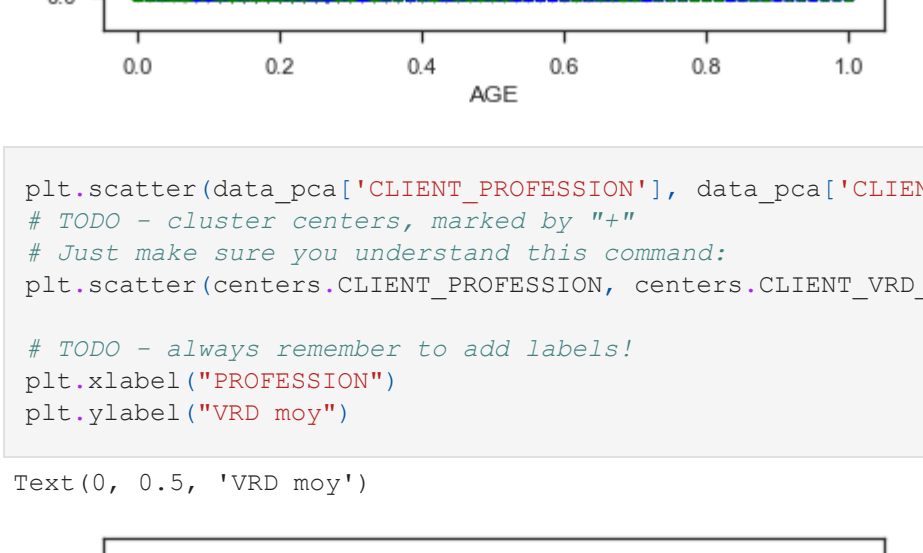


In [187…

```python
plt.scatter(data_pca['CLIENT_PROFESSION'], data_pca['CLIENT_VRD_MOY'],c=colors[data_pca['SEGMENT_PCA']])
# TODO - cluster centers, marked by "+"
# Just make sure you understand this command:
plt.scatter(centers.CLIENT_PROFESSION, centers.CLIENT_VRD_MOY, linewidths=3, marker='+', s=300, c='black')

# TODO - always remember to add labels!
plt.xlabel("PROFESSION")
plt.ylabel("VRD moy")
```

Out[187…

```
Text(0, 0.5, 'VRD moy')
```

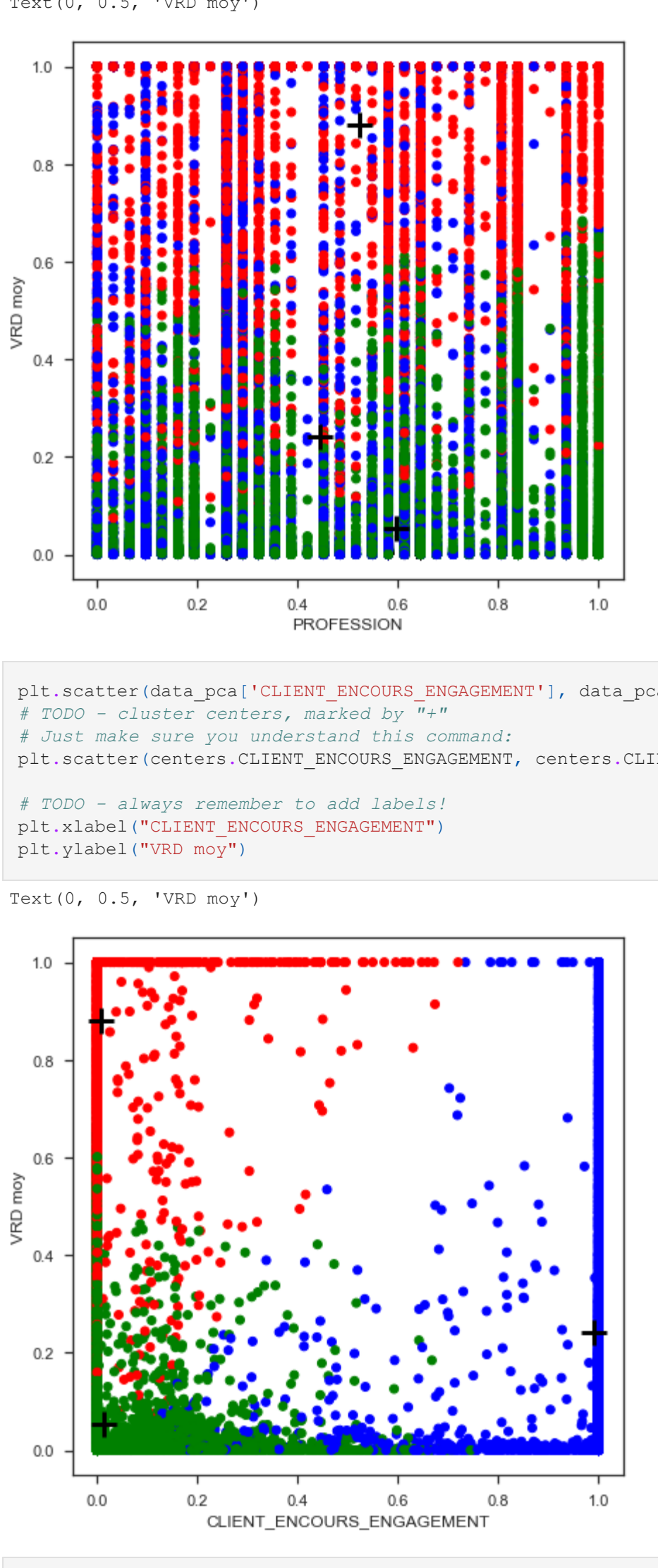


In [188…

```python
plt.scatter(data_pca['CLIENT_ENCOURS_ENGAGEMENT'], data_pca['CLIENT_VRD_MOY'],c=colors[data_pca['SEGMENT_PCA']]
# TODO - cluster centers, marked by "+"
# Just make sure you understand this command:
plt.scatter(centers.CLIENT_ENCOURS_ENGAGEMENT, centers.CLIENT_VRD_MOY, linewidths=3, marker='+', s=300, c='blac

# TODO - always remember to add labels!
plt.xlabel("CLIENT_ENCOURS_ENGAGEMENT")
plt.ylabel("VRD moy")
```

Out[188…

```
Text(0, 0.5, 'VRD moy')
```

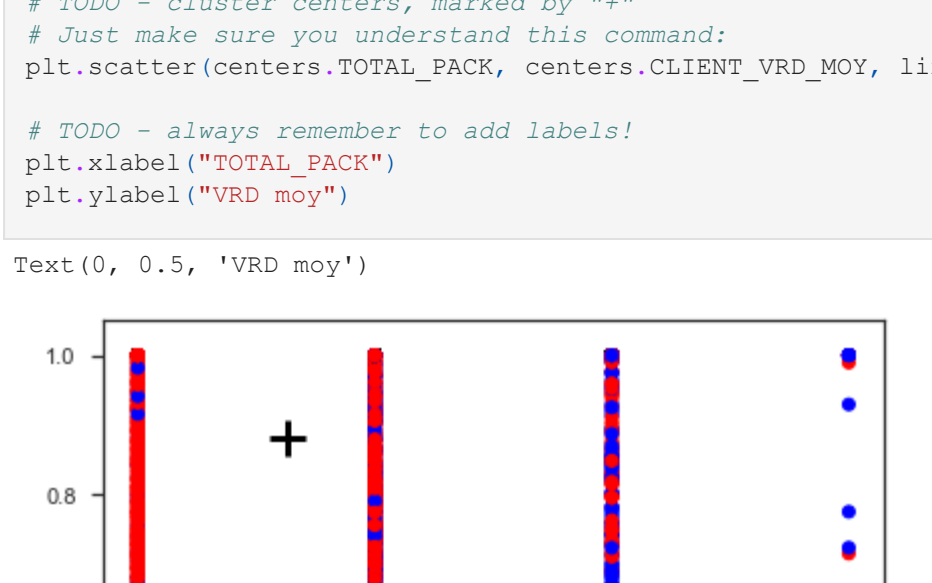


In [189…

```python
plt.scatter(data_pca['TOTAL_PACK'], data_pca['CLIENT_VRD_MOY'],c=colors[data_pca['SEGMENT_PCA']])
# TODO - cluster centers, marked by "+"
# Just make sure you understand this command:
plt.scatter(centers.TOTAL_PACK, centers.CLIENT_VRD_MOY, linewidths=3, marker='+', s=300, c='black')

# TODO - always remember to add labels!
plt.xlabel("TOTAL_PACK")
plt.ylabel("VRD moy")
```

Out[189…

```
Text(0, 0.5, 'VRD moy')
```



In [190…

```python
data_vis = data_vis.drop(columns=['Component 1', 'Component 2','Component 3'])
```

In [191…

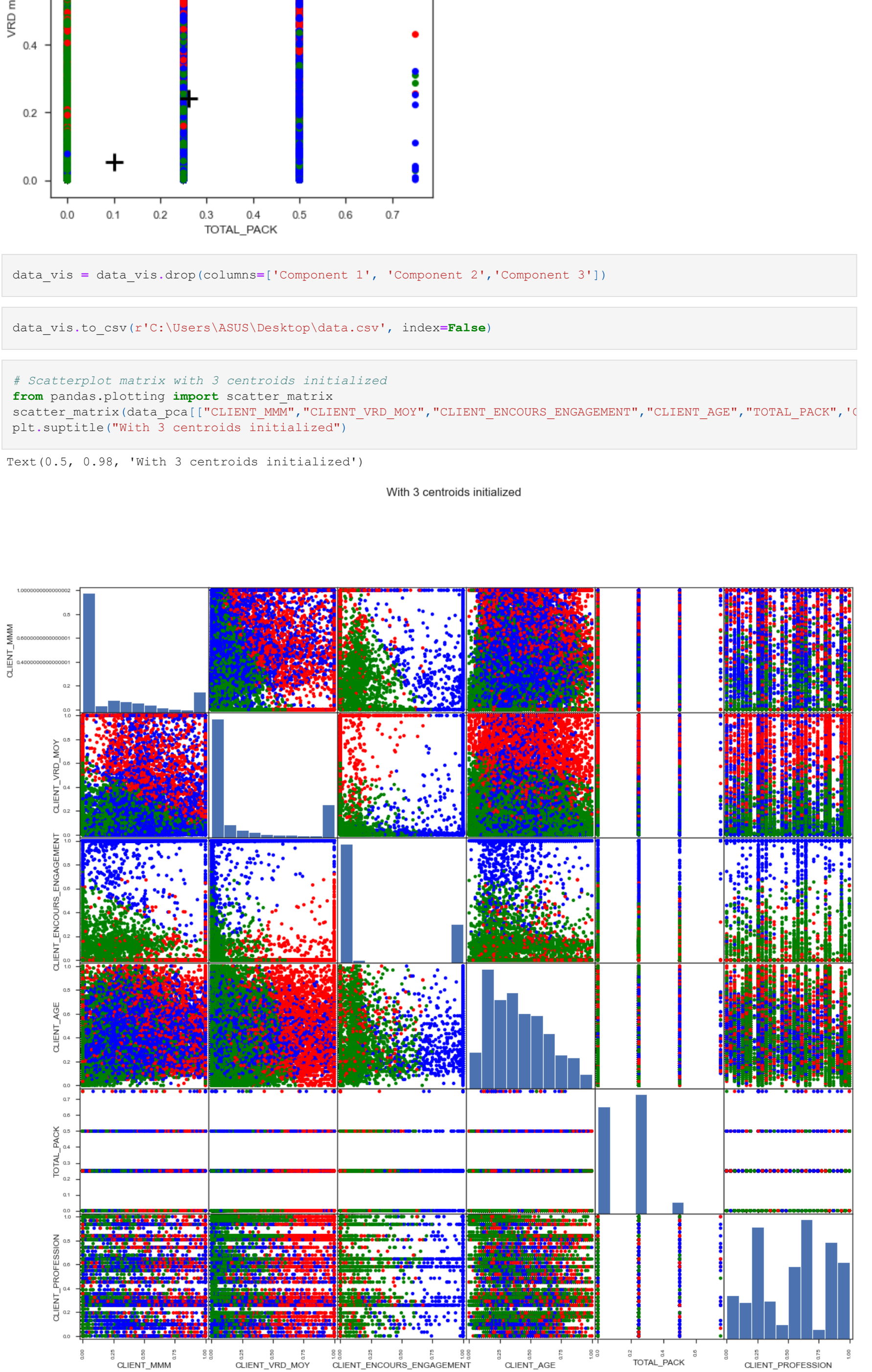
```python
data_vis.to_csv(r'C:\Users\ASUS\Desktop\data.csv', index=False)
```

In [192…

```python
# Scatterplot matrix with 3 centroids initialized
from pandas.plotting import scatter_matrix
scatter_matrix(data_pca[["CLIENT_MMM","CLIENT_VRD_MOY","CLIENT_ENCOURS_ENGAGEMENT","CLIENT_AGE","TOTAL_PACK",'C
plt.suptitle("With 3 centroids initialized")
```

Out[192…

```
Text(0.5, 0.98, 'With 3 centroids initialized')
```



Cluster 1 (Less active):

Relatively low average monthly flow (3980.838766) , low engagement value (5693.198826) , low average deposit value (32725.813561) with average age (51.594150)

Cluster 2 (Active):

Relatively normal average monthly flow (8620.316234) , low engagement value (26822.685970) , normal average deposit value (23745.129939) with average age (44.007985)

Cluster 3 (More active):

Relatively high average monthly flow (195454.947502) , high engagement value (104553.369407) , hight average deposit value (319753.123302) with average age (48.560799)

In [203…

```
pip install nbconvert
```

```
Requirement already satisfied: nbconvert in c:\users\asus\anaconda\lib\site-packages (6.1.0)
Requirement already satisfied: defusedxml in c:\users\asus\anaconda\lib\site-packages (from nbconvert) (0.7.1)
Requirement already satisfied: mistune<2,>=0.8.1 in c:\users\asus\anaconda\lib\site-packages (from nbconvert)
(0.8.4)
Requirement already satisfied: nbformat>=4.4 in c:\users\asus\anaconda\lib\site-packages (from nbconvert) (5.1.
3)
Requirement already satisfied: jinja2>=2.4 in c:\users\asus\anaconda\lib\site-packages (from nbconvert) (2.11.
3)
Requirement already satisfied: nbclient<0.6.0,>=0.5.0 in c:\users\asus\anaconda\lib\site-packages (from nbconve
rt) (0.5.3)
Requirement already satisfied: traitlets>=5.0 in c:\users\asus\anaconda\lib\site-packages (from nbconvert) (5.
1.0)
Requirement already satisfied: jupyterlab-pygments in c:\users\asus\anaconda\lib\site-packages (from nbconvert)
(0.1.2)
Requirement already satisfied: bleach in c:\users\asus\anaconda\lib\site-packages (from nbconvert) (4.0.0)
Requirement already satisfied: testpath in c:\users\asus\anaconda\lib\site-packages (from nbconvert) (0.5.0)
Requirement already satisfied: jupyter-core in c:\users\asus\anaconda\lib\site-packages (from nbconvert) (4.8.
1)
Requirement already satisfied: entrypoints>=0.2.2 in c:\users\asus\anaconda\lib\site-packages (from nbconvert)
(0.3)
Requirement already satisfied: pandocfilters>=1.4.1 in c:\users\asus\anaconda\lib\site-packages (from nbconver
t) (1.4.3)
Requirement already satisfied: pygments>=2.4.1 in c:\users\asus\anaconda\lib\site-packages (from nbconvert) (2.
10.0)
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\asus\anaconda\lib\site-packages (from jinja2>=2.4->
nbconvert) (1.1.1)Note: you may need to restart the kernel to use updated packages.

Requirement already satisfied: nest-asyncio in c:\users\asus\anaconda\lib\site-packages (from nbclient<0.6.0,>=
0.5.0->nbconvert) (1.5.1)
Requirement already satisfied: jupyter-client>=6.1.5 in c:\users\asus\anaconda\lib\site-packages (from nbclient
<0.6.0,>=0.5.0->nbconvert) (6.1.12)
Requirement already satisfied: async-generator in c:\users\asus\anaconda\lib\site-packages (from nbclient<0.6.
0,>=0.5.0->nbconvert) (1.10)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\asus\anaconda\lib\site-packages (from jupyter-c
lient>=6.1.5->nbclient<0.6.0,>=0.5.0->nbconvert) (2.8.2)
Requirement already satisfied: tornado>=4.1 in c:\users\asus\anaconda\lib\site-packages (from jupyter-client>=
6.1.5->nbclient<0.6.0,>=0.5.0->nbconvert) (6.1)
Requirement already satisfied: pyzmq>=13 in c:\users\asus\anaconda\lib\site-packages (from jupyter-client>=6.1.
5->nbclient<0.6.0,>=0.5.0->nbconvert) (22.2.1)
Requirement already satisfied: pywin32>=1.0 in c:\users\asus\anaconda\lib\site-packages (from jupyter-core->nbc
onvert) (228)
Requirement already satisfied: ipython-genutils in c:\users\asus\anaconda\lib\site-packages (from nbformat>=4.4
->nbconvert) (0.2.0)
Requirement already satisfied: jsonschema!=2.5.0,>=2.4 in c:\users\asus\anaconda\lib\site-packages (from nbform
at>=4.4->nbconvert) (3.2.0)
Requirement already satisfied: pyrsistent>=0.14.0 in c:\users\asus\anaconda\lib\site-packages (from jsonschema!
=2.5.0,>=2.4->nbformat>=4.4->nbconvert) (0.18.0)
Requirement already satisfied: attrs>=17.4.0 in c:\users\asus\anaconda\lib\site-packages (from jsonschema!=2.5.
0,>=2.4->nbformat>=4.4->nbconvert) (21.2.0)
Requirement already satisfied: six>=1.11.0 in c:\users\asus\anaconda\lib\site-packages (from jsonschema!=2.5.0,
>=2.4->nbformat>=4.4->nbconvert) (1.16.0)
Requirement already satisfied: setuptools in c:\users\asus\anaconda\lib\site-packages (from jsonschema!=2.5.0,>
=2.4->nbformat>=4.4->nbconvert) (58.0.4)
Requirement already satisfied: packaging in c:\users\asus\anaconda\lib\site-packages (from bleach->nbconvert)
(21.0)
Requirement already satisfied: webencodings in c:\users\asus\anaconda\lib\site-packages (from bleach->nbconver
t) (0.5.1)
Requirement already satisfied: pyparsing>=2.0.2 in c:\users\asus\anaconda\lib\site-packages (from packaging->bl
each->nbconvert) (3.0.4)
```

In [ ]: