# Assignment -Week4 Exercise 4.2 - Descriptive Modeling Clustering Exercise

Course: DSC630 - Predictive Analytics
Instructor: Fadi Alsaleem

'''Karthikeyan Chellamuthu '''

'''Date :06-26-2022'''

Using R or Python - You will be using the dataset als_data.csv to apply clustering methods for this assignment. This data gives anonymized data on ALS patients. With this data, complete the following steps:

- Remove any data that is not relevant to the patient’s ALS condition.
- Apply a standard scalar to the data.
- Create a plot of the cluster silhouette score versus the number of clusters in a K-means cluster.
- Use the plot created in (3) to choose an optimal number of clusters for K-means. Justify your choice.
- Fit a K-means model to the data with the optimal number of clusters chosen in part (4).
- Fit a PCA transformation with two features to the scaled data.
- Make a scatterplot of the PCA transformed data coloring each point by its cluster value.
- Summarize your results and make a conclusion.
- You can use R or Python to complete this assignment. Submit your code and output to the submission link. Make sure to add comments to all your *code and to document your steps, process, and analysis.*

In [1]:
```
# Exercise 4.2 - Descriptive Modeling Clustering Exercise we can use either R or Pyt
```

In [32]:
```
# Import the necessary library required
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
# Import from scikit learn for the  necessary librarys
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn import metrics
from scipy.spatial.distance import cdist
# Import from scikit learn for the  necessary librarys
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_samples, silhouette_score
import matplotlib.cm as cm
import matplotlib.style as style
#Import PCA Library
from sklearn.decomposition import PCA
```

In [4]:
```
# verify and set the attributes in  pandas dataframe
```

```
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
```

In [5]:

```
# create a dataframe to consume the als data provided
df_patientals = pd.read_csv('als_data.csv')
df_patientals.head(10)
```

Out[5]:

| | ID | Age_mean | Albumin_max | Albumin_median | Albumin_min | Albumin_range | ALSFRS_slope | AL |
|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 65 | 57.0 | 40.5 | 38.0 | 0.066202 | -0.965608 | |
| **1** | 2 | 48 | 45.0 | 41.0 | 39.0 | 0.010453 | -0.921717 | |
| **2** | 3 | 38 | 50.0 | 47.0 | 45.0 | 0.008929 | -0.914787 | |
| **3** | 4 | 63 | 47.0 | 44.0 | 41.0 | 0.012111 | -0.598361 | |
| **4** | 5 | 63 | 47.0 | 45.5 | 42.0 | 0.008292 | -0.444039 | |
| **5** | 6 | 36 | 51.0 | 47.0 | 46.0 | 0.009058 | -0.118353 | |
| **6** | 7 | 55 | 46.0 | 44.0 | 40.0 | 0.010850 | -1.225580 | |
| **7** | 8 | 55 | 45.0 | 42.0 | 38.0 | 0.018519 | -0.760417 | |
| **8** | 9 | 37 | 48.0 | 46.0 | 41.0 | 0.012681 | -1.010148 | |
| **9** | 11 | 72 | 44.0 | 42.0 | 38.0 | 0.010714 | -0.107861 | |

In [9]:

```
# summary to get the summary results of the given data set to perform the clustering
# Recomed to create a summary and grouping to know more about your data frame of the

df_patientals.describe()
```

Out[9]:

| | ID | Age_mean | Albumin_max | Albumin_median | Albumin_min | Albumin_range | AL |
|---|---|---|---|---|---|---|---|
| **count** | 2223.000000 | 2223.000000 | 2223.000000 | 2223.000000 | 2223.000000 | 2223.000000 | 2 |
| **mean** | 1214.874944 | 54.550157 | 47.011134 | 43.952542 | 40.766347 | 0.013779 | |
| **std** | 696.678300 | 11.396546 | 3.233980 | 2.654804 | 3.193087 | 0.009567 | |
| **min** | 1.000000 | 18.000000 | 37.000000 | 34.500000 | 24.000000 | 0.000000 | |
| **25%** | 614.500000 | 47.000000 | 45.000000 | 42.000000 | 39.000000 | 0.009042 | |
| **50%** | 1213.000000 | 55.000000 | 47.000000 | 44.000000 | 41.000000 | 0.012111 | |
| **75%** | 1815.500000 | 63.000000 | 49.000000 | 46.000000 | 43.000000 | 0.015873 | |
| **max** | 2424.000000 | 81.000000 | 70.300000 | 51.100000 | 49.000000 | 0.243902 | |

In [7]:

```
# Verify the attributes to find type of data and anomalies or fileter the unwanted c
df_patientals.dtypes
```

Out[7]:

```
ID                                  int64
Age_mean                            int64
Albumin_max                       float64
Albumin_median                    float64
Albumin_min                       float64
```

```
Albumin_range                      float64
ALSFRS_slope                       float64
ALSFRS_Total_max                     int64
ALSFRS_Total_median                float64
ALSFRS_Total_min                     int64
ALSFRS_Total_range                 float64
ALT.SGPT._max                      float64
ALT.SGPT._median                   float64
ALT.SGPT._min                      float64
ALT.SGPT._range                    float64
AST.SGOT._max                        int64
AST.SGOT._median                   float64
AST.SGOT._min                      float64
AST.SGOT._range                    float64
Bicarbonate_max                    float64
Bicarbonate_median                 float64
Bicarbonate_min                    float64
Bicarbonate_range                  float64
Blood.Urea.Nitrogen..BUN._max      float64
Blood.Urea.Nitrogen..BUN._median   float64
Blood.Urea.Nitrogen..BUN._min      float64
Blood.Urea.Nitrogen..BUN._range    float64
bp_diastolic_max                     int64
bp_diastolic_median                float64
bp_diastolic_min                     int64
bp_diastolic_range                 float64
bp_systolic_max                      int64
bp_systolic_median                 float64
bp_systolic_min                      int64
bp_systolic_range                  float64
Calcium_max                        float64
Calcium_median                     float64
Calcium_min                        float64
Calcium_range                      float64
Chloride_max                       float64
Chloride_median                    float64
Chloride_min                       float64
Chloride_range                     float64
Creatinine_max                     float64
Creatinine_median                  float64
Creatinine_min                     float64
Creatinine_range                   float64
Gender_mean                          int64
Glucose_max                        float64
Glucose_median                     float64
Glucose_min                        float64
Glucose_range                      float64
hands_max                            int64
hands_median                       float64
hands_min                            int64
hands_range                        float64
Hematocrit_max                     float64
Hematocrit_median                  float64
Hematocrit_min                     float64
Hematocrit_range                   float64
Hemoglobin_max                     float64
Hemoglobin_median                  float64
Hemoglobin_min                     float64
Hemoglobin_range                   float64
leg_max                              int64
leg_median                         float64
leg_min                              int64
leg_range                          float64
mouth_max                            int64
```

```
mouth_median                        float64
mouth_min                             int64
mouth_range                         float64
onset_delta_mean                      int64
onset_site_mean                       int64
Platelets_max                         int64
Platelets_median                    float64
Platelets_min                       float64
Potassium_max                       float64
Potassium_median                    float64
Potassium_min                       float64
Potassium_range                     float64
pulse_max                             int64
pulse_median                        float64
pulse_min                             int64
pulse_range                         float64
respiratory_max                       int64
respiratory_median                  float64
respiratory_min                       int64
respiratory_range                   float64
Sodium_max                          float64
Sodium_median                       float64
Sodium_min                          float64
Sodium_range                        float64
SubjectID                             int64
trunk_max                             int64
trunk_median                        float64
trunk_min                             int64
trunk_range                         float64
Urine.Ph_max                        float64
Urine.Ph_median                     float64
Urine.Ph_min                        float64
dtype: object
```

In [8]:

```
# Verify the shape before dropping the total number of attributes present in the df
df_patientals.shape
```

Out[8]:

```
(2223, 101)
```

In [10]:

```
#  1. Data Clensing Remove any data that is not relevant to the patient’s ALS condit

df_patientals.drop([ "ID","SubjectID"], axis=1, inplace=True)
```

In [12]:

```
# Verify the shape after  dropping the total number of attributes present in the df
df_patientals.shape
```

Out[12]:

```
(2223, 99)
```

In [13]:

```
# print few samle records from the dataframe
df_patientals.head(10)
```

Out[13]:

| | Age_mean | Albumin_max | Albumin_median | Albumin_min | Albumin_range | ALSFRS_slope | ALSFRS |
|---|---|---|---|---|---|---|---|
| **0** | 65 | 57.0 | 40.5 | 38.0 | 0.066202 | -0.965608 | |
| **1** | 48 | 45.0 | 41.0 | 39.0 | 0.010453 | -0.921717 | |
| **2** | 38 | 50.0 | 47.0 | 45.0 | 0.008929 | -0.914787 | |

| | Age_mean | Albumin_max | Albumin_median | Albumin_min | Albumin_range | ALSFRS_slope | ALSFRS |
|---|---|---|---|---|---|---|---|
| **3** | 63 | 47.0 | 44.0 | 41.0 | 0.012111 | -0.598361 | |
| **4** | 63 | 47.0 | 45.5 | 42.0 | 0.008292 | -0.444039 | |
| **5** | 36 | 51.0 | 47.0 | 46.0 | 0.009058 | -0.118353 | |
| **6** | 55 | 46.0 | 44.0 | 40.0 | 0.010850 | -1.225580 | |
| **7** | 55 | 45.0 | 42.0 | 38.0 | 0.018519 | -0.760417 | |
| **8** | 37 | 48.0 | 46.0 | 41.0 | 0.012681 | -1.010148 | |
| **9** | 72 | 44.0 | 42.0 | 38.0 | 0.010714 | -0.107861 | |

In [14]:
```
# 2. Apply a standard scalar to the data.
scaler = StandardScaler()
A = scaler.fit_transform(df_patientals)
print(A)
```

```
[[ 0.91713698  3.08941722 -1.30078105 ... -0.88037551  0.46305355
   1.86853157]
 [-0.57487867 -0.62201561 -1.11240084 ...  0.1926645  -1.13720768
  -0.41915124]
 [-1.45253494  0.92441474  1.14816173 ... -0.88037551 -1.13720768
  -0.41915124]
 ...
 [-0.6626443  -0.31272954  0.01788044 ...  2.33874452  0.46305355
  -0.41915124]
 [-1.54030057  0.61512867  0.01788044 ... -0.88037551 -1.13720768
  -0.41915124]
 [-0.57487867  0.3058426   0.39464087 ... -1.95341552 -1.13720768
  -0.41915124]]
```

In [17]:
```
# compute the mean and verifiy if its closer to 0 after applying the standard scalar
np.mean(A),np.std(A)
```

Out[17]:
```
(-8.908541299845311e-17, 1.0)
```

In [ ]:
```
'''Mean value should be close to 0 after applying StandardScalar and Standard deviat
```

In [20]:
```
#  verify the data frame after apply a standard scalar to the data.
A.shape
```

Out[20]:
```
(2223, 99)
```

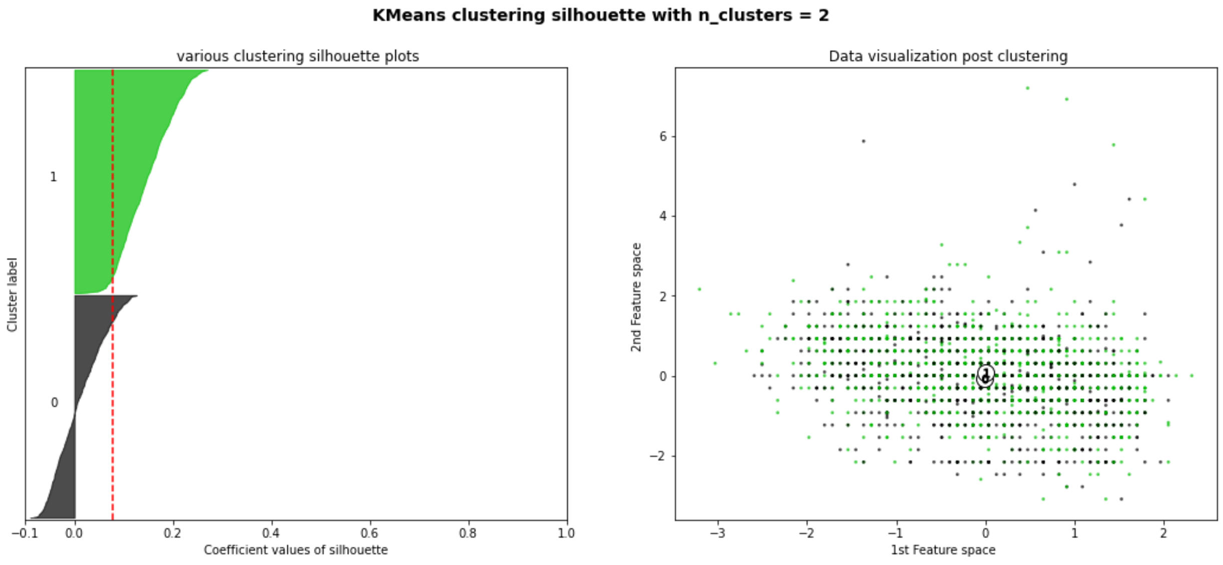
In [21]:
```
'''Results are expected as we have removed two columns'''
```

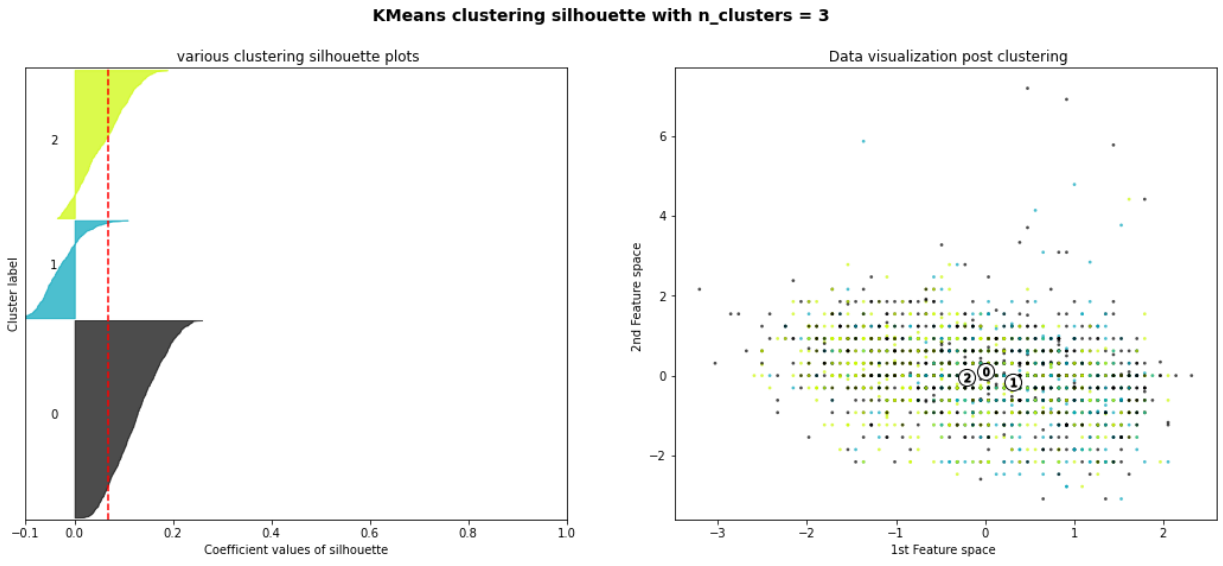
Out[21]:
```
'Results are expected as we have removed two columns'
```

In [27]:
```
#3. Create a plot of the cluster silhouette score versus the number of clusters in a
# Reference https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhou

range_n_clusters = [2, 3, 4, 5, 6, 7, 8, 9, 10]
silhouette_avg_n_clusters = []
```

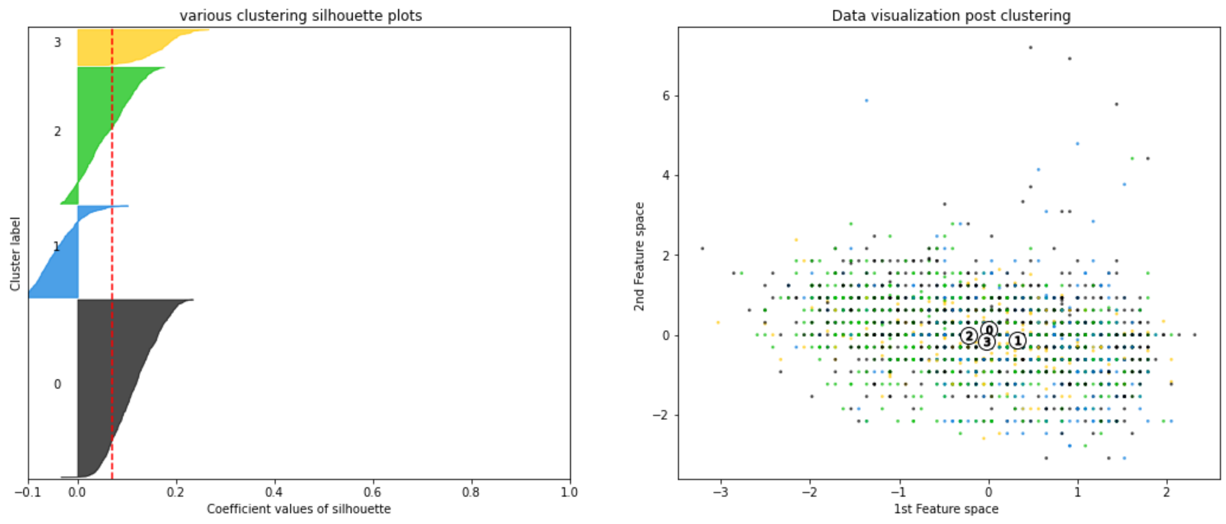
```python
for n_clusters in range_n_clusters:

    fig, (ax1, ax2) = plt.subplots(1, 2)
    fig.set_size_inches(18, 7)
    ax1.set_xlim([-0.1, 1])
    ax1.set_ylim([0, len(A) + (n_clusters + 1) * 10])
    clusterer = KMeans(n_clusters=n_clusters, random_state=42)
    cluster_labels = clusterer.fit_predict(A)
    silhouette_avg = silhouette_score(A, cluster_labels)
    print("For n_clusters =", n_clusters,
          "The average silhouette_score is :", silhouette_avg)

    silhouette_avg_n_clusters.append(silhouette_avg)
    sample_silhouette_values = silhouette_samples(A, cluster_labels)

    y_lower = 10
    for i in range(n_clusters):

        ith_cluster_silhouette_values = \
            sample_silhouette_values[cluster_labels == i]

        ith_cluster_silhouette_values.sort()

        size_cluster_i = ith_cluster_silhouette_values.shape[0]
        y_upper = y_lower + size_cluster_i

        color = cm.nipy_spectral(float(i) / n_clusters)
        ax1.fill_betweenx(np.arange(y_lower, y_upper),
                          0, ith_cluster_silhouette_values,
                          facecolor=color, edgecolor=color, alpha=0.7)


        ax1.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))

        y_lower = y_upper + 10
    ax1.set_title(" various clustering silhouette plots")
    ax1.set_xlabel("Coefficient values of silhouette")
    ax1.set_ylabel("Cluster label")


    ax1.axvline(x=silhouette_avg, color="red", linestyle="--")

    ax1.set_yticks([])
    ax1.set_xticks([-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])

    colors = cm.nipy_spectral(cluster_labels.astype(float) / n_clusters)
    ax2.scatter(A[:, 0], A[:, 1], marker='.', s=30, lw=0, alpha=0.7,
                c=colors, edgecolor='k')


    centers = clusterer.cluster_centers_

    ax2.scatter(centers[:, 0], centers[:, 1], marker='o',
                c="white", alpha=1, s=200, edgecolor='k')

    for i, c in enumerate(centers):
        ax2.scatter(c[0], c[1], marker='$%d$' % i, alpha=1,
                    s=50, edgecolor='k')

    ax2.set_title(" Data visualization post clustering")
    ax2.set_xlabel("1st Feature space ")
    ax2.set_ylabel("2nd Feature space ")
    plt.suptitle(("KMeans clustering silhouette "
                  "with n_clusters = %d" % n_clusters),
                 fontsize=14, fontweight='bold')

plt.show()

style.use("fivethirtyeight")
plt.plot(range_n_clusters, silhouette_avg_n_clusters)
plt.xlabel("Number of Clusters (k)")
plt.ylabel("silhouette score")
plt.show()
```

```
For n_clusters = 2 The average silhouette_score is : 0.07878005888570402
For n_clusters = 3 The average silhouette_score is : 0.0687707291658565
For n_clusters = 4 The average silhouette_score is : 0.06973816142698218
For n_clusters = 5 The average silhouette_score is : 0.05697679932842005
For n_clusters = 6 The average silhouette_score is : 0.06477886829610223
For n_clusters = 7 The average silhouette_score is : 0.05187647631845004
For n_clusters = 8 The average silhouette_score is : 0.04954004349267961
For n_clusters = 9 The average silhouette_score is : 0.04393719582297171
For n_clusters = 10 The average silhouette_score is : 0.046121611845315456
```

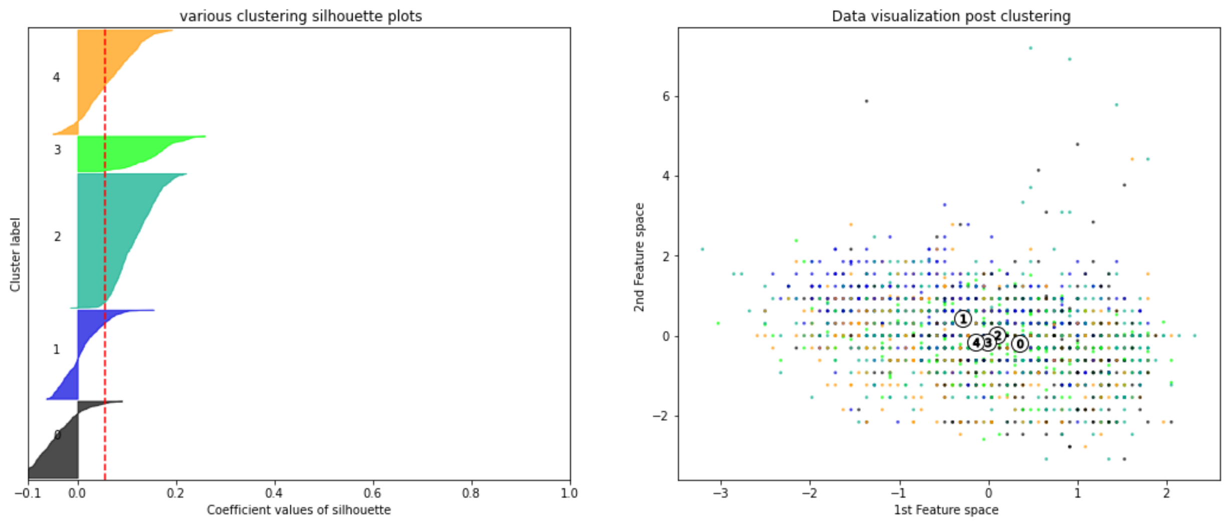




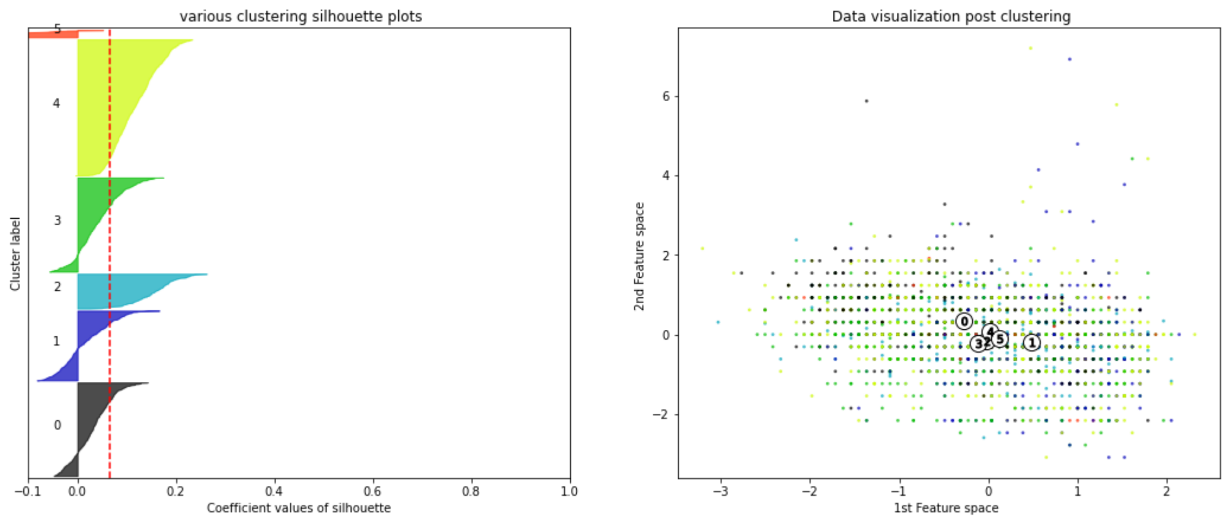
**KMeans clustering silhouette with n_clusters = 4**

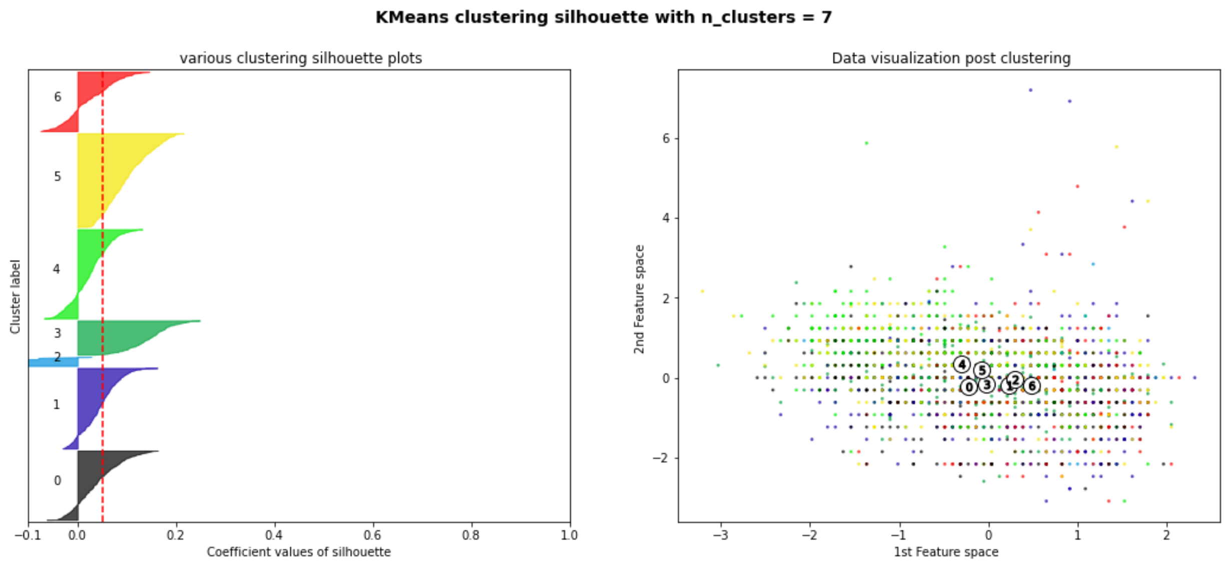


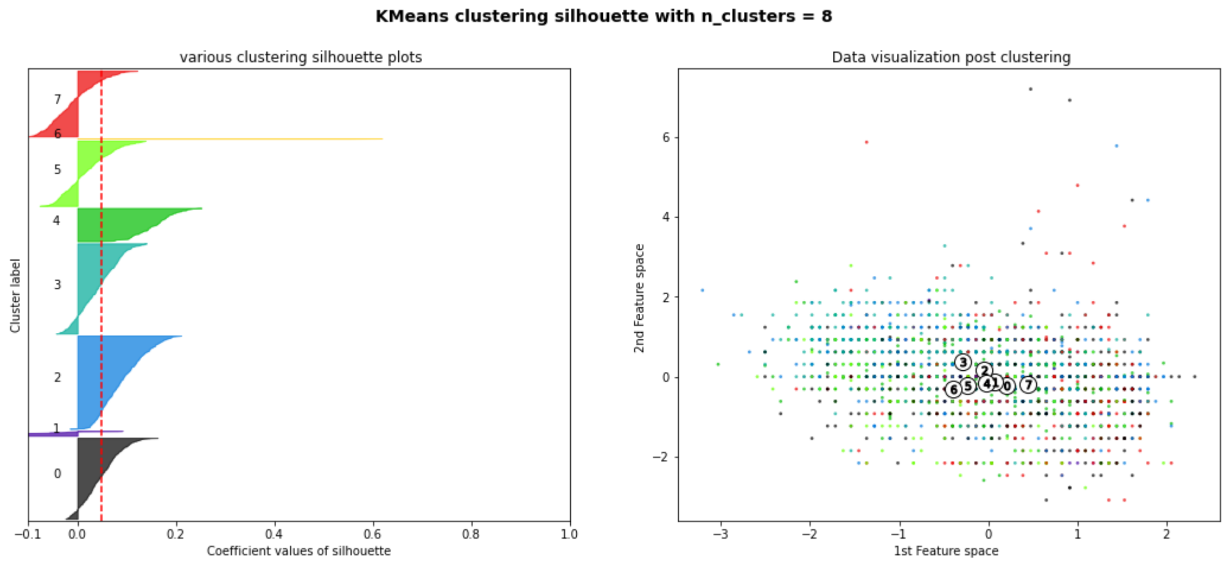
**KMeans clustering silhouette with n_clusters = 5**

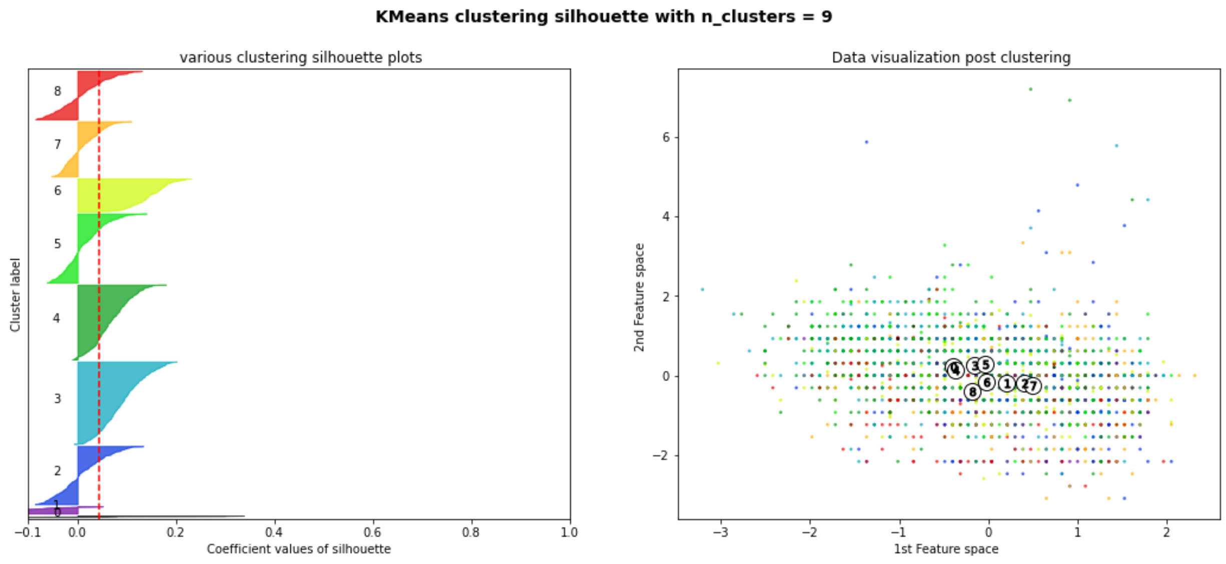


**KMeans clustering silhouette with n_clusters = 6**

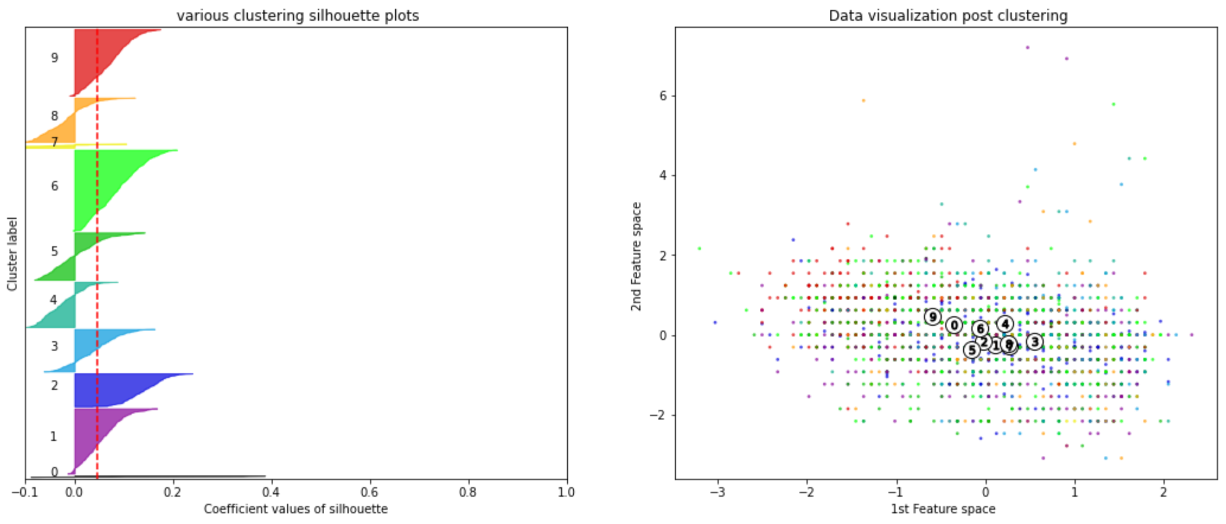


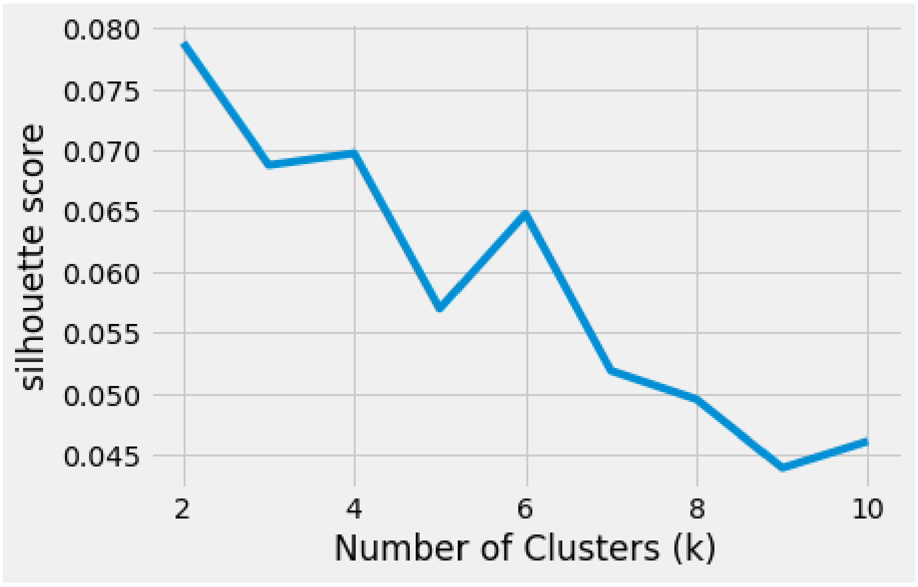

KMeans clustering silhouette with n_clusters = 7
various clustering silhouette plots
Cluster label
Coefficient values of silhouette
Data visualization post clustering
2nd Feature space
1st Feature space


KMeans clustering silhouette with n_clusters = 8
various clustering silhouette plots
Cluster label
Coefficient values of silhouette
Data visualization post clustering
2nd Feature space
1st Feature space


KMeans clustering silhouette with n_clusters = 9
various clustering silhouette plots
Cluster label
Coefficient values of silhouette
Data visualization post clustering
2nd Feature space
1st Feature space

**KMeans clustering silhouette with n_clusters = 10**





```
In [29]: # 4. Use the plot created in (3) to choose an optimal number of clusters for K-means

         '''Justification: Out of all the silhouette scores nothing can be considered as its
```

```
Out[29]: 'Justification: Out of all the silhouette scores nothing can be considered as its be
         low average of the score'
```

```
In [30]: #5. Fit a K-means model to the data with the optimal number of clusters chosen in pa

         kmeans = KMeans(init="random",n_clusters=2, n_init=10, max_iter=300, random_state=42
         kmeans.fit(A)
```

```
Out[30]: KMeans(init='random', n_clusters=2, random_state=42)
```

```
In [35]: #  6. Fit a PCA transformation with two features to the scaled data
         pca_patientals = PCA(n_components=2)
         pca_patientals.fit(A)
```

```
Out[35]: PCA(n_components=2)
```

```
In [37]: pca_patientals.transform(A)
```

```
Out[37]: array([[-1.42673269, -2.31916782],
       [-1.44023999, -4.87204741],
       [ 1.61786833, -0.42797708],
       ...,
       [-0.4329176 ,  4.24406748],
       [-0.33078897,  3.31700946],
       [ 1.46800628,  0.58321707]])
```

In [39]:

```
# add an additional variable to transform A
score_patientals = pca_patientals.transform(A)
```

In [56]:

```
# compute Eigen values
print('Variance Ratio: {}'.format(pca_patientals.explained_variance_ratio_))
print('Eigenvalues: {}'.format(pca_patientals.explained_variance_))
```

```
Variance Ratio: [0.11330548 0.06446611]
Eigenvalues: [11.22229079  6.38501759]
```

In [52]:

```
# scored assigned clusters are being assinged to a dataframe
df_patientals_kscore = pd.concat([df_patientals.reset_index(drop=True), pd.DataFrame
df_patientals_kscore.columns.values[-2: ] = ['Value_PCA_A', 'Value_PCA_B']
df_patientals_kscore['K-means PCA clusterin'] = kmeans.labels_
df_patientals_kscore['Segment'] = df_patientals_kscore['K-means PCA clusterin'].map(
```

In [46]:

```
# Display few records from the newly created dataframe
df_patientals_kscore.head(10)
```

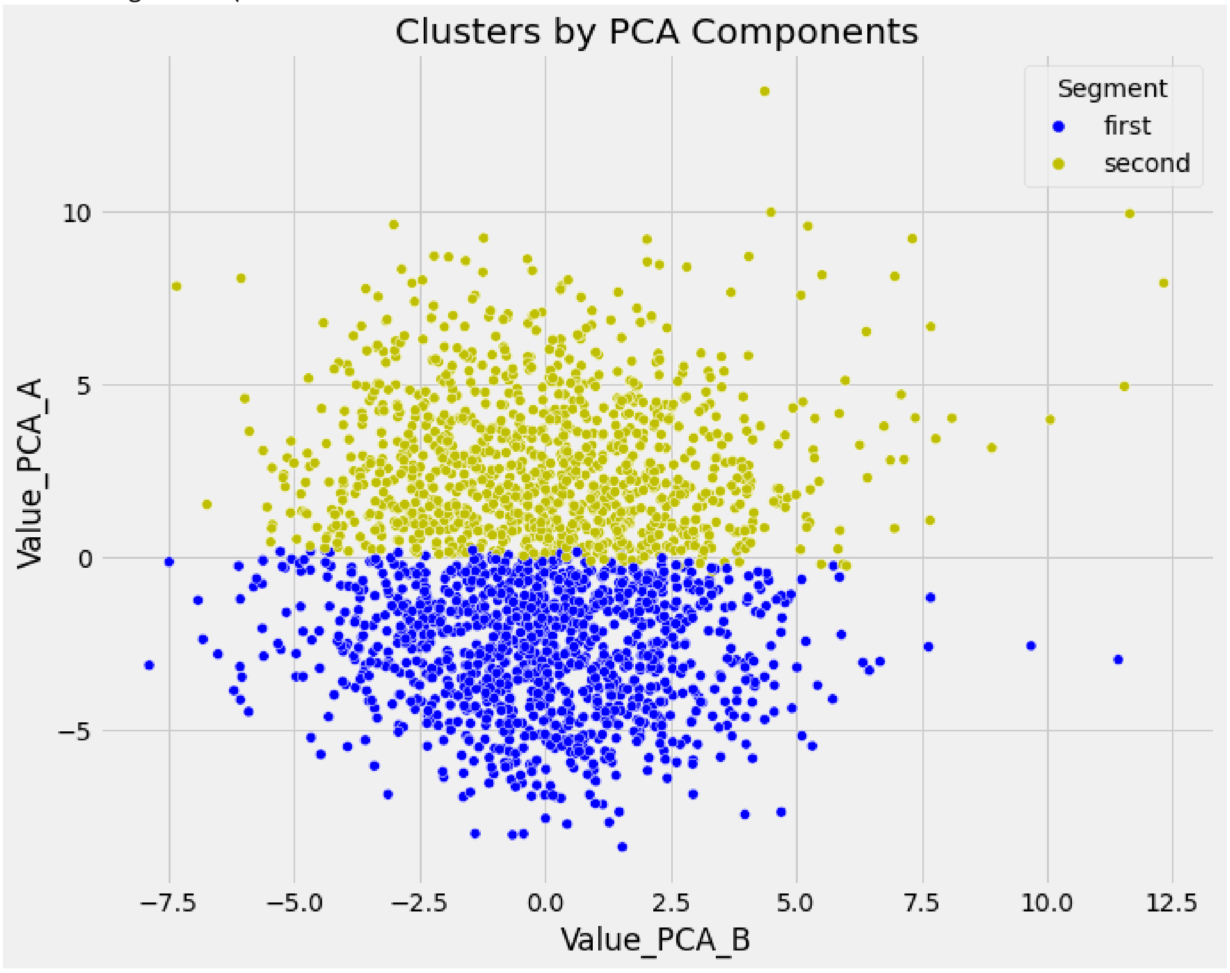
Out[46]:

| | Age_mean | Albumin_max | Albumin_median | Albumin_min | Albumin_range | ALSFRS_slope | ALSFRS |
|---|---|---|---|---|---|---|---|
| **0** | 65 | 57.0 | 40.5 | 38.0 | 0.066202 | -0.965608 | |
| **1** | 48 | 45.0 | 41.0 | 39.0 | 0.010453 | -0.921717 | |
| **2** | 38 | 50.0 | 47.0 | 45.0 | 0.008929 | -0.914787 | |
| **3** | 63 | 47.0 | 44.0 | 41.0 | 0.012111 | -0.598361 | |
| **4** | 63 | 47.0 | 45.5 | 42.0 | 0.008292 | -0.444039 | |
| **5** | 36 | 51.0 | 47.0 | 46.0 | 0.009058 | -0.118353 | |
| **6** | 55 | 46.0 | 44.0 | 40.0 | 0.010850 | -1.225580 | |
| **7** | 55 | 45.0 | 42.0 | 38.0 | 0.018519 | -0.760417 | |
| **8** | 37 | 48.0 | 46.0 | 41.0 | 0.012681 | -1.010148 | |
| **9** | 72 | 44.0 | 42.0 | 38.0 | 0.010714 | -0.107861 | |

In [58]:

```
# 7. Make a scatterplot of the PCA transformed data coloring each point by its clust

x_axis = df_patientals_kscore['Value_PCA_B']
y_axis = df_patientals_kscore['Value_PCA_A']
plt.figure(figsize =(10,8))
sns.scatterplot(x_axis,y_axis, hue=df_patientals_kscore['Segment'], palette = ['b',
plt.title('Clusters by PCA Components')
plt.show()
```

```
C:\Users\LENOVO\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarnin
g: Pass the following variables as keyword args: x, y. From version 0.12, the only v
alid positional argument will be `data`, and passing other arguments without an expl
icit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [61]:

```
#  8. Summarize your results and make a conclusion
'''
Here with i am concluding with my results and again is proven that using the using K
Also abouve Justification concludes that out of all the silhouette scores nothing ca

'''
```

Out[61]:

```
'\nHere with i am concluding with my results and again is proven that using the usin
g K-means we can visually separate almost the entire data set, overall the clusterin
g was good. However, there are few very minor overlaps between blue and yellow segme
nt.\nAlso abouve Justification concludes that out of all the silhouette scores nothi
ng can be considered as its below average of the score, Eigen values are computed an
d it was found 11.4% and 6.5% respectively.\n\n'
```

In [ ]: