ELG5255 Applied Machine Learning

Group Assignment #3

<u>Group 4:</u>

1. Yomna Mohamed Sayed Ahmed ID: 300327217
2. Khaled Mohamed Mohamed Mahmoud ID: 300327242
3. Marwa Hamdi Boraie Mahmoud ID: 300327267

Part 1: *Calculations*

Use the k-means algorithm and Euclidean distance to cluster the following 5 data points into 2 clusters: A1=(2,5), A2=(5,8), A3=(7,5), A4=(1,2), A5=(4,9). Suppose that the initial centroids (centers of each cluster) are A2 and A4. Using k-means, cluster the 5 points and show the followings for one iteration only:

(a) Show step-by-step the performed calculations to cluster the 5 points.

→Initial cluster centers are: C1→A2 (5, 8) and C2→A4 (1, 2).

→ Calculate the Euclidean distance function between two points a = (x1, y1) and b = (x2, y2) is defined as:

$$d\,(a, b) = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}$$

- We calculate the distance of each point from each of the center of the two clusters.
- The distance is calculated by using the given distance function.

→ The following illustration shows the calculation of distance between point A1 (2, 5) and each of the center of the two clusters..

Calculating Distance between A1 (2, 5) and C1 (5, 8), C2 (1, 2)

$$d_{(A1,\,C1)} = \sqrt{(5-2)^2 + (8-5)^2} = 4.24$$

$$d_{(A1,\,C2)} = \sqrt{(1-2)^2 + (2-5)^2} = 3.16$$

Calculating Distance between A2 (5, 8) and C1 (5, 8), C2 (1, 2)

$$d_{(A2,\,C1)} = \sqrt{(5-5)^2 + (8-8)^2} = 0$$

$$d_{(A2,\,C2)} = \sqrt{(1-5)^2 + (2-8)^2} = 7.21$$

Calculating Distance between A3 (7, 5) and C1 (5, 8), C2 (1, 2) d $_{(A3, C1)}$ = $\sqrt{(5-7)^2 + (8-5)^2}$ = 3.6

d $_{(A3, C2)}$ = $\sqrt{(1-7)^2 + (2-5)^2}$ = 6.7

Calculating Distance between A4 (1, 2) and C1 (5, 8), C2 (1, 2)

d $_{(A4, C1)}$ = $\sqrt{(5-1)^2 + (8-2)^2}$ = 7.21

d $_{(A4, C2)}$ = $\sqrt{(1-1)^2 + (2-2)^2}$ = 0

Calculating Distance between A5 (4, 9) and C1 (5, 8), C2 (1, 2)

d $_{(A5, C1)}$ = $\sqrt{(5-4)^2 + (8-9)^2}$ = 1.41

d $_{(A5, C2)}$ = $\sqrt{(1-4)^2 + (2-9)^2}$ = 7.62

Then,

- We draw a table showing all the results.
- Using the table, we decide which point belongs to which cluster.
- The given point belongs to that cluster whose center is nearest to it.

Given Points	**Distance from center (5, 8) of Cluster (1)**	**Distance from center (1, 2) of Cluster (2)**	**Point belongs to Cluster**
A1 = (2 , 5)	4.24	3.16	C2
A2 = (5 , 8)	0	7.21	C1
A3 = (7 , 5)	3.6	6.7	C1
A4 = (1 , 2)	7.21	0	C2
A5 =(4 , 9)	1.41	7.62	C1

∴ First cluster contains points → A2 (5 , 8) , A3 (7 , 5) , A5 (4 , 9) &

Second cluster contains points → A1 (2 , 5) , A4 = (1 , 2)

- Calculating the new centroids:

  → The new cluster center is computed by mean of all points contained in that cluster.

  For Cluster (1) :

  ➔ The new centroid of cluster 1 =(5+7+4/3 , 8+5+9/3) = (5.3 , 7.3)

  For Cluster (2) :

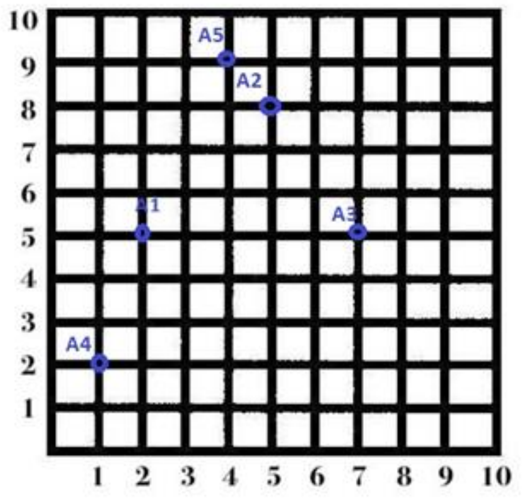
  ➔ The new centroid of cluster 2 =(2+1/2, 5+2/2) = (1.5, 3.5)

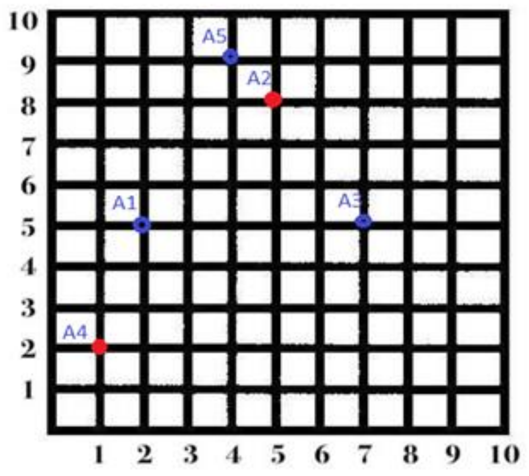
(b) Draw a 10 by 10 space with all the clustered 5 points and the coordinates of the new centroids.

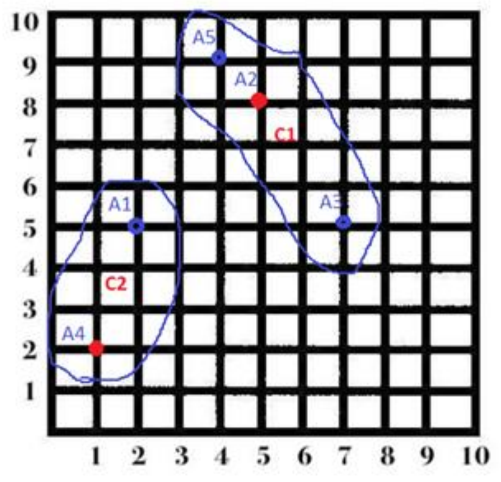
→calculate the centroid for cluster #1 and cluster #2:

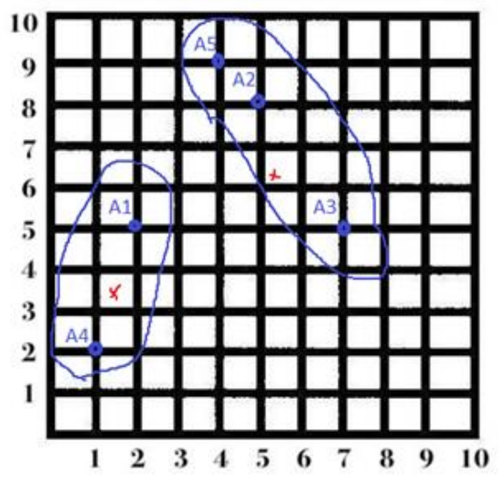
C1= ((5+7+4)/3 , (8+5+9)/3) = (5.3 , 7.3) , C2= ((2+1)/2 , (5+2)/2) =(1.5 , 3.5)

I









c)Calculate the silhouette score and WSS score.

Silhouette score:

→ Silhouette Score = (b-a)/max(a,b)

Where,

- a (cohesion) = average intra-cluster distance (the average distance between each point within a cluster).
- b (separation) = average inter-cluster distance (the average distance between all clusters).

→ First, we will calculate cohesion score for all the data points using Euclidian distance.

- For cluster #1 → A2 = (5 , 8) , A3 = (7 , 5) , A5 = (4 , 9)

$d_{(A2,A3)} = \sqrt{(7-5)^2 + (5-8)^2} = 3.61$

$d_{(A2,A5)} = \sqrt{(4-5)^2 + (9-8)^2} = 1.41$

$d_{(A3,A5)} = \sqrt{(4-7)^2 + (9-5)^2} = 5$

Cohesion for point A2 (5, 8)→ a = (3.61 + 1.41)/2 = 2.51

Cohesion for point A3 (7, 5)→ a= (3.61 + 5)/2 = 4.30

Cohesion for point A5 (4, 9)→ a = (1.41 + 5)/2 = 3.21

- For cluster #2 → A1 =(2 , 5) , A4 = (1 , 2)

d $_{(A1,A4)}$ = $\sqrt{(1-2)^2+(2-5)^2}$ = 3.16

Cohesion for point A1 (2, 5)→ a = 3.16

Cohesion for point A4 (1, 2)→ a = 3.16

→Then, we will calculate Separation score for all the data points using Euclidian distance.

- calculate the distance between these points A2 =(5, 8) , A3 = (7 , 5) , A5 = (4 , 9) in the first cluster and other points A1 =(2, 5) , A4 = (1, 2) in other cluster.

d $_{(A2,A1)}$ = $\sqrt{(2-5)^2+(5-8)^2}$ = 4.24

d $_{(A3,A1)}$ = $\sqrt{(2-7)^2+(5-5)^2}$ = 5

d $_{(A5,A1)}$ = $\sqrt{(2-4)^2+(5-9)^2}$ = 4.47

d $_{(A2,A4)}$ = $\sqrt{(1-5)^2+(2-8)^2}$ = 7.21

d $_{(A3,A4)}$ = $\sqrt{(1-7)^2+(2-5)^2}$ = 6.71

d $_{(A5,A4)}$ = $\sqrt{(1-4)^2+(2-9)^2}$ = 7.62

Separation for point A1 (2, 5)→ b = (4.27+5+4.47)/3 = 4.57

Separation for point A2 (5, 8)→ b = (4.24+7.21)/2 = 5.73

Separation for point A3 (7, 5)→ b = (5+6.71)/2 = 5.86

Separation for point A4 (1, 2)→ b = (7.21+6.71+7.62)/3 = 7.18

Separation for point A5 (4, 9)→ b = (4.47+7.62)/2 = 6.05

→ We show all the results for cohesion score and separation score in this table.

Given Points	Cohesion Score (a)	Separation Score (b)
A1= (2, 5)	a= 3.16	b= 4.57
A2= (5, 8)	a= 2.51	b= 5.73
A3= (7, 5)	a= 4.30	b= 5.86
A4= (1, 2)	a= 3.16	b= 7.18
A5= (4, 9)	a= 3.21	b= 6.05

∵Silhouette Score = (b-a)/max(a,b)

∴S_{A1} = (4.57 – 3.16) / max(3.16, 4.57) = 0.31

, S_{A2} = (5.73 – 2.51) / max(2.51, 5.73) = 0.56

, S_{A3} = (5.86 – 4.30) / max(4.30, 5.86) = 0.27

, S_{A4} = (7.18 – 3.16) / max(3.16, 7.18) = 0.56

, S_{A5} = (6.05 – 3.21) / max(3.21, 6.05) = 0.47

∴ The overall Silhouette Score = 1/5 (S_{A1} +S_{A2}+ S_{A3} +S_{A4} +S_{A5})

= 1/5 (0.31 + 0.56 + 0.27 + 0.56 + 0.47)

= 0.434

__WSS score:__

- C1 = (5.3 , 7.3)   ,,,   C2 = (1.5 , 3.5)

$$WSS = \sum_{i=1}^{m} (x_i - c_i)^2$$

$$WSS = \sum^{m} (x_i - c_i)^2$$

$\rightarrow$ we will measure the distance between each point in the cluster and the centroid of the same cluster.

- For cluster #1 $\rightarrow$ A2 (5,8) , A3 (7,5) , A5 (4,9)

$d_{(A2,C1)} = (5.3 - 5)^2 + (7.3 - 8)^2 = 0.58$

$d_{(A3,C1)} = (5.3 - 7)^2 + (7.3 - 5)^2 = 8.18$

$d_{(A3,C1)} = (5.3 - 4)^2 + (7.3 - 9)^2 = 4.58$

WSS for cluster 1 = 0.58 + 8.18 + 4.58 =13.34

- For cluster #2 $\rightarrow$ A1 (2, 5) , A4 (1,2)

$d_{(A1,C2)} = (1.5 - 2)^2 + (3.5 - 5)^2 = 2.5$

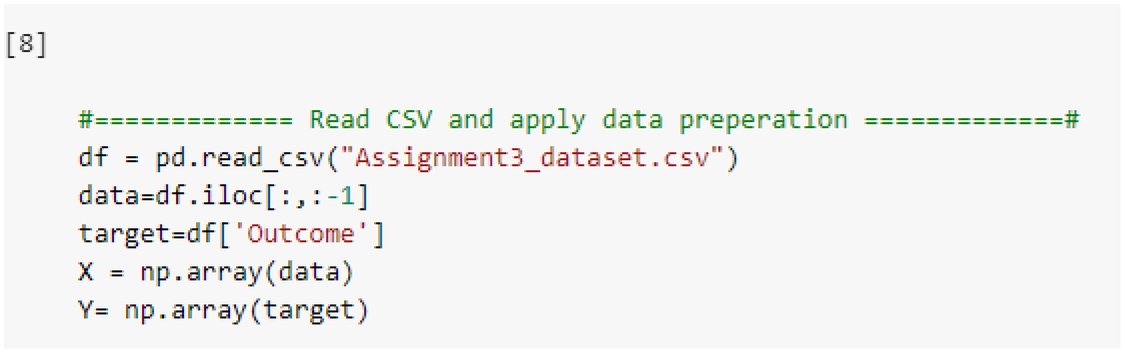
$d_{(A4,C2)} = (1.5 - 1)^2 + (3.5 - 2)^2 = 2.5$

WSS for cluster 2 = 2.5 + 2.5 = 5

The overall WSS = 13.14 + 5 = 18.34

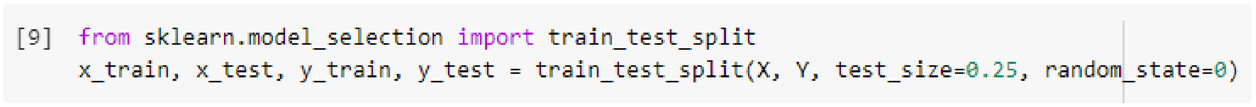
Part 2 : Programming

Loading Data:



```
[8]

    #============= Read CSV and apply data preperation =============#
    df = pd.read_csv("Assignment3_dataset.csv")
    data=df.iloc[:,:-1]
    target=df['Outcome']
    X = np.array(data)
    Y= np.array(target)
```

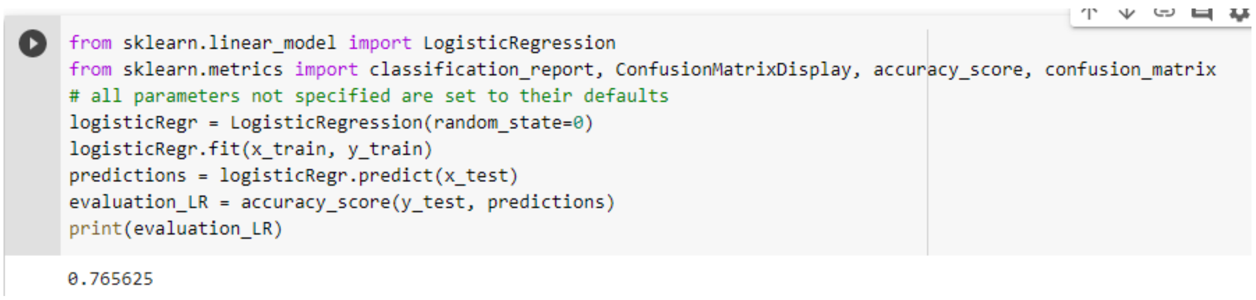
Using train test split function in scikitlearn to split the dataset into a training set, a testing set.



```
[9]  from sklearn.model_selection import train_test_split
     x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.25, random_state=0)
```

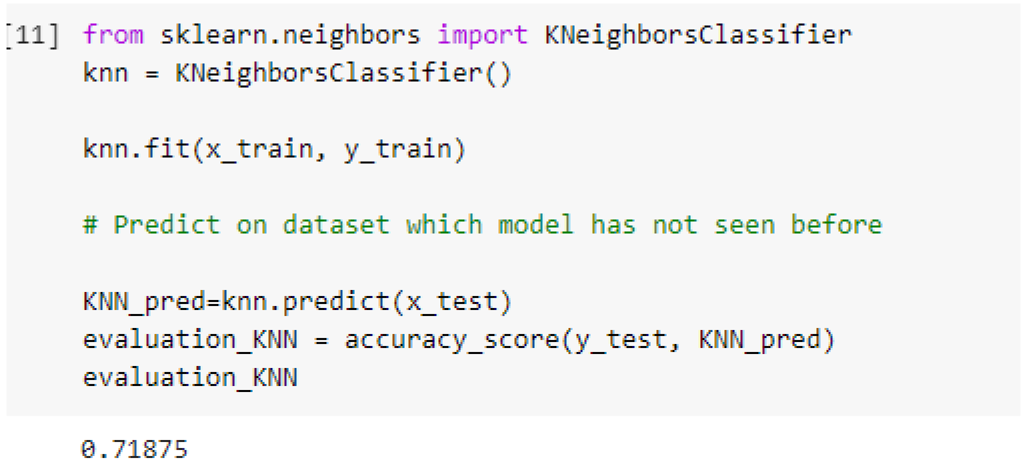
Provide the accuracy of LR and K-NN classifier as baseline performances.

-LR



```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, ConfusionMatrixDisplay, accuracy_score, confusion_matrix
# all parameters not specified are set to their defaults
logisticRegr = LogisticRegression(random_state=0)
logisticRegr.fit(x_train, y_train)
predictions = logisticRegr.predict(x_test)
evaluation_LR = accuracy_score(y_test, predictions)
print(evaluation_LR)

0.765625
```

-KNN

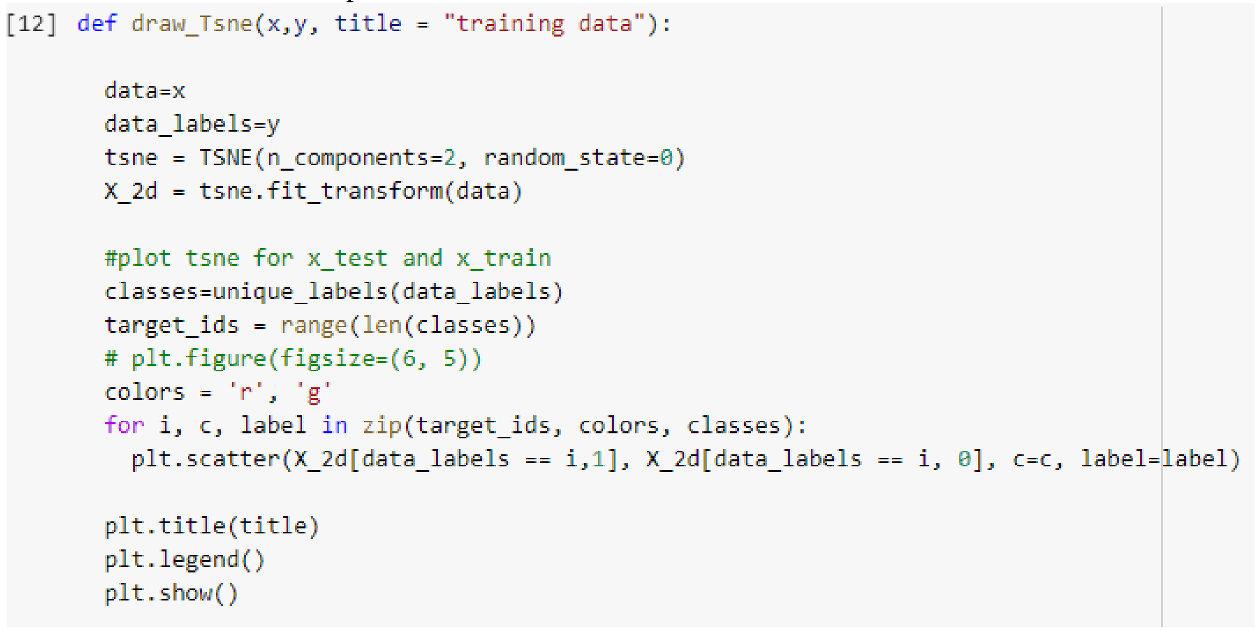


```
[11]  from sklearn.neighbors import KNeighborsClassifier
      knn = KNeighborsClassifier()

      knn.fit(x_train, y_train)

      # Predict on dataset which model has not seen before

      KNN_pred=knn.predict(x_test)
      evaluation_KNN = accuracy_score(y_test, KNN_pred)
      evaluation_KNN

      0.71875
```

Getting Acurrecy=76% for LR & 71% for KNN

b)

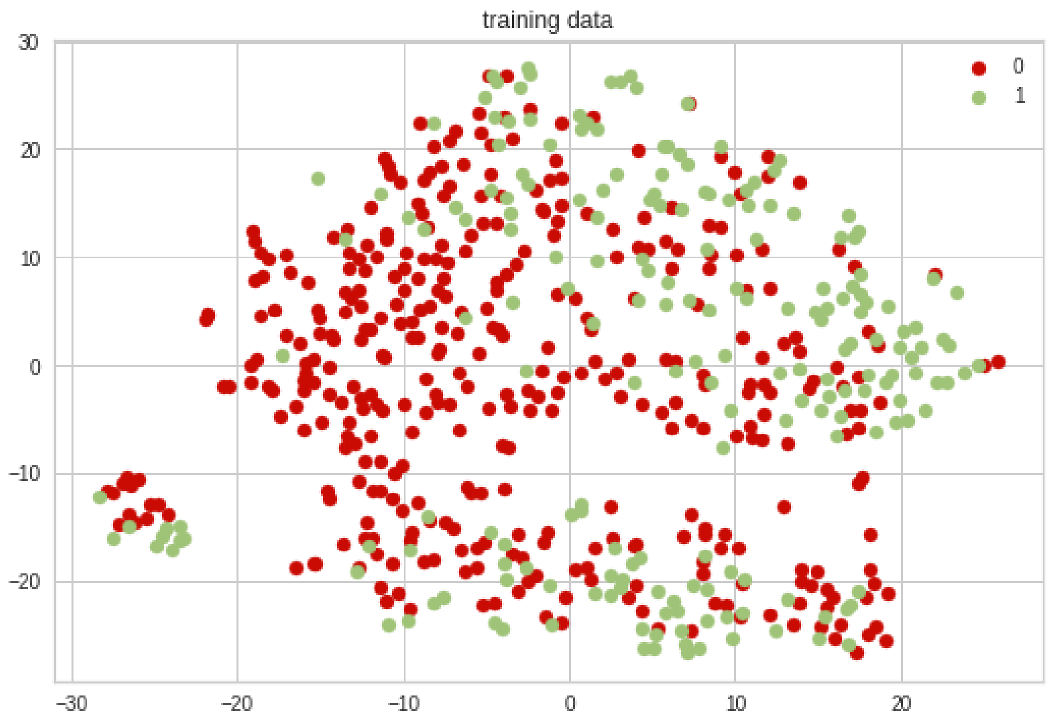
.-This Function draw a TSNE plot



```
[12] def draw_Tsne(x,y, title = "training data"):

       data=x
       data_labels=y
       tsne = TSNE(n_components=2, random_state=0)
       X_2d = tsne.fit_transform(data)

       #plot tsne for x_test and x_train
       classes=unique_labels(data_labels)
       target_ids = range(len(classes))
       # plt.figure(figsize=(6, 5))
       colors = 'r', 'g'
       for i, c, label in zip(target_ids, colors, classes):
         plt.scatter(X_2d[data_labels == i,1], X_2d[data_labels == i, 0], c=c, label=label)

       plt.title(title)
       plt.legend()
       plt.show()
```

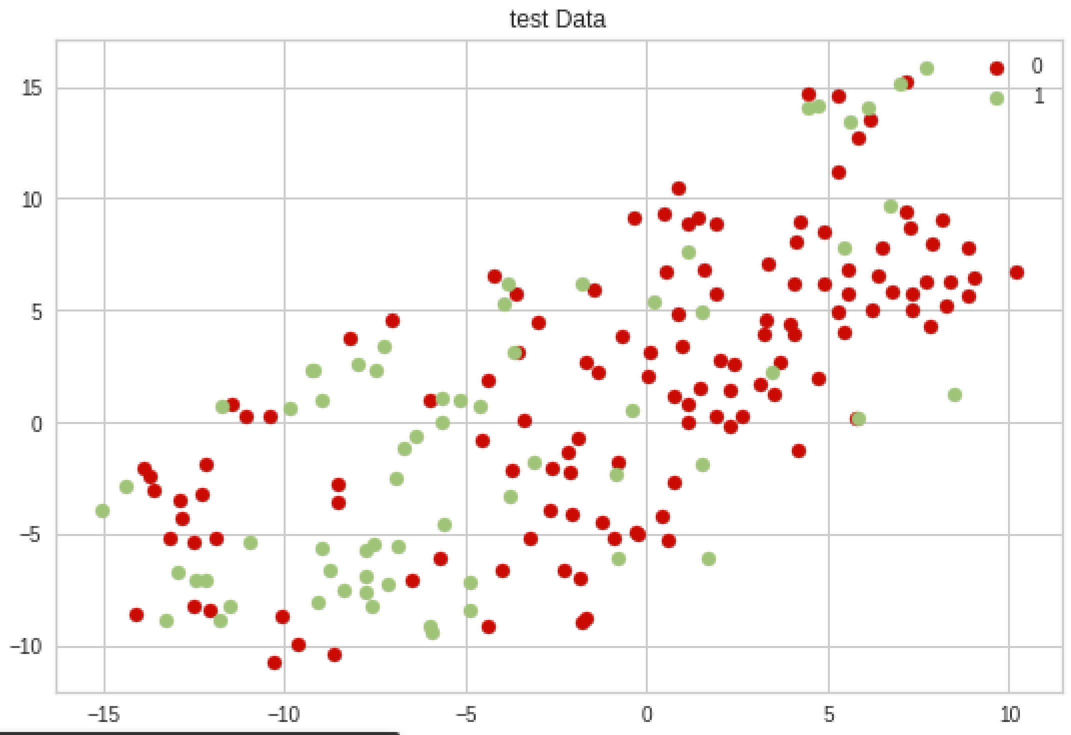
Provide 2D TSNE plots, one for the training set

```
draw_Tsne(x_train,y_train,title="training data")
```



Provide 2D TSNE plots, one for the testing set

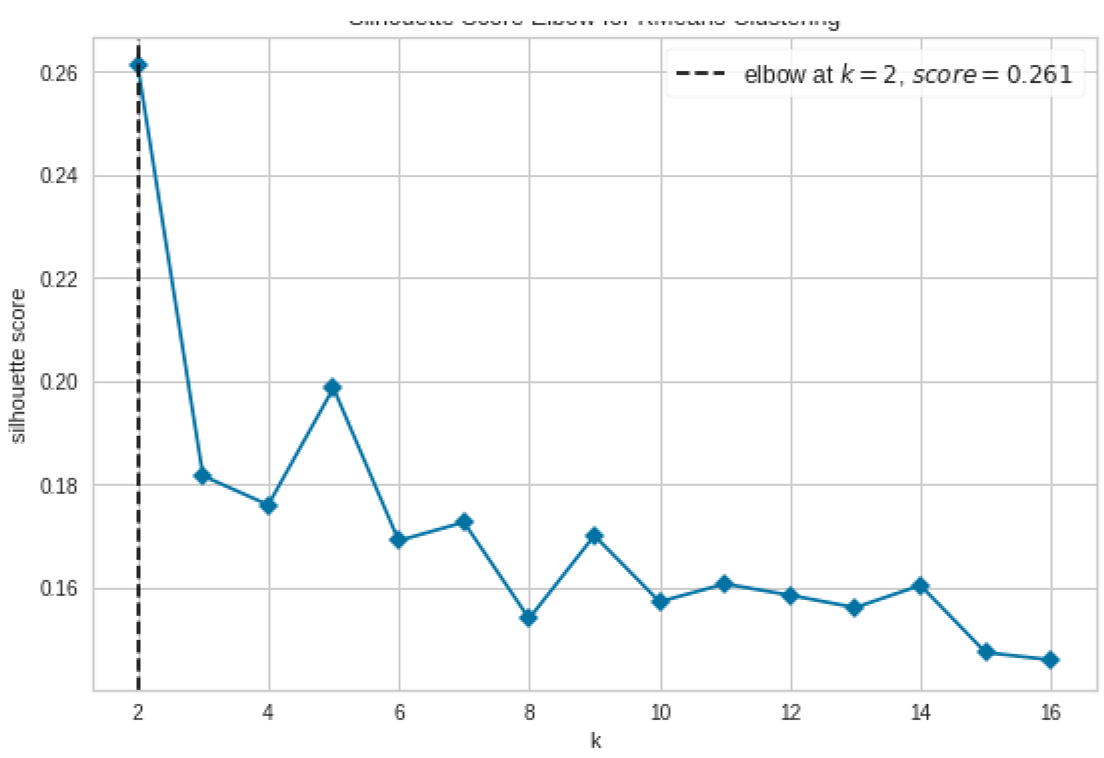
```
[15] draw_Tsne(x_test,y_test,title = "test Data")
```



2- The best number of cluster for k-means clustering algorithm

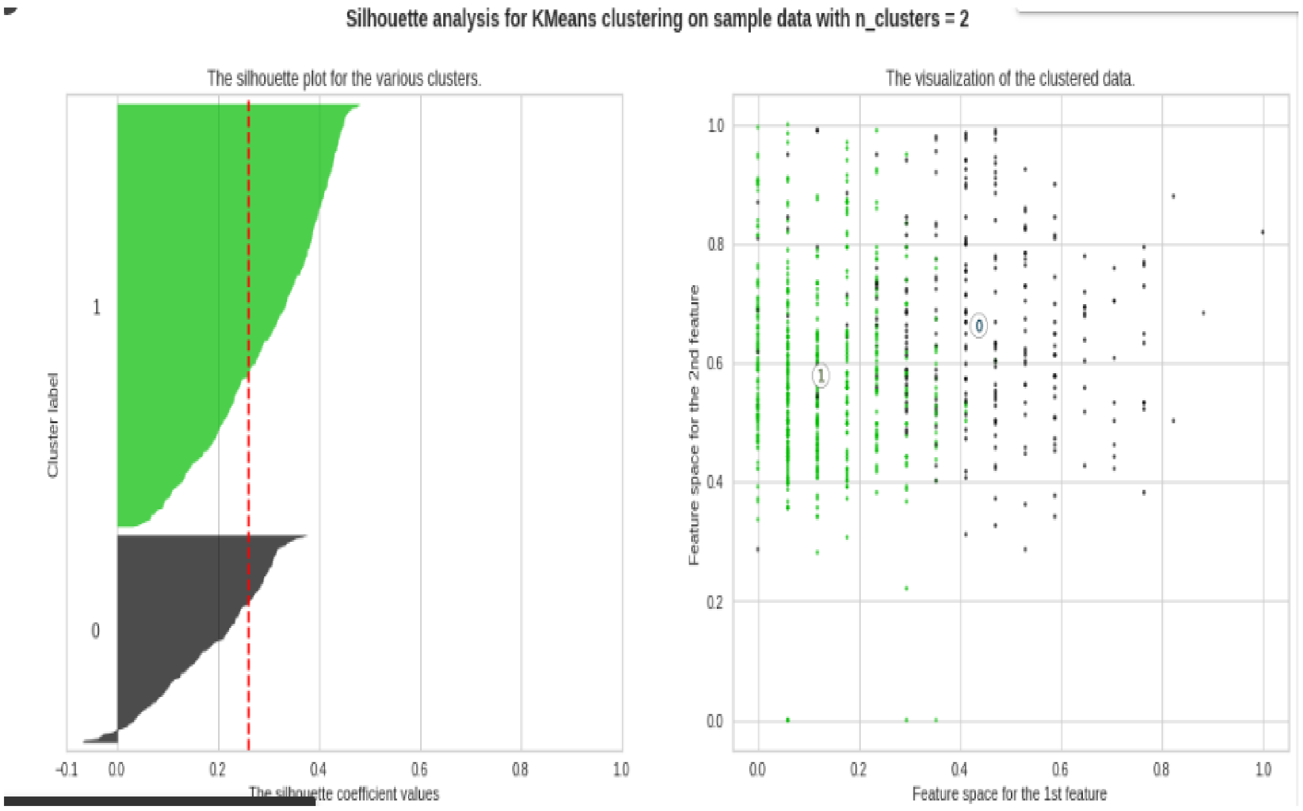
(a) Plotting the silhouette score vs the number of clusters.

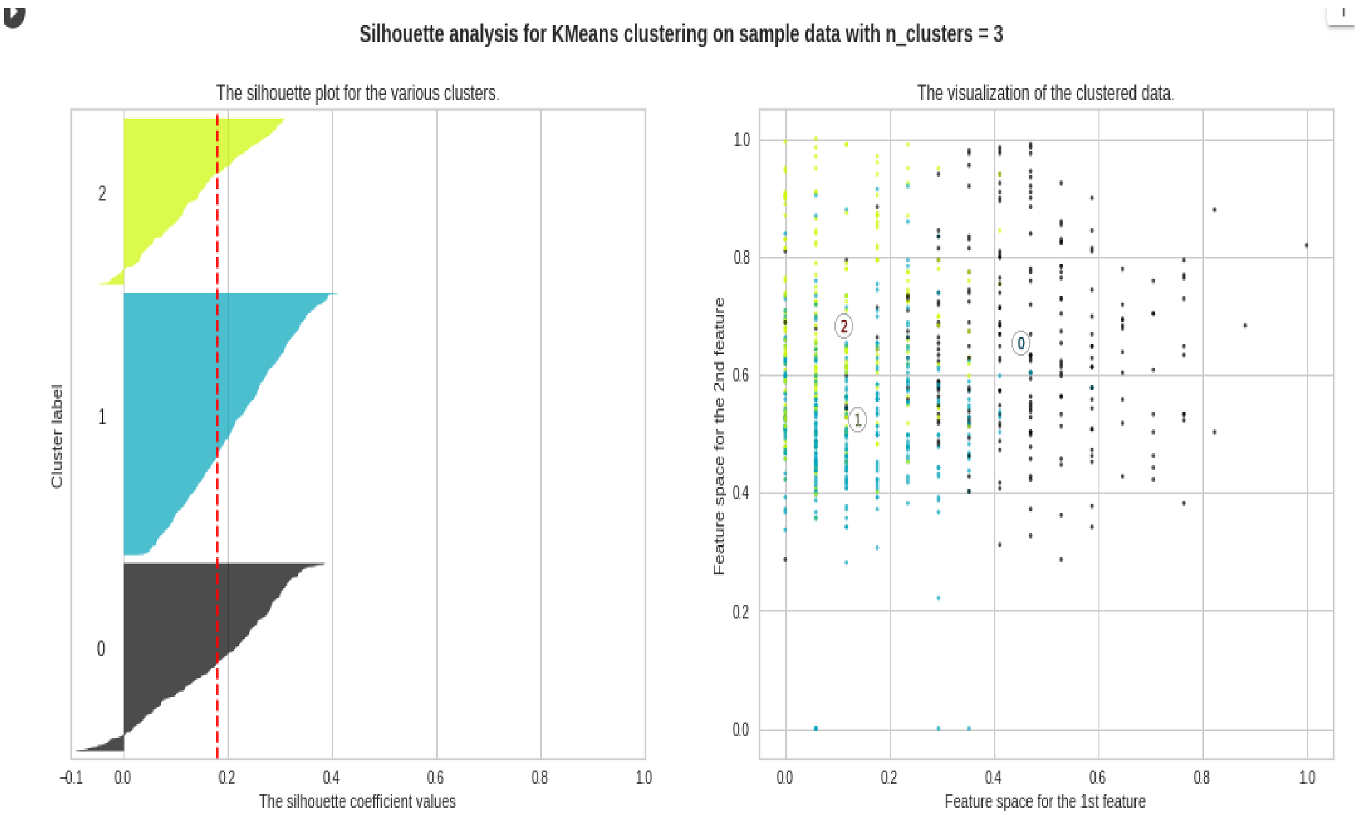
```
from yellowbrick.cluster.elbow import kelbow_visualizer
model = kelbow_visualizer(KMeans(random_state=0), X, k=(2,17),metric='silhouette',timings=False)

pass
```

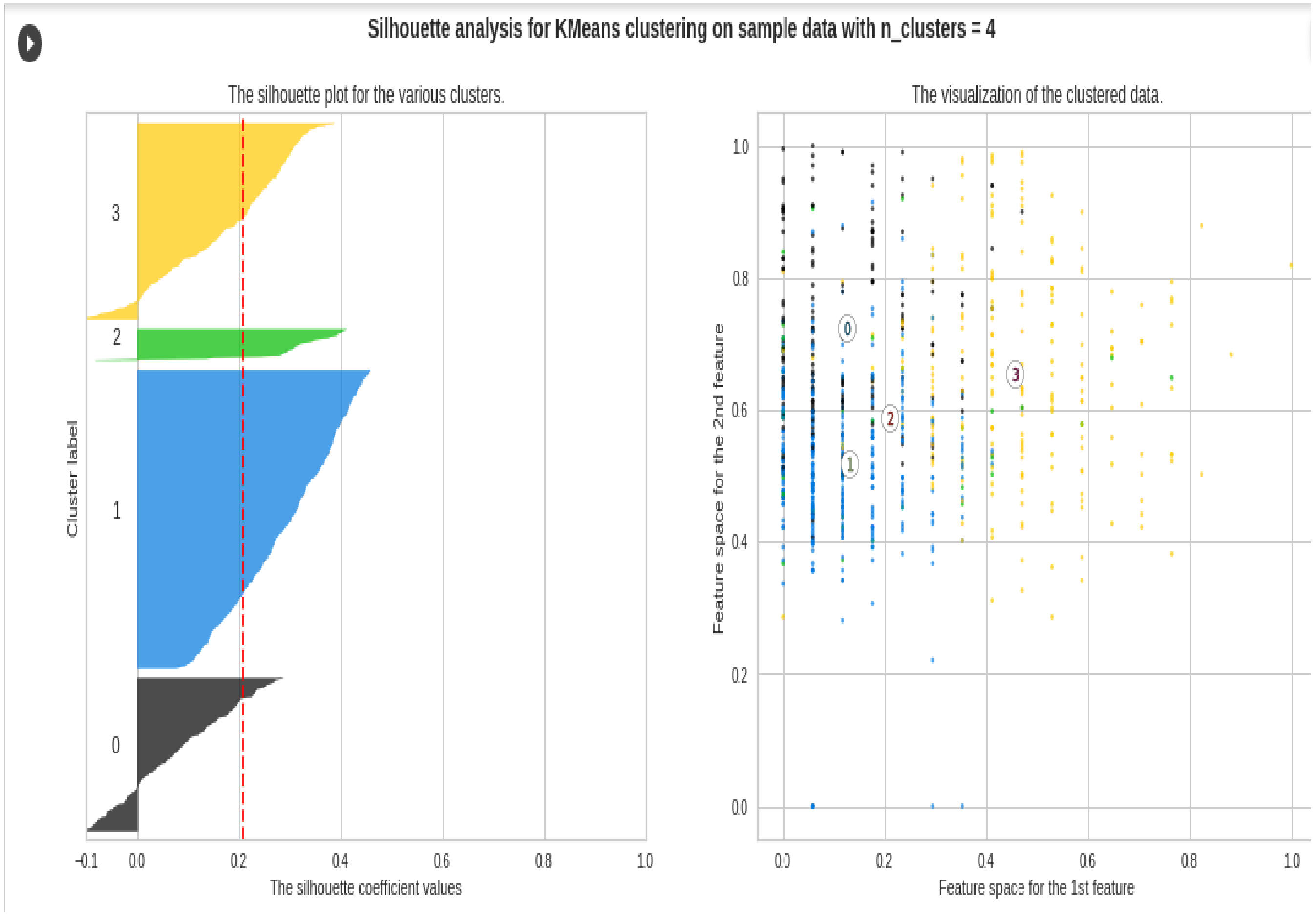


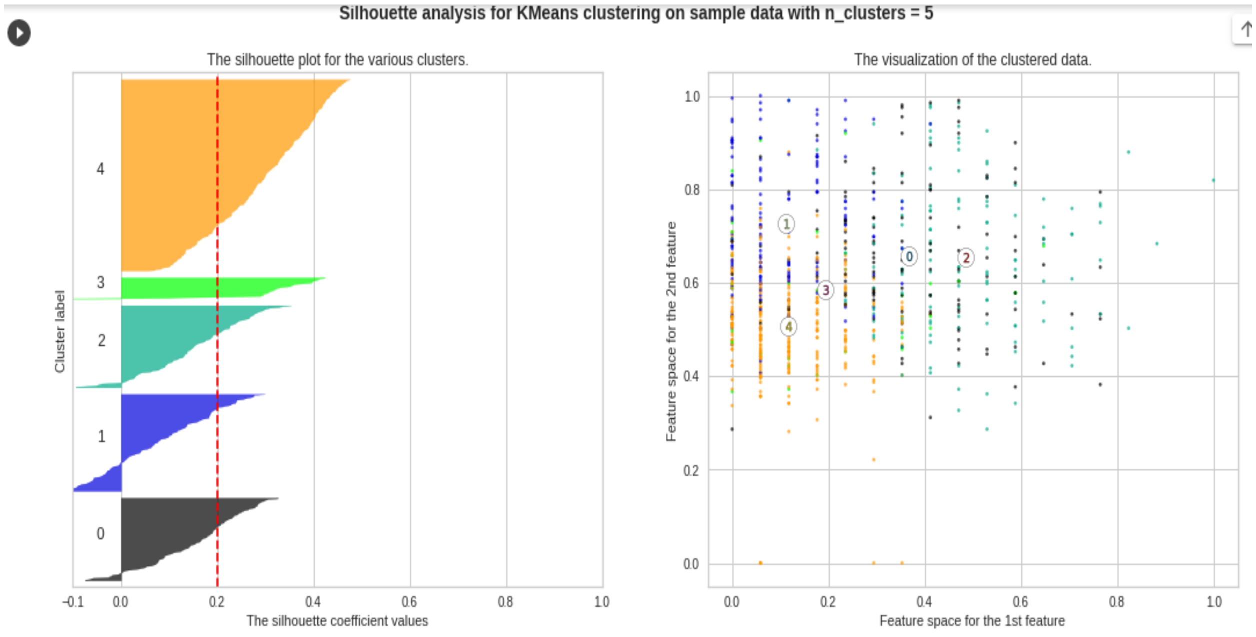
From this plot we found the best K = 2 , slihouette score 0.261

```
For n_clusters = 2 The average silhouette_score is : 0.26114611150604655
For n_clusters = 3 The average silhouette_score is : 0.18151103414798983
For n_clusters = 4 The average silhouette_score is : 0.20808119301398584
For n_clusters = 5 The average silhouette_score is : 0.20075741354829582
```


Silhouette analysis for KMeans clustering on sample data with n_clusters = 2
The silhouette plot for the various clusters.
Cluster label
1
0
-0.1
0.0
0.2
0.4
0.6
0.8
1.0
The silhouette coefficient values
The visualization of the clustered data.
Feature space for the 2nd feature
Feature space for the 1st feature
0
1


Silhouette analysis for KMeans clustering on sample data with n_clusters = 3
The silhouette plot for the various clusters.
Cluster label
2
1
0
-0.1
0.0
0.2
0.4
0.6
0.8
1.0
The silhouette coefficient values
The visualization of the clustered data.
Feature space for the 2nd feature
Feature space for the 1st feature
2
0
1


Silhouette analysis for KMeans clustering on sample data with n_clusters = 4
The silhouette plot for the various clusters.
Cluster label
3
2
1
0
-0.1
0.0
0.2
0.4
0.6
0.8
1.0
The silhouette coefficient values
The visualization of the clustered data.
Feature space for the 2nd feature
Feature space for the 1st feature
0
1
2
3


Silhouette analysis for KMeans clustering on sample data with n_clusters = 5
The silhouette plot for the various clusters.
Cluster label
4
3
2
1
0
-0.1
0.0
0.2
0.4
0.6
0.8
1.0
The silhouette coefficient values
The visualization of the clustered data.
Feature space for the 2nd feature
Feature space for the 1st feature
0
1
2
3
4

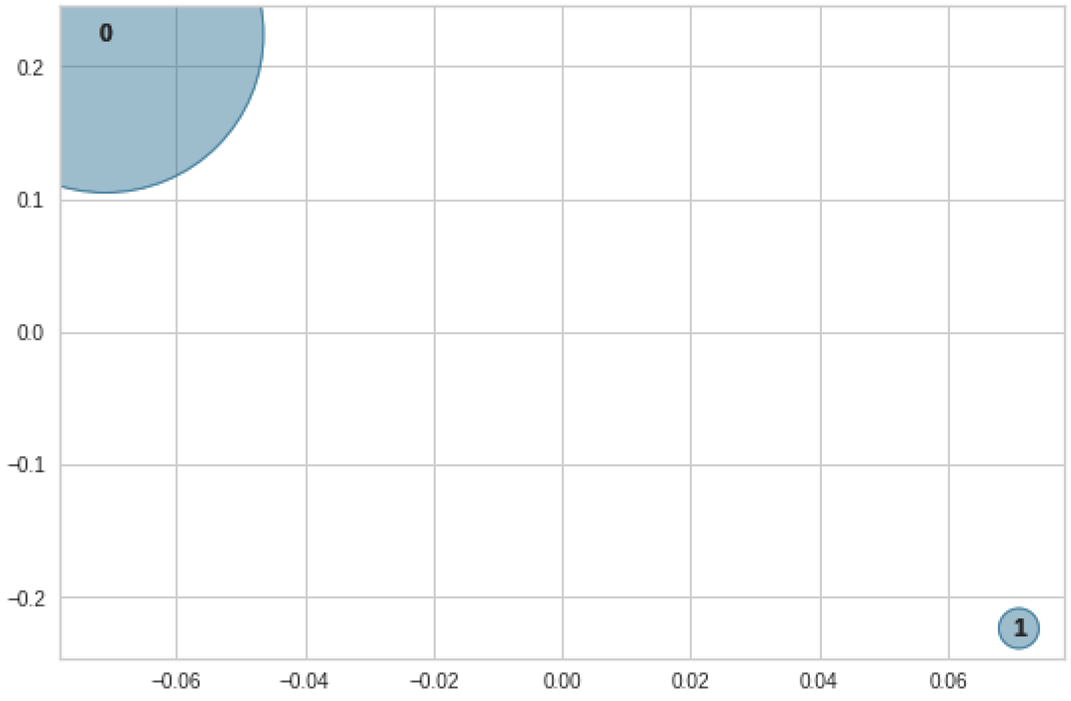
b) Determine the optimal number of clusters for k-Means

The optimal number of clusters is 2 for k means

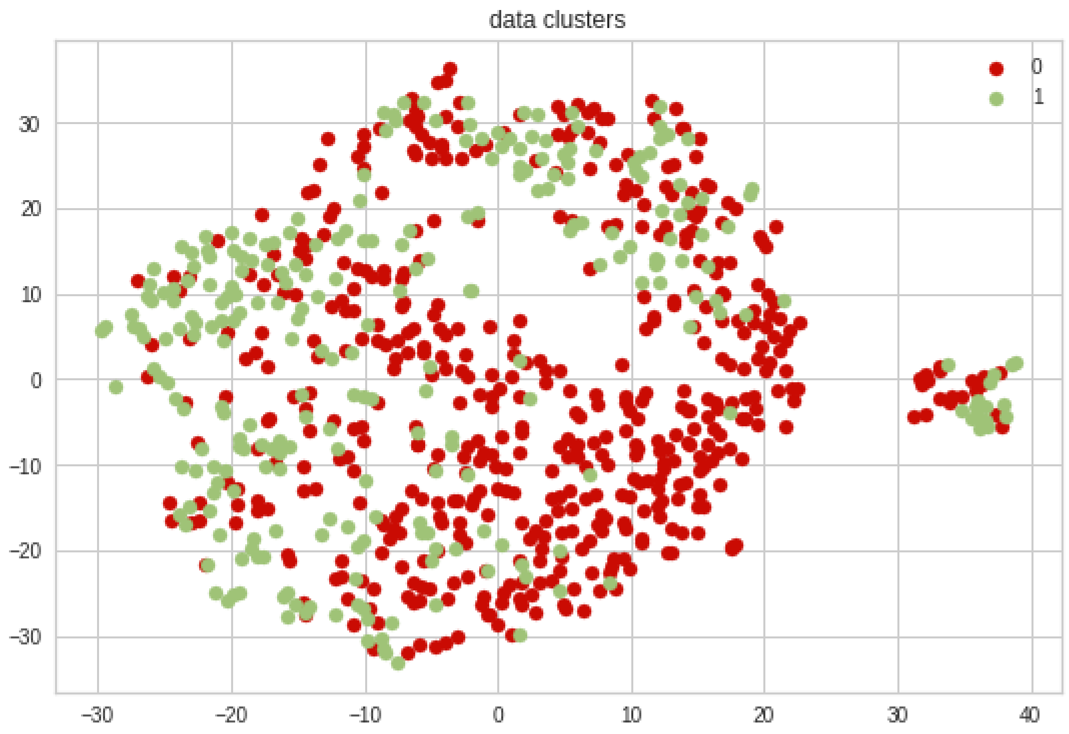
c) Plot the clustered data with optimum number of clusters.

```
[ ] model_kmean = KMeans(2,random_state=0)
    visualizer = InterclusterDistance(model_kmean,random_state=0)
    visualizer.fit(X)
    visualizer.draw()
```

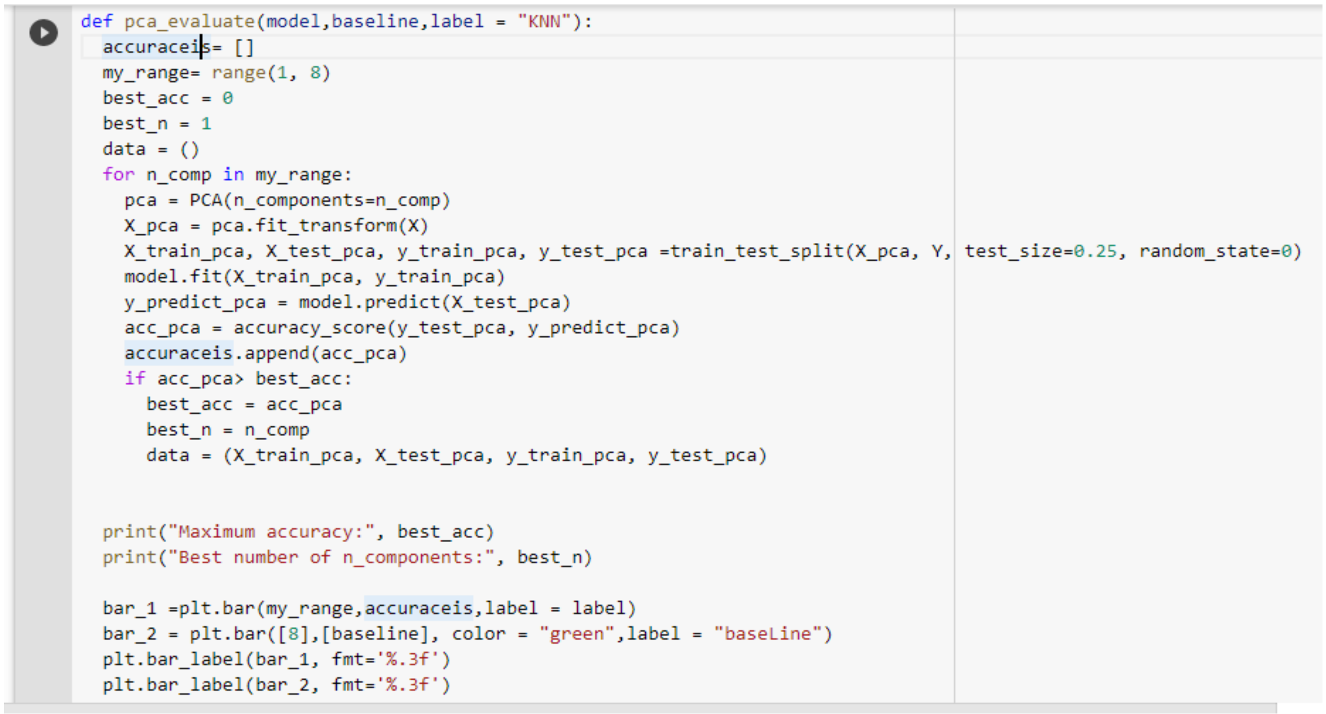
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2eac271050>
```



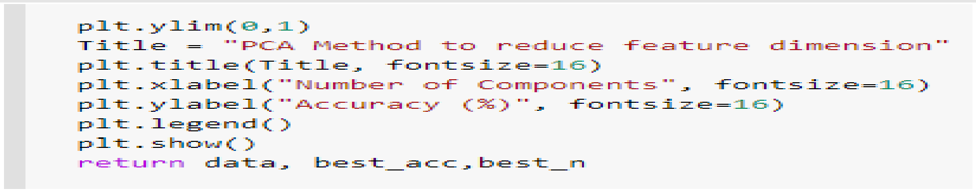
```
[ ]  draw_Tsne(X,Y,"data clusters")
```



3)Apply the Dimensionality Reduction (DR) methods]

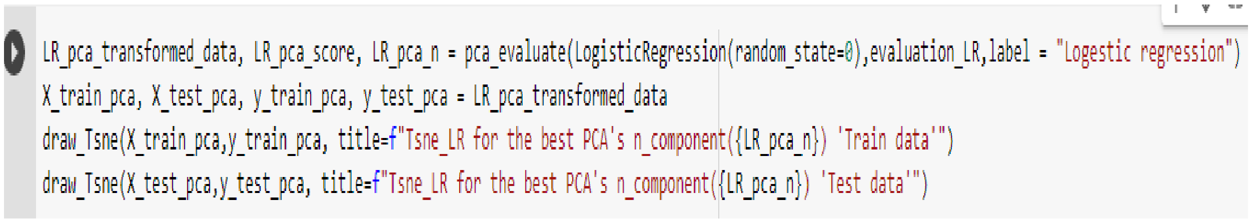


```
def pca_evaluate(model,baseline,label = "KNN"):
  accuraceis= []
  my_range= range(1, 8)
  best_acc = 0
  best_n = 1
  data = ()
  for n_comp in my_range:
    pca = PCA(n_components=n_comp)
    X_pca = pca.fit_transform(X)
    X_train_pca, X_test_pca, y_train_pca, y_test_pca =train_test_split(X_pca, Y, test_size=0.25, random_state=0)
    model.fit(X_train_pca, y_train_pca)
    y_predict_pca = model.predict(X_test_pca)
    acc_pca = accuracy_score(y_test_pca, y_predict_pca)
    accuraceis.append(acc_pca)
    if acc_pca> best_acc:
      best_acc = acc_pca
      best_n = n_comp
      data = (X_train_pca, X_test_pca, y_train_pca, y_test_pca)


  print("Maximum accuracy:", best_acc)
  print("Best number of n_components:", best_n)

  bar_1 =plt.bar(my_range,accuraceis,label = label)
  bar_2 = plt.bar([8],[baseline], color = "green",label = "baseLine")
  plt.bar_label(bar_1, fmt='%.3f')
  plt.bar_label(bar_2, fmt='%.3f')
```

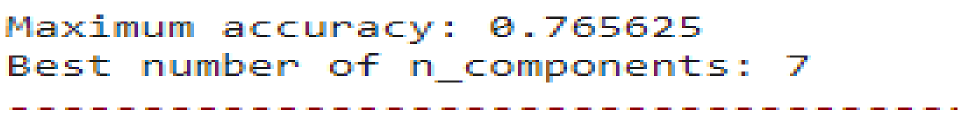


```
  plt.ylim(0,1)
  Title = "PCA Method to reduce feature dimension"
  plt.title(Title, fontsize=16)
  plt.xlabel("Number of Components", fontsize=16)
  plt.ylabel("Accuracy (%)", fontsize=16)
  plt.legend()
  plt.show()
  return data, best_acc,best_n
```

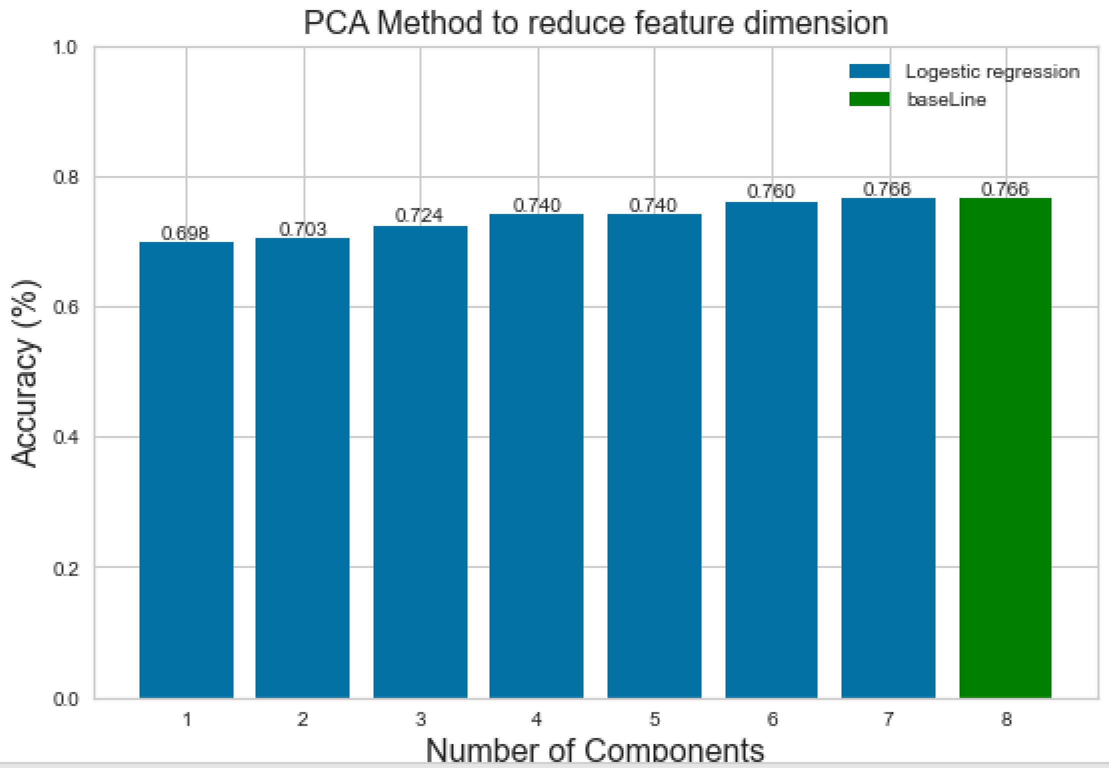
-The best value of n components based on test accuracies for LR=7 .

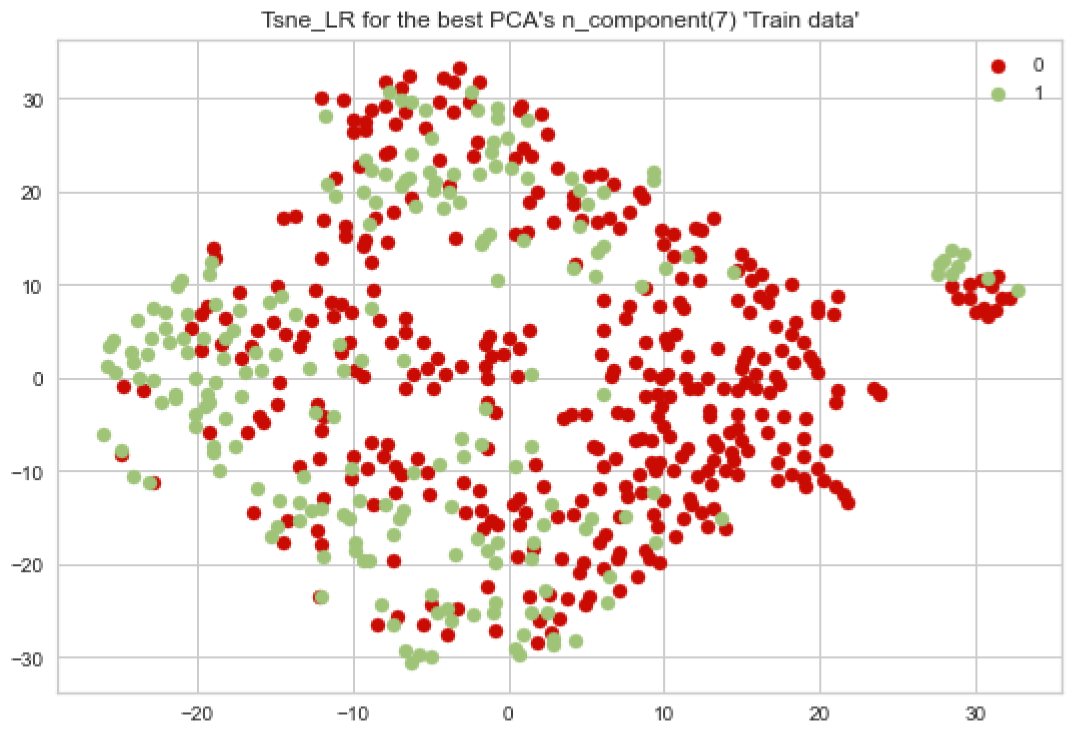


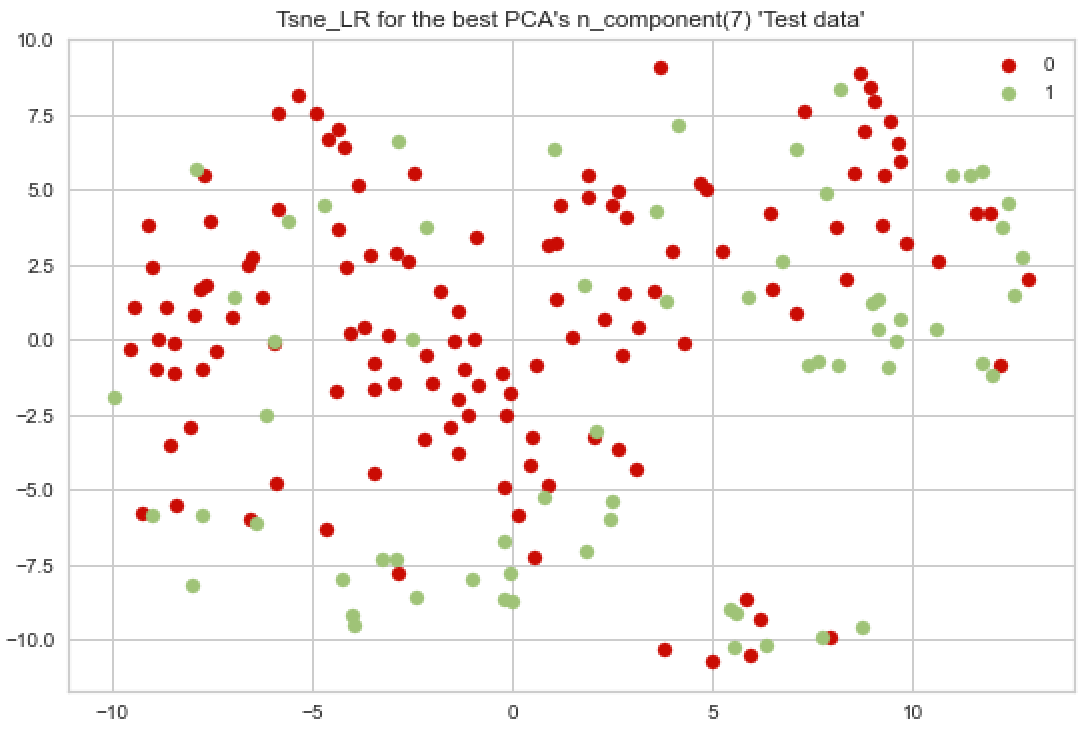
```
LR_pca_transformed_data, LR_pca_score, LR_pca_n = pca_evaluate(LogisticRegression(random_state=0),evaluation_LR,label = "Logestic regression")
X_train_pca, X_test_pca, y_train_pca, y_test_pca = LR_pca_transformed_data
draw_Tsne(X_train_pca,y_train_pca, title=f"Tsne_LR for the best PCA's n_component({LR_pca_n}) 'Train data'")
draw_Tsne(X_test_pca,y_test_pca, title=f"Tsne_LR for the best PCA's n_component({LR_pca_n}) 'Test data'")
```



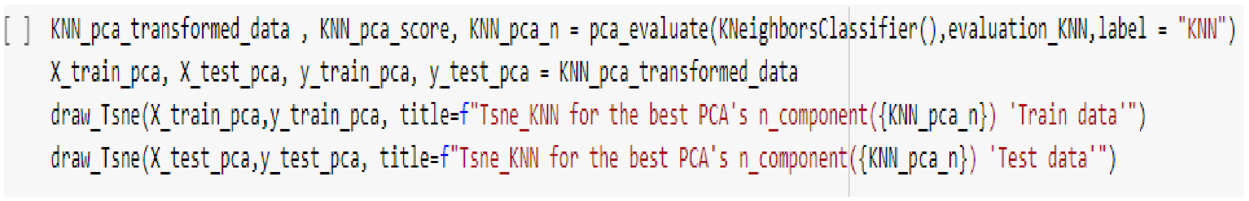
```
Maximum accuracy: 0.765625
Best number of n_components: 7
-----------------------------------------
```


PCA Method to reduce feature dimension
Logestic regression
baseLine
Accuracy (%)
Number of Components
0.698
0.703
0.724
0.740
0.740
0.760
0.766
0.766


Tsne_LR for the best PCA's n_component(7) 'Train data'
0
1

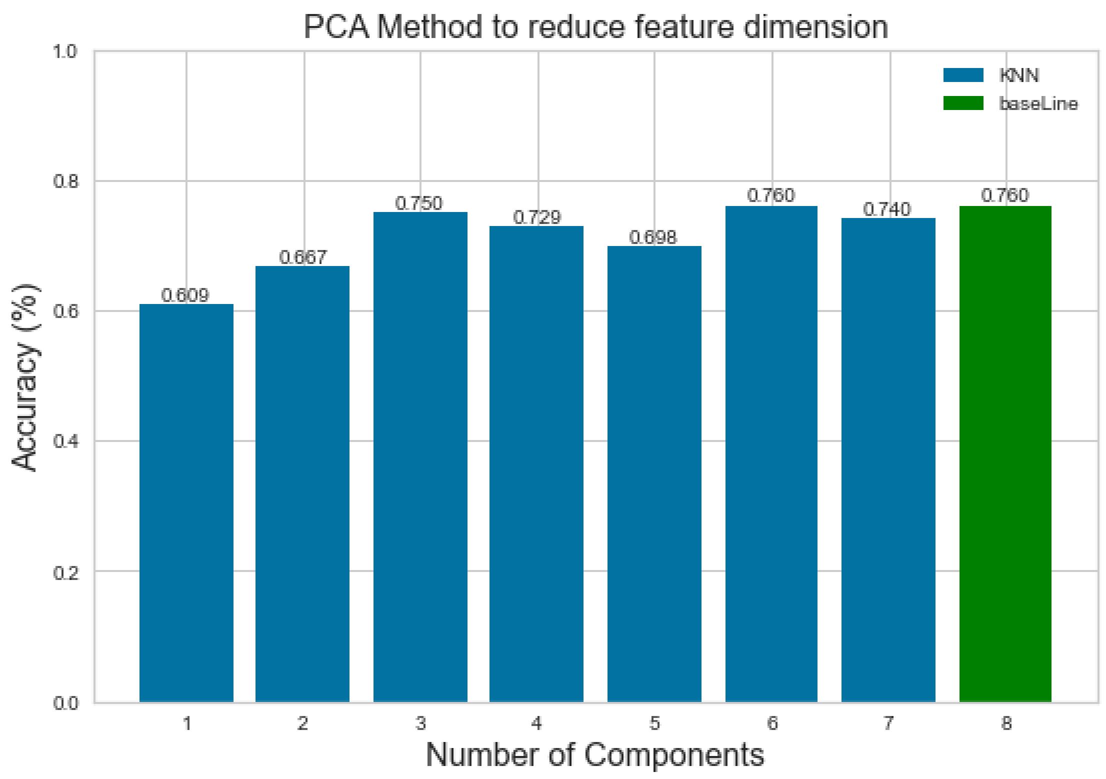


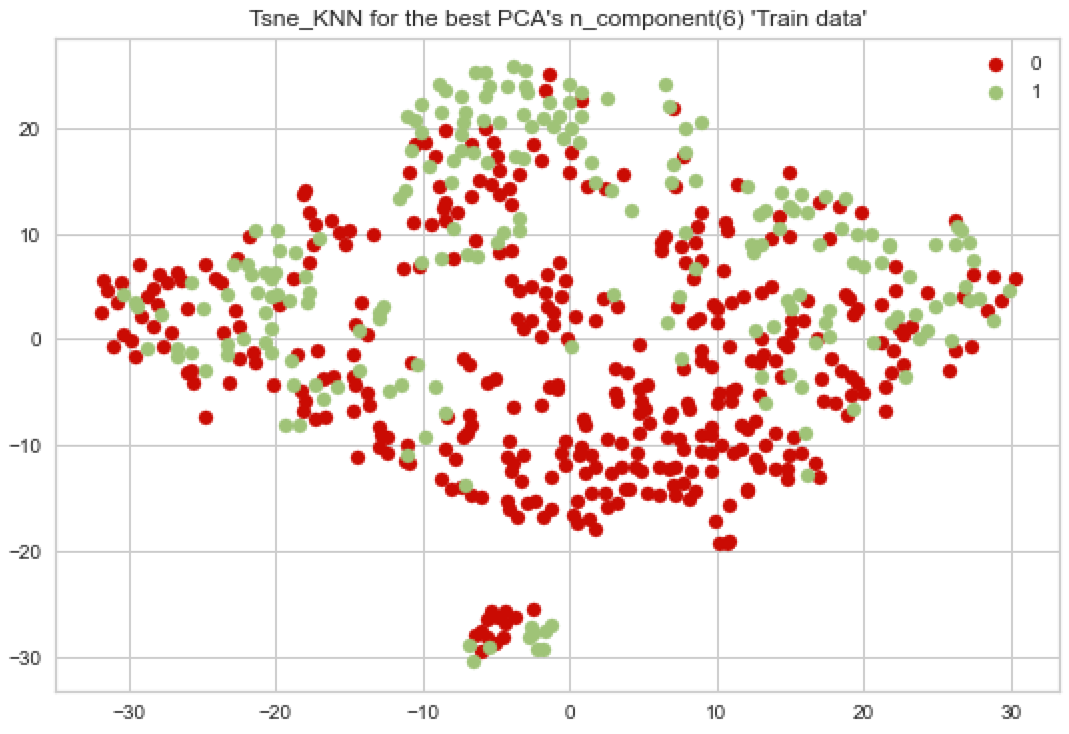
-The best value of n components based on test accuracies for KNN=6 .



```
[ ] KNN_pca_transformed_data , KNN_pca_score, KNN_pca_n = pca_evaluate(KNeighborsClassifier(),evaluation_KNN,label = "KNN")
    X_train_pca, X_test_pca, y_train_pca, y_test_pca = KNN_pca_transformed_data
    draw_Tsne(X_train_pca,y_train_pca, title=f"Tsne_KNN for the best PCA's n_component({KNN_pca_n}) 'Train data'")
    draw_Tsne(X_test_pca,y_test_pca, title=f"Tsne_KNN for the best PCA's n_component({KNN_pca_n}) 'Test data'")
```

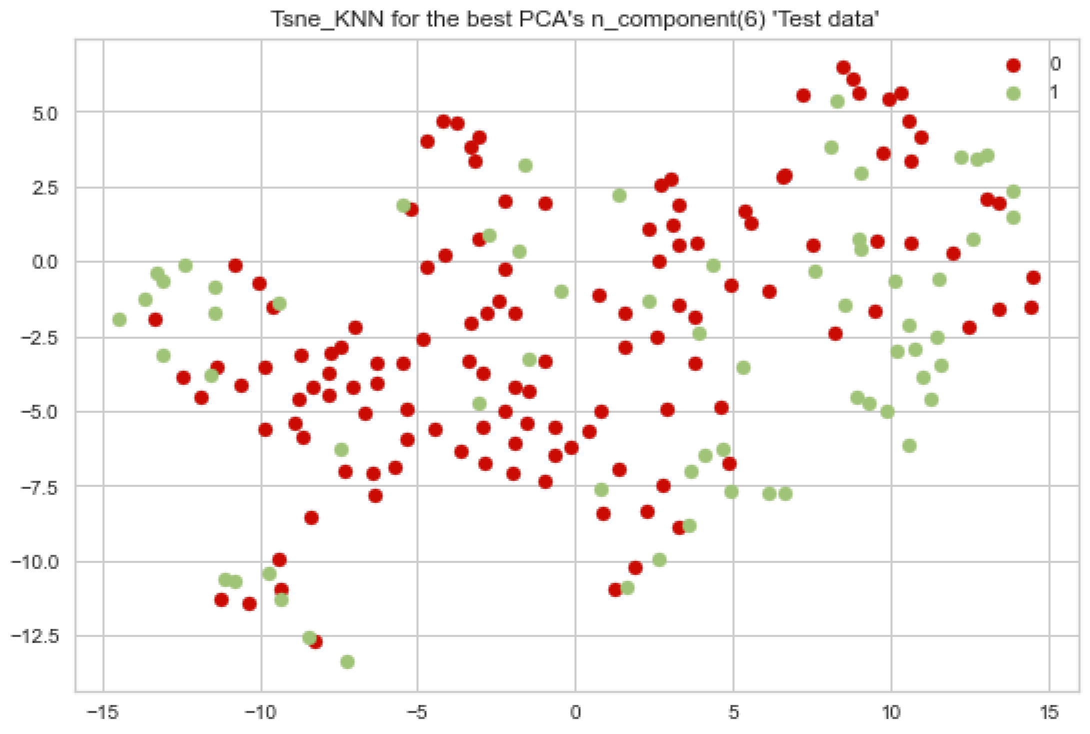
```
Maximum accuracy: 0.7604166666666666
Best number of n_components: 6
```





Tsne_KNN for the best PCA's n_component(6) 'Test data'

]



4) Find the best number of features based on both, the LR and K-NN classifiers' test accuracies.

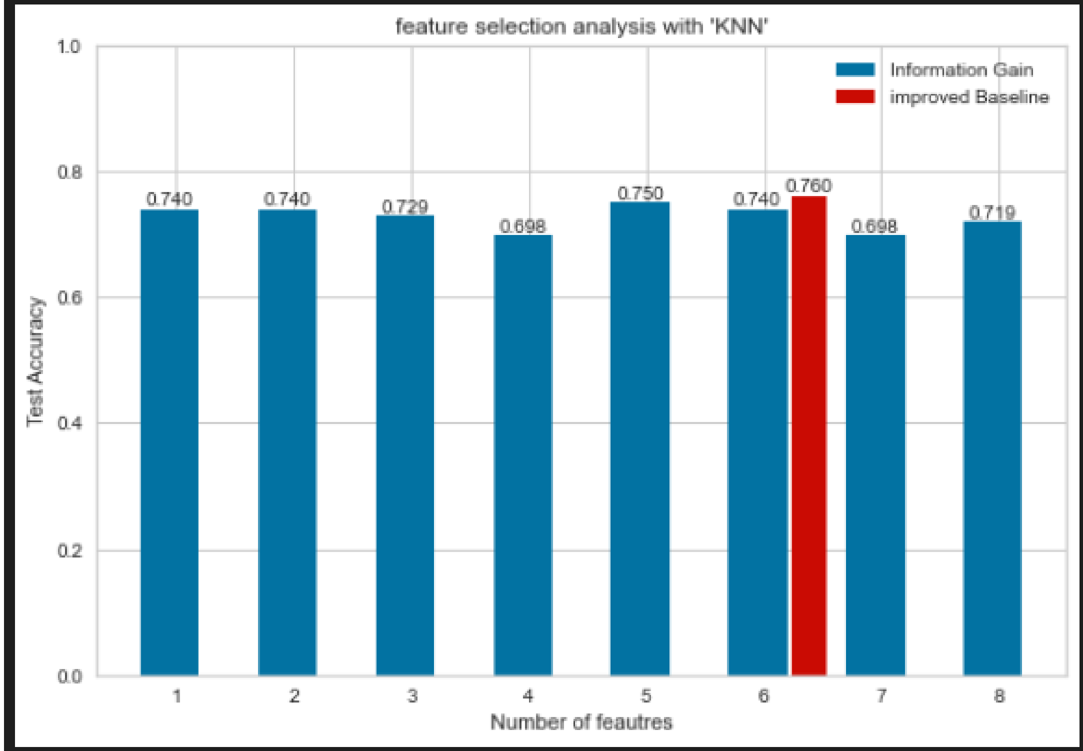
- Setting the improved baseline from the previouse step

```
[ ]  evaluation_KNN = KNN_pca_score
     evaluation_LR = LR_pca_score
     KNN_imp_baseline_location = KNN_pca_n
     LR_imp_baseline_location = LR_pca_n
```

(a) Filter Methods with Information gain and ANOVA

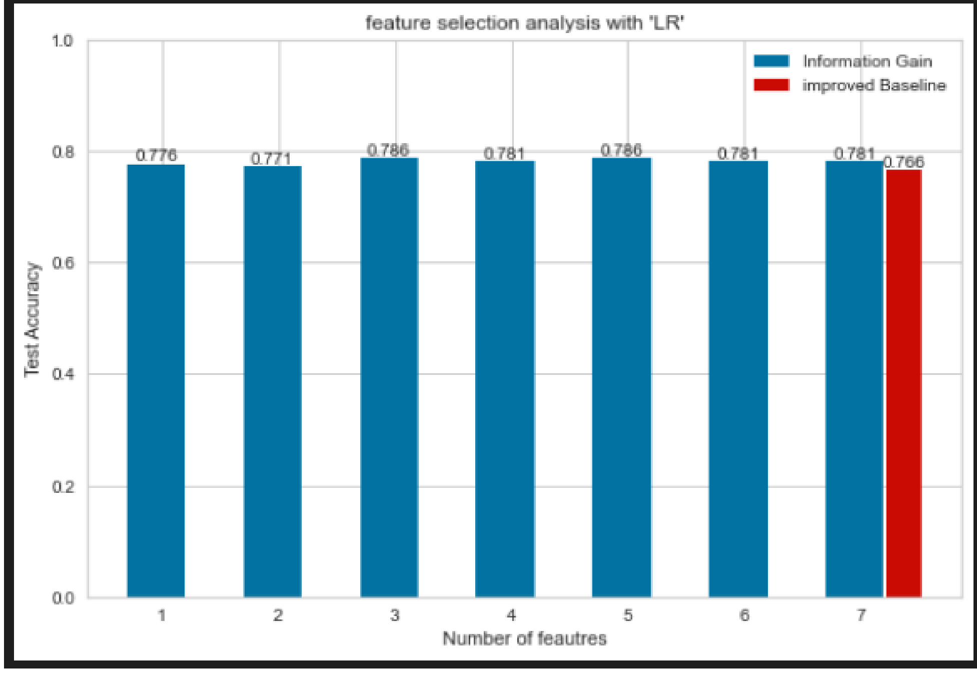
- First with KNN:

```
maximum score with anova = 0.7447916666666666 , and the number of features = 7
maximum score with Information Gain = 0.75 , and the number of features = 5
----------------------------------------------------------------------------------
the best Method is Information Gain with accuracy = 0.75
the number of features = 5
```



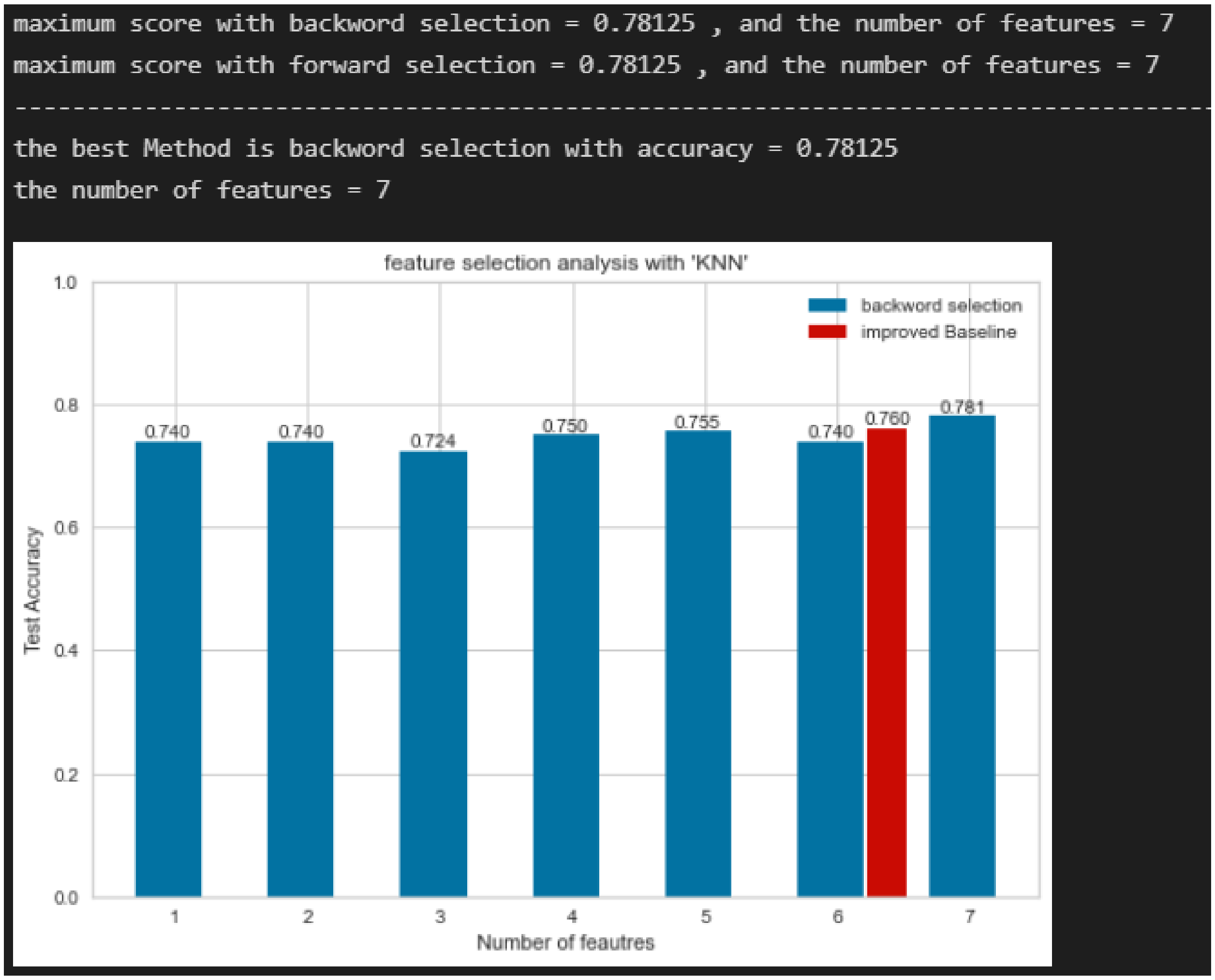
- Secondly with LR:

```
maximum score with anova = 0.7864583333333334 , and the number of features = 3
maximum score with information gain = 0.7864583333333334 , and the number of features = 3
-----------------------------------------------------------------------------------------
the best Method is Information Gain with accuracy = 0.7864583333333334
the number of features = 3
```



(b) Wrapper Methods (Forward Feature Elimination, Recursive Feature Elimination).

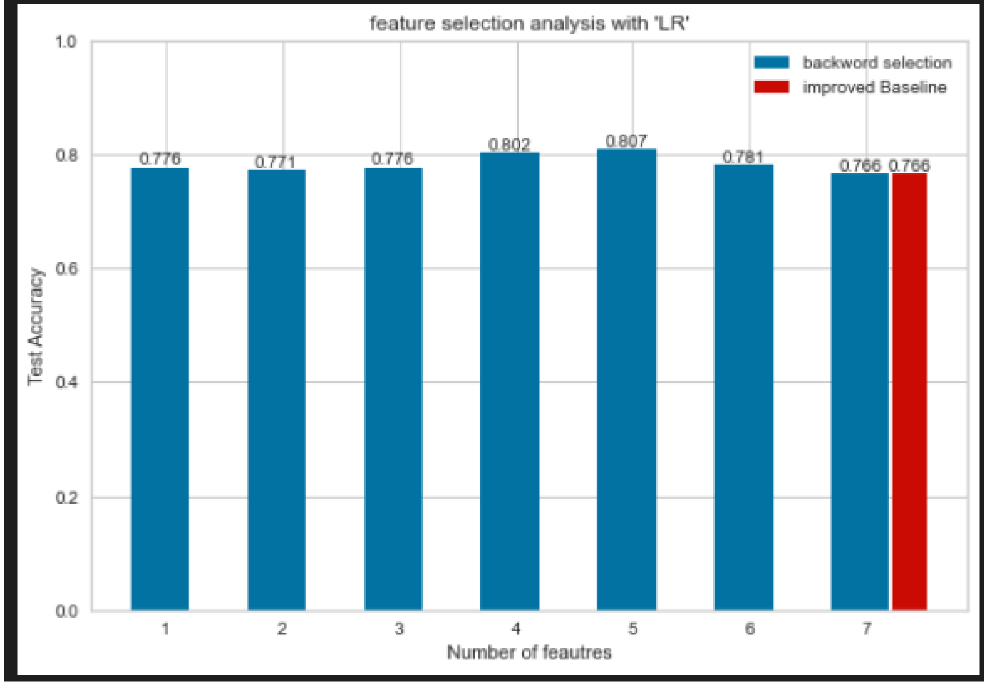
- First with KNN:



```
maximum score with backword selection = 0.78125 , and the number of features = 7
maximum score with forward selection = 0.78125 , and the number of features = 7
--------------------------------------------------------------------------------
the best Method is backword selection with accuracy = 0.78125
the number of features = 7
```

- Secondly with LR

```
maximum score with backword selection = 0.8072916666666666 , and the number of features = 5
maximum score with forward selection = 0.8072916666666666 , and the number of features = 5
-------------------------------------------------------------------------------------
the best Method is backword selection with accuracy = 0.8072916666666666
the number of features = 5
```

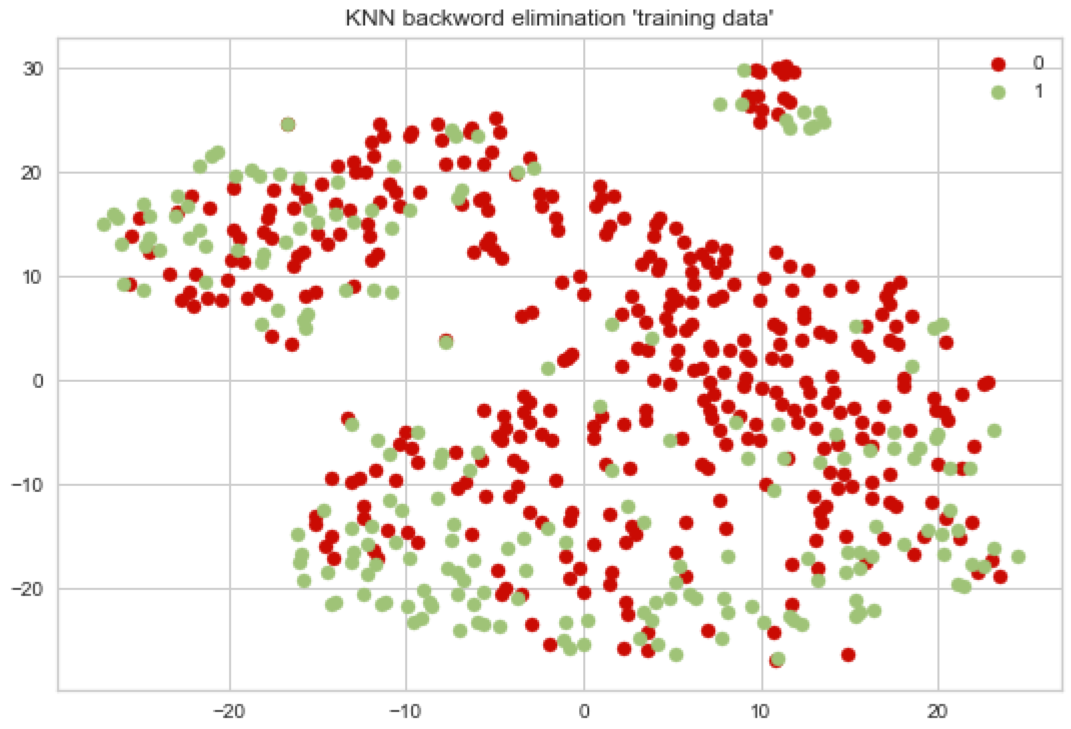


(c) Provide 2D TSNE plots, one for the training set and one for the test set, using

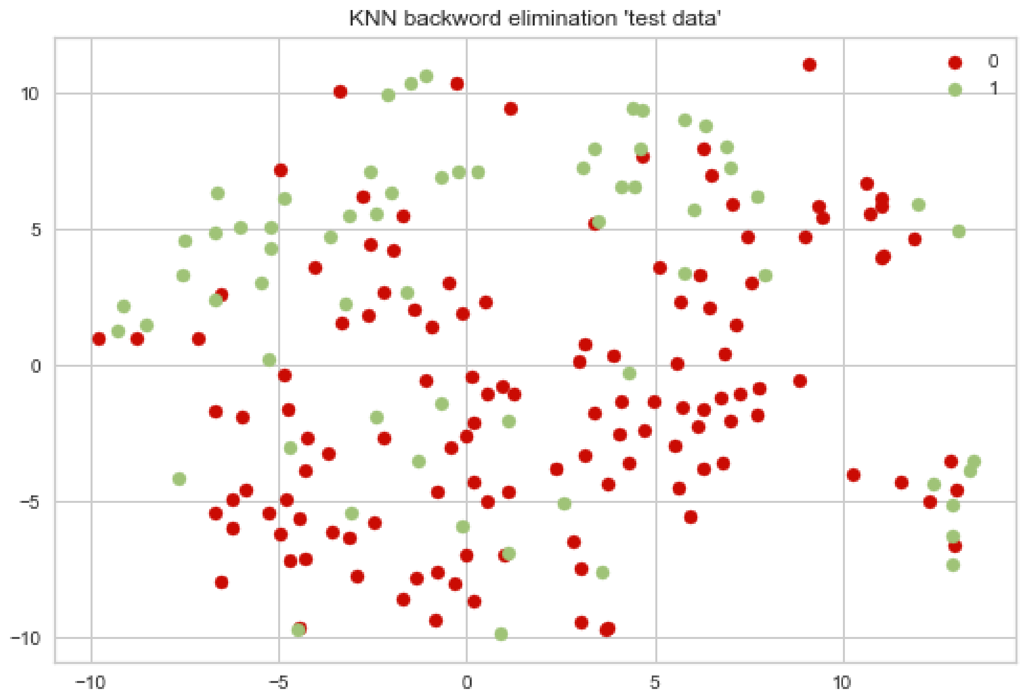
KNN

the best Method is backword selection with accuracy = 0.78125 the number of features = 7

```
[ ]  x_train_new, x_test_new, y_train_new, y_test_new = KNN_back_transformed_data
     # print("training data")
     draw_Tsne(x_train_new,y_train_new,title= "KNN backword elimination 'training data'")
```



```
[ ]  draw_Tsne(x_test_new,y_test_new,title= "KNN backword elimination 'test data'")
```



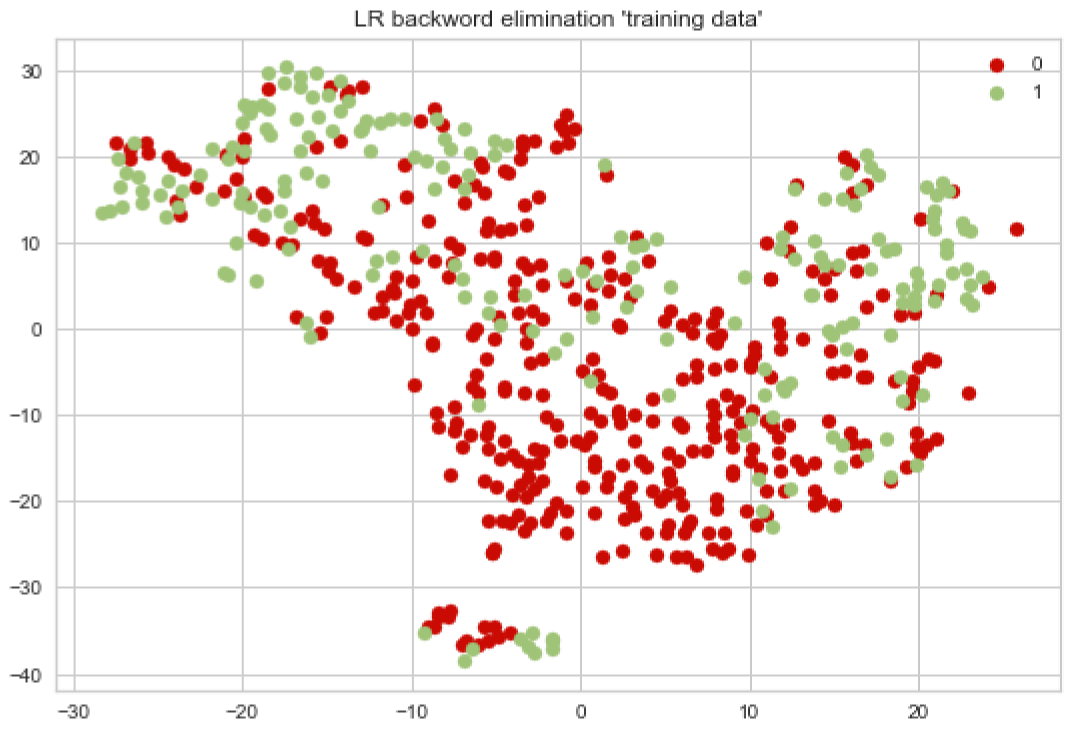
-The best for LR is the Wrapper Method with Backward Feature Elimination

```
the best Method is backword selection with accuracy = 0.8072916666666666 the number of features = 5
```

```
[ ]  x_train_new, x_test_new, y_train_new, y_test_new = LR_back_transformed_data
     print("training data")
     draw_Tsne(x_train_new,y_train_new,title= "LR backword elimination 'training data'")
```

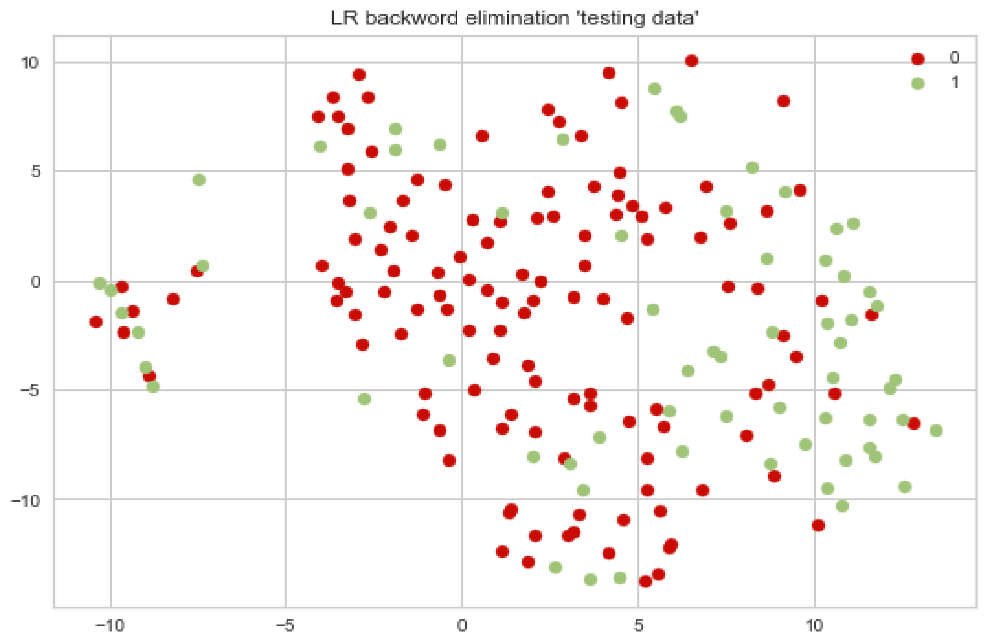
```
training data
```

```
training data
```



```
[ ]  print("test data")
     draw_Tsne(x_test_new,y_test_new,title= "LR backword elimination 'testing data'")
```

```
test data
```

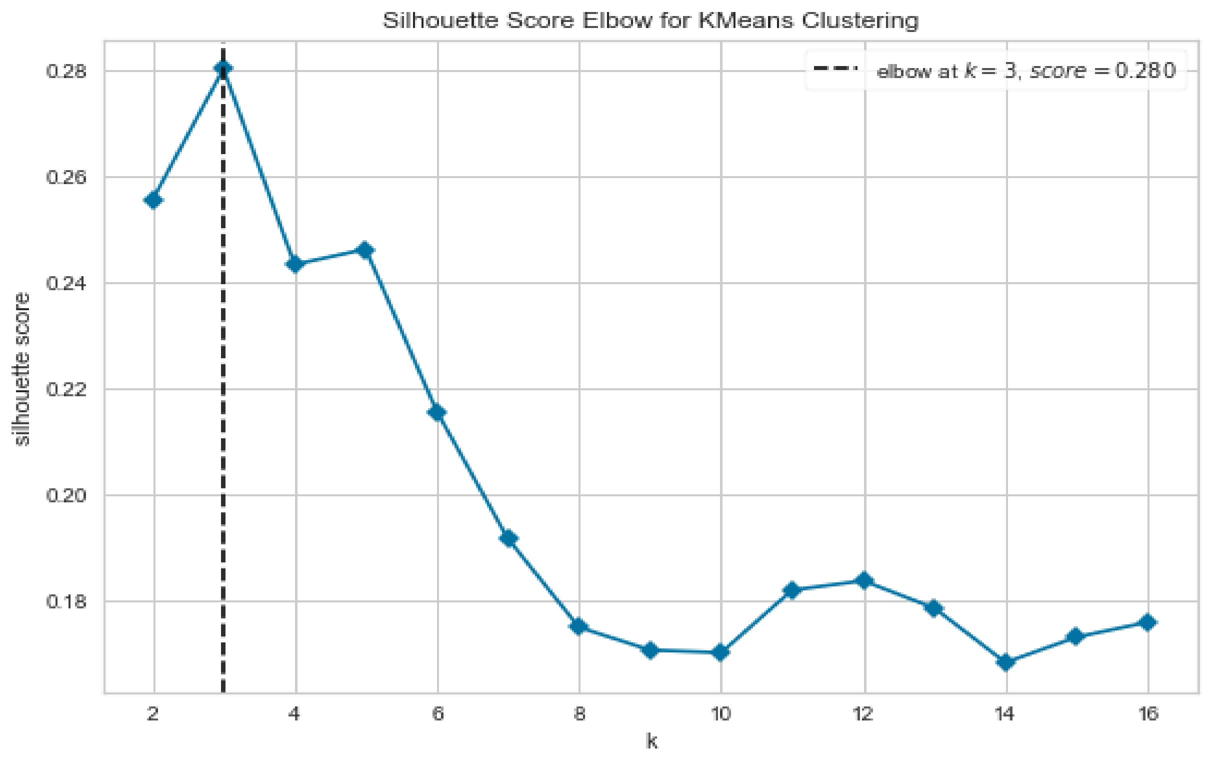


5) Choose the best number of cluster for k-means clustering algorithm on the processed

(a) Plot the silhouette score vs the number of clusters

FROM Q4 the best number of dimentions are 5 with LR

```
[ ]  from yellowbrick.cluster.elbow import kelbow_visualizer
     x_train_new, x_test_new, y_train_new, y_test_new = LR_back_transformed_data
     X_train = np.concatenate((x_train_new,x_test_new),axis=0)
     y_train = np.concatenate((y_train_new, y_test_new),axis=0)
     model = kelbow_visualizer(KMeans(random_state=0),X_train, k=(2,17),metric='silhouette',timings=False)

     pass
```



We noticed that with 5 features the best K = 3 that gives the highest silhouette score

6. Choose the best number of neurons for SOM algorithm, using the best features from

- LR -> the best Method is backword selection with accuracy = 0.8072916666666666 the number of features = 5

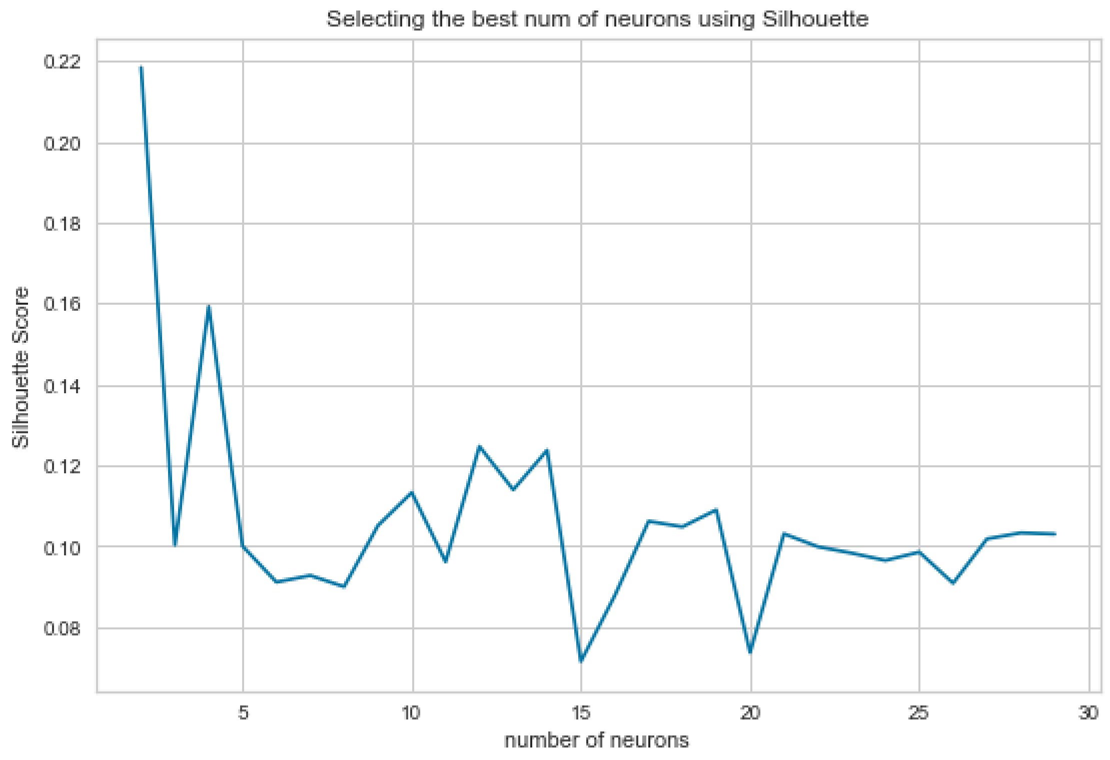
```
[ ]  x_train_new, x_test_new, y_train_new, y_test_new = LR_back_transformed_data
     X_train = np.concatenate((x_train_new,x_test_new),axis=0)
     y_train = np.concatenate((y_train_new, y_test_new),axis=0)
```

```
[ ]  !pip install minisom
     !pip install sklearn-som
```

```
[ ]  from minisom import MiniSom
     from sklearn_som.som import SOM
```

a)Plot the silhouette score vs the number of neurons (max 30 neurons)

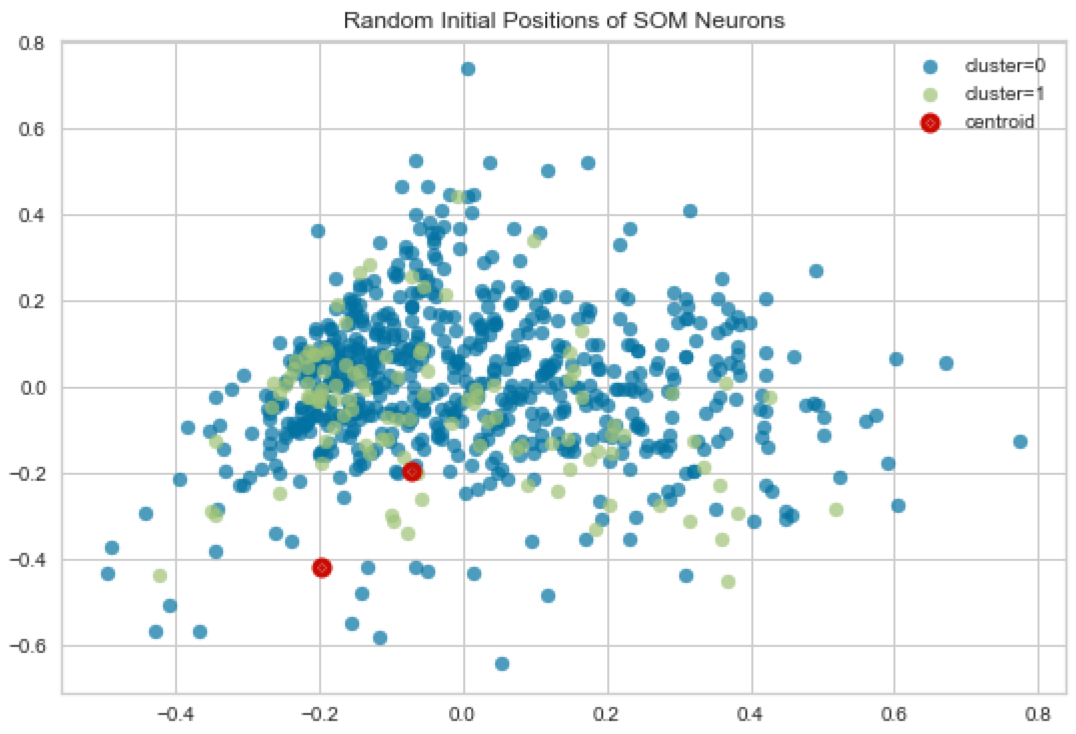
```
the best Silhouette Score= 0.21836682545149846 ,num of nearons = 2
<function matplotlib.pyplot.show(close=None, block=None)>
```

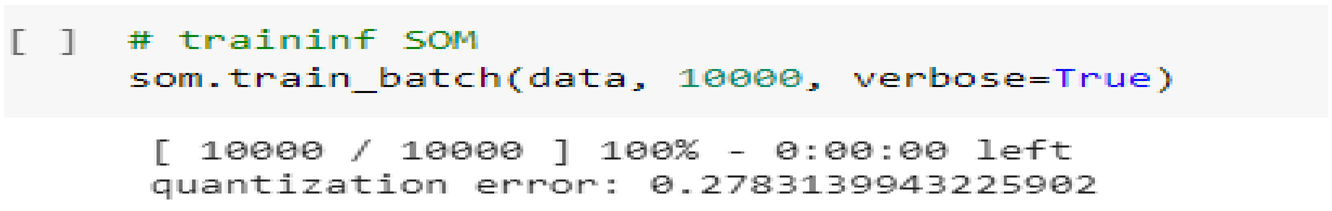


(c) Plot the initial and final Neuron positions

We used PCA to help in the plotting only.

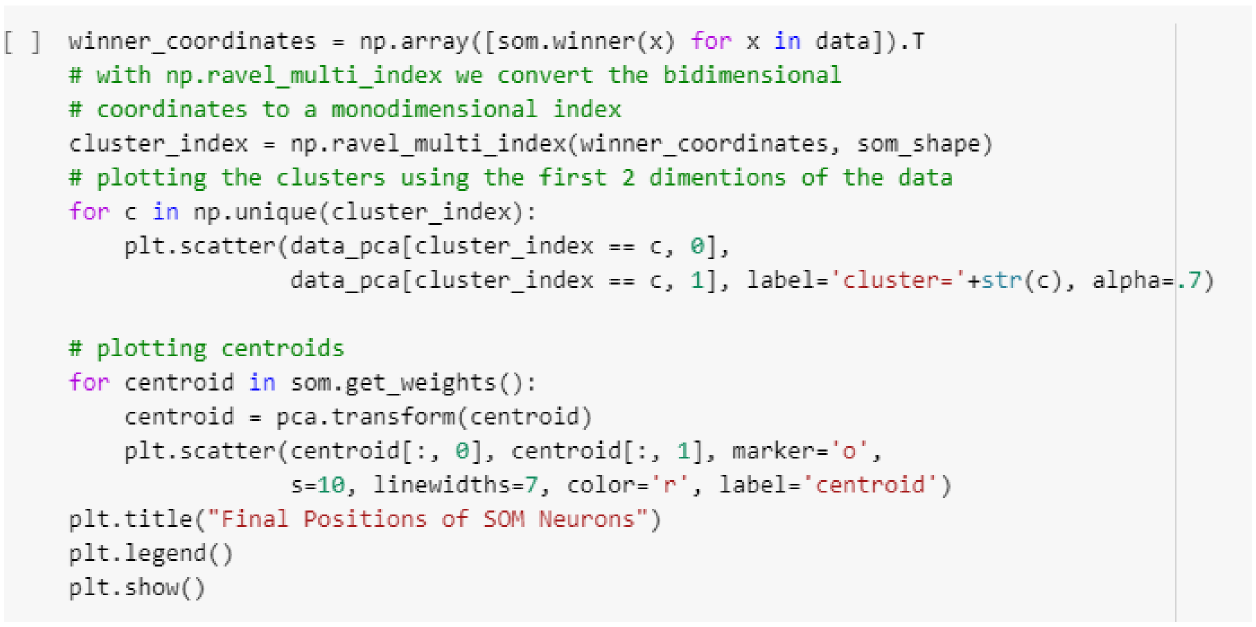
```
[ ] som_shape = (1, best_n)
    data = X_train
    som = MiniSom(som_shape[0], som_shape[1], data.shape[1], sigma=.5, learning_rate=.5,
                  neighborhood_function='gaussian', random_seed=0)
    winner_coordinates = np.array([som.winner(x) for x in data]).T
    # with np.ravel_multi_index we convert the bidimensional
    # coordinates to a monodimensional index
    cluster_index = np.ravel_multi_index(winner_coordinates, som_shape)

    # just for plotting
    pca = PCA(n_components=2)
    data_pca = pca.fit_transform(data)
    # plotting the clusters using the first 2 dimentions of the data
    for c in np.unique(cluster_index):
        plt.scatter(data_pca[cluster_index == c, 0],
                    data_pca[cluster_index == c, 1], label='cluster='+str(c), alpha=.7)

    # plotting centroids
    for centroid in som.get_weights():
        centroid = pca.transform(centroid)
        plt.scatter(centroid[:, 0], centroid[:, 1], marker='o',
                    s=10, linewidths=7, color='r', label='centroid')
    plt.title("Random Initial Positions of SOM Neurons")
    plt.legend()
    plt.show()
```



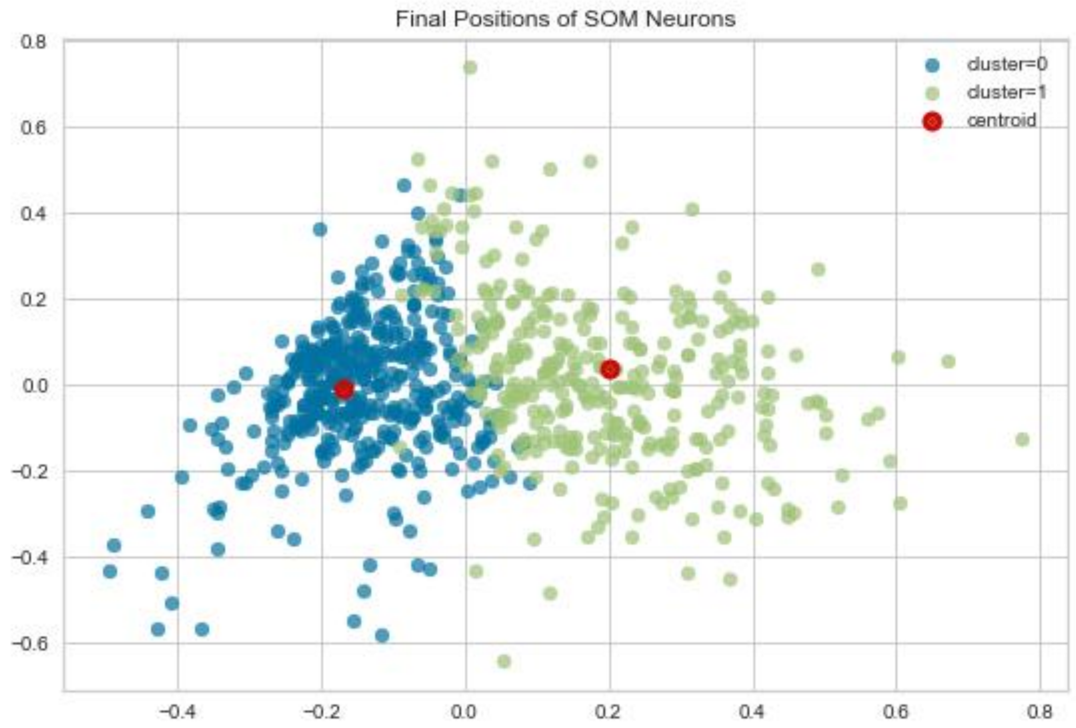


```python
# traininf SOM
som.train_batch(data, 10000, verbose=True)
```

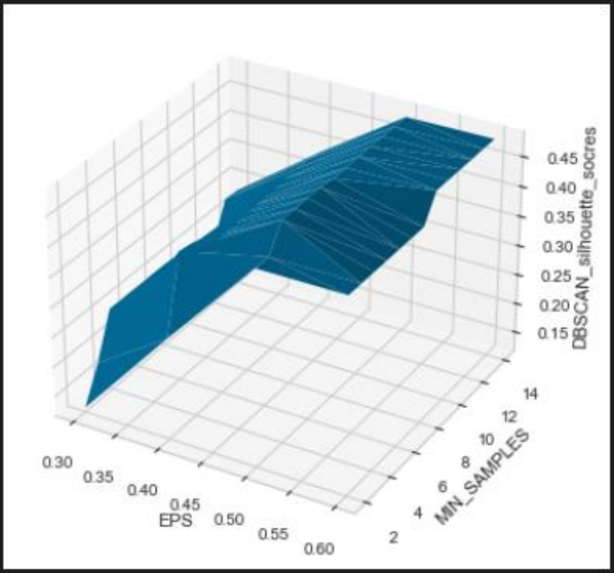
```
 [ 10000 / 10000 ] 100% - 0:00:00 left 
 quantization error: 0.2783139943225902
```



```python
winner_coordinates = np.array([som.winner(x) for x in data]).T
# with np.ravel_multi_index we convert the bidimensional
# coordinates to a monodimensional index
cluster_index = np.ravel_multi_index(winner_coordinates, som_shape)
# plotting the clusters using the first 2 dimentions of the data
for c in np.unique(cluster_index):
    plt.scatter(data_pca[cluster_index == c, 0],
                data_pca[cluster_index == c, 1], label='cluster='+str(c), alpha=.7)

# plotting centroids
for centroid in som.get_weights():
    centroid = pca.transform(centroid)
    plt.scatter(centroid[:, 0], centroid[:, 1], marker='o',
                s=10, linewidths=7, color='r', label='centroid')
plt.title("Final Positions of SOM Neurons")
plt.legend()
plt.show()
```

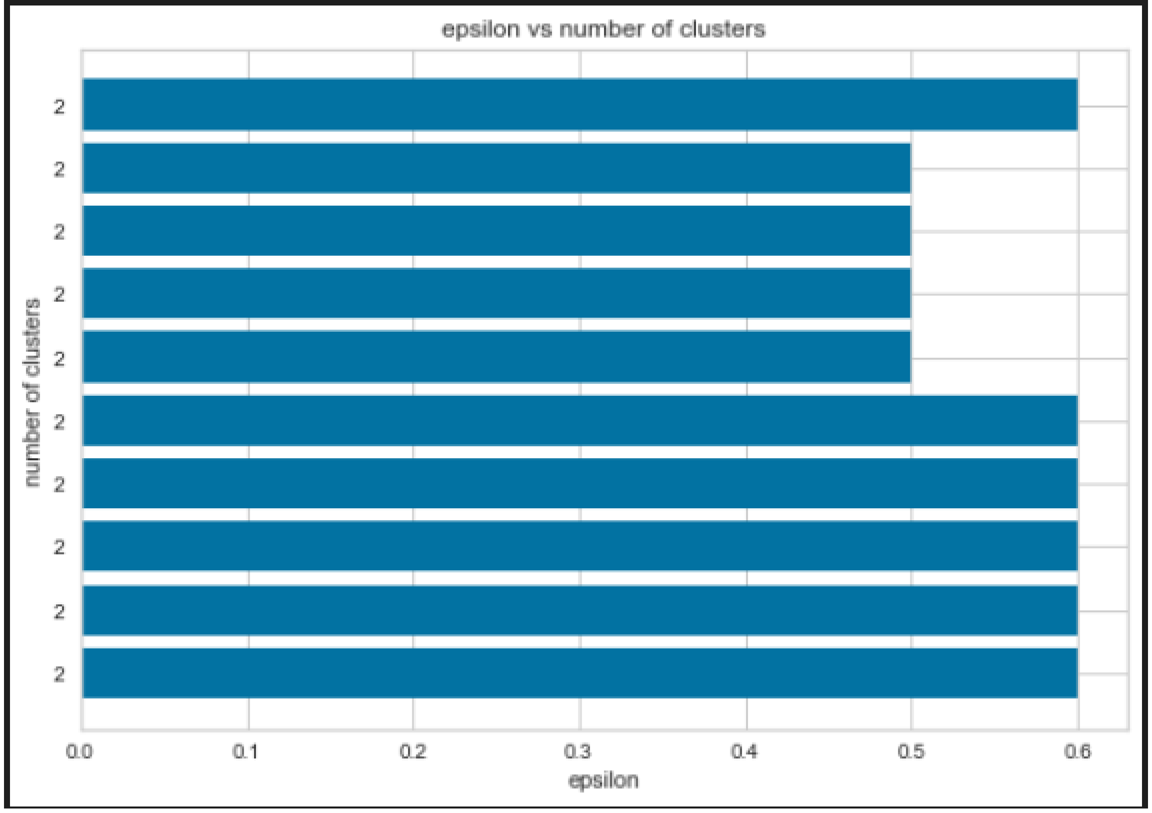

Final Positions of SOM Neurons
cluster=0
cluster=1
centroid
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8

7. Tune the epsilon (0.3-0.7) and minpoints (2-15) values to obtain the same number of clusters in Q6 by using DBSCAN.

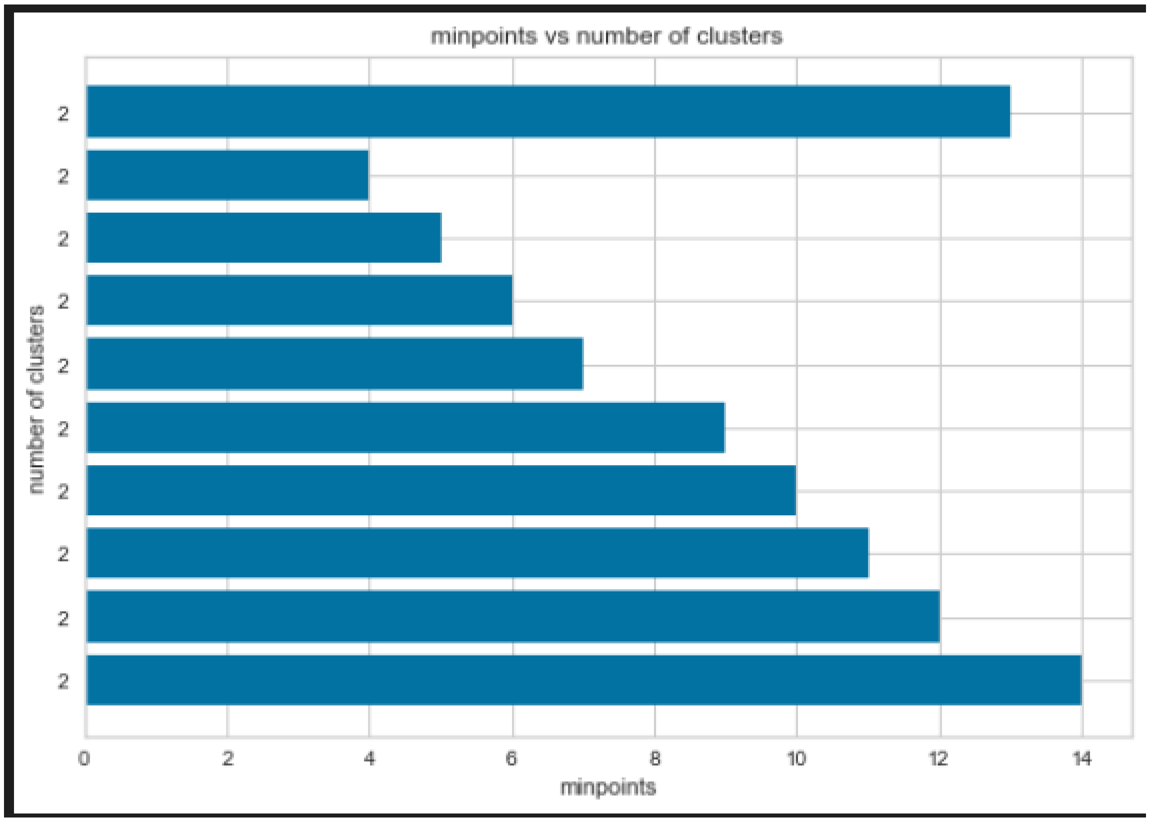


```
the best n_clustrs = 2 with sihouette score = 0.48500793075175636 and eps = 0.5, min_samples =4
```

a) First plot is epsilon vs number of clusters.



b) Second plot is minpoints vs number of clusters.



8) Conclusion:

a) The results of K-means from Q2 and Q5:

Before applying DR methods and FS, the best number of clusters is 2 with a silhouette score = 0.261, but after applying DR methods and FS the silhouette score slightly increased to be 0.280 with the number of clusters = 3.
As we know the labels of the data are 2 (0,1), which implies that before applying DR methods and FS is more realistic.

b) The results of TSNE plots from Q1, Q3, and Q4.

The lower the diminutions with the best accuracies lead to a better separation of the classes using Tsne, and the better separation means that we did well in the feature engineering phase.