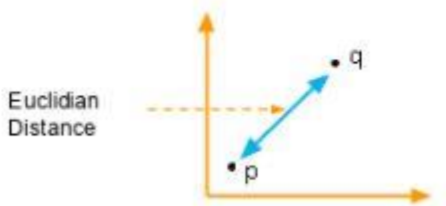
Assignment 3

Mostafa Mahmoud Abdelwahab Nofal
Nada Abdellatef Shaker Seddik
Hadeer Mamoduh Abdelfattah Mohammed
GRUOP |

Part 1: Calculations 1. 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.

$$d=\sqrt{\sum_{i=1}^{n}(q_i-p_i)^2}$$



C1=A2

C2=A4

1.Calculating distance between A1 and C1:

$d\ (A1, C1) =\sqrt{(5-2)^2 +(8-5)^2} =3\sqrt{2}$

2.Calculating distance between A1 and C2:

$d\ (A1, C2) =\sqrt{(1-2)^2 +(2-5)^2} =\sqrt{10}$

3.Calculating distance between A2 and C1:

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

4.Calculating distance between A2 and C2:

$d\ (A2, C2) =\sqrt{(5-2)^2 +(1-8)^2} =\sqrt{58}$

5.Calculating distance between A3 and C1:

$d\ (A3, C1) =\sqrt{(5-7)^2 +(8-5)^2} =\sqrt{13}$

6.Calculating distance between A3 and C2:

$d\ (A3, C2) =\sqrt{(1-7)^2 +(2-5)^2} =3\sqrt{5}$

7.Calculating distance between A4 and C1:

$d\ (A4, C1) = \sqrt{(5-1)^2 + (8-2)^2} = 2\sqrt{13}$

8.Calculating distance between A4 and C2:

$d\ (A4, C2) = \sqrt{(1-7=1)^2 + (2-2)^2} = 0$

9.Calculating distance between A5 and C1:

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

10.Calculating distance between A5 and C2:

$d\ (A5, C2) = \sqrt{(1-4)^2 + (2-9)^2} = \sqrt{58}$

Conclusion:

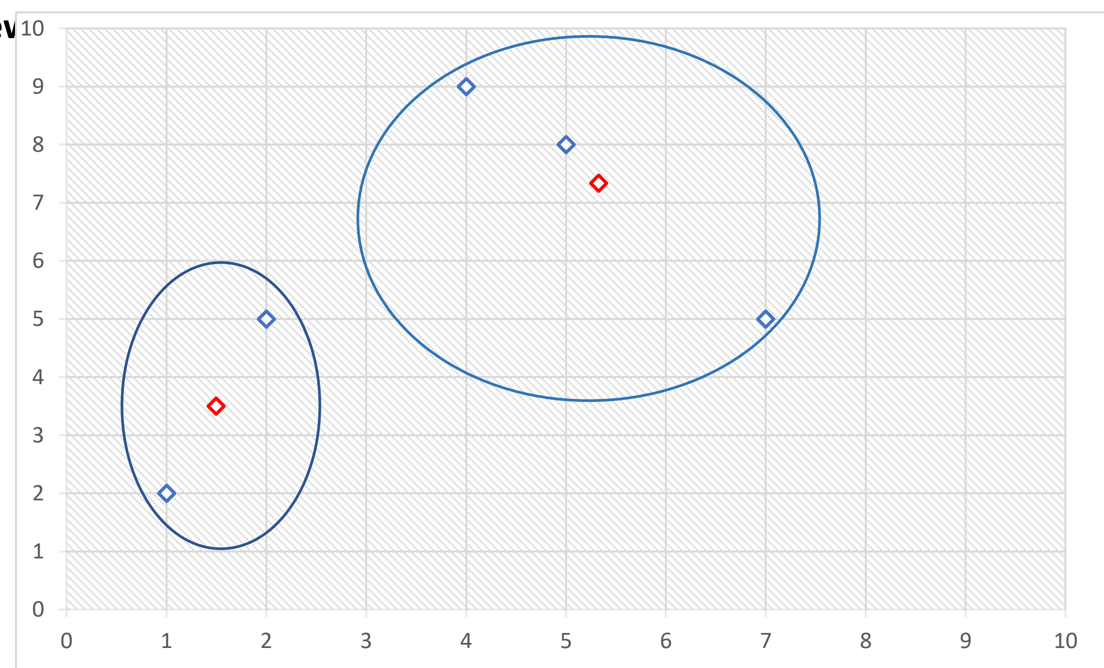
Given Points	Distance from center of cluster (1)	Distance from center of cluster (2)	Points belongs to cluster
A1(2,5)	$3\sqrt{2}$	$\sqrt{10}$	2
A2(5,8)	0	$\sqrt{58}$	1
A3(7,5)	$\sqrt{13}$	$3\sqrt{5}$	1
A4(1,2)	$2\sqrt{13}$	0	2
A5(4,9)	$\sqrt{2}$	$\sqrt{58}$	1

Recalculate the mean of each cluster:

-New center of cluster (1) = ((5+7+4)/3), (8+5+9)/3) = (5.33,7.33)

-New center 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 nev



c)Calculate the silhouette score and WSS score:

<u>1.WSS</u>

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

Where x_i = data point and c_i = closest point to centroid

WSS = (2-1.5) ^{2}+(5-3.5) ^{2}+ (5-5.33) ^{2}+(8-7.33) +(7-5.33) ^{2}+(5-7.33) +(1-1.5) ^{2} +(2-3.5) ^{2} +(4-5.33) ^{2} +(9-7.33) ^{2}

WSS=18.3334

<u>2. silhouette:</u>

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \text{ if } |C_i| > 1$$

and

$$s(i) = 0, \text{ if } |C_i| = 1$$

Cluster 1:

A2=(5,8), A3=(7,5), A5=(4,9)

Cluster 2:

A1= (2,5), A4=(1,2)

calculation of A3:

d (A3, A2) =$\sqrt{(5-7)^2+(8-5)^2}=\sqrt{13}$

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

-a(i) for A3 = ($\sqrt{13}$ +5)/2 = 4.3

d (A3, A4) =$\sqrt{(1-7)^2+(2-5)^2}=3\sqrt{5}$

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

-b(i) for A3 = (3$\sqrt{5}$ +5)/2=5.854

S (i) for A3 = (5.854- 4.3)/5.854 =-777/2927 =0.265

calculation of A2:

d (A2, A3) =$\sqrt{(5-7)^2+(8-5)^2}=\sqrt{13}$

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

-a(i) for A2 = ($\sqrt{13}$ +$\sqrt{2}$)/2 = 2.5

d (A2, A4) =$\sqrt{(5-1)^2+(8-2)^2}=2\sqrt{13}$

d (A2, A1) =$\sqrt{(5-2)^2+(8-5)^2}=3\sqrt{2}$

-b(i) for A2 = (3$\sqrt{2}$ +2$\sqrt{13}$)/2=5.726

S (i) for A3 = (5.726-2.5)/ 5.726= 0.563

calculation of A5:

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

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

-a(i) for A5 = (5 +$\sqrt{2}$)/2 = 3.2

d (A5, A4) =$\sqrt{(1-4)^2+(2-9)^2}=\sqrt{58}$

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

-b(i) for A5 = (2$\sqrt{5}$+$\sqrt{58}$)/2=6.043

S (i) for A5=(6.043-3.2)/ 6.043= 0.47

<u>calculation of A1:</u>

d (A1, A4) =$\sqrt{(1-2)^2+(2-5)^2}$ =$\sqrt{10}$=3.16

-a(i) for A1 =3.16

d (A2, A1) =$\sqrt{(5-2)^2+(8-5)^2}$ =$3\sqrt{2}$

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

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

-b(i) for A1 = ($2\sqrt{5}+3\sqrt{2}+5$**)/3= 4.571**

S(i) for A1=(4.571-√10)/ 4.571=0.308

<u>calculation of A4:</u>

d (A1, A4) =$\sqrt{(1-2)^2+(2-5)^2}$ =$\sqrt{10}$=3.16

-a(i) for A4=3.16

d (A3, A4) =$\sqrt{(1-7)^2+(2-5)^2}$ =$3\sqrt{5}$

d (A2, A4) =$\sqrt{(5-1)^2+(8-2)^2}$ =$2\sqrt{13}$

d (A5, A4) =$\sqrt{(1-4)^2+(2-9)^2}$ =$\sqrt{58}$

-b(i) for A4 = ($2\sqrt{13}+3\sqrt{5}+\sqrt{58}$**)/3= 7.178**

S(i) for A1= (7.178-√10)/ 7.178=0.5594

AVR_of_S(i) =(0.5594+0.308+0.47+0.563+0.265)/5

= 0.43308

Part 2: Programming

In this task, scikit-learn is used to implement Logistic Regression (LR) and K-Nearest Neighbor (K-NN) classifiers on the provided Diabetic dataset. The dataset has been standardized and split into training and testing. Through this assignment, the first 576 rows (75%) are used for training and the remaining 192 rows (25%) are used for testing. There are 2 classes in this dataset, and each sample in the provided dataset has 8 features.

Read the dataset

```
[4]  data = pd.read_csv("/content/Assignment3_dataset (1).csv")
```

We read the dataset and split it into training and testing datasets.

1.Train test split

```
[5]  X=data.iloc[:,:-1]
     Y=data.iloc[:,-1]
```

```
[7]  train_data = data.iloc[:576,:]
     test_data = data.iloc[576:,:]
```

```
train_data
```

Split to features and label

```
[9]  x_train = train_data.iloc[:,:-1]
     x_test = test_data.iloc[:,:-1]

     y_train = train_data.iloc[:,-1:]
     y_test = test_data.iloc[:,-1:]
```

1. Apply LR and KNN Models and their accuracies:

a. Logistic Regression:

Apply models

Logistic Regression

```python
LR_model = LogisticRegression()
LR_model = LR_model.fit(x_train, y_train)
y_pred_LR = LR_model.predict(x_test)

y_pred_LR
```

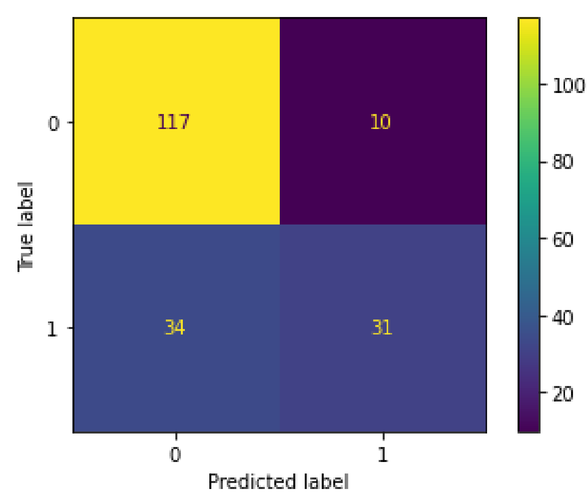
```
array([0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0])
```

Accuracy for LR:

Accuracies

```python
get_accuracies(y_test, y_pred_LR)
```

```
Classification Report:

              precision    recall  f1-score   support

           0       0.77      0.92      0.84       127
           1       0.76      0.48      0.58        65

    accuracy                           0.77       192
   macro avg       0.77      0.70      0.71       192
weighted avg       0.77      0.77      0.75       192


Confusion Matrix:

[[117  10]
 [ 34  31]]

Accuracy Score:

0.7708333333333334
```

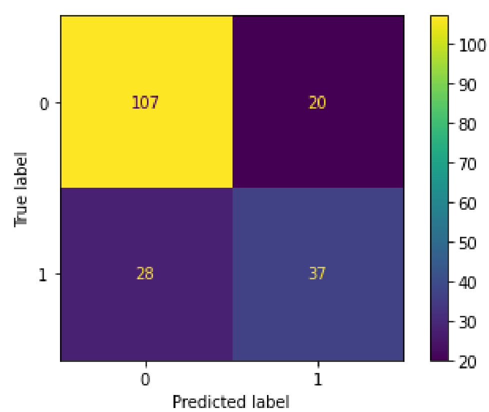


b. K-Nearest Neighbor

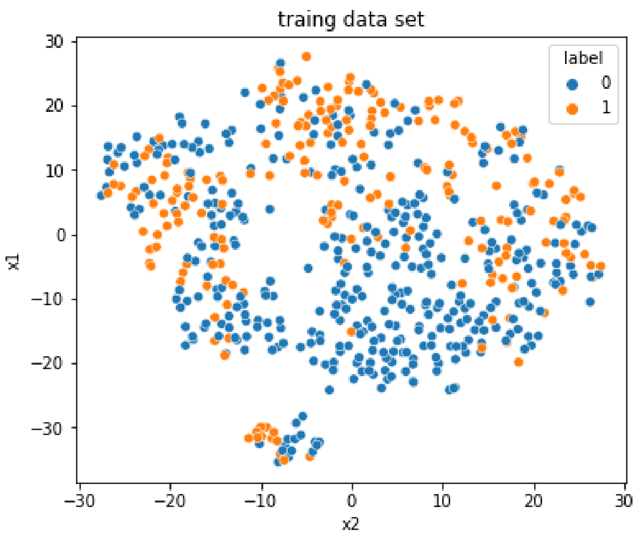
```
K-Nearest Neighbor

KNN_model = KNeighborsClassifier()
KNN_model = KNN_model.fit(x_train, y_train)
y_pred_KNN = KNN_model.predict(x_test)

y_pred_KNN

array([0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0,
       1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0,
       0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1,
       0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0])
```

Accuracy for KNN:

```
Accuracies

get_accuracies(y_test, y_pred_KNN)

Classification Report:

              precision    recall  f1-score   support

           0       0.79      0.84      0.82       127
           1       0.65      0.57      0.61        65

    accuracy                           0.75       192
   macro avg       0.72      0.71      0.71       192
weighted avg       0.74      0.75      0.75       192


Confusion Matrix:

[[107  20]
 [ 28  37]]

Accuracy Score:

0.75
```

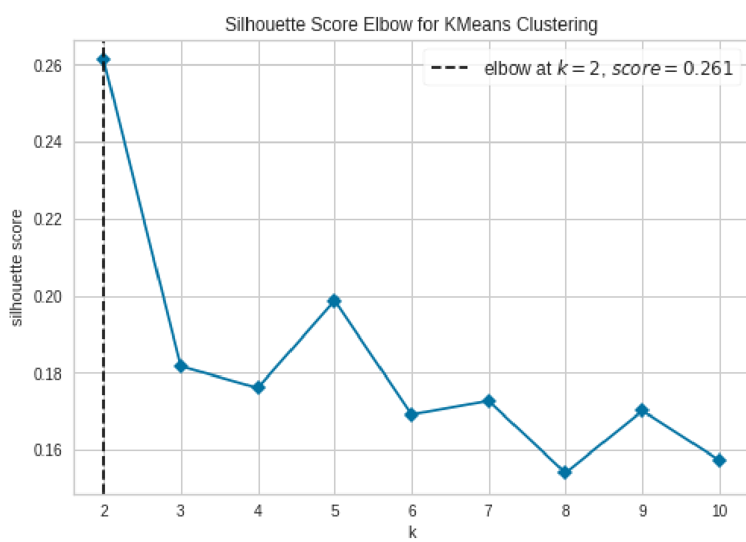


2. Plot TSNE diagram for training and testing dataset



2) The best number of clusters for k-means clustering algorithm:

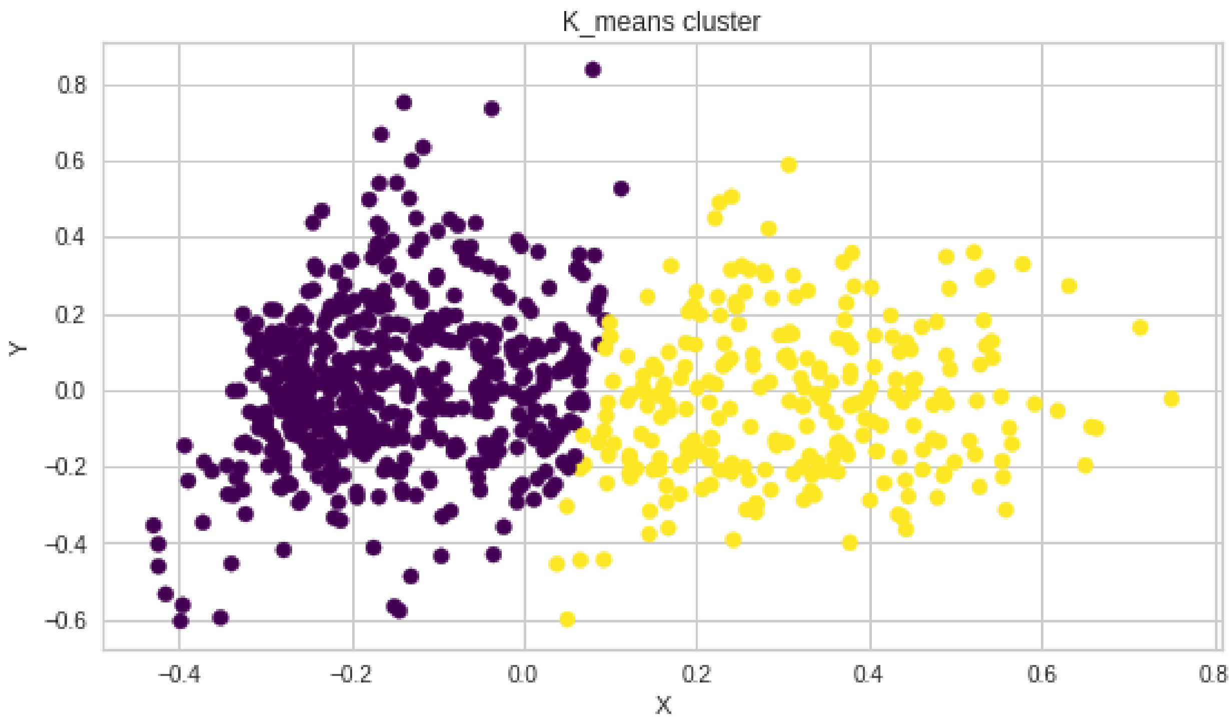
a) Plot the silhouette score vs the number of clusters.



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

- Best K value is 2 and its score is 0.26114611150604655

c) Plot the clustered data with optimum number of clusters which was 2.

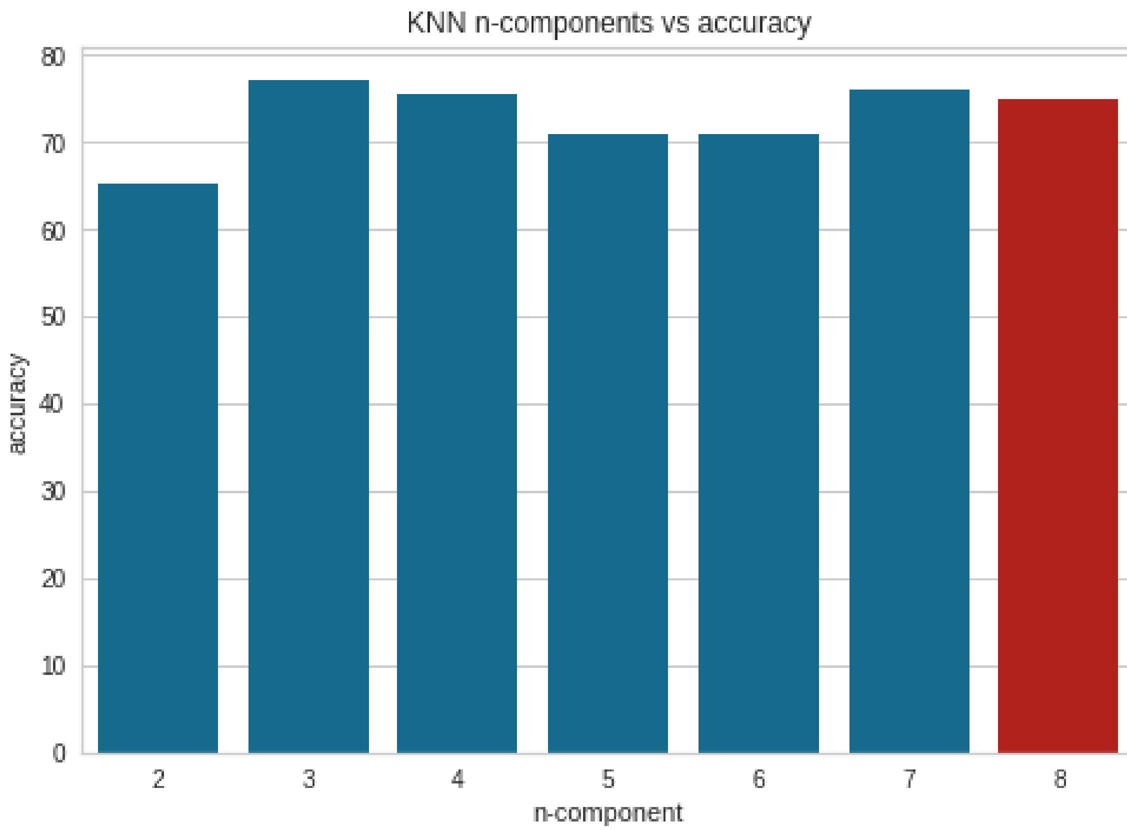


2) Apply the following Dimensionality Reduction (DR) methods:

a) KNN model using PCA

```
accuracy_dic_KNN={}
accuracy_list_KNN=[]
for i in range(2,8):
  pca_KNN=PCA_function(X,i)
  pca_x_train_KNN, pca_x_test_KNN, y_train_pca_knn, y_test_pca_knn = pca_KNN[0:576], pca_KNN[576:], Y[0:576], Y[576:]


  pca_KNN_model = KNeighborsClassifier()
  pca_KNN_model = KNN_model.fit(pca_x_train_KNN, y_train.values.ravel())
  pca_y_pred_KNN = KNN_model.predict(pca_x_test_KNN)

  accuracy_list_KNN.append(accuracy_score(y_test, pca_y_pred_KNN)*100)
best_n= accuracy_list_KNN.index(max(accuracy_list_KNN))+2
max_acc=max(accuracy_list_KNN)
print("Best value of n components for KNN: ",best_n," with maximmum accuracy ",max(accuracy_list_KNN))
accuracy_list_KNN.append(accuracy_score(y_test, y_pred_KNN)*100)
accuracy_dic_KNN={"n-component":[2,3,4,5,6,7,8],"accuracy":accuracy_list_KNN}
accuracy_df_KNN= Loading… me(accuracy_dic_KNN)
ax=sns.barplot(x="n-component", y='accuracy', data=accuracy_df_KNN,
             palette=["b" if x!=8 else 'r' for x in accuracy_df_KNN['n-component']]).set(title='KNN n-components vs accuracy')
```

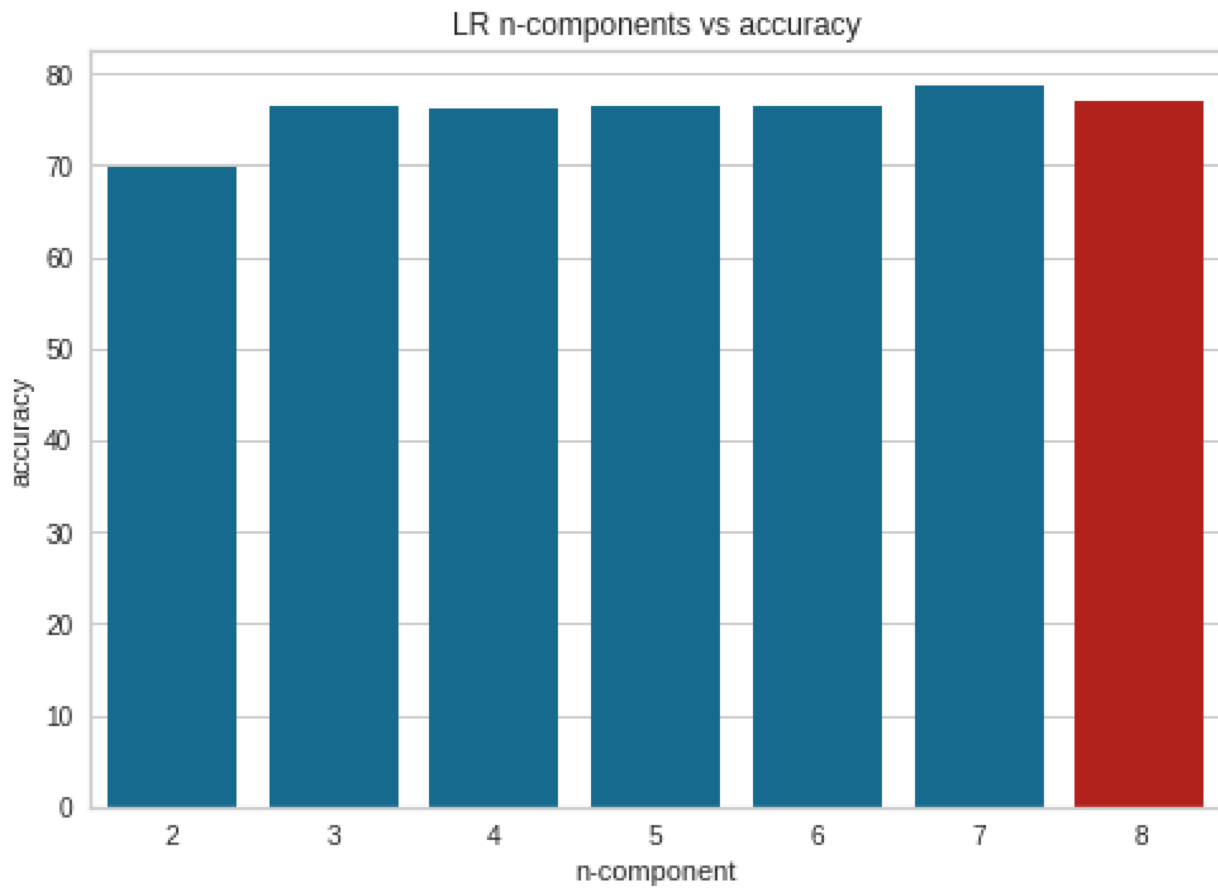
Best value of n components for KNN: 3 with maximum accuracy 77.08



b) Logistic Regression model using PCA (n components=n, random state=0)

```
accuracy_dic={}
accuracy_listLR=[]
for i in range(2,8):
  pca_LR=PCA_function(X,i)
  pca_x_train_LR, pca_x_test_LR, y_train_pca_LR, y_test_pca_LR = pca_LR[0:576], pca_LR[576:], Y[0:576], Y[576:]

  pca_LR_model = LogisticRegression()
  pca_LR_model = pca_LR_model.fit(pca_x_train_LR, y_train.values.ravel())
  pca_y_pred_LR = pca_LR_model.predict(pca_x_test_LR)

  accuracy_listLR.append(accuracy_score(y_test, pca_y_pred_LR)*100)
best_n2= accuracy_listLR.index(max(accuracy_listLR))+2
max_acc2=max(accuracy_listLR)
print("Best value of n components for LR: ",best_n2," with maximmum accuracy ",max(accuracy_listLR))

accuracy_listLR.append(accuracy_score(y_test, y_pred_LR)*100)
accuracy_dic={"n-component":[2,3,4,5,6,7,8],"accuracy":accuracy_listLR}
accuracy_df=pd.DataFrame(accuracy_dic)
ax=sns.barplot(x="n-component", y='accuracy', data=accuracy_df,
             palette=["b" if x!=8 else 'r' for x in accuracy_df['n-component']]).set(title='LR n-components vs accuracy')
```

Best value of n components for LR: 7 with maximum accuracy 78.64

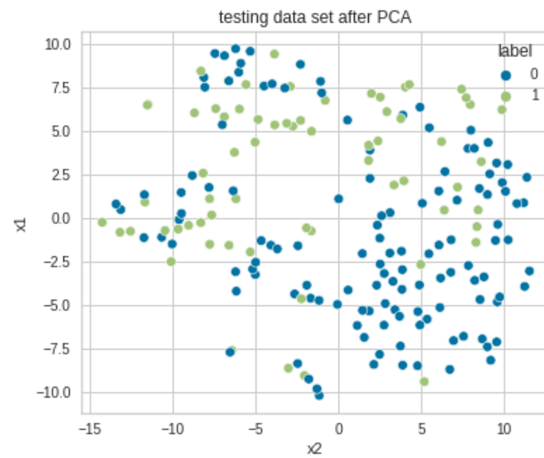


c) 2D TSNE plots, one for the training set and one for the test set using best n-components from PCA dimensionality reduction.

LR:

Training:

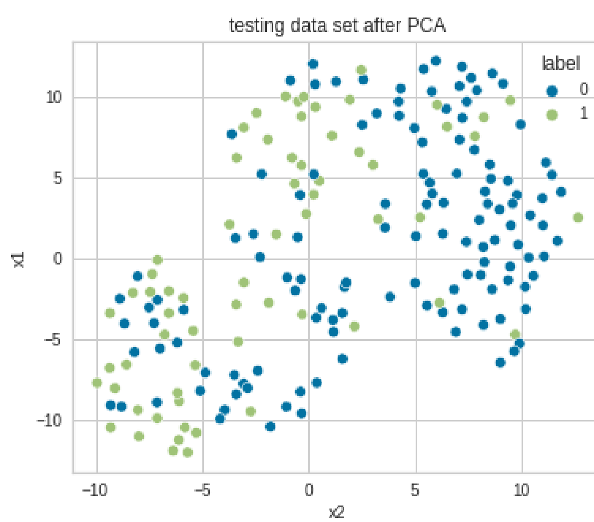
Testing:



KNN:

Training:

Testing:



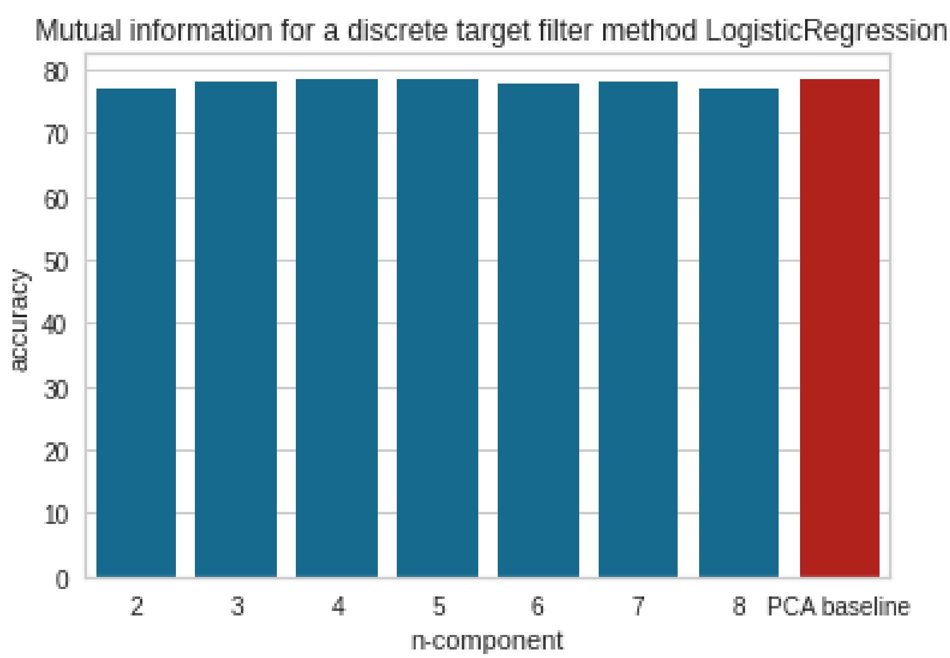
3) Feature Selection:

a) Filter Methods

```
def filter_selecton(x_train1, y_train1, x_test1, y_test1,  model_name,m):
  accuracy_dic={}
  accuracy_list=[]
  accuracy_list2=[]
  model = model_name
  for i in range(2,9):
    fsm =  SelectKBest(mutual_info_classif, k=i)
    acc = select_feature(x_train1, y_train1, x_test1, y_test1, fsm, model)
    accuracy_list.append(acc)


  print('max mutal',max(accuracy_list))
  best_n=accuracy_list.index(max(accuracy_list))+2
  print("Best value of n components: ",best_n, "from Mutual information for a discrete target filter method")
  if m=='LogisticRegression':
    accuracy_list.append(max_acc2)
  else:
    accuracy_list.append(max_acc)


  accuracy_dic={"n-component":[2,3,4,5,6,7,8,"PCA baseline"],"accuracy":accuracy_list}
  accuracy_df=pd.DataFrame(accuracy_dic)


  ax=sns.barplot(x="n-component", y='accuracy', data=accuracy_df,
             palette=["b" if x!='PCA baseline' else 'r' for x in accuracy_df['n-component']]).set(title=' Mutual information for a discrete tar
```

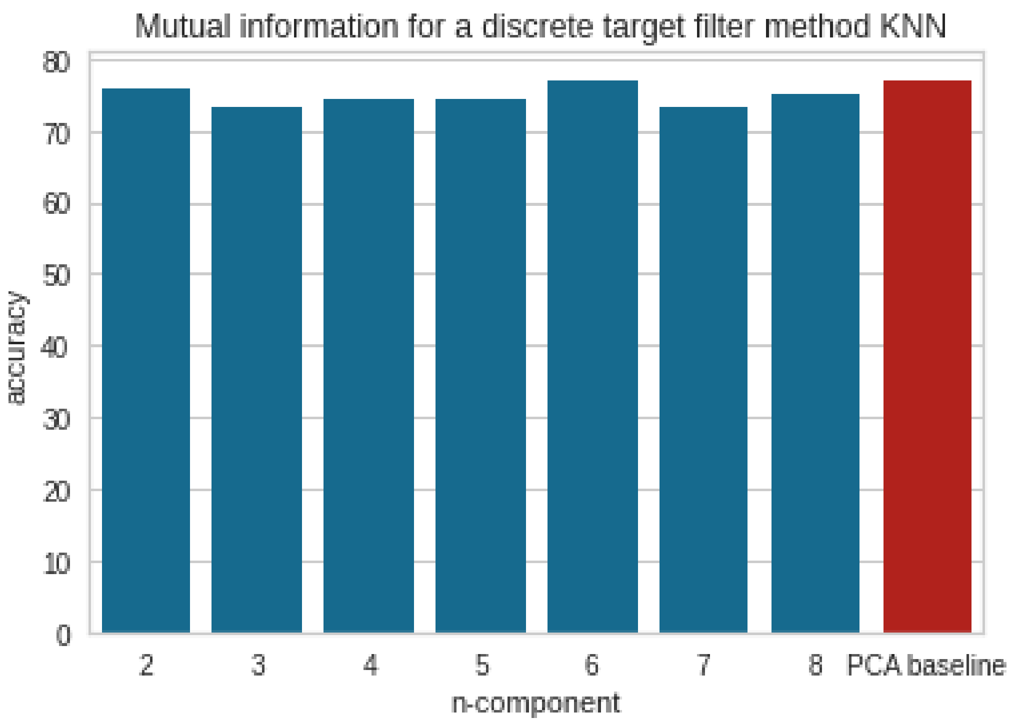

I. Filter Methods using Information Gain on LR model:

- Maximum Information Gain: 78.64
- Best value of n components: 4 from Mutual information for a discrete target filter method



II. Filter Methods using Information Gain on KNN model:

- Maximum Information Gain: 77.08
- Best value of n components: 6 from Mutual information for a discrete target filter method



b) Wrapper Methods:
Function to fit, transform and predict the data

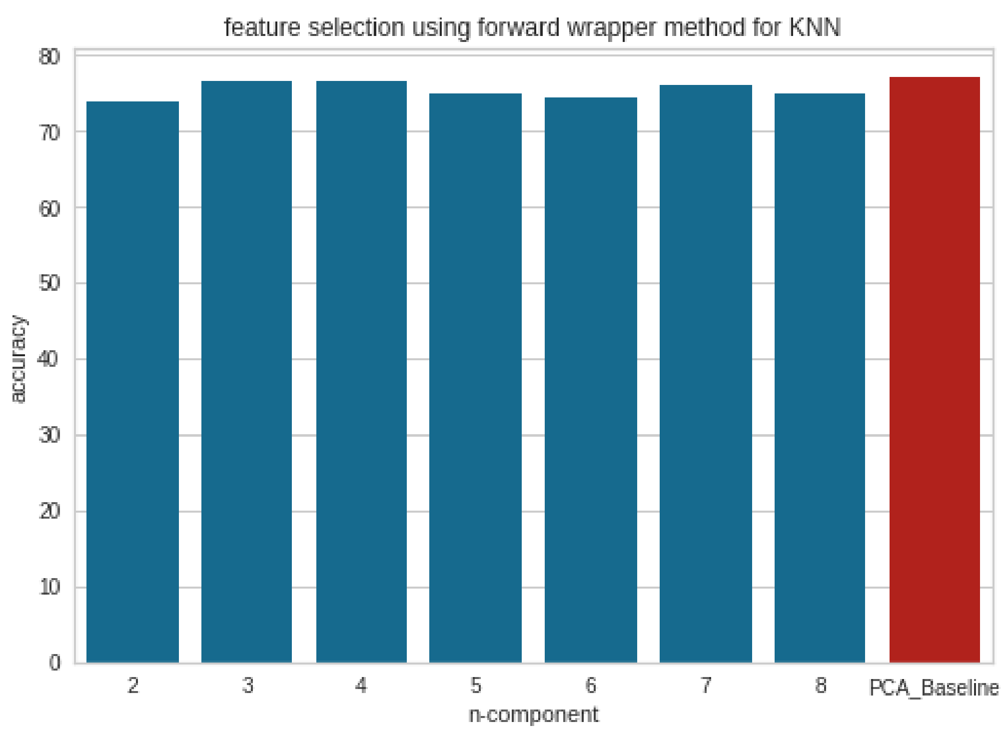
```
[164] def wrapper_select_feature(X_train, y_train, X_test, y_test, label, model,i):
        fs = SFS(model,
                 k_features=i,
                 forward=label,
                 verbose=2,
                 scoring='roc_auc',
                 cv=4)
        fs.fit(np.array(X_train), y_train.values.ravel())
        filtered_features= X_train.columns[list(fs.k_feature_idx_)]
        l=list(filtered_features)
        X_train_new = X_train.loc[:,l]
        X_test_new = X_test.loc[:,l]
        model.fit(X_train_new, y_train.values.ravel())
        y_pred = model.predict(X_test_new)
        acc = accuracy_score(y_test, y_pred) * 100

        return acc,X_train_new,X_test_new
```

Function to determine the best features based on maximum accuracy and plot it with the number of features

```python
def wrapper_selecton(x_train1, y_train1, x_test1, y_test1,  model_name,m):
  accuracy_dic={}
  accuracy_list=[]
  feat=[]
  feat_test=[]
  model = model_name
  for i in range(2,9):
    acc,df_X_train_new,df_X_test_new = wrapper_select_feature(x_train1, y_train1, x_test1, y_test1,True , model_name,i)
    accuracy_list.append(acc)
    feat.append(df_X_train_new)
    feat_test.append(df_X_test_new)
    index=accuracy_list.index(max(accuracy_list))
    feat_name=feat[index]
    test_name=feat_test[index]


  print(accuracy_list)
  print('max forward',max(accuracy_list))
  #print('list: ',feat_name)



  best_n=accuracy_list.index(max(accuracy_list))+2
  print("Best value of n components: ",best_n, "forward wrapper method")
  if m=='LogisticRegression':
    accuracy_list.append(max_acc2)
  else:
    accuracy_list.append(max_acc)

  #accuracy_list.append(accuracy_score(y_test, y_pred_LR))
  accuracy_dic={"n-component":[2,3,4,5,6,7,8,"PCA_Baseline"],"accuracy":accuracy_list}
  accuracy_df=pd.DataFrame(accuracy_dic)
  ax=sns.barplot(x="n-component", y='accuracy', data=accuracy_df,palette=["b" if x!='PCA_Baseline' else 'r' for x in accuracy_df['n-component']]).set(title='feature selection using forward wrap
  return feat_name,test_name,ax
```

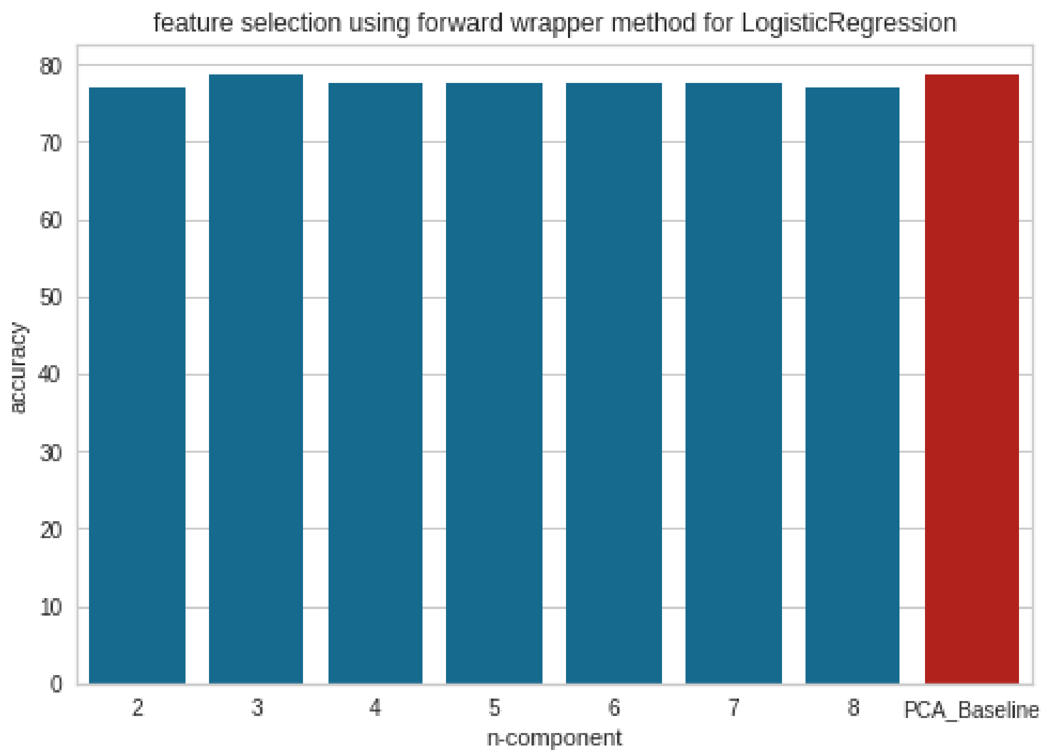
I. Wrapper Method using Forward on KNN model:

- Maximum forward accuracy: 76.5625
- Best value of n components: 3 using forward wrapper method with KNN



II. Wrapper Method using Forward on LR model:

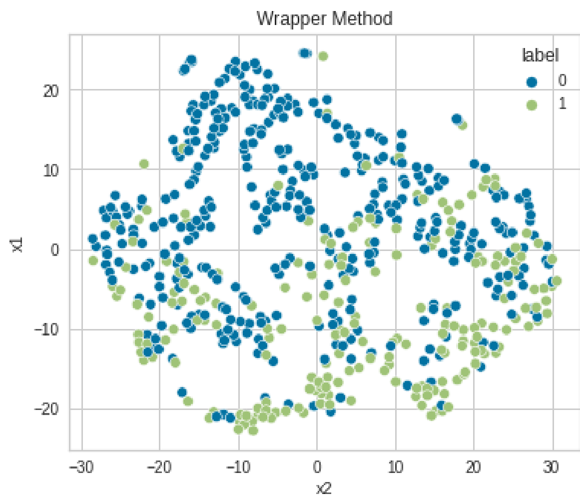
- Maximum forward accuracy: 78.64583333333334
- Best value of n components: 3 forward wrapper method with LR



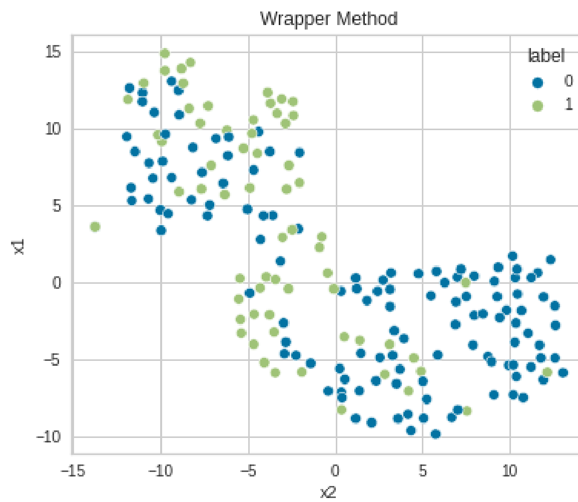
c) Plot TSNE for training and testing dataset after Feature Selection using forward rapper method.

i. LR

Training

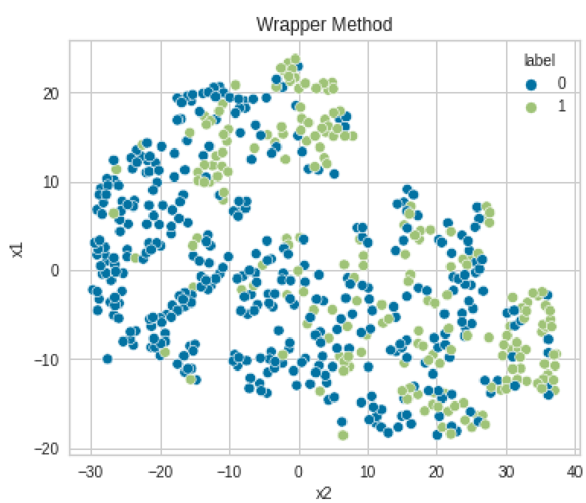


Testing

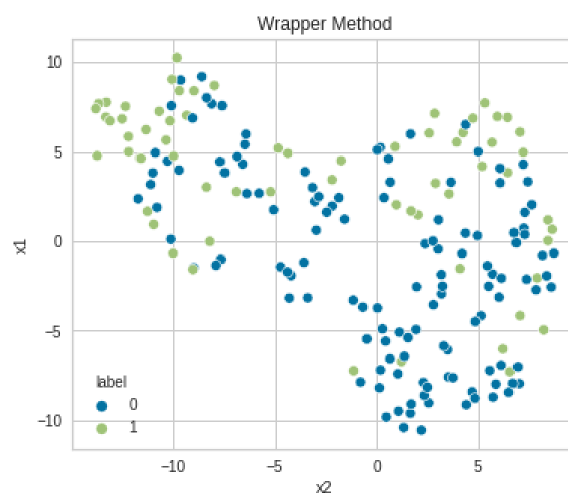


ii. KNN

Training

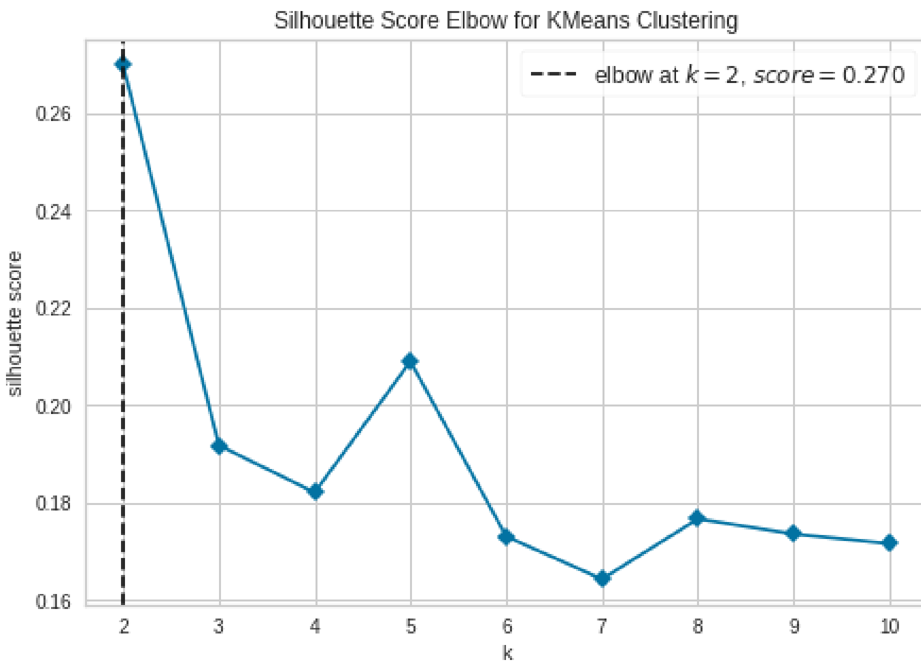


Testing



5) Choose best k from DR:

silhouette score vs the number of clusters



Best K value is 2 and its score is 0.27

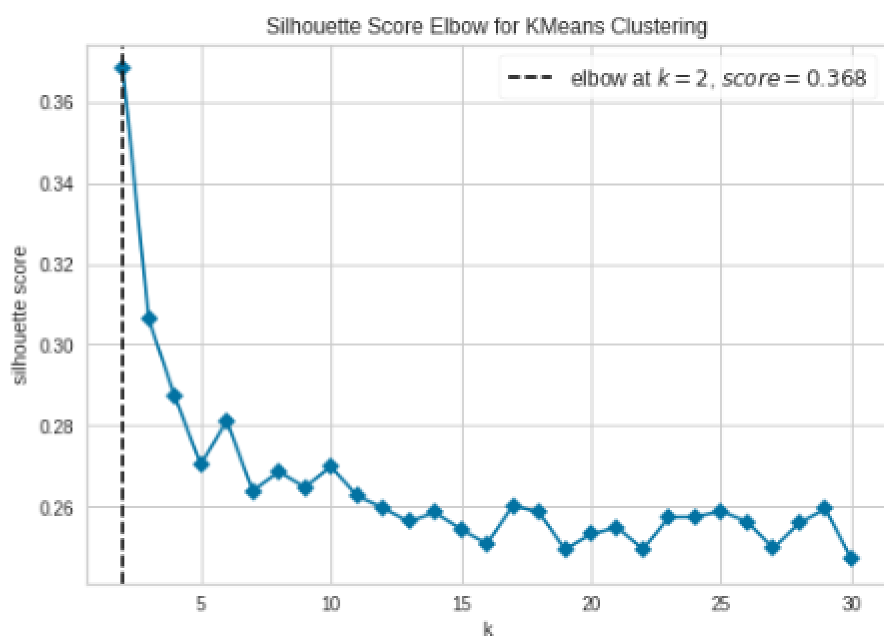
Q6) Self Organizing Map

We used the best features from question 3 which was 3 to train SOM.

Loop through the 30 neurons

```
[56]
from minisom import MiniSom
def minisom_sil(data, dim):
  s = []
  for i in range(2,31):
    som_shape = (i, 1)
    som = MiniSom(som_shape[0], som_shape[1],dim, random_seed=0)
    som.train_batch(data, 1000)
    # each neuron represents a cluster
    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)
    s.append(silhouette_score(data, cluster_index))
  best_k(data, 31)

```

Plotting the silhouette score for the 30 neurons



The optimal number of neurons for SOM is 2

We trained a new MiniSom model with our best number of neurons to calculate the initial and final weights for it before and after training.

```
[58]
from minisom import MiniSom
som_shape = (2, 1)
som = MiniSom(som_shape[0], som_shape[1],3, random_seed=0)
initial_weights = som.get_weights().copy()
som.train_batch(np.array(x_best_pca), 1000)
final_weights = som.get_weights().copy()

winner_coordinates = np.array([som.winner(x) for x in np.array(x_best_pca)]).T
cluster_index = np.ravel_multi_index(winner_coordinates, som_shape)
```

Initial Weights

```
[59]
initial_weights

array([[[ 0.20053839,  0.88405308,  0.42217829]],

       [[ 0.26298257, -0.44732694,  0.8548326 ]]])
```

Final Weights

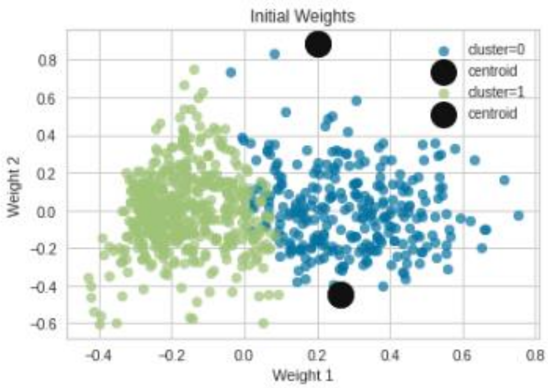
```
[60]
final_weights

array([[[ 0.30397215, -0.03699393, -0.03066724]],

       [[-0.15368857, -0.13045023, -0.09615449]]])
```

In our case, the shape of our weights is (2,1,8) so we should reshape it to be able to plot them, so we reshaped them to (2,3).

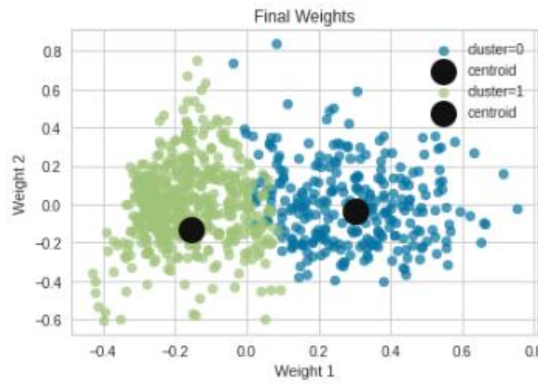
```
[61]
final_weights = final_weights.reshape(2,3)
initial_weights = initial_weights.reshape(2,3)
```

```
[62]
initial_weights

array([[ 0.20053839,  0.88405308,  0.42217829],
       [ 0.26298257, -0.44732694,  0.8548326 ]])
```

Initial Weights Plot



Final Weights Plot



Q7) SOM Algorithm

Tune the hyperparameters epsilon and minpoints to get the clusters that are equal to our number of clusters in question 6 which was 2 after filtering the noise.

```
from sklearn.cluster import DBSCAN
clusters_2 = []
epsilon_ = np.linspace(0.3, 0.7, 50).tolist()
minpoints_ = np.arange(2, 16).tolist()

for i in epsilon_:
  for j in minpoints_:

    model = DBSCAN(eps=i, min_samples=j).fit(X)
    Clusters = list(np.unique(model.labels_))
    DB_Predict = model.fit_predict(X)
    if -1 in Clusters:
      Clusters.remove(-1)       #filer the noise
    # print(len(num_clusters))
    # print(num_clusters)


    if len(Clusters) == 2:
      list1.append([len(Clusters),
                    i,
                    j,
                    silhouette_score(X ,DB_Predict, random_state=0)])

      clusters_2.extend(list1)

    list1 = []
```

We used the itemgetter library to sort our list based on the highest silhouette score.

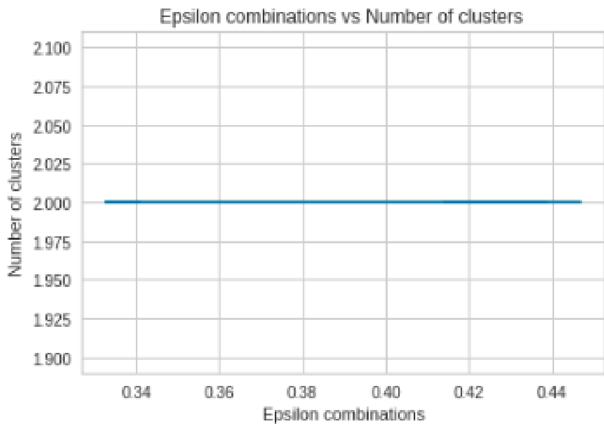
```
from operator import itemgetter

top_sil = sorted(clusters_2, key = itemgetter(3), reverse = True)
top_sil_df = pd.DataFrame(top_sil, columns = ["Cluster_num", "Epsilon", "Minpoints", "Silhouette"])
top_sil_df[:10]
```

Output of the top 10 sorted combinations of epsilon and minpoints.

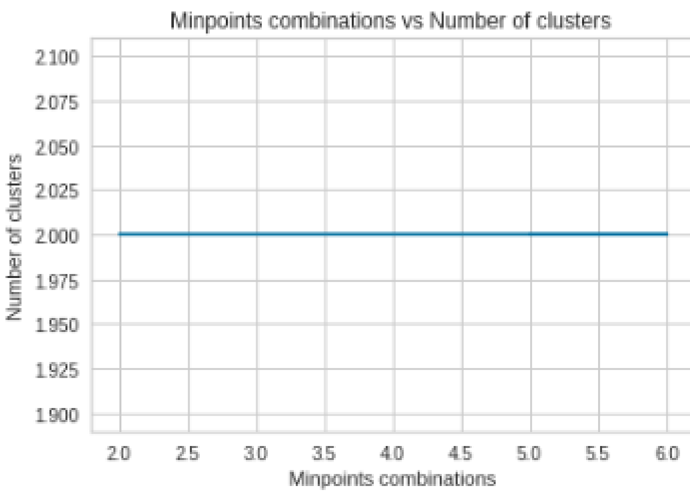
```
5 top_sil_df[:10]
```

	Cluster_num	Epsilon	Minpoints	Silhouette
0	2	0.446939	2	0.427027
1	2	0.430612	2	0.416000
2	2	0.422449	2	0.413268
3	2	0.414286	2	0.407059
4	2	0.438776	2	0.400058
5	2	0.332653	3	0.310725
6	2	0.332653	4	0.310725
7	2	0.340816	6	0.310229
8	2	0.332653	5	0.308900
9	2	0.332653	6	0.308900

Epsilon combinations vs Number of clusters

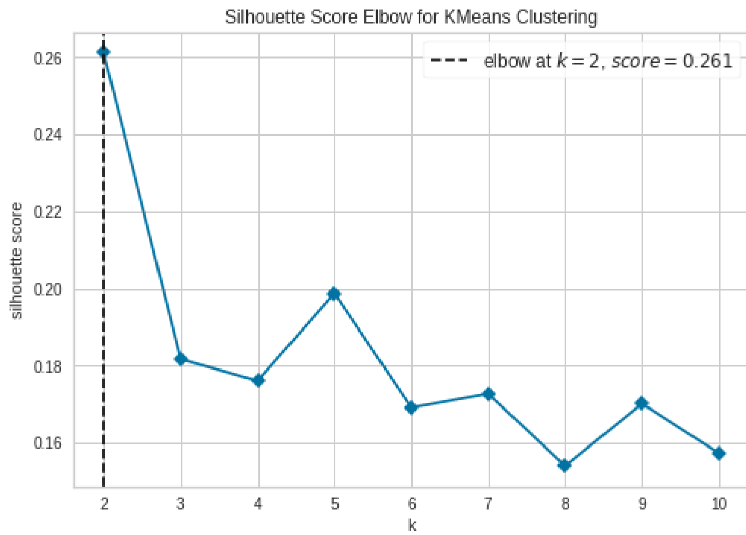


Minpoints combinations vs Number of clusters



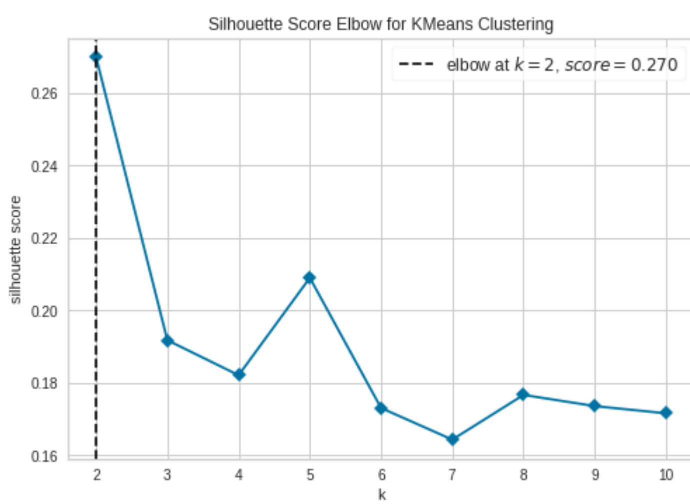
Q8)

Result from Silouette before PCA



Best K value is 2 and its score is 0.26

Result from Silhouette after PCA

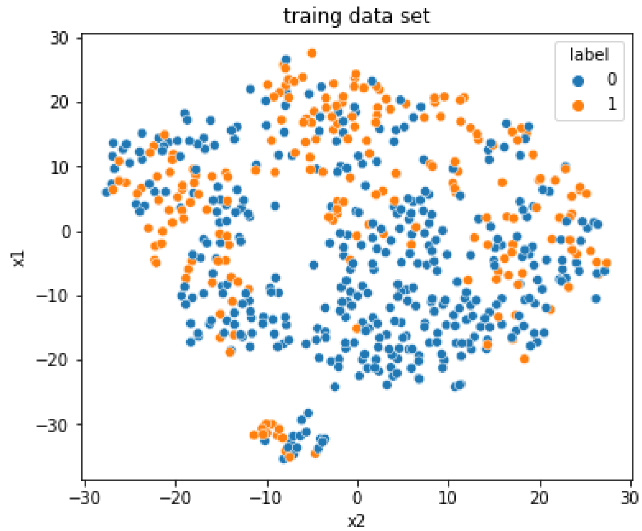


Best K value is 2 and its score is 0.27

Conclusion

After we applied the PCA on the data, the silhouette score slightly increased so that k mean clustering will improve.

b) Results of TSNE from Q1

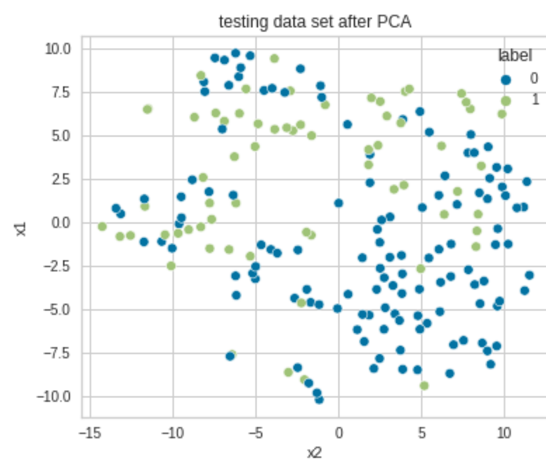


Results of TSNE from Q3 after dimentionality reduction

Logistic Regression Model:

Training data

Testing data



KNN Model:

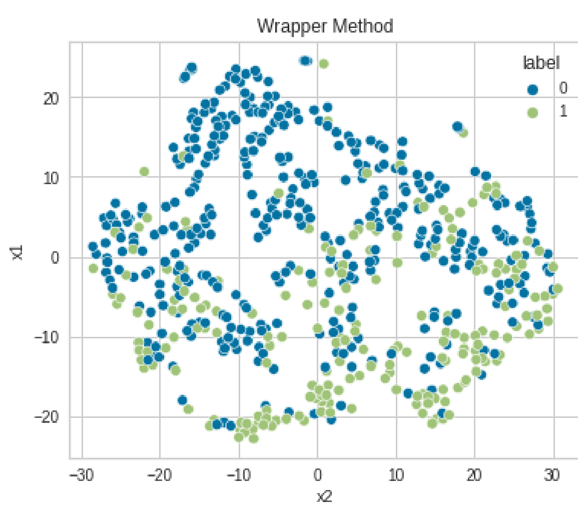
Training data

Testing data

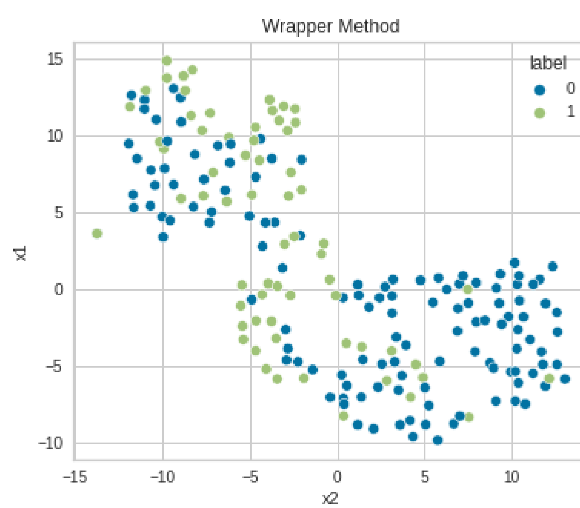
Results of TSNE from Q4 after feature selection

i. Logistic regression model

Training

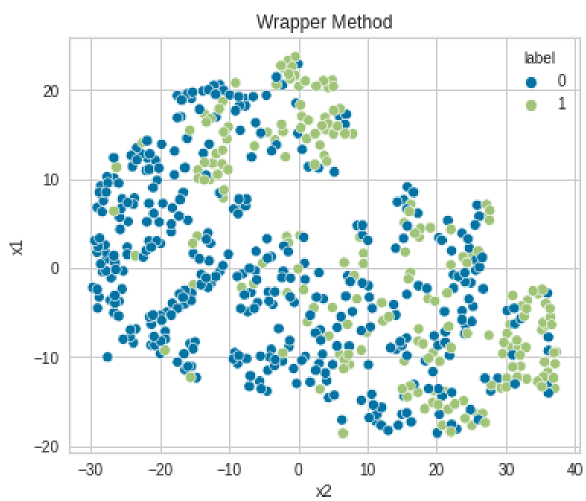


Testing

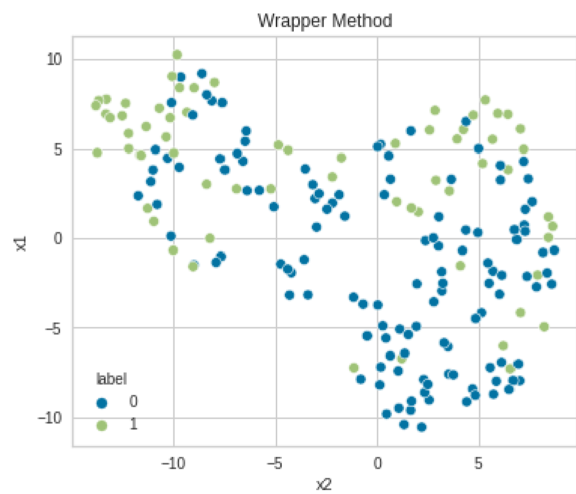


ii. KNN model

Training



Testing



Conclusion

After we applied the feature selection on the data as shown in the TSNE graph, the data points were classified better than the dimensionality reduction from question 3