Machine Learning Assignment 3

Team members names:

1.Abdulrahman Ahmed

2.Amir Youssef

3.Mohamed Elesawy

1. A1= (2,5), A2= (5,8), A3= (7,5), A4= (1,2), A5= (4,9)
   C1= A2, C2= A4
   a.

	C1	C2	Decision
A1	$\sqrt{(5-2)^2+(8-5)^2}$ $=\sqrt{9+9}=\sqrt{18}$ $=4.243$	$\sqrt{(1-2)^2+(2-5)^2}$ $=\sqrt{1+9}=\sqrt{10}$ $=3.162$	C2
A2	$\sqrt{(5-5)^2+(8-8)^2}$ $=\sqrt{0+0}=\sqrt{0}=0$	$\sqrt{(1-5)^2+(2-8)^2}$ $=\sqrt{16+36}=\sqrt{52}$ $=7.211$	C1
A3	$\sqrt{(7-5)^2+(5-8)^2}$ $=\sqrt{4+9}=\sqrt{13}$ $=3.742$	$\sqrt{(1-7)^2+(2-5)^2}$ $=\sqrt{36+9}=\sqrt{45}$ $=6.708$	C1
A4	$\sqrt{(5-1)^2+(8-2)^2}$ $=\sqrt{16+36}=\sqrt{52}$ $=7.211$	$\sqrt{(1-1)^2+(2-2)^2}$ $=\sqrt{0+0}=\sqrt{0}=0$	C2
A5	$\sqrt{(5-4)^2+(8-9)^2}$ $=\sqrt{1+1}=\sqrt{2}$ $=1.414$	$\sqrt{(1-4)^2+(2-9)^2}$ $=\sqrt{9+49}=\sqrt{58}$ $=7.616$	C1

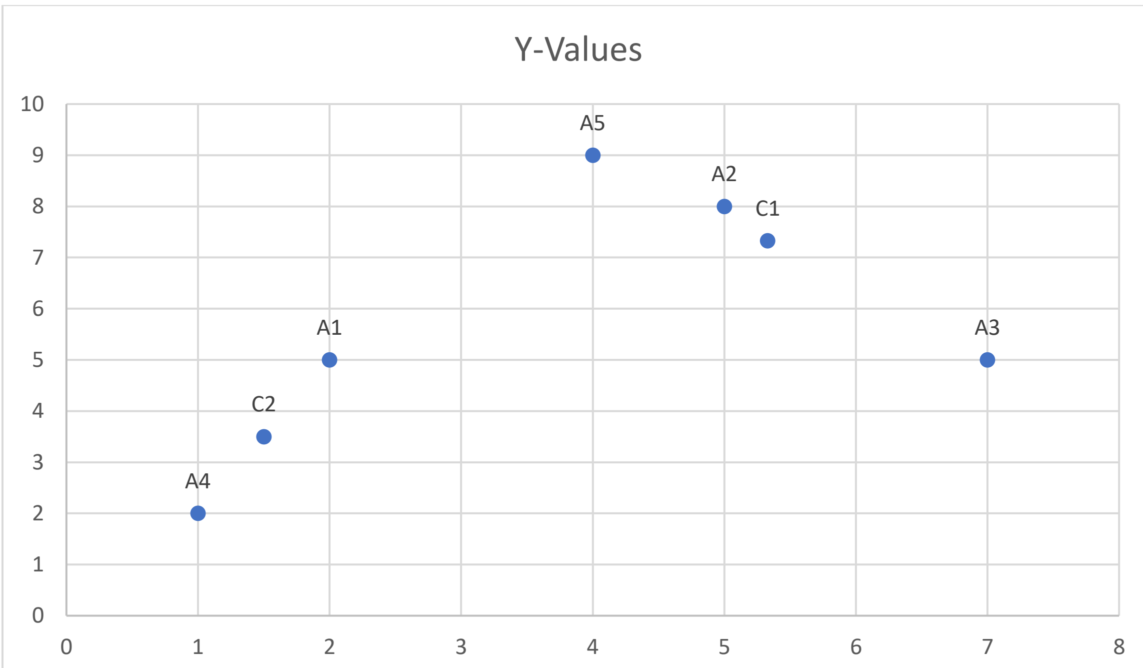
New C1 = $(\frac{5+7+4}{3}, \frac{8+5+9}{3}) = (5.33,\ 7.33)$

New C2 = $(\frac{2+1}{2}, \frac{5+2}{2}) = (1.5,\ 3.5)$

	C1	C2	Decision
A1	$\sqrt{(5.33-2)^2+(7.33-5)^2}$ $= \sqrt{11.0889+5.4289}$ $= \sqrt{16.5178} = 4.064$	$\sqrt{(1.5-2)^2+(3.5-5)^2}$ $= \sqrt{0.25+2.25} = \sqrt{2.5}$ $= 1.581$	C2
A2	$\sqrt{(5.33-5)^2+(7.33-8)^2}$ $= \sqrt{0.1089+0.4489}$ $= \sqrt{0.5578} = 0.747$	$\sqrt{(1.5-5)^2+(3.5-8)^2}$ $= \sqrt{12.25+20.25}$ $= \sqrt{32.5} = 5.701$	C1
A3	$\sqrt{(5.33-7)^2+(7.33-5)^2}$ $= \sqrt{2.7889+5.4289}$ $= \sqrt{8.2178} = 2.867$	$\sqrt{(1.5-7)^2+(3.5-5)^2}$ $= \sqrt{30.25+2.25}$ $= \sqrt{32.5} = 5.701$	C1
A4	$\sqrt{(5.33-1)^2+(7.33-2)^2}$ $= \sqrt{18.7489+28.4089}$ $= \sqrt{47.1578} = 6.867$	$\sqrt{(1.5-1)^2+(3.5-2)^2}$ $= \sqrt{0.25+2.25} = \sqrt{2.5}$ $= 1.581$	C2
A5	$\sqrt{(5.33-4)^2+(7.33-9)^2}$ $= \sqrt{1.7689+2.7889}$ $= \sqrt{4.5578} = 2.135$	$\sqrt{(1.5-4)^2+(3.5-9)^2}$ $= \sqrt{6.25+30.25}$ $= \sqrt{36.5} = 6.042$	C1

The algorithm converged after 1 iteration giving the same centroids.

b.



c.

	a(i)	b(i)	S(i)
A1	$\sqrt{(1-2)^2+(2-5)^2}$ $= \sqrt{1+9} = \sqrt{10}$ $= 3.162$	$\sqrt{(2-5)^2+(5-8)^2}$ $= \sqrt{9+9} = \sqrt{18}$ $= 4.243$ $\sqrt{(2-7)^2+(5-5)^2}$ $= \sqrt{25+0} = \sqrt{25}$ $= 5$ $\sqrt{(2-4)^2+(5-9)^2}$ $= \sqrt{4+16} = \sqrt{20}$ $= 4.472$	0.31

		4.243+5+4.472 = 13.715 13.715 / 3 = 4.572	
A2	$\sqrt{(5-7)^2+(8-5)^2}$ $=\ \sqrt{4+9}=\sqrt{13}$ $=\ 3.606$ $\sqrt{(5-4)^2+(8-9)^2}$ $=\ \sqrt{1+1}=\sqrt{2}$ $=\ 1.414$ 3.606+1.414 = 5.020 5.020 / 2 = 2.510	$\sqrt{(2-5)^2+(5-8)^2}$ $=\ \sqrt{9+9}=\sqrt{18}$ $=\ 4.243$ $\sqrt{(1-5)^2+(2-8)^2}$ $=\ \sqrt{16+36}=\sqrt{52}$ $=\ 7.211$ 4.243 + 7.211 = 11.454 11.454 / 2 = 5.727	0.56
A3	$\sqrt{(5-7)^2+(8-5)^2}$ $=\ \sqrt{4+9}=\sqrt{13}$ $=\ 3.606$ $\sqrt{(7-4)^2+(5-9)^2}$ $=\ \sqrt{9+16}=\sqrt{25}$ $=\ 5$ 3.606+5 = 8.606 8.606 / 2= 4.303	$\sqrt{(2-7)^2+(5-5)^2}$ $=\ \sqrt{25+0}=\sqrt{25}$ $=\ 5$ $\sqrt{(1-7)^2+(2-5)^2}$ $=\ \sqrt{36+9}=\sqrt{45}$ $=\ 6.708$ 5 + 6.708 = 11.708 11.708 / 2 = 5.854	0.26
A4	$\sqrt{(1-2)^2+(2-5)^2}$ $=\ \sqrt{1+9}=\sqrt{10}$ $=\ 3.162$	$\sqrt{(1-5)^2+(2-8)^2}$ $=\ \sqrt{16+36}=\sqrt{52}$ $=\ 7.211$ $\sqrt{(1-7)^2+(2-5)^2}$ $=\ \sqrt{36+9}=\sqrt{45}$ $=\ 6.708$ $\sqrt{(1-4)^2+(2-9)^2}$ $=\ \sqrt{9+49}=\sqrt{58}$ $=\ 7.616$ 7.211+6.708+7.616 = 21.535 21.535 / 3 = 7.178	0.56

A5	$\sqrt{(5-4)^2+(8-9)^2}$ $=\sqrt{1+1}=\sqrt{2}$ $=1.414$ $\sqrt{(7-4)^2+(5-9)^2}$ $=\sqrt{9+16}=\sqrt{25}$ $=5$ 5 + 1.414 = 6.414 6.414 / 2 = 3.207	$\sqrt{(1-4)^2+(2-9)^2}$ $=\sqrt{9+49}=\sqrt{58}$ $=7.616$ $\sqrt{(2-4)^2+(5-9)^2}$ $=\sqrt{4+16}=\sqrt{20}$ $=4.472$ 7.616 + 4.472 = 12.088 12.088 / 2 = 6.044	0.47

WSS = (5-5.33)^2 + (8-7.33)^2 +(7-5.33)^2 + (5-7.33)^2 + (4-5.33)^2 + (5-7.33)^2 +(2-1.5)^2 + (5-3.5)^2 + (1-1.5)^2 + (2-3.5)^2 = 18.33

2. Programming:

   2.1

   a.

   The data was split as requested, the first 576 rows for training and the last 192 rows for testing.

```
def supervsplting(df):
  X=df.iloc[:,:-1]
  y=df.iloc[:,-1]
  Xtrain , ytrain,Xtest,ytest= X[:576],y[:576],X[576:],y[576:]
  return Xtrain , ytrain,Xtest,ytest
```

   b.

   The baseline accuracy for each model was provided

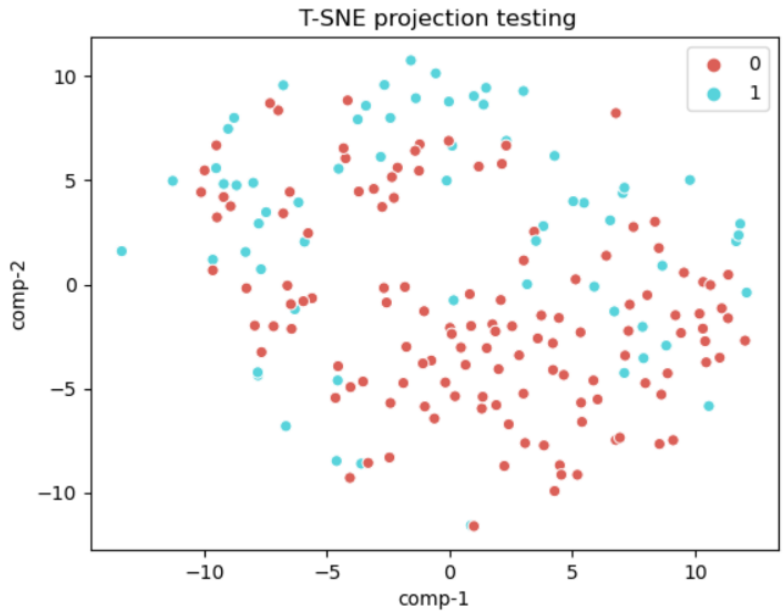
```
from sklearn.neighbors import KNeighborsClassifier
Knn = Pipeline(
    [
        (
            "KNN",
            KNeighborsClassifier(
            ),
        ),
    ]
)
knn_yhat,Knn_acc=get_predect(Knn,Xtrain, ytrain,Xtest,ytest)

```

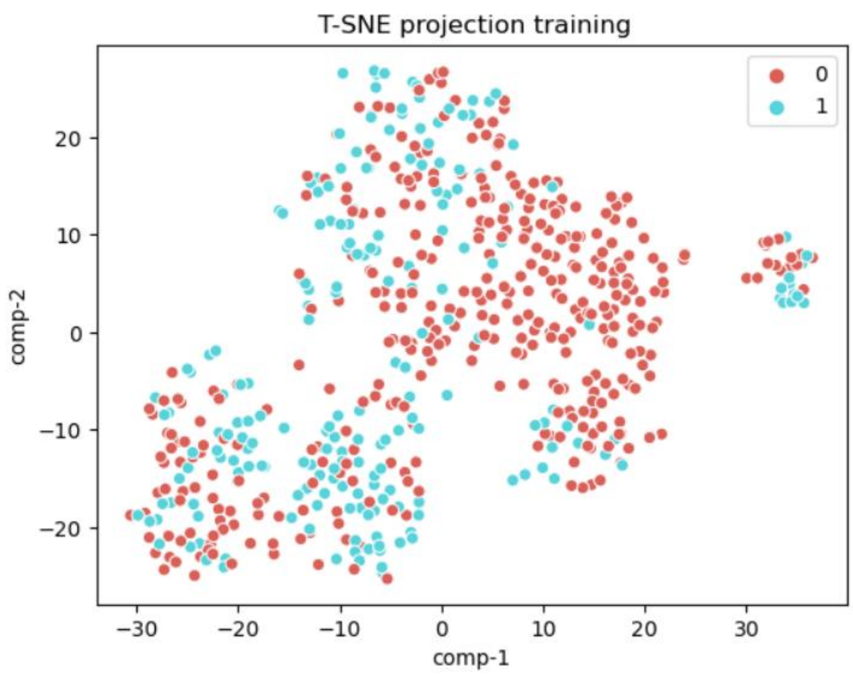
```
75.0
```

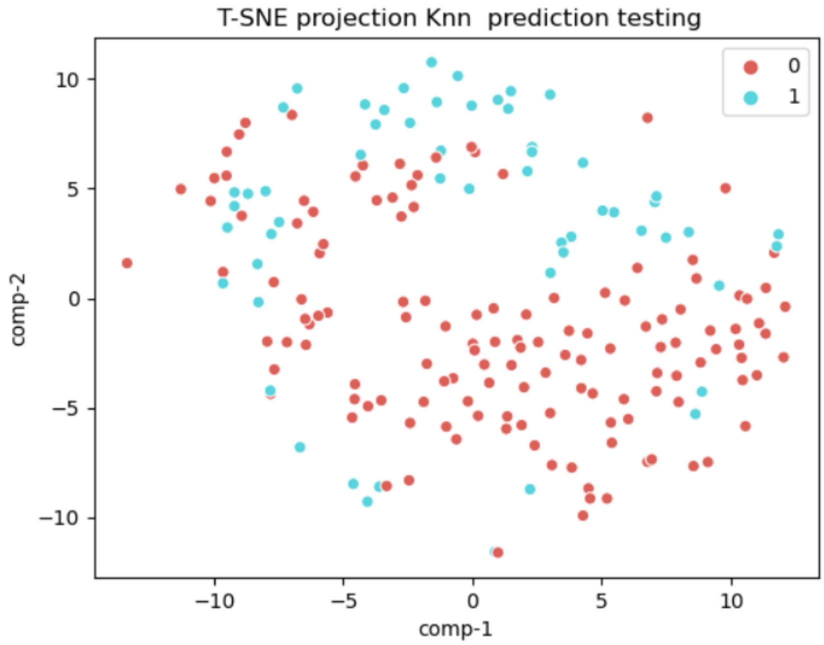
```
from sklearn.linear_model import LogisticRegression
LR = Pipeline(
    [
        (
            "LR",
            LogisticRegression(
            ),
        ),
    ]
)
LR_yhat,LR_acc=get_predect(LR,Xtrain, ytrain,Xtest,ytest)

```

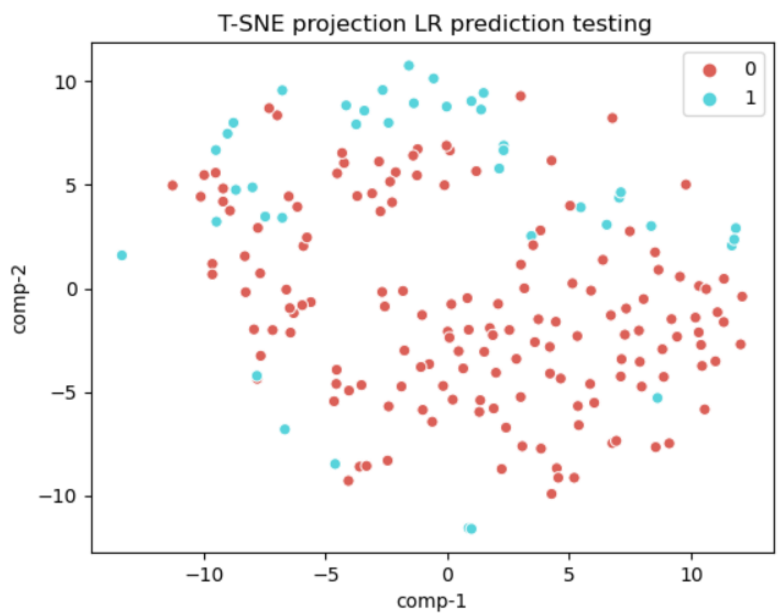
```
77.08333333333334
```

c. The training and testing set were plotted with TSNE as well as the predictions of each model on test set


T-SNE projection testing
comp-2
comp-1
0
1
10
5
0
−5
−10
−10
−5
0
5
10


T-SNE projection training
comp-2
comp-1
0
1
20
10
0
−10
−20
−30
−20
−10
0
10
20
30





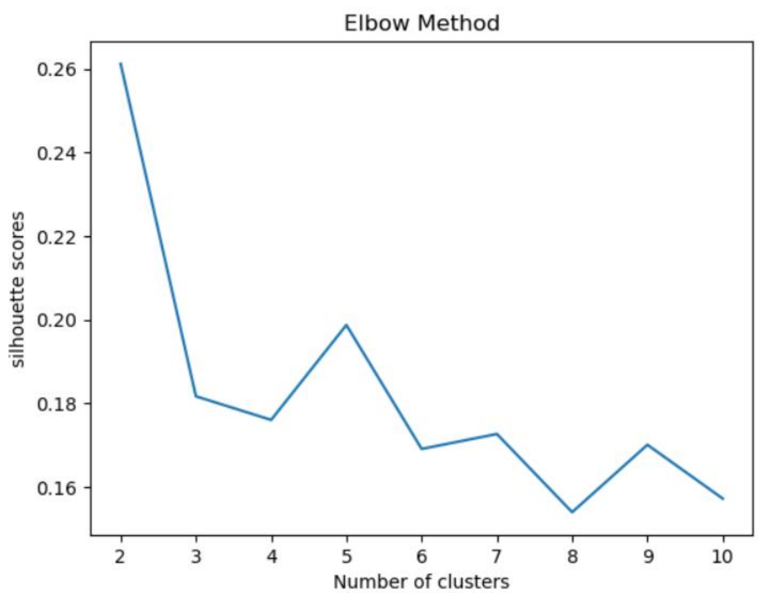
2.2.

a. The silhouette scores were calculated for every number of clusters and a line graph was plotted

```python
silhouette_scores = []
ari_scores = []
for n in range(2, 11):

    Kmeanclusterer["kmeans"].n_clusters= n
    Kmeanclusterer.fit(X)
    silhouette_coef = silhouette_score(X,
        Kmeanclusterer["kmeans"].labels_)

    # Add metrics to their lists
    silhouette_scores.append(silhouette_coef)
```

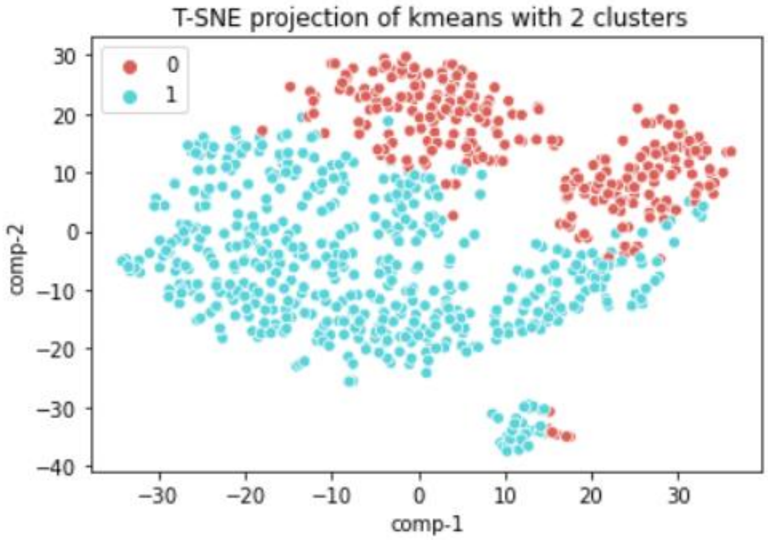
```python
silhouette_scores
```

```
[0.26114611150604655,
 0.18169513309010932,
 0.17605264582594735,
 0.19873872021116648,
 0.169117598675857 6,
 0.17268493735804227,
 0.15399722639062866,
 0.17010353169339418,
 0.1572624866433923]
```



b. Best silhouette score can be shown at k = 2 from the graph

c. TSNE was used to show data after clustering with 2 clusters (optimum number)



2.3

a. Each number of pca components were tested on each classifier to figure out the best for both

23]:

```
Lraccs = []
for i in range(2, 9):
    Pca["pca"].n_components= i

    newXtr=Pca.fit_transform(Xtrain)
    newXte=Pca.transform(Xtest)
    ypred,acc=get_predect(LR,newXtr, ytrain,newXte,ytest)
    Lraccs.append(acc)
plt.plot(range(2, 9), [LR_acc,LR_acc,LR_acc,LR_acc,LR_acc,LR_acc,LR_a

plt.plot(range(2, 9), Lraccs)
plt.title('number components VS acc')
plt.xlabel('number components')
plt.ylabel('LR acc')
plt.show()
```

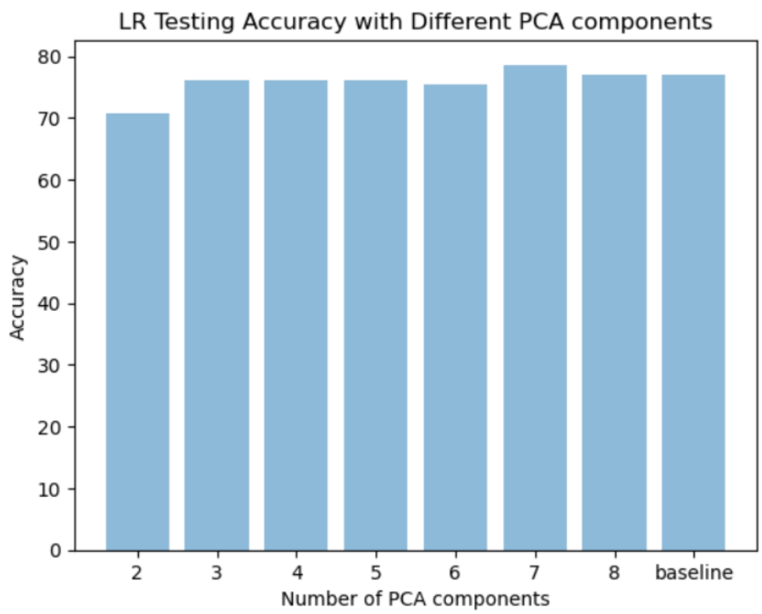
```
70.83333333333334
76.04166666666666
76.04166666666666
76.04166666666666
75.52083333333334
78.64583333333334
77.08333333333334
```

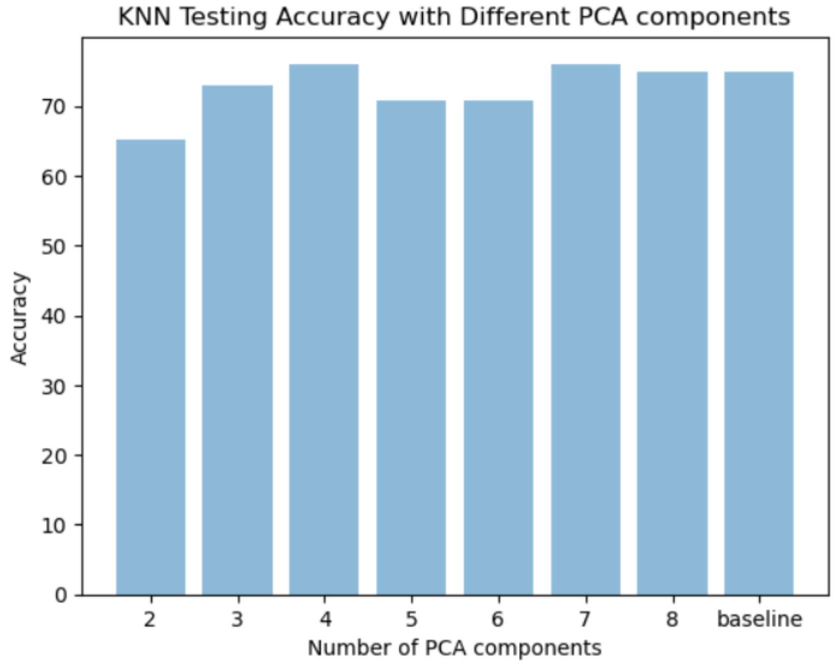
5]:

```python
Knnaccs = []
for i in range(2, 9):
    Pca["pca"].n_components= i

    newXtr=Pca.fit_transform(Xtrain)
    newXte=Pca.transform(Xtest)
    ypred,acc=get_predect(Knn,newXtr, ytrain,newXte,ytest)
    Knnaccs.append(acc)

plt.plot(range(2, 9), [Knn_acc,Knn_acc,Knn_acc,Knn_acc,Knn_acc,Knn_acc,Knn_acc])
plt.plot(range(2, 9), Knnaccs)

plt.title('number components VS acc')
plt.xlabel('number components')
plt.ylabel('Knn acc')
plt.show()
```

```
65.10416666666666
72.91666666666666
76.04166666666666
70.83333333333334
70.83333333333334
76.04166666666666
75.0
```

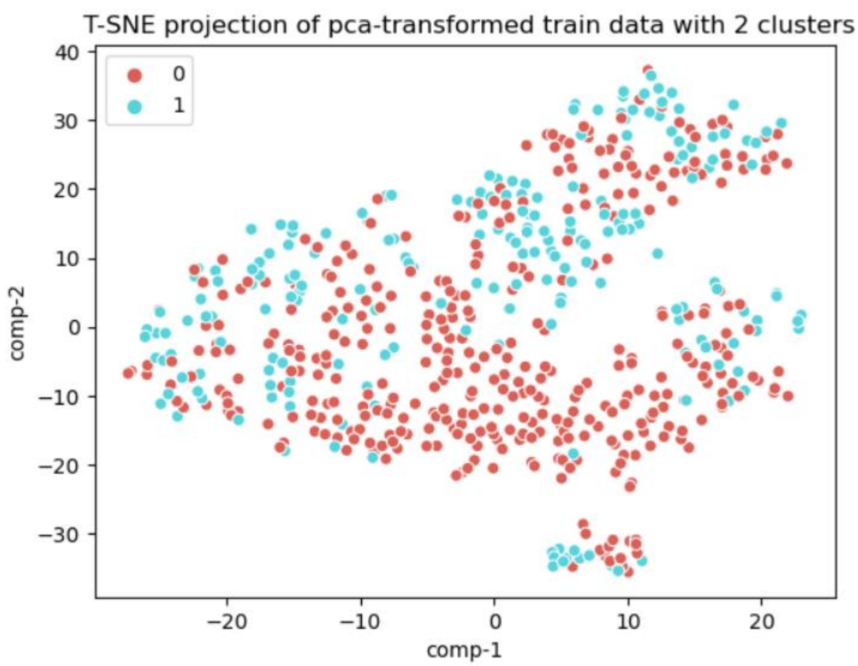
b. The Number of Components-Accuracy graph with baseline performances was plotted for each classifier

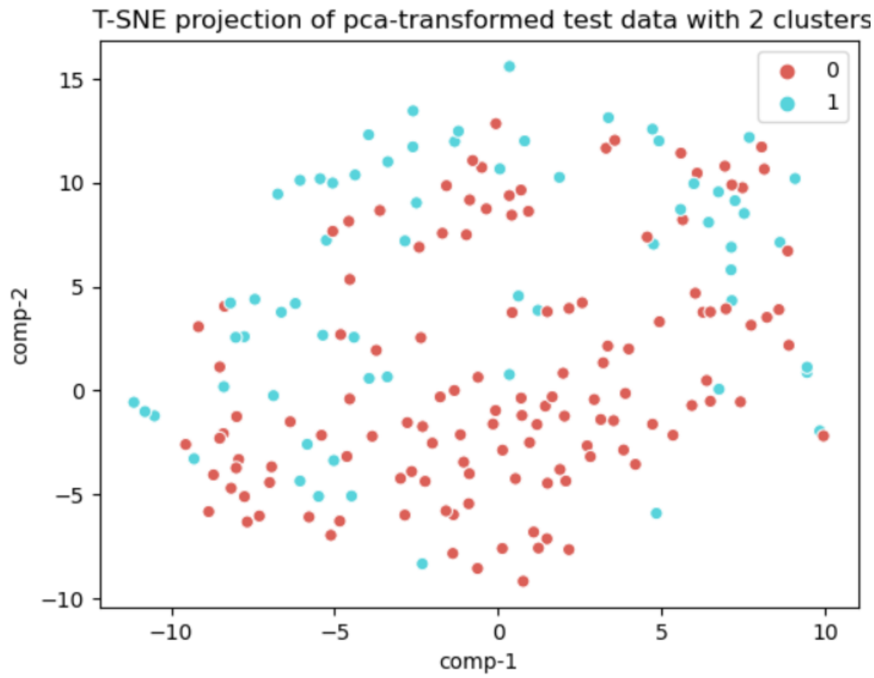




It is clear in the graph that seven dimension is best for all classifiers.

c. The TSNE plots were used to show the training and testing datasets.





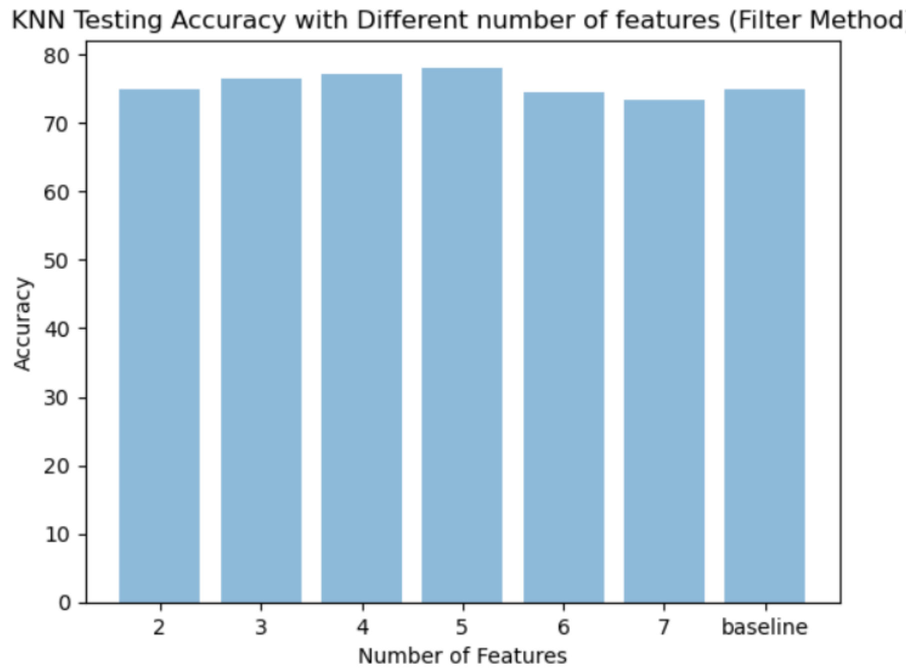
2.4

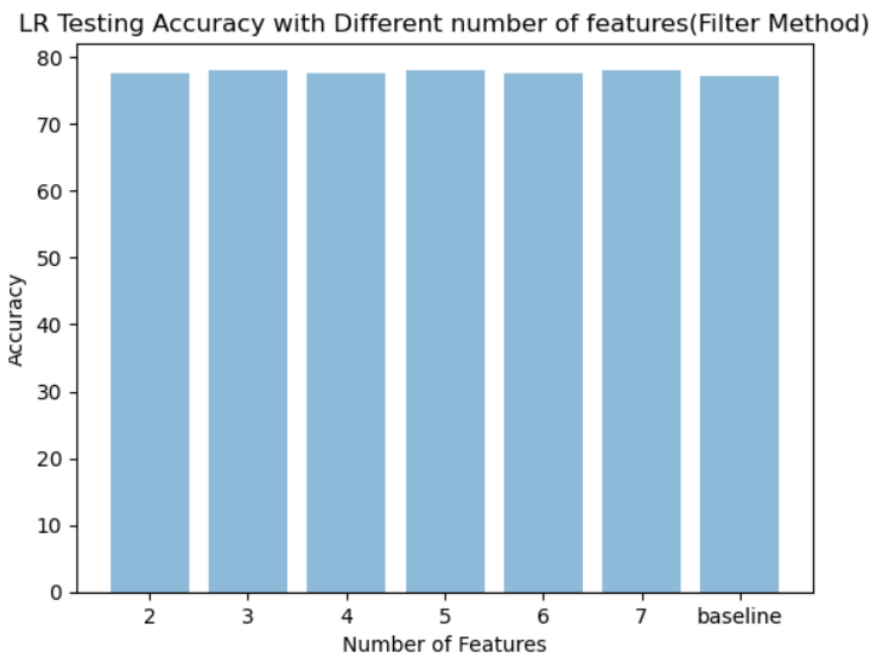
a.

Chi Square method was used for filtering features based on their correlation with the target.

```
def filter(pip,Xtrain, ytrain,Xtest,ytest):
  acclis=[]
  pred=[]
  from sklearn.feature_selection import chi2

  for i in range(2,8):
    sel_five_cols = SelectKBest(chi2, k=i)
    sel_five_cols.fit(Xtrain, ytrain)
    newXtrain=Xtrain[Xtrain.columns[sel_five_cols.get_support()]]
    newXtest=Xtest[Xtrain.columns[sel_five_cols.get_support()]]
    ypred,acc=get_predect(pip,newXtrain, ytrain,newXtest,ytest)
    acclis.append(acc)
    pred.append(ypred)
  return pred,acclis
```

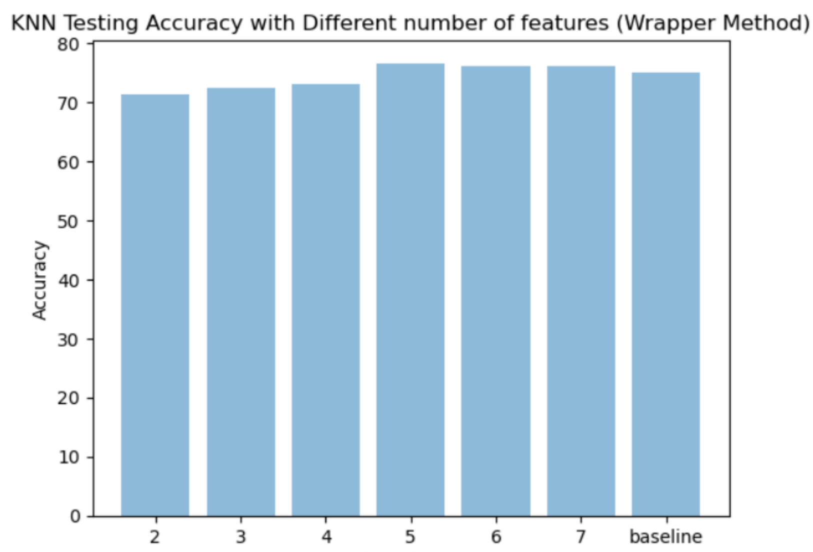
Number of features vs test accuracy was plotted for each classifier

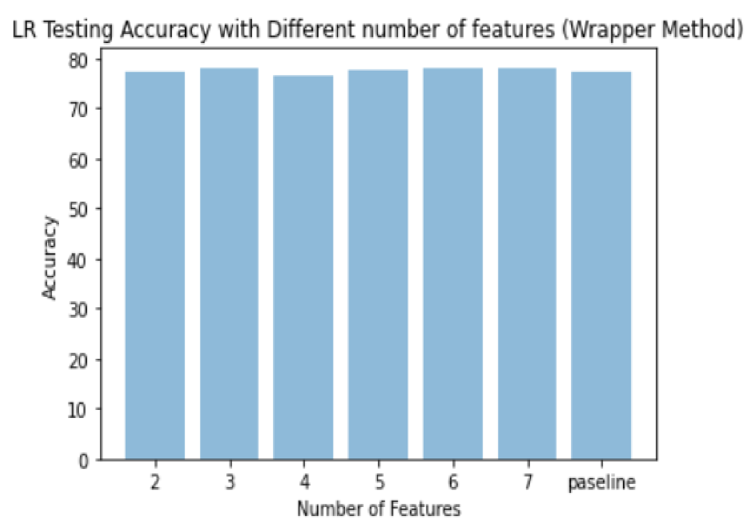

KNN Testing Accuracy with Different number of features (Filter Method)
Accuracy
80
70
60
50
40
30
20
10
0
2
3
4
5
6
7
baseline
Number of Features


LR Testing Accuracy with Different number of features(Filter Method)
Accuracy
80
70
60
50
40
30
20
10
0
2
3
4
5
6
7
baseline
Number of Features

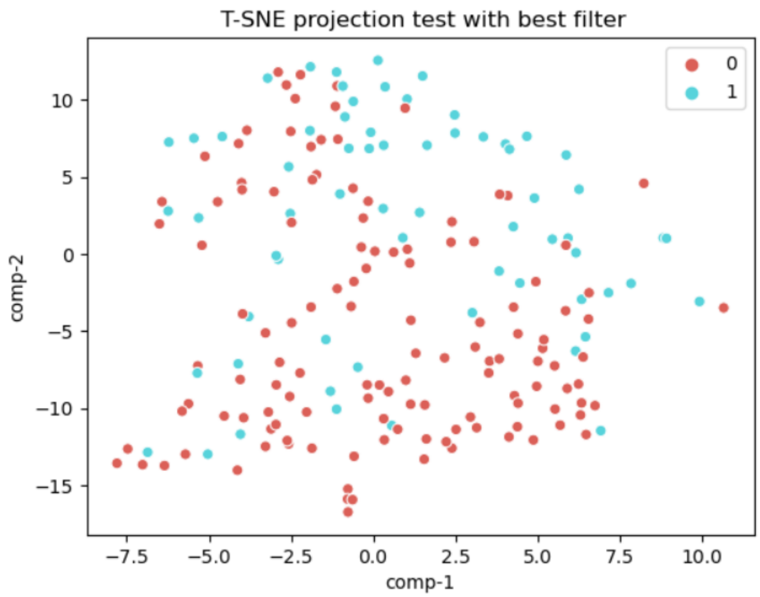
b. Sequential Feature Selection was used for Wrapper method.

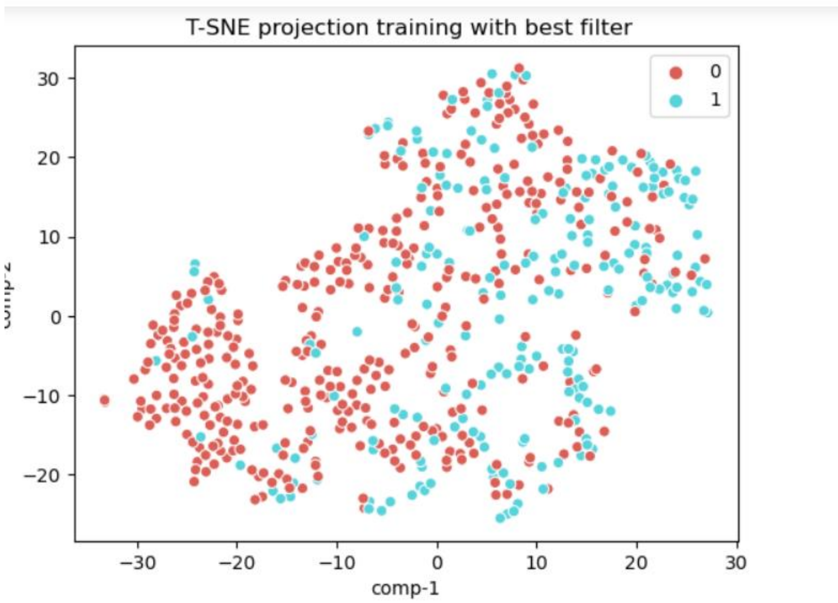
```
def warber(pip,Xtrain, ytrain,Xtest,ytest):  #importing the necessary libraries
  # Sequential Forward Selection(sfs)
  acclis=[]
  pred=[]

  for i in range(2,8):

    sfs = SequentialFeatureSelector(pip, n_features_to_select=i)
    sfs.fit(Xtrain, ytrain)
    newii=sfs.transform(Xtrain)
    newii2=sfs.transform(Xtest)


    ypred,acc=get_predect(pip,newii, ytrain,newii2,ytest)
    acclis.append(acc)
    pred.append(ypred)
  return pred,acclis
```

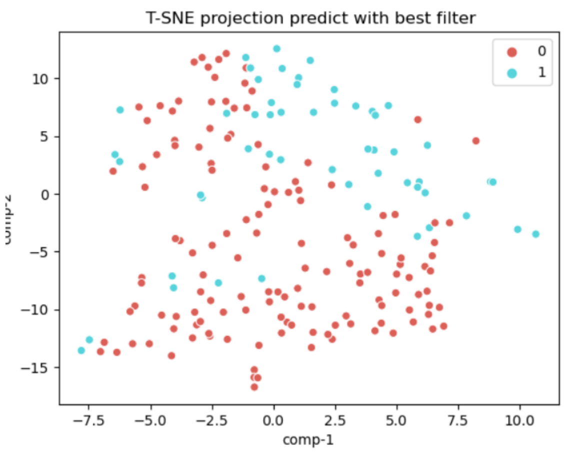




c. The TSNE for best feature selection method was plotted for training and testing set.





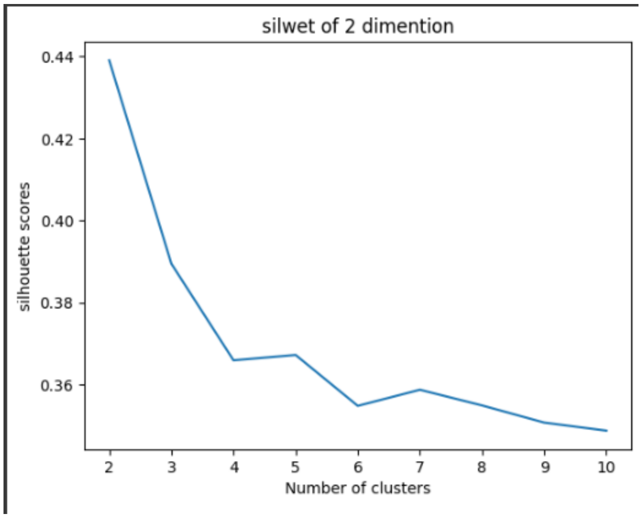


Problem 2.5:

A. In this problem we tried multiple combination to find the best number of dimantions vs number of cluster using silhouette score as matric

```
for i in range(2, 9):
    Pca["pca"].n_components= i

    newX=Pca.fit_transform(X)
    silhouette_scores = []
    arcores = []
    for n in range(2, 11):

        Kmeanclusterer["kmeans"].n_clusters= n
        Kmeanclusterer.fit(newX)
        silhouette_coef = silhouette_score(newX,
            Kmeanclusterer["kmeans"].labels_)

        # Add metrics to their lists
        silhouette_scores.append(silhouette_coef)
    plt.plot(range(2, 11), silhouette_scores)
    plt.title('silwet of {:} dimention'.format(i))
    plt.xlabel('Number of clusters')
    plt.ylabel('silhouette scores')
    plt.show()
```

B. And we fond that having only 2 dimensions only has the highest silhouette



Problem 2.6:

A. Using the code blow we have run som model with deffrent number of neurons

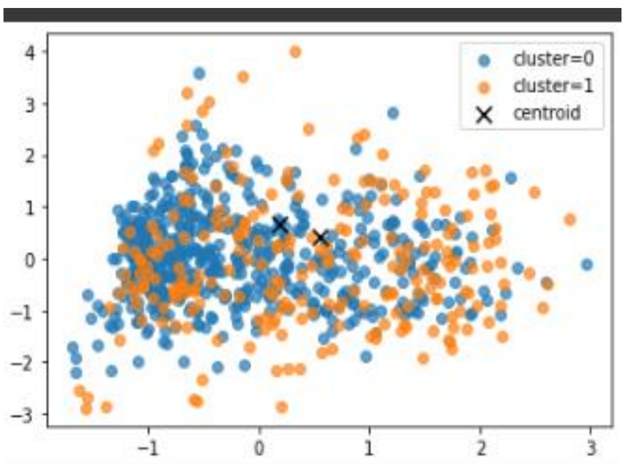
```
_, dim=newX.shape
somsh=[]
for i in range(2,31) :
        model=SOM(i, n=1, dim=dim)

        predLabels = model.fit_predict(newX)
        score = silhouette_score(newX, predLabels, random_state=0)
        somsh.append(score)
        print(f'{model.__class__.__name__}: '
              f'Silhouette Score = {score}; ')
```

B. We found that the optimal number of neurons2 which give us silhouette =0.44

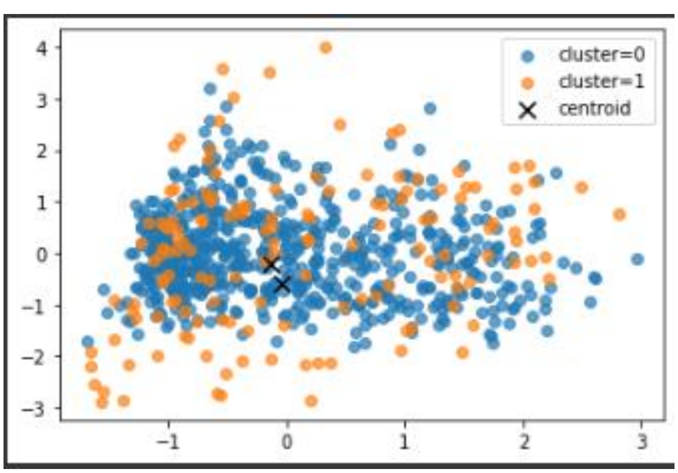
C. For this point we trained the minisom model frist with 1 ittration then done the full training using the best Pca componant witch was 7 from q3

```python
from minisom import MiniSom
import numpy as np
import pandas as pd
Pca["pca"].n_components=7

newX=Pca.fit_transform(X)

data = newX
# data normalization
data = (data - np.mean(data, axis=0)) / np.std(data, axis=0)


# Initialization and training
som_shape = (1, 2)

som = MiniSom(som_shape[0], som_shape[1], data.shape[1], sigma=.5, learning_rate=.5,
              neighborhood_function='gaussian', random_seed=0)
k = MiniSom(som_shape[0], som_shape[1], data.shape[1], sigma=.5, learning_rate=.5,
              neighborhood_function='gaussian', random_seed=0)
som.train_batch(data, 1000, verbose=True)
k.train_batch(data, 1, verbose=True)
```

And got the initial Neuron positions like this



To the final Neuron positions



Problem 2.7:

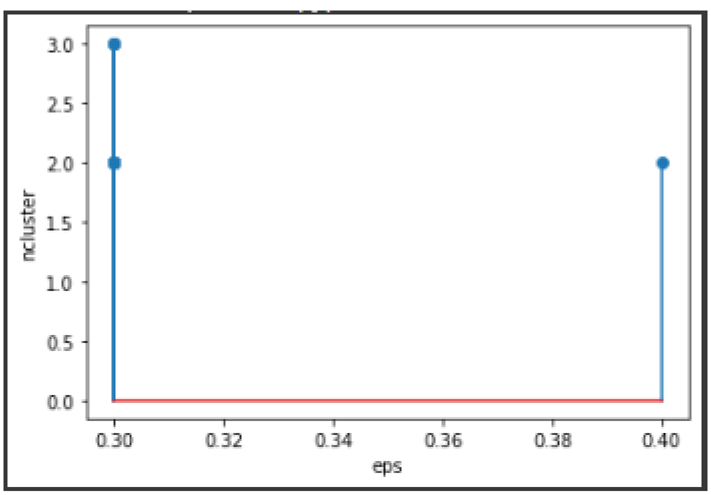
A. In here we didn't want to take the anomaly into the clustering so we conted the cluster with 2 or more because the matric of silhouette doesn't work with one cluster we got this combination

```
epsilon=[0.3,.4,.5,.6,0.7]
minpoints=range(2,16)
mo = pd.DataFrame()
#find DBSCAN optimal eps and min-samples
epsList, msList, ncluster,labels = list(), list(), list(),list()
for eps in epsilon:
  for ms in minpoints:
    model = DBSCAN(eps=eps, min_samples=ms).fit(X)
    label =model.labels_
#3shan anomoly ems7  - (1 if -1 in label else 0)
    numc=(len(set(label)) - (1 if -1 in label else 0))

    epsList.append(eps)
    msList.append(ms)
    ncluster.append(numc)
    labels.append(label)
mo['epsList'],mo['msList'],mo['ncluster'],mo['labels']=epsList, msList, ncluster,labels
```

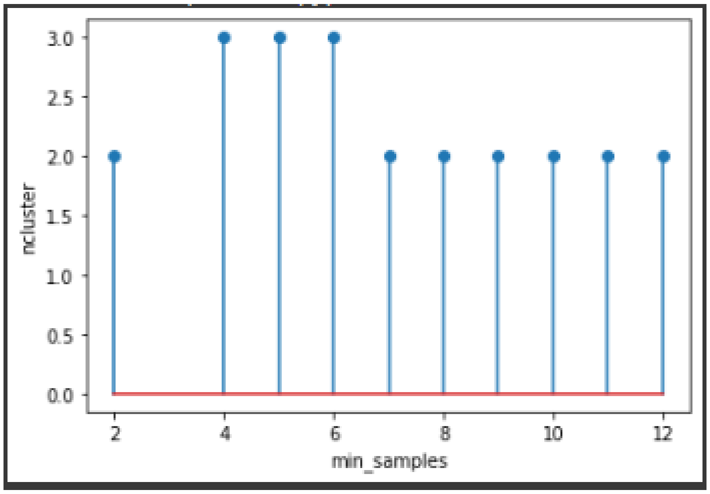
14	0.4	2	2	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	0.286938
2	0.3	4	3	[0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, ...	0.255704
5	0.3	7	2	[0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, ...	0.255690
3	0.3	5	3	[0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, ...	0.255503
6	0.3	8	2	[0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, ...	0.255465
4	0.3	6	3	[0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, ...	0.250714
7	0.3	9	2	[0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0,...	0.241373
8	0.3	10	2	[0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0,...	0.234830
9	0.3	11	2	[0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0,...	0.231672
10	0.3	12	2	[0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0,...	0.223967

We got the following

A- Epsilon VS number of cluster



B. Minpoint VS number of clusters



Problem 2.8:

A. The difference between Q2 and Q5 is that when we reduce the dimension we fond that the silhouette score has risen but they both have 2 as the optimal number of cluster

We found that the the filter for Knn and Wrapper with LR had a high accuracy which made the t-sne for the prediction become nearer to the actual label also the shape of the data on the t-sne chang each time we try different method with data either it was DR or FS