Applied machine learning
Group assignment 3

Team members-Group 8:

• Abdelrhman Gaber Youssef Saad Rezkallah
• Eman Metwally Mohammed Abood
• Basma Reda Shaban Abd-Elsalam Abd-Elwahab

Part 1: calculation

Solution:

a) d(a,b) denotes the Euclidean distance between a and b.

It is obtained directly from the distance matrix or calculated as follows:

$d(a, b) = \sqrt{(xb - xa)^2 + (yb - ya)^2}$

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

The initial centroids → A2, A4

For A1

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

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

A1 ∈ A4 , A1 ∈ Cluster2

For A2

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

$d(A1, A4) = \sqrt{(1-5)^2 + (2-8)^2}$ = $2\sqrt{13}$

A2 ∈ A2 , A2 ∈ Cluster1

For A3

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

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

A3 ∈ A2 , A3 ∈ Cluster1

For A4

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

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

A4 ∈ A4 , A4 ∈ Cluster2

For A5

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

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

A5 ∈ A2 , A5 ∈ Cluster1

New clusters :-

1 :{A2 , A3 , A5}　　　　**2**: {A1 , A4}

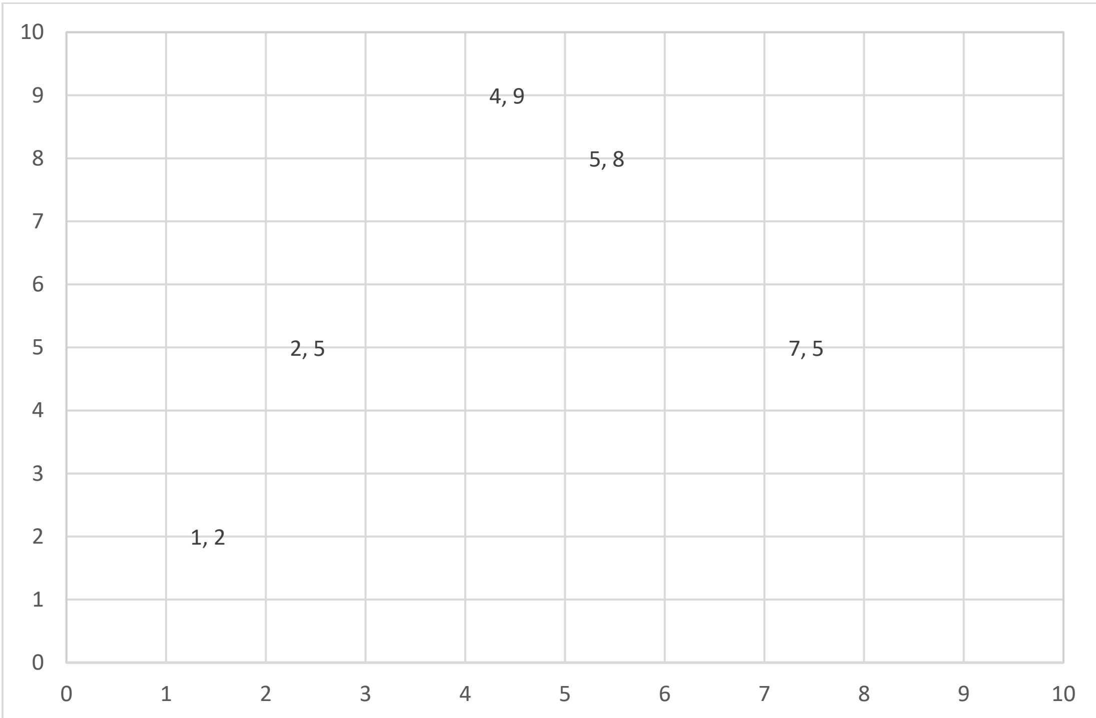
The Mean for each clusters :-

Cluster 1 = $\left(\frac{5+7+4}{3}, \frac{8+5+9}{3}\right) = (\frac{16}{3}, \frac{22}{3})$

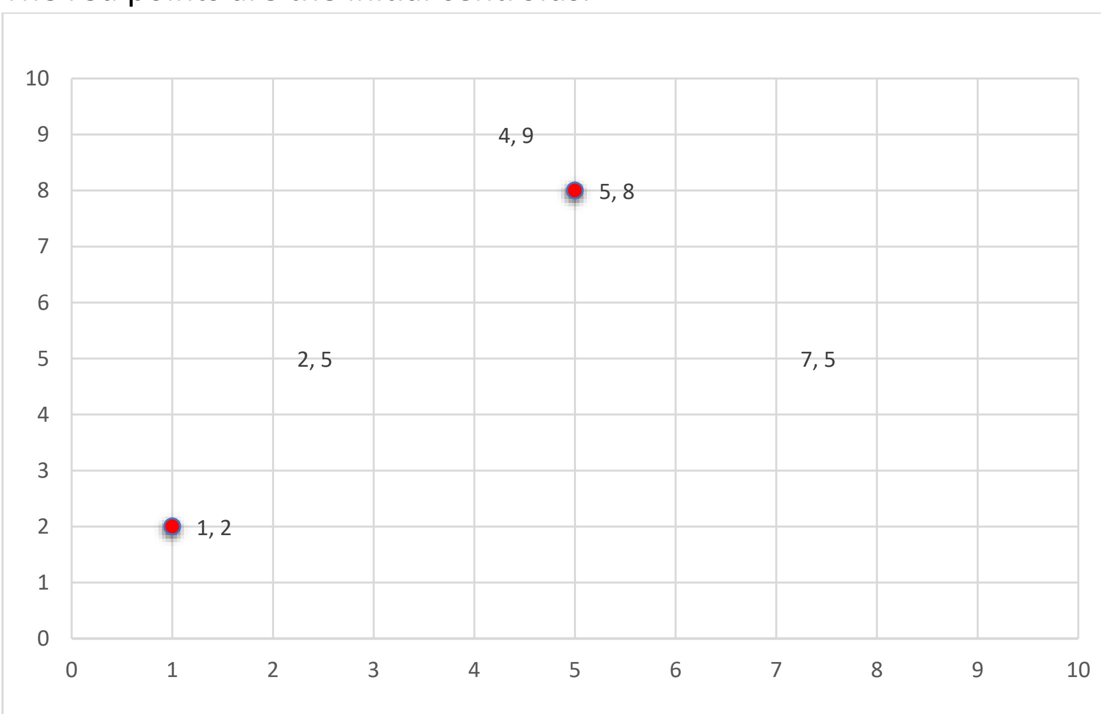
Cluster 2 = $\left(\frac{1+2}{2}, \frac{5+2}{2}\right) = (\frac{3}{2}, \frac{7}{2})$

b)

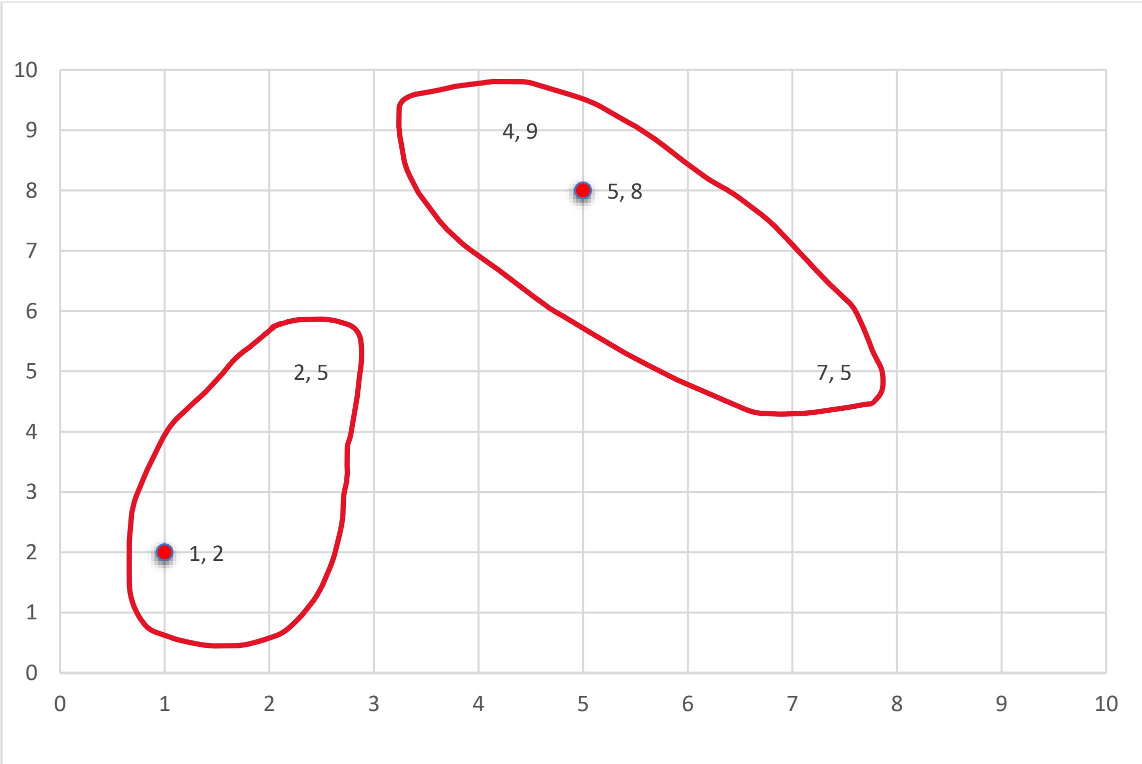
The locations for the data points:



The red points are the initial centroids.



The clusters of cluster 1 and cluster 2:



The X points are the centroid for each clusters.

c)

Calculate the silhouette score and WSS score.

1. **Calculate WSS score.**

For calculating WSS the equation equal:

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

WSS $= (5-5.3)^2 + (8-7.3)^2 + (7-5.3)^2 + (5-7.3)^2 + (4-5.3)^2 + (9-7.3)^2 + (2-1.5)^2 + (5-3.5)^2 + (1-1.5)^2 + (2-3.5)^2$

= 18.34

2. **Calculate silhouette score.**

For calculating silhouette the equation equal:

$$S(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

According to cluster 1 :

A2 :

- d(A2,A3)= $\sqrt{(5-7)^2 + (8-5)^2}$ = 3.6
- d(A2,A5)= $\sqrt{(5-4)^2 + (8-9)^2}$ = 1.4
- a(A2) = $\frac{\text{d(A2,A3)} + \text{d(A2,A5)}}{2} = \frac{3.6 + 1.4}{2}$ = 2.5
- d(A2,A1)= $\sqrt{(5-2)^2 + (8-5)^2}$ = 4.2
- d(A2,A4)= $\sqrt{(5-1)^2 + (8-2)^2}$ = 7.2
- b(A2) = $\frac{4.2 + 7.2}{2}$ = 5.7

➢ $S(\text{A2}) = \frac{b(\text{A2}) - a(\text{A2})}{\max\{a(\text{A2}), b(\text{A2})\}} = \frac{5.7 - 2.5}{5.7}$ = 0.56

A3 :

- d(A3,A2)= $\sqrt{(5-7)^2 + (8-5)^2}$ = 3.6
- d(A3,A5)= $\sqrt{(7-4)^2 + (5-9)^2}$ = 5

- a(A3) = $\frac{\mathrm{d(A3,A2)} + \mathrm{(A3,A5))}}{2}$ = $\frac{3.6 + 5}{2}$ = 4.3

- d(A3,A1)= $\sqrt{(7-2)^2 + (5-5)^2}$ = 5
- d(A3,A4)= $\sqrt{(7-1)^2 + (5-2)^2}$ = 6.7

- b(A3) = $\frac{5 + 6.7}{2}$ = 5.85

➢ $S(\mathrm{A3}) = \frac{b(\mathrm{A3}) - a(\mathrm{A3})}{\max\{a(\mathrm{A3}), b(\mathrm{A3})\}}$ = $\frac{5.85 - 4.3}{5.85}$ = 0.26

A5 :

- d(A5,A2)= $\sqrt{(4-5)^2 + (9-8)^2}$ = 1.4

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

- a(A5) = $\frac{\mathrm{d(A5,A2)} + \mathrm{(A5,A3))}}{2}$ = $\frac{1.4 + 5}{2}$ = 3.2

- d(A5,A1)= $\sqrt{(4-2)^2 + (9-5)^2}$ = 4.4
- d(A5,A4)= $\sqrt{(4-1)^2 + (9-2)^2}$ = 7.6

- b(A5) = $\frac{4.4 + 7.6}{2}$ = 6

➢ $S(\mathrm{A5}) = \frac{b(\mathrm{A5}) - a(\mathrm{A5})}{\max\{a(\mathrm{A5}), b(\mathrm{A5})\}}$ = $\frac{6 - 3.2}{6}$ = 0.46

According to cluster 2 :

A1 :

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

- a(A1) = $\frac{\mathrm{d(A1,A4)} + \mathrm{d(A1,A4)}}{1}$ = $\frac{3.16}{1}$ = 3.16

- d(A1,A2)= $\sqrt{(5-2)^2 + (8-5)^2}$ = 4.2

- d(A1,A3)= $\sqrt{(7-2)^2+(5-5)^2}$ = 5
- d(A5,A1)= $\sqrt{(4-2)^2+(9-5)^2}$ = 4.4

- b(A1) = $\frac{4.2+\ 5+4.4}{3}$ = 4.53

➢ $S(\mathrm{A1}) = \frac{b(\mathrm{A1})-a(\mathrm{A1})}{\max\{a(\mathrm{A1}),b(\mathrm{A1})\}} = \frac{4.53-3.16}{4.53}$ = 0.302

A4 :

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

- a(A4) = $\frac{\mathrm{d(A1,A4)}+\mathrm{d(A1,A4)}}{1} = \frac{3.16}{1}$ = 3.16

- d(A4,A2)= $\sqrt{(5-1)^2+(8-2)^2}$ = 7.2
- d(A4,A3)= $\sqrt{(7-1)^2+(5-2)^2}$ = 6.7
- d(A4,A5)= $\sqrt{(4-1)^2+(9-2)^2}$ = 7.6

- b(A4) = $\frac{7.2+\ 6.7+\ 7.6}{3}$ = 7.166

➢ $S(\mathrm{A4}) = \frac{b(\mathrm{A4})-a(\mathrm{A4})}{\max\{a(\mathrm{A4}),b(\mathrm{A4})\}} = \frac{7.166-3.16}{7.166}$ = 0.55

AVG S(i) = $\frac{0.302+0.55+0.46+0.26+0.56}{5}$ = 0.43

Part 2: programming

Import important Libraries and load the dataset

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	0.411765	0.623116	0.573770	0.333333	0.254137	0.380030	0.035440	0.266667	0
1	0.294118	0.542714	0.590164	0.434343	0.088652	0.538003	0.078992	0.200000	0
2	0.058824	0.437186	0.491803	0.373737	0.088652	0.554396	0.184031	0.016667	0
3	0.058824	0.723618	0.672131	0.464646	0.212766	0.687034	0.109735	0.416667	1
4	0.058824	0.557789	0.508197	0.131313	0.215130	0.357675	0.025619	0.033333	0

get some information about the data and describe it:

```
#get some informations about the data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    float64
 1   Glucose                   768 non-null    float64
 2   BloodPressure             768 non-null    float64
 3   SkinThickness             768 non-null    float64
 4   Insulin                   768 non-null    float64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    float64
 8   Outcome                   768 non-null    int64  
dtypes: float64(8), int64(1)
memory usage: 54.1 KB
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	0.226180	0.607510	0.566438	0.207439	0.094326	0.476790	0.168179	0.204015	0.348958
std	0.198210	0.160666	0.158654	0.161134	0.136222	0.117499	0.141473	0.196004	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.058824	0.497487	0.508197	0.000000	0.000000	0.406855	0.070773	0.050000	0.000000
50%	0.176471	0.587940	0.590164	0.232323	0.036052	0.476900	0.125747	0.133333	0.000000
75%	0.352941	0.704774	0.655738	0.323232	0.150414	0.545455	0.234095	0.333333	1.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

1- Split the dataset to train and test with random_state = 0

```python
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,shuffle=False)
```

X_train will be:

```
[7] x_train
```

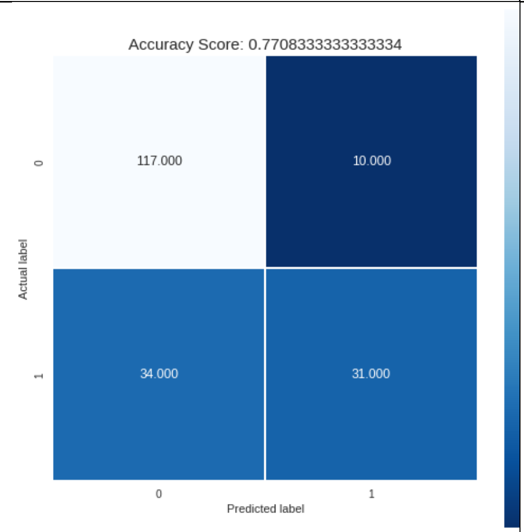
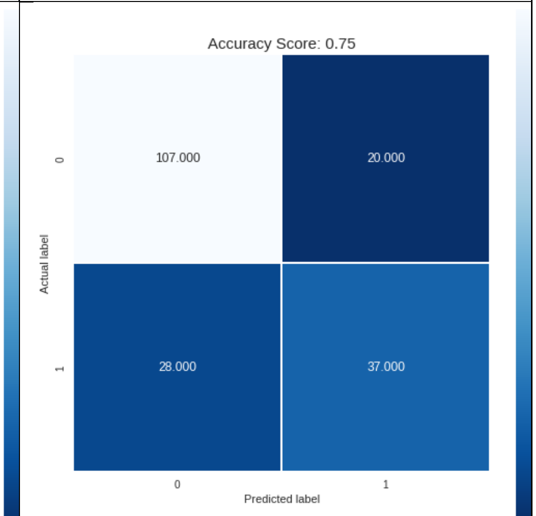
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Ag
0	0.411765	0.623116	0.573770	0.333333	0.254137	0.380030	0.035440	0.26666
1	0.294118	0.542714	0.590164	0.434343	0.088652	0.538003	0.078992	0.20000
2	0.058824	0.437186	0.491803	0.373737	0.088652	0.554396	0.184031	0.01666
3	0.058824	0.723618	0.672131	0.464646	0.212766	0.687034	0.109735	0.41666
4	0.058824	0.557789	0.508197	0.131313	0.215130	0.357675	0.025619	0.03333
...	...	...	...	...	...	...	...	.
571	0.117647	0.874372	0.721311	0.373737	0.141844	0.663189	0.242528	0.05000
572	0.352941	0.628141	0.622951	0.000000	0.000000	0.503726	0.018360	0.55000
573	0.529412	0.763819	0.639344	0.343434	0.202128	0.509687	0.347993	0.20000
574	0.058824	0.467337	0.573770	0.313131	0.000000	0.453055	0.101196	0.03333
575	0.117647	0.527638	0.475410	0.404040	0.111111	0.520119	0.062767	0.06666

576 rows × 8 columns

2- Train Logistic regression model and KNN model:

Logistic regression estimates the probability of an event occurring, such as voted or didn't vote, based on a given dataset of independent variables. Since the outcome is a probability, the dependent variable is bounded between 0 and 1.

K Nearest Neighbor algorithm falls under the Supervised Learning category and is used for classification (most commonly) and regression. It is a versatile algorithm also used for imputing missing values and resampling datasets. As the name (K Nearest Neighbor) suggests it considers K Nearest Neighbors (Data points) to predict the class or continuous value for the new Datapoint.

<table>
<tr><th></th><th>Logistic Regression</th><th>KNN</th></tr>
<tr><td>Accuracy score</td><td>0.7708333333333334</td><td>0.75</td></tr>
<tr><td>Confusion matrix</td><td>
</td><td>
</td></tr>
<tr><td>Classification report for training</td><td>
<pre>
              precision   recall f1-score   support

           0   0.78   0.91   0.84   373
           1   0.76   0.53   0.62   203

    accuracy   0.78   576
   macro avg   0.77   0.72   0.73   576
weighted avg   0.77   0.78   0.76   576
</pre></td><td>
<pre>
              precision   recall f1-score   support

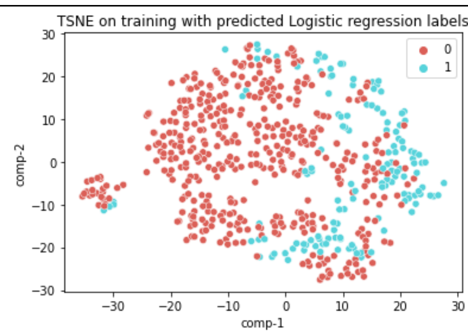
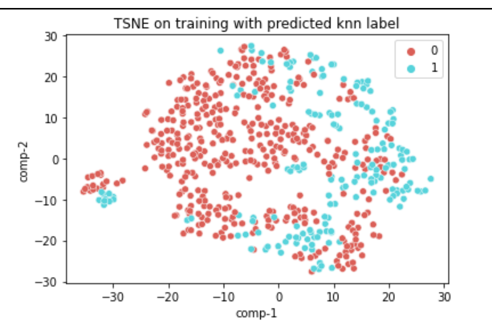
           0   0.84   0.91   0.87   373
           1   0.80   0.68   0.74   203

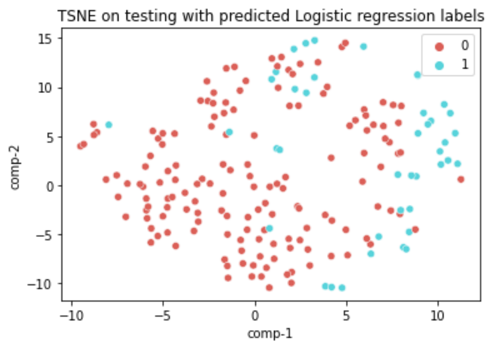
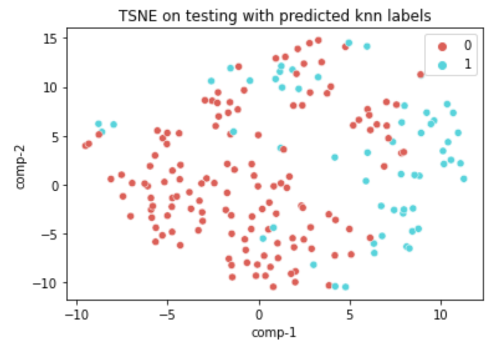
    accuracy   0.83   576
   macro avg   0.82   0.79   0.80   576
weighted avg   0.83   0.83   0.82   576
</pre></td></tr>
<tr><td>Classification report for testing</td><td>
<pre>
              precision   recall f1-score   support

           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
</pre></td><td>
<pre>
              precision   recall f1-score   support

           0   0.79   0.84   0.82   127
           1   0.65   0.57   0.61   65

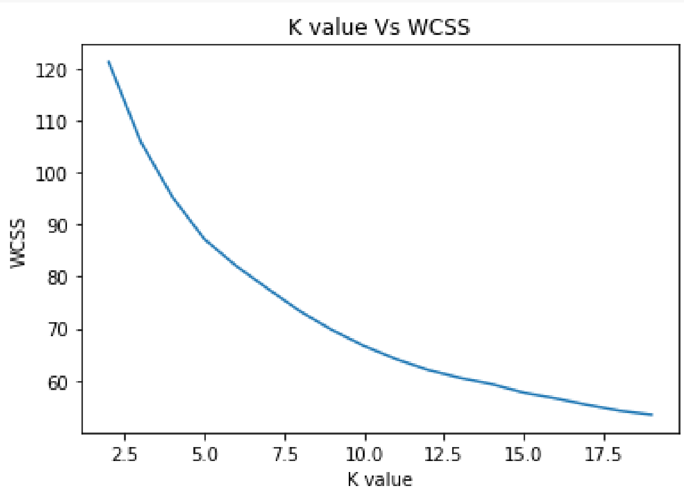
    accuracy   0.75   192
   macro avg   0.72   0.71   0.71   192
weighted avg   0.74   0.75   0.75   192
</pre></td></tr>
<tr><td>TSNE on training</td><td>
</td><td>
</td></tr>
</table>

<table>
<tr><td colspan="3">According to the accuracy, the KNN model can predict the label more efficient than the LR model.</td></tr>
<tr><td>TSNE on testing</td><td>
</td><td>
</td></tr>
<tr><td colspan="3">Accosrding to the testing accuracy, the LR model can predict the actual labels slightly than the KNN model.</td></tr>
</table>

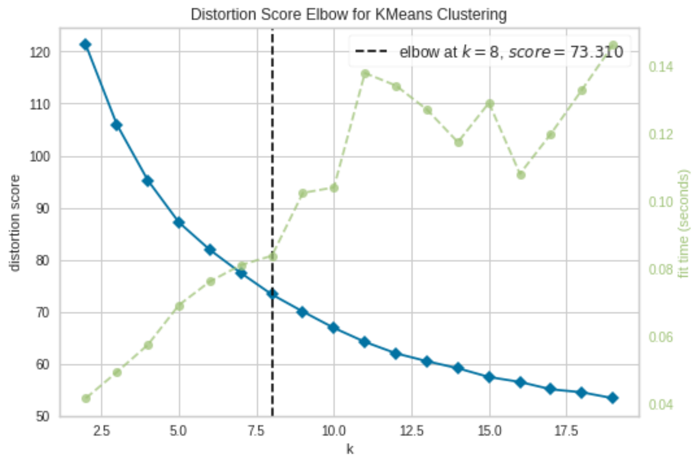
1- compare between the values of K and silhouette scores:

[31]

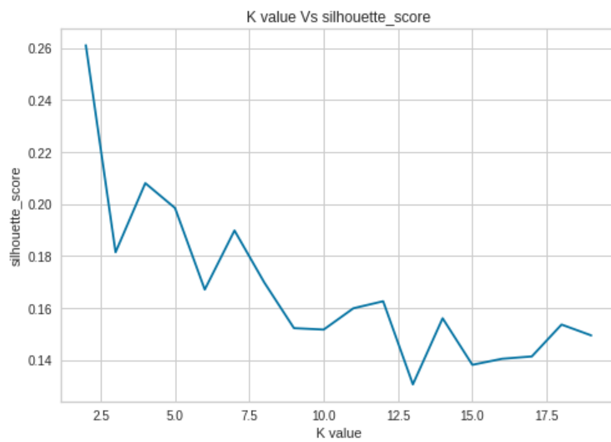
```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
wcss=[]
scores =[]
for i in range(2,20):
  kmean = KMeans(n_clusters=i)
  kmean.fit(X)
  wcss.append(kmean.inertia_)
  y_pred= kmean.predict(X)
  scores.append(silhouette_score(X,y_pred))
plt.xlabel('K value')
plt.ylabel('WCSS')
sns.lineplot(x=range(2,20),y=wcss,).set(title="K value Vs WCSS")
```



Visualize the k values with the distribution scores:

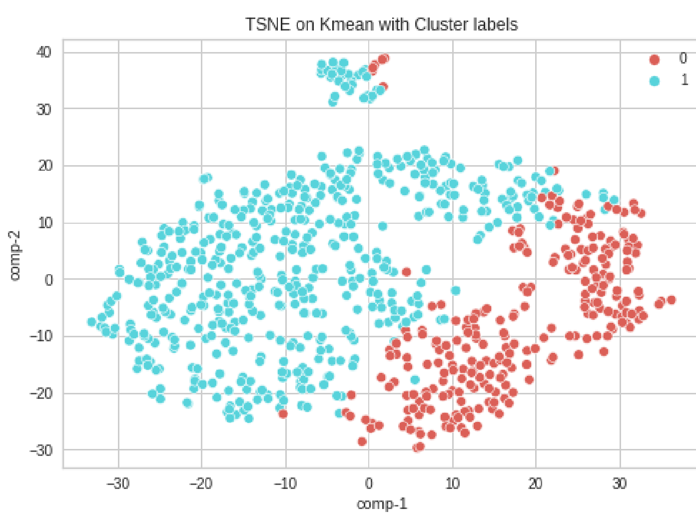


Visualize the k values with the silhouette scores:



So, the optimal number of k based on the silhouette score is when k = 2

TSNE on K-means with the cluster labels:



As we can see from the TSNE plot, the K-Means algorithm on can separate between the two cluster labels well.

3- Apply dimensionality reduction methods:

1- Apply the principle component analysis(PCA):

Principal component analysis, or PCA, is a statistical procedure that allows you to summarize the information content in large data tables by means of a smaller set of "summary indices" that can be more easily visualized and analyzed.

```
from sklearn.decomposition import PCA
scores_lr = []
scores_knn = []
for i in range(2,8):
  pca= PCA(n_components=i)
  x_train_pca = pca.fit_transform(x_train)
  x_test_pca = pca.transform(x_test)
  lr = LogisticRegression()
  lr.fit(x_train_pca,y_train)
  y_pred_lr = lr.predict(x_test_pca)
  scores_lr.append(accuracy_score(y_pred_lr,y_test))
  knn = KNeighborsClassifier()
  knn.fit(x_train_pca,y_train)
  y_pred_knn = knn.predict(x_test_pca)
  scores_knn.append(accuracy_score(y_pred_knn,y_test))
```

Scores of logistic regression with PCA	Scores of KNN with PCA
[0.7083333333333334, 0.7604166666666666, 0.7604166666666666, 0.7604166666666666, 0.7552083333333334, 0.7864583333333334]	[0.6510416666666666, 0.7291666666666666, 0.7604166666666666, 0.7083333333333334, 0.7083333333333334, 0.7604166666666666]

So,

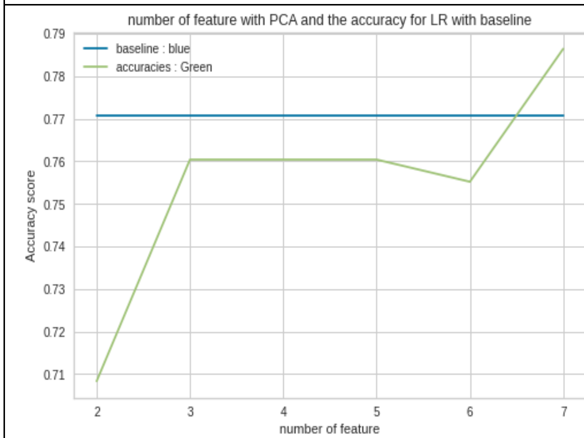
n =7 is the best number number of coponebt that achieve high accuarcy on LR and KNN

```
(2, 0.7083333333333334, 0.6510416666666666)
(3, 0.7604166666666666, 0.7291666666666666)
(4, 0.7604166666666666, 0.7604166666666666)
(5, 0.7604166666666666, 0.7083333333333334)
```

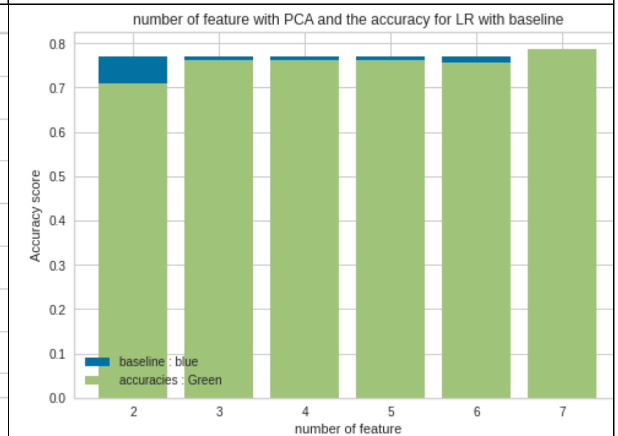
```
(6, 0.7552083333333334, 0.7083333333333334)
(7, 0.7864583333333334, 0.7604166666666666)
```

1- Plot the Number of Components-Accuracy graph with baseline performances for each classifier when n = 7

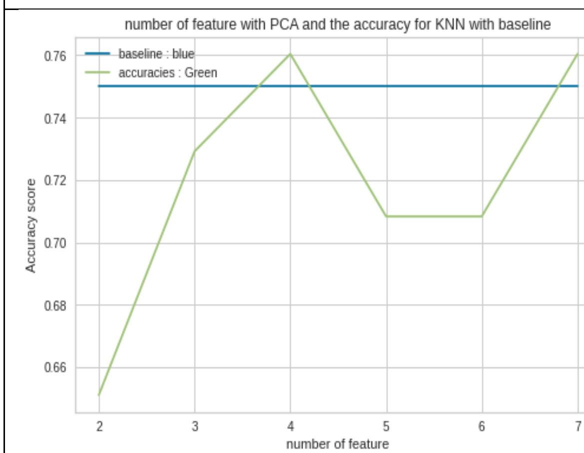
LR accuracies with PCA baseline



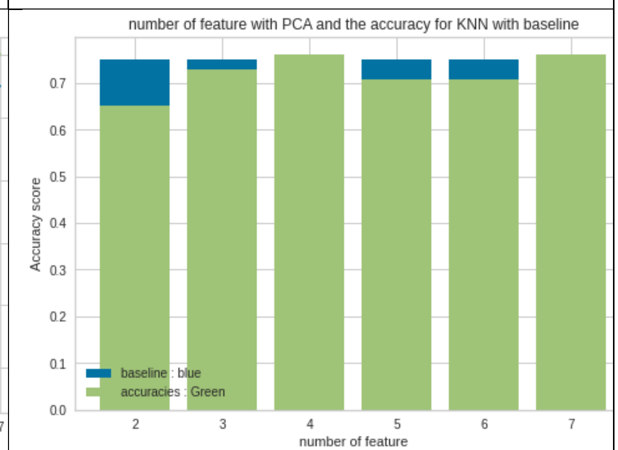
Bar chart of LR accuracies with PCA baseline



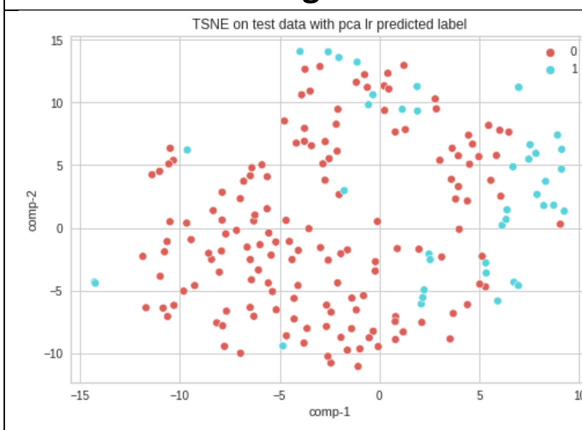
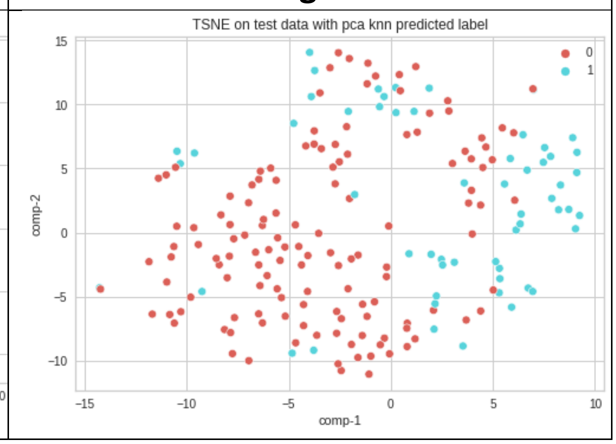
KNN accuracies with PCA baseline



Bar chart of KNN accuracies with PCA baseline



LR accuracy with PCA	KNN accuracy with PCA
0.7864583333333334	0.7604166666666666

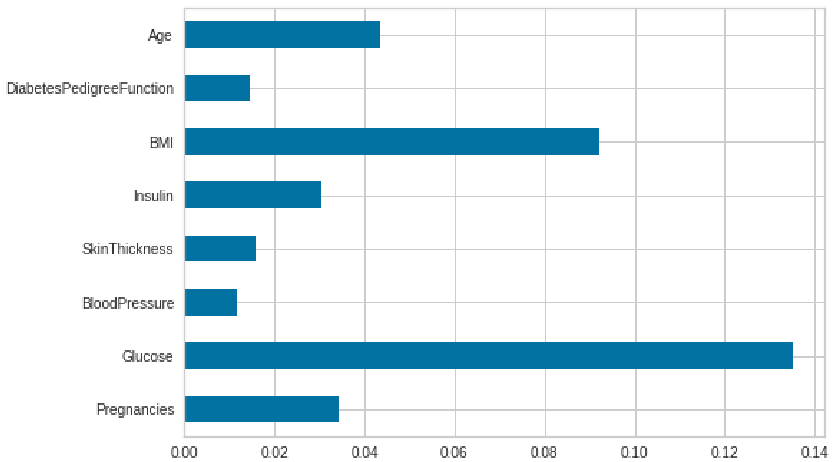
<table>
<tr><th>TSNE on training LR with PCA</th><th>TSNE on training KNN with PCA</th></tr>
<tr><td>
</td><td>
</td></tr>
<tr><td colspan="2">The LR and knn can’t separate between the two cluster labels well.</td></tr>
<tr><th>TSNE on testing LR with PCA</th><th>TSNE on testing KNN with PCA</th></tr>
<tr><td>
</td><td>
</td></tr>
</table>

4- Apply feature selection methods:

1- **Apply filter method:** Filter methods pick up the intrinsic properties of the features measured via univariate statistics instead of cross-validation performance. These methods are faster and less computationally expensive than wrapper methods. When dealing with high-dimensional data, it is computationally cheaper to use filter methods.

Apply information gain: Information gain calculates the reduction in entropy from the transformation of a dataset. It can be used for feature selection by evaluating the Information gain of each variable in the context of the target variable.

```python
# information Gain
from traitlets.traitlets import ForwardDeclaredInstance
from sklearn.feature_selection import mutual_info_classif
importance = mutual_info_classif(X,y)
feat_importance = pd.Series(importance,df.columns[0:len(df.columns)-1])
feat_importance.plot(kind='barh')
```



based on information gain we will select the highest four feature that has information gain.

Accuracy of LR with information gain	Accuracy of KNN with information gain
0.7864583333333334	0.765625

Classification report on training LR with information gain

	precision	recall	f1-score	support
0	0.77	0.90	0.83	373
1	0.73	0.50	0.59	203
accuracy			0.76	576
macro avg	0.75	0.70	0.71	576
weighted avg	0.75	0.76	0.75	576

Classification report on training KNN with information gain

	precision	recall	f1-score	support
0	0.85	0.85	0.85	373
1	0.72	0.71	0.72	203
accuracy			0.80	576
macro avg	0.79	0.78	0.78	576
weighted avg	0.80	0.80	0.80	576

Classification report on testing LR with information gain

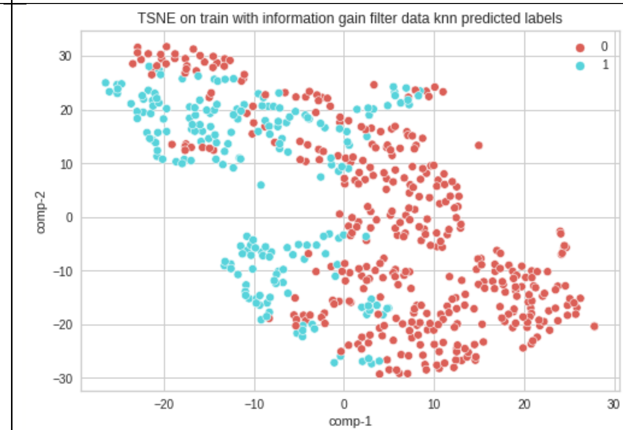
	precision	recall	f1-score	support
0	0.80	0.91	0.85	127
1	0.75	0.55	0.64	65
accuracy			0.79	192
macro avg	0.77	0.73	0.74	192
weighted avg	0.78	0.79	0.78	192

Classification report on testing KNN with information gain

	precision	recall	f1-score	support
0	0.80	0.86	0.83	127
1	0.68	0.58	0.63	65
accuracy			0.77	192
macro avg	0.74	0.72	0.73	192
weighted avg	0.76	0.77	0.76	192

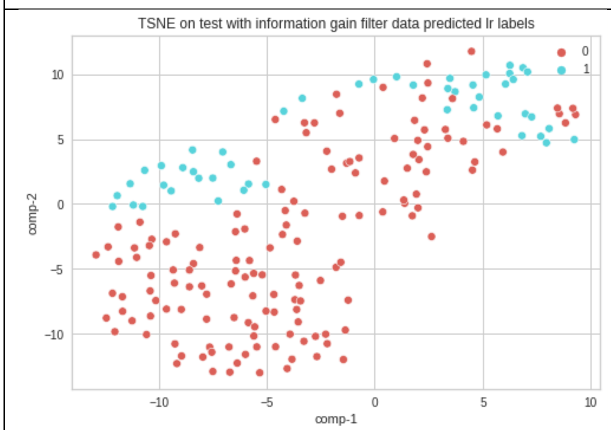
TSNE on train with information gain filter d ata lr predicted labels

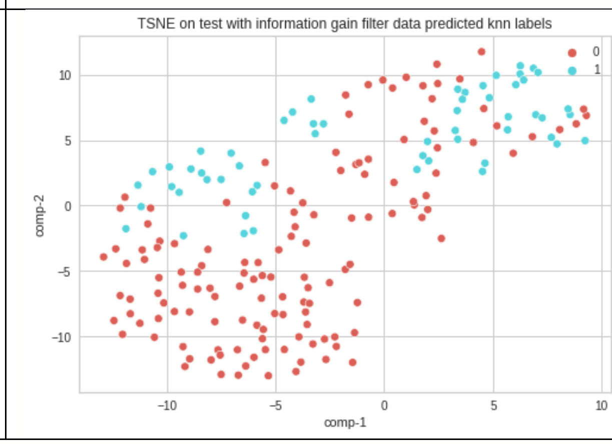
TSNE on train with information gain filter dat a KNN predicted labels



TSNE on test with information gain filter da ta predicted lr labels

TSNE on test with information gain filter data predicted knn labels

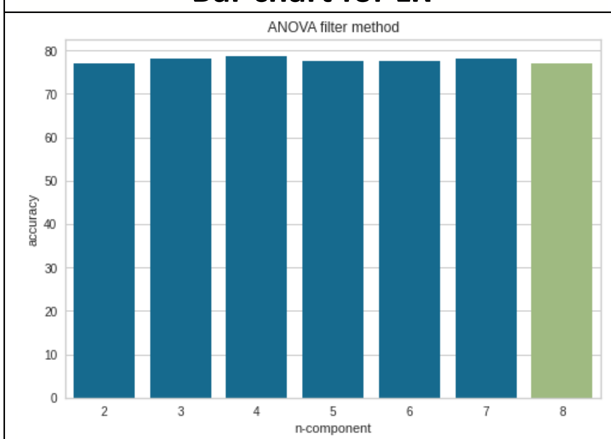


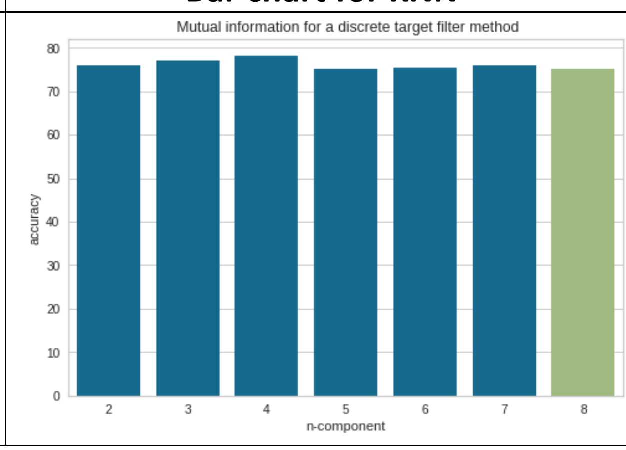


On the testing after applying the information gain, the KNN and LR accuracies are closed to each other.

Bar chart for LR

Bar chart for KNN





According to the accuracies, the LR model is better than KNN model with information gain.

Apply variance threshold:

The variance threshold is a simple baseline approach to feature selection. It removes all features which variance doesn't meet some threshold. By default, it removes all zero-variance features, i.e., features that have the same value in all samples. We assume that features with a higher variance may contain more useful information, but note that we are not taking the relationship between feature variables or feature and target variables into account, which is one of the drawbacks of filter methods.

```python
# Variance threshold
from sklearn.feature_selection import VarianceThreshold
v = VarianceThreshold(threshold=0.02)
v.fit(X)
v.get_support()
```

```
array([ True,  True,  True,  True, False, False, False,  True])
```

Accuracy of LR with variance threshold	Accuracy of KNN with variance threshold
0.7708333333333334	0.78125

Classification report for training LR

	precision	recall	f1-score	support
0	0.75	0.90	0.82	373
1	0.71	0.46	0.56	203
accuracy			0.74	576
macro avg	0.73	0.68	0.69	576
weighted avg	0.74	0.74	0.73	576

Classification report for training KNN

	precision	recall	f1-score	support
0	0.82	0.88	0.85	373
1	0.75	0.66	0.70	203
accuracy			0.80	576
macro avg	0.79	0.77	0.77	576
weighted avg	0.80	0.80	0.80	576

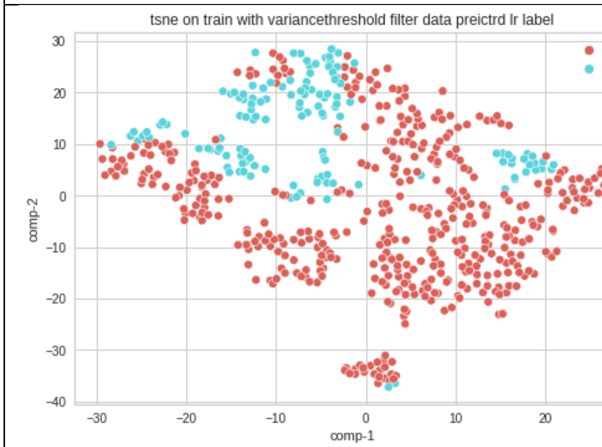
Classification report for testing LR

	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

Classification report for testing KNN

	precision	recall	f1-score	support
0	0.83	0.84	0.84	127
1	0.68	0.66	0.67	65
accuracy			0.78	192
macro avg	0.76	0.75	0.75	192
weighted avg	0.78	0.78	0.78	192

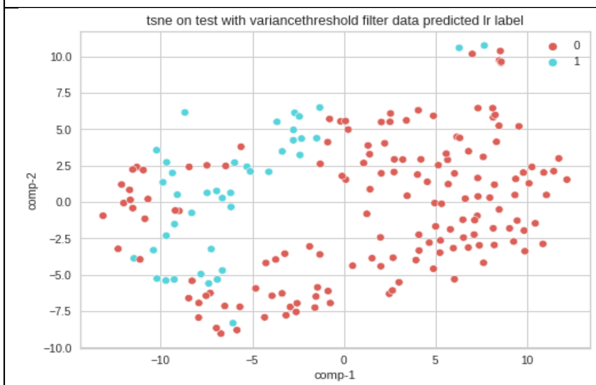
TSNE on train with variancethreshold filter data preictrd lr labels



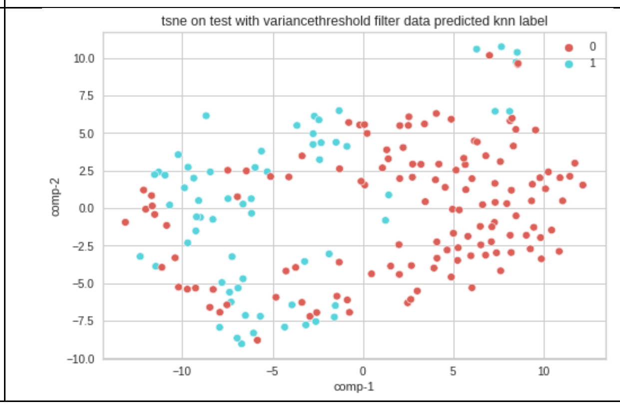
TSNE on train with variancethreshold filter dat a predicted knn labels

As we can see, the KNN model can separate the labels well than the LR model.

TSNE on test with variancethreshold filter data predicted lr labels

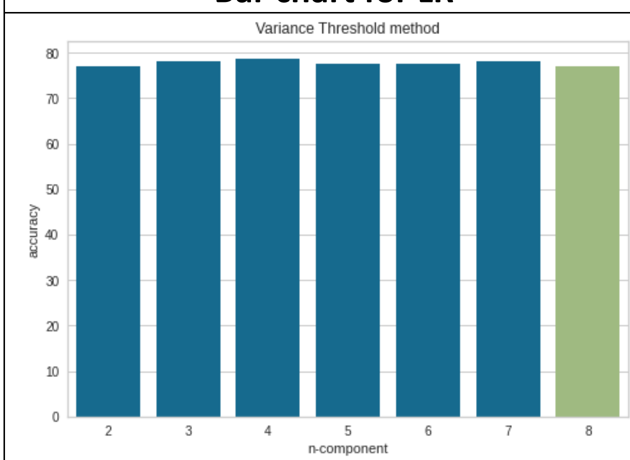


TSNE on test with variancethreshold filter dat a predicted knn labels

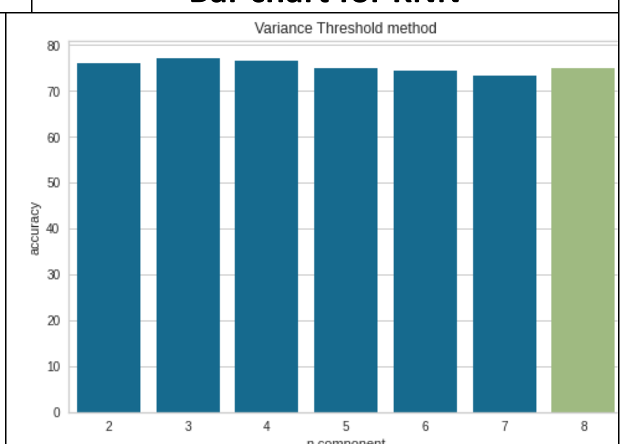


When testing the two models, we can see that they are too close from each other.

Bar chart for LR



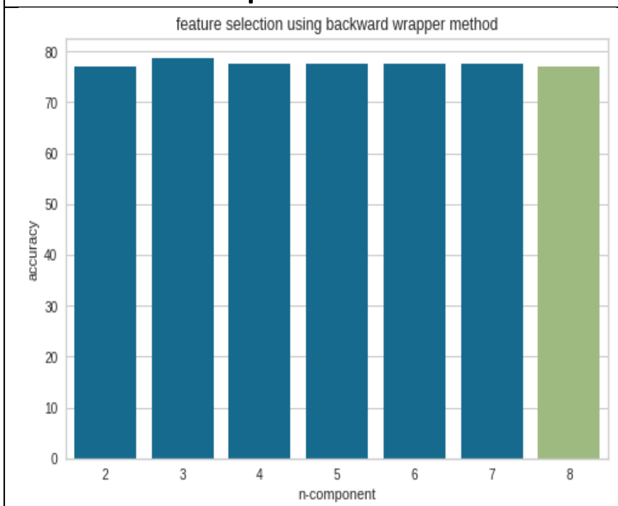
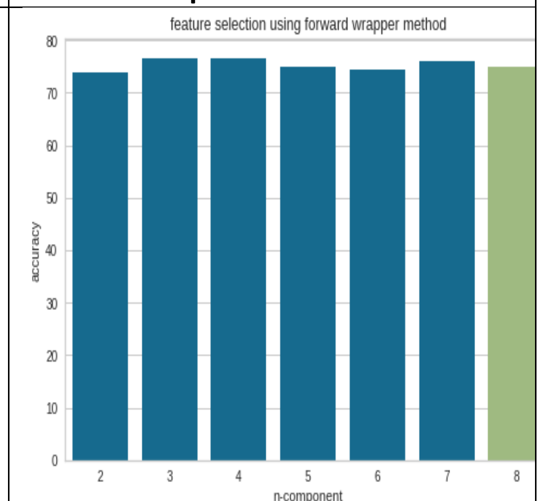
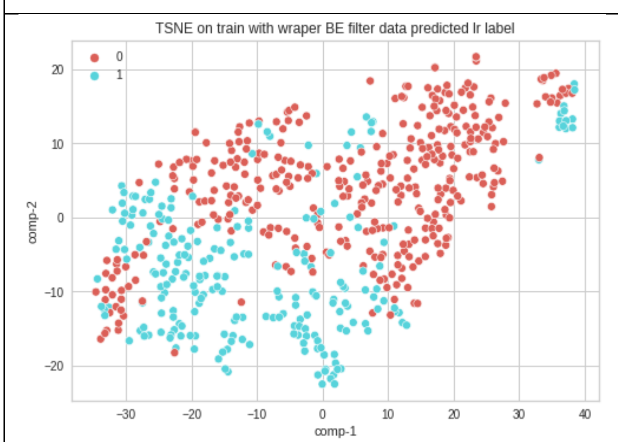
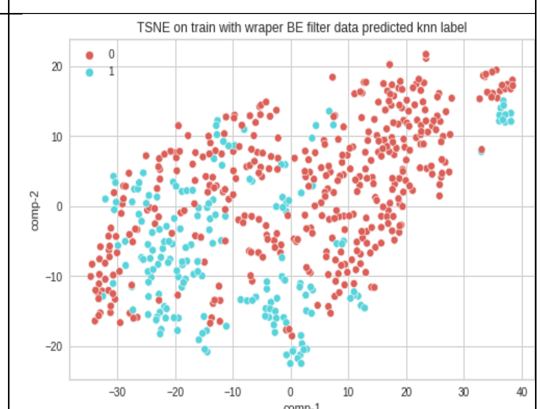
Bar chart for KNN

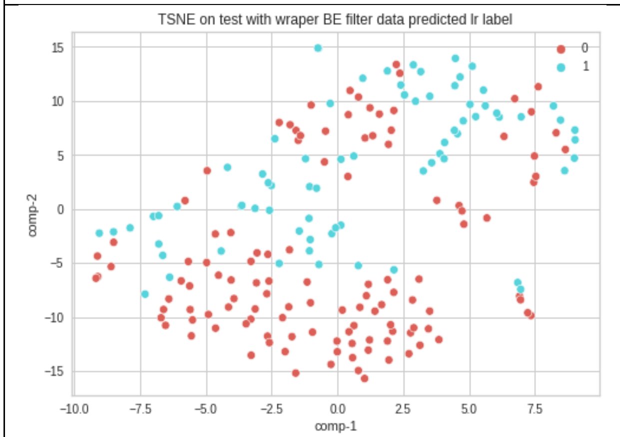
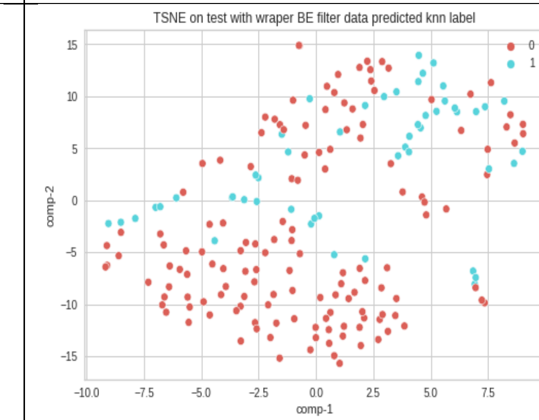


Apply wrapper methods:

Wrappers require some method to search the space of all possible subsets of features, assessing their quality by learning and evaluating a classifier with that feature subset. The feature selection process is based on a specific machine learning algorithm that we are trying to fit on a given dataset. It follows a greedy search approach by evaluating all the possible combinations of features against the evaluation criterion. The wrapper methods usually result in better predictive accuracy than filter methods.

1- **Apply backward feature selection:**

Accuracy score for LR	**Accuracy score for KNN**
0.7825	0.76
Baseline performance for LR	**Baseline performance for KNN**
	
TSNE on train with wrapper BE filter data pr edicted lr label	**TSNE on train with wrapper BE filter data predicted knn label**
	

<table>
<tr><th>TSNE on test with wrapper BE filter data pre dicted lr label</th><th>TSNE on test with wrapper BE filter d ata predicted knn label</th></tr>
<tr><td>
</td><td>
</td></tr>
<tr><td colspan="2">The LR model is better than KNN model on the Backward selection testing</td></tr>
</table>

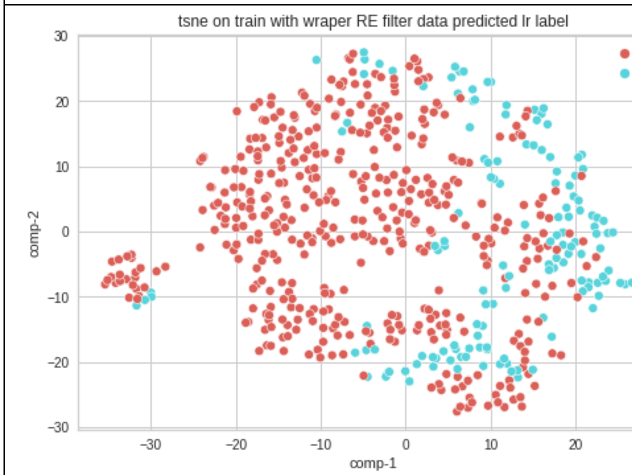
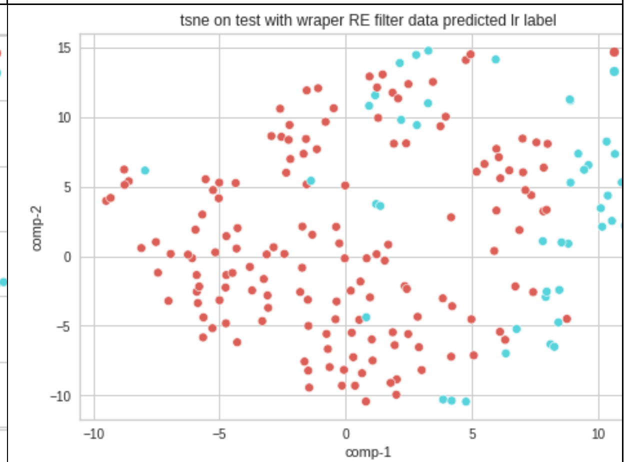
Apply Recursive feature elimination on LR:

the goal of recursive feature elimination (RFE) is to select features by recursively considering smaller and smaller sets of features. First, the estimator is trained on the initial set of features and the importance of each feature is obtained either through a coef_ attribute or through a feature_importances_ attribute.

Accuracy is: 0.7760416666666666

```
              precision    recall  f1-score   support

           0       0.78      0.91      0.84       373
           1       0.76      0.53      0.62       203

    accuracy                           0.78       576
   macro avg       0.77      0.72      0.73       576
weighted avg       0.77      0.78      0.76       576

              precision    recall  f1-score   support

           0       0.78      0.92      0.84       127
           1       0.76      0.49      0.60        65

    accuracy                           0.78       192
   macro avg       0.77      0.71      0.72       192
weighted avg       0.77      0.78      0.76       192
```

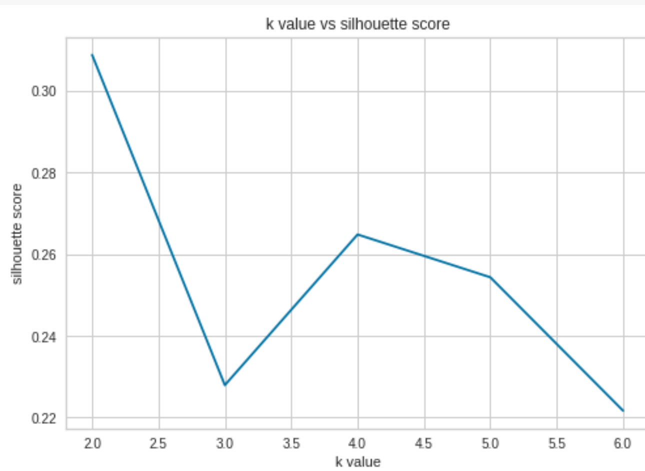
can`t apply RFE with KNN because doesn`t have a feature importance attribute

TSNE on train LR	TSNE on test LR
	

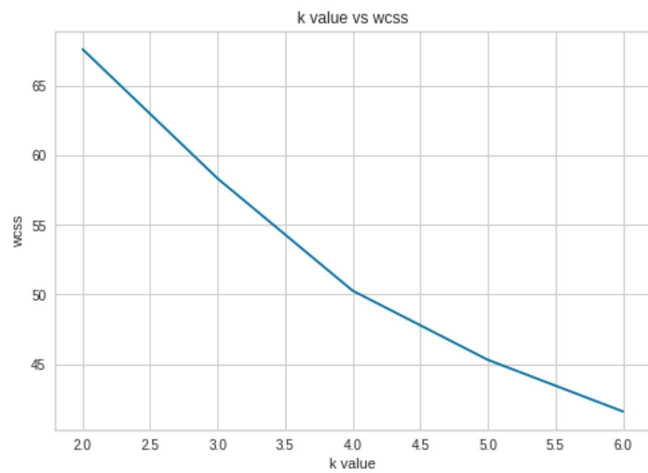
5- Choose the best number of cluster for k-means clustering algorithm on the processed data, using the best features:

Silhouette score vs number of clusters:

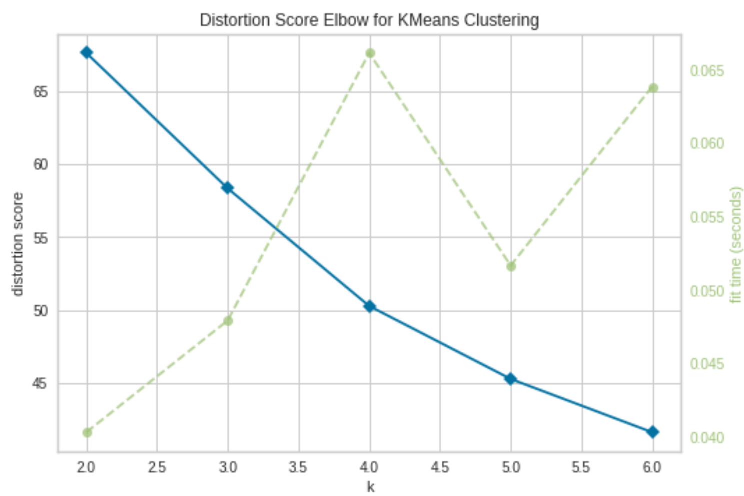
```
[23] wcss=[]
    scores =[]
    for i in range(2,7):
      kmean = KMeans(n_clusters=i)
      y_pred = kmean.fit_predict(x_train_b_knn)
      wcss.append(kmean.inertia_)
      scores.append(silhouette_score(x_train_b_knn,y_pred))


    plt.plot(range(2,7),scores)
    plt.title('k value vs silhouette score')
    plt.xlabel('k value')
    plt.ylabel('silhouette score')
```



k value vs wcss:

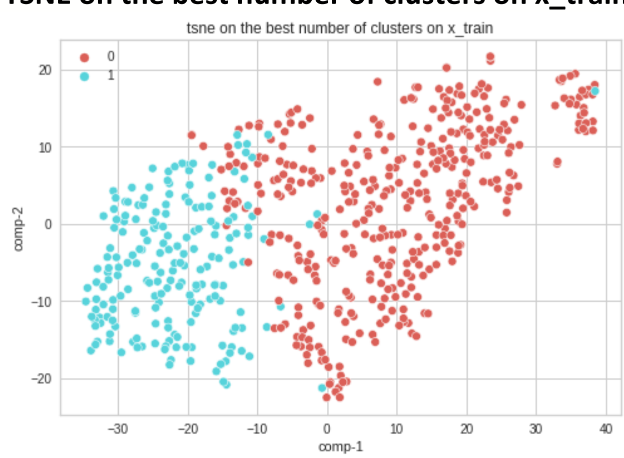


Distribution score elbow for k-means clustering:



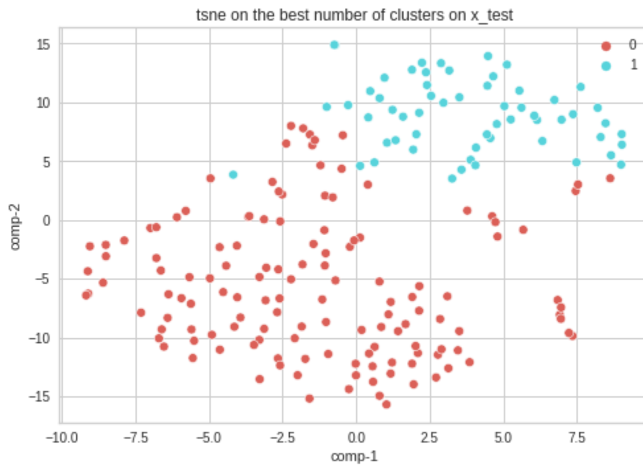
So the best number of k is k = 2

TSNE on the best number of clusters on x_train



TSNE on the best number of clusters on x_test

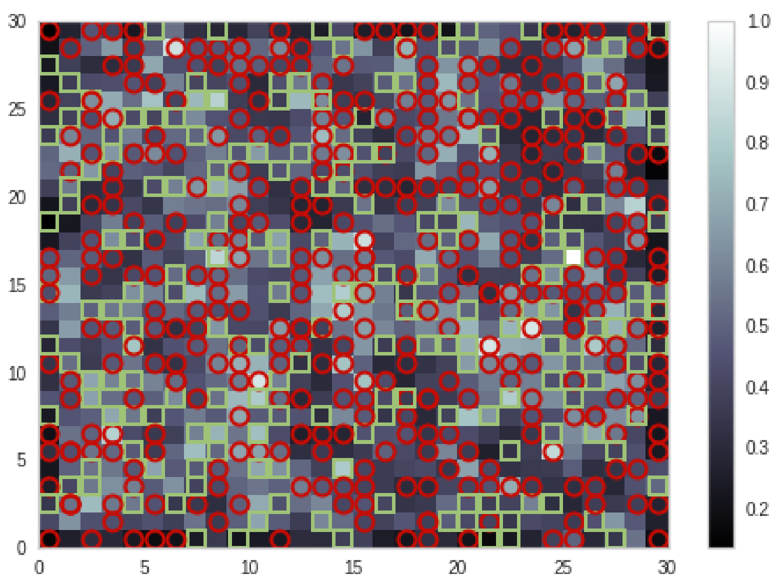
according to the TSNE plot, the algorithm can separate between the clusters well.



6- Train the SOM algorithm:

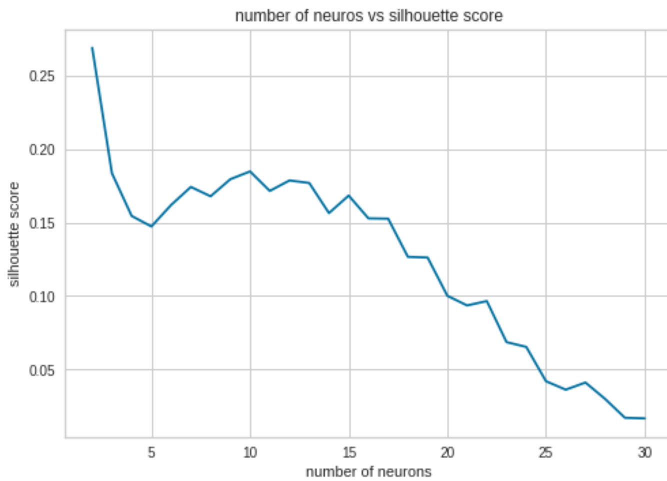
An SOM is mainly used for data visualization and provides a quick visual summary of the training instances. In a 2D rectangular grid, each cell is represented by a weight vector. For a trained SOM, each cell weight represents a summary of a few training examples. Cells in the close vicinity of each other have similar weights, and like examples can be mapped to cells in a small neighborhood of each other.

The final result on training SOM model, the 30*30 neuron:



So the best number of neuron is 2 based on the silhouette score

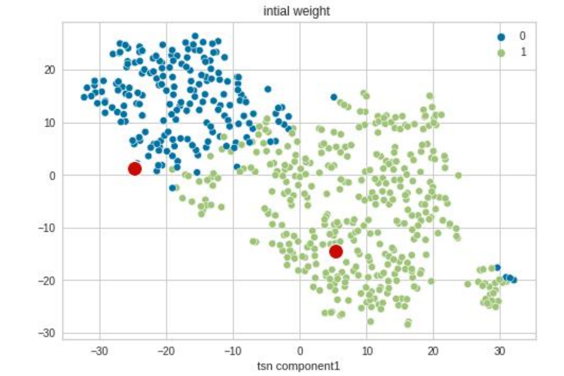
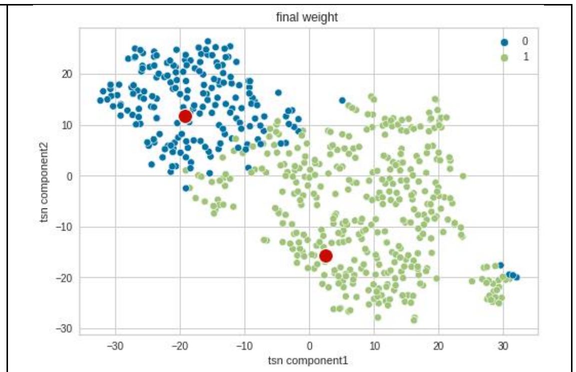
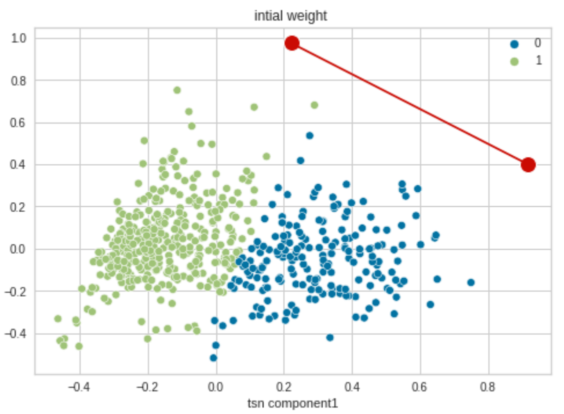
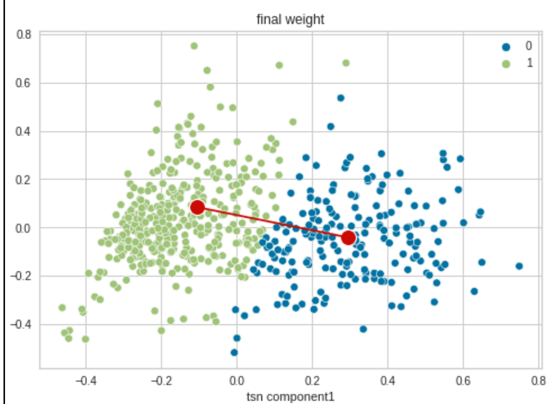
```python
# the best number of neuron is 2 based on the highest silhouette score
plt.plot(range(2,31),scores)
plt.xlabel('number of neurons')
plt.ylabel('silhouette score')
plt.title('number of neuros vs silhouette score')
```



Apply PCA on the SOM Model:

```python
138] som = MiniSom(2,1, 6, sigma=0.3, learning_rate=0.5,random_seed=0) # initialization of 2*1 SOM
     intial_weight = som.get_weights().copy()
     som.train_batch(np.array(x_train_b_knn), 1000) # trains the SOM with 1000 iterations
     final_weight = som.get_weights()
     w = np.array([som.winner(x) for x in np.array(x_train_b_knn)]).T
     index = np.ravel_multi_index(w,(2,1))
```

intial_weighT	**final_weight**
array([[[0.16419828, 0.72385141, 0.34567421, 0.15097751, -0.25680907, 0.49075687]], [[-0.09085602, 0.57031478, 0.67496669, -0.1696774 , 0.4246722 , 0.04206313]]])	array([[[0.47710267, 0.71042386, 0.59288791, 0.08145516, 0.46140247, 0.36316582]], [[0.13668112, 0.6537922 , 0.54311031, 0.13120688, 0.47819656, 0.12776533]]])
TSNE plot for all data with two components and intial weight for the neuron	**TSNE plot for all data with two components and final weight for the neuron**

<table>
<tr><td>
</td><td>
</td></tr>
<tr><td>TSNE plot for all data with two components and intial weight for the neuron with PCA on 2 components</td><td>TSNE plot for all data with two components and final weight for the neuron with PCA on 2 components</td></tr>
<tr><td>
</td><td>
</td></tr>
</table>

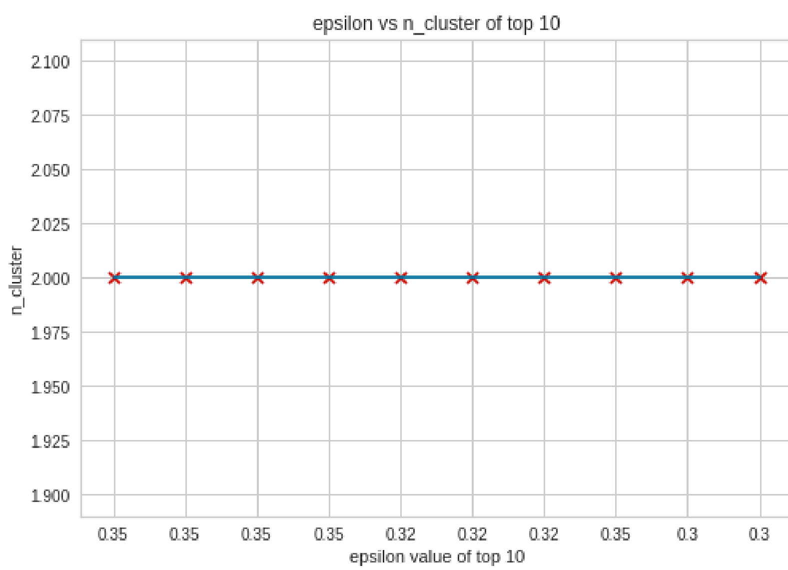
Part 7: Tune the epsilon (0.3-0.7) and minpoints (2-15) values to obtain the same number of clusters in Q6 by using DBSCAN:

The ***DBSCAN algorithm*** is based on this intuitive notion of “clusters” and “noise”. The key idea is that for each point of a cluster, the neighborhood of a given radius has to contain at least a minimum number of points.

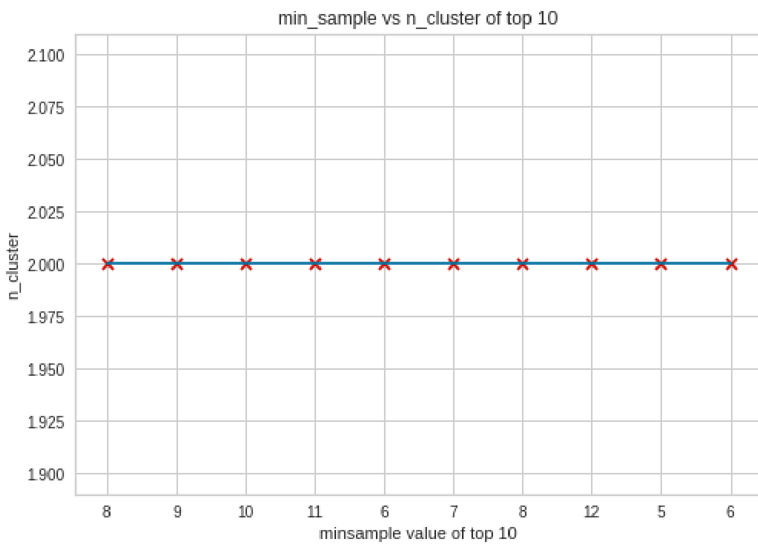
Top 10 clusters:

	eps	minsample	silhouette	cluster
34	0.35	8	0.264563	2
35	0.35	9	0.263837	2
36	0.35	10	0.259480	2
37	0.35	11	0.258518	2
18	0.32	6	0.254182	2
19	0.32	7	0.250709	2
20	0.32	8	0.250598	2
38	0.35	12	0.248660	2
3	0.30	5	0.246779	2
4	0.30	6	0.244219	2

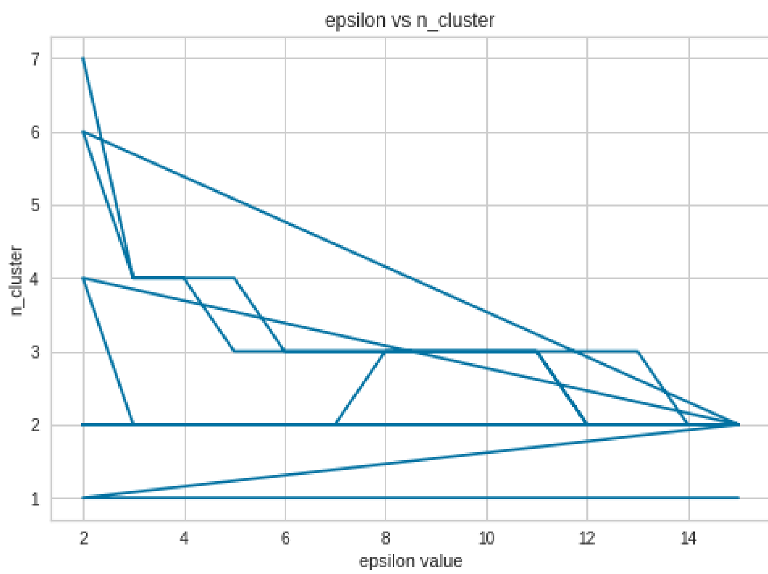
Plot the number of neurons vs number of clusters(top 10):



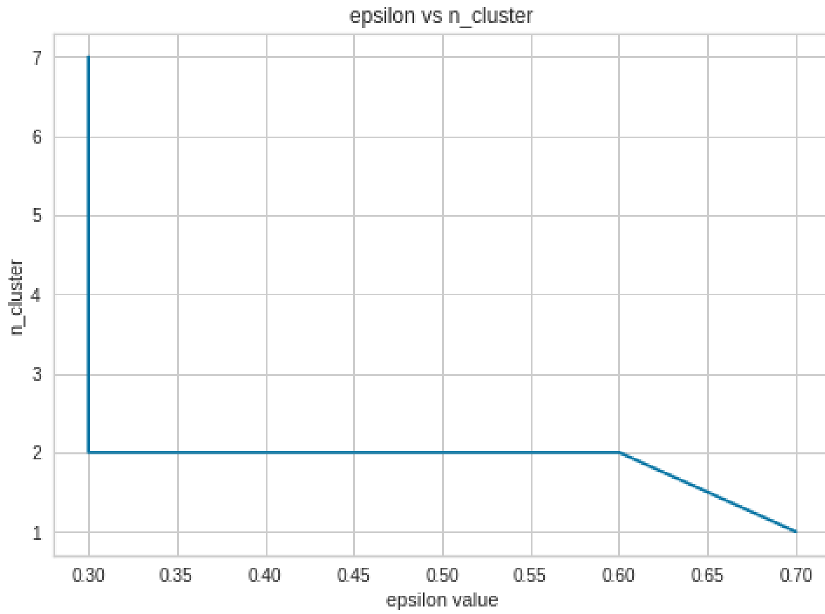
plotting the minpoints vs number of clusters(top 10):

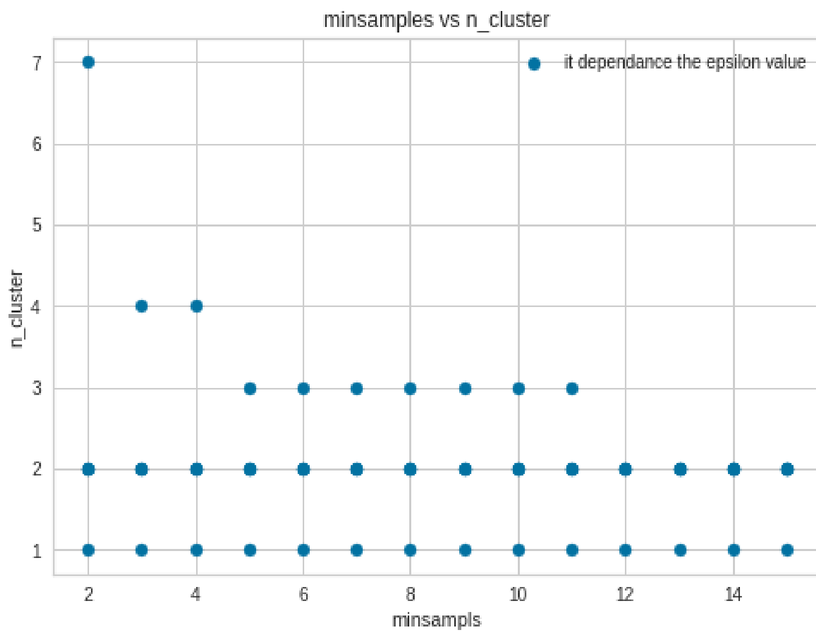


Number of epsilon vs number of clusters:



The best epsilon and min samples is 0.5 as epsilon and samples[2:15] this values achieve the h igh siluoette score on the best feature

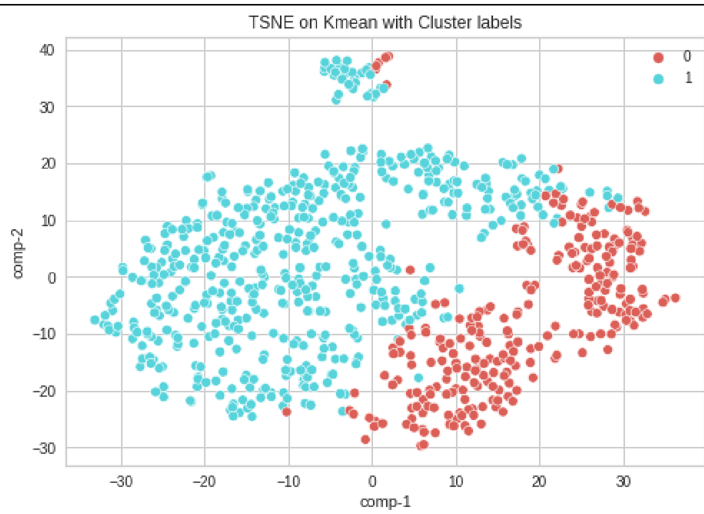

epsilon vs n_cluster
n_cluster
epsilon value
0.30
0.35
0.40
0.45
0.50
0.55
0.60
0.65
0.70
1
2
3
4
5
6
7


minsamples vs n_cluster
it dependance the epsilon value
n_cluster
minsampls
2
4
6
8
10
12
14
1
2
3
4
5
6
7

Conclusion:

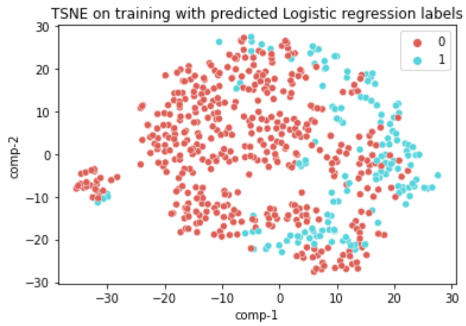
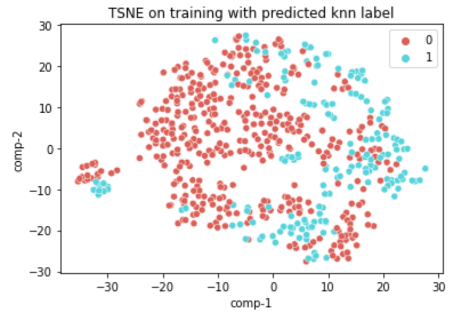
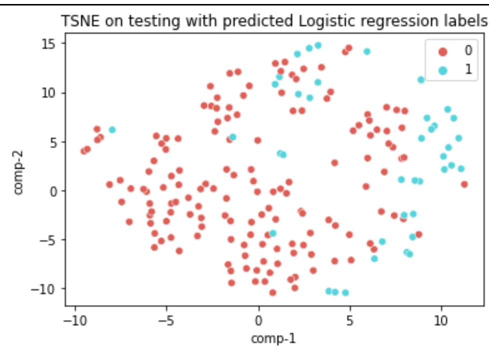
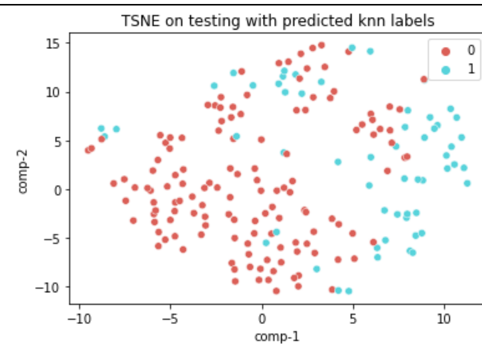
In this assignment we learned how to apply the logistic regression model and the KNN model, applying the different feature selection methods like the filter selection and wrapper methods, applying the principle component analysis, and applying SOM algorithm and DBSCAN algorithm. According to our results of K-means algorithm we can say that:

TSNE plot for K-Means algorithm with the optimal k value based on the silhouette score before dimensionality reduction is 0.26



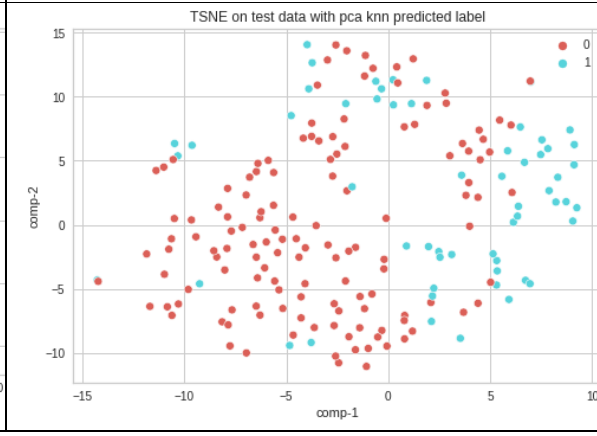
The **silhouette score is increased by 0.31 after dimensionality reduction and feature selection, and the highest silhouette score means that we can separate between the clusters and the distance between them is increased, and we can see the difference between the two plots.**

From question 1, 3 and 4 we can compare between the results as:

<table>
<tr><th colspan="3">Question 1</th></tr>
<tr><td>TSNE on training</td><td>
</td><td>
</td></tr>
<tr><td>TSNE on testing</td><td>
</td><td>
</td></tr>
<tr><td colspan="3">Accosrding to the testing accuracy, the LR model can predict the actual labels slightly than the KNN model.
Accuracy of LR is 77%, and KNN is 75%</td></tr>
</table>

After applying PCA

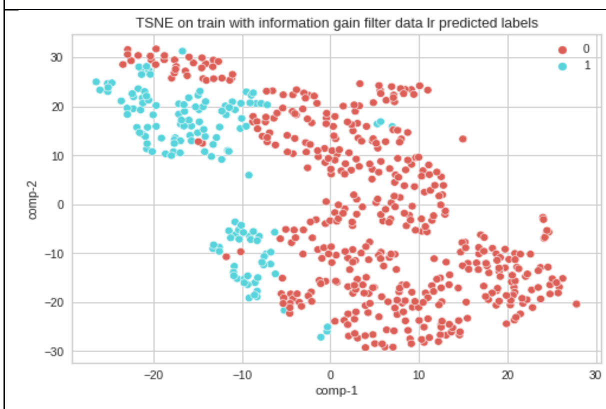
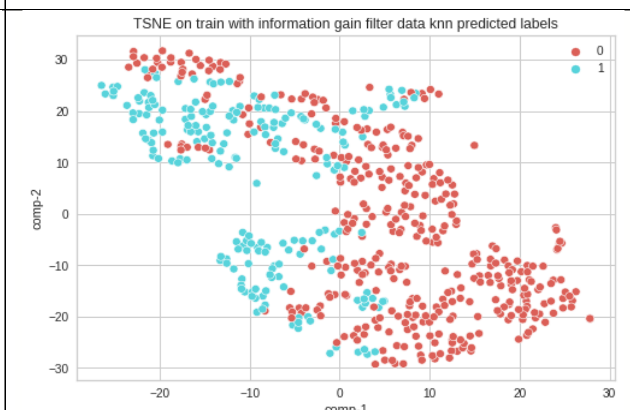
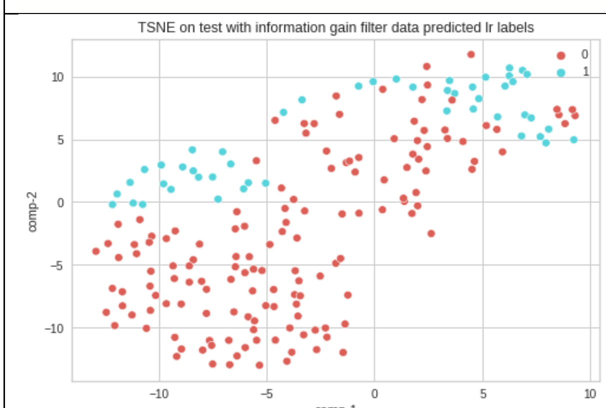
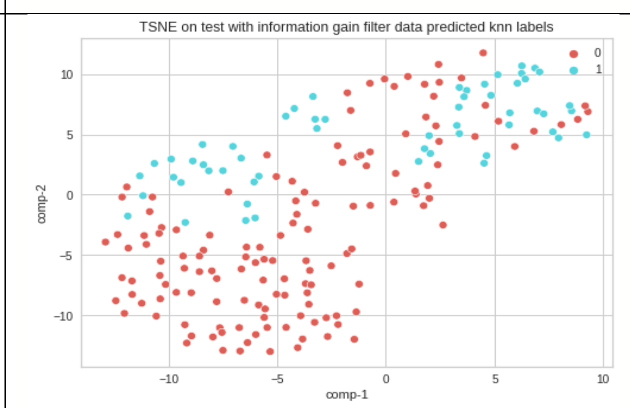
<table>
<tr><th>TSNE on training LR with PCA</th><th>TSNE on training KNN with PCA</th></tr>
<tr><td>
</td><td>
</td></tr>
<tr><td colspan="2">The LR and knn can’t separate between the two cluster labels well.</td></tr>
</table>

TSNE on testing LR with PCA	TSNE on testing KNN with PCA
	

accuracy for LR with PCA: 78%

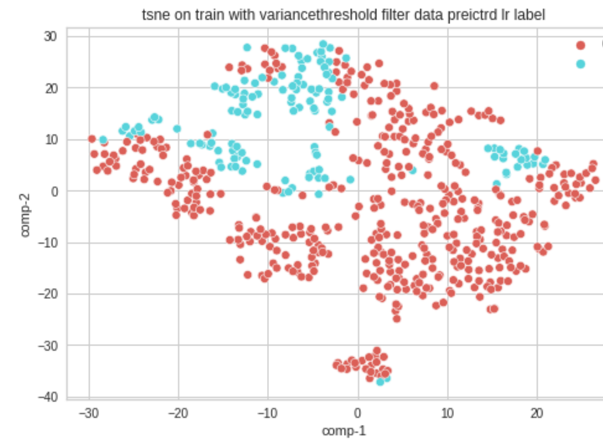
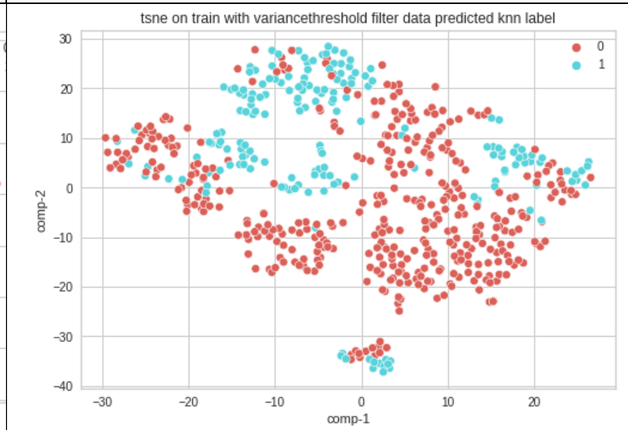
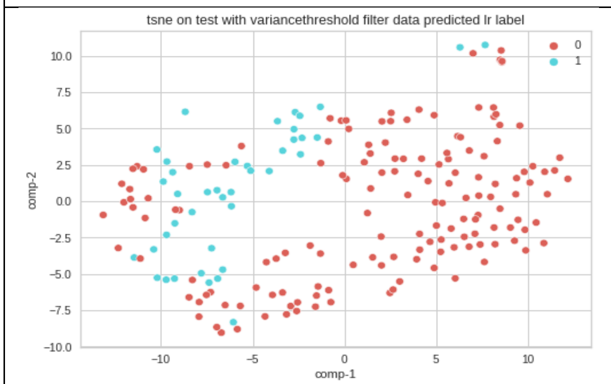
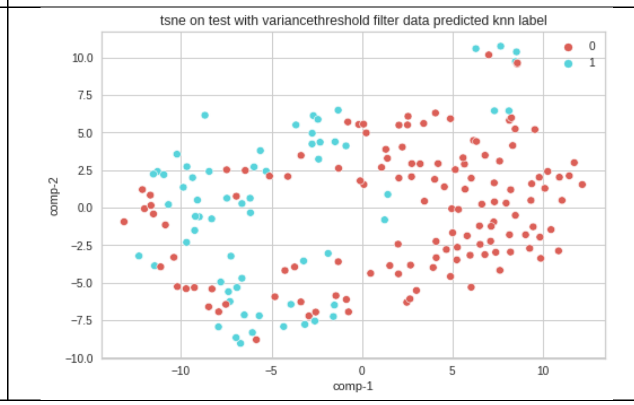
accuracy for KNN with PCA: 76%

after applying the information gain:

TSNE on train with information gain filter d ata lr predicted labels	TSNE on train with information gain filter dat a KNN predicted labels
	
TSNE on test with information gain filter da ta predicted lr labels	**TSNE on test with information gain filter data predicted knn labels**
	

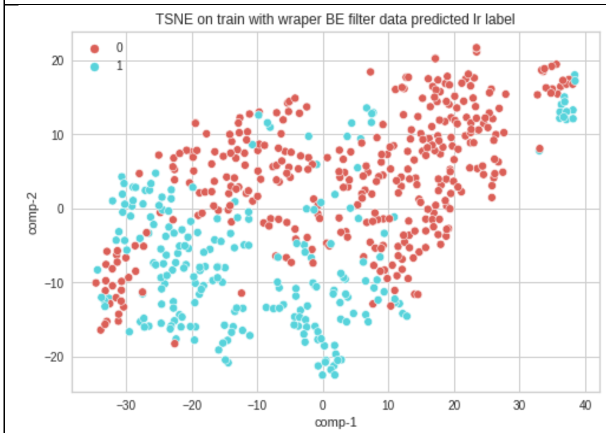
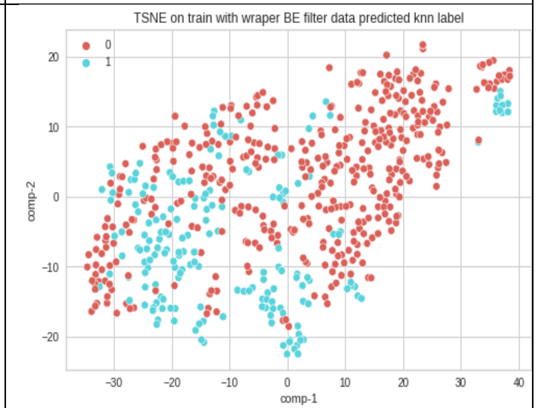
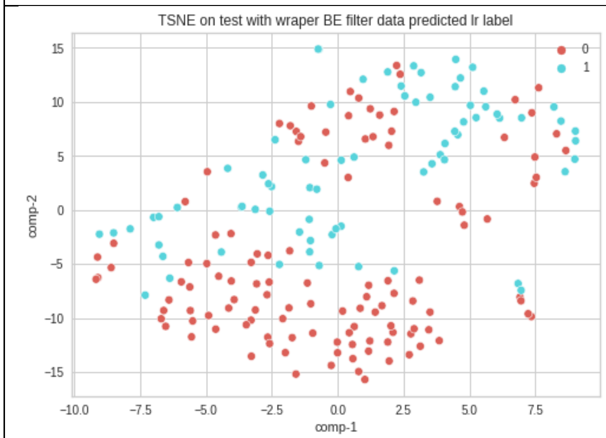
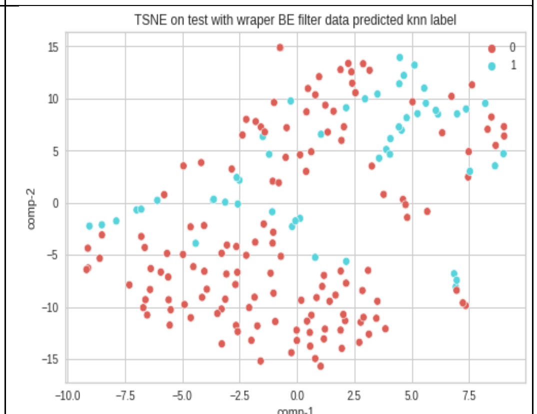
Accuracy of LR is 79%
Accuracy of KNN is 76%

After apply variance threshold:

<table>
<tr><td>TSNE on train with variancethreshold filter data preictrd lr labels</td><td>TSNE on train with variancethreshold filter da ta predicted knn labels</td></tr>
<tr><td>
</td><td>
</td></tr>
<tr><td colspan="2">As we can see, the KNN model can separate the labels well than the LR model.</td></tr>
<tr><td>TSNE on test with variancethreshold filter d ata predicted lr labels</td><td>TSNE on test with variancethreshold filter dat a predicted knn labels</td></tr>
<tr><td>
</td><td>
</td></tr>
<tr><td colspan="2">When testing the two models, we can see that they are too close from each other.</td></tr>
</table>

Accuracy of LR is 77%
Accuracy of KNN is 78%

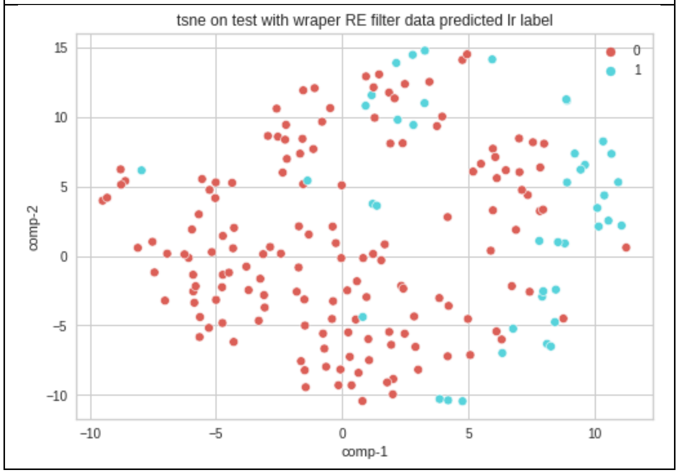
After applying wrapper method:

<table>
<tr><td>TSNE on train with wrapper BE filter data predicted lr label</td><td>TSNE on train with wrapper BE filter data predicted knn label</td></tr>
<tr><td>
</td><td>
</td></tr>
<tr><td>TSNE on test with wrapper BE filter data predicted lr label</td><td>TSNE on test with wrapper BE filter data predicted knn label</td></tr>
<tr><td>
</td><td>
</td></tr>
<tr><td colspan="2">The LR model is better than KNN model on the Backward selection testing</td></tr>
</table>

Accuracy of LR is 78%
Accuracy of KNN is 76%

Recursive elimination selection:

TSNE on test LR



Accuracy of LR is 78%

So, after comparing between the TSNE plots for all models, we can say that the LR model accuracies are closed to each other from 76: 79%, and the KNN model accuracies are between 75: 78%

References:

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

https://www.datacamp.com/tutorial/k-nearest-neighbor-classification-scikit-learn

https://www.kdnuggets.com/2020/04/dbscan-clustering-algorithm-machine-learning.html

https://stackabuse.com/self-organizing-maps-theory-and-implementation-/in-python-with-numpy

colab code link: https://colab.research.google.com/drive/18YEqf-coNeP721YTjcpqt9DSHEmu4gHJ?usp=sharing