(Group_10) Assignment_3

Prepared for

Prof: Murat SIMSEK
TA: Samhita Kuili, Mahrokh Hezaveh

Faculty of Engineering, University of u Ottawa

ELG5255: Applied Machine Learning

Prepared by

Yousef Abd Al Haleem Ahmed Shindy

Esraa Mahmoud Said Ahmed

Esmael Alahmady Ebrahim Ezz

Part 1: Calculations:

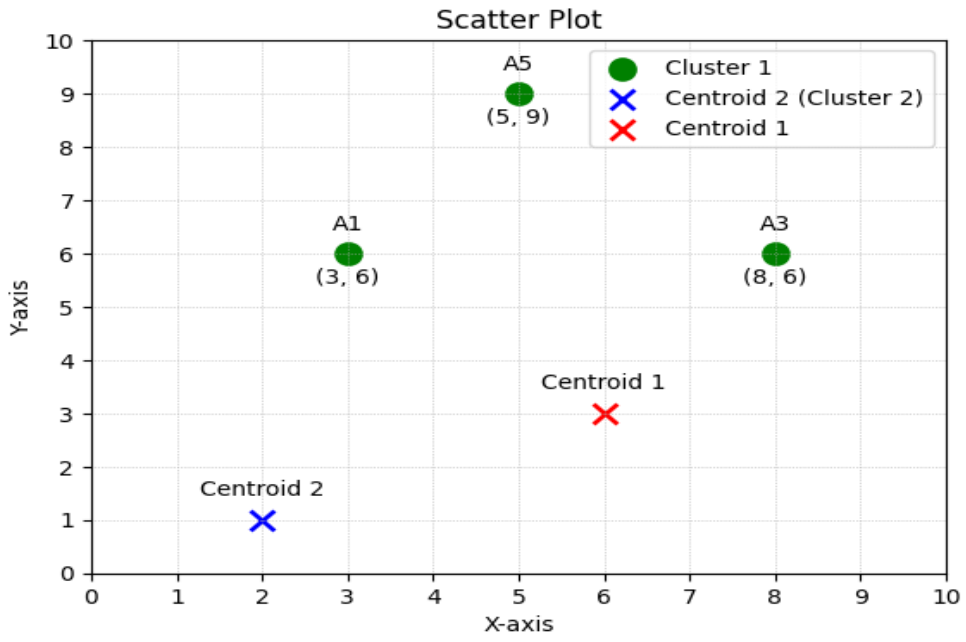
a) We compute the Euclidean distance between each data point and the initial centroids and assign each data point to the cluster with the shortest distance.

- **cluster the 5 points.**

Data Points			Euclidean distance $\sqrt{(X_1 - y_1)^2 + (x_2 - y_2)^2}$		Cluster
	X	Y	A2 = (6, 3)	A4 = (2, 1)	1
A1	3	6	$\sqrt{(3-6)^2+(6-3)^2} = 3\sqrt{2}$ = **4.24**	$\sqrt{(3-2)^2+(6-1)^2} = \sqrt{26} = 5.09$	1
A2	6	3	$\sqrt{(6-6)^2+(3-3)^2} = 0$	$\sqrt{(6-2)^2+(3-1)^2} = 2\sqrt{5}$ = 4.47	1
A3	8	6	$\sqrt{(8-6)^2+(6-3)^2} = \sqrt{13}$ = **3.6**	$\sqrt{(8-2)^2+(6-1)^2} = \sqrt{61}$ = 7.8	1
A4	2	1	$\sqrt{(2-6)^2+(1-3)^2} = 2\sqrt{5}$ = **4.47**	$\sqrt{(2-2)^2+(1-1)^2} = 0$	2
A5	5	9	$\sqrt{(5-6)^2+(9-3)^2} = \sqrt{37}$ = **6.08**	$\sqrt{(5-2)^2+(9-1)^2} = \sqrt{73}$ = 8.54	1

b) We visualize the outcome. Next, we update the centroids based on the newly assigned clusters and illustrate the new positions of the centroids.

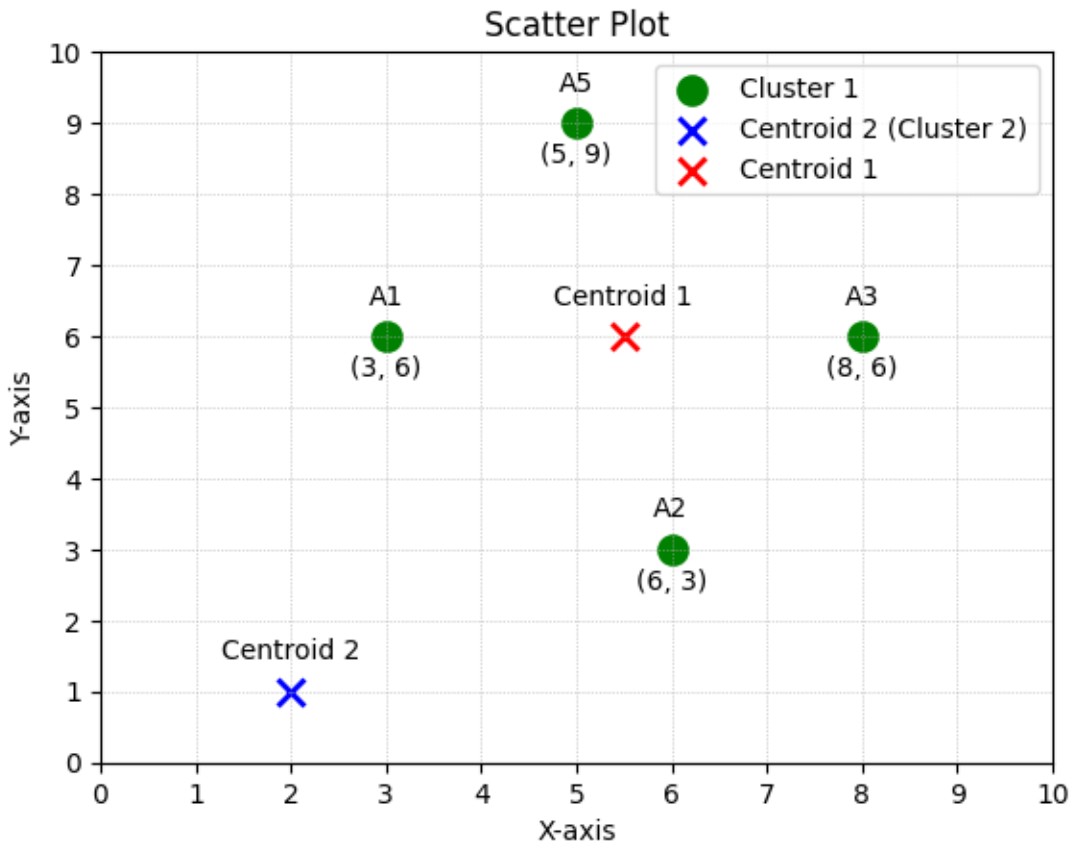
- **Plot cluster points with initial centroids.**



➢ **Assign new centroids.**

Data Points		
	X	**Y**
A1	3	6
A2	6	3
A3	8	6
A5	5	9
SUM	22	24
AVERAGE	5.5	6
New Centroids	**(5.5, 6)**	

➢ **Plot cluster points with new centroids.**



c) To cluster the points, we first calculate a dissimilarity matrix that measures the distances between points. Then, we compute two values: a) the average distance of a point to other points within its cluster, and b) the minimum average distance of a point to points in other clusters. These values are used to determine the silhouette score for each cluster. Finally, we calculate the overall silhouette score for the two clusters by considering the average silhouette score across all points based on equations below.

- **dissimilarity matrix**

Data Points	**Euclidean distance** $\sqrt{(X_1 - y_1)^2 + (x_2 - y_2)^2}$				
	A1 = (3, 6)	**A2 = (6, 3)**	**A3 = (8, 6)**	**A4 = (2, 1)**	**A5 = (5, 9)**
A1 = (3, 6)				$\sqrt{(3-2)^2 + (6-1)^2} = \sqrt{26}$ = **5.09**	
A2 = (6, 3)				$\sqrt{(6-2)^2 + (3-1)^2} = 2\sqrt{5}$ = **4.47**	
A3 = (8, 6)				$\sqrt{(8-2)^2 + (6-1)^2} = \sqrt{61}$ = **7.8**	
A4 = (2, 1)	$\sqrt{(3-2)^2 + (6-1)^2} = \sqrt{26}$ = **5.09**	$\sqrt{(6-2)^2 + (3-1)^2} = 2\sqrt{5}$ = **4.47**	$\sqrt{(8-2)^2 + (6-1)^2} = \sqrt{61}$ = **7.8**		$\sqrt{(5-2)^2 + (9-1)^2} = \sqrt{73}$ = **8.54**
A5 = (5, 9)				$\sqrt{(5-2)^2 + (9-1)^2} = \sqrt{73}$ = **8.54**	
SUM	$\sqrt{26}$	$2\sqrt{5}$	$\sqrt{61}$	**25.925**	$\sqrt{73}$
Avg b				**6.48**	

➢ Silhouette Coefficients

Data Points			Silhouette coefficient $S(i) = \frac{b(i)-a(i)}{\max\{b(i),a(i)\}}$		
	X	Y	b (i)	a (i)	coefficient
A1	3	6	$\sqrt{26}$	4.28	$\frac{\sqrt{26}-4.28}{\sqrt{26}} = 0.16$
A2	6	3	$\sqrt{20}$	4.64	$\frac{\sqrt{20}-4.64}{4.64} = -0.036$
A3	8	6	$\sqrt{61}$	4.28	$\frac{\sqrt{61}-4.28}{\sqrt{61}} = 0.45$
A4	2	1	6.48	0	$\frac{6.48-0}{6.48} = 1$
A5	5	9	$\sqrt{73}$	4.64	$\frac{\sqrt{73}-4.64}{\sqrt{73}} = 0.45$
					Overall Silhouette= 0.4048

➢ WSS score

We calculated the within-cluster sum of squares (WSS) by first finding the mean value for each cluster. Then, we subtract the mean value of the cluster from each data point belonging to that cluster. Next, we square and sum up all these differences across all data points within the cluster. to get the WSS.

Data Points			$(x_i - C_i)^2$		WSS = $\sum_{i=1}^{n}(X_i - C_i)^2$
	X	Y	New Centroid = (5.5, 6)		Summation
A1	3	6	$(3-5.5)^2$ = 6.25	$(6-6)^2$ = 0	6.25
A2	6	3	$(6-5.5)^2$ = 0.25	$(3-6)^2$ = 9	9.25
A3	8	6	$(8-5.5)^2$ = 6.25	$(6-6)^2$ = 0	6.25
A4	2	1			
A5	5	9	$(5-5.5)^2$ = 0.25	$(9-6)^2$ = 9	9.25
SUM					31

Part 2: Programming:

1)

a) create training and test datasets for remaining parts according to day feature.

- We generate a training set by selecting the records that have values 0, 1, and 2 in the 'day' column. Likewise, for the testing set, we include only the records with value 3 in the 'day' column. After that, we remove the 'ID' and 'Day' columns from both datasets.

- **Training set:**

```
train_data = df[df["Day"].isin([0, 1, 2])]
#train_data = train_data.drop(['ID','Day'],axis=1)
train_data
```

	ID	Latitude	Longitude	Day	Hour	Minute	Duration	RemainingTime	Resources	Coverage	OnPeakHours	GridNumber	Ligitimacy
0	1	45.442142	-75.303369	1	4	13	40	40	9	91	0	131380	1
1	1	45.442154	-75.304366	1	4	23	40	30	9	91	0	131380	1
2	1	45.442104	-75.303963	1	4	33	40	20	9	91	0	121996	1
3	1	45.441868	-75.303577	1	4	43	40	10	9	91	0	121996	1
4	2	45.447727	-75.147722	2	15	49	30	30	5	47	0	140784	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
14479	3999	45.445303	-75.165596	2	1	18	20	20	10	80	0	131397	1
14480	3999	45.445574	-75.165168	2	1	28	20	10	10	80	0	131397	1
14481	4000	45.436682	-75.152416	0	12	21	30	30	4	63	0	122015	1
14482	4000	45.436978	-75.153278	0	12	31	30	20	4	63	0	122015	1
14483	4000	45.436983	-75.153240	0	12	41	30	10	4	63	0	122015	1

7255 rows × 13 columns

- **Testing set:**

```
[47] test_data = df[df["Day"] == 3]
     #test_data = test_data.drop(['ID','Day'],axis=1)
     test_data
```

	ID	Latitude	Longitude	Day	Hour	Minute	Duration	RemainingTime	Resources	Coverage	OnPeakHours	GridNumber	Ligitimacy
16	6	45.410236	-75.208755	3	22	25	30	30	10	32	0	75088	1
17	6	45.409787	-75.208022	3	22	35	30	20	10	32	0	75088	1
18	6	45.409407	-75.207825	3	22	45	30	10	10	32	0	65704	1
26	10	45.544018	-75.146364	3	20	39	20	20	2	82	0	300312	1
27	10	45.544576	-75.146364	3	20	49	20	10	2	82	0	300312	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
14429	3984	45.541816	-75.177356	3	4	36	60	10	9	43	0	300308	1
14445	3990	45.461207	-75.209171	3	3	4	40	40	4	60	0	159544	1
14446	3990	45.461241	-75.209067	3	3	14	40	30	4	60	0	159544	1
14447	3990	45.461261	-75.209205	3	3	24	40	20	4	60	0	159544	1
14448	3990	45.461007	-75.208843	3	3	34	40	10	4	60	0	159544	1

2460 rows × 13 columns

b) confusion matrixes and F1 scores of NB and KNN classifiers.

We define:

- **confusion_mx function** to calculate confusion matrix for Bernoulli Naive Bayes and K-nearest neighbors' classifiers. Additionally, we plot the confusion matrix as a heatmap.

```
[14] def confusion_mx (y_test,y_pred,label):
         cm = confusion_matrix(y_test,y_pred)
         plt.figure(figsize=(4, 3))
         sns.heatmap(cm, annot=True,fmt='g', cmap='Blues')
         plt.xlabel('Predicted labels')
         plt.ylabel('True labels')
         plt.title(f'Confusion Matrix {label}',size= 10)
         plt.show()
```

- **BNB function** to train a Bernoulli Naive Bayes classifier, make predictions on the labels, and invoke the **confusion_mx** function to compute the cm. Afterwards, it calculates the F1 score.

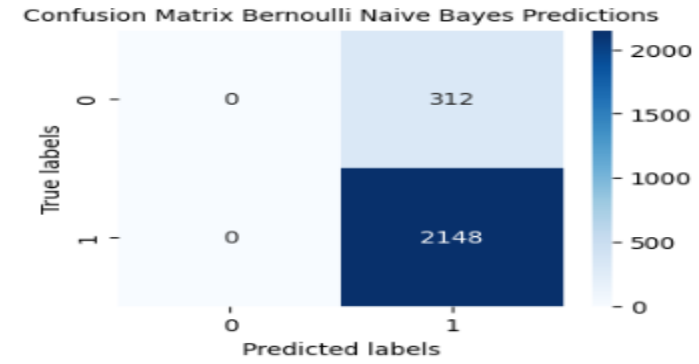
```
def BNB(x_train, x_test, y_train, y_test):
    BNB = BernoulliNB()
    BNB.fit(x_train , y_train)
    y_pred = BNB.predict(x_test)
    confusion_mx(y_test , y_pred , "Bernoulli Naive Bayes Predictions")
    f1 = f1_score(y_test, y_pred )
    print("F1-Score is : ",f1)
    return f1
```

- **KNN function** to train a K-nearest neighbors' classifier, make predictions on the labels, and invoke the **confusion_mx** function to compute the cm. Afterwards, it calculates the F1 score.

```
def KNN(x_train , x_test , y_train , y_test ):
      knn = KNeighborsClassifier(n_neighbors = 5)
      model = knn.fit(x_train , y_train)
      knn_pred = model.predict(x_test)
      confusion_mx(y_test , knn_pred , "KNN Prediction")
      f1 = f1_score(y_test, knn_pred )
      print("F1-Score is : ",f1)
      return f1
```

- **confusion matrixes and F1 scores of NB**

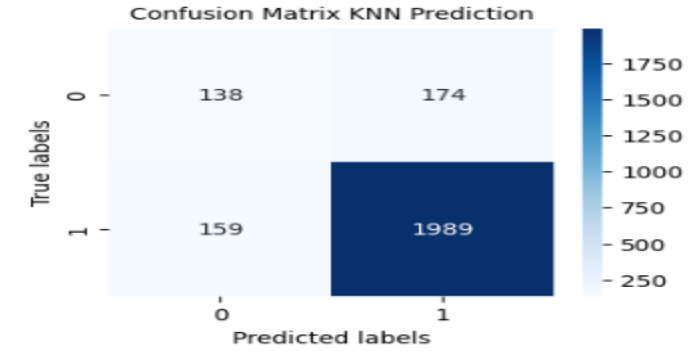
```
[17] BNB_F1 = BNB(X_train , X_test , Y_train , Y_test)
```




```
F1-Score is :  0.9322916666666667
```

- **confusion matrixes and F1 scores of KNN**

```
[21] KNN_F1 = KNN(X_train , X_test , Y_train , Y_test)
```



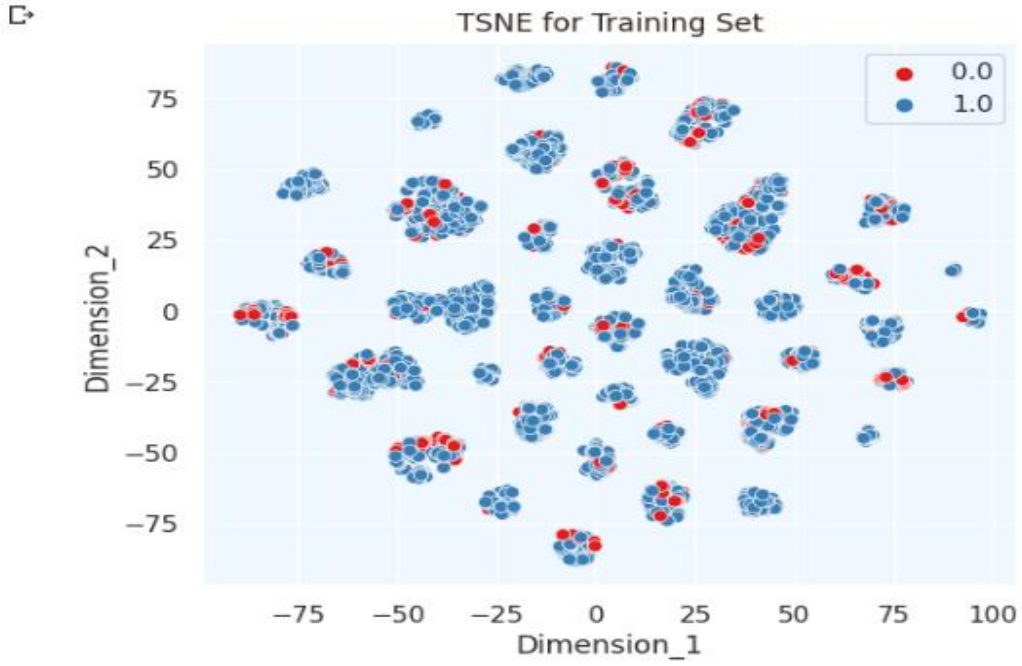

```
F1-Score is :  0.9227557411273487
```

c) 2D TSNE plots for Training and Testing sets.

- We define **draw_TSNE function** to reduce the dimensionality of training and testing sets and plots a scatterplot using seaborn to visualize the data points in the 2D space using different color palette.

```
[22] def draw_TSNE(X_train, Y_train , palette):
       tsne = TSNE(n_components=2, random_state=0)
       X_2d = tsne.fit_transform(X_train)
       dftsne = pd.DataFrame(X_2d)
       dftsne['cluster'] = Y_train
       dftsne.columns = ['X1','X2','cluster']
       plt.figure(figsize=(6, 5))
       sns.set(rc={ 'axes.facecolor':'aliceblue', 'figure.facecolor':'white'})
       sns.scatterplot(data = dftsne , x = 'X2', y = 'X1', hue = 'cluster' , legend = "full" , alpha = 1 , palette=palette)
       plt.legend()
       plt.show()
```

➢ **2D TSNE plots for Training set:**



➢ **2D TSNE plots for Testing set:**

2) Dimensionality Reduction (DR) methods

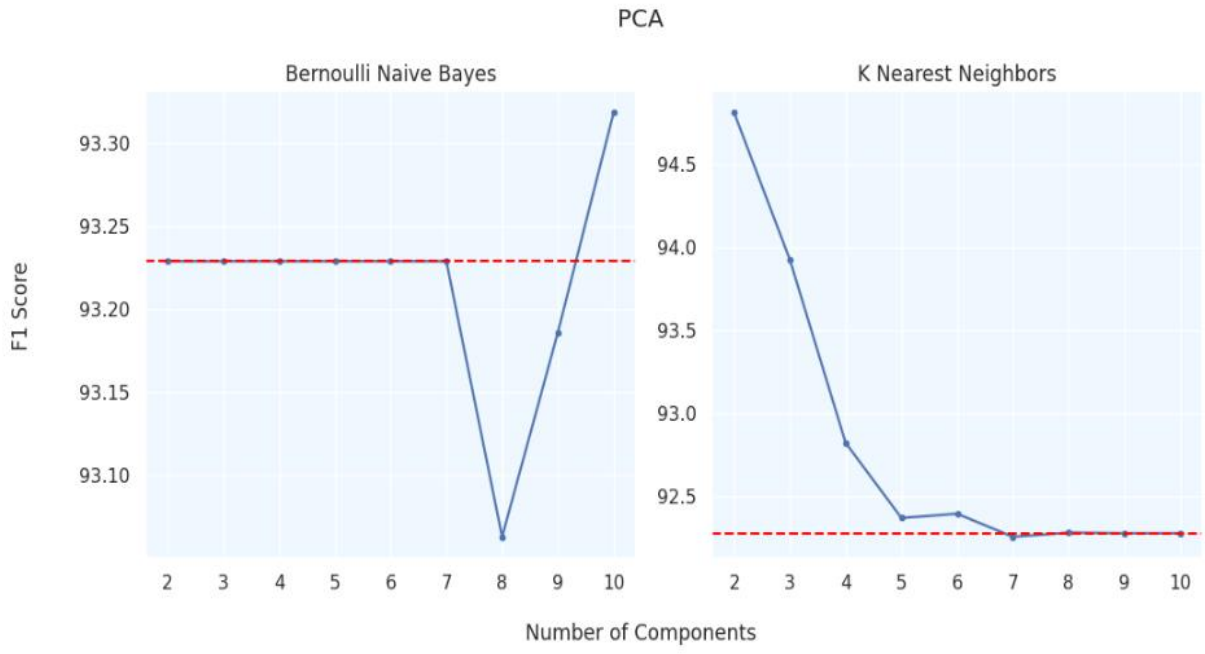
a) number of components vs f1 score for PCA and AE using (NB and KNN)

- We define a **N_PCA function** to display the F1-Score for PCA data with Bernoulli Naive Bayes and K Nearest Neighbors models. Additionally, it identifies the maximum F1 score achieved by both classifiers according to the number of components.

- **F1-Score for PCA data for Bernoulli Naive Bayes and knn.**

```
N_PCA(X_train , X_test , Y_train , Y_test )
```

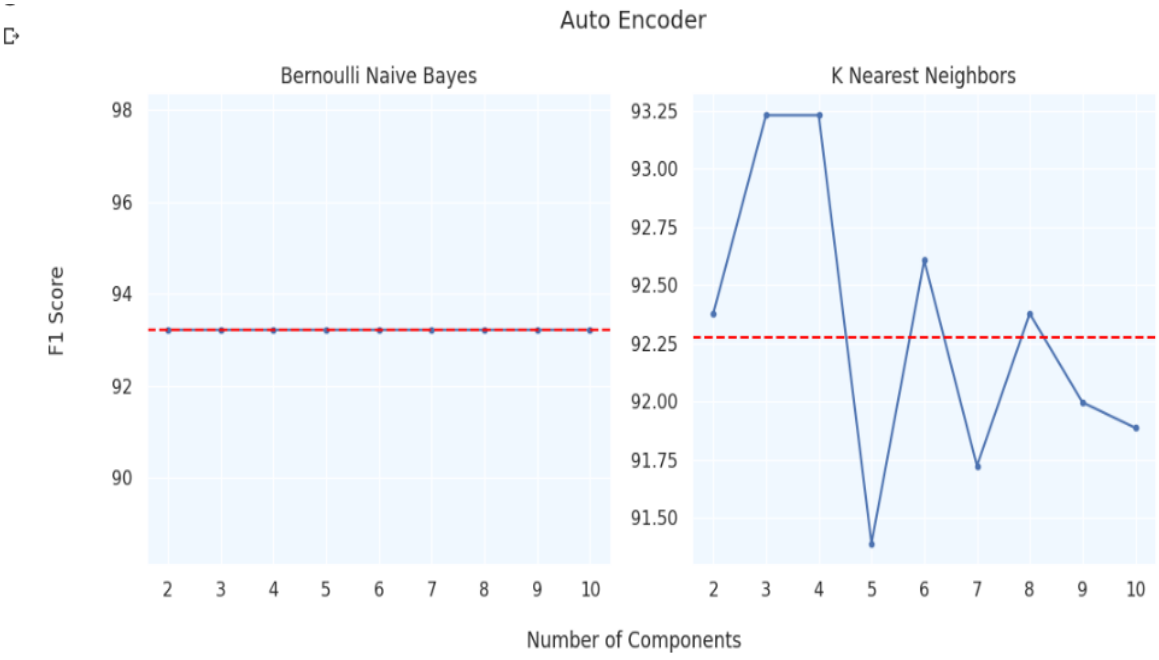
```
F1-Score for PCA data for Bernoulli Naive Bayes Model :

 ['93.23', '93.23', '93.23', '93.23', '93.23', '93.23', '93.06', '93.19', '93.32']  %

The Max F1 score for Bernoulli Naive Bayes Model :  93.32

And the number of components :  10

----------------------------------------------------------

F1-Score for PCA data for K Nearest Neighbors Model :

 ['94.81', '93.92', '92.82', '92.37', '92.39', '92.25', '92.28', '92.28', '92.28']  %

The Max F1 score for K Nearest Neighbors Model :  94.81

And the number of components :  2
```



- We define a **N_AE function** to display the F1-Score for Auto Encoder data with Bernoulli Naive Bayes and K Nearest Neighbors models. Additionally, it identifies the maximum F1 score achieved by both classifiers according to the number of components.

- **F1-Score for Auto Encoder data for Bernoulli Naive Bayes and knn.**

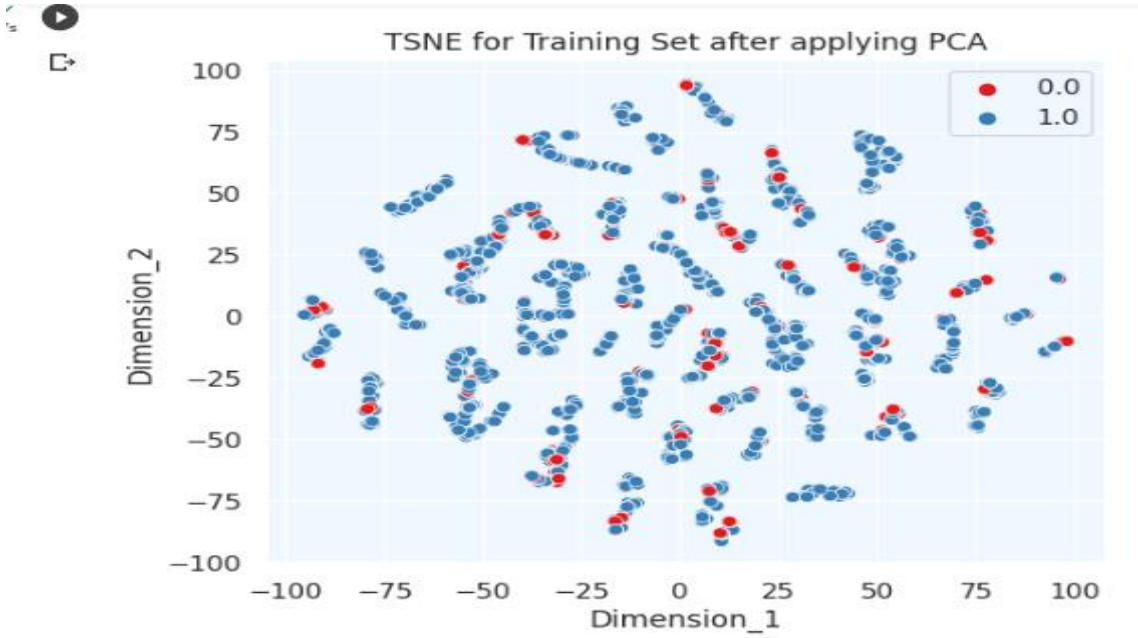
```
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:686
  warnings.warn(
F1-Score for Auto Encoder data for Bernoulli Naive Bayes Model :

 ['93.23', '93.23', '93.23', '93.23', '93.23', '93.23', '93.23', '93.23', '93.23']  %


The Max F1 score for Bernoulli Naive Bayes Model :  93.23

And the number of components :  2


----------------------------------------------------------


F1-Score for Auto Encoder data for K Nearest Neighbors Model :

 ['92.38', '93.23', '93.23', '91.39', '92.61', '91.72', '92.38', '92.00', '91.89']  %


The Max F1 score for K Nearest Neighbors Model :  93.23

And the number of components :  3
```



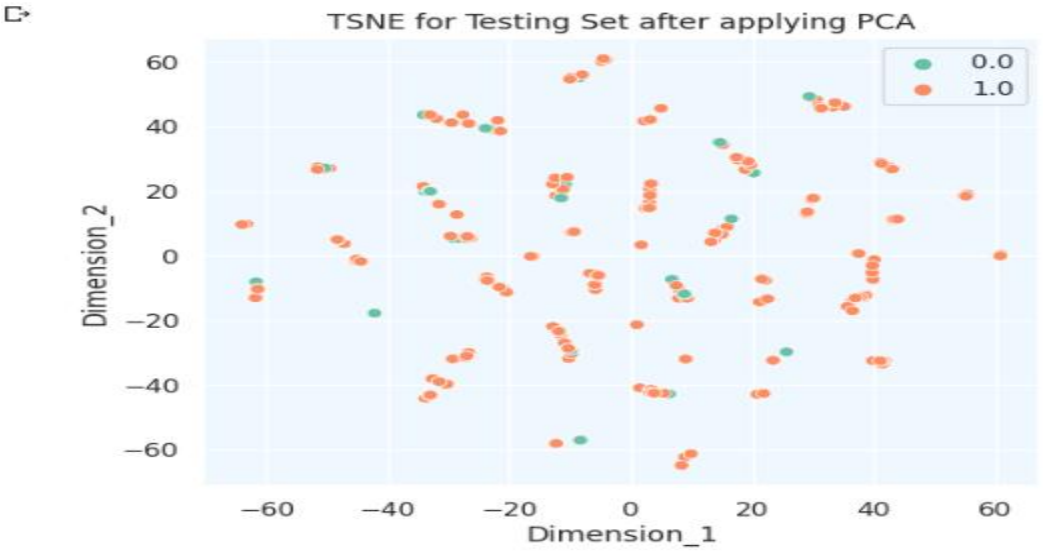
b) 2D TSNE plots for the best performance (DR) method based on f1 score of (NB and KNN)

- We plot the best performance (DR) method which is PCA with F1 score **94.81** for **knn** with **2** dimensions using **draw_TSNE function** to plot the reduced dimensions training and testing sets.

- **2D TSNE plots for Training set:**



- **2D TSNE plots for Testing set:**



3)Feature Selection methods:

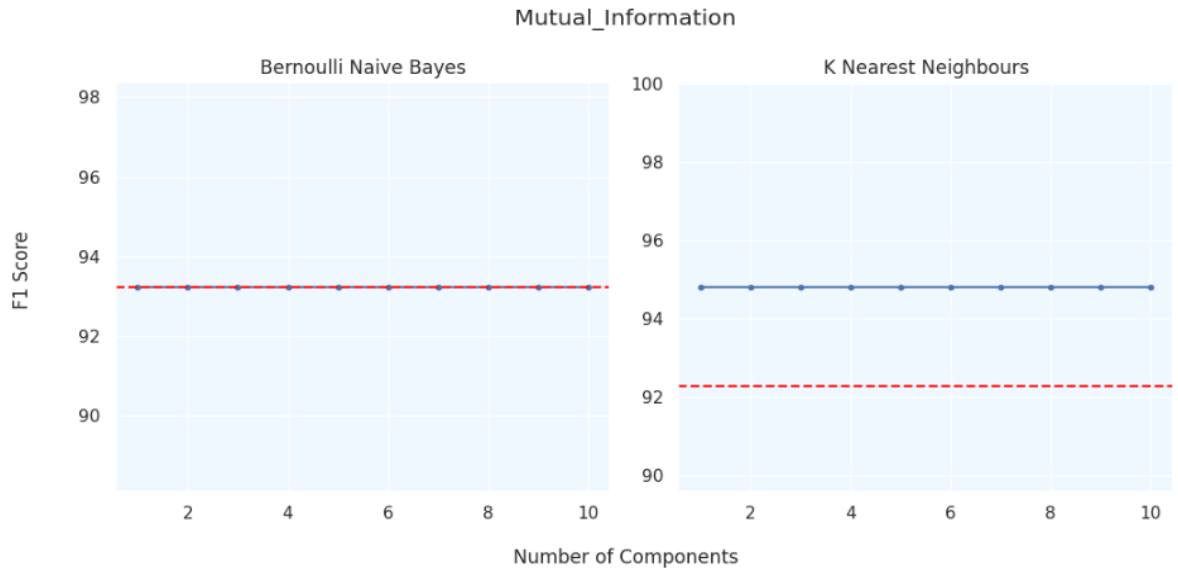
a) Filter Methods with improved base line performance

- We created **Mutual_Info** function to display the F1-Score after applying Mutual_Information () with Bernoulli Naive Bayes and K Nearest Neighbors models. Additionally, it identifies the maximum F1 score achieved by both classifiers according to the best number of features.

- **F1-Score after applying Mutual Information for Bernoulli Naive Bayes and knn.**

```
Mutual_Info(X_train_pca, Y_train,X_test_pca, Y_test)
```

```
F1-Score after applying Mutual_Information for Bernoulli Naive Bayes Model :
 ['93.23', '93.23', '93.23', '93.23', '93.23', '93.23', '93.23', '93.23', '93.23', '93.23']  %

The Max F1 score for Bernoulli Naive Bayes Model :  93.23
And the best number of features :  1

----------------------------------------------------------

F1-Score after applying Mutual_Information for K Nearest Neighbours Model :
 ['94.81', '94.81', '94.81', '94.81', '94.81', '94.81', '94.81', '94.81', '94.81', '94.81']  %

The Max F1 score for K Nearest Neighbours Model :  94.81
And the best number of features :  1
```

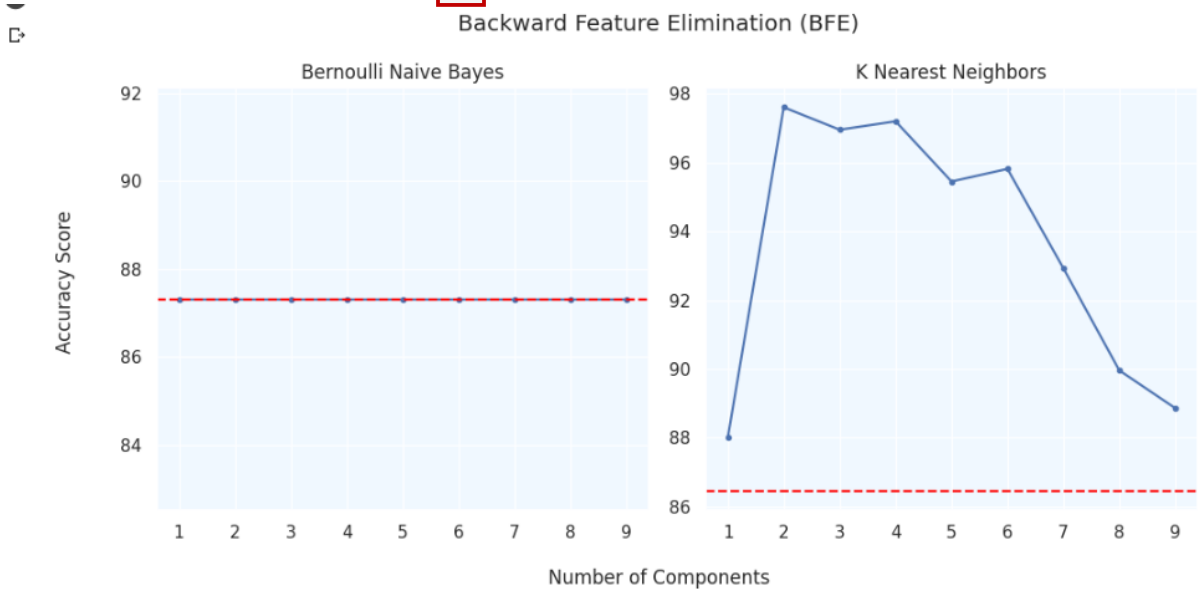


b) Wrapper Methods with the base line performance

- We created **select_features_backward_acc** function to display the Accuracy Score after applying SequentialFeatureSelector () with direction="**backward**" as we have small number of features with Bernoulli Naive Bayes and K Nearest Neighbors models. Additionally, it identifies the maximum Accuracy score achieved by both classifiers according to the best number of features.

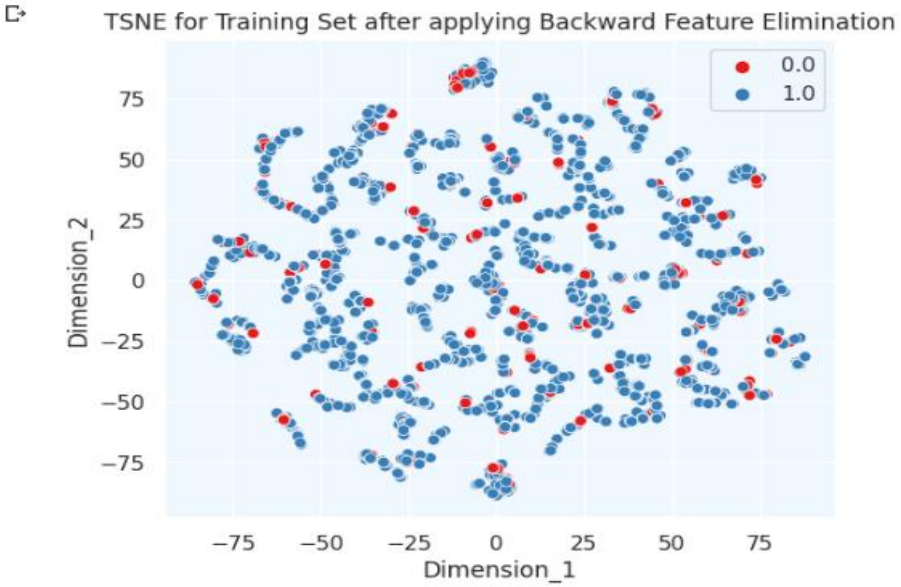
- **Accuracy Score after applying** SequentialFeatureSelector () **for Bernoulli Naive Bayes and knn.**

```
Accuracy-Score after applying Backward Feature Elimination for Bernoulli Naive Bayes Model :

 ['87.32', '87.32', '87.32', '87.32', '87.32', '87.32', '87.32', '87.32', '87.32']  %

The Max Accuracy score for Bernoulli Naive Bayes Model :  87.32

And the best number of features :  1

----------------------------------------------------------

 Accuracy-Score after applying Backward Feature Elimination for K Nearest Neighbors Model :

['88.01', '97.60', '96.95', '97.20', '95.45', '95.81', '92.93', '89.96', '88.86']  %

The Max Accuracy score for K Nearest Neighbors Model :  97.60

And the best number of features :  2
```

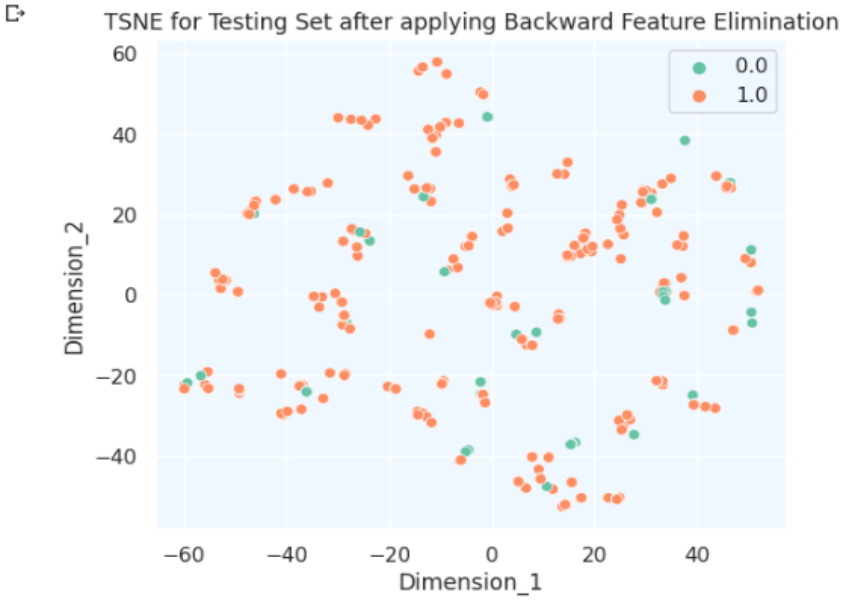


c) 2D TSNE plots for the best method.

- We plot the best performance method “wrapper” which is Backward Feature Elimination with Accuracy score **97.60** for **knn** when number of features = **2** using **draw_TSNE function** to plot the reduced dimensions training and testing sets.
- **2D TSNE plots for Training Set:**



- **2D TSNE plots for Testing Set:**

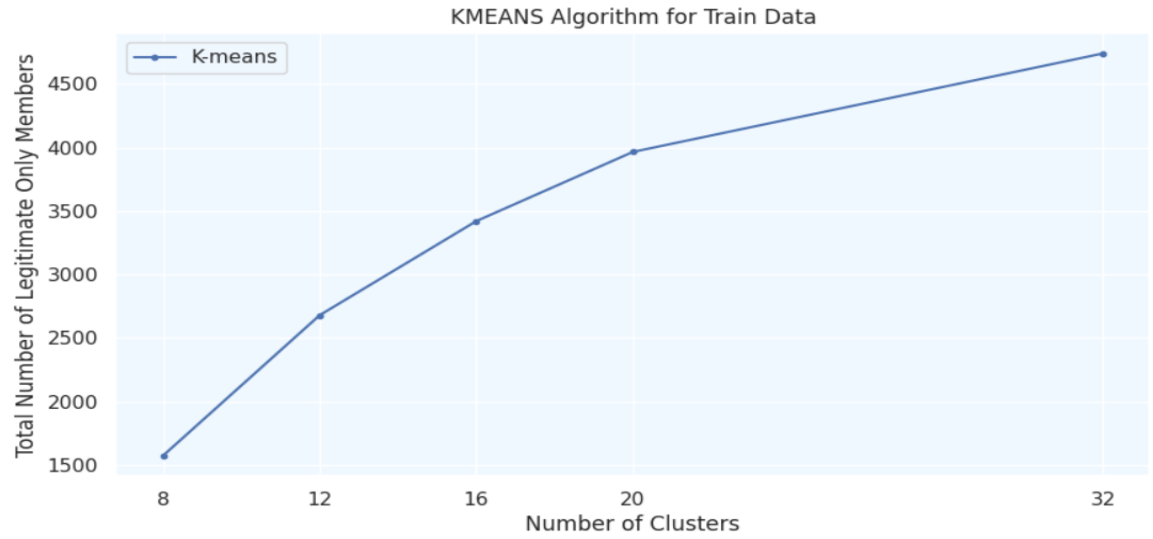


4)clustering based methods

a) Apply K-Means

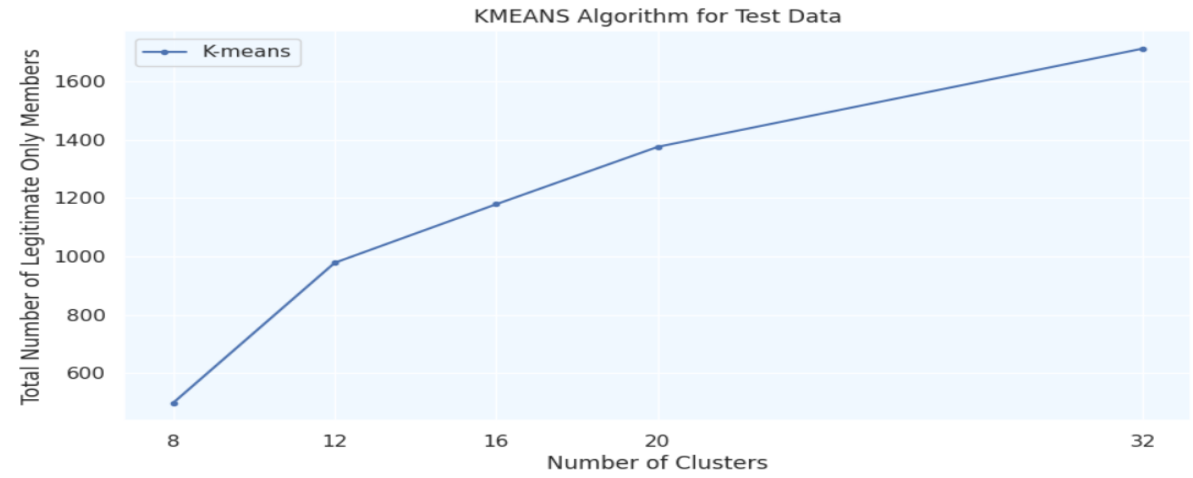
- We applied Kmeans algorithm with number of clusters 8, 12, 16, 20, 32 the for each cluster we get the legitimate only clusters from the **Y_train** data we have before. Then we plot the number of legitimate only clusters vs the number of clusters
- **Kmeans for Training Data:**

```
Sum of legitimate only Members :  [1573, 2679, 3420, 3965, 4740]
```



- **Kmeans for Testing Data:**

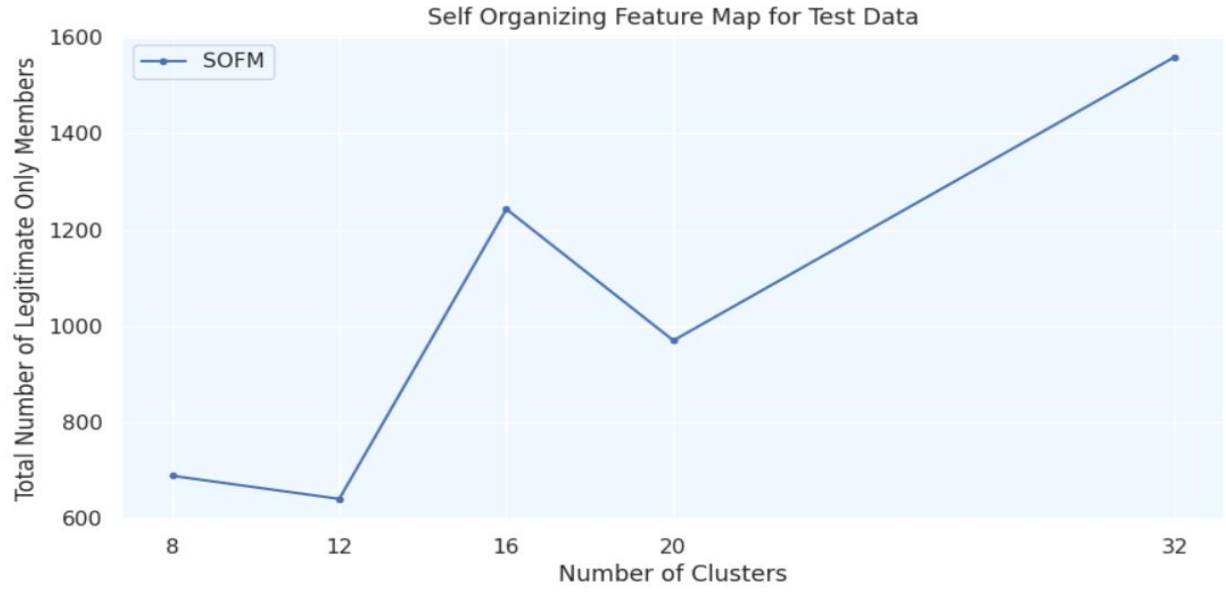
```
Sum of legitimate only Members :  [497, 978, 1179, 1376, 1713]
```



b) Apply Self Organizing Feature Map:

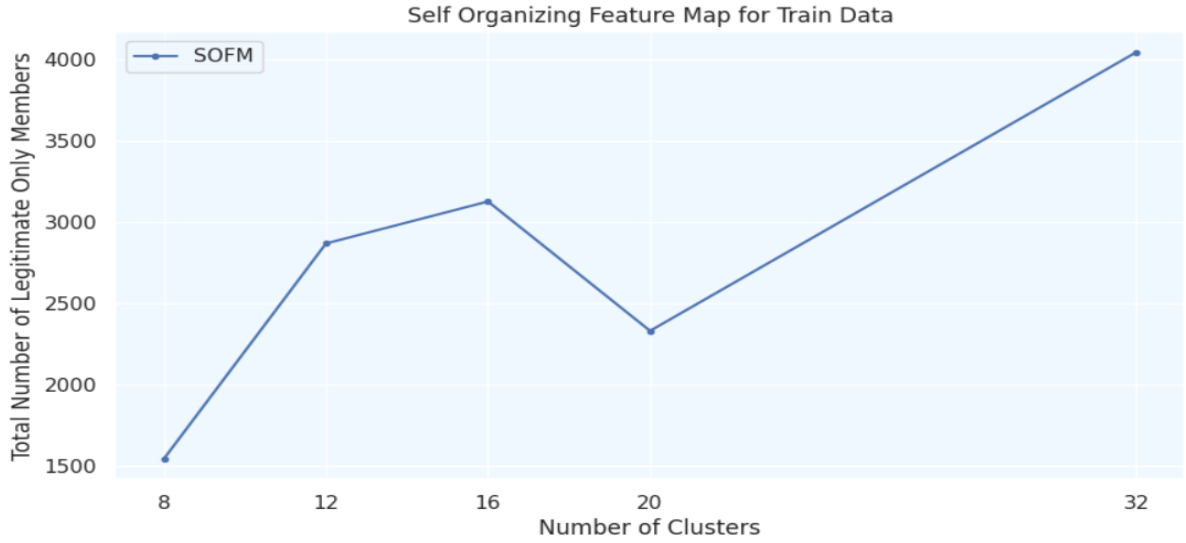
- We applied SOFM with the same number of clusters in Kmeans to compare the results in both of them. Then we plot the number of legitimate only clusters vs the number of clusters
- **SOFM for Training Data:**

```
Sum of legitimate only Members:  [689, 641, 1243, 970, 1559]
```



- **SOFM for Testing Data:**

```
Sum of legitimate only Members:  [1545, 2868, 3126, 2331, 4043]
```

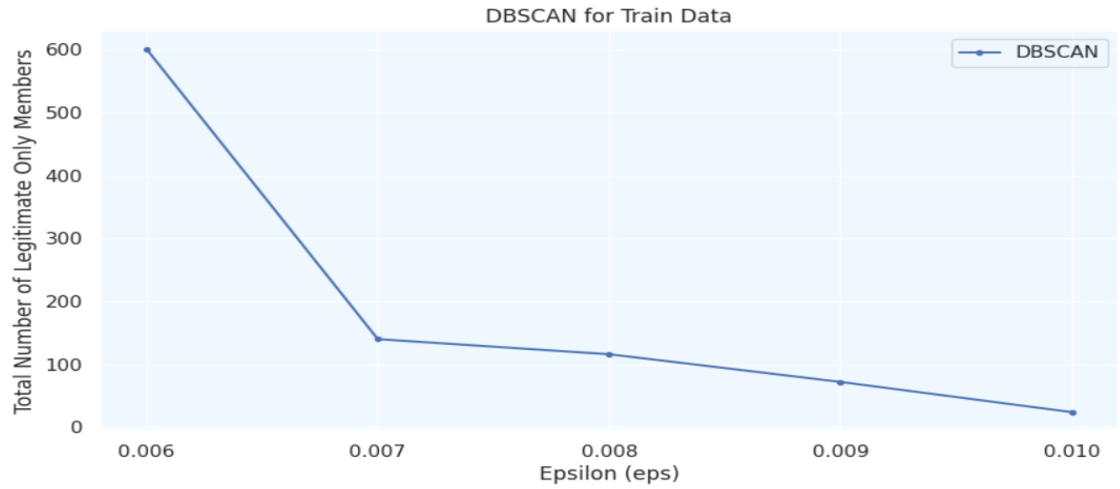


c) Apply DBSCAN

- We applied DBSCAN with different numbers of **Epsilon 0.006, 0.007, 0.008, 0.009, 0.01** and set the **min samples** with fixed number that the default min_samples and we got 5 different of clusters from the above two models

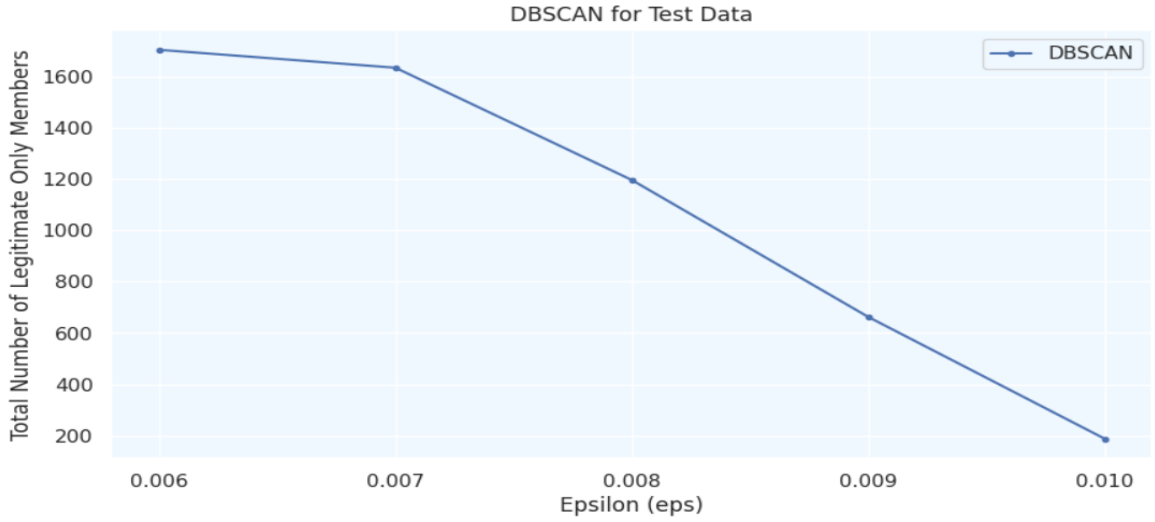
- **DBSCAN for Training Data:**

```
Number of Clusters for each EPS :  [44, 17, 13, 9, 4]
Sum of legitimate only Members :  [601, 140, 116, 72, 24]
```



- **DBSCAN for Testing Data:**

```
Number of Clusters for each EPS :  [116, 79, 48, 27, 18]
Sum of legitimate only Members :  [1704, 1634, 1196, 662, 187]
```



5) Conclusion

1. we split our data into training and testing data which contains 7255, 2460 rows respectively then we split these data into X and Y for each set.
   We choose **Bernoulli Naïve Bayes** Classifier because of the nature of label data which consists of only ones and zeros which is binary classification.
   We got almost 93 % in F1 score which indicates the Bernoulli classifier works quite good due to the imbalance distribution of data, and for the **KNN** classifier we got 92 % in F1 score it is good but of course less than the Bernoulli classifier.
   In visualizing the **TSNE** for Training and Testing sets we noticed that the data is very imbalanced the non-legitimate members are only about 15 % on the whole members

2. we applied **PCA** reduction technique to reduce the dimensionality to reduce power consumption and resources in case of bigger problem.
   We choose different numbers of components from 2 to 10 to see which is the best number of components for each classifier.
   For the **Bernoulli** classifier the F1 score didn't changed much for the different numbers of components but we got the best F1 score when the number of components is **10** and it is higher than the baseline F1 score.
   For the **KNN** Classifier the F1 score has slight change for each number of components and we got the best F1 score when it was **2** and it is much higher than the baseline F1 score.
   Coming to the **Auto Encoder** part we picked the same numbers as in the PCA.
   In the **Bernoulli** Classifier there wasn't any change in the F1 score during the changing numbers of components and it is the same baseline F1 score.
   In the **KNN** we got the best F1 score for the number of components is 3 and it is higher than the baseline F1 score by 1 %.

Then the best F1 score in the 4 different stages is when we applied **PCA** with number of components equal 2 in the **KNN** Classifier.
We took the data with number of components equal 2 in **PCA** and plot the TSNE visualization and the results changed than before the PCA the data became smoother and it isn't so much complicated due to the number of components is just 2

3. In point 3 firstly We created **Mutual_Info** function to display the F1-Score after applying SelectKBest() with score_function =**mutual_info_classif**
to Bernoulli Naive Bayes and K Nearest Neighbors models, mutual information was our choice as we apply a variance threshold, but it achieved a lower performance. Additionally, this function identifies the maximum F1 score achieved by both classifiers according to the best number of features with results knn (f1= 94.81, number of features= 1)
with Bernoulli Naive Baye (f1=93.23, number of features= 1).
Secondly, we created **select_features_backward_acc** function to display the Accuracy Score after applying SequentialFeatureSelector () with direction="**backward**" as we have small number of features to Bernoulli Naive Bayes and K Nearest Neighbors models. SequentialFeatureSelector was our choice as we tried to use recursive feature elimination, but we couldn't as knn and Bernoulli have no coefficient or importance parameters Additionally, it identifies the maximum Accuracy score achieved by both classifiers according to the best number of features with results knn (accuracy = 97.60, number of features= 2)
with Bernoulli Naive Baye (accuracy = 87.32, number of features= 1).
Finally, we plot the best performance method "wrapper" which is Backward Feature Elimination with Accuracy score **97.60** for **knn** when number of features = **2** using **draw_TSNE function** to plot the reduced dimensions training and testing sets.

4. In the **Kmeans** we put a list of clusters numbers 8, 12, 16, 20, 32 to get the most legitimate members only and here when we go up the list the number of legitimate only members clearly increase not only in the training data but also in the testing data
   In the **SOFM** with the same list of clusters number there isn't no pattern in increasing or decreasing the number of legitimate members only one time it increases then the second it decreases not only in training data but also in testing data
   In the **DBSCAN** there is no clusters number to pass to the model we have **Epsilon** and **Min Samples** we fixed the number of min samples with the default number which is 5 and we changed the number of Epsilon to get different numbers of clusters for each number, we picked 5 numbers 0.006, 0.007, 0.008, 0.009, 0.01 and we got these number of clusters 44, 17, 13, 9, 4 respectively for Training Data and 116, 79, 48, 27, 18 for Testing Data here we noticed a pattern in decreasing the number of legitimate members only when we go down with the number of clusters in the both Training and Testing Data