Professional Master's in Artificial Intelligence
APPLIED MACHINE LEARNING (ELG 5255)

Subject: Assignment 3 (Dimensionality Reduction & Unsupervised Learning)

By

Abdelmageed Ahmed Abdelmageed Hassan

Mohamed Sayed Abdelwahab Hussein

Under Supervision
Dr. Murat Simsek

1. Load the dataset and split features and labels for training and test sets

```
[155] pokemon_train = pd.read_csv("/content/gdrive/MyDrive/Pokemon_train.csv")
      pokemon_test = pd.read_csv("/content/gdrive/MyDrive/Pokemon_test.csv")
      print(len(pokemon_train))
      print(len(pokemon_test))

      display(pokemon_train.head())
```

```
1251
313
```

Here we will split the features and labels for training and testing sets and convert class names to numbers from 0 to 16 using LabelEncoder

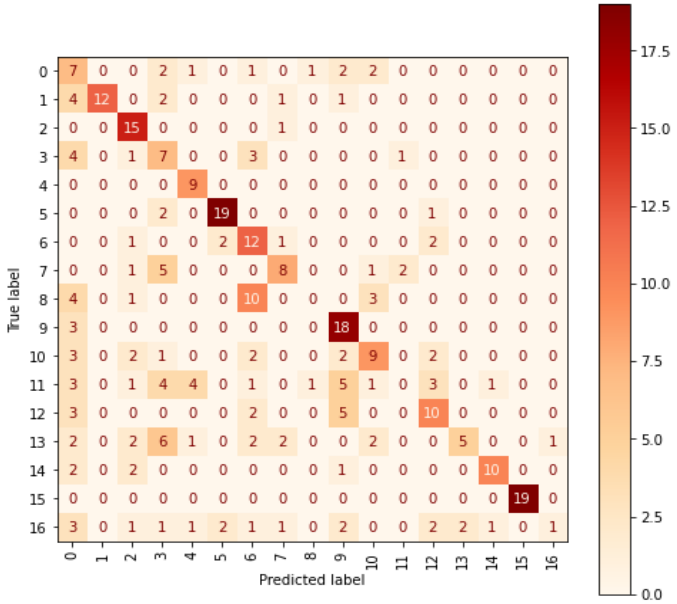
```
[ ] X_train = pokemon_train.drop('type1',axis=1).values
    y_train = pokemon_train['type1']
    le = preprocessing.LabelEncoder()
    y_train = le.fit_transform(y_train)

    X_test = pokemon_test.drop('type1',axis=1).values
    y_test = pokemon_test['type1']
    y_test = le.fit_transform(y_test)
```

2. Apply Gaussian Naïve Bayes classifier (GNB) and Support Vector Machine (SVM) to the dataset

(2.1) Gaussian Naïve Bayes classifier (GNB)

```
[157] NB_model = GaussianNB()
      NB_model.fit(X_train,y_train)
      y_predic_test = NB_model.predict(X_test)
      test_accuracy_nb_base = accuracy_score(y_test, y_predic_test)
      print("\n testing accuracy :   "  ,round(test_accuracy_nb_base,3),"\n")
      print(classification_report(y_test, y_predic_test))
      fig, ax = plt.subplots(figsize=(8, 8))
      plot_confusion_matrix(NB_model, X_test, y_test, xticks_rotation='vertical', cmap=plt.cm.OrRd, ax=ax);
```

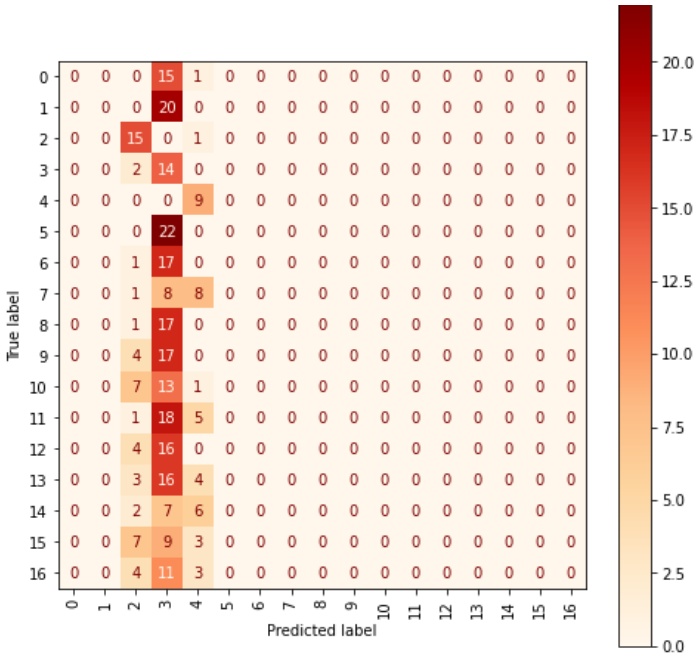
we got the accuracy of **51** % after fitting training data to the Gaussian Naïve Bayes Classifier with default parameters.



(2.2) Support Vector Machine (SVM)

```
[6]  svm_model = svm.SVC()
     svm_model.fit(X_train,y_train)
     y_predic_test2 = svm_model.predict(X_test)
     test_accuracy_svm_base = accuracy_score(y_test, y_predic_test2)
     print("\n testing accuracy :  "  ,round(test_accuracy_svm_base,3),"\n")
     print(classification_report(y_test, y_predic_test2))
     fig, ax = plt.subplots(figsize=(8, 8))
     plot_confusion_matrix(svm_model, X_test, y_test, xticks_rotation='vertical', cmap=plt.cm.OrRd, ax=ax);
```

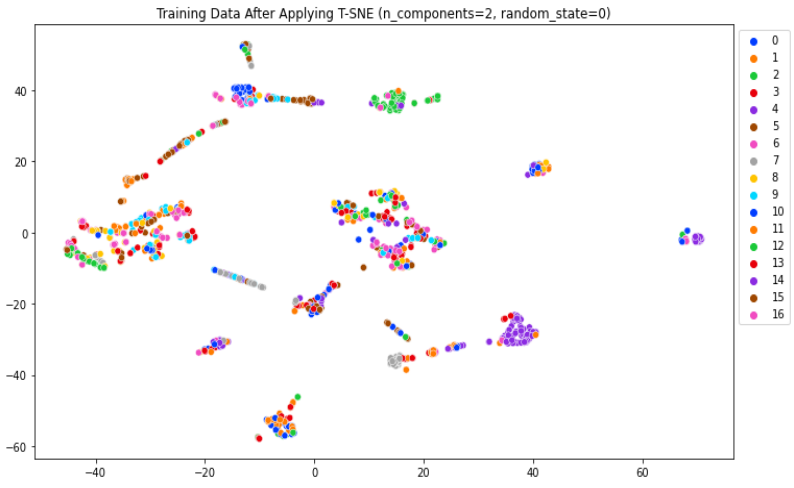
we got the accuracy of **12** % after fitting training data to the Support Vector Machine Classifier with default parameters.



Apply TSNE(n_components=2, random_state=0) to training and test set and visualize

(2.3) Apply TSNE with (n_components=2) and visualize

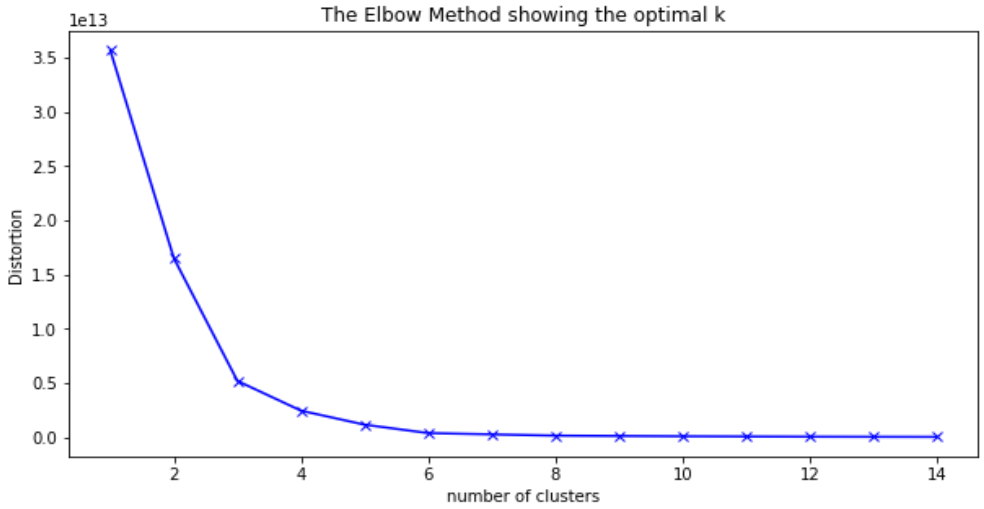
```
[7]  tsne = TSNE(n_components=2, random_state=0)
     X_train_tsne = tsne.fit_transform(X_train)
     X_test_tsne = tsne.fit_transform(X_test)

     print('Shape of the Training Data befor T-SNE:', X_train.shape)
     print('Shape of the Training Data after T-SNE:', X_train_tsne.shape)
     print('Shape of the Testing Data befor T-SNE:', X_test.shape)
     print('Shape of the Testing Data after T-SNE:', X_test_tsne.shape)

     Shape of the Training Data befor T-SNE: (1251, 32)
     Shape of the Training Data after T-SNE: (1251, 2)
     Shape of the Testing Data befor T-SNE: (313, 32)
     Shape of the Testing Data after T-SNE: (313, 2)
```



3. Choose the best number of clusters for k-means clustering algorithm

(3.1) Using the elbow rule, plot the distortion score (a.k.a inertia) vs the number of clusters

```
[12] distortions_km = []
     for k in range(1,15):
         km_model = KMeans(n_clusters=k)
         km_model.fit(X_train)
         distortions_km.append(km_model.inertia_)

     plt.figure(figsize=(10,5))
     plt.plot(range(1,15), distortions_km, 'bx-')
     plt.xlabel('number of clusters')
     plt.ylabel('Distortion')
     plt.title('The Elbow Method showing the optimal k')
     plt.show();
```



from above figure we can determine the optimal number of clusters for K-Means as **4**

let's make clustering using K-Means algorithm on the training data using no. clusters = 4

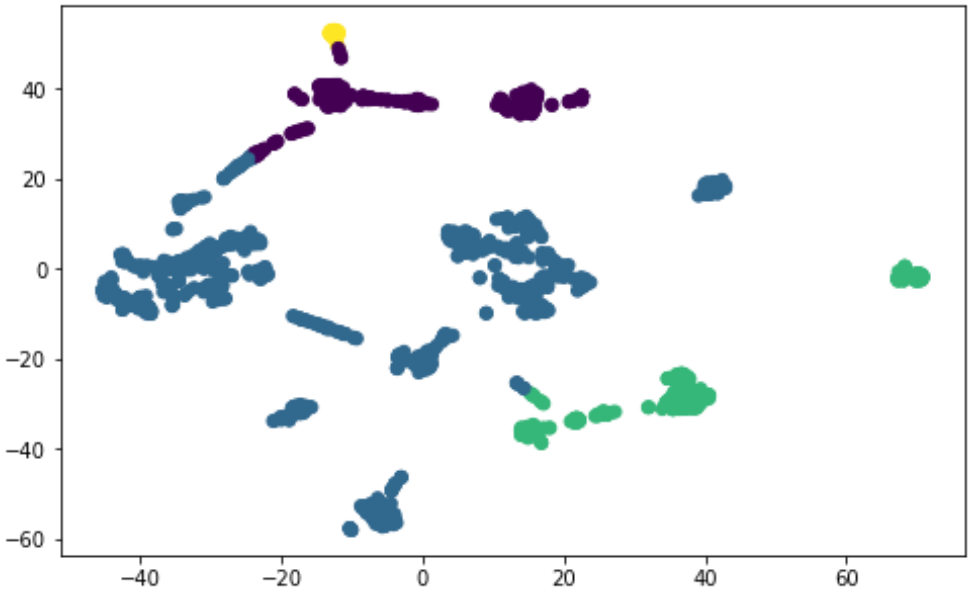
(3.2) Determine the optimal number of clusters for k-means

```
[80] # the optimal number of clustera is 4
     km_model = KMeans(n_clusters=4)
     km_preds = km_model.fit_predict(X_train)
```

(3.3) Plot clustered data with optimum number of clusters

```
[83] t_sne = TSNE(2)
     plt.figure(figsize=(8,5))
     plt.scatter(t_sne.fit_transform(X_train)[:, 0], t_sne.fit_transform(X_train)[:, 1], c=km_preds, cmap='viridis');
```

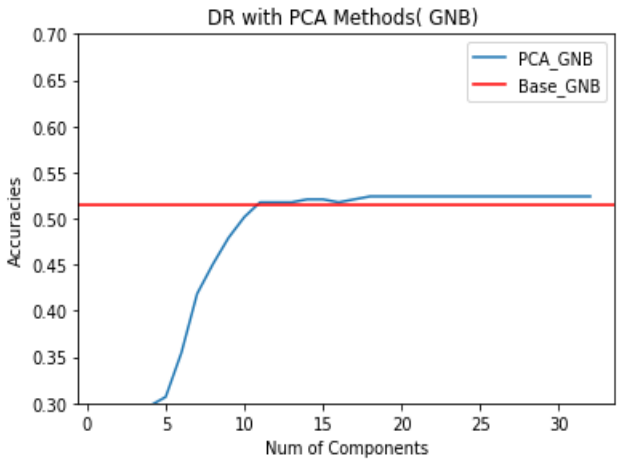
Plot clustered data with optimum number of cluster (Using T-SNE for visualization only)



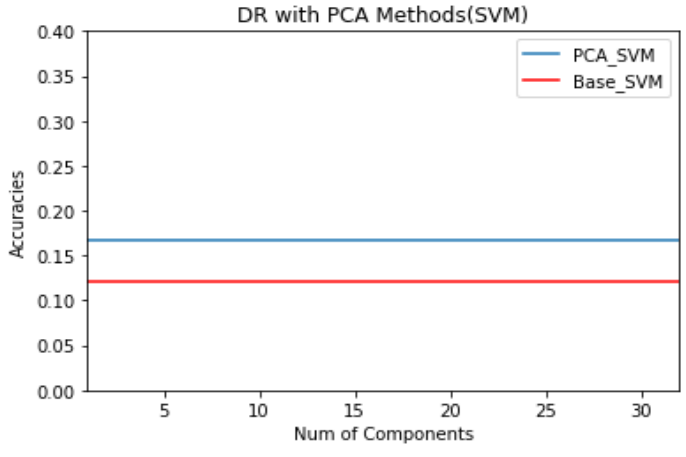
4. **Plot the number of features Accuracy graph with baseline performances for each classifier as shown below. DR method should be applied training set, and test set should be transformed accordingly. Graph should be plotted based on the test accuracy.**

4.1 GNB and SVM classifiers test accuracies based on Principal Component Analysis (PCA)

```
Maximum accuracy of GNB based PCA: 0.5239616613418531
base GNB accuracy: 0.5143769968051118
Best number of features: 18
```

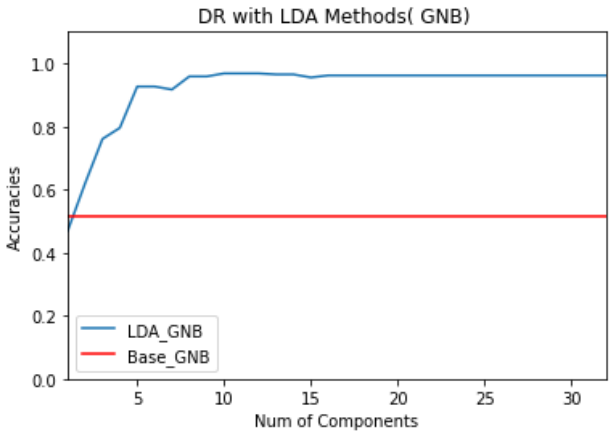



```
Maximum accuracy of SVM based on PCA: 0.16613
base SVM  accuracy: 0.12140575079872204
Best number of features: 1
```

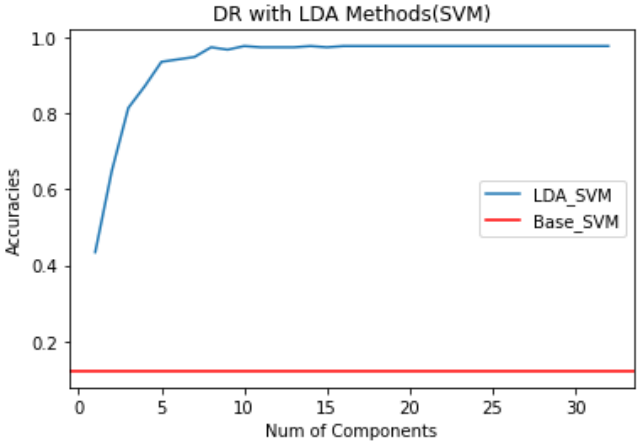


4.2 GNB and SVM classifiers test accuracies based on Linear Discriminant Analysis (LDA)

```
Maximum accuracy of GNB based on LDA   : 0.961661
base GNB accuracy: 0.5143769968051118
Best number of features: 10
```




```
Maximum accuracy of SVM based LDA : 0.9776357827476039
Base SVM  accuracy: 0.12140575079872204
Best number of features: 10
```



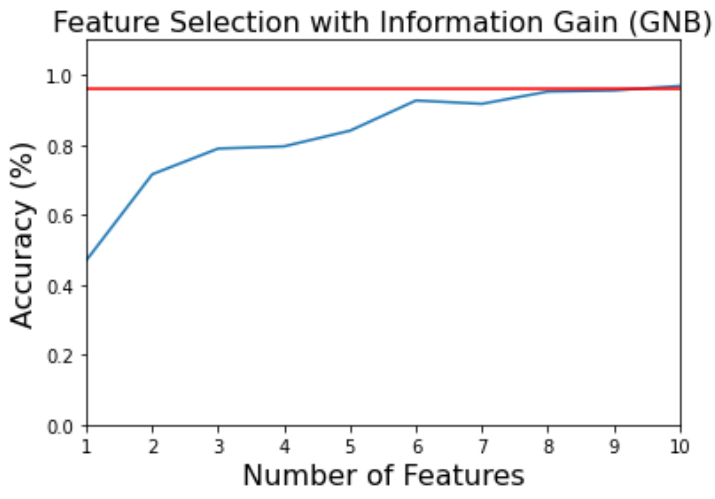
best results we got is SVM with LDA dimensionality reduction (10 component with 0.977 accuracy)

5. Feature Selection methods

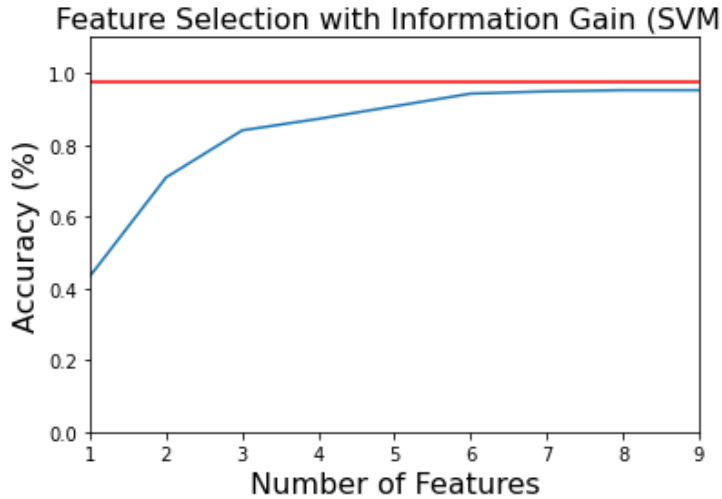
The new base line accuracy (improved base line accuracy) is 0.961 for GNB and 0.977 for SVM based on LDA dimensionality reduction

5.1 Filter Methods (using Information Gain methods)

```
Maximum accuracy: 0.9680511182108628
Base GNB accuracy: 0.9616613418530351
Best number of features: 10
```




```
Maximum SVM_FS accuracy: 0.952076677316294
 The Improved Base SVM accuracy: 0.9776357827476039
Best number of features: 8
```



5.2 Wrapper Methods (using Forward Feature Elimination methods)

forward elimination based GNB estimator

avg_score represent the cross validation score for each numbers of features

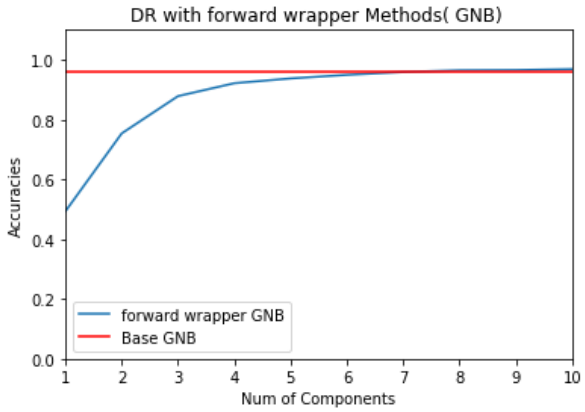
	feature_names	cv_scores	avg_score
1	(0,)	[0.49800796812749004, 0.508, 0.508, 0.464, 0.488]	0.493202
2	(0, 4)	[0.7689243027888446, 0.784, 0.72, 0.724, 0.772]	0.753785
3	(0, 2, 4)	[0.8884462151394422, 0.896, 0.868, 0.856, 0.884]	0.878489
4	(0, 2, 4, 8)	[0.9043824701195219, 0.916, 0.928, 0.936, 0.924]	0.921676
5	(0, 2, 4, 7, 8)	[0.9322709163346613, 0.948, 0.932, 0.944, 0.932]	0.937654
6	(0, 2, 4, 6, 7, 8)	[0.9442231075697212, 0.948, 0.948, 0.96, 0.948]	0.949645
7	(0, 2, 4, 6, 7, 8, 9)	[0.952191235059761, 0.956, 0.968, 0.96, 0.96]	0.959238
8	(0, 2, 4, 5, 6, 7, 8, 9)	[0.9681274900398407, 0.964, 0.964, 0.968, 0.96]	0.964825
9	(0, 2, 3, 4, 5, 6, 7, 8, 9)	[0.9641434262948207, 0.972, 0.96, 0.968, 0.964]	0.965629
10	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)	[0.9760956175298805, 0.972, 0.964, 0.972, 0.96]	0.968819

forward elimination based SVM estimator

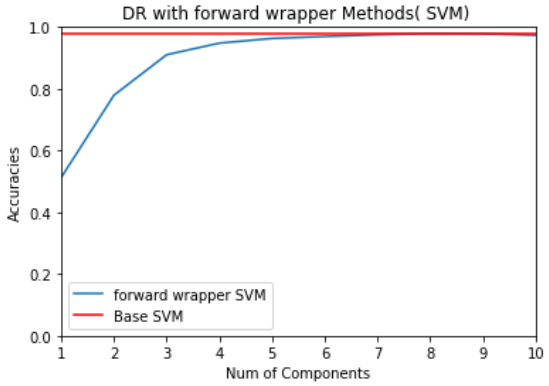
avg_score represent the cross validation score for each numbers of features

	feature_names	cv_scores	avg_score
1	(2,)	[0.5219123505976095, 0.516, 0.544, 0.492, 0.492]	0.513182
2	(2, 4)	[0.7888446215139442, 0.788, 0.784, 0.76, 0.776]	0.779369
3	(2, 4, 8)	[0.900398406374502, 0.912, 0.904, 0.924, 0.912]	0.91048
4	(2, 4, 7, 8)	[0.9442231075697212, 0.964, 0.936, 0.96, 0.936]	0.948045
5	(2, 4, 6, 7, 8)	[0.9641434262948207, 0.968, 0.96, 0.98, 0.944]	0.963229
6	(0, 2, 4, 6, 7, 8)	[0.9760956175298805, 0.976, 0.956, 0.98, 0.96]	0.969619
7	(0, 2, 4, 5, 6, 7, 8)	[0.9760956175298805, 0.984, 0.96, 0.988, 0.968]	0.975219
8	(0, 2, 4, 5, 6, 7, 8, 9)	[0.9880478087649402, 0.984, 0.964, 0.988, 0.972]	0.97921
9	(0, 1, 2, 4, 5, 6, 7, 8, 9)	[0.9920318725099602, 0.984, 0.964, 0.988, 0.964]	0.978406
10	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)	[0.9840637450199203, 0.984, 0.956, 0.98, 0.964]	0.973613

```
Maximum  GBN_LDA accuracy is :  0.9688191235059762
and it had achieved using :  10  components

The Improved Base accuracy of GNB is :  0.9616613418530351
```

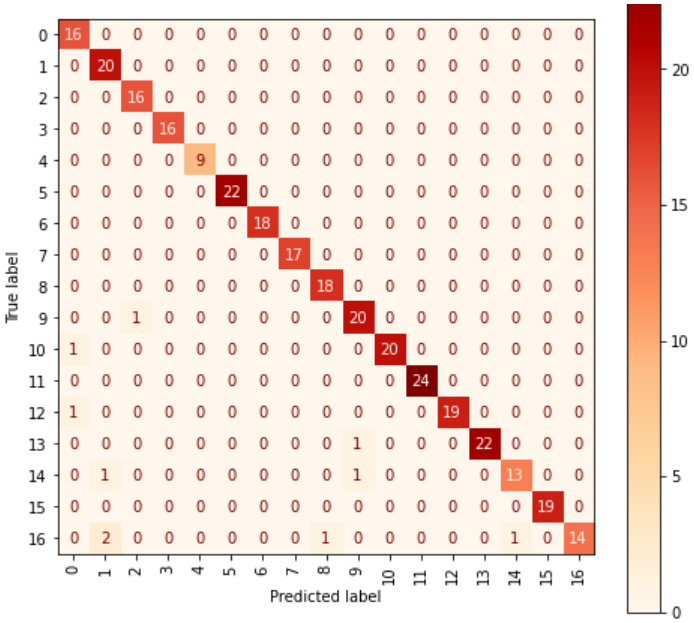


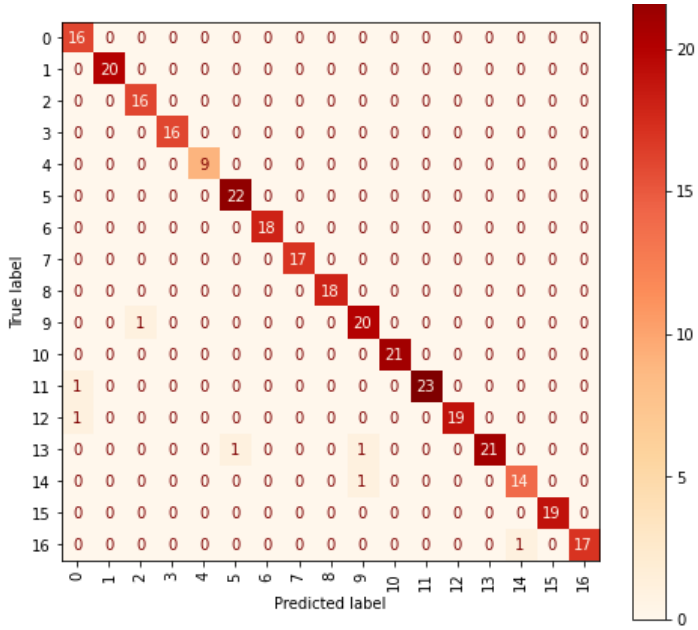

```
Maximum  SVM_LDA accuracy is :  0.9792095617529881
and it had achieved using :  8  components

The Improved Base accuracy of SVM is :  0.9776357827476039
```




```
testing accuracy of GNB using LDA with 10 Components  :   0.968

              precision    recall  f1-score   support

           0       0.89      1.00      0.94        16
           1       0.87      1.00      0.93        20
           2       0.94      1.00      0.97        16
           3       1.00      1.00      1.00        16
           4       1.00      1.00      1.00         9
           5       1.00      1.00      1.00        22
           6       1.00      1.00      1.00        18
           7       1.00      1.00      1.00        17
           8       0.95      1.00      0.97        18
           9       0.91      0.95      0.93        21
          10       1.00      0.95      0.98        21
          11       1.00      1.00      1.00        24
          12       1.00      0.95      0.97        20
          13       1.00      0.96      0.98        23
          14       0.93      0.87      0.90        15
          15       1.00      1.00      1.00        19
          16       1.00      0.78      0.88        18

    accuracy                           0.97       313
   macro avg       0.97      0.97      0.97       313
weighted avg       0.97      0.97      0.97       313
```

```
testing accuracy :   0.978

              precision    recall  f1-score   support

           0       0.89      1.00      0.94        16
           1       1.00      1.00      1.00        20
           2       0.94      1.00      0.97        16
           3       1.00      1.00      1.00        16
           4       1.00      1.00      1.00         9
           5       0.96      1.00      0.98        22
           6       1.00      1.00      1.00        18
           7       1.00      1.00      1.00        17
           8       1.00      1.00      1.00        18
           9       0.91      0.95      0.93        21
          10       1.00      1.00      1.00        21
          11       1.00      0.96      0.98        24
          12       1.00      0.95      0.97        20
          13       1.00      0.91      0.95        23
          14       0.93      0.93      0.93        15
          15       1.00      1.00      1.00        19
          16       1.00      0.94      0.97        18

    accuracy                           0.98       313
   macro avg       0.98      0.98      0.98       313
weighted avg       0.98      0.98      0.98       313
```



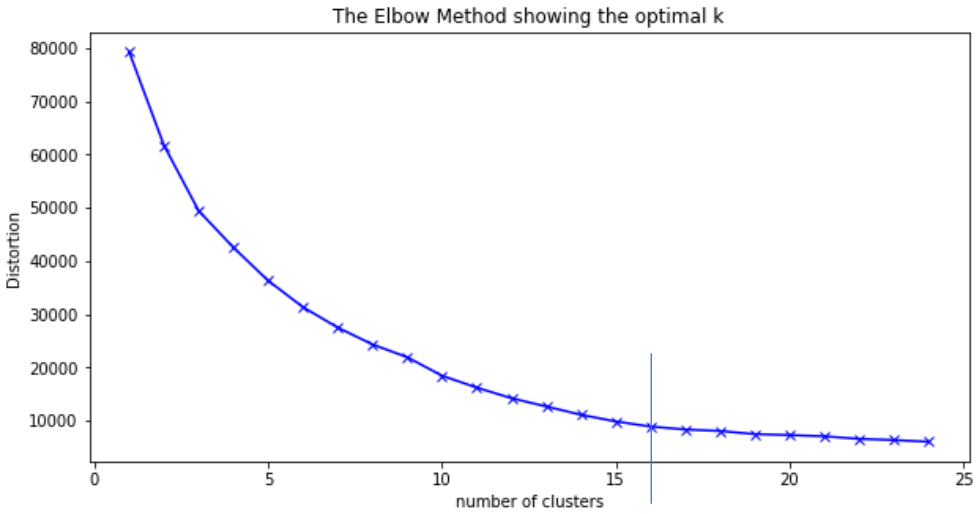


6. Choose the best number of clusters for k-means clustering algorithm on the processed data (using the best features or dimensionality from Q4 and Q5)

based on LDA dimensionality reduction with 10 components and feature selection with forward wrapper (select the most importance components " 8 components") we get best accuracy with SVM 0.978, so we will choose these component (0, 2, 4, 5, 6, 7, 8, 9) to build our K-Means clustering algorithm.

```
#using the best features or dimensionality from Q4 and Q5 ()
forward_train_2_best = lda_results_tr[:,com_8]
forward_test_2_best = lda_results_ts[:,com_8]
```

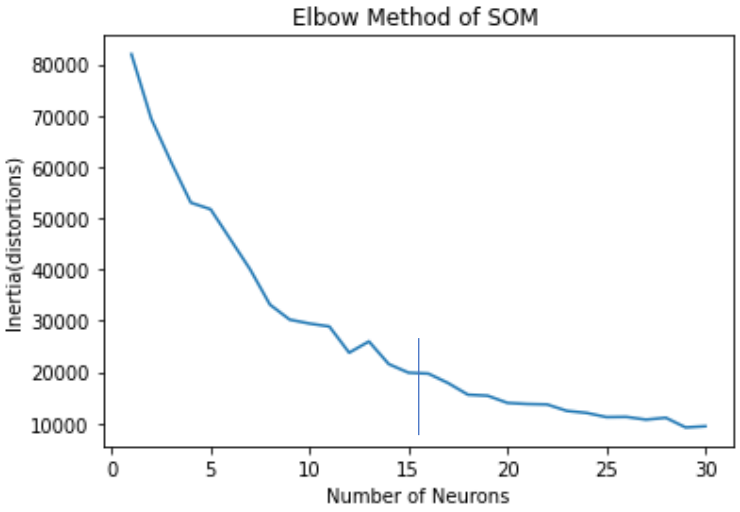
```
distortions2 = []
for k in range(1,25):
    km_model2 = KMeans(n_clusters=k)
    km_model2.fit(forward_train_2_best)
    distortions2.append(km_model2.inertia_)

plt.figure(figsize=(10,5))
plt.plot(range(1,25), distortions2, 'bx-')
plt.xlabel('number of clusters')
plt.ylabel('Distortion')
plt.title('The Elbow Method showing the optimal k')
plt.show();
```



From the above figure the optimal number of clusters is **16**

7. Choose the best number of neurons for SOM algorithm using the best features or dimensionality from Q4 and Q5

7.1 Using the elbow rule, plot the distortion score (a.k.a inertia) vs the number of neurons (max 30 neurons)

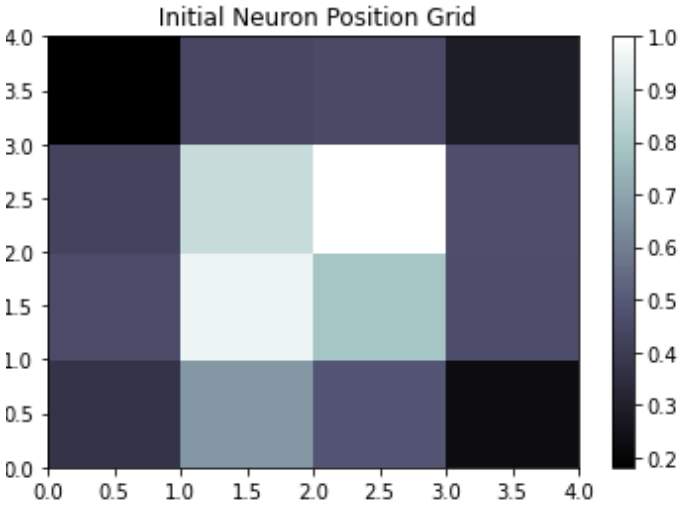


7.2 the optimal number of neurons for SOM equal 16

7.3 Plot the initial and final Neuron positions

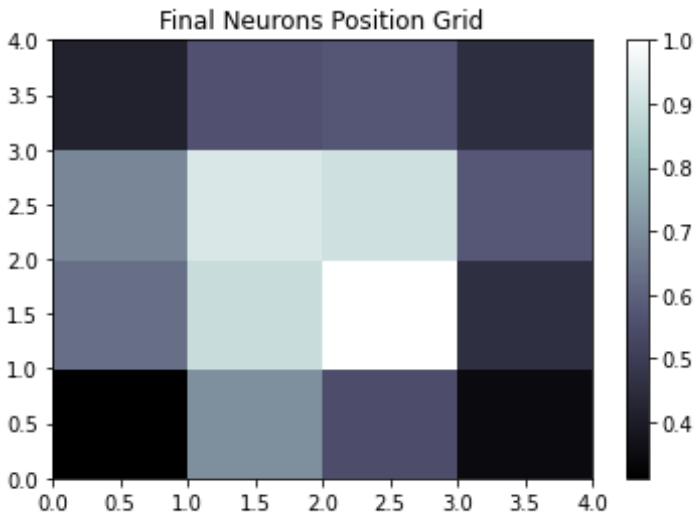
Plotting the initial position of neurons with the initial random weights for neurons

```
[[0.31092682 0.59737286 0.5892029  0.4675928 ]
 [0.60737088 0.80603229 0.90022851 0.65307211]
 [0.65522432 0.90104601 1.         0.72705571]
 [0.37509916 0.78271736 0.79801671 0.51864805]]
```



Plotting the final position of neurons with the final weights for neurons after training SOM model

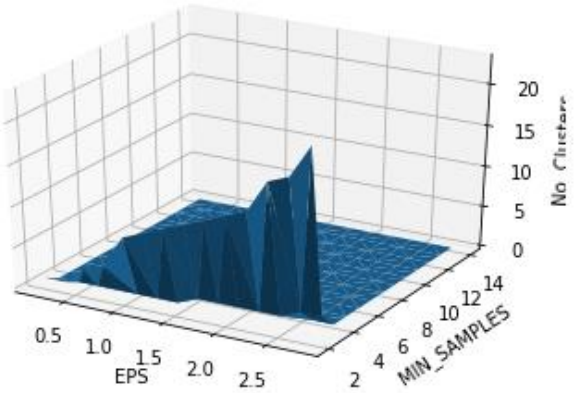
```
[[0.31092682 0.59737286 0.5892029  0.4675928 ]
 [0.60737088 0.80603229 0.90022851 0.65307211]
 [0.65522432 0.90104601 1.         0.72705571]
 [0.37509916 0.78271736 0.79801671 0.51864805]]
```



8. Tune the epsilon (0.2-3) and minpoints (2-15) values in the given intervals to obtain same number of clusters in Q6 by using DBSCAN. Plot the epsilon and minpoints values using a 3D figure.

8.1 Using Original Training Data

```
100%|██████████| 14/14 [00:06<00:00,  2.10it/s]
```



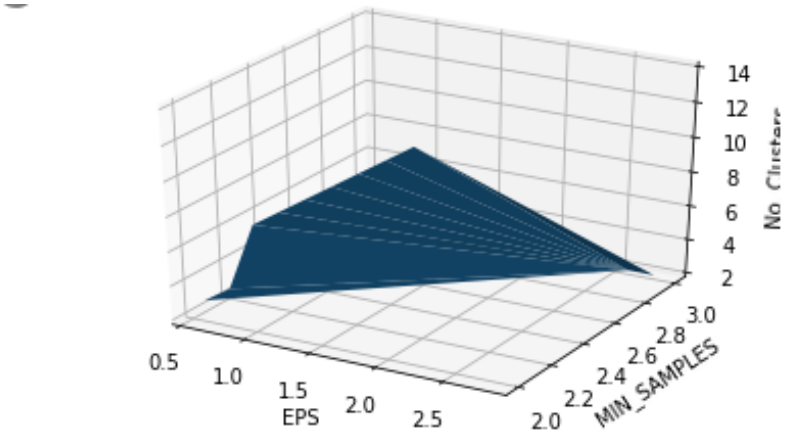

```
2.8000000000000003 2 0.09992006394884093 23 1203
```

EPS = 2.8, min_samples = 2, Accuracy = 0.09, no. clusters = 23, noise = 1203

10 combination of epsilon and minpoints that brings you closer to the desired cluster number: -

Our desired cluster number is 16, so we will select a range of cluster numbers which close to the desired number (16), so we will create a data frame with lists of eps, min_samples, accuracy, noise and number of clusters as shown below, that contains the 10 combinations of parameters which brings us closer to the desired cluster number.

	eps	min_samples	accuracy	cluster	noise
26	0.6	2	0.071143	3	1245
39	0.8	2	0.072742	4	1243
52	1.0	2	0.077538	8	1235
65	1.2	2	0.079137	9	1233
78	1.4	2	0.079137	10	1231
91	1.6	2	0.081535	11	1228
104	1.8	2	0.083133	12	1226
117	2.0	2	0.084732	13	1224
130	2.2	2	0.086331	14	1222
157	2.6	3	0.071143	2	1245

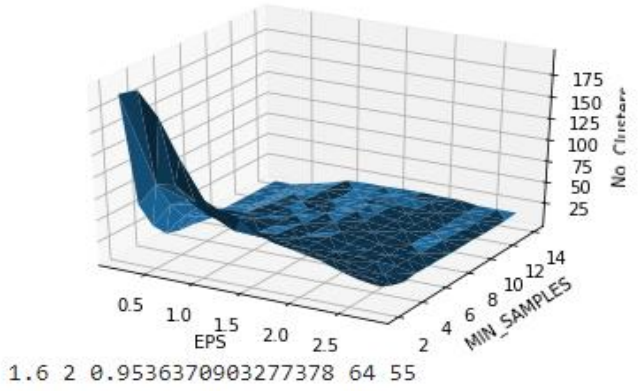



```
2.2 2 0.08633093525179857 14 1222
```

EPS = 2.2, min_samples = 2, Accuracy = 0.086, no. clusters = 14, noise = 1222

8.2 Using LDA (Wrapper with 8 Components)

```
100%|██████████| 14/14 [00:03<00:00,  3.73it/s]
```



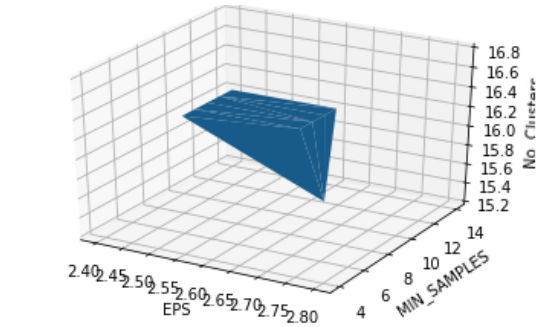

```
1.6 2 0.9536370903277378 64 55
```

EPS = 1.6, min_samples = 2, Accuracy = 0.953, no. clusters = 64, noise = 55

10 combination of epsilon and minpoints that brings you closer to the desired cluster number: -

Our desired cluster number is 16, so we will select all of cluster numbers which qual to the desired number (16), so we will create a data frame with lists of eps, min_samples, accuracy, noise and number of clusters as shown below, that contains the 10 combinations of parameters which brings us closer to the desired cluster number.

	eps	min_samples	accuracy	cluster	noise
151	2.4	10	0.864908	16	144
152	2.4	11	0.863309	16	148
153	2.4	12	0.850520	16	164
154	2.4	13	0.850520	16	165
155	2.4	14	0.847322	16	174
165	2.6	11	0.868905	16	134
166	2.6	12	0.869704	16	135
167	2.6	13	0.860911	16	146
168	2.6	14	0.860911	16	146
171	2.8	4	0.692246	16	48




```
2.6000000000000005 12 0.8697042366107114 16 135
```

EPS = 2.6, min_samples = 12, Accuracy = 0.869, no. clusters = 16, noise = 135