CS6301 MACHINE LEARNING LAB WEEK 10 – DIMENSIONALITY REDUCTION

SRIHARI. S – 2018103601

Date: 19-04-2021 Monday

Aim: To implement Principal Component Analysis and Linear Discriminant analysis and visualize datasets with a large number of features.

Principal Component Analysis:

PCA is a dimensionality-reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set.It reduces the number of variables of a data set, while preserving as much information as possible.

Algorithm:

• Write N datapoints xi = (x1i, x2i, . . . , xMi) as row vectors

• Put these vectors into a matrix X (which will have size N ×M)

• Centre the data by subtracting off the mean of each column, putting it into matrix B

• Compute the covariance matrix $C = 1/N\ B^T B$

• Compute the eigenvalues and eigenvectors of C, so $V^{-1} CV = D$, where V holds the eigenvectors of C and D is the M ×M diagonal eigenvalue matrix.

• Sort the columns of D into order of decreasing eigenvalues, and apply the same order to the columns of V

• Reject those with eigenvalue less than some n (eta), leaving L dimensions in the data

Linear Discriminant Analysis:

It is a dimensionality reduction technique. It is used as a pre-processing step in Machine Learning and applications of pattern classification. The goal of LDA is to project the features in higher dimensional space onto a lowerdimensional space in order to avoid the curse of dimensionality and also reduce resources and dimensional costs. LDA is a supervised classification technique that is considered a part of crafting competitive machine learning models. This

category of dimensionality reduction is used in areas like image recognition and predictive analysis in marketing.

Algorithm:

LDA focuses primarily on projecting the features in higher dimension space to lower dimensions. You can achieve this in three steps:

- Firstly, you need to calculate the separability between classes which is the distance between the mean of different classes. This is called the between-class variance.
- Secondly, calculate the distance between the mean and sample of each class. It is also called the within-class variance.
- Finally, construct the lower-dimensional space which maximizes the between-class variance and minimizes the within-class variance. P is considered as the lowerdimensional space projection, also called Fisher's criterion.

Implementation of PCA:

Url: https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(diagnostic)

Dataset Description: Breast cancer wisconsin (diagnostic) dataset

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809

5 rows × 30 columns

Data Set Characteristics:

• Number of Instances: 569

• Number of Attributes: 30 numeric, predictive attributes and the class

• Attribute Information:

- radius (mean of distances from center to points on the perimeter)

- texture (standard deviation of gray-scale values)

- smoothness (local variation in radius lengths)

- compactness (perimeter^2 / area - 1.0)

- concavity (severity of concave portions of the contour)

- concave points (number of concave portions of the contour)

- symmetry

-fractal dimension ("coastline approximation" - 1)

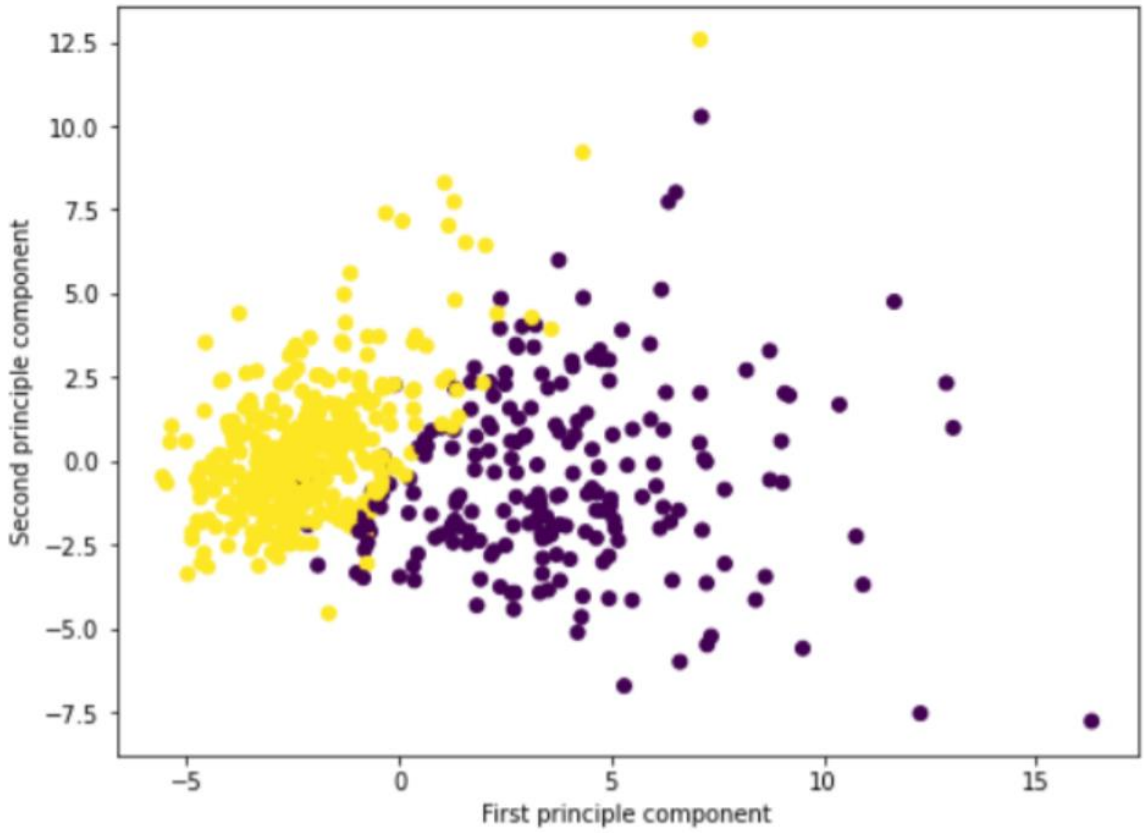
• The mean, standard error, and "worst" or largest (mean of the threeworst/largest values) of these features were computed for each image, resulting in 30 features. For instance, field 0 is Mean Radius, field10 is Radius SE, field 20 is Worst Radius.

• Class

- WDBC-Malignant

- WDBC-Benign

OUTPUT:



```
Train accuracy  95.72 %
Test accuracy  92.98 %

              precision    recall  f1-score   support

           0       0.89      0.92      0.91        64
           1       0.95      0.93      0.94       107

    accuracy                           0.93       171
   macro avg       0.92      0.93      0.93       171
weighted avg       0.93      0.93      0.93       171
```

CODE:

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
%matplotlib inline
from sklearn.datasets import load_breast_cancer
cancer=load_breast_cancer()
X=cancer.data
y=cancer.target
df=pd.DataFrame(cancer['data'],columns=cancer['feature_names'])
df.head()
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scaler.fit(df)
scaled_data=scaler.transform(df)
from sklearn.decomposition import PCA
pca=PCA(n_components=2)
pca.fit(scaled_data)
x_pca=pca.transform(scaled_data)
scaled_data.shape
x_pca.shape
scaled_data
x_pca
plt.figure(figsize=(8,6))
```

```
plt.scatter(x_pca[:,0],x_pca[:,1],c=cancer['target'])
plt.xlabel('First principle component')
plt.ylabel('Second principle component')
X_train_new, X_test_new, y_train, y_test = train_test_split(x_pca, y, test_size =
0.3,
random_state=20, stratify=y)
knn_pca = KNeighborsClassifier(7)
knn_pca.fit(X_train_new,y_train)
print("Train Accuracy ",knn_pca.score(X_train_new,y_train)*100,"%")
print("Test Accuracy ",knn_pca.score(X_test_new,y_test) *100,"%")
```

Implementation Of LDA:

Dataset Used: bmd.csv (bone mineral density)

	age	weight_kg	height_cm	bmd	fracture
0	57.052768	64.0	155.5	0.8793	0
1	75.741225	78.0	162.0	0.7946	0
2	70.778900	73.0	170.5	0.9067	0
3	78.247175	60.0	148.0	0.7112	0
4	54.191877	55.0	161.0	0.7909	0
...	...	...	...	...	...
164	77.982543	74.0	164.0	0.7941	1
165	50.285303	59.0	161.0	0.7971	1
166	46.359721	67.0	169.0	0.8037	1
167	54.788368	70.0	166.0	0.8072	1
168	69.994822	68.5	165.0	0.8664	1

The file bmd.csv contains 169 records of bone densitometries (measurement of bone mineral density). The following variables were collected:

- id – patient's number
- age – patient's age
- fracture – hip fracture (fracture / no fracture)
- weight_kg – weight measured in Kg

• height_cm – height measure in cm

• waiting_time – time the patient had to wait for the densitometry (in minutes)

• bmd – bone mineral density measure in the hip

```
#Import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
#Load dataset
n_components = 2
data = pd.read_csv('bmd.csv')
data = data[['age','weight_kg','height_cm','bmd','fracture']]
data
#Normalizing the attributes and encoding labels
from sklearn.preprocessing import StandardScaler
stdsc = StandardScaler()
X_train_std = stdsc.fit_transform(data.iloc[:,range(0,4)].values)
from sklearn.preprocessing import LabelEncoder
class_le = LabelEncoder()
y = class_le.fit_transform(data['fracture'].values)
#Between class variance
S_W = np.zeros((4,4))
for i in range(2):
 S_W += np.cov(X_train_std[y==i].T)
S_W
#Distance between mean and sample of class
N=np.bincount(y)
vecs=[]
[vecs.append(np.mean(X_train_std[y==i],axis=0)) for i in range(2)]
mean_overall = np.mean(X_train_std, axis=0)
S_B=np.zeros((4,4))
for i in range(2):
 S_B += N[i]*(((vecs[i]-mean_overall).reshape(4,1)).dot(((vecs[i]-
mean_overall).reshape(1,4))))
S_B
```

```
#Display eigen values
eigen_vals, eigen_vecs = np.linalg.eig(np.linalg.inv(S_W).dot(S_B))
eigen_pairs = [(np.abs(eigen_vals[i]), eigen_vecs[:,i]) for i in
range(len(eigen_vals))]
eigen_pairs = sorted(eigen_pairs,key=lambda k: k[0], reverse=True)
print('Eigenvalues in decreasing order:\n')
for eigen_val in eigen_pairs:
 print(eigen_val[0])
#Finding LD1 & LD2
tot = sum(eigen_vals.real)
discr = [(i / tot) for i in sorted(eigen_vals.real, reverse=True)]
cum_discr = np.cumsum(discr)
W=np.hstack((eigen_pairs[0][1][:,].reshape(4,1),eigen_pairs[1][1][:,
].reshape(4,1))).real
X_train_lda = X_train_std.dot(W)
#Adding LD1 & LD2 Value to dataframe
data=pd.DataFrame(X_train_lda)
data['class']=y
data.columns=["LD1","LD2","class"]
data.head()
#Visualizing the data after LDA
import seaborn as sns
markers = ['x','o']
sns.lmplot(x="LD1", y="LD2", data=data, markers=markers,fit_reg=False,
hue='class',
legend=False, palette='rainbow')
plt.legend(loc='upper center')
plt.show()
#KNN Classifier
X_train, X_test, y_train, y_test = train_test_split(data[['LD1','LD2']], data['class'],
test_size=0.20)
knn= KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
print("Train accuracy ",knn.score(X_train,y_train)*100,"%")
print("Test accuracy ",knn.score(X_test,y_test)*100,”%”)
```

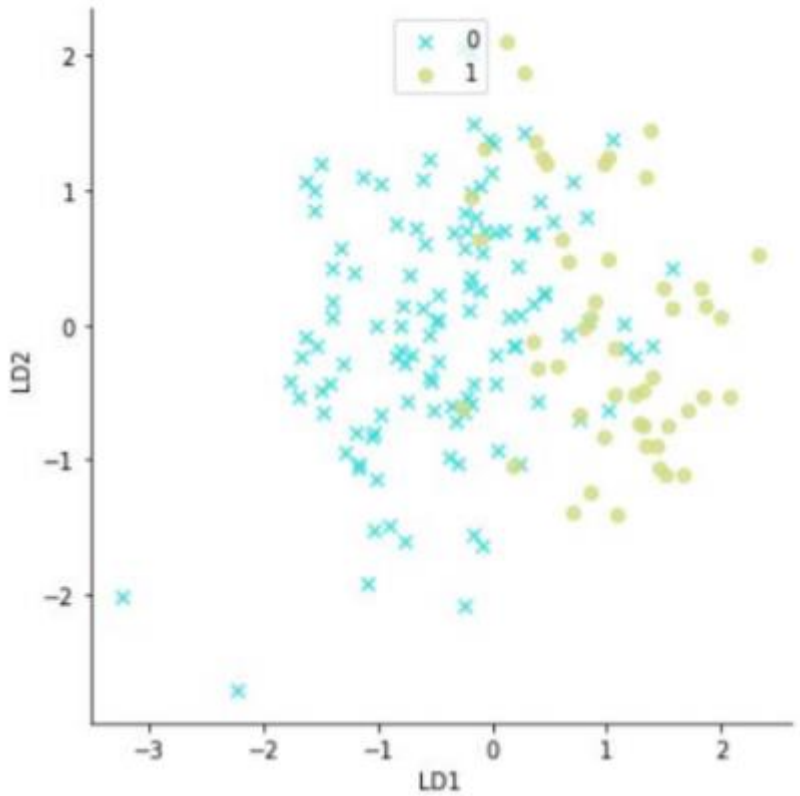
OUTPUT

Between Class Variance

```
array([[ 1.9725685 , -0.04759746, -0.42736814, -0.30524193],
       [-0.04759746,  1.78794962,  0.71452081,  0.59460869],
       [-0.42736814,  0.71452081,  2.03028906,  0.56144434],
       [-0.30524193,  0.59460869,  0.56144434,  1.0964439 ]])
```

Within class variance

```
array([[ 17.63488395, -19.24443281,  -4.31743302, -34.05701777],
       [-19.24443281,  21.00088638,   4.71148831,  37.16542689],
       [ -4.31743302,   4.71148831,   1.05700882,   8.33795637],
       [-34.05701777,  37.16542689,   8.33795637,  65.77193605]])
```




```
Train accuracy  92.59 %
Test accuracy  85.29 %
```

Implementation of PCA and LDA on a dataset having more than 100 columns:

Dataset: MNIST Dataset

The MNIST database is a large database of handwritten digits that is commonly used for training various image processing systems which contains 60,000 training images and 10,000 testing images. Half of the training set and half of the test set were taken from NIST's training dataset, while the other half of the training set and the other half of the test set were taken from NIST's testing dataset.

Input:

train - Notepad

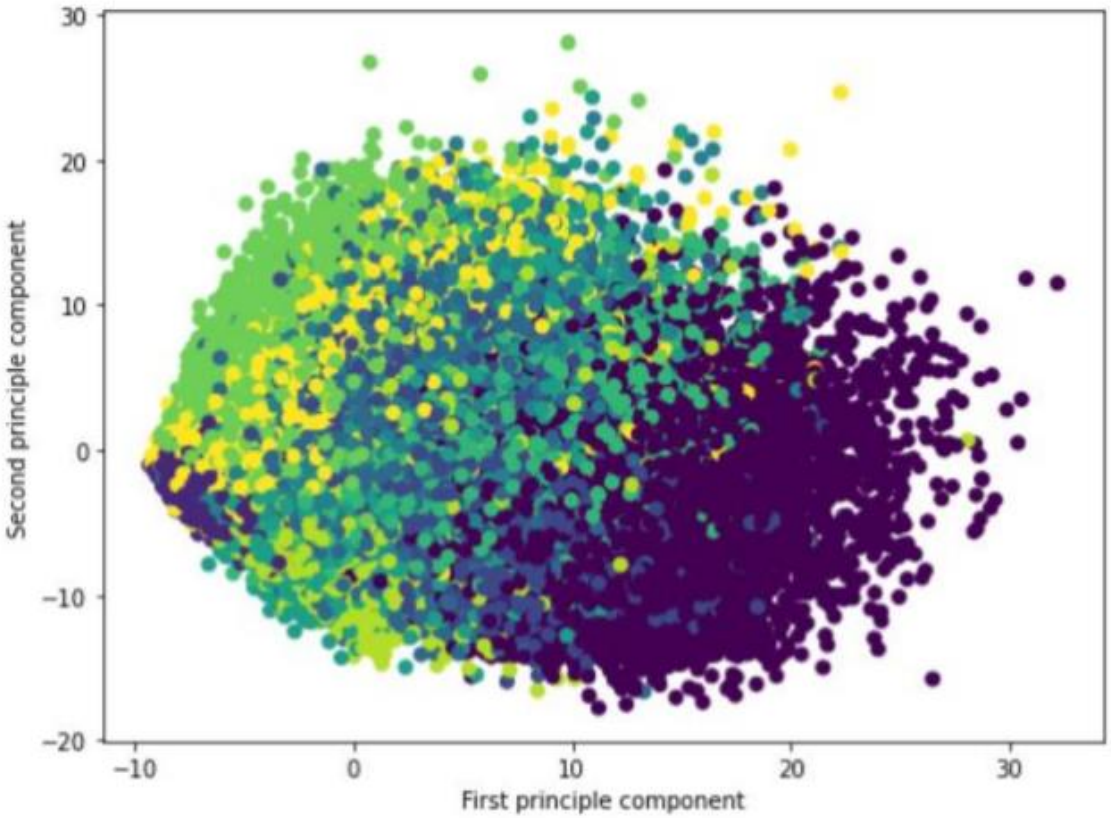
File Edit Format View Help

```
[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   3  18  18  18 126 136 175  26 166 255
 247 127   0   0   0   0   0   0   0   0   0   0   0   0  30  36  94 154
 170 253 253 253 253 253 225 172 253 242 195  64   0   0   0   0   0   0
   0   0   0   0   0  49 238 253 253 253 253 253 253 253 253 251  93  82
  82  56  39   0   0   0   0   0   0   0   0   0   0   0   0  18 219 253
 253 253 253 253 198 182 247 241   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0  80 156 107 253 253 205  11   0  43 154
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0  14   1 154 253  90   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0 139 253 190   2   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0  11 190 253  70   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  35 241
 225 160 108   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0  81 240 253 253 119  25   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0  45 186 253 253 150  27   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0  16  93 252 253 187
```

PCA USING MNIST:

```
#Import Library
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
```

```
%matplotlib inline
#Import dataset
data = pd.read_csv("mnist_train.csv")
X = data.drop('label', axis=1)
y = data.label
data.head()
#Normalizing Dataset using StandardScaler
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaled_data=scaler.fit_transform(X)
scaled_data
#Initializing PCA
from sklearn.decomposition import PCA
pca=PCA(n_components=2)
pca.fit(scaled_data)
x_pca=pca.transform(scaled_data)
scaled_data.shape
x_pca.shape
scaled_data
x_pca
#Printing transformed data
#Visualizing data after applying PCA
plt.figure(figsize=(8,6))
plt.scatter(x_pca[:,0],x_pca[:,1],c=cancer['target'])
plt.xlabel('First principle component')
plt.ylabel('Second principle component')
#KNN Classifier
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3,
random_state=20,
stratify=y)
knn = KNeighborsClassifier(3)
knn.fit(X_train,y_train)
print("Train Accuracy",knn.score(X_train,y_train)*100,"%")
print("Test Accuracy ",knn.score(X_test,y_test) *100,"%")
```



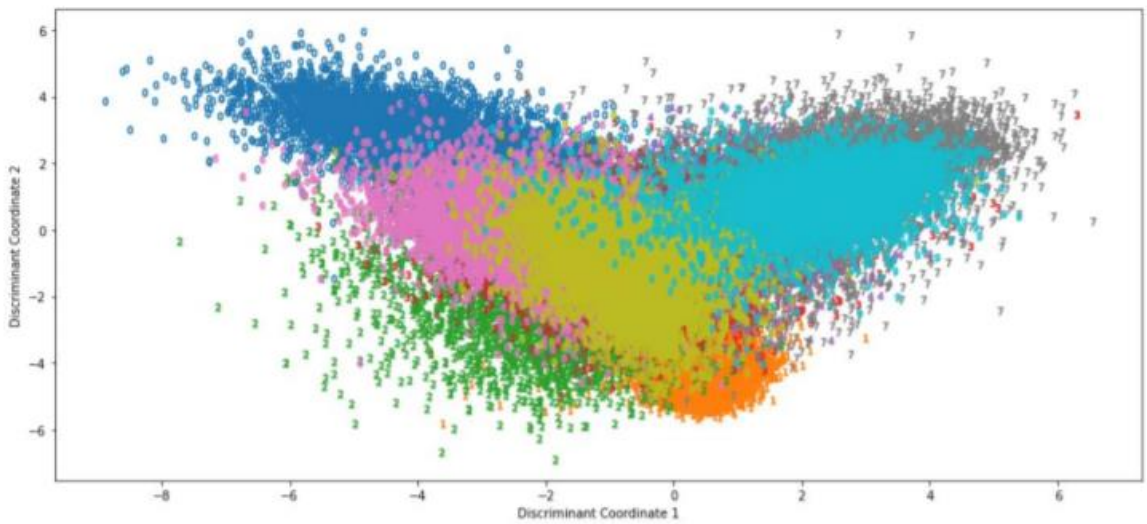

```
Train accuracy  98.5 %
Test accuracy  96.95 %
```

LDA USING MNIST

```
#Import library
import matplotlib.pyplot as plt
from sklearn import datasets, svm, metrics
import pandas as pd
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
%matplotlib inline
#Loading data
data = pd.read_csv("mnist_train.csv")
X = data.drop('label', axis=1)
y = data.label
```

```
target_names = data.label
data.head()
# Create a classifier: a Fisher's LDA classifier
lda = LinearDiscriminantAnalysis(n_components=2, solver='eigen',
shrinkage=0.1)
lda = lda.fit(X, y)
X_r_lda = lda.transform(X)
# Visualize transformed data on learnt discriminant coordinates
plt.figure(figsize=[13,6])
for i, target_name in zip([0,1,2,3,4,5,6,7,8,9], target_names):
 plt.scatter(X_r_lda[y == i, 0], X_r_lda[y == i, 1], alpha=.8,label=target_name,
marker='$%.f$'%i)
plt.xlabel('Discriminant Coordinate 1')
plt.ylabel('Discriminant Coordinate 2')
plt.tight_layout()
#KNN Classifier
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3,
random_state=20,
stratify=y)
knn = KNeighborsClassifier(7)
knn.fit(X_train,y_train)
print("Train score ",knn.score(X_train,y_train),"%")
print("Test score ",knn.score(X_test,y_test),"%")
```

	precision	recall	f1-score	support
0	0.97	0.96	0.97	5335
1	0.96	0.98	0.97	6047
2	0.83	0.94	0.88	5352
3	0.92	0.89	0.91	5514
4	0.94	0.92	0.93	5266
5	0.92	0.90	0.91	4875
6	0.95	0.95	0.95	5331
7	0.93	0.92	0.93	5659
8	0.93	0.89	0.91	5264
9	0.92	0.91	0.91	5357




```
Train accuracy  97.63 %
Test accuracy  96.68 %
```

TABULAR INFERENCE

LINEAR DISCRIMINANT ANALYSIS

DATASET	Training Accuracy	Testing Accuracy
MNIST	97.63%	96.68%
BONE MINERAL DENSITY	92.59%	85.29%

PRINCIPAL COMPONENT ANALYSIS

DATASET	Training Accuracy	Testing Accuracy
MNIST	98.5%	96.95%
BREAST CANCER DATASET	95.72%	92.98%

DATASET	Precision	Recall	F1-Score
MNIST	0.95	0.94	0.94
BREAST CANCER DATASET	0.92	0.93	0.93

Inference: Thus, PCA and LDA are implemented and the performance metrics are recorded. The higher dimensional data has been visualized on a lower dimension.