Linear Discriminant Analysis

Name - Amith Korada

Course - AI & ML
(Batch - 4)

Duration - 12 Months

Problem Statement - Building a Machine Learning model to extract features of the face using PCA.

Prerequisites -

What things you need to install the software and how to install them:

Python 3.6 This setup requires that your machine has the latest version of python. The following URL https://www.python.org/downloads/ can be referred to as download python.

The second and easier option is to download anaconda and use its anaconda prompt to run the commands. To install anaconda check this URL https://www.anaconda.com/download/ You will also need to download and install the below 3 packages after you install either python or anaconda from the steps above Sklearn (scikit-learn) numpy scipy if you have chosen to install python 3.6 then run the below commands in command prompt/terminal to install these packages pip install -U sci-kit-learn pip install NumPy pip install scipy if you have chosen to install anaconda then run the below commands in anaconda prompt to install these packages conda install -c sci-kit-learn conda install -c anaconda numpy conda install -c anaconda scipy.

Dataset Used - Created a Dummy dataset

1. Creating a Dummy dataset with 4 features and 2 classes -

```
from sklearn.datasets import make_classification
```

```
X,y = make_classification(n_samples=100, n_features=4,n_classes=2)
```

```
X.shape
```

```
(100, 4)
```

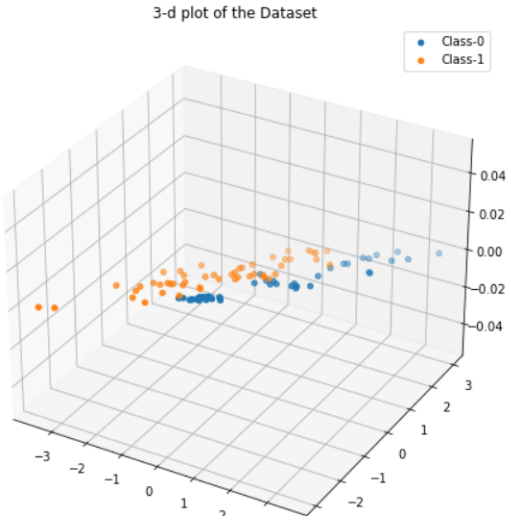
```
import numpy as np
```

```
np.unique(y)
```

```
array([0, 1])
```

2. Visualising the dataset in 3-D -

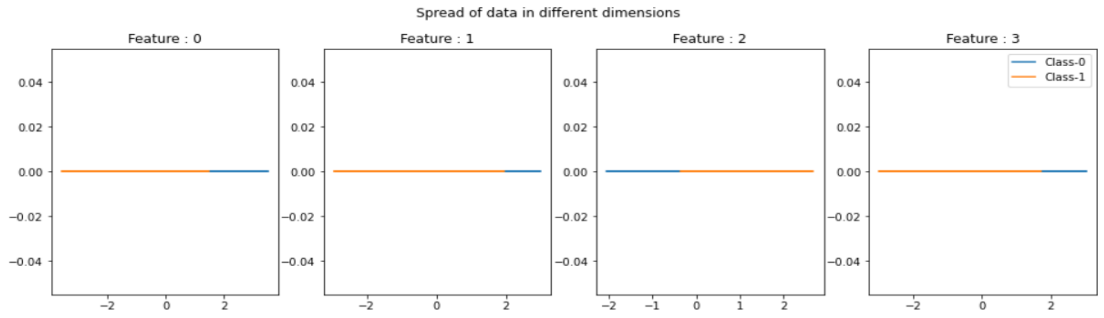
```
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(8,8))
ax = fig.add_subplot(111, projection='3d')
plt.title("3-d plot of the Dataset")
for i, target_name in zip([0, 1], ['Class-0','Class-1']):
    ax.scatter(X[y == i, 0], X[y == i, 1],
               label = target_name)
plt.legend(loc='best')
plt.show()
```



3. Plotting different attributes of data to check which contribute more

```
fig = plt.figure(figsize=(25, 10))
fig.suptitle("\n\nSpread of data in different dimensions" )

# Plot results
for j in range(4):
    ax = fig.add_subplot(2, 6, 2 + j + (j >3))
    for i, target_name in zip([0, 1],  ['Class-0','Class-1']):
        ax.set_title("Feature : %s" %(j))
        ax.plot(X[y == i, j],np.zeros_like(X[y == i, j]),label=target_name)
plt.legend(loc='best')
plt.show()
```



4. LDA to decrease the number of features to 1 and Decision Tree Classifier for classification -

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
lda1 = LDA(n_components=1)
```

```
from sklearn.tree import DecisionTreeClassifier
tree = DecisionTreeClassifier(criterion='entropy')
```

5. Checking Accuracy after reducing data to 1D

```
# Fit the method's model
lda1.fit(X_train, y_train)

# Fit a Decision Tree classifier on the embedded training set
tree.fit(lda1.transform(X_train), y_train)

# Compute the Decision Tree accuracy on the embedded test set
acc = tree.score(lda1.transform(X_test), y_test)
X_1 = lda1.transform(X)
plt.figure(figsize=(8,3))
for i, target_name in zip([0, 1],  ['Class-0','Class-1']):
    plt.scatter(X_1[y == i, 0],np.zeros_like(X[y == i, j]), alpha=.8,
                label=target_name)
plt.legend(loc='best')
plt.title("LDA, Decision Tree\nTest accuracy = {:.2f}".format(acc))
plt.show()
```