Project Report 2

Project Based On LDA - Speech Classification

Name : Dhivakar.R

Course: AI and ML

Problem Statement: Applying LDA technique on iris dataset to decrease number of features

Prerequisites

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

Python 3.6 This setup requires that your machine has latest version of python. The following url https://www.python.org/downloads/ can be referred to download python. Once you have python downloaded and installed, you will need to setup PATH variables (if you want to run python program directly, detail instructions are below in how to run software section). To do that check this: https://www.pythoncentral.io/add-python-to-path-python-is-not- recognized-as-an-internal-or-externalcommand/ . Setting up PATH variable is optional as you can also run program without it and more instruction are given below on this topic.

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 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 below commands in command prompt/terminal to install these packages pip install -U scikit-learn pip install numpy pip install scipy if you have chosen to install anaconda then run below commands in anaconda prompt to install these packages conda install -c scikit-learn conda install -c anaconda numpy conda install -c anaconda scipy

Dataset used:

The data source used is IRIS dataset which is an in-built dataset available in scikit-learn library.

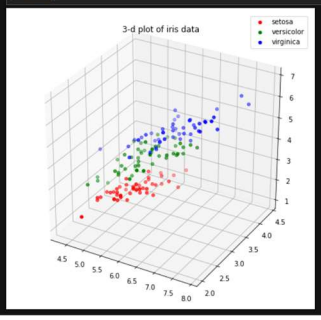
Method used for Process: LDA (Linear Discriminant Analysis)

Importing the libraries and loading dataset :

```
[25]: from sklearn import datasets
      from sklearn.model_selection import train_test_split
      from sklearn.tree import DecisionTreeClassifier
      from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
      from mpl_toolkits.mplot3d.axes3d import get_test_data
      import matplotlib.pyplot as plt
      import numpy as np
```

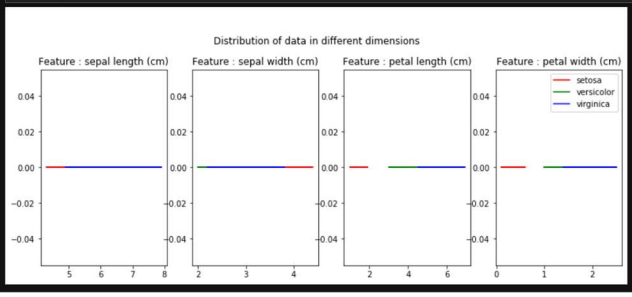
Plotting the iris data

```
[26]: iris = datasets.load_iris()
      X = iris.data
      y = iris.target
      features = iris.feature_names
      target_names = iris.target_names
      colors = ["red", "green", "blue"]

[27]: print(features)
      print(target_names)

      ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
      ['setosa' 'versicolor' 'virginica']

[31]: # Plotting  iris data
      fig = plt.figure(figsize=(8,8))
      ax = fig.add_subplot(111, projection='3d')
      plt.title("3-d plot of iris data")
      for color, i, target_name in zip(colors, [0, 1, 2], target_names):
          ax.scatter(X[y == i, 0], X[y == i, 1], X[y == i, 2], color = color, label = target_name)
      plt.legend()
      plt.show()
```



Checking on different attributes

```
[19]: # checking on different attributes
      fig = plt.figure(figsize=(20,10))
      fig.suptitle("\n\nDistribution of data in different dimensions")
      for j in range(4):
          ax = fig.add_subplot(2, 6, 2*j+(j>3))
          for color, i, target_name in zip(colors, [0, 1, 2], target_names):
              ax.set_title('Feature : %s' %(features[j]))
              ax.plot(X[y == i, j], np.zeros_like(X[y == i, j]), color = color, label = target_name)
      plt.legend()
      plt.show()
```

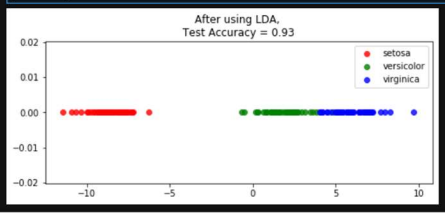


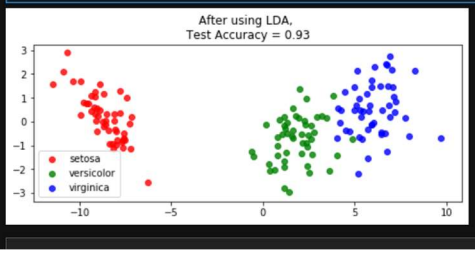
Accuracy without lda

```
[12]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

[13]: tree = DecisionTreeClassifier(criterion = "entropy")
      tree.fit(X_train, y_train)
      accuracy = tree.score(X_test, y_test)
      print("Accuracy without LDA: ", accuracy)

      Accuracy without LDA:  0.9333333333333333
```

Accuracy after using LDA for data in 1D

```
[15]: # Accuracy after using LDA for data in 1D
lda1.fit(X_train, y_train)

tree.fit(lda1.transform(X_train), y_train)

accuracy = tree.score(lda1.transform(X_test), y_test)
X_1 = lda1.transform(X)
plt.figure(figsize=(8,3))
for color, i, target_name in zip(colors, [0, 1, 2], target_names):
    plt.scatter(X_1[y == i, 0], np.zeros_like(X[y == i, j]), alpha= .8, color = color, label = target_name)
plt.legend()
plt.title("After using LDA, \nTest Accuracy = {:.2f}".format(accuracy))
plt.show()
```



Accuracy after using LDA for data in 2D

```
[18]: # Accuracy after using LDA for data in 2D
lda2.fit(X_train, y_train)
tree.fit(lda2.transform(X_train), y_train)
accuracy = tree.score(lda2.transform(X_test), y_test)
X_2 = lda2.transform(X)
plt.figure(figsize=(8,3))
for color, i, target_name in zip(colors, [0, 1, 2], target_names):
    plt.scatter(X_2[y == i, 0], X_2[y == i, 1], alpha= .8, color = color, label = target_name)
plt.legend()
plt.title("After using LDA, \nTest Accuracy = {:.2f}".format(accuracy))
plt.show()
```