CAPSTONE PROJECT REPORT

Name - Paleti Samuel Yashaswi
Course - Machine learning and AI
Duration - 10 months
Question - 2

Speech Classification

In order to implement LDA, first generate a dummy dataset (say IRIS dataset having 4 features) and the use LDA to decrease the number of features to one/two. Now using this modified dataset, try to learn a classifier to test the performance of LDA for dimensionality reduction.

Data Set Used - dataset which is used is imported from the scikit-learn library

Method used is Linear Discriminant analysis

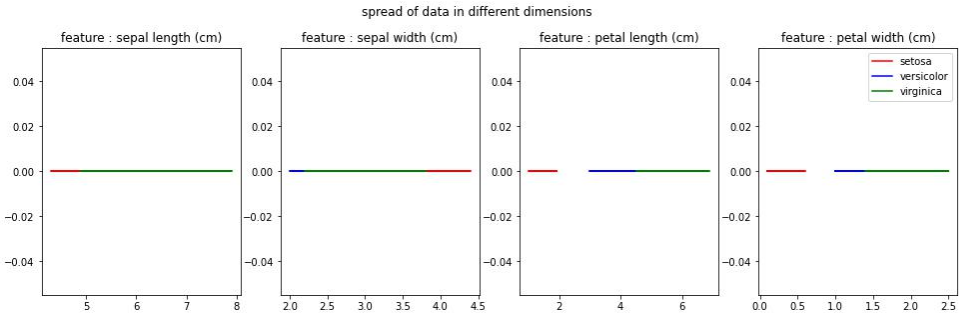
Importing the datasets and plotting with original features

```
import numpy as np
import pandas as pd
from sklearn.metrics import accuracy_score,f1_score
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
from sklearn.model_selection import train_test_split
from sklearn import datasets
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier

data=datasets.load_iris()
X=data.data
y=data.target
print(data.keys())
features=data.feature_names
target_names=data.target_names
print(features)
print(target_names)
colors=['red','blue','green']


fig=plt.figure(figsize=(8,8))
ax=fig.add_subplot(111,projection='3d')
plt.title("3-d plot of the iris dataset")
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_names)
plt.legend(loc='best')
plt.show()
```

```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename'])
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
['setosa' 'versicolor' 'virginica']
```

Checking and plotting the weightage of features

```
fig=plt.figure(figsize=(25,10))
fig.suptitle("\n\nspread 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(loc='best')
plt.show()
```



Training the Data with Decision tree Classifier and reducing into 2 feature data with LDA

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3)


tree= DecisionTreeClassifier(criterion='entropy')

tree.fit(X_train,y_train)
acc=tree.score(X_test,y_test)
print("without LDA",acc)
```

```
without LDA 0.9555555555555556
```

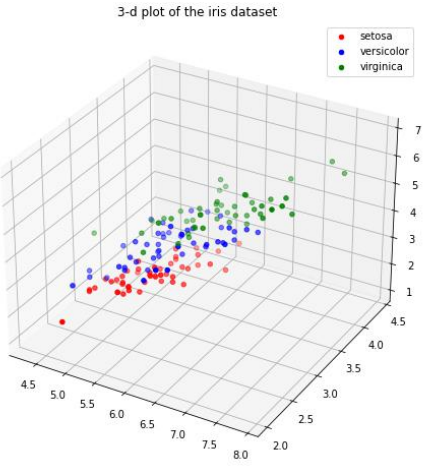
```
lda2=LDA(n_components=2)
lda1=LDA(n_components=1)


lda2.fit(X_train,y_train)

tree.fit(lda2.transform(X_train),y_train)

acc=tree.score(lda2.transform(X_test),y_test)
X_2=lda2.transform(X)
plt.figure(figsize=(12,5))
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(loc='best')
plt.title("LDA, Decision Tree accuracy={:.2f}".format(acc))
plt.show()
```

Reducing the data into one dimension data with LDA

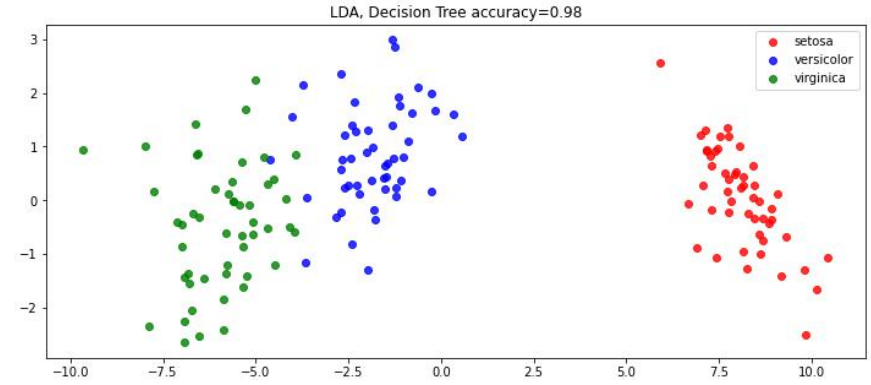
```
lda1.fit(X_train,y_train)

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

acc=tree.score(lda1.transform(X_test),y_test)
X_1=lda1.transform(X)
plt.figure(figsize=(10,4))
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,1]),alpha=.8,color=color,label=target_name)
plt.legend(loc='best')
plt.title("LDA, Decision Tree accuracy={:.2f}".format(acc))
plt.show()
```

OUTPUT

Original Data Plot



2-D feature Data plot



1D Feature plot