Project Report

Speech Classification Using LDA

Name: Rishi Raj Singh
Course: AI and ML
(Batch 4)

- **Problem Statement**

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

- **Prerequisites**

  - Software:
    - Python 3 (Use anaconda as your python distributor as well)

  - Tools:
    - Pandas
    - Numpy
    - Matplotlib
    - Seaborn

  - Dataset: IRIS Dataset

- **Method Used**

Linear Discriminant Analysis or LDA is a dimensionality reduction technique used to reduce the number of dimensions (i.e. variables) in a dataset while retaining as much information as possible. Using LDA based classification, we can find discriminative features for a given audio segment to achieve the task of Automatic Speech Classification such that speech belonging to the same class are close together, but samples from different classes are far apart from each other.

- **Implementation:**

Load all required librarieslibraries and Dataset

```
In [2]:
# Importing the necessary Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn
from sklearn.datasets import load_iris
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
```

```
In [3]:
#Loading the Data
iris = load_iris()
```

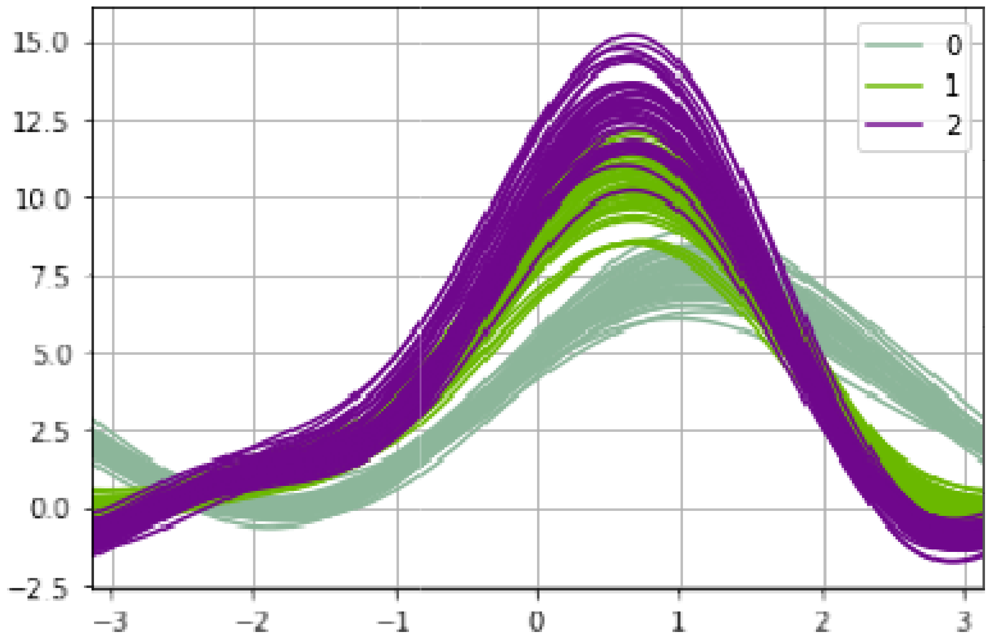
```
In [4]:
df = pd. DataFrame(data= np. c_[iris['data'], iris['target']. astype('int32')],
                    columns= iris['feature_names'] + ['target'])
```

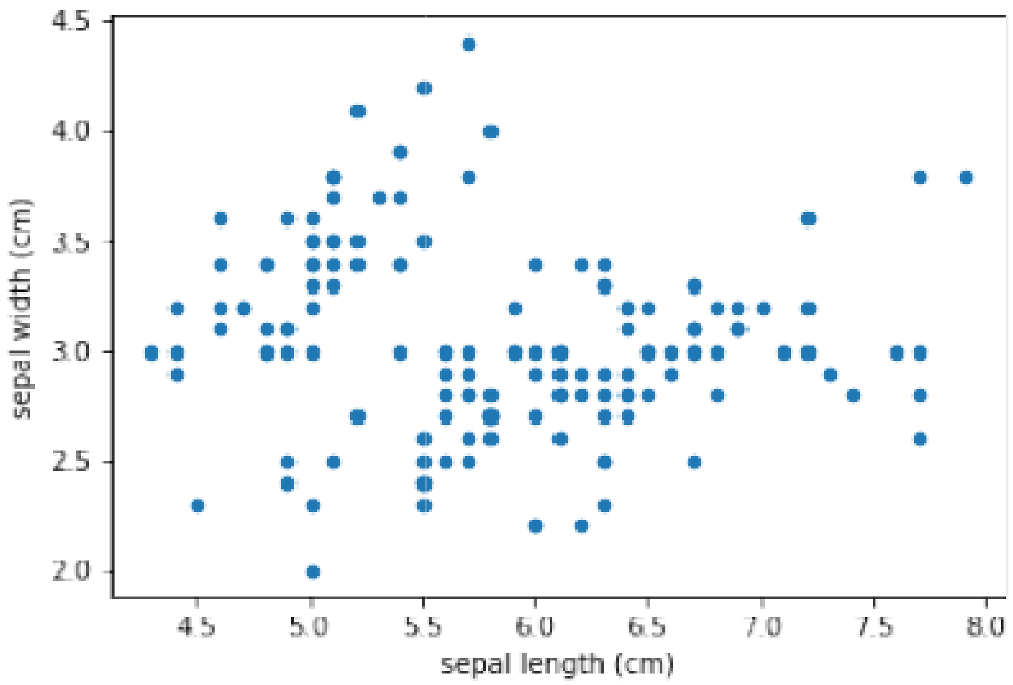
```
In [5]:
df.head()
```

Out[5]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0.0
1	4.9	3.0	1.4	0.2	0.0
2	4.7	3.2	1.3	0.2	0.0
3	4.6	3.1	1.5	0.2	0.0
4	5.0	3.6	1.4	0.2	0.0

Visualization and preprocessingand preprocessing data





Splitting Data and ApplyingApplying LDA

```
In [12]:  from sklearn.model_selection import train_test_split
          X_train,X_test,y_train,y_test = train_test_split(X,y,random_state=0,test_size=0.2)
```

```
In [13]:  LDA = LinearDiscriminantAnalysis()
```

```
In [14]:  LDA.fit_transform(X_train,y_train)
```

Output of LDA

```
In [19]:  print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00         6

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

Applying Logistic Regression and AccuracApplying Logistic Regression and Accuracy

```
In [21]:  from sklearn.linear_model import LogisticRegression
          import warnings
```

```
In [22]:  clf = LogisticRegression()
          clf.fit(X_train,y_train)
          pred1 = clf.predict(X_test)
```

```
In [23]:  pred1

Out[23]:  array([2, 1, 0, 2, 0, 2, 0, 1, 1, 1, 2, 1, 1, 1, 1, 0, 1, 1, 0, 0, 2, 1,
                 0, 0, 2, 0, 0, 1, 1, 0])
```

```
In [24]:  print(accuracy_score(y_test,pred1))

          1.0
```