**LINEAR DISCRIMINANT ANALYSIS**

**CSE 303: Machine Learning**

Submitted by

Name: Guntur Ridhi

Roll No: AP22110011467

Section: CSE M

Lab Date: 22/10/2024

Submission Date: 28/10/2024

**A picture containing text

Description automatically generated**

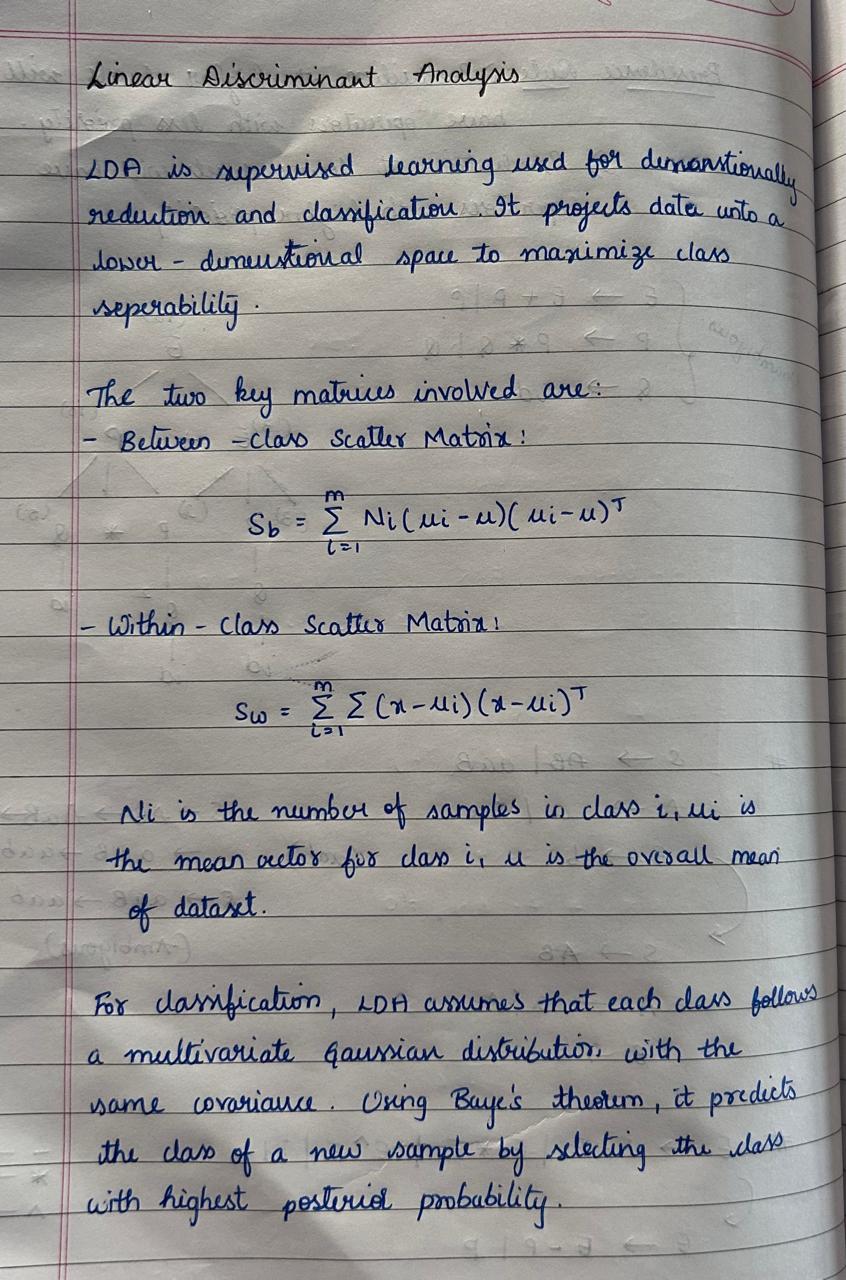
**Department Computer Science and Engineering**

**School of Engineering and Sciences**

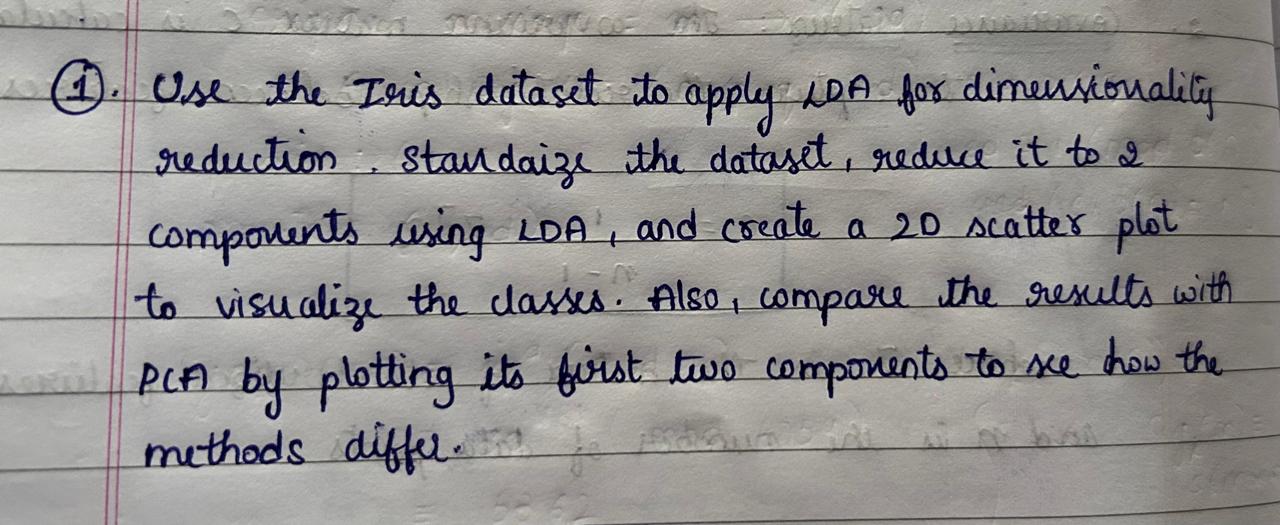
**SRM University–AP**

**Amaravati, Andhra Pradesh – 522 240, India**

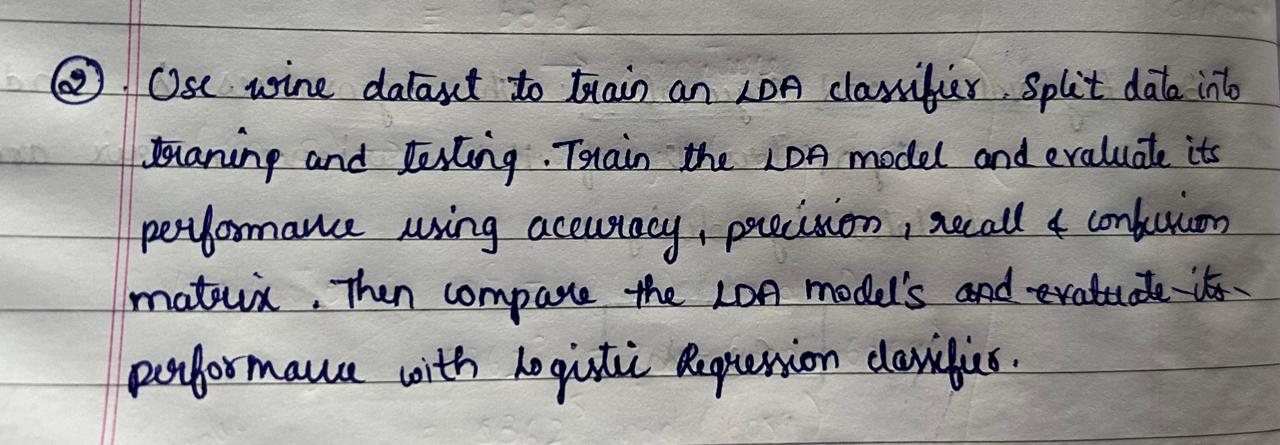
1. **Algorithm Description**



1. **Solution**



|  |
| --- |
| import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  import seaborn as sns  from sklearn.datasets import load\_iris  data = load\_iris()  X = data.data # Feature matrix (4 features)  y = data.target # Target labels (3 classes)  target\_names = data.target\_names  from sklearn.preprocessing import StandardScaler  scaler = StandardScaler()  X\_standardized = scaler.fit\_transform(X)  print("Standardized data:")  print(X\_standardized[:5])  from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis  lda = LinearDiscriminantAnalysis(n\_components=2)  X\_lda = lda.fit\_transform(X\_standardized, y)  print(f"Transformed shape with LDA: {X\_lda.shape}")  plt.figure(figsize=(8, 6))  for i, target in enumerate(np.unique(y)):  plt.scatter(X\_lda[y == target, 0], X\_lda[y == target, 1],  label=target\_names[target], alpha=0.7)  plt.title("2D LDA Projection of Iris Dataset")  plt.xlabel("LD1")  plt.ylabel("LD2")  plt.legend()  plt.show()  from sklearn.decomposition import PCA  pca = PCA(n\_components=2)  X\_pca = pca.fit\_transform(X\_standardized)  plt.figure(figsize=(8, 6))  for i, target in enumerate(np.unique(y)):  plt.scatter(X\_pca[y == target, 0], X\_pca[y == target, 1],  label=target\_names[target], alpha=0.7)  plt.title("2D PCA Projection of Iris Dataset")  plt.xlabel("PC1")  plt.ylabel("PC2")  plt.legend()  plt.show() |



|  |
| --- |
| from sklearn.datasets import load\_wine  data = load\_wine()  X = data.data  y = data.target  print(f"Feature matrix shape: {X.shape}")  print(f"Target vector shape: {y.shape}")  print(f"Classes: {data.target\_names}")  from sklearn.model\_selection import train\_test\_split  X\_train, X\_test, y\_train, y\_test = train\_test\_split(  X, y, test\_size=0.3, random\_state=42, stratify=y  )  print(f"Training set shape: {X\_train.shape}")  print(f"Testing set shape: {X\_test.shape}")  from sklearn.preprocessing import StandardScaler  from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis as LDA  scaler = StandardScaler()  X\_train\_scaled = scaler.fit\_transform(X\_train)  X\_test\_scaled = scaler.transform(X\_test)  lda = LDA()  lda.fit(X\_train\_scaled, y\_train)  y\_pred\_lda = lda.predict(X\_test\_scaled)  from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, confusion\_matrix, classification\_report  accuracy = accuracy\_score(y\_test, y\_pred\_lda)  precision = precision\_score(y\_test, y\_pred\_lda, average='weighted')  recall = recall\_score(y\_test, y\_pred\_lda, average='weighted')  print(f"LDA Accuracy: {accuracy:.2f}")  print(f"LDA Precision: {precision:.2f}")  print(f"LDA Recall: {recall:.2f}")  conf\_matrix = confusion\_matrix(y\_test, y\_pred\_lda)  print("\nConfusion Matrix:\n", conf\_matrix)  print("\nClassification Report:\n", classification\_report(y\_test, y\_pred\_lda))  from sklearn.linear\_model import LogisticRegression  log\_reg = LogisticRegression(max\_iter=10000)  log\_reg.fit(X\_train\_scaled, y\_train)  y\_pred\_log\_reg = log\_reg.predict(X\_test\_scaled)  accuracy\_log = accuracy\_score(y\_test, y\_pred\_log\_reg)  precision\_log = precision\_score(y\_test, y\_pred\_log\_reg, average='weighted')  recall\_log = recall\_score(y\_test, y\_pred\_log\_reg, average='weighted')  print(f"\nLogistic Regression Accuracy: {accuracy\_log:.2f}")  print(f"Logistic Regression Precision: {precision\_log:.2f}")  print(f"Logistic Regression Recall: {recall\_log:.2f}")  conf\_matrix\_log = confusion\_matrix(y\_test, y\_pred\_log\_reg)  print("\nLogistic Regression Confusion Matrix:\n", conf\_matrix\_log) |

1. **Code Repository:**

GitHub Link: <https://github.com/Ridhi-215/LDA>