**LINEAR DISCRIMINANT ANALYSIS(LDA)**

**CSE 303: Machine Learning**

Submitted by

Name:Roda Chinthapalli

Roll No: AP22110011496

Section -M

Lab Date: 22/10/24

Submission Date:28/10/24

**A picture containing text

Description automatically generated**

**Department Computer Science and Engineering**

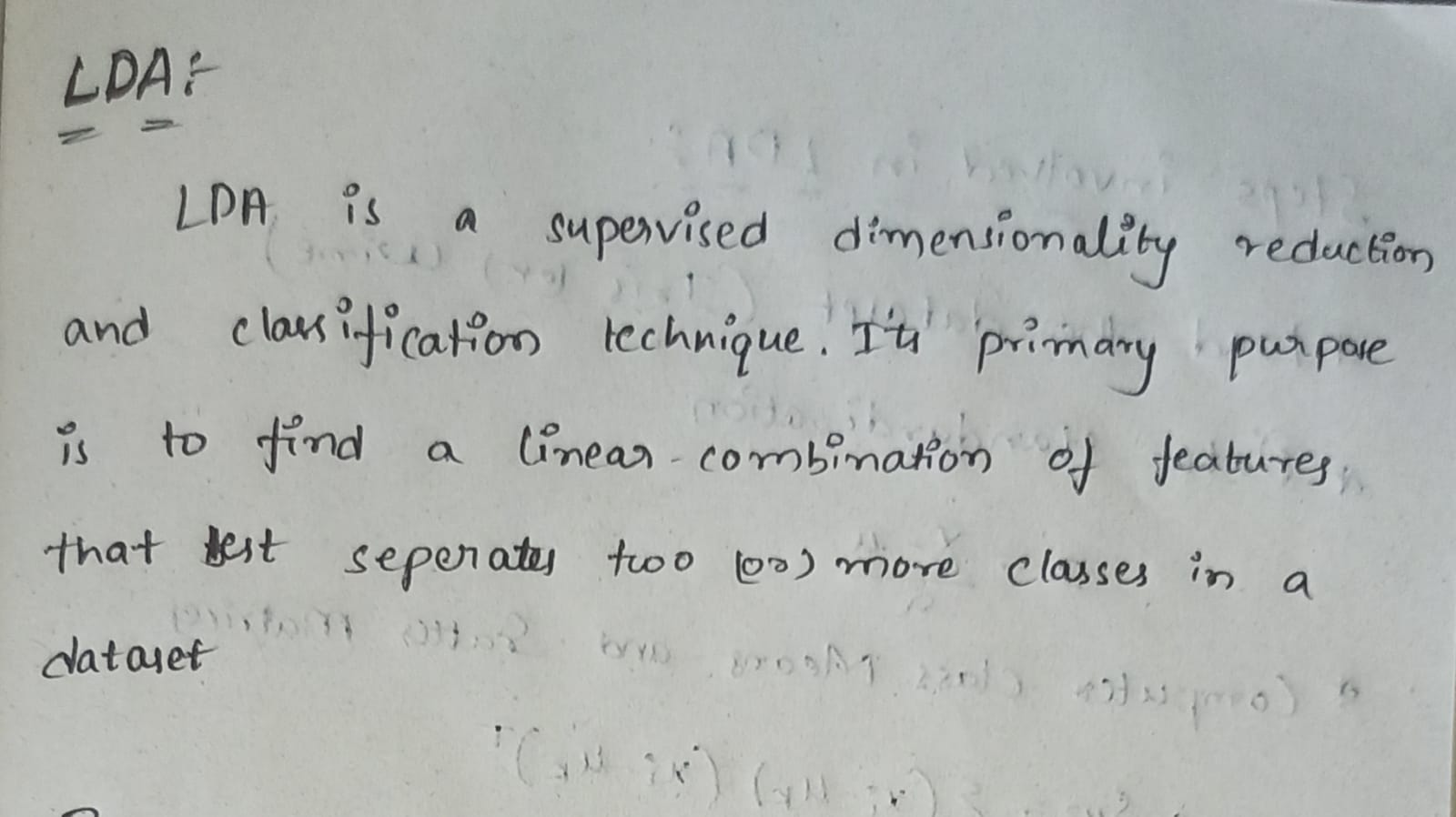
**School of Engineering and Sciences**

**SRM University–AP**

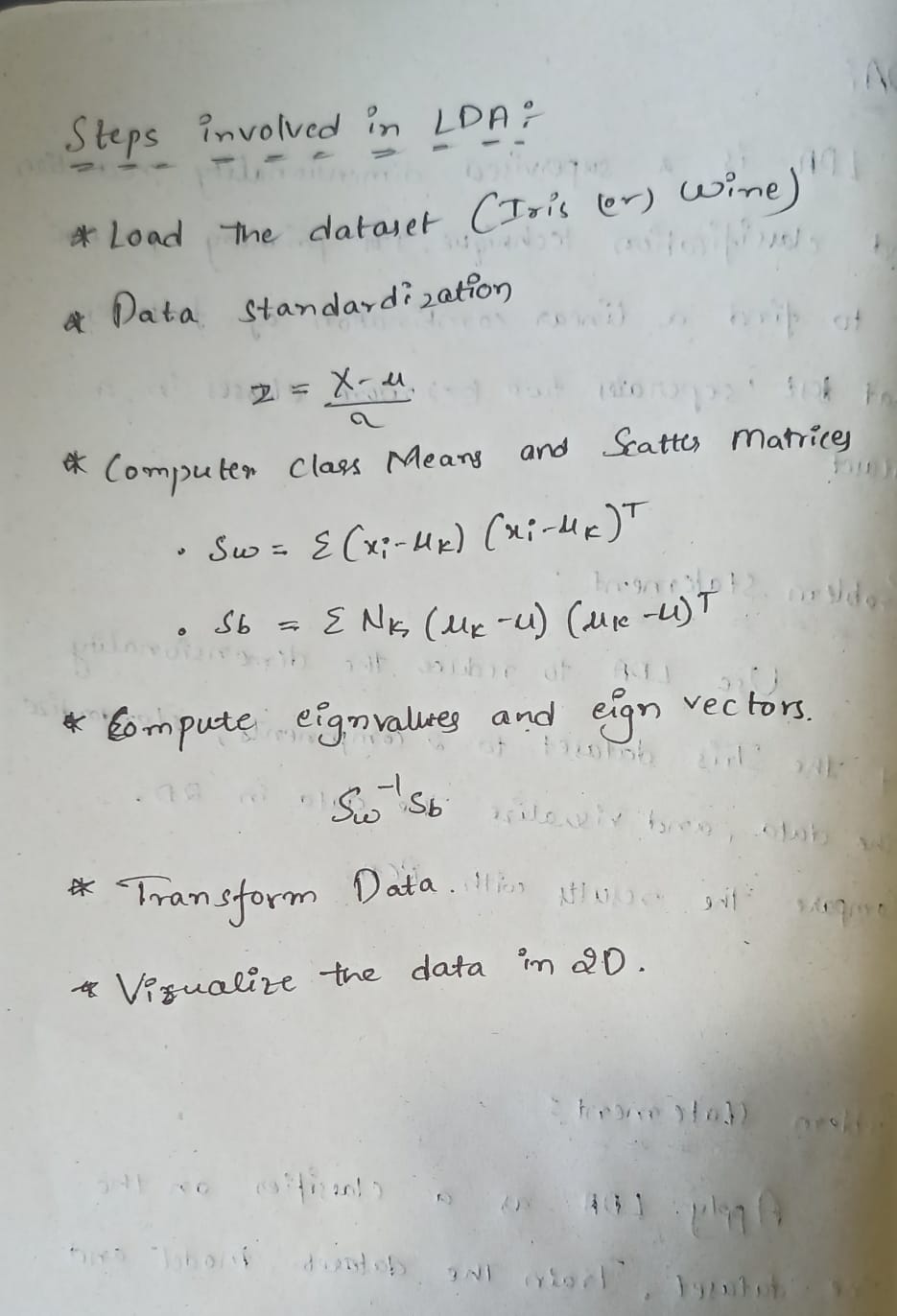
**Amaravati, Andhra Pradesh – 522 240, India**

**LINEAR DISCRIMINANT ANALYSIS(LDA)**

**LDA**

****

**STEPS INVOLVED IN LDA**

****

**Lab Exercise 1: Introduction to LDA for Dimensionality Reduction**

**Objective:** Understand the basic principles of LDA and apply it to reduce the dimensionality of a simple classification dataset.

**Instructions:**

**1. Load the Dataset:**

o Use the Iris dataset (available in sklearn or seaborn). The dataset has 4 features and 3 classes.

**2. Data Standardization:**

o Standardize the data so that it has a mean of 0 and a standard deviation of 1.

**3. Apply LDA:**

o Implement LDA using sklearn&#39;s LinearDiscriminantAnalysis class. Reduce the dimensionality of the dataset to 2

components.

**4. Visualization:**

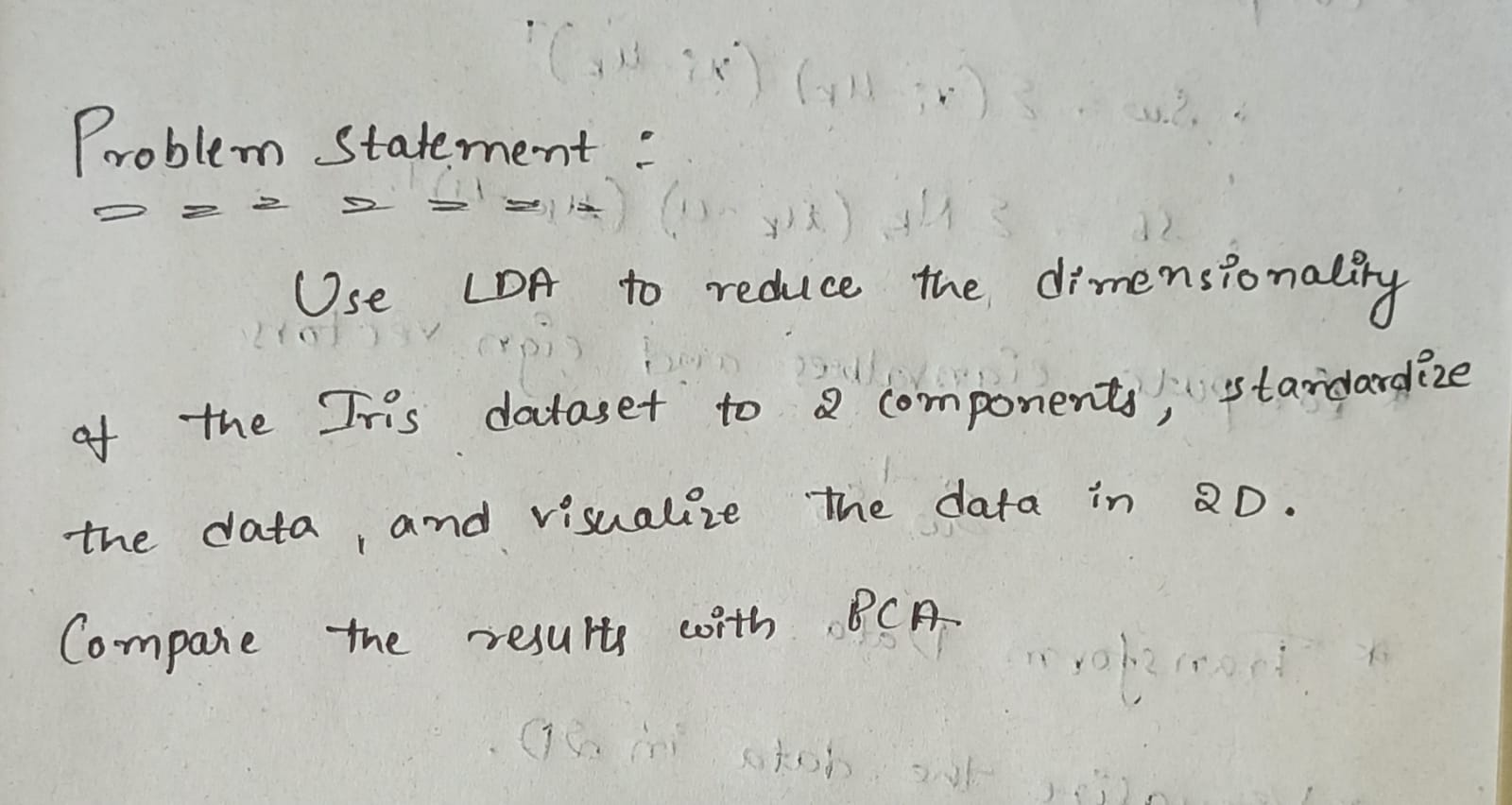
o Create a 2D scatter plot of the transformed data, using different colors for each class.

**5. Compare LDA with PCA:**

o Plot the first two components from PCA (from the previous lab if done) on the same dataset and compare the

results with LDA.

**PROBLEM STATEMENT:**

****

**SOLUTION:**

**Dataset: Iris**

**Load the Dataset**

**import pandas as pd**

**# Load dataset from your file path**

**iris = pd.read\_csv('/content/iris.csv')**

**X = iris.iloc[:, :-1] # Features (first 4 columns)**

**y = iris.iloc[:, -1] # Target (last column for species)**

**Data Standardisation**

**from sklearn.preprocessing import StandardScaler**

**scaler = StandardScaler()**

**X\_scaled = scaler.fit\_transform(X)**

**Apply LDA**

**from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis as LDA**

**lda = LDA(n\_components=2)**

**X\_lda = lda.fit\_transform(X\_scaled, y)**

**Visulization**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**# Create a DataFrame for plotting**

**lda\_df = pd.DataFrame(X\_lda, columns=['LD1', 'LD2'])**

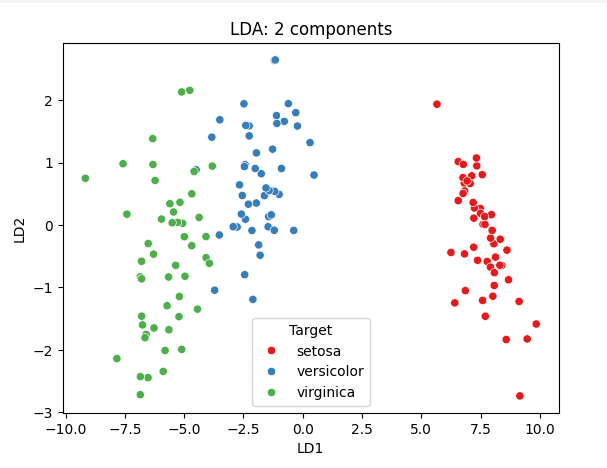
**lda\_df['Target'] = y**

**# Plot the LDA components**

**sns.scatterplot(x='LD1', y='LD2', hue='Target', data=lda\_df, palette='Set1')**

**plt.title('LDA: 2 components')**

**plt.show()**

****

**Compare LDA with PCA**

**from sklearn.decomposition import PCA**

**pca = PCA(n\_components=2)**

**X\_pca = pca.fit\_transform(X\_scaled)**

**# Create a DataFrame for PCA plotting**

**pca\_df = pd.DataFrame(X\_pca, columns=['PC1', 'PC2'])**

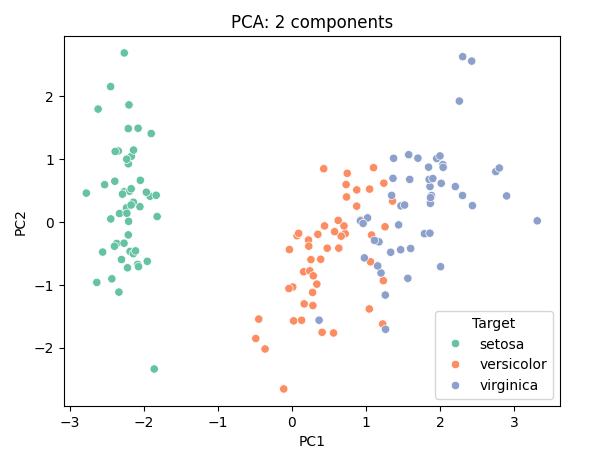
**pca\_df['Target'] = y**

**# Plot PCA components**

**sns.scatterplot(x='PC1', y='PC2', hue='Target', data=pca\_df, palette='Set2')**

**plt.title('PCA: 2 components')**

**plt.show()**



Lab Exercise 2: LDA for Classification

**Objective:** Learn how LDA can be used as a classifier by applying it to a multi-class classification problem.

**Instructions:**

**1. Load the Dataset:**

o Use the Wine dataset from sklearn, which contains 13 features and 3 classes.

**2. Split the Data:**

o Split the data into training and testing sets (e.g., 70% training, 30% testing).

**3. Train an LDA Model:**

o Use LDA as a classifier by training a LinearDiscriminantAnalysis model on the training data.

**4. Evaluate the Model:**

o Predict the labels on the test set and compute the model&#39;s accuracy, precision, recall, and confusion matrix.

**5. Compare with Logistic Regression:**

o Train a logistic regression model on the same dataset, and compare its performance with the LDA classifier.

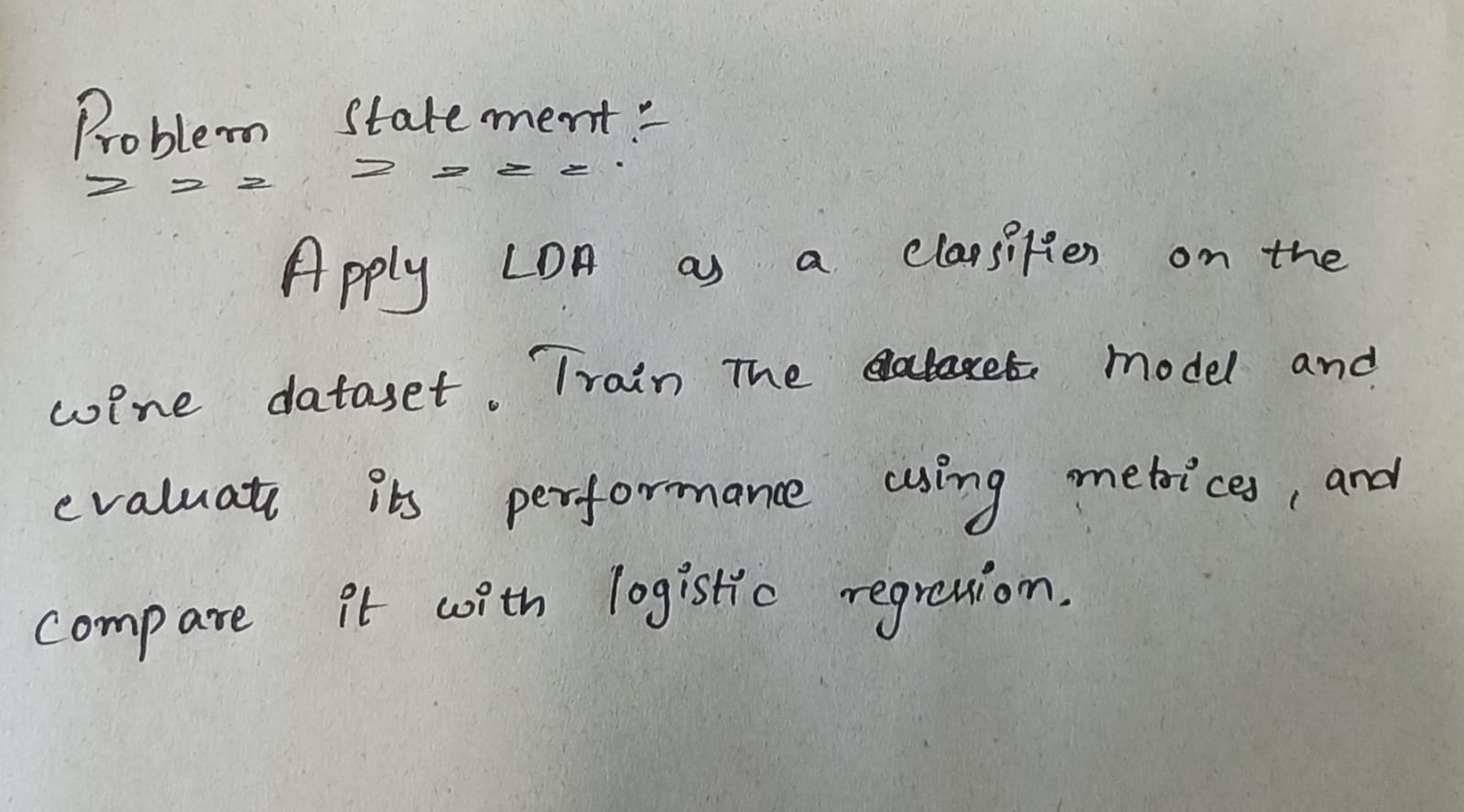
**6. Visualize Decision Boundaries (Optional):**

o For an intuitive understanding, visualize the decision boundaries for both the LDA and logistic regression

models in a 2D space (you can reduce the dataset to 2 dimensions using LDA or PCA for visualization

purposes).

**PROBLEM STATEMENT:**

****

**SOLUTION:**

**Dataset: Wine**

**Load the Data**

**import pandas as pd**

**from zipfile import ZipFile**

**# Load dataset from zip file**

**with ZipFile('/content/wine.zip', 'r') as zip\_ref:**

**zip\_ref.extractall('/content')**

**# Read the dataset**

**wine = pd.read\_csv('//content/winequality-red.csv')**

**X = wine.iloc[:, :-1] # Features (first 13 columns)**

**y = wine.iloc[:, -1] # Target (last column for wine class)**

**Split the Data**

**from sklearn.model\_selection import train\_test\_split**

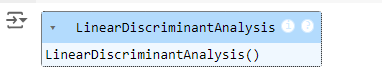
**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)**

**Train an LDA Model**

**from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis as LDA**

**lda = LDA()**

**lda.fit(X\_train, y\_train)**

****

**Evaluate the Model**

**# Compute evaluation metrics with zero\_division parameter**

**precision = precision\_score(y\_test, y\_pred, average='weighted', zero\_division=0)**

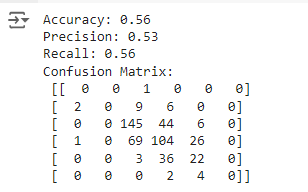
**recall = recall\_score(y\_test, y\_pred, average='weighted', zero\_division=0)**

**print(f'Accuracy: {accuracy:.2f}')**

**print(f'Precision: {precision:.2f}')**

**print(f'Recall: {recall:.2f}')**

**print('Confusion Matrix:\n', conf\_matrix)**

****

**Compare the Logistic Regression**

**# Import necessary libraries**

**from sklearn.model\_selection import train\_test\_split**

**from sklearn.preprocessing import StandardScaler**

**from sklearn.linear\_model import LogisticRegression**

**from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, confusion\_matrix**

**# Assuming you have already loaded the dataset into X and y**

**# Example: X, y = load\_wine(return\_X\_y=True)**

**# Split the dataset into training and testing sets (70% train, 30% test)**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)**

**# Standardize the data**

**scaler = StandardScaler()**

**X\_train\_scaled = scaler.fit\_transform(X\_train)**

**X\_test\_scaled = scaler.transform(X\_test)**

**# Train Logistic Regression model with increased max\_iter**

**logreg = LogisticRegression(max\_iter=1000)**

**logreg.fit(X\_train\_scaled, y\_train)**

**# Make predictions**

**y\_pred\_logreg = logreg.predict(X\_test\_scaled)**

**# Compute evaluation metrics**

**accuracy\_logreg = accuracy\_score(y\_test, y\_pred\_logreg)**

**precision\_logreg = precision\_score(y\_test, y\_pred\_logreg, average='weighted', zero\_division=0)**

**recall\_logreg = recall\_score(y\_test, y\_pred\_logreg, average='weighted', zero\_division=0)**

**conf\_matrix\_logreg = confusion\_matrix(y\_test, y\_pred\_logreg)**

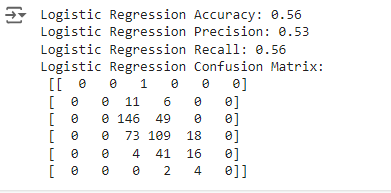
**# Print evaluation metrics**

**print(f'Logistic Regression Accuracy: {accuracy\_logreg:.2f}')**

**print(f'Logistic Regression Precision: {precision\_logreg:.2f}')**

**print(f'Logistic Regression Recall: {recall\_logreg:.2f}')**

**print('Logistic Regression Confusion Matrix:\n', conf\_matrix\_logreg)**

****

**Visuvalize the Boundaries**

**# Import required libraries**

**import numpy as np**

**import matplotlib.pyplot as plt**

**from sklearn.datasets import load\_wine**

**from sklearn.model\_selection import train\_test\_split**

**from sklearn.preprocessing import StandardScaler**

**from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis as LDA**

**from sklearn.decomposition import PCA**

**from sklearn.linear\_model import LogisticRegression**

**from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, confusion\_matrix**

**# 1. Load the Wine dataset**

**data = load\_wine()**

**X = data.data**

**y = data.target**

**# 2. Split the dataset into training and testing sets (70% train, 30% test)**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)**

**# 3. Standardize the data (mean = 0, std = 1)**

**scaler = StandardScaler()**

**X\_train\_scaled = scaler.fit\_transform(X\_train)**

**X\_test\_scaled = scaler.transform(X\_test)**

**# 4. Reduce dataset to 2D using LDA**

**lda\_2d = LDA(n\_components=2)**

**X\_train\_lda = lda\_2d.fit\_transform(X\_train\_scaled, y\_train)**

**# Function to plot decision boundaries**

**def plot\_decision\_boundaries(X, y, model, title):**

**x\_min, x\_max = X[:, 0].min() - 1, X[:, 0].max() + 1**

**y\_min, y\_max = X[:, 1].min() - 1, X[:, 1].max() + 1**

**xx, yy = np.meshgrid(np.arange(x\_min, x\_max, 0.01), np.arange(y\_min, y\_max, 0.01))**

**# Predict over the grid**

**Z = model.predict(np.c\_[xx.ravel(), yy.ravel()])**

**Z = Z.reshape(xx.shape)**

**# Plot the contour and data points**

**plt.contourf(xx, yy, Z, alpha=0.3)**

**plt.scatter(X[:, 0], X[:, 1], c=y, edgecolors='k', marker='o')**

**plt.title(title)**

**plt.show()**

**# 5. Train a new LDA model on the 2D-reduced data**

**lda\_model = LDA()**

**lda\_model.fit(X\_train\_lda, y\_train)**

**# Plot decision boundaries for LDA**

**plot\_decision\_boundaries(X\_train\_lda, y\_train, lda\_model, "LDA Decision Boundaries")**

**# 6. Reduce dataset to 2D using PCA for Logistic Regression**

**pca = PCA(n\_components=2)**

**X\_train\_pca = pca.fit\_transform(X\_train\_scaled)**

**# 7. Train Logistic Regression on the 2D-reduced data**

**logreg = LogisticRegression(max\_iter=2000) # Increased max\_iter to 2000**

**logreg.fit(X\_train\_pca, y\_train)**

**# Plot decision boundaries for Logistic Regression**

**plot\_decision\_boundaries(X\_train\_pca, y\_train, logreg, "Logistic Regression Decision Boundaries")**

**# Evaluate the Logistic Regression model on test data (optional)**

**X\_test\_pca = pca.transform(X\_test\_scaled)**

**y\_pred\_logreg = logreg.predict(X\_test\_pca)**

**# Compute evaluation metrics**

**accuracy\_logreg = accuracy\_score(y\_test, y\_pred\_logreg)**

**precision\_logreg = precision\_score(y\_test, y\_pred\_logreg, average='weighted')**

**recall\_logreg = recall\_score(y\_test, y\_pred\_logreg, average='weighted')**

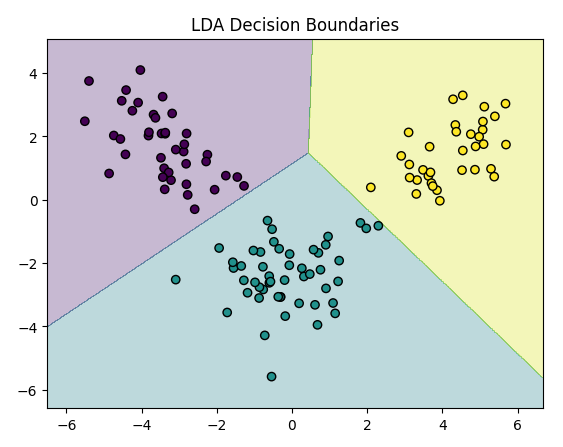
**conf\_matrix\_logreg = confusion\_matrix(y\_test, y\_pred\_logreg)**

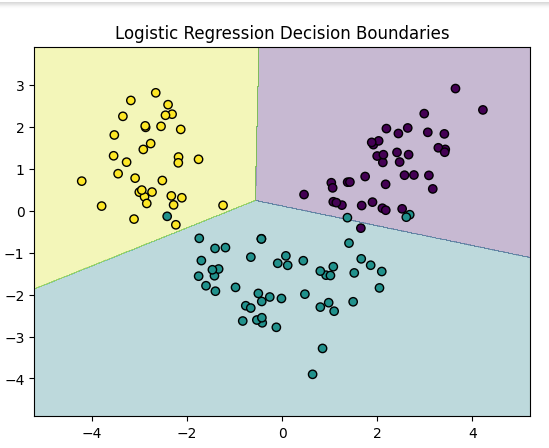
**print(f'Logistic Regression Accuracy: {accuracy\_logreg:.2f}')**

**print(f'Logistic Regression Precision: {precision\_logreg:.2f}')**

**print(f'Logistic Regression Recall: {recall\_logreg:.2f}')**

**print('Logistic Regression Confusion Matrix:\n', conf\_matrix\_logreg)**

****

****

**CODE REPOSITORY:**

**GITHUB:** **https://github.com/Roda1458/LDA**