In [1]:

```
import numpy as np
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix

# Load the Iris dataset
iris = datasets.load_iris()
X = iris.data
y = iris.target
print(X[:5])

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=iris.target, test_size=0.4, random_state=10)

def Compute_linear_discriminant(cp, x):
    # cp is class prototype, x is new input
    return np.dot(cp, x) - 0.5 * np.dot(cp, cp)

def calculate_prototypes(X_train, y_train):
    class_prototypes = []
    for class_label in np.unique(y_train):
        # make an array of all the samples belonging to a given class_label
        class_samples = X_train[y_train == class_label]
        # find the mean of these class label samples
        class_prototype = np.mean(class_samples, axis=0)
        class_prototypes.append(class_prototype)
    return np.array(class_prototypes)

def ld_classifier(test_sample, class_prototypes):
    predicted_class = None
    max_discriminant = float('-inf')
    for i in range(len(class_prototypes)):
        discriminant = Compute_linear_discriminant(class_prototypes[i], test_sample)
        if discriminant > max_discriminant:
            max_discriminant = discriminant
            predicted_class = i
    return predicted_class

# Calculate class prototypes from the training data
class_prototypes = calculate_prototypes(X_train, y_train)

# Make predictions for all the test samples
predictions = []
for test_sample in X_test:
    predicted_class = ld_classifier(test_sample, class_prototypes)
    predictions.append(predicted_class)

# Performance evaluation of classifier
accuracy = np.mean(np.array(predictions) == y_test)
print("Accuracy:", accuracy)
print('\nConfusion Matrix is:\n', confusion_matrix(y_test, predictions))
```

```
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]]
Accuracy: 0.9833333333333333

Confusion Matrix is:
 [[20  0  0]
[ 0 20  0]
[ 0  1 19]]
```

In [ ]: