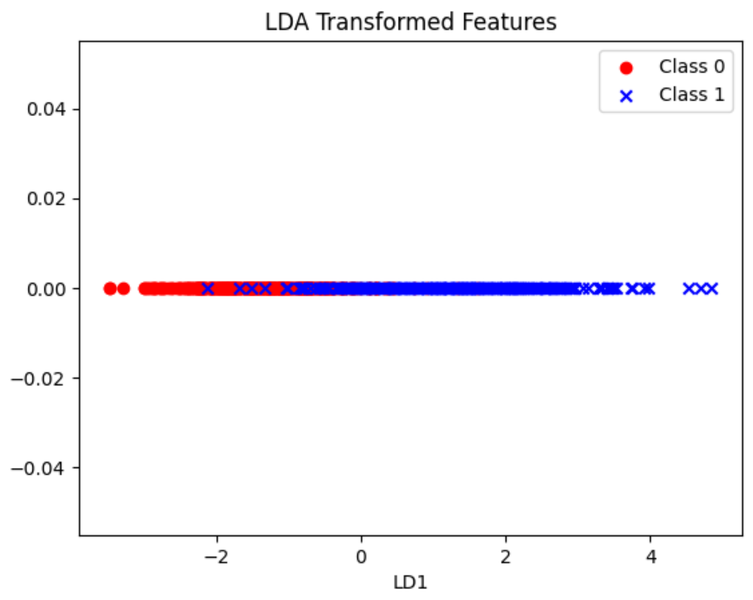
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.metrics import accuracy_score, confusion_matrix

# Generate some synthetic data for classification
X, y = make_classification(n_samples=1000, n_features=2, n_classes=2, n_clusters_per_class=1, n_redundant=0, random_state=42)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Create an LDA classifier
lda_classifier = LinearDiscriminantAnalysis(n_components=1)  # Set n_components to 1

# Fit the classifier on the training data and transform the features to the lower-dimensional space
X_train_lda = lda_classifier.fit_transform(X_train, y_train)
X_test_lda = lda_classifier.transform(X_test)

# Visualize the transformed features in the lower-dimensional space
plt.scatter(X_train_lda[y_train == 0], np.zeros_like(X_train_lda[y_train == 0]), label='Class 0', c='red', marker='o')
plt.scatter(X_train_lda[y_train == 1], np.zeros_like(X_train_lda[y_train == 1]), label='Class 1', c='blue', marker='x')
plt.xlabel('LD1')
plt.legend()
plt.title('LDA Transformed Features')
plt.show()

# Train a classifier on the LDA-transformed features
lda_classifier.fit(X_train_lda, y_train)

# Make predictions on the test data
y_pred = lda_classifier.predict(X_test_lda)

# Calculate accuracy of the classifier
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: {:.2f}%".format(accuracy * 100))

# Create a confusion matrix to evaluate the classifier's performance
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)
```



```
Accuracy: 88.67%
Confusion Matrix:
[[139   8]
 [ 26 127]]
```