تمرین شماره ۵

درس شناسایی الگو

استاد : جناب دکتر فدائی اسلام

دانشجویان : سامان تبریزی – هادی صفرعرب

سوال ۱ قسمت الف)

در این سوال یک دیتاست در رابطه با پیش بینی شروع بیماری قند در اختیار داریم و میخواهیم ارتباط هر ویژگی را به نتیجه کلاس بندی مشخص کنیم . همانطور که صورت سوال خواسته است دو ویژگی توده وزنی و فشار خون را مورد بررسی قرار میدهیم و نمودار ROC را برای این دو ویژگی رسم میکنیم.

مراحل کار به شرح زیر میباشد:

```
import pandas as pd , requests
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression  # یا مدل دسته‌بندی مورد نظر خود
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from io import StringIO
```

۱. **درخواست دادن دیتا از اینترنت**:

- از کتابخانه `requests` برای ارسال درخواست HTTP به URL مربوط به دیتاست Pima Indians Diabetes استفاده شده است.

```
# Data set url
url = 'https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.csv'

# Make a request to download the CSV content
response = requests.get(url)

# Check if the request was successful (status code 200)
if response.status_code == 200:
    # Read the CSV content into a pandas DataFrame
```

```
    csv_content = StringIO(response.text)
    df = pd.read_csv(csv_content)

    # Display the DataFrame
    print(df)
else:
    print(f"Failed to download CSV file. Status code:
{response.status_code}")
```

۲. **خواندن دیتا از فایل CSV در حافظه**:

- از `StringIO` از کتابخانه `io` برای خواندن محتوای CSV به صورت رشته (String) استفاده شده است.

- از `pd.read_csv` برای تبدیل رشته CSV به دیتافریم پانداس استفاده شده است.

```
    # Read the CSV content into a pandas DataFrame
    csv_content = StringIO(response.text)
    df = pd.read_csv(csv_content)
```

۳. **پاک کردن سطرهایی که مقدار ستون سوم آنها برابر با ۰ است**:

- این کار باعث حذف سطرهایی می‌شود که دارای مقدار صفر در ستون سوم هستند.

```
# حذف سطرهایی که مقدار ستون سوم آنها برابر با ۰ است
df = df[df.iloc[:, 2] != 0]
```

۴. **آماده‌سازی داده برای مدل سازی**:

- انتخاب ستون مورد نظر (اینجا ستون سوم) برای ورودی `X` و برچسب‌ها `y`.

- تبدیل ویژگی‌ها به یک آرایه دو بعدی با استفاده از `values.reshape(-۱, ۱)`.

```

# فرض کنید داده‌های مورد نیاز در ستون سوم (index 2) قرار دارند
X = df.iloc[:, 2].values.reshape(-1, 1)  # تبدیل به آرایه ۲ بعدی برای
استفاده در مدل
```

```
# برچسب‌های مرتبط با داده‌ها (باشند CSV برای مثال، فرض کنید برچسب‌ها در ستون اخر فایل)
y = df.iloc[:, -1].values
```

۵. **تقسیم داده به داده‌های آموزش و تست**:

- از `train_test_split` برای تقسیم داده به دو مجموعه آموزش و تست استفاده شده است.

```
# جدا کردن داده‌ها به داده‌های آموزشی و تست
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

۶. **ساخت مدل دسته‌بندی (لجستیک رگرسیون)**:

- از `LogisticRegression` از scikit-learn برای ساخت یک مدل دسته‌بندی استفاده شده است.

```
# ساخت مدل دسته‌بندی (در اینجا از یک مدل رگرسیون لجستیک به‌عنوان مثال استفاده شده است)
model = LogisticRegression()
model.fit(X_train, y_train)
```

۷. **پیش‌بینی احتمالات و محاسبه نمودار ROC**:

- مدل بر روی داده‌های تست فراخوانی شده و احتمالات پیش‌بینی برچسب یک محاسبه می‌شود.

- از `roc_curve` و `auc` برای محاسبه نمودار ROC و مساحت زیر نمودار ROC استفاده شده است.

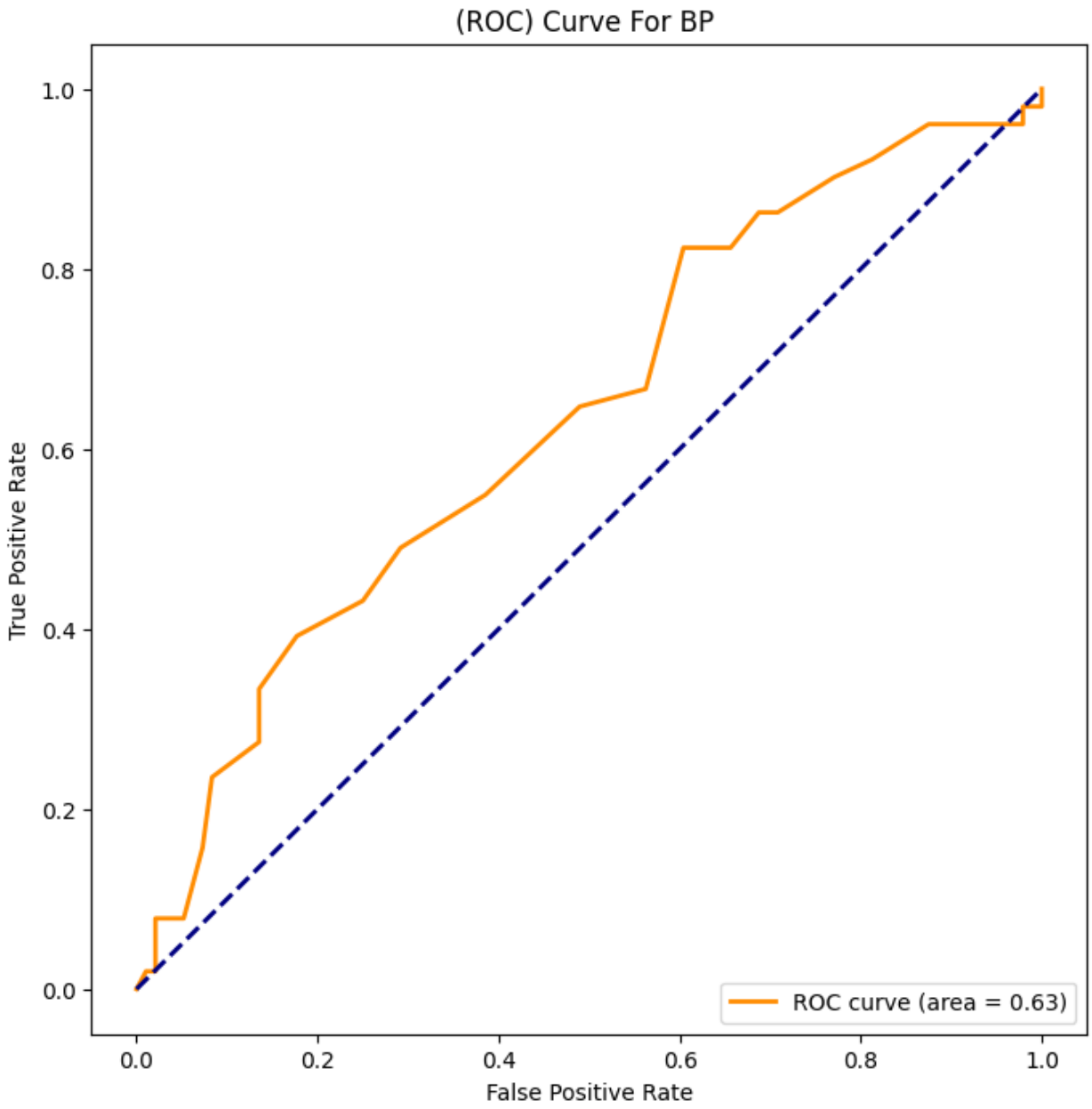
```
# احتمالات پیش‌بینی برچسب یک را محاسبه کنید
y_probs = model.predict_proba(X_test)[:, 1]
# محاسبه نرخ TPR و FPR
fpr, tpr, thresholds = roc_curve(y_test, y_probs)

# محاسبه مساحت زیر نمودار ROC (AUC)
roc_auc = auc(fpr, tpr)
```

۸. **ترسیم نمودار ROC**:

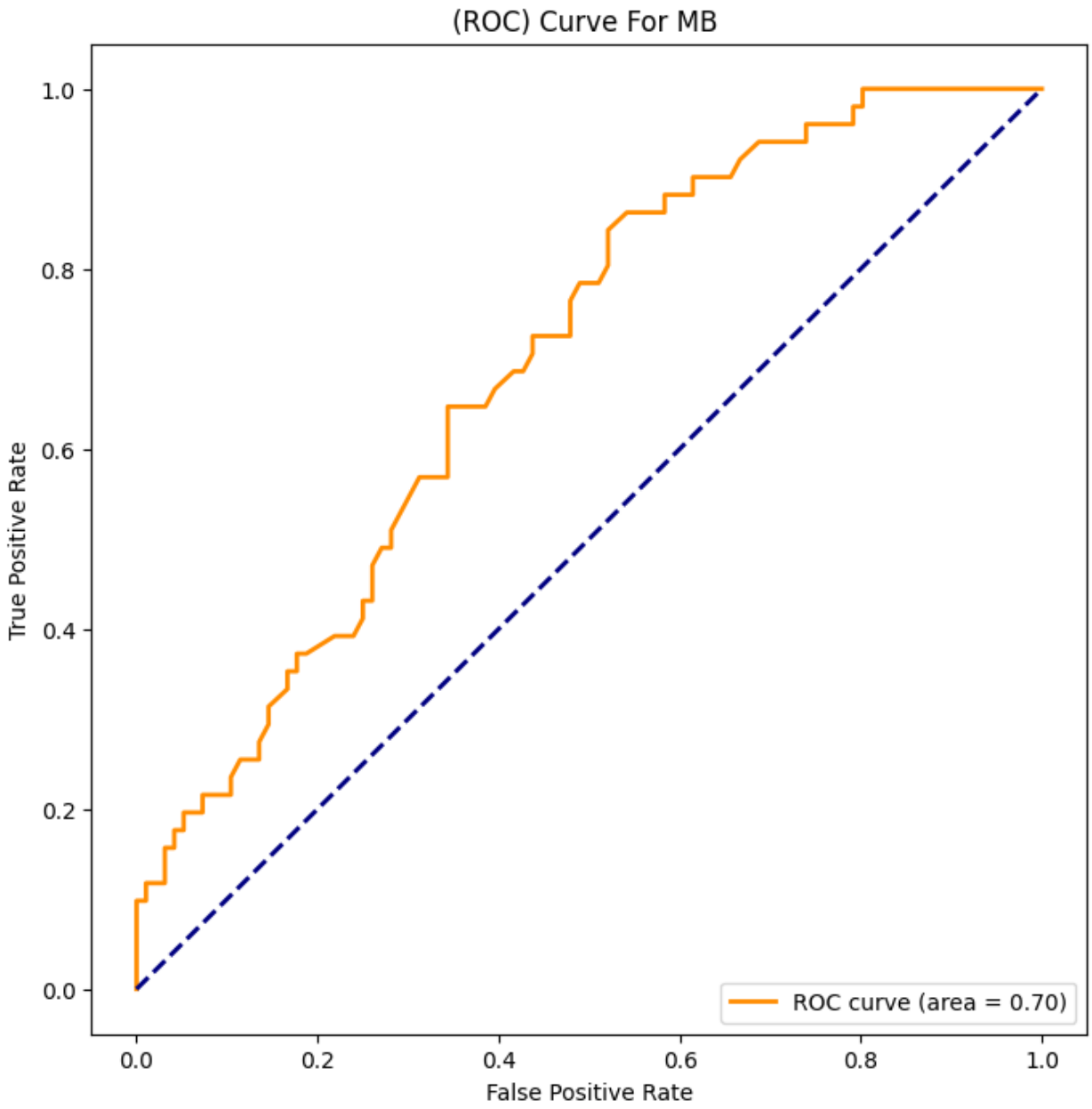
- از `matplotlib` برای ترسیم نمودار ROC با استفاده از مقادیر False Positive Rate (FPR) و True Positive Rate (TPR) استفاده شده است.

```
# نمودار ROC
plt.figure(figsize=(8, 8))
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC
curve (area = {:.2f})'.format(roc_auc))
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('(ROC) Curve For BP ')
plt.legend(loc='lower right')
plt.show()
```

خروجی این کد برای ویژگی فشار خون به صورت زیر ترسیم میشود:



با تکرار مراحل بالا برای ویژگی توده وزنی نمودارROC این ویژگی نیز به صورت زیر میباشد



سوال۱ قسمت ب)

در این قسمت از ما خواسته شده است تا با کمک LDA مقادیر داده ها را روی یک خط تصویر کنیم و با توجه به پراکندگی داده ها یک نقطه برای جدا سازی بهینه داده ها مشخص کنیم.

برای این کار ابتدا باید داده ها را در قالب دیتا فریم پانداس به کد پایتون خودمون وارد کنیم تا توانایی کار با این داده هارا داشته باشیم.

پس در قدم اول کتابخانه های لازم را وارد میکنیم .

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.discriminant_analysis import
LinearDiscriminantAnalysis
from sklearn.preprocessing import StandardScaler
```

۱. **خواندن داده**:

- در این قسمت از کتابخانه pandas برای خواندن داده از یک فایل CSV به نام 'pima-indians-diabetes.csv' استفاده شده است. داده دارای ستون‌های مختلفی است که با نام‌های مشخص شده در `names` در DataFrame (`df`) ذخیره می‌شوند.

```
# Load the dataset
df = pd.read_csv('pima-indians-diabetes.csv', header=None,
names=['Number of times pregnant', 'Plasma glucose
concentration a 2 hours in an oral glucose tolerance test',
'Diastolic blood pressure (mm Hg)', 'Triceps skinfold
thickness (mm)', '2-Hour serum insulin (mu U/ml)', 'Body
mass index (weight in kg/(height in m)^2)', 'Diabetes
pedigree function', 'Age (years)', 'Class variable (0 or
1)'])
# Replace zeros with NaNs in specific columns as they are
considered missing values
for column in ['Plasma glucose concentration a 2 hours in an
oral glucose tolerance test', 'Diastolic blood pressure (mm
Hg)', 'Triceps skinfold thickness (mm)', '2-Hour serum
insulin (mu U/ml)', 'Body mass index (weight in kg/(height
in m)^2)']:
    df[column].replace(0, np.nan, inplace=True)
```

۲. **پر کردن مقادیر نامعتبر (صفر) با NaN**:

- در این دیتاست بدلیل آنکه نیاز داریم دقت مدل بالا باشد، و از طرف دیگر تعدادی از دیتاها ۰ میباشند .
پس این داده هارا حذف میکنیم تا با دیتاهای valid مدل را ایجاد کنیم.

```
# Drop rows with NaNs
df.dropna(inplace=True)
```

۳. **حذف سطرهای دارای NaN**:

- سطرهایی که حاوی حداقل یک مقدار NaN هستند، حذف شده‌اند.

۴. **انتخاب ویژگی‌ها و هدف**:

- متغیرهای مستقل (ویژگی‌ها) و وابسته (هدف) از داده انتخاب شده‌اند.

```
# انتخاب ویژگی‌ها و هدف
features = df.drop('Class variable (0 or 1)', axis=1)
target = df['Class variable (0 or 1)']
```

۵. **استانداردسازی ویژگی‌ها**:

- از `StandardScaler` از scikit-learn برای استانداردسازی ویژگی‌ها استفاده شده است. این کار به معنای تبدیل داده‌ها به یک مقیاس استاندارد می‌باشد.

```
# استانداردسازی ویژگی‌ها
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)
```

۶. **اعمال LDA**:

- از کلاس `LinearDiscriminantAnalysis` از scikit-learn برای اجرای تحلیل تباعد خطی با تعداد اجزا (`n_components`) برابر با ۱ استفاده شده است.

```
# اعمال LDA
lda = LinearDiscriminantAnalysis(n_components=1)
lda_result = lda.fit_transform(features_scaled, target)
```

۷. **تصویر نمودن داده‌ها بر روی یک خط**:

- نتایج تحلیل تباعد خطی روی یک خط به صورت نمودار Scatter Plot نمایش داده شده‌اند. رنگ نقاط بر اساس کلاس (صفر یا یک) تعیین شده و مقدار تباعد خطی در نمودار نشان داده شده است.

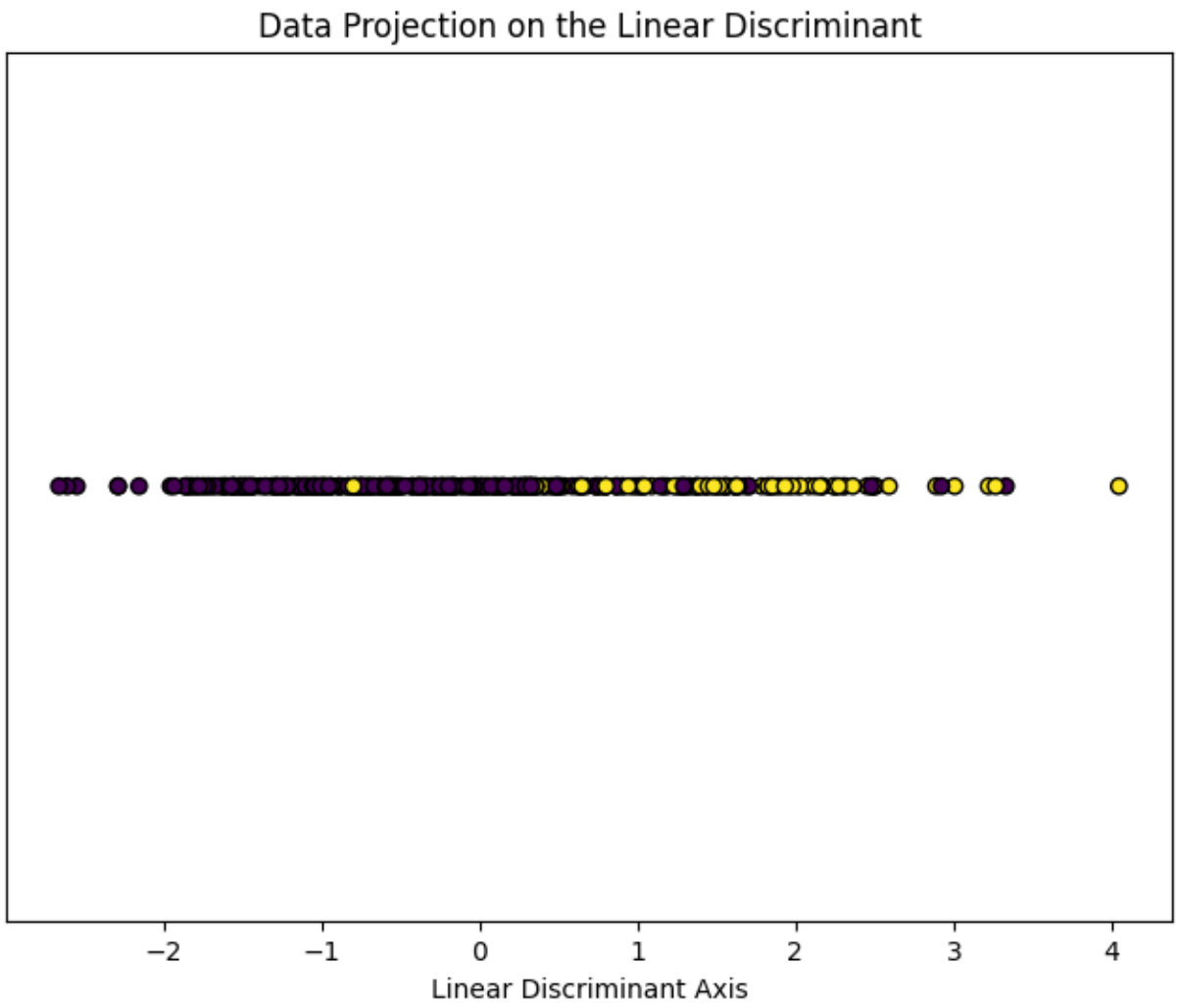
```
#  تصویر نمودن داده‌ها بر روی یک خط با بیشترین جداپذیری
plt.figure(figsize=(8, 6))
plt.scatter(lda_result[:, 0], np.zeros_like(lda_result),
c=target, cmap='viridis', marker='o', edgecolor='k')
plt.title('Data Projection on the Linear Discriminant')
plt.xlabel('Linear Discriminant Axis')
plt.yticks([])
plt.show()
```

۸. **محاسبه و نمایش آستانه (Threshold)**:

- میانگین مقادیر تباعد خطی به عنوان آستانه تعیین شده و نمایش داده شده است.

```
threshold = np.mean(lda_result)
print(f'Threshold: {threshold}')
```

خروجی کد برای داده های دیتاست به صورت زیر خواهد بود. با توجه به نمودار ، مقدار آستانه جدا کننده عدد Threshold: ۱,۳۱۴۱۴۱۵۳۹۳۰۲۲۲۶e-۱۶ میباشد

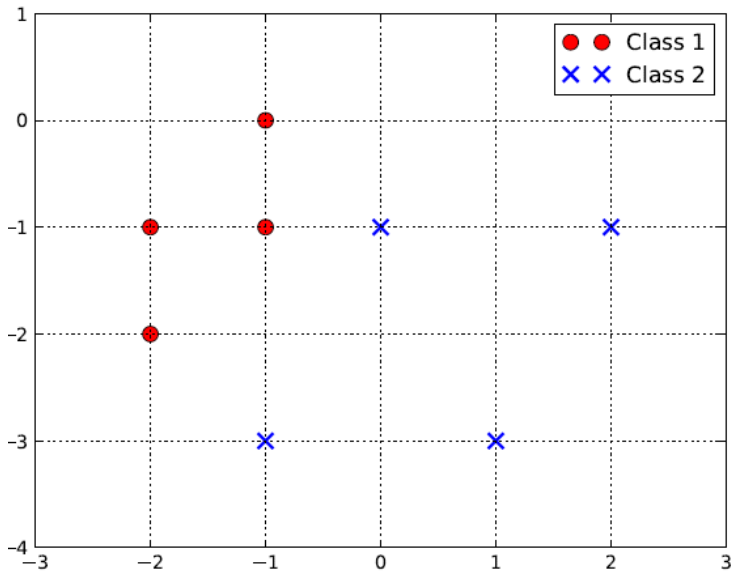


Threshold: ١,٣١٤١٤١٥٣٩٣٥٢٢٢٦e-١٦

سوال ۲)

با توجه به نقاط داده شده میتوانیم یک ابر صفحه یا خط را مشخص کنیم که جداکننده داده ها باشد.

برای این کار مراحل زیر را انجام خواهیم داد.



این کد یک تحلیل LDA بر روی دو ویژگی (x۱ و x۲) از داده‌های آموزش و دسته‌بندی آنها به دو دسته مختلف (۰ و ۱) را نشان می‌دهد. داده‌ها به صورت دستی ایجاد شده‌اند و برخی از آن‌ها توسط خط ترتیب داده شده‌اند.

در ادامه توضیحاتی در مورد هر بخش از کد آورده شده است:

```
import numpy as np
from sklearn.discriminant_analysis import
LinearDiscriminantAnalysis
```

۱. **ساخت داده‌های آموزش**:

- یک ماتریس `x` به صورت دستی ایجاد شده است که هر سطر آن نمایانگر یک نمونه داده با دو ویژگی (x۱ و x۲) است.

- یک بردار `y` نیز برای نشان دادن دسته‌بندی هر نمونه به دو دسته (۰ یا ۱) ایجاد شده است.

```
x = np.array([[-2, -1], [-2, -2], [-1, 0], [-1, -1], [-1, -
۳], [۰, -۱], [۱, -۳], [۲, -۱]])
y = np.array([0, 0, 0, 0, 1, 1, 1, 1])
```

۲. **اجرای LDA**:

- از کلاس `LinearDiscriminantAnalysis` از scikit-learn برای اجرای تحلیل تباعد خطی با تعداد اجزا (`n_components`) برابر با ۱ استفاده شده است.

- با استفاده از `fit_transform`، داده‌ها به اجزا تباعد خطی تبدیل می‌شوند.

```
lda = LinearDiscriminantAnalysis(n_components=1)
x_lda = lda.fit_transform(x, y)
```

۳. **چاپ ضرایب خط جداکننده**:

- با استفاده از `lda.coef_`، ضرایب خط جداکننده برای تباعد خطی چاپ می‌شوند.

```
# The coefficients of the separating line
print('Coefficients of the separating line: ', lda.coef_)
```

۴. **ترسیم داده‌ها و خط جداکننده**:

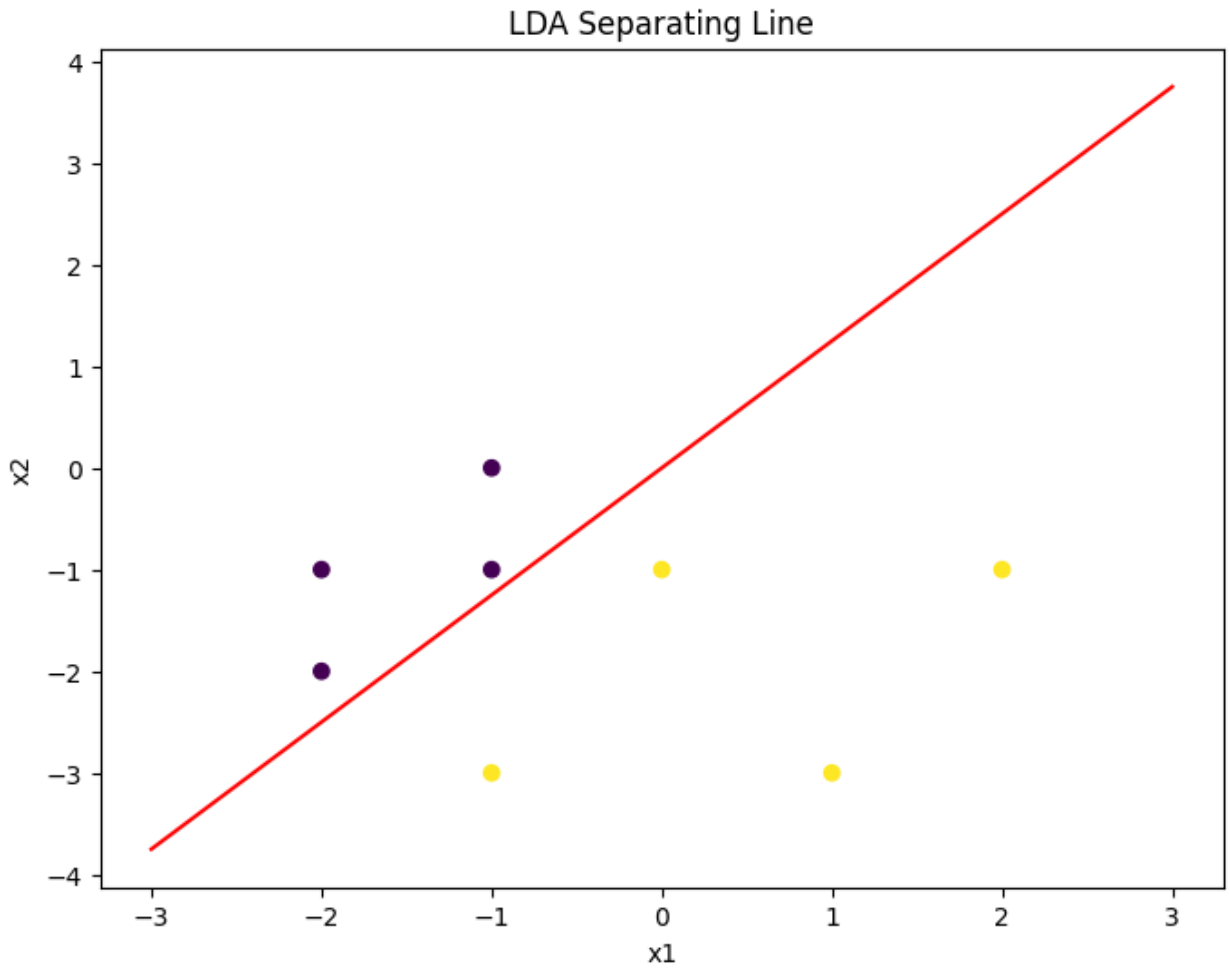
- از `matplotlib` برای ترسیم داده‌های آموزش به صورت Scatter Plot با رنگهای متفاوت بر اساس دسته‌بندی `y` استفاده شده است.

```
# Plot the original data points
plt.figure(figsize=(8, 6))
plt.scatter(x[:, 0], x[:, 1], c=y)

# Create a range of values for the x-axis
x_values = np.linspace(-3, 3, 400)
```

```
# Calculate the corresponding y values for the separating
line
y_values = -(lda.coef_[0][0] * x_values) / lda.coef_[0][1]

# Plot the separating line
plt.plot(x_values, y_values, color='red')

plt.title('LDA Separating Line')
plt.xlabel('x1')
plt.ylabel('x2')
plt.show()
```

- یک خط جداکننده LDA با استفاده از ضرایب به دست آمده از تحلیل رسم شده است.
- نمودار نشان می‌دهد چگونه خط LDA بهترین جداکننده برای دو دسته است.

به عبارت دیگر، LDA تلاش می‌کند یک خط جداکننده ایجاد کند که دو دسته را با ماکزیمم تفاوت ممکن از هم جدا کند.


LDA Separating Line
x1
x2
4
3
2
1
0
−1
−2
−3
−4
−3
−2
−1
0
1
2
3