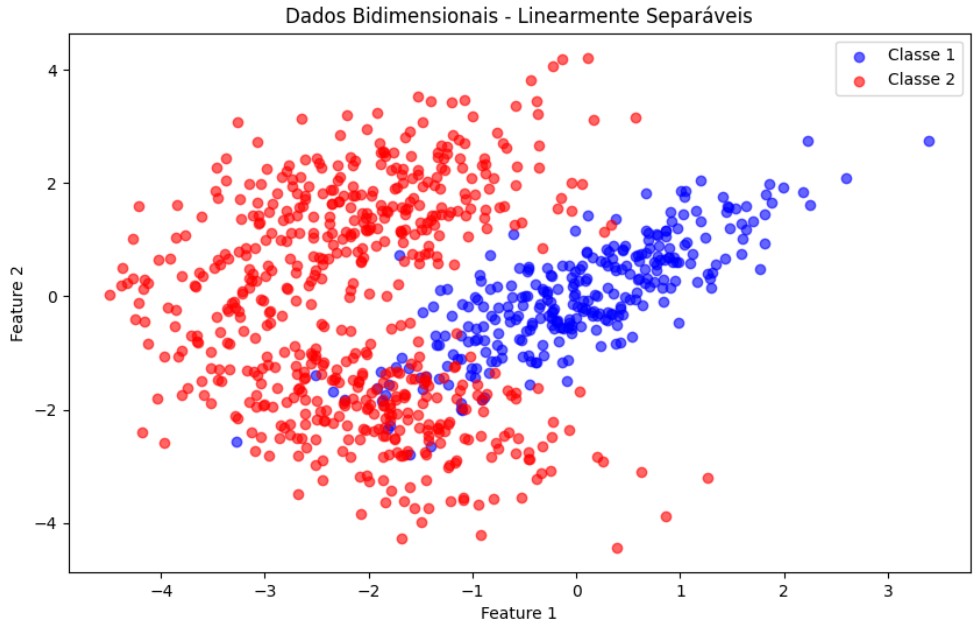
Exercício: Treinamento e Comparação de Classificadores LDA e QDA

Parte 1: Gerando os Dados

1. **Gere um conjunto de dados bidimensional com duas classes**.
    - O conjunto de dados deve ter a seguinte forma



2. **Visualize os dados**.
    - Crie um gráfico de dispersão onde cada classe tenha uma cor diferente como indicado acima.

O código abaixo mostra como criar estes dados. Repare que a variável n define o número de amostras. A classe 2 terá o dobro de amostras que a classe 1.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import multivariate_normal

# Definindo médias e matrizes de covariância para as duas classes
mean_class1 = [0, 0]
mean_class2 = [-2, -2]
mean_class2b = [-2, 1.5]

cov_matrix_class1 = [[1, 0.8], [0.8, 1]] # Matriz de covariância da classe 1
cov_matrix_class2 = [[1, -0.6], [-0.6, 1]] # Matriz de covariância da classe 2
cov_matrix_class2b = [[1, 0.6], [0.6, 1]] # Matriz de covariância da classe 2b

# Gerando dados para cada classe
n=300
np.random.seed(42)
```

```
data_class1 = np.random.multivariate_normal(mean_class1, cov_matrix_class1, n)
data_class2 = np.random.multivariate_normal(mean_class2, cov_matrix_class2, n)
data_class2b = np.random.multivariate_normal(mean_class2b, cov_matrix_class2b,
n)
data_class2 = np.vstack((data_class2, data_class2b))

# Visualizando os dados e as médias
plt.figure(figsize=(10, 6))
plt.scatter(data_class1[:, 0], data_class1[:, 1], color='blue', label='Classe
1', alpha=0.6)
plt.scatter(data_class2[:, 0], data_class2[:, 1], color='red', label='Classe
2', alpha=0.6)
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.title('Dados Bidimensionais - Linearmente Separáveis')
plt.legend()
plt.show()
```

Parte 2: Treinamento e Avaliação dos Classificadores

1. **Divida o conjunto de dados em treino e teste**, com aproximadamente 70% dos dados para o treino e 30% para o teste.

```
from sklearn.model_selection import train_test_split

# Concatenando dados para as classes
X = np.vstack((data_class1, data_class2))
y = np.hstack((np.zeros(n), np.ones(2*n))) # Classe 0 para data_class1,
Classe 1 para data_class2


# Divisão em treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)
```

2. **Treine um classificador LDA** com os dados de treino e use-o para prever as classes dos dados de teste.
    - Calcule a acurácia do classificador no conjunto de teste.
    - Plote a fronteira de decisão do LDA no gráfico de dispersão.
3. **Treine um classificador QDA** com os mesmos dados de treino e use-o para prever as classes dos dados de teste.
    - Calcule a acurácia do classificador QDA no conjunto de teste.

- Plote a fronteira de decisão do QDA no mesmo gráfico, para que possam ser comparadas.

O código abaixo mostra como treinar o LDA e o QDA

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis,
QuadraticDiscriminantAnalysis

# Treinamento do classificador LDA
lda = LinearDiscriminantAnalysis()
lda.fit(X_train, y_train)

# Treinamento do classificador QDA
qda = QuadraticDiscriminantAnalysis()
qda.fit(X_train, y_train)
```

O código abaixo mostra como calcular a acurácia dos modelos e como plotar a fronteira de classificação.

```
from sklearn.metrics import accuracy_score

# Previsões
y_pred_lda = lda.predict(X_test)
y_pred_qda = qda.predict(X_test)

# Avaliação das precisões
accuracy_lda = accuracy_score(y_test, y_pred_lda)
accuracy_qda = accuracy_score(y_test, y_pred_qda)
print(f'Precisão LDA: {accuracy_lda:.2f}')
print(f'Precisão QDA: {accuracy_qda:.2f}')

# Visualização dos dados e fronteiras de decisão
plt.figure(figsize=(12, 6))

# Gráfico 1: Fronteira de decisão LDA
plt.subplot(1, 2, 1)
plt.scatter(data_class1[:, 0], data_class1[:, 1], color='blue',
label='Classe 1', alpha=0.6)
plt.scatter(data_class2[:, 0], data_class2[:, 1], color='red',
label='Classe 2', alpha=0.6)

# Criando a grade para a fronteira de decisão LDA
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.linspace(x_min, x_max, 100), np.linspace(y_min,
y_max, 100))
Z_lda = lda.predict(np.c_[xx.ravel(), yy.ravel()])
Z_lda = Z_lda.reshape(xx.shape)
```

```
# Plotando a fronteira de decisão LDA
plt.contourf(xx, yy, Z_lda, alpha=0.2)
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.title('Fronteira de Decisão LDA')
plt.legend()
plt.axis('equal')

# Gráfico 2: Fronteira de decisão QDA
plt.subplot(1, 2, 2)
plt.scatter(data_class1[:, 0], data_class1[:, 1], color='blue',
label='Classe 1', alpha=0.6)
plt.scatter(data_class2[:, 0], data_class2[:, 1], color='red',
label='Classe 2', alpha=0.6)

# Criando a grade para a fronteira de decisão QDA
Z_qda = qda.predict(np.c_[xx.ravel(), yy.ravel()])
Z_qda = Z_qda.reshape(xx.shape)

# Plotando a fronteira de decisão QDA
plt.contourf(xx, yy, Z_qda, alpha=0.2)
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.title('Fronteira de Decisão QDA')
plt.legend()
plt.axis('equal')

plt.tight_layout()
plt.show()
```

Parte 3: Análise e Discussão

1. Gere as curvas ROC para cada modelo treinado sem usar bibliotecas. Você deve variar na mão o limiar e calcular os valores de TP e FP e plotar a curva para cada classificador (LDA e QDA) . Você vai usar a função de discriminantes fornecida pelo método de treinamento e calcular a classificação no conjunto de testes variando o limiar. O código abaixo mostra como fazer isso para o LDA

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

# Treinamento do modelo LDA
lda = LinearDiscriminantAnalysis()
lda.fit(X_train, y_train)
```

```
# Obtenção da função discriminante (valores para a classe 1)
discriminant_scores = lda.decision_function(X_test)

# Limiar de decisão variando para cálculo da curva ROC manual
thresholds = np.linspace(min(discriminant_scores),
max(discriminant_scores), 100)
tpr_list = [] # Taxa de Verdadeiros Positivos
fpr_list = [] # Taxa de Falsos Positivos

# Cálculo de TPR e FPR para cada limiar
for threshold in thresholds:
y_pred = (discriminant_scores >= threshold).astype(int) # Classe 1 se score
>= threshold
# Taxa de Verdadeiros Positivos (TPR)
tp = np.sum((y_pred == 1) & (y_test == 1))
fn = np.sum((y_pred == 0) & (y_test == 1))
tpr = tp / (tp + fn)
tpr_list.append(tpr)
# Taxa de Falsos Positivos (FPR)
fp = np.sum((y_pred == 1) & (y_test == 0))
tn = np.sum((y_pred == 0) & (y_test == 0))
fpr = fp / (fp + tn)
fpr_list.append(fpr)

# Plot da Curva ROC
plt.figure(figsize=(8, 6))
plt.plot(fpr_list, tpr_list, color='blue', label='Curva ROC (Manual)')
plt.plot([0, 1], [0, 1], 'k--', label='Classificador Aleatório')
plt.xlabel('Taxa de Falsos Positivos (FPR)')
plt.ylabel('Taxa de Verdadeiros Positivos (TPR)')
plt.title('Curva ROC (LDA - Calculada Manualmente)')
plt.legend()
plt.grid(True)
plt.show()
```

Para o caso do QDA basta substituir a função de discriminante como mostrado abaixo

```
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis

# Treinamento do modelo QDA
qda = QuadraticDiscriminantAnalysis()
qda.fit(X_train, y_train)

# Obtenção da função discriminante (valores para a classe 1)
discriminant_scores = qda.decision_function(X_test)

# Limiar de decisão variando para cálculo da curva ROC manual
thresholds = np.linspace(min(discriminant_scores),
```

```
max(discriminant_scores), 100)
tpr_list_qda = [] # Taxa de Verdadeiros Positivos
fpr_list_qda = [] # Taxa de Falsos Positivos

# Cálculo de TPR e FPR para cada limiar
for threshold in thresholds:
y_pred = (discriminant_scores >= threshold).astype(int) # Classe 1 se score
>= threshold
# Taxa de Verdadeiros Positivos (TPR)
tp = np.sum((y_pred == 1) & (y_test == 1))
fn = np.sum((y_pred == 0) & (y_test == 1))
tpr = tp / (tp + fn) if (tp + fn) > 0 else 0
tpr_list_qda.append(tpr)
# Taxa de Falsos Positivos (FPR)
fp = np.sum((y_pred == 1) & (y_test == 0))
tn = np.sum((y_pred == 0) & (y_test == 0))
fpr = fp / (fp + tn) if (fp + tn) > 0 else 0
fpr_list_qda.append(fpr)

# Plot da Curva ROC
plt.figure(figsize=(8, 6))
plt.plot(fpr_list_qda, tpr_list_qda, color='green', label='Curva ROC (QDA -
Manual)')
plt.plot([0, 1], [0, 1], 'k--', label='Classificador Aleatório')
plt.xlabel('Taxa de Falsos Positivos (FPR)')
plt.ylabel('Taxa de Verdadeiros Positivos (TPR)')
plt.title('Curva ROC (QDA - Calculada Manualmente)')
plt.legend()
plt.grid(True)
plt.show()
```

2. Compare as precisões dos classificadores LDA e QDA no conjunto de teste e compare as curvas ROC (plote as curvas num mesmo gráfico para comparação). Responda às perguntas:
    - Qual dos dois classificadores teve melhor desempenho em termos de precisão?
    - Com base nas fronteiras de decisão, qual parece melhor ajustado aos dados? Por quê?
3. Discuta em que tipos de problemas (além deste) seria vantajoso utilizar LDA em vez de QDA e vice-versa, considerando as suposições e diferenças entre os dois métodos.
4. Mude o centro da classe 1 para (3,3), refaça os gráficos anteriores e analise o que aconteceu e tire suas conclusões.

O relatório em pdf não deve conter códigos e deve mostrar tudo o que foi pedido no exercício acima.