Exercício 04 - Reconhecimento de Padrões

Aluno: Giovanni Martins de Sá Júnior - 2017001850

1 de outubro de 2023

Neste quarto, exercício, será implementado um Classificador de Bayes unidimensional, separando o exercício em diferentes etapas. Inicialmente, são criados o conjunto de dados e saparados em conjuntos de testes e treinamento. Em seguida, será feito o treinamento do classificador com o conjunto de treinamento, e finalmente, os testes. A partir disso, será verificado a precisão do classificador diante do processo de treinamento e teste.

1 Criação do conjunto de dados

```
# Importação de Bibliotecas

# Etapa 1: Criação do conjunto de dados

# Definir as características das distribuições normais
np.random.seed(0)  # Para reprodutibilidade
mean1, std1, n_samples1 = 2, 0.8, 240
mean2, std2, n_samples2 = 4, 0.4, 120

# Amostrar os dados para as duas classes
class1_data = np.random.normal(mean1, std1, n_samples1)
class2_data = np.random.normal(mean2, std2, n_samples2)
```

2 Separação dos dados de treinamento e teste

```
# Etapa 2: Separar os dados em treinamento e teste

x = np.concatenate((class1_data, class2_data), axis=0)
y = np.concatenate((np.zeros(n_samples1), np.ones(n_samples2)), axis=0)
xTrain, xTest, yTrain, yTest = train_test_split(x, y, test_size=0.10, random_state=42)
```

3 Treinamento do Classificador

```
# Etapa 3: Treinar o classificador Gaussian Naive Bayes
classifier = GaussianNB()
classifier.fit(xTrain.reshape(-1, 1), yTrain)
```

4 Aplicação do Classificador

```
# Etapa 4: Aplicar o classificador ao conjunto de teste
yPred = classifier.predict(xTest.reshape(-1, 1))

def print_classification_report(yTest, yPred):
    accuracy = accuracy_score(yTest, yPred) * 100

    cm = confusion_matrix(yTest, yPred)
    report = classification_report(yTest, yPred)

    print('Relatório de Classificação:')
    print(f'Percentual de acertos: {accuracy:.2f}%')
    print(f'Percentual de erros: {100 - accuracy:.2f}%\n')

    print('Matriz de Confusão:')
    print(cm)

    print('\nRelatório de Classificação:')
    print(report)

print_classification_report(yTest, yPred)
```

5 Resultados

5.1 Precisão do Modelo

1. Precisão da Classificação:
   (a) Percentual de acertos: 94.44 %
   (b) Percentual de erros: 5.56 %
2. Matriz de Confusão: $\begin{bmatrix} 22 & 1 \\ 1 & 12 \end{bmatrix}$
3. Dados:

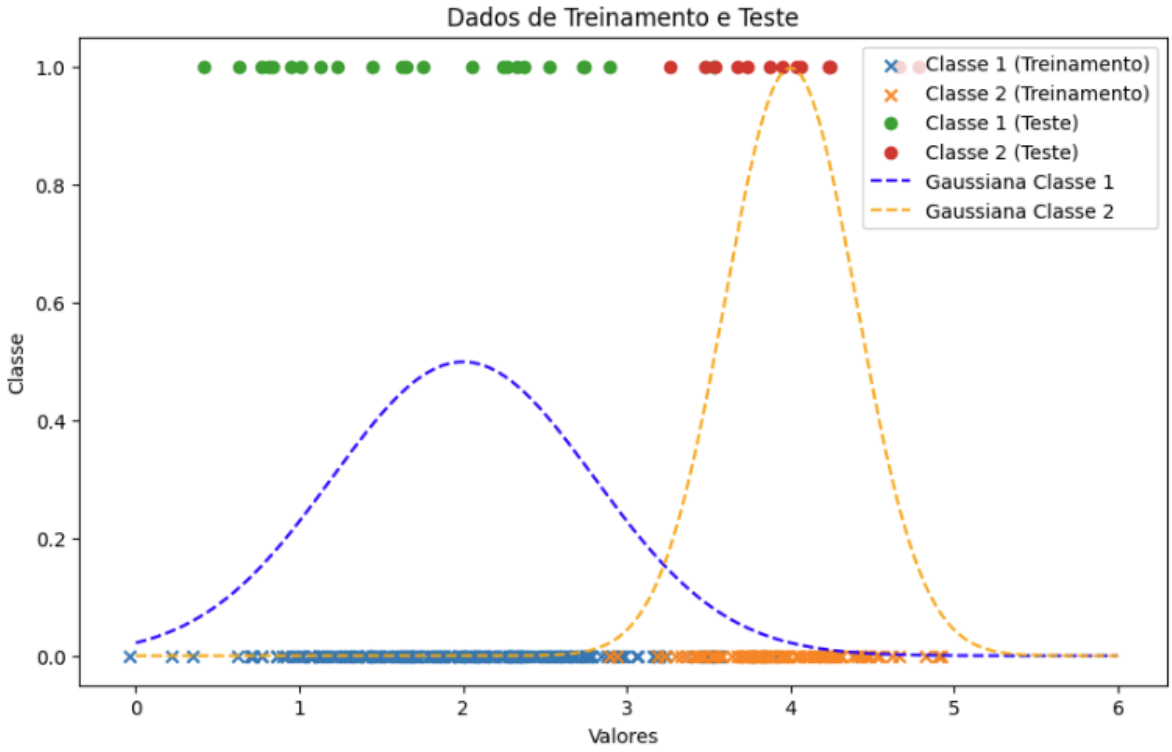
Output	Precision	Recall	F1-Score	Support
0.0	0.96	0.96	0.96	23
1.0	0.92	0.92	0.92	13

5.2 Gráfico com os resultados

```
# Plotar as gaussianas no gráfico
x = np.linspace(0, 6, 1000)
plt.figure(figsize=(10, 6))
plt.scatter(xTrain[yTrain == 0], np.zeros_like(xTrain[yTrain == 0]), marker='x',
label='Classe 1 (Treinamento)')
plt.scatter(xTrain[yTrain == 1], np.zeros_like(xTrain[yTrain == 1]), marker='x',
label='Classe 2 (Treinamento)')
plt.scatter(xTest[yTest == 0], np.ones_like(xTest[yTest == 0]), marker='o',
label='Classe 1 (Teste)')
plt.scatter(xTest[yTest == 1], np.ones_like(xTest[yTest == 1]), marker='o',
label='Classe 2 (Teste)')

# Plotar as gaussianas
plt.plot(x, norm.pdf(x, mean1, std1), label='Gaussiana Classe 1', linestyle='--', color='blue')
plt.plot(x, norm.pdf(x, mean2, std2), label='Gaussiana Classe 2', linestyle='--', color='orange')

plt.legend()
plt.title('Dados de Treinamento e Teste')
plt.xlabel('Valores')
plt.ylabel('Classe')
plt.show()
```


Dados de Treinamento e Teste
Classe 1 (Treinamento)
Classe 2 (Treinamento)
Classe 1 (Teste)
Classe 2 (Teste)
Gaussiana Classe 1
Gaussiana Classe 2
Classe
Valores
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
5
6