Projeto Integrador Parte D

Alunas:

- **Gabriella Braz**
- **Giovana Ribeiro**

1. **Use o mesmo conjunto de dados já escolhido anteriormente ou escolha um novo conjunto de dados.**

```
[96]
    df = pd.read_csv("data/tb_1.csv")
    print(df.dtypes)
```

```
gender                       object
race_ethnicity               object
parental_level_of_education  object
lunch                        object
test_preparation_course      object
math_score                    int64
reading_score                 int64
writing_score                 int64
dtype: object
```

2. **Implemente o algoritmo KNN para classificação.**

```
# # Projeto Integrador Parte B - Preparação dos Dados
#
# Entregas:
# 1) Faça um relatório respondendo cada pergunta separadamente.
# 2) Link para a base utilizada.
# 3) Código completo em Python.
#
# Dando continuidade ao Projeto Integrador - Parte A, faça uma análise dos mesmos dados utilizados anteriormente, respondendo às seguintes questões:
#

# ### ALUNAS
# - Gabriella Braz
# - Giovana Ribeiro

# Importar bibliotecas necessárias
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import (
roc_auc_score,
accuracy_score,
precision_score,
recall_score,
```

```
f1_score,
confusion_matrix,
)
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

df = pd.read_csv("data/tb_1.csv")
print(df.dtypes)

df["target"] = (
(df["math_score"] + df["reading_score"] + df["writing_score"]) / 3 >= 60
).astype(int)

# Separar preditores e alvo
X = df.drop("target", axis=1)
y = df["target"]

# Transformar variáveis categóricas em dummies
X = pd.get_dummies(X, drop_first=True)

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

# Treinar modelo KNN
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)

# Previsões
y_pred = knn.predict(X_test)
y_pred_proba = knn.predict_proba(X_test)[:, 1]

# AUC-ROC (como é binário, podemos calcular diretamente)
auc_roc = roc_auc_score(y_test, y_pred_proba)

# Acurácia
accuracy = accuracy_score(y_test, y_pred)
# Precision
precision = precision_score(y_test, y_pred, average="weighted")
# Recall
recall = recall_score(y_test, y_pred, average="weighted")
# F1-Score
f1 = f1_score(y_test, y_pred, average="weighted")
```

```python
# Exibir resultados
print(f"AUC-ROC Score: {auc_roc:.4f}")
print(f"Acurácia: {accuracy:.4f}")
print(f"Precisão: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1-Score: {f1:.4f}")

# Matriz de confusão
cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(6, 5))
sns.heatmap(
cm,
annot=True,
fmt="d",
cmap="Blues",
xticklabels=["Reprovado", "Aprovado"],
yticklabels=["Reprovado", "Aprovado"],
)
plt.title("Matriz de Confusão")
plt.xlabel("Previsão")
plt.ylabel("Classe Real")
plt.show()
```

3. **Analise a acurácia, precisão, recall, f1_score e área sob a curva roc.**

```python
# AUC-ROC (como é binário, podemos calcular diretamente)
auc_roc = roc_auc_score(y_test, y_pred_proba)

# Acurácia
accuracy = accuracy_score(y_test, y_pred)
# Precision
precision = precision_score(y_test, y_pred, average="weighted")
# Recall
recall = recall_score(y_test, y_pred, average="weighted")
# F1-Score
f1 = f1_score(y_test, y_pred, average="weighted")

# Exibir resultados
print(f"AUC-ROC Score: {auc_roc:.4f}")
print(f"Acurácia: {accuracy:.4f}")
print(f"Precisão: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1-Score: {f1:.4f}")
```

```
[31]
...  AUC-ROC Score: 0.9995
     Acurácia: 0.9833
     Precisão: 0.9834
     Recall: 0.9833
     F1-Score: 0.9834
```

Os resultados do modelo KNN indicam um desempenho excelente na classificação dos alunos como aprovados ou reprovados. A acurácia de 98,33% mostra que o modelo acerta a grande maioria das previsões, e os valores elevados de precisão (98,34%) e recall

(98,33%) indicam que ele consegue identificar corretamente tanto os alunos aprovados quanto os reprovados, com mínima ocorrência de falsos positivos ou falsos negativos. O F1-Score igualmente alto (98,34%) confirma o equilíbrio entre precisão e recall, reforçando a confiabilidade do modelo.

A AUC-ROC próxima de 1 (0,9995) evidencia que o modelo tem uma capacidade quase perfeita de distinguir entre as duas classes. A matriz de confusão complementa essa análise, mostrando que houve apenas 2 falsos positivos e 3 falsos negativos em 300 amostras, o que significa que os erros de classificação são extremamente baixos. Esses resultados indicam que o KNN, com as variáveis categóricas transformadas em dummies, é muito eficaz para este conjunto de dados.