Esse algoritmo reúne um conjunto de funções auxiliares para ajudar na análise e na avaliação de modelos de classificação, principalmente em cenários onde se trabalha com dados de previsão de resultados binários (por exemplo, “sobreviveu” ou “não sobreviveu”). Em outras palavras, ele facilita tanto a visualização exploratória dos dados quanto a avaliação do desempenho de um modelo preditivo. Vamos entender, de forma geral para o particular, cada parte do código:

**Contexto Geral**

No campo de *Data Science* e *Machine Learning*, após preparar os dados e treinar um modelo de classificação, é fundamental:

* **Explorar os dados** para entender a distribuição das variáveis e identificar possíveis problemas (como dados faltantes ou variáveis com muitos níveis);
* **Avaliar o desempenho do modelo** utilizando métricas (acurácia, AUC, etc.) e visualizações (matriz de confusão, curva ROC);
* **Diagnosticar e comparar as previsões** do modelo com os dados reais para identificar discrepâncias e ajustar o modelo se necessário.

O algoritmo que você apresentou implementa funções que auxiliam nesses processos.

**Análise dos Blocos de Código**

**1. Importações e Configurações Iniciais**

import pandas as pd

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix, balanced\_accuracy\_score, roc\_auc\_score, roc\_curve

import seaborn as sns

import matplotlib.pyplot as plt

**O que faz:**

* **Pandas:** Manipulação e análise de dados em estruturas tipo DataFrame.
* **Sklearn Metrics:** Cálculo de diversas métricas de avaliação para modelos de classificação.
* **Seaborn e Matplotlib:** Geração de gráficos e visualizações dos dados e dos resultados dos modelos.

**2. Função descritiva**

def descritiva(df\_, var, vresp='survived', max\_classes=5):

"""

Gera um gráfico descritivo da taxa de sobreviventes por categoria da variável especificada.

"""

df = df\_.copy()

if df[var].nunique() > max\_classes:

df[var] = pd.qcut(df[var], max\_classes, duplicates='drop')

fig, ax1 = plt.subplots(figsize=(10, 6))

sns.pointplot(data=df, y=vresp, x=var, ax=ax1)

# Criar o segundo eixo y para a taxa de sobreviventes

ax2 = ax1.twinx()

sns.countplot(data=df, x=var, palette='viridis', alpha=0.5, ax=ax2)

ax2.set\_ylabel('Frequência', color='blue')

ax2.tick\_params(axis='y', labelcolor='blue')

ax1.set\_zorder(2)

ax1.patch.set\_visible(False)

plt.show()

**O que faz e como funciona:**

* **Objetivo Geral:**  
  Visualizar, para uma variável categórica (ou transformada em categorias), como a taxa de sobrevivência varia entre as categorias e, ao mesmo tempo, ver a distribuição (frequência) dos dados.
* **Detalhamento:**
  + **Preparação dos dados:**  
    Se a variável tiver mais categorias que o permitido (max\_classes), ela é transformada em categorias utilizando *quantile binning* (pd.qcut). Isso ajuda a resumir variáveis contínuas em poucos grupos.
  + **Gráfico com dois eixos y:**
    - O **primeiro eixo** (ax1) usa um *pointplot* para mostrar a taxa de sobreviventes (vresp) em cada categoria da variável var.
    - O **segundo eixo** (ax2) usa um *countplot* para exibir a frequência de registros em cada categoria.
  + **Resultado:**  
    Um gráfico que, de forma simultânea, apresenta a taxa de sobrevivência (linha/ pontos) e a quantidade de observações (barras), facilitando a interpretação dos dados.

**3. Função avalia\_clf**

def avalia\_clf(clf, y, X, rótulos\_y=['Não Sobreviveu', 'Sobreviveu'], base='treino'):

# Calcular as classificações preditas

pred = clf.predict(X)

# Calcular a probabilidade de evento

y\_prob = clf.predict\_proba(X)[:, -1]

# Calculando acurácia e matriz de confusão

cm = confusion\_matrix(y, pred)

ac = accuracy\_score(y, pred)

bac = balanced\_accuracy\_score(y, pred)

print(f'\nBase de {base}:')

print(f'A acurácia da árvore é: {ac:.1%}')

print(f'A acurácia balanceada da árvore é: {bac:.1%}')

# Calculando AUC

auc\_score = roc\_auc\_score(y, y\_prob)

print(f"AUC-ROC: {auc\_score:.2%}")

print(f"GINI: {(2\*auc\_score-1):.2%}")

# Visualização gráfica

sns.heatmap(cm,

annot=True, fmt='d', cmap='viridis',

xticklabels=rótulos\_y,

yticklabels=rótulos\_y)

# Relatório de classificação do Scikit

print('\n', classification\_report(y, pred))

# Gerar a Curva ROC

fpr, tpr, thresholds = roc\_curve(y, y\_prob)

# Plotar a Curva ROC

plt.figure(figsize=(8, 6))

plt.plot(fpr, tpr, color='blue', label=f'Curva ROC (AUC = {auc\_score:.2f})')

plt.plot([0, 1], [0, 1], color='red', linestyle='--')

plt.xlabel("Taxa de Falsos Positivos (FPR)")

plt.ylabel("Taxa de Verdadeiros Positivos (TPR)")

plt.title(f"Curva ROC - base de {base}")

plt.legend(loc="lower right")

plt.grid()

plt.show()

**O que faz e como funciona:**

* **Objetivo Geral:**  
  Avaliar o desempenho de um modelo classificador utilizando diversas métricas e visualizações.
* **Detalhamento:**
  + **Predição e Probabilidades:**  
    A função utiliza o classificador (clf) para prever os resultados (pred) e obter as probabilidades de ocorrência do evento (por exemplo, a chance de sobreviver).
  + **Cálculo de Métricas:**
    - **Matriz de confusão:** Mostra os acertos e erros do modelo.
    - **Acurácia e Acurácia Balanceada:** Indicadores de desempenho geral, considerando ou corrigindo o desbalanceamento de classes.
    - **AUC-ROC e GINI:** Avaliam a capacidade do modelo em discriminar entre as classes.
  + **Visualizações:**
    - Um **heatmap** da matriz de confusão para facilitar a interpretação visual dos erros e acertos.
    - A **curva ROC**, que mostra a taxa de verdadeiros positivos versus falsos positivos, auxiliando a entender o desempenho do modelo em diferentes limiares.
  + **Relatório de Classificação:**  
    Imprime um relatório detalhado com métricas como precisão, recall e F1-score para cada classe.

**4. Função relatorio\_missing**

def relatorio\_missing(df):

print(f'Número de linhas: {df.shape[0]} | Número de colunas: {df.shape[1]}')

return pd.DataFrame({'Pct\_missing': df.isna().mean().apply(lambda x: f"{x:.1%}"),

'Freq\_missing': df.isna().sum().apply(lambda x: f"{x:,.0f}").replace(',','.')})

**O que faz e como funciona:**

* **Objetivo Geral:**  
  Gerar um relatório sobre a quantidade e a proporção de dados faltantes (missing values) em cada coluna do DataFrame.
* **Detalhamento:**
  + Exibe o número total de linhas e colunas.
  + Cria e retorna um novo DataFrame onde:
    - **Pct\_missing:** Mostra a porcentagem de valores ausentes em cada coluna.
    - **Freq\_missing:** Indica a quantidade absoluta de valores faltantes, formatada para melhor visualização.

Essa função é essencial na etapa de *data cleaning*, pois ajuda a identificar variáveis que podem precisar de tratamento especial devido à alta taxa de missing.

**5. Função diagnóstico**

def diagnóstico(df\_, var, vresp='survived', pred='pred', max\_classes=5):

"""

Gera um gráfico descritivo da taxa de sobreviventes por categoria da variável especificada,

comparando valores reais e preditos.

"""

df = df\_.copy()

if df[var].nunique() > max\_classes:

df[var] = pd.qcut(df[var], max\_classes, duplicates='drop')

fig, ax1 = plt.subplots(figsize=(10, 6))

sns.pointplot(data=df, y=vresp, x=var, ax=ax1)

sns.pointplot(data=df, y=pred, x=var, ax=ax1, color='red', linestyles='--', ci=None)

ax2 = ax1.twinx()

sns.countplot(data=df, x=var, palette='viridis', alpha=0.5, ax=ax2)

ax2.set\_ylabel('Frequência', color='blue')

ax2.tick\_params(axis='y', labelcolor='blue')

ax1.set\_zorder(2)

ax1.patch.set\_visible(False)

plt.show()

**O que faz e como funciona:**

* **Objetivo Geral:**  
  Além de exibir a taxa de sobrevivência para cada categoria de uma variável, essa função compara os valores reais (observados) com os valores preditos pelo modelo.
* **Detalhamento:**
  + **Preparação dos dados:**  
    Assim como na função descritiva, se o número de categorias for alto, a variável é dividida em grupos utilizando *pd.qcut*.
  + **Gráficos:**
    - Um *pointplot* exibe a taxa de sobrevivência real (vresp).
    - Outro *pointplot*, sobreposto e em cor vermelha com linha tracejada, mostra os valores preditos (pred).
    - Um *countplot* no segundo eixo y mostra a frequência dos registros para cada categoria.
  + **Resultado:**  
    O gráfico permite comparar visualmente o desempenho do modelo em termos de previsão versus o comportamento real dos dados.

**Conclusão**

Resumindo, esse algoritmo é um conjunto de ferramentas para:

* **Visualização Exploratória:**  
  Entender a distribuição das variáveis e como elas se relacionam com o resultado (ex.: sobrevivência).
* **Avaliação de Modelos:**  
  Calcular métricas importantes e gerar gráficos (como a curva ROC e a matriz de confusão) para avaliar a qualidade de um classificador.
* **Diagnóstico e Limpeza:**  
  Identificar dados faltantes e comparar as previsões do modelo com os dados reais.

Essas funções são particularmente úteis em projetos de *machine learning* que envolvem análise de dados e a necessidade de interpretar e validar modelos de classificação de forma prática e visual.