In [142]:

```python
import numpy as np
import math
```

CLASSIFICAÇÃO

Matriz confusão

A **matriz de confusão** é uma tabela usada para avaliar o desempenho de um modelo de classificação. Ela é composta por quatro valores principais:

- **TP (True Positive):** O número de casos em que o modelo previu corretamente a classe positiva (1) e os valores reais também eram positivos (1).
- **TN (True Negative):** O número de casos em que o modelo previu corretamente a classe negativa (0) e os valores reais também eram negativos (0).
- **FP (False Positive):** O número de casos em que o modelo previu incorretamente a classe positiva (1) quando os valores reais eram negativos (0).
- **FN (False Negative):** O número de casos em que o modelo previu incorretamente a classe negativa (0) quando os valores reais eram positivos (1).

A matriz de confusão é representada da seguinte forma:

$$\begin{bmatrix} \text{Verdadeiros Positivos (TP)} & \text{Falsos Positivos (FN)} \\ \text{Falsos Negativos (FP)} & \text{Verdadeiros Negativos (TN)} \end{bmatrix}$$

Essa matriz fornece uma representação visual das previsões corretas e incorretas feitas pelo modelo em relação às classes positivas e negativas, permitindo uma avaliação rápida da quantidade de acertos e erros.

In [143]:

```python
def matriz_confusao(y_true, y_pred):
    # Verifica se as listas têm o mesmo comprimento
    if len(y_true) != len(y_pred):
        raise ValueError("As listas y_true e y_pred devem ter o mesmo comprimento.")

    # Transformando as listas em arrays NumPy para facilitar os cálculos
    y_true = np.array(y_true)
    y_pred = np.array(y_pred)

    # Calcula a matriz de confusão
    TP = np.sum((y_true == 1) & (y_pred == 1))
    TN = np.sum((y_true == 0) & (y_pred == 0))
    FP = np.sum((y_true == 0) & (y_pred == 1))
    FN = np.sum((y_true == 1) & (y_pred == 0))

    # Retorna a matriz de confusão como uma matriz 2x2
    matriz_confusao = np.array([[TP, FP], [FN, TN]])

    return matriz_confusao
```

In [144]:

```python
print("1 negativo verdadeiro, 1 falso positivo e 7 falsos negativos:\n")
y_true = [0,1,1,1,1,1,1,1,0]
y_pred = [0,0,0,0,0,0,0,0,1]

print(matriz_confusao(y_true,y_pred))
print("\n")


print("2 falsos negativos e 7 positivos verdadeiros\n")
y_true = [1,1,1,1,1,1,1,1,1]
y_pred = [0,1,1,1,1,1,1,1,0]
print(matriz_confusao(y_true,y_pred))
```

```
1 negativo verdadeiro, 1 falso positivo e 7 falsos negativos:

[[0 1]
 [7 1]]


2 falsos negativos e 7 positivos verdadeiros

[[7 0]
 [2 0]]
```

Acurácia

A **acurácia** é uma métrica usada na avaliação de modelos de classificação. Ela mede a proporção de previsões corretas em relação ao total de previsões feitas por um modelo.

A fórmula da acurácia é dada por:

$$\text{Acuracia} = \frac{TP + TN}{TP + TN + FP + FN}$$

A acurácia representa a porcentagem de previsões corretas feitas por um modelo em relação ao total de previsões realizadas. A acurácia quantifica a precisão geral do modelo e é expressa como uma porcentagem, indicando o quão bem o modelo acertou as previsões em relação ao conjunto de dados de teste. Quanto maior a acurácia, maior a proporção de acertos do modelo.

In [145]:

```python
def acuracia(y_true, y_pred):
    # Verifica se as listas têm o mesmo comprimento
    if len(y_true) != len(y_pred):
        raise ValueError("As listas y_true e y_pred devem ter o mesmo comprimento.")

    # Transformando as listas em arrays NumPy para facilitar os cálculos
    y_true = np.array(y_true)
    y_pred = np.array(y_pred)

    # Verifica quantas previsões coincidem com os valores reais
    corretas = np.sum(y_true == y_pred)

    # Calcula a acurácia como a proporção de previsões corretas
    acuracia = corretas / len(y_true)

    return round(acuracia * 100, 2)
```

In [146]:

```python
print("1 negativo verdadeiro, 1 falso positivo e 7 falsos negativos:\n")
y_true = [0,1,1,1,1,1,1,1,0]
y_pred = [0,0,0,0,0,0,0,0,1]

print(acuracia(y_true,y_pred))
print(acuracia(y_true,y_pred),"% de acurácia", sep='')

print("\n")

print("2 falsos negativos e 7 positivos verdadeiros\n")
y_true = [1,1,1,1,1,1,1,1,1]
y_pred = [0,1,1,1,1,1,1,1,0]

print(acuracia(y_true,y_pred))
print(acuracia(y_true,y_pred),"% de acurácia", sep='')
```

```
1 negativo verdadeiro, 1 falso positivo e 7 falsos negativos:

11.11
11.11% de acurácia


2 falsos negativos e 7 positivos verdadeiros

77.78
77.78% de acurácia
```

Recall

Recall mede a proporção de exemplos positivos reais que foram corretamente identificados pelo modelo. Ele é calculado por:

$\text{Recall} = \frac{\text{TP}}{\text{TP}+\text{FN}}$

In [147]:

```python
def recall(y_true, y_pred):
    # Verifica se as listas têm o mesmo comprimento
    if len(y_true) != len(y_pred):
        raise ValueError("As listas y_true e y_pred devem ter o mesmo comprimento.")

    # Transformando as listas em arrays NumPy para facilitar os cálculos
    y_true = np.array(y_true)
    y_pred = np.array(y_pred)

    TP = np.sum((y_true == 1) & (y_pred == 1))
    FN = np.sum((y_true == 1) & (y_pred == 0))

    recall = TP/(TP+FN)

    return recall
```

In [148]:

```python
print("2 falsos negativos e 7 positivos verdadeiros\n")
y_true = [1,1,1,1,1,1,1,1,1]
y_pred = [0,1,1,1,1,1,1,1,0]

print("Recall =",recall(y_true,y_pred))
```

```
2 falsos negativos e 7 positivos verdadeiros

Recall = 0.7777777777777778
```

Precision

A precisão é uma métrica de avaliação em problemas de classificação que mede a proporção de previsões positivas corretas feitas pelo modelo em relação ao total de previsões positivas feitas.

A fórmula para calcular a precisão:

$\text{Precisão} = \frac{\text{TP}}{\text{TP} + \text{FP}}$

A precisão é uma métrica importante em cenários em que é crítico evitar falsos positivos.

In [149]:

```python
def precision(y_true, y_pred):
    # Verifica se as listas têm o mesmo comprimento
    if len(y_true) != len(y_pred):
        raise ValueError("As listas y_true e y_pred devem ter o mesmo comprimento.")

    # Transformando as listas em arrays NumPy para facilitar os cálculos
    y_true = np.array(y_true)
    y_pred = np.array(y_pred)

    TP = np.sum((y_true == 1) & (y_pred == 1))
    FP = np.sum((y_true == 0) & (y_pred == 1))


    precision = TP/(TP+FP)

    return precision
```

In [150]:

```python
print("2 falsos negativos e 7 positivos verdadeiros\n")
y_true = [1,1,1,1,1,1,1,1,1]
y_pred = [0,1,1,1,1,1,1,1,0]

print("Precisão =",precision(y_true,y_pred))
```

```
2 falsos negativos e 7 positivos verdadeiros

Precisão = 1.0
```

F1

O F1 é calculado fazendo uma média harmônica entre a precisão (precision) e o recall (revocação), equilibrando assim a capacidade do modelo de fazer previsões corretas e sua capacidade de recuperar exemplos positivos.

$\text{F1} = 2 \cdot \frac{\text{Precisao} \cdot \text{Recall}}{\text{Precisao} + \text{Recall}}$

O F1 varia de 0 a 1, onde um valor mais alto indica um melhor desempenho do modelo em equilibrar precisão e recall. O F1 é particularmente útil em situações em que as classes estão desequilibradas e você deseja avaliar o modelo de forma abrangente.

No entanto, o F1 não leva em consideração a classificação correta de exemplos negativos (TN - Verdadeiros Negativos) e pode não ser a métrica mais adequada em todos os cenários.

In [151]:

```python
def F1(y_true, y_pred):
    R = recall(y_true, y_pred)
    P = precision(y_true, y_pred)

    F1 = 2 * ((P * R)/(P + R))

    return F1
```

In [152]:

```python
print("2 falsos negativos e 7 positivos verdadeiros\n")
y_true = [1,1,1,1,1,1,1,1,1]
y_pred = [0,1,1,1,1,1,1,1,0]

print("F1 =", F1(y_true,y_pred))
```

```
2 falsos negativos e 7 positivos verdadeiros

F1 = 0.8750000000000001
```

Matthews Correlation Coefficient (MCC)

O Matthews Correlation Coefficient (MCC) é uma métrica de avaliação de desempenho comumente utilizada em problemas de classificação binária. Ele leva em consideração os quatro elementos da matriz de confusão (verdadeiros positivos, verdadeiros negativos, falsos positivos e falsos negativos) para fornecer uma medida global da qualidade das previsões de um modelo.

$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$

O MCC varia de **-1** a **+1**:

- Um MCC de +1 indica uma correspondência perfeita entre as previsões do modelo e os valores reais, onde o modelo faz todas as previsões corretamente.
- Um MCC de 0 indica que o modelo está fazendo previsões equivalentes às de um modelo aleatório.
- Um MCC de -1 indica um desempenho inversamente perfeito, onde o modelo faz todas as previsões erradas.

In [153]:

```python
def matthews_correlation_coefficient(y_true, y_pred):
    # Verifica se as listas têm o mesmo comprimento
    if len(y_true) != len(y_pred):
        raise ValueError("As listas y_true e y_pred devem ter o mesmo comprimento.")

    # Transformando as listas em arrays NumPy para facilitar os cálculos
    y_true = np.array(y_true)
    y_pred = np.array(y_pred)

    # Calcula a matriz de confusão
    TP = np.sum((y_true == 1) & (y_pred == 1))
    TN = np.sum((y_true == 0) & (y_pred == 0))
    FP = np.sum((y_true == 0) & (y_pred == 1))
    FN = np.sum((y_true == 1) & (y_pred == 0))

    print(TP,FN)
    print(FP,TN)


    numerador = (TP * TN) - (FP * FN)
    denominador = ((TP + FP) * (TP + FN) * (TN + FP) * (TN + FN)) ** 0.5

    if denominador == 0:
        return 0.0  # Evitar divisão por zero

    mcc = numerador / denominador
    return mcc
```

In [154]:

```python
print("1 negativo verdadeiro, 1 falso positivo e 7 falsos negativos:\n")
y_true = [0,1,1,1,1,1,1,1,0]
y_pred = [0,0,0,0,0,0,0,0,1]

print("MCC =",matthews_correlation_coefficient(y_true,y_pred))

print("\n")

print("2 falsos negativos e 7 positivos verdadeiros\n")
y_true = [1,1,1,1,1,1,1,1,1]
y_pred = [0,1,1,1,1,1,1,1,0]

print("MCC =",matthews_correlation_coefficient(y_true,y_pred))

print("\n")

print("7 positivos verdadeiros e 2 negativos verdadeiros\n")
y_true = [1,1,1,0,0,1,1,1,1]
y_pred = [1,1,1,0,0,1,1,1,1]

print("MCC =",matthews_correlation_coefficient(y_true,y_pred))
```

```
1 negativo verdadeiro, 1 falso positivo e 7 falsos negativos:

0 7
1 1
MCC = -0.6614378277661476


2 falsos negativos e 7 positivos verdadeiros

7 2
0 0
MCC = 0.0


7 positivos verdadeiros e 2 negativos verdadeiros

7 0
0 2
MCC = 1.0
```

REGRESSÃO

Mean Absolute Error (Erro Médio Absoluto)

O Mean Absolute Error (MAE) é uma métrica de avaliação usada principalmente em problemas de regressão para medir o quão próximo as previsões do modelo estão dos valores reais. É uma métrica que calcula a média das diferenças absolutas entre as previsões do modelo e os valores reais.

A fórmula para calcular o MAE é a seguinte:

$MAE = \frac{1}{n} \sum_{i=1}^{n} |y_{\text{true}}^{(i)} - y_{\text{pred}}^{(i)}|$

Onde:

- n é o número total de exemplos no conjunto de dados.
- $y_{\text{true}}^{(i)}$ representa o valor real do i-ésimo exemplo.
- $y_{\text{pred}}^{(i)}$ representa a previsão do modelo para o i-ésimo exemplo.

O módulo evita o cancelamento de erros negativos e positivos, entretanto, o módulo atrapalha o cálculo do gradiente e, por isso, não é utilizado em funções de custo.

O MAE mede o erro médio absoluto das previsões do modelo em relação aos valores reais. Quanto menor o valor do MAE, melhor o desempenho do modelo, pois indica que as previsões estão mais próximas dos valores reais. Um MAE igual a 0 indica que o modelo faz previsões perfeitas.

O MAE é uma métrica útil para entender o desempenho de modelos de regressão e comparar diferentes modelos em termos de quão bem eles se ajustam aos dados observados.

In [155]:

```python
def mean_absolute_error(y_true, y_pred):
    # Verifica se as listas têm o mesmo comprimento
    if len(y_true) != len(y_pred):
        raise ValueError("As listas y_true e y_pred devem ter o mesmo comprimento.")

    # Calcula o MAE
    mae = sum(abs(y_true[i] - y_pred[i]) for i in range(len(y_true))) / len(y_true)

    return mae
```

In [156]:

```python
y_true = [10,4,7,4,7,12,4,2,0]
y_pred = [8,4,7,3,4,10,3,2,1]

print("y_true = [10,4,7,4,7,12,4,2,0]\ny_pred = [8,4,7,3,4,10,3,2,1]\nMNE =", mean_absolute_error(y_true,y_pred),"\n")

y_true = [10,4,7,4,7,12,4,2,0]
y_pred = [10,4,7,4,7,12,4,2,0]

print("y_true = [10,4,7,4,7,12,4,2,0]\ny_pred = [10,4,7,4,7,12,4,2,0]\nMNE =", mean_absolute_error(y_true,y_pred),"\n")
```

```
y_true = [10,4,7,4,7,12,4,2,0]
y_pred = [8,4,7,3,4,10,3,2,1]
MNE = 1.1111111111111112

y_true = [10,4,7,4,7,12,4,2,0]
y_pred = [10,4,7,4,7,12,4,2,0]
MNE = 0.0
```

Mean Squared Error (MSE)

O Mean Squared Error (MSE) é uma métrica de avaliação frequentemente usada em problemas de regressão para medir o erro médio quadrático das previsões em relação aos valores reais. É uma métrica importante para avaliar o quão bem um modelo de regressão está se ajustando aos dados.

$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$

Onde:

- n é o número de exemplos no conjunto de dados.
- y_i são os valores reais.
- $\hat{y}_i$ são as previsões do modelo.

O MSE calcula a média dos erros quadráticos, dando maior peso a erros maiores. Portanto, erros maiores contribuem mais para o valor total do MSE.

Quanto menor o valor do MSE, melhor, pois indica que as previsões do modelo estão mais próximas dos valores reais. Por outro lado, um valor mais alto do MSE indica que as previsões estão mais distantes dos valores reais, o que significa um desempenho pior do modelo de regressão.

O MSE é uma métrica comum usada para avaliar a qualidade de modelos de regressão, e muitas vezes é usado em conjunto com outras métricas para fornecer uma imagem completa do desempenho do modelo.

In [157]:

```python
def mean_squared_error(y_true, y_pred):

    # Verifica se as listas têm o mesmo comprimento
    if len(y_true) != len(y_pred):
        raise ValueError("As listas y_true e y_pred devem ter o mesmo comprimento.")

    # Transformando as listas em arrays NumPy para facilitar os cálculos
    y_true = np.array(y_true)
    y_pred = np.array(y_pred)

    # Calcula o MSE
    mse = sum((y_true[i] - y_pred[i])**2 for i in range(len(y_true))) / len(y_true)

    return mse
```

In [158]:

```python
y_true = [10,4,7,4,7,12,4,2,0]
y_pred = [8,4,7,3,4,10,3,2,1]

print("y_true = [10,4,7,4,7,12,4,2,0]\ny_pred = [8,4,7,3,4,10,3,2,1]\nMSE =", mean_squared_error(y_true,y_pred),"\n")

y_true = [10,4,7,4,7,12,4,2,0]
y_pred = [10,4,7,4,7,12,4,2,0]

print("y_true = [10,4,7,4,7,12,4,2,0]\ny_pred = [10,4,7,4,7,12,4,2,0]\nMSE =", mean_squared_error(y_true,y_pred),"\n")
```

```
y_true = [10,4,7,4,7,12,4,2,0]
y_pred = [8,4,7,3,4,10,3,2,1]
MSE = 2.2222222222222223

y_true = [10,4,7,4,7,12,4,2,0]
y_pred = [10,4,7,4,7,12,4,2,0]
MSE = 0.0
```

Root Mean Squared Error (RMSE)

O Root Mean Squared Error (RMSE) é uma métrica de avaliação de desempenho usada em problemas de regressão para medir o erro médio quadrático das previsões em relação aos valores reais. É uma métrica valiosa para avaliar a qualidade de modelos de regressão, pois fornece uma medida interpretável do desvio padrão dos erros das previsões.

$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2}$

Onde:

- n é o número de exemplos no conjunto de dados.
- y_i são os valores reais.
- $\hat{y}_i$ são as previsões do modelo.

O RMSE calcula a raiz quadrada da média dos erros quadráticos, dando uma medida de desvio padrão dos erros das previsões. Quanto menor o valor do RMSE, melhor, pois indica que as previsões do modelo estão mais próximas dos valores reais.

O RMSE é uma métrica útil para avaliar a qualidade de modelos de regressão e é frequentemente usado em conjunto com o MSE para fornecer uma medida mais interpretável do erro. No RMSE, os erros maiores têm um peso maior devido à raiz quadrada, tornando-o mais sensível a erros significativos.

In [159]:

```python
def root_mean_squared_error(y_true, y_pred):
    mse = mean_squared_error(y_true, y_pred)

    rmse = math.sqrt(mse)

    return rmse
```

In [160]:

```python
y_true = [10,4,7,4,7,12,4,2,0]
y_pred = [8,4,7,3,4,10,3,2,1]

print("y_true = [10,4,7,4,7,12,4,2,0]\ny_pred = [8,4,7,3,4,10,3,2,1]\nRMSE =", root_mean_squared_error(y_true,y_pred),"\n

y_true = [10,4,7,4,7,12,4,2,0]
y_pred = [10,4,7,4,7,12,4,2,0]

print("y_true = [10,4,7,4,7,12,4,2,0]\ny_pred = [10,4,7,4,7,12,4,2,0]\nRMSE =", root_mean_squared_error(y_true,y_pred),"\
```

```
y_true = [10,4,7,4,7,12,4,2,0]
y_pred = [8,4,7,3,4,10,3,2,1]
RMSE = 1.4907119849998598

y_true = [10,4,7,4,7,12,4,2,0]
y_pred = [10,4,7,4,7,12,4,2,0]
RMSE = 0.0
```

Mean Absolute Percentage Error (MAPE)

O **Mean Absolute Percentage Error (MAPE)** é uma métrica usada em problemas de regressão para avaliar o desempenho de modelos de previsão. Ele mede a porcentagem média de erro absoluto em relação aos valores reais. O MAPE é particularmente útil quando se deseja entender o erro relativo das previsões em relação aos valores reais, expressando-o como uma porcentagem.

$MAPE = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right|$

Onde:

- n é o número de exemplos no conjunto de dados.
- y_i são os valores reais.
- $\hat{y}_i$ são as previsões do modelo.

O MAPE calcula a diferença percentual média entre as previsões do modelo $\hat{y}_i$ e os valores reais y_i. Essa métrica é expressa como uma porcentagem, o que a torna facilmente interpretável.

O MAPE tem algumas limitações, incluindo a sensibilidade a valores zero nos dados reais, que podem levar a divisões por zero.

No MAPE, um valor menor indica um melhor ajuste do modelo às previsões, uma vez que representa uma menor porcentagem de erro em relação aos valores reais.

In [161]:

```python
def mean_absolute_percentage_error(y_true, y_pred):
    n = len(y_true)
    mape = (100 / n) * sum(abs((y_true[i] - y_pred[i]) / y_true[i]) for i in range(n) if y_true[i] != 0)

    return mape
```

In [162]:

```python
y_true = [10,4,7,4,7,12,4,2,1]
y_pred = [8,4,7,3,4,10,3,2,1]

print("y_true = [10,4,7,4,7,12,4,2,1]\ny_pred = [8,4,7,3,4,10,3,2,1]\nMAPE =", mean_absolute_percentage_error(y_true,y_pr

y_true = [10,4,7,4,7,12,4,2,3]
y_pred = [10,4,7,4,7,12,4,2,2]

print("y_true = [10,4,7,4,7,12,4,2,3]\ny_pred = [10,4,7,4,7,12,4,2,2]\nMAPE =", mean_absolute_percentage_error(y_true,y_p

y_true = [10,4,7,4,7,12,4,2,3]
y_pred = [10,4,7,4,7,12,4,2,3]

print("y_true = [10,4,7,4,7,12,4,2,3]\ny_pred = [10,4,7,4,7,12,4,2,3]\nMAPE =", mean_absolute_percentage_error(y_true,y_p
```

```
y_true = [10,4,7,4,7,12,4,2,1]
y_pred = [8,4,7,3,4,10,3,2,1]
MAPE = 14.391534391534393

y_true = [10,4,7,4,7,12,4,2,3]
y_pred = [10,4,7,4,7,12,4,2,2]
MAPE = 3.7037037037037033

y_true = [10,4,7,4,7,12,4,2,3]
y_pred = [10,4,7,4,7,12,4,2,3]
MAPE = 0.0
```