Métricas de Desempenho de Classificação

Lista de Exercícios

Na resolução dos exercícios a seguir **NÃO** utilize funções de bibliotecas prontas.

In [1]:
```
NEG, POS = 0, 1
```

[01] Implemente a função `binarizar()` de tal forma que o vetor de entrada `v` seja binarizado de acordo com um limiar especificado.

In [2]:
```
def binarizar(v, limiar=0.5):
    """
    Transforma um vetor de entrada v em binário
    Args:
        v: um vetor de entrada
        limiar: um limiar de corte
    Retorna:
        Um vetor binarizado: acima ou igual ao valor do limiar cada valor em 
    Exemplos
        binarizar([0,0,1]) => [0,0,1]
        binarizar([0.6, 0.7, 0.2], limiar=0.5) => [1,1,0]
    """
    return [POS if prob >= limiar else NEG for prob in v]
```

[02] Implemente a função `classificar()`, que deve calcular os verdadeiros positivos (vp), verdadeiros negativos (vn), falsos positivos (fp) e falsos negativos (fn), dados um vetor `y` e `y_pred`.

In [3]:
```
def classificar(y, y_pred, limiar=0.5):
    """
    Contabiliza os VPs, VNs, FPs e FNs a partir dos vetores y e y_pred.
    Args:
        y: vetor com as classificações reais
        y_pred: vetor com as classificações do modelo
    Retorna:
        As quantidades de VPs, VNs, FPs e FNs
    """
    assert len(y) == len(y_pred)
    vp, vn, fp, fn = 0, 0, 0, 0

    binarizado = binarizar(y_pred, limiar)
    for i in range(len(y)):
        if (y[i] == binarizado[i]):
            if (binarizado[i] == NEG):
                vn += 1
            elif (binarizado[i] == POS):
                vp += 1
        elif (y[i] != binarizado[i]):
            # neg => pos
            if (binarizado[i] == POS):
                fp += 1
            # pos => neg
            elif  (binarizado[i] == NEG):
                fn += 1
    return vp, vn, fp, fn
```

[03] Implemente a função `acurácia()`, que deve calcular a acurácia a partir dos vetores `y` e `y_pred`.

In [28]:
```
def acurácia(y, y_pred, limiar=0.5):
    """
    Retorna a acurácia a partir dos vetores y e y_pred
    Args:
        y: vetor com as classificaçõe reais
        y_pred: vetor com as predições
    Retorna
        O valor da acurácia
    """
    vp, vn, fp, fn = classificar(y, y_pred, limiar)
    return round((vp + vn) / (vp + vn + fp + fn), 2)
```

[04] Implemente a função `precisão()`, que deve calcular a precisão a partir dos vetores `y` e `y_pred`.

In [27]:
```
def precisão(y, y_pred, limiar=0.5):
    """
    Retorna a precisão a partir dos vetores y e y_pred
    Args:
        y: vetor com as classificaçõe reais
        y_pred: vetor com as predições
    Retorna
        O valor da precisão
    """
    vp, vn, fp, fn = classificar(y, y_pred, limiar)
    if (vp + fp == 0):
        return .0
    return round(vp / (vp + fp), 2)
```

[05] Implemente a função `revocação()`, que deve calcular a revocação a partir dos vetores `y` e `y_pred`.

In [26]:
```
def revocação(y, y_pred, limiar=0.5):
    """
    Retorna a revocação a partir dos vetores y e y_pred
    Args:
        y: vetor com as classificaçõe reais
        y_pred: vetor com as predições
    Retorna
        O valor da revocação
    """
    vp, vn, fp, fn = classificar(y, y_pred, limiar)
    if (vp + fn == 0):
        return .0
    return round(vp / (vp + fn), 2)
```

[06] Implemente a função `fbeta()`, que deve calcular a métrica `F` a partir dos vetores `y` e `y_pred` e um valor de `beta`.

$$F_\beta = (1 + \beta^2) \cdot \frac{precision \cdot recall}{\beta^2 \cdot precision + recall}$$

In [25]:
```
def fbeta(y, y_pred, beta=1, limiar=0.5):
```

```
    """
    Retorna a revocação a partir dos vetores y e y_pred
    Args:
        y: vetor com as classificaçõe reais
        y_pred: vetor com as predições
        beta: Tipicamnete, 0.5, 1 ou 2
        limiar: Um valor de corte para a binarização
    Retorna
        O valor da revocação
    """
    p = precisão(y, y_pred, limiar)
    r = revocação(y, y_pred, limiar)
    if (p + r == 0):
        return .0
    return round((1 + beta ** 2) * (p * r) / (beta ** 2 * p + r), 2)
```

[07] Implemente a função `taxa_erro()`, que deve calcular a taxa de erro $(1 - acurácia)$ a partir dos vetores `y` e `y_pred`.

In [24]:
```
def taxa_erro(y, y_pred, limiar=0.5):
    """
    Retorna a taxa de erro () a partir dos vetores y e y_pred
    Args:
        y: vetor com as classificaçõe reais
        y_pred: vetor com as predições
        limiar: Um valor de corte para a binarização
    Retorna
        O valor da taxa de erro
    """
    return round(1 - acurácia(y, y_pred, limiar), 2)
```

[08] Implemente a função `m_confusão()`, que deve imprimir os valores de VPs, VNs, FPs e FNs a partir dos vetores `y` e `y_pred`.

In [9]:
```
def m_confusão(y, y_pred, limiar=0.5):
    """
    Imprime os VPs, VNs, FPs e FNs
    Args:
        y: vetor com as classificaçõe reais
        y_pred: vetor com as predições
        limiar: Um valor de corte para a binarização
    Retorna
        None. Não há retorno, uma vez que a função deve imprimir seus cálculo
    """
    vp, vn, fp, fn = classificar(y, y_pred, limiar)
    print("Matriz de Confusão")
    cols = 15
    print("+" + "=" * cols + "+")
    print(f"| VP: {vp} | FP: {fp} |")
    print("+" + "-" * cols + "+")
    print(f"| FN: {fn} | VN: {vn} |")
    print("+" + "=" * cols + "+")
```

In [13]:
```
def linha_sep(cols, sep='-'):
    print(f"+" + sep * cols + "+")
```

In [63]:
```
def relat_clf(y, y_pred, limiar=0.5):
    p = precisão(y, y_pred, limiar)
```

```
    r = revocação(y, y_pred, limiar)
    a = acurácia(y, y_pred, limiar)
    f1 = fbeta(y, y_pred, beta=1, limiar=0.5)
    f2 = fbeta(y, y_pred, beta=2, limiar=0.5)
    f05 = fbeta(y, y_pred, beta=0.5, limiar=0.5)
    te = taxa_erro(y, y_pred, limiar)

    print("Relatório de Desempenho")
    cols = 80
    linha_sep(cols, "=")
    print(f"Acurácia: {a}  | Taxa de Erro: {te}   | Precisão: {p}  | Revocaçã
    linha_sep(cols)
    print(f"f1: {f1}  | f2: {f2}  | f.5: {f05}")
    linha_sep(cols)
    print("")
    m_confusão(y, y_pred, limiar)
```

[09] Estude as métricas `acurácia`, `taxa de erro`, `precisão`, `revocação`, `F1`, `F2`, `F0.5` para os seguintes vetores:

a) `y=[0,0,0,0,0,0,0,0,0,0]` e `y_pred=[1,1,1,1,1,1,1,1,1,1]`:

Se o modelo não acertar nenhuma classificação, então a acurácia, precisão, revocação e as métricas f são 0.

In [64]:
```
y=[0,0,0,0,0,0,0,0,0,0]
y_pred=[1,1,1,1,1,1,1,1,1,1]
relat_clf(y, y_pred, limiar=0.5)
```

```
Relatório de Desempenho
+================================================================================
===+
Acurácia: 0.0  | Taxa de Erro: 1.0   | Precisão: 0.0  | Revocação: 0.0
+--------------------------------------------------------------------------------
---+
f1: 0.0  | f2: 0.0  | f.5: 0.0
+--------------------------------------------------------------------------------
---+

Matriz de Confusão
+===============+
| VP: 0 | FP: 10 |
+---------------+
| FN: 0 | VN: 0 |
+===============+
```

b) `y=[1,1,1,1,1,1,1,1,1,1]` e `y_pred=[0,0,0,0,0,0,0,0,0,0]`:

Se o modelo não acertar nenhuma classificação, então a acurácia, precisão, revocação e as métricas f são 0. Esse segundo caso confirma que as métricas são 0 independentemente de erros aos VPs ou VNs.

In [65]:
```
y=[1,1,1,1,1,1,1,1,1,1]
y_pred=[0,0,0,0,0,0,0,0,0,0]
relat_clf(y, y_pred, limiar=0.5)
```

```
Relatório de Desempenho
+================================================================================
===+
Acurácia: 0.0  | Taxa de Erro: 1.0   | Precisão: 0.0  | Revocação: 0.0
+--------------------------------------------------------------------------------
---+
f1: 0.0  | f2: 0.0  | f.5: 0.0
```

```
+-------------------------------------------------------------------------
---+

Matriz de Confusão
+===============+
| VP: 0 | FP: 0 |
+---------------+
| FN: 10 | VN: 0 |
+===============+
```

c) `y=[0,0,0,0,0,0,0,0,0,1]` e `y_pred=[0,0,0,0,0,0,0,0,0,0]` :

Acertar muitos VNs faz com que haja uma boa acurácia, mas isso não é refletido na precisão, revocação e nas métricas f.

In [66]:
```
y=[0,0,0,0,0,0,0,0,0,1]
y_pred=[0,0,0,0,0,0,0,0,0,0]
relat_clf(y, y_pred, limiar=0.5)
```

```
Relatório de Desempenho
+=========================================================================
===+
Acurácia: 0.9  | Taxa de Erro: 0.1   | Precisão: 0.0  | Revocação: 0.0
+-------------------------------------------------------------------------
---+
f1: 0.0  | f2: 0.0  | f.5: 0.0
+-------------------------------------------------------------------------
---+

Matriz de Confusão
+===============+
| VP: 0 | FP: 0 |
+---------------+
| FN: 1 | VN: 9 |
+===============+
```

d) `y=[1,1,1,1,1,1,1,1,1,0]` e `y_pred=[1,1,1,1,1,1,1,1,1,1]` :

Ao contrário do anterior, acertar muitos VPs faz com que o modelo tenha não somente uma boa acurácia, mas boa precisão e revocação também. Falsos positivos diminuem a precisão e afetam as métricas $f2$ e $f.5$ de modos diferentes. A redução da precisão não afeta $f2$ tanto quanto f.5.

In [67]:
```
y=[1,1,1,1,1,1,1,1,1,0]
y_pred=[1,1,1,1,1,1,1,1,1,1]
relat_clf(y, y_pred, limiar=0.5)
```

```
Relatório de Desempenho
+=========================================================================
===+
Acurácia: 0.9  | Taxa de Erro: 0.1   | Precisão: 0.9  | Revocação: 1.0
+-------------------------------------------------------------------------
---+
f1: 0.95  | f2: 0.98  | f.5: 0.92
+-------------------------------------------------------------------------
---+

Matriz de Confusão
+===============+
| VP: 9 | FP: 1 |
+---------------+
| FN: 0 | VN: 0 |
+===============+
```

e) `y=[1,1,1,0,0,0]` , `y_pred=[.8, .7, .8, .2, .3, .4]` e `limiar=.5` :

Acertar todas as previsões leva todas as métricas aos seus máximos. No entanto, esse cenário não é realista.

In [68]:
```
y=[1,1,1,0,0,0]
y_pred=[.8, .7, .8, .2, .3, .4]
relat_clf(y, y_pred, limiar=0.5)
```

```
Relatório de Desempenho
+=============================================================================
===+
Acurácia: 1.0   | Taxa de Erro: 0.0    | Precisão: 1.0   | Revocação: 1.0
+-----------------------------------------------------------------------------
---+
f1: 1.0   | f2: 1.0   | f.5: 1.0
+-----------------------------------------------------------------------------
---+

Matriz de Confusão
+================+
| VP: 3 | FP: 0 |
+---------------+
| FN: 0 | VN: 3 |
+================+
```

f) `y=[1,1,1,0,0,0]` , `y_pred=[.2, .7, .8, .2, .3, .4]` e `limiar=.5` :

Falsos negativos diminuem a revocação, sem afetar a precisão. $f2$ *A redução da revocação não afeta* $f.5$ *tanto quanto* $f2$.

In [69]:
```
y=[1,1,1,0,0,0]
y_pred=[.2, .7, .8, .2, .3, .4]
relat_clf(y, y_pred, limiar=0.5)
```

```
Relatório de Desempenho
+=============================================================================
===+
Acurácia: 0.83   | Taxa de Erro: 0.17    | Precisão: 1.0   | Revocação: 0.67
+-----------------------------------------------------------------------------
---+
f1: 0.8   | f2: 0.72   | f.5: 0.91
+-----------------------------------------------------------------------------
---+

Matriz de Confusão
+================+
| VP: 2 | FP: 0 |
+---------------+
| FN: 1 | VN: 3 |
+================+
```

g) `y=[1,1,1,0,0,0]` , `y_pred=[.8, .7, .8, .2, .3, .8]` e `limiar=.5` :

Falsos positivos diminuem a precisão, sem afetar a revocação. A redução da precisão não afeta $f2$ *tanto quanto f.5.*

In [70]:
```
y=[1,1,1,0,0,0]
y_pred=[.8, .7, .8, .2, .3, .8]
relat_clf(y, y_pred, limiar=0.5)
```

```
Relatório de Desempenho
+=============================================================================
===+
Acurácia: 0.83   | Taxa de Erro: 0.17    | Precisão: 0.75   | Revocação: 1.0
+-----------------------------------------------------------------------------
```

```
---+
f1: 0.86   | f2: 0.94   | f.5: 0.79
+-------------------------------------------------------------------------------
---+

Matriz de Confusão
+===============+
| VP: 3 | FP: 1 |
+---------------+
| FN: 0 | VN: 2 |
+===============+
```

h) `y=[1,1,1,0,0,0]`, `y_pred=[.2, .7, .8, .2, .3, .8]` e `limiar=.5`:

Em situações reais, não é possível classificar as instâncias perfeitamente. Inevitavelmente, instâncias positivas e negativas serão classificadas com probabilidades similares. Independentemente da direção, movimentar um limiar irá reclassificar corretamente algumas corretamente ao mesmo tempo em que irá reclassificar outras erroneamente. Essa característica é o que faz com que haja na prática uma relação inversa entre a precisão e revocação.

In [71]:

```
y=[1,1,1,0,0,0]
y_pred=[.2, .7, .8, .2, .3, .8]
relat_clf(y, y_pred, limiar=0.5)
```

```
Relatório de Desempenho
+================================================================================
===+
Acurácia: 0.67   | Taxa de Erro: 0.33   | Precisão: 0.67   | Revocação: 0.67
+-------------------------------------------------------------------------------
---+
f1: 0.67   | f2: 0.67   | f.5: 0.67
+-------------------------------------------------------------------------------
---+

Matriz de Confusão
+===============+
| VP: 2 | FP: 1 |
+---------------+
| FN: 1 | VN: 2 |
+===============+
```