Exercíco 4 - Treinamento de Perceptron Simples

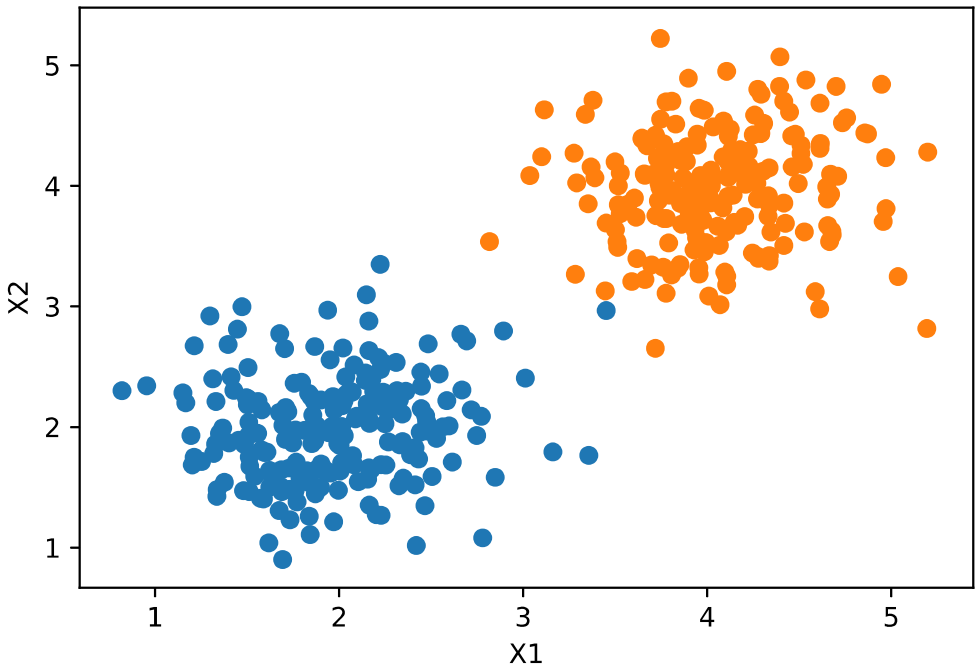
Aluno: Leonam Rezende Soares de Miranda

Exercício 1

Inicialmente, devem-se amostrar duas distribuições Normais no espaço R^2, ou seja, duas distriuições com duas variáveis (Ex: x_1 e x_2). As distriuições são caracterizadas como $N(2, 2, \sigma^2)$ e $(4, 4, \sigma^2)$. O Código a seguir gera essas duas distribuições.

In [47]:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, plot_confusion_matrix, classification_rep
ort, accuracy_score

mu1 = [2, 2]
sigma1 = [[0.2, 0], [0, 0.2]]  # diagonal covariance
# 200 amostras da classe 1
N1 = 200
X_1 = np.random.multivariate_normal(mu1, sigma1, N1)

mu2 = [4, 4]
sigma2 = [[0.2, 0], [0, 0.2]]  # diagonal covariance
# 200 amostras da classe 2
N2 = 200
X_2 = np.random.multivariate_normal(mu2, sigma2, N2)

plt.scatter(X_1[:,0], X_1[:,1])
plt.scatter(X_2[:,0], X_2[:,1])
plt.xlabel('X1')
plt.ylabel('X2')

#Todos os pontos
X = np.vstack((X_1, X_2))
#Classes de todos os pontos
Y = np.squeeze(np.vstack((np.zeros((N1, 1)), np.ones((N2, 1)))))
#Sepera as classes únicas
classes = np.unique(Y)
```



Nesta atividade deve ser feito o treinamento do perceptron com o objetivo de encontrar o vetor de pesos w e encontrar a reta de separação. Os códigos da função de ativação derau e o perceptron simples, se encontram abaixo:

In [39]:

```
def degrau(z):
    """
    Calcula o derau sobre a entrada z

    Argumentos:
    z -- A scalar or numpy array of any size.

    Return:
    s -- derau(z)
    """

    return np.where(z >= 0.0, 1, 0)
```

In [ ]:

```
def predict(X, W):
    """
    Prediz a daída para entrada X a partir do vetor de pesos W

    Argumentos:
    X -- matrix com os dados de entrada
    W -- vetor de pesos

    Return:
    Y -- valores preditos
    """
    return degrau(np.dot(X, W[1:]) + W[0])
```

In [41]:

```
def perceptron(X, Y, eta = 0.01, maxepocas=100):
    """
    Rotina de treinamento do erceptron simples.

    Argumentos:
    X -- matrix com os dados de entrada
    Y: -- rótulos de saída
    eta: passo do treinamento
    tol: tolerância de erro
    maxepocas: número máximo de iterações

    Retornos:
    wt -- derau(z)
    """

    # Vetor pesos inicialmentee nulo
    W = np.zeros(X.shape[1] +1)

    for nepocas in range(maxepocas):
        for xi, yi in zip(X, Y):
            #z = np.dot(xi, w)
            yhati = predict(xi, W)
            W += eta * (yi - yhati) * np.insert(xi,0,1)

    return W

W = perceptron(X,Y)
```

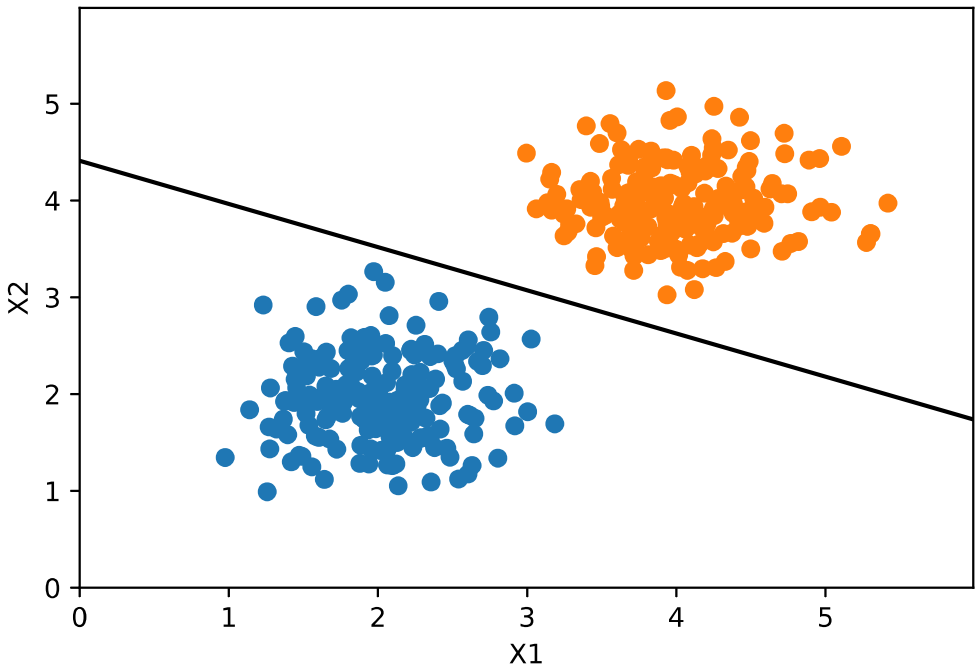
O código a seguir, plota os pontos juntamente com a reta de separação obtida. É possível observa que foi obtido 100% de precisão com os dados de treinamento.

In [8]:

```
x_1 = np.arange(0, 6, 0.01)
x_2 = np.arange(0, 6, 0.01)


slope = -(W[0]/W[2])/(W[0]/W[1])
intercept = -W[0]/W[2]


y = (slope*x_1) + intercept

plt.plot(x_1, y, 'k')
# Z = np.zeros((x_1.shape[0], x_2.shape[0]))

# for i in range(x_1.shape[0]):
#     for j in range(x_2.shape[0]):
#         Z[i,j] = degrau(np.dot(np.insert(np.vstack((x_1[i], x_2[j])).T,0,1, axis=1),
W))

# fig, ax = plt.subplots(figsize=(6,6))

# ax.contour(x_1, x_2, Z, cmap="Greys_r")
plt.scatter(X_1[:,0], X_1[:,1])
plt.scatter(X_2[:,0], X_2[:,1])
plt.xlabel('X1')
plt.ylabel('X2')
plt.xlim(x_1.min(), x_1.max())
plt.ylim(x_2.min(), x_2.max())
```

Out[8]:

```
(0.0, 5.99)
```



Exercício 2

Nesta segunda atividade o aluno deverá criar um conjunto de amostras de cada uma das duas distribuições do Exercício 1, ou seja, 200 amostras da classe 1 e 200 amostras da classe 2 divididas em conjunto de treinamento e teste. O conjunto de treinamento irá conter 70% das amostras e o de teste 30%, amostradas de forma aleatória do conjunto total de 400 amostras. Após a separação dos dois conjuntos o aluno usará o conjunto de treinamento para encontrar os pesos do perceptron e utilizará o conjunto de teste para avaliar o seu desempenho. Apresente a acurácia e a matriz de confusão.

Antes de realizar o experimento, será apresentado so conceitos da matriz de confusão.

2.1 Matriz de Confusão

A matriz de confusão oferece uma visão geral de como um modelo está se saindo. Portanto, é um ótimo ponto de partida para qualquer avaliação de um modelo de classificação. Resumimos a maioria das métricas que podem ser derivadas da matriz de confusão no gráfico a seguir:

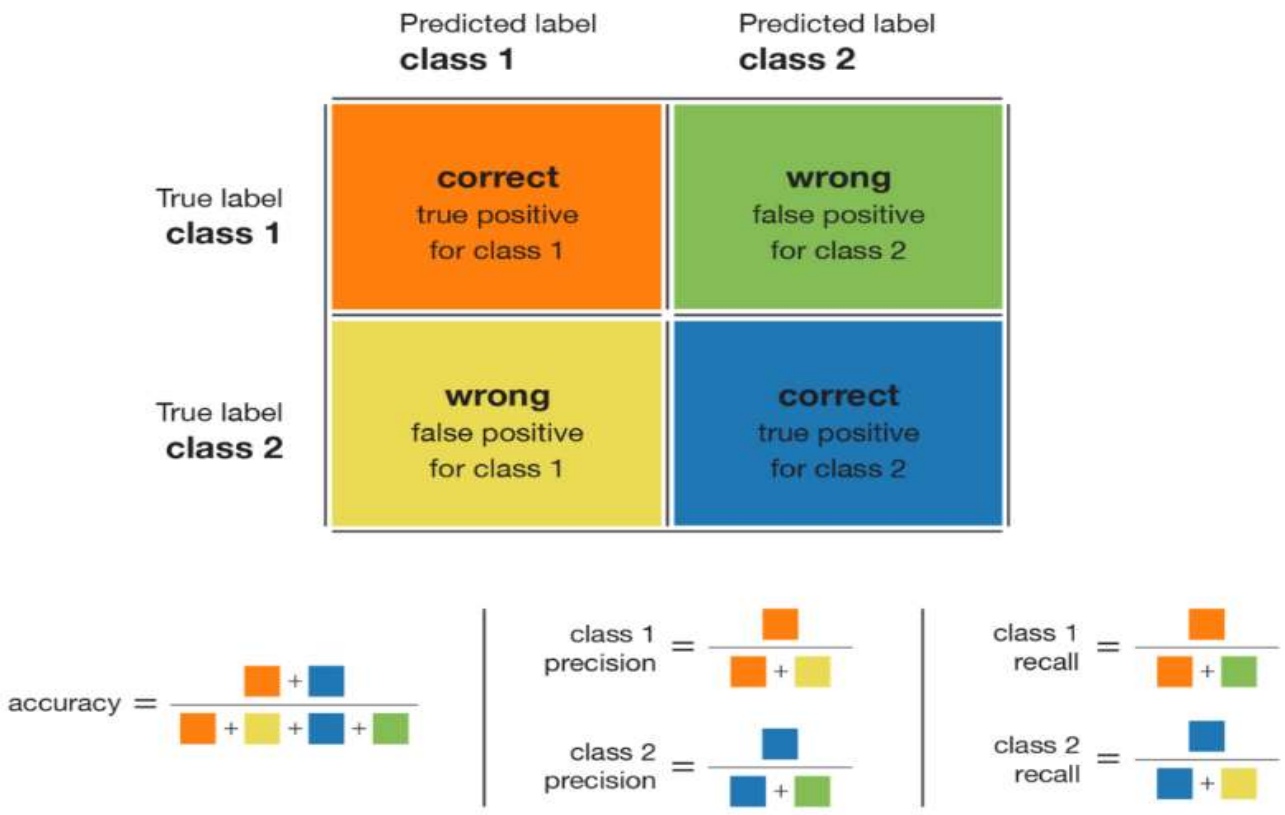


Figura 1: Matrix de Confusão (Fonte: https://medium.com/@itbodhi/handling-imbalanced-data-sets-in-machine-learning-5e5f33c70163).

Sendo:

- TP = Verdadeiro Positivo - O modelo previu a classe positiva corretamente, para ser uma classe positiva.
- FP = Falso Positivo - O modelo previu a classe negativa incorretamente, para ser uma classe positiva.
- FN = Falso Negativo - O modelo previu a classe positiva incorretamente, para ser a classe negativa.
- TN = True Negative - O modelo previu a classe negativa corretamente, para ser a classe negativa.

Sobre as métricas de performance que podem ser extraídas da matriz de confusão temos que:

- **Acurácia**: é basicamente o número total de previsões corretas dividido pelo número total de previsões;

- **Precisão**: de uma classe define quão confiável é o resultado quando o modelo responde que um ponto pertence àquela classe;
- **Recall**: de uma classe expressa o quão bem o modelo é capaz de detectar aquela classe;
- **F1 score**: de uma classe é dada pela média harmônica de precisão e recall $\frac{2 \times precisão \times recall}{precisão + recall}$, combinando precisão e recall de uma classe em uma única métrica.

2.2 Error Tipo 1 e Erro Tipo 2

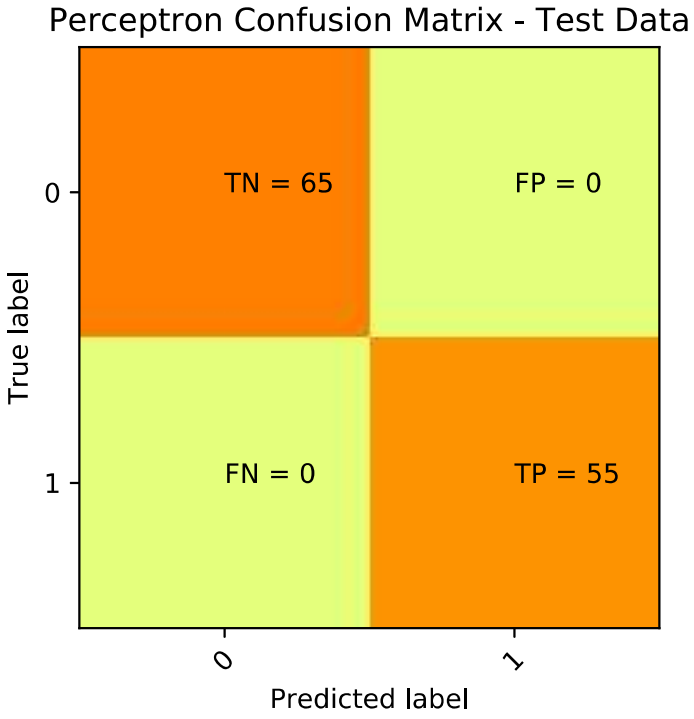
Existem dois tipos de erros que podem ser identificados aqui:

- **Erro Tipo 1**: O modelo previu que a instância seria uma classe Positiva, mas está incorreto. Isso é falso positivo (FP).
- **Erro Tipo 2**: O modelo previu que a instância seria a classe Negativa, mas isso está incorreto. Isso é falso negativo (FN).

In [9]:

```
# Separa os dados - 70% para treinamento e 30% para testes
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3)

# Treina o modelo com os dados de treinamento
W2 = perceptron(X_train,Y_train)

Y_pred = predict(X_test, W2)

print('\nAcurácia do classificador perceptron sobre conjunto de treinamento : {:.2f}'.f
ormat(accuracy_score(Y_train, predict(X_train, W2))))
print('\nAcurácia do classificador perceptron sobre conjunto de teste : {:.2f}'.format(
accuracy_score(Y_test, Y_pred)))

# matrix de confusão
cm = confusion_matrix(Y_test, Y_pred)
print('\nClassification report:\n', classification_report(Y_test, Y_pred))

plt.clf()
plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Wistia)
classNames = ['Negative','Positive']
plt.title('Perceptron Confusion Matrix - Test Data')
plt.ylabel('True label')
plt.xlabel('Predicted label')
tick_marks = np.arange(len(classes))
plt.xticks(tick_marks, rotation=45)
plt.yticks(tick_marks)
s = [['TN','FP'], ['FN', 'TP']]

for i in range(2):
    for j in range(2):
        plt.text(j,i, str(s[i][j])+" = "+str(cm[i][j]))
plt.show()
```

```
Acurácia do classificador perceptron sobre conjunto de treinamento : 1.00

Acurácia do classificador perceptron sobre conjunto de teste : 1.00

Classification report:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00        65
         1.0       1.00      1.00      1.00        55

    accuracy                           1.00       120
   macro avg       1.00      1.00      1.00       120
weighted avg       1.00      1.00      1.00       120
```



Exercíco 3

Neste exercício será trabbalhado com a base de dados conhecida como Iris. Essa base de dados possui 150 amostras e 4 características, sendo 50 para cada uma das três espécies de plantas que constitui a base. Será feito o treinamento do Perceptron para separar a espécie 1 (50 primeiras amostras) das outras duas espécies e avaliar o desempenho do mesmo. Com isso a espécie 1 será a Classe 1 e o conjunto das espécies 2 e 3 será a Classe 2.

O código a seguir faz a importação da ase de dados e otém os rótulos e os atributos.

In [10]:

```
from sklearn import datasets

iris = datasets.load_iris()

# 2. Carregar os dados da Iris e armazená-los
X = iris.data
Y = iris.target

# Obtém os nomes das classes únicas
classes = np.unique(iris.target_names)

# 3. Rotular as amostras da Classe 1 com o valor de 0 e as amostras da Classe 2 com
Y[Y!=0] = 1

# 4. Separa os dados - 70% para treinamento e 30% para testes
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3)

# 5. Utilizar as amostras de treinamento para fazer o treinamento do Perceptron
# 6. Extração do vetor de pesos
W = perceptron(X_train,Y_train)

print(W)
```

```
[-0.01  -0.009 -0.044  0.061  0.026]
```

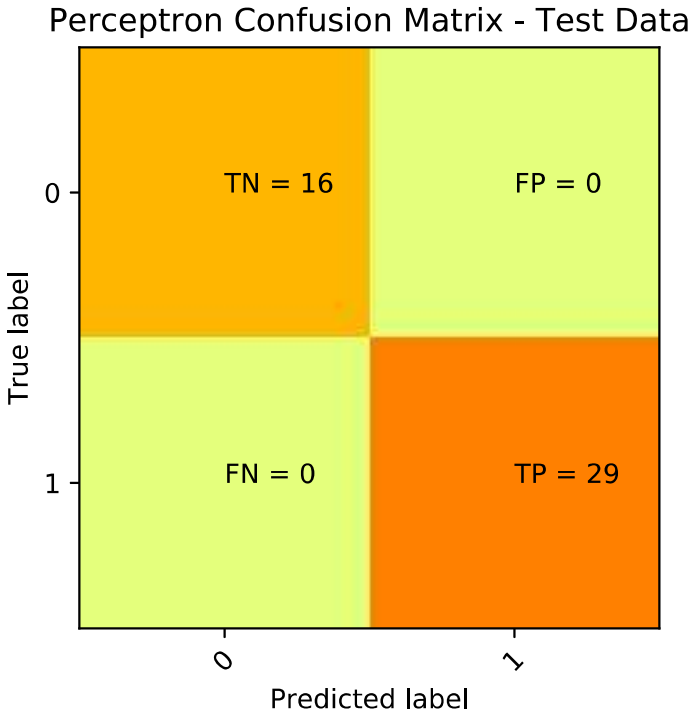
In [11]:

```
# 7. A partir do vetor de pesos, predizer os dados com os dados de treinamento
Y_pred = predict(X_test, W)

# 8. Calcular o erro percentual
print('\nErro percentual do calssificador perceptron sobre conjunto de teste : {:.2f}'.
format(100*np.sum(Y_test!= Y_pred)/Y_test.shape[0])+'%')
```

```
Erro percentual do calssificador perceptron sobre conjunto de teste : 0.0
0%
```

In [12]:

```
# 9. Imprimir a matriz de confusão
cm = confusion_matrix(Y_test, Y_pred)
plt.clf()
plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Wistia)
classNames = ['Negative','Positive']
plt.title('Perceptron Confusion Matrix - Test Data')
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.xticks(tick_marks, rotation=45)
plt.yticks(tick_marks)
s = [['TN','FP'], ['FN', 'TP']]

for i in range(2):
    for j in range(2):
        plt.text(j,i, str(s[i][j])+" = "+str(cm[i][j]))
plt.show()
```

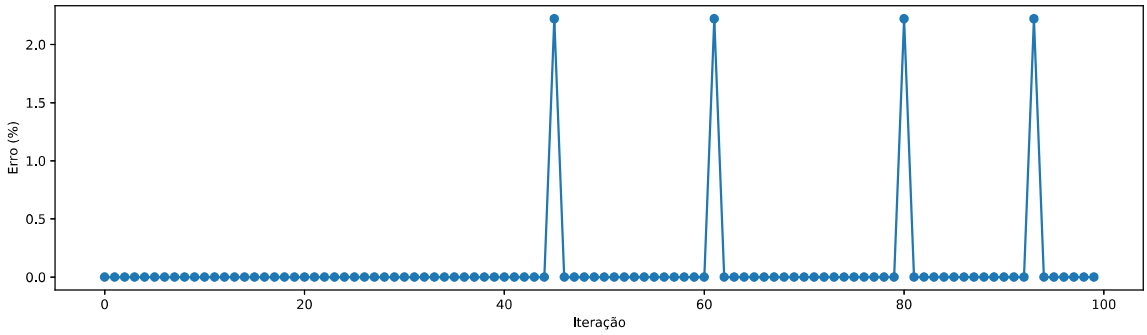


In [19]:

```
# 10. Criar um loop para repetir 100 vezes os itens 4-8, armazenando o valor do erro pe
rcentual do item 8. Plotar o erro percentual em função do número de iteração e imprima
 o valor da variância do erro.

erros = []

for i in range(100):
    # 4. Separa os dados - 70% para treinamento e 30% para testes de forma aleatória
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3)

    # 5. Utilizar as amostras de treinamento para fazer o treinamento do Perceptron
    # 6. Extração do vetor de pesos
    W = perceptron(X_train,Y_train)

    # 7. A partir do vetor de pesos, predizer os dados com os dados de treinamento
    Y_pred = predict(X_test, W)

    # 8. Calcular o erro percentual
    erros.append(100*np.sum(Y_test!= Y_pred)/Y_test.shape[0])

plt.figure(figsize=(15,4))
plt.plot(erros, '-o')
plt.ylabel('Erro (%)')
plt.xlabel('Iteração')
```

Out[19]:

```
Text(0.5, 0, 'Iteração')
```



Exercício 4

Neste exercício deve ser feito o treinamento do Perceptron utilizando a base Breast Cancer.

In [52]:

```
from sklearn.datasets import load_breast_cancer
from sklearn.preprocessing import StandardScaler

# 1. Carregar a base de dados, remover dados faltantes e normalizar os dados
data = load_breast_cancer()

X = data.data
Y = data.target


count = 0
# Loop para verificar se há dados nulos/fatantes no vetor de rótulos Y
for i, j in zip(pd.DataFrame(Y).isnull().sum(), pd.DataFrame(X).isnull().sum()):
    if i!=0 or j!=0:
        count+=1

if count == 0:
    print('Não há dados faltantes!')


# Standardization - Todos os atriutos agora terão média zero e variação unitária
scaler = StandardScaler()
X = scaler.fit_transform(X)

# 2. Separa os dados - 70% para treinamento e 30% para testes
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3)

# 3. Realizar o treinamento do Perceptron
W = perceptron(X_train,Y_train)

# 4. Realizar a previsão para os dados de teste
Y_pred = predict(X_test, W)

# 5. Calcular a acurácia
print('Foi obtida uma acurácia de {:.2f}'.format(100*accuracy_score(Y_test, Y_pred))+'%
nos dados de teste')
```

```
Não há dados faltantes!
Foi obtida uma acurácia de 96.49% nos dados de teste
```

In [54]:

```
#5. Repetir 30 vezes as etapas a partir do item 2 e calcular a média e o desvio padrão
 das acurácias

accuracy_scores = []
for i in range(30):
    # 2. Separa os dados - 70% para treinamento e 30% para testes
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3)

    # 3. Realizar o treinamento do Perceptron
    W = perceptron(X_train,Y_train)

    # 4. Realizar a previsão para os dados de teste
    Y_pred = predict(X_test, W)

    # Armazena a acurácia % obtida
    accuracy_scores.append(100*accuracy_score(Y_test, Y_pred))

print('A acurácia média obtida foi de {:.2f}'.format(np.mean(accuracy_scores)) + '%')
print('O desvio padrão dos valores de acurácia foi de {:.2f}'.format(np.std(accuracy_sc
ores)))
```

```
A acurácia média obtida foi de 96.22%
O desvio padrão dos valores de acurácia foi de 1.37
```