KNN usando SKLearn e o dataset Iris

In [2]:

```
#Import data and modules
import numpy as np
from sklearn import datasets
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from collections import Counter
```

In [4]:

```
iris = datasets.load_iris()
```

In [5]:

```
X = iris.data[:, [0, 1, 2, 3]]
y = iris.target
```

In [9]:

```
#split the data into training and test datasets.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, rand

print('There are {} samples in the training set and {} samples in the test se
```

```
There are 105 samples in the training set and 45 samples in the test set
```

In [10]:

```
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
```

Out[10]:

```
KNeighborsClassifier()
```

In [13]:

```
y_pred = knn.predict(X_test)
```

In [14]:

```
y_pred
```

Out[14]:

```
array([2, 1, 0, 2, 0, 2, 0, 1, 1, 1, 2, 1, 1, 1, 1, 0, 1, 1, 0, 0, 2, 1,
       0, 0, 2, 0, 0, 1, 1, 0, 2, 1, 0, 2, 2, 1, 0, 2, 1, 1, 2, 0, 2, 0,
       0])
```

In [15]:

```
y_test
```

Out[15]:

```
array([2, 1, 0, 2, 0, 2, 0, 1, 1, 1, 2, 1, 1, 1, 1, 0, 1, 1, 0, 0, 2, 1,
       0, 0, 2, 0, 0, 1, 1, 0, 2, 1, 0, 2, 2, 1, 0, 1, 1, 1, 2, 0, 2, 0,
       0])
```

In [16]:

```
Counter(y_test)
```

Out[16]:

```
Counter({2: 11, 1: 18, 0: 16})
```

In [17]:

```
Counter(y_pred)
```

Out[17]:

```
Counter({2: 12, 1: 17, 0: 16})
```

Acurácia

In [10]:

```
hit_total = 0
for i in range(len(y_pred)):
    if y_pred[i] == y_test[i]:
        hit_total += 1

print(hit_total, "of", len(y_test), '=', round((hit_total/len(y_test)) * 100,
```

```
44 of 45 = 97.78 %
```

Questões

1. Calcular as métricas Falso Positivo, Falso Negativo, Verdadeiro Positivo e Verdadeiro Negativo

Solução:

In [19]:

```
fp = [0, 0, 0]
fn = [0, 0, 0]
tp = [0, 0, 0]
tn = [0, 0, 0]

for i in range(len(y_pred)):
    if y_pred[i] == y_test[i]:
        tp[y_pred[i]] += 1

        for n in [0, 1, 2]:
            if n != y_pred[i]:
                tn[n] += 1
    else:
        fp[y_pred[i]] += 1
        fn[y_test[i]] += 1

        for n in [0, 1, 2]:
            if n != y_pred[i] and n != y_test[i]:
                tn[n] += 1

print(fp, fn, tp, tn)
```

```
[0, 0, 1] [0, 1, 0] [16, 17, 11] [29, 27, 33]
```

2. Calcular as métricas Precision, Recall e F1-Score

Solução:

In [20]:

```
def precision(tp, fp, fn, tn):
    return tp / (tp + fp)

def recall(tp, fp, fn, tn):
    return tp / (tp + fn)

def f1_score(tp, fp, fn, tn):
    p = precision(tp, fp, fn, tn)
    r = recall(tp, fp, fn, tn)
    return 2 * p * r / (p + r)
```

In [21]:

```
print('Precision Type 0 =', precision(tp[0], fp[0], fn[0], tn[0]))
print('Precision Type 1 =', precision(tp[1], fp[1], fn[1], tn[1]))
print('Precision Type 2 =', precision(tp[2], fp[2], fn[2], tn[2]))
```

```
Precision Type 0 = 1.0
Precision Type 1 = 1.0
Precision Type 2 = 0.9166666666666666
```

In [22]:

```
print('Recall Type 0 =', recall(tp[0], fp[0], fn[0], tn[0]))
print('Recall Type 1 =', recall(tp[1], fp[1], fn[1], tn[1]))
print('Recall Type 2 =', recall(tp[2], fp[2], fn[2], tn[2]))
```

```
Recall Type 0 = 1.0
Recall Type 1 = 0.9444444444444444
Recall Type 2 = 1.0
```

In [23]:

```
print('F1-Score Type 0 =', f1_score(tp[0], fp[0], fn[0], tn[0]))
print('F1-Score Type 1 =', f1_score(tp[1], fp[1], fn[1], tn[1]))
print('F1-Score Type 2 =', f1_score(tp[2], fp[2], fn[2], tn[2]))
```

```
F1-Score Type 0 = 1.0
F1-Score Type 1 = 0.9714285714285714
F1-Score Type 2 = 0.9565217391304348
```

In [ ]: