k – Plus proches voisins ¶

Chargement des librairies

Entrée [1]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import sklearn
```

Chargement des données:

Dans cette partie pratique nous allons utiliser un jeu de données décrivant des manchots, le jeu Palmer Penguins (https://allisonhorst.github.io/palmerpenguins/).

Le jeu de données contient des informations décrivant un certain nombre de manchots appartenant à trois espèces :

- Manchot d'Adélie (*Pygoscelis adeliae*), `Adelie` dans les données
- Manchot papou (*Pygoscelis papua*), `Gentoo` dans les données
- Manchot à jugulaire (*Pygoscelis antarcticus*), `Chinstrap` dans les données.

Entrée [2]:

```
penguins = pd.read_csv("data/penguins.csv")
penguins.head()
```

Out[2]:

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	se
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	ma
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	fema
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	fema
3	Adelie	Torgersen	36.7	19.3	193.0	3450.0	fema
4	Adelie	Torgersen	39.3	20.6	190.0	3650.0	ma

Plus proches voisins pour la classification

Chargement des données

Entrée [3]:

```
X = penguins[["bill_length_mm", "bill_depth_mm", "flipper_length_mm"]].to_numpy()
y_classif = pd.Categorical(penguins["sex"]).astype('category').codes

y_classif
```

Out[3]:

```
array([1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1,
       0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1,
       0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1,
       0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1,
       0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1,
       0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1,
       0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1,
       0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0,
       1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0,
       0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0,
       1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0,
       1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1,
       0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0,
       1, 1, 0], dtype=int8)
```

Séparation des données en jeu d'entraînement et jeu de test

Entrée [4]:

```
from sklearn import model_selection
(X_train, X_test, y_train, y_test) = model_selection.train_test_split(X, y_classif, test_
                                                                      random_state=25, stratify=y_c
```

Pour la classification, nous utilisons [la classe `KNeighborsClassifier` (https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html)](https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html).

1. Instancions un objet de la classe `KNeighborsClassifier` .

Entrée [5]:

```
from sklearn import neighbors
knnclass = neighbors.KNeighborsClassifier(n_neighbors=7)
```

2. Entraînons cet objet sur les données d'entraînement avec la méthode `fit` :

Entrée [6]:

```
knnclass.fit(X_train, y_train)
```

Out[6]:

```
KNeighborsClassifier(n_neighbors=7)
```

3. Enfin, prédisons les étiquettes des données du jeu de test en utilisant la méthode `predict` :

Entrée [7]:

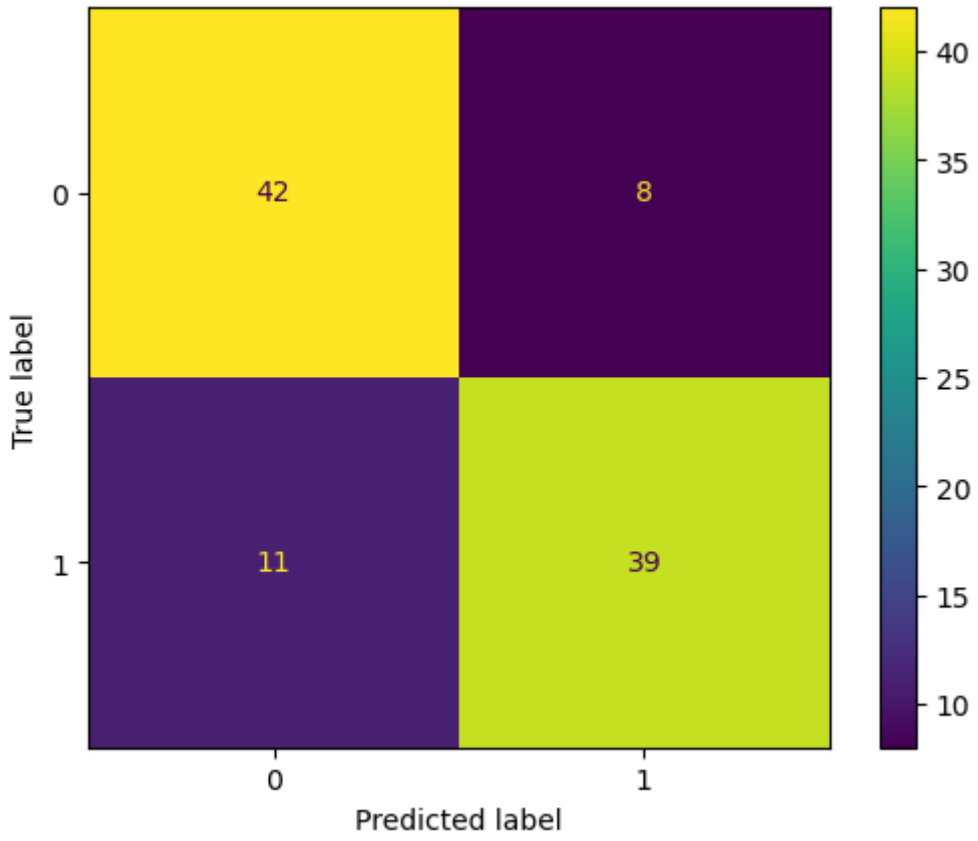
```
y_test_pred = knnclass.predict(X_test)
```

Performance

Regardons la **matrice de confusion** des prédictions :

Entrée [8]:

```
from sklearn import metrics
#metrics.plot_confusion_matrix(knnclass, X_test, y_test)
metrics.ConfusionMatrixDisplay.from_estimator(knnclass, X_test, y_test)
plt.show()
```



Entrée [9]:

```
print("%d manchots mâles ont été incorrectement prédits femelle." % metrics.confusion_mat
print("%d manchots femelles ont été incorrectement prédits mâles." % metrics.confusion_ma
print("%.f%% des prédictions du modèle sur le jeu de test sont correctes." % (100*metrics
```

```
11 manchots mâles ont été incorrectement prédits femelle.
8 manchots femelles ont été incorrectement prédits mâles.
81% des prédictions du modèle sur le jeu de test sont correctes.
```

Performance du modèle par GridSearchCV

Entrée [10]:

```
k_values = [5, 7, 9, 13, 15, 17, 19, 21]
params={'n_neighbors': k_values}

model = neighbors.KNeighborsClassifier()

knnclass_cv = model_selection.GridSearchCV( model, params, cv=5, scoring='accuracy')

knnclass_cv.fit(X_train, y_train)

print("La valeur optimale de k est %d, pour une accuracy cross-validée de %.f%%."
      % (knnclass_cv.best_params_['n_neighbors'], (100*knnclass_cv.best_score_)))
```

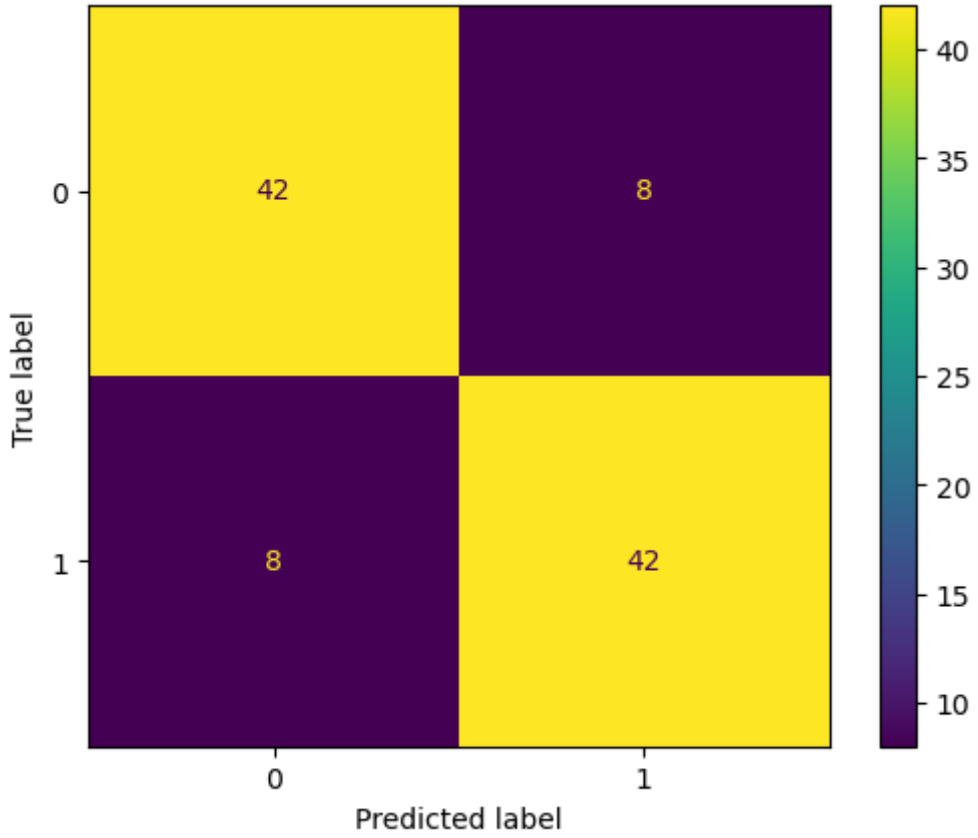
```
La valeur optimale de k est 5, pour une accuracy cross-validée de 83%.
```

Entrée [11]:

```
y_test_pred = knnclass_cv.best_estimator_.predict(X_test)
metrics.ConfusionMatrixDisplay.from_estimator(knnclass_cv.best_estimator_, X_test, y_test
```

Out[11]:

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1b9b42
af730>
```



Entrée [12]:

```
print("%d manchots mâles ont été incorrectement prédits femelle." % metrics.confusion_mat
print("%d manchots femelles ont été incorrectement prédits mâles." % metrics.confusion_ma
print("%.f%% des prédictions du modèle sur le jeu de test sont correctes." % (100*metrics
```

```
8 manchots mâles ont été incorrectement prédits femelle.
8 manchots femelles ont été incorrectement prédits mâles.
84% des prédictions du modèle sur le jeu de test sont correctes.
```