– Régressions linéaire et logistique

Le but de ce notebook est de présenter la régression linéaire et la régression logistique.

Chargement des librairies

Entrée [57]:

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

1. Régression linéaire

Chargement des données

Entrée [58]:

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

Out[58]:

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

Entrée [59]:

```
X = penguins[["bill_length_mm", "bill_depth_mm", "flipper_length_mm"]].to_numpy()
y_reg = penguins["body_mass_g"]
```

Entrée [62]:

```
from sklearn import linear_model

linreg = linear_model.LinearRegression()

linreg.fit(X, y_reg)

print("Le poids (g) d'un manchot est prédit par  %.2f " % linreg.intercept_, end='')
print("+ %.2f x bill_length_mm + %.2f x bill_depth_mm + %.2f x flipper_length_mm"
      % (tuple(linreg.coef_)))

y_pred = linreg.predict(X)
```

```
Le poids (g) d'un manchot est prédit par  -6445.48 + 3.29 x bill_length_
mm + 17.84 x bill_depth_mm + 50.76 x flipper_length_mm
```

Performance du modèle

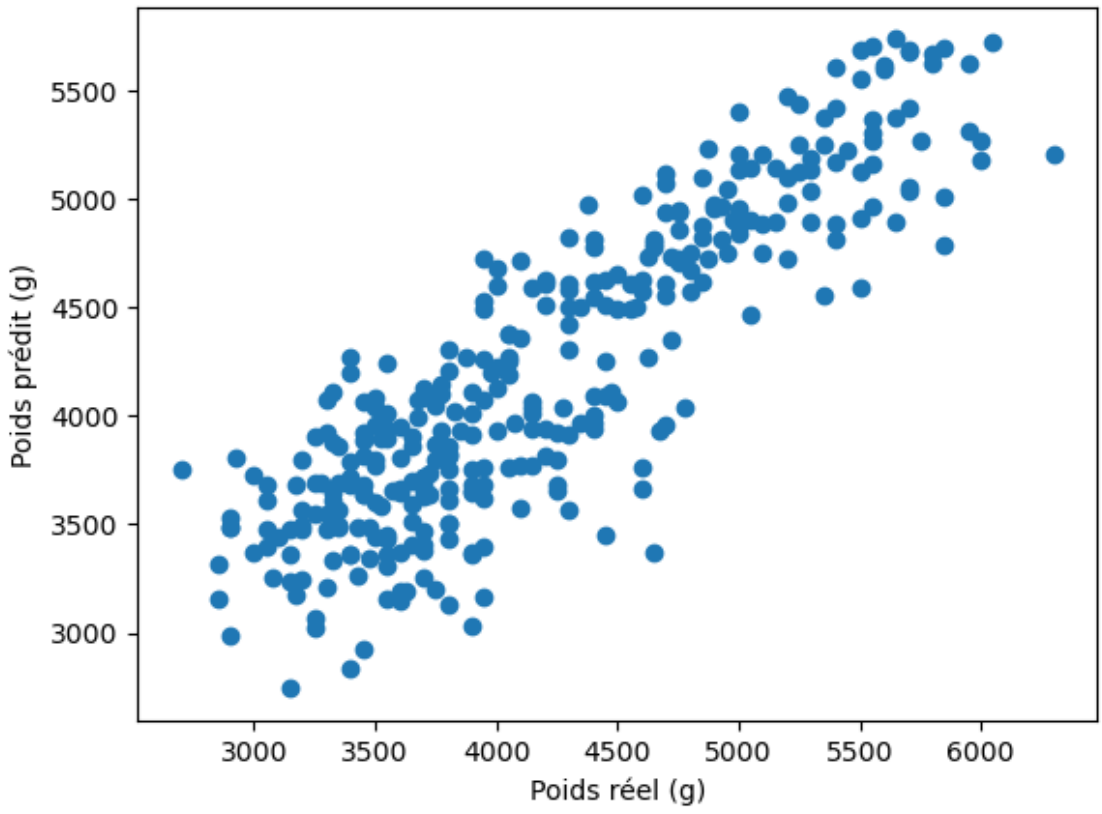
Entrée [63]:

```
plt.scatter(y_regress, y_pred)

plt.xlabel("Poids réel (g)")
plt.ylabel("Poids prédit (g)")
```

Out[63]:

```
Text(0, 0.5, 'Poids prédit (g)')
```



Entrée [64]:

```
from sklearn import metrics

print("La RMSE de notre modèle est %.2f g"
      % (metrics.mean_squared_error(y_reg, y_pred, squared=False)))

print("Le coefficient de détermination de notre modèle est R2 = %.2f"
    % (metrics.r2_score(y_regress, y_pred)))
```

```
La RMSE de notre modèle est 390.64 g
Le coefficient de détermination de notre modèle est R2 = 0.76
```

Utilisation d'un jeu de test

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

Entrée [65]:

```
from sklearn import model_selection
(X_train, X_test, y_train, y_test) = model_selection.train_test_split(X, y_reg,
                                                                      test_size=0.3, ra
X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

Out[65]:

```
((233, 3), (233,), (100, 3), (100,))
```

Generalization du performance

Entrée [66]:

```
linreg.fit(X_train, y_train)
y_test_pred = linreg.predict(X_test)

print("La RMSE de notre modèle est %.2f g"
      % (metrics.mean_squared_error(y_test, y_test_pred, squared=False)))
print("Le coefficient de détermination de notre modèle est R2 = %.2f"
      % (metrics.r2_score(y_test, y_test_pred)))
```

```
La RMSE de notre modèle est 411.80 g
Le coefficient de détermination de notre modèle est R2 = 0.74
```

3. Régression logistique

Chargement des données

Entrée [67]:

```
y_classif = pd.Categorical(penguins["sex"]).astype('category').codes
```

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

Entrée [68]:

```
(X_train, X_test, y_train, y_test) = model_selection.train_test_split(X, y_classif,
                                                                      test_size=0.3,
                                                                      random_state=25,
                                                                      stratify=y_classi

logreg = linear_model.LogisticRegression(penalty='none')
logreg.fit(X_train, y_train)
```

Out[68]:

```
LogisticRegression(penalty='none')
```

Entrée [69]:

```
print("La probabilité qu'un manchot soit mâle est prédite par sigma (%.2f "
      % logreg.intercept_[0], end='')
print("+ %.2f x bill_length_mm + %.2f x bill_depth_mm + %.2f x flipper_length_mm)"
      % (tuple(logreg.coef_[0])))
```

```
La probabilité qu'un manchot soit mâle est prédite par sigma (-62.71 +
0.15 x bill_length_mm + 1.50 x bill_depth_mm + 0.15 x flipper_length_mm)
```