ML : Régression Linéaire

Définition et Principe

- **Définition**
  - ✓ Les algorithmes de régression linéaire modélisent **la relation entre des variables prédictives et une variable cible**. La relation est modélisée par fonction mathématique de prédiction
- **Principe**
  - Disposer d'un jeu de données **Dataset (x, y)** où x (variables prédictives) et y l'objectif (variable cible)
  - Développer un **Modèle** dont les **paramètres** sont à trouver : $y=f(x)$ $f(x)$ une fonction linéaire
  - Définir une **Fonction Coût** qui mesure l'erreur entre le modèle et les valeurs **y** du Dataset
  - Utiliser un **Algorithme d'Apprentissage** qui cherche le modèle qui minimise la Fonction Coût (modèle aux plus petites erreurs)

ML : Régression Linéaire

Dataset d'une régression Linéaire

- **Notation**

✓ Régression Linéaire

- En général, on a n caractéristiques notées x_1, x_2, ..., x_n
- Pour un Dataset de m exemples on aura $y^{(1)}$ $x_1^{(1)}$ $x_2^{(1)}$ ... $x_n^{(1)}$, $y^{(2)}$ $x_1^{(2)}$ $x_2^{(2)}$… $x_n^{(2)}$, ..., $y^{(m)}$ $x_1^{(m)}$ $x_2^{(m)}$... $x_n^{(m)}$ les m valeurs de y pour les m valeurs des n caractéristiques x_i

✓ Régression Linéaire simple (univariée)

- Dataset avec m exemples et n = 1 caractéristique (variable $x = x_1$)
  - On note : $y^{(1)}$ $x^{(1)}$ $y^{(2)}$ $x^{(2)}$ ... $y^{(m)}$ $x^{(m)}$ les m valeurs de y pour les m valeurs de x

ML : Régression Linéaire

Dataset d'une régression Linéaire

- **Exemples**
  - ✓ Régression Linéaire
    - On désire prédire le prix d'une maison en fonction de sa superficie et de sa localisation, on a ici :
      - n = 2, x_1 la superficie et x_2 la localisation
      - y la valeur cible correspond au prix
  - ✓ Régression Linéaire simple
    - Dans une forêt, l'on a identifié un ensemble de m=14 arbres de la même espèce l'on a relevé pour chaque arbre la taille (caractéristique x) et le rayon (l'objectif y). On désire prédire :
      - Le rayon des arbres de taille 135 cm et 232 cm
      - La taille d'un arbre de rayon 25

ML : Régression Linéaire

Modèle d'une régression Linéaire simple

- **Représentation graphique des données**
  - ✓ Donne un nuage de points qui suit :
    - Une tendance linéaire (modèle linéaire)
    - Le modèle est de la forme : $Y = aX + b + \varepsilon$ où $f(x) = ax + b$ avec :
      - Y, variable cible, aléatoire dépendante
      - a et b, coefficients (pente et ordonnée à l'origine) à estimer par la machine
      - X, variable explicative, indépendante
      - ε, variable aléatoire représentant l'erreur
      - Pour la régression multiple on a : $Y = aX1 + bX2 + cX3 + \ldots + K + \varepsilon$
    - Sous forme matricielle, on a : $Y = AX + \varepsilon$

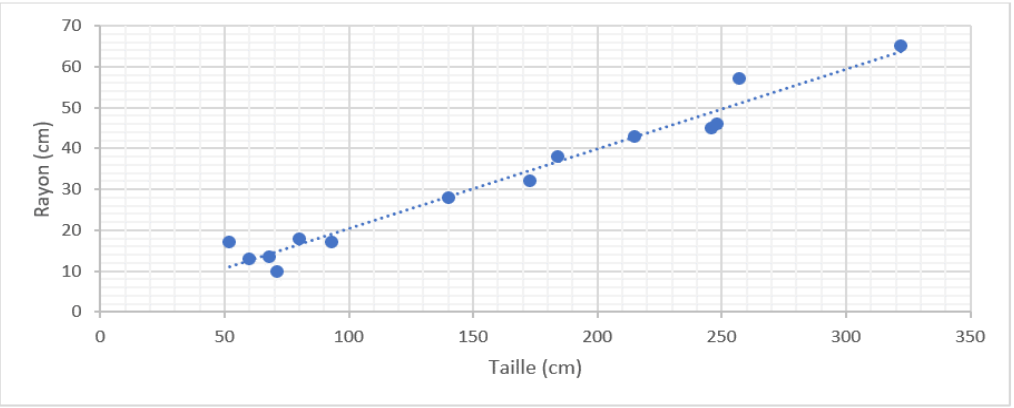
	Taille (cm)	Rayon (cm)
Arbre n°1	184	38
Arbre n°2	246	45
A3	322	65
A4	257	57
A5	248	46
A6	215	43
A7	173	32
A8	93	17
A9	71	10
A10	52	17
A11	68	13,6
A12	60	13
A13	80	18
A14	140	28
A15	135	??
A16	??	25
A17	232	??

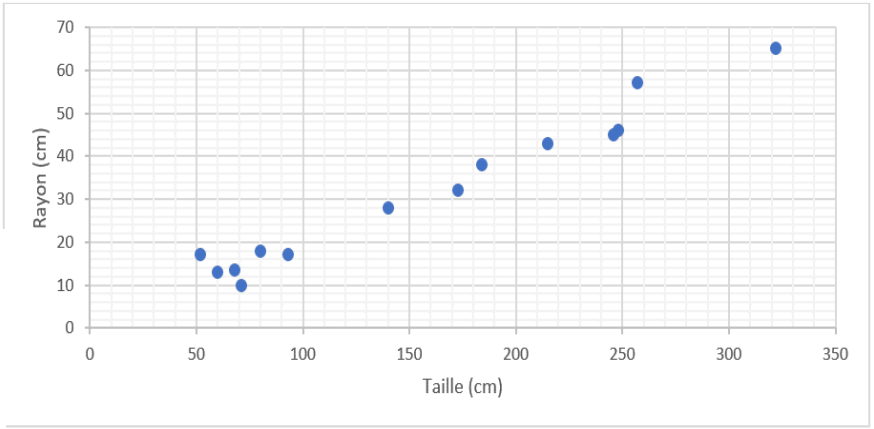
ML : Régression Linéaire

Modèle d'une régression Linéaire simple

- **Représentation graphique des données**
  - ✓ Exemple : tailles et rayons des arbres

À vue d'œil, on a une tendance linéaire.
D'où la droite $f(x) = ax + b$



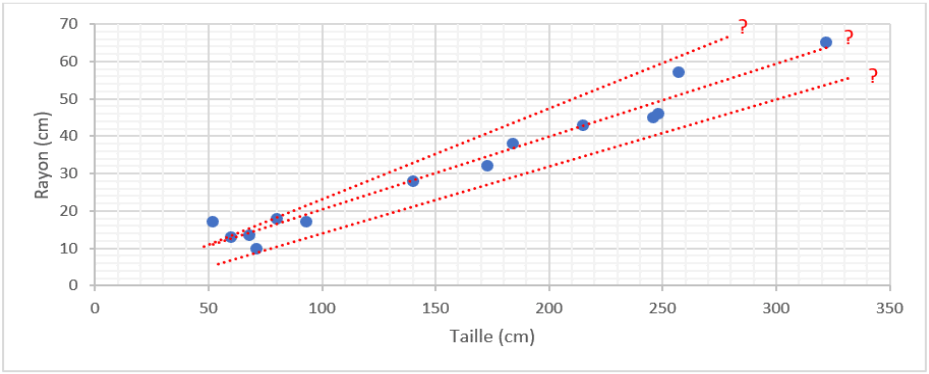


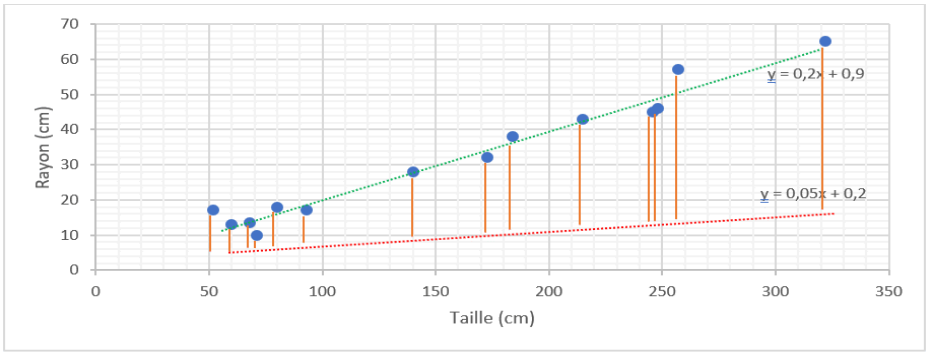
Problème : comment la machine saura que c'est une droite?

ML : Régression Linéaire

Fonction Coût

- **Plusieurs droites pour le nuage de points**
  - ✓ $\exists^{ce}$ d'une infinité de droites pour un nuage de point
  - ✓ Quelle est la meilleure droite?
    - Celle qui minimise les écarts entre la réalité (tailles et rayons réellement observés) et les prédictions (tailles et rayons calculés)





La droite rouge est moins bonne que la verte

ML : Régression Linéaire

Fonction Coût

- **Norme Euclidienne**
  - ✓ Mesurer les erreurs entre les prédictions $f(x)$ et les valeurs y du Dataset
    - Calcul du carré de la différence : $(f(x) - y)^2$
    - C'est la **norme euclidienne** : distance entre $f(x)$ et y
- **Erreur Quadratique Moyenne**
  - ✓ La fonction Coût se définit par la moyenne de toutes les erreurs
    - $J = [(f(x^{(1)})-y^{(1)})^2 + \ldots + (f(x^{(m)})-y^{(m)})^2]/m$
  - ✓ Par convention, on rajoute un coefficient ½ pour la dérivée
    - $J(a,b) = \frac{1}{2m}\sum_{i=1}^{m}(ax^{(i)} + b - y^{(i)})^2$ c'est l'Erreur Quadratique Moyenne (Mean Squared Error)

ML : Régression Linéaire

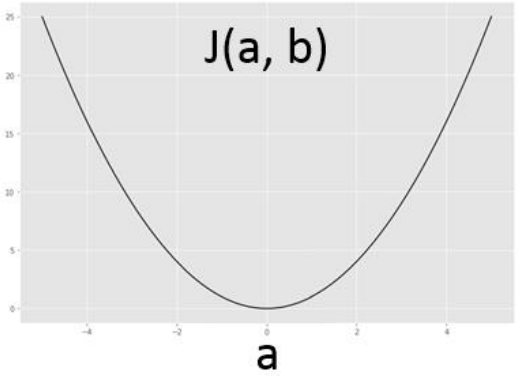
Algorithme d'apprentissage

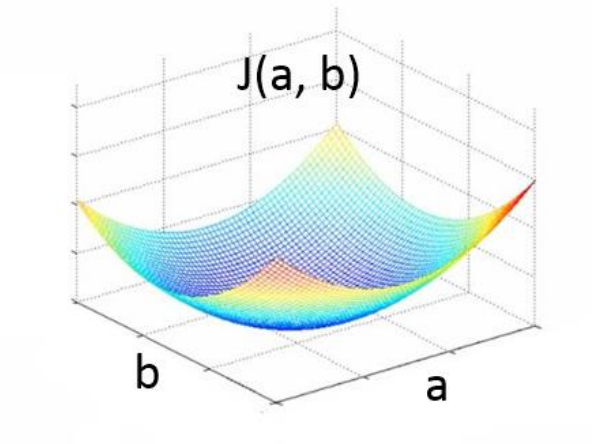
- **Rappel**
  - ✓ Machine apprend qd elle trouve les paramètres du modèle qui **minimisent** la fonction Coût
    - On développe donc un algorithme de **minimisation**
    - ∃ce de plusieurs méthodes (moindres carrés, Newton, Gradient Descent, simplex, etc).
- **Convexité de la fonction**
  - ✓ La fonction Coût ne dépend que de 2 paramètres : a et b
  - ✓ Elle se présente sous l'apparence d'une vallée
    - On dit qu'elle est convexe
    - Propriété très importante pour converger vers le minimum avec l'algorithme de **Gradient Descent**

ML : Régression Linéaire

Algorithme d'apprentissage

- **Convexité de la fonction**





- ✓ Méthode des moindres carrés utilisé pour problèmes simples
- ✓ Algorithme de Gradient Descent pour les régression plus compliquées

ML : Régression Linéaire

Algorithme d'apprentissage

- **Algorithme Gradient Descent**
  - ✓ Processus itératif
    1. définir valeur initiale pour a et b
    2. Calculer dérivée de J(a,b) par rapport à a $(\frac{\partial j(a,b)}{\partial a})$
    3. Calculer dérivée de J(a,b) par rapport à b $(\frac{\partial j(a,b)}{\partial b})$
    4. Si les dérivées sont proches de 0, arrêt les valeurs de a et b sont trouvées
    5. Sinon calculer nouveau a et b de la manière suivante :
       - **« nouveau » = « ancien » - « learning_rate » *« Dérivée »**
       - **« learning_rate »** est le taux d'apprentissage (en général = à 0,01) son identification fait l'objet de méthodes spécifiques
       - Puis reprendre au point 1

ML : Régression Linéaire

Travaux pratiques N°1

- **Reprendre l'exemple des arbres de la forêt**
  - ✓ Définir une Machine Learning en Python
    1. Saisir le Dataset dans un fichier CSV
    2. Représenter graphiquement le nuage de points
    3. Définir la fonction de coût
    4. Implémenter l'algorithme du Gradient Descent pour déterminer a et b
    5. Représenter graphiquement la fonction $f(x) = ax + b$ dans le nuage de points
  - ✓ Définir une Machine Learning en R
    - Suivre les mêmes étapes que précédemment
  - ✓ Définir une Machine Learning dans un autre langage

ML : Régression Linéaire

Formulation matricielle

- **Le Dataset dans matrices X, Y**
  - ✓ Dataset (x, y) contenant m exemples et n caractéristiques
    - Données $y^{(i)}$ dans un vecteur Y de dimension (m x 1)
    - Données $x_j^{(i)}$ dans une matrice X à laquelle on ajoute une colonne biais (colonne remplie de « 1 »), sa dimension est donc (m x n+1)
    - Dans la régression linéaire univariée $n = 1$, X de dimension (m x 2)
      - X et Y sont :

$$X = \begin{bmatrix} x^{(1)} & 1 \\ x^{(2)} & 1 \\ x^{(3)} & 1 \\ \dots & \dots \\ x^{(m)} & 1 \end{bmatrix} \qquad Y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ \dots \\ y^{(m)} \end{bmatrix}$$

ML : Régression Linéaire

Formulation matricielle

- **Le modèle F = X.θ**
  - ✓ Le modèle linéaire $f(x^{(i)}) = ax^{(i)} + b$
    - Ne permet de faire calcul d'une prédiction à la fois, pour $x^{(i)}$ unique
    - Pour remédier, on crée vecteur F dimension (m x 1) contenant toutes les prédictions
  - ✓ Pour calculer ce vecteur
    - On utilise formule : F = X.θ où $\theta = \begin{bmatrix} a \\ b \end{bmatrix}$ est le vecteur paramètre contenant tous les paramètres du modèle. On a donc produit matriciel :

$$F = X.\theta = \begin{bmatrix} x^{(1)} & 1 \\ \dots & \dots \\ x^{(m)} & 1 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} ax^{(1)} + 1 \times b \\ \dots \\ ax^{(m)} + 1 \times b \end{bmatrix} = \begin{bmatrix} f(x^{(1)}) \\ \dots \\ f(x^{(m)}) \end{bmatrix}$$

    - N.B. : la colonne **biais** de la matrice **X** permet d'effectuer le produit matriciel **X.θ**. On dit également que le paramètre b est un paramètre **biais**.

ML : Régression Linéaire

Formulation matricielle

- **La Fonction Coût J(θ)**
  - ✓ La fonction coût, l'erreur quadratique moyenne s'exprime par :
    - $J(a,b) = \frac{1}{2m}\sum_{i=1}^{m}(ax^{(i)} + b - y^{(i)})^2$
    - On sait que $ax^{(i)}+b$ peut s'écrire sous la forme X.θ
    - Les $y^{(i)}$ sont représentés par le vecteur Y
    - Y et X.θ sont de dimensions (m x 1) on peut écrire la différence X.θ - Y
    - On prend le carré de chaque élément du vecteur X.θ-Y, puis on fait la somme de chacun de ces éléments, avant de diviser le tout par $\frac{1}{2m}$
      - La forme matricielle de la fonction coût est alors :

$$J(\theta) = \frac{1}{2m}\sum (X.\theta - Y)^2$$

      - N.B.: J n'est ni matrice, ni vecteur c'est un scalaire

ML : Régression Linéaire

Formulation matricielle

- **Les Gradients**
  - ✓ Définir une formule spécifique de calcul de gradient pour chaque paramètre
    - $\frac{\partial J(a,b)}{\partial a} = \frac{1}{m}\sum_{i=1}^{m} x^{(i)} \times (ax^{(i)} + b - y^{(i)})$
    - $\frac{\partial J(a,b)}{\partial b} = \frac{1}{m}\sum_{i=1}^{m} (ax^{(i)} + b - y^{(i)})$
  - ✓ Rassembler le calcul de ces 2 gradients en un vecteur gradient :
    - $\frac{\partial J(\theta)}{\partial \theta} = \begin{bmatrix} \frac{\partial J(a,b)}{\partial a} \\ \frac{\partial J(a,b)}{\partial b} \end{bmatrix}$ vecteur de dimension (2 x 1) ou bien (n+1 x 1)
    - Ce vecteur s'obtient par la formule suivante :
      - $\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{m} X^T . (X . \theta - Y)$

ML : Régression Linéaire

Formulation matricielle

- **La Descente de Gradient**
  - ✓ Rappel :
    - L'algorithme de Descente de Gradient met à jour les paramètres a et b du modèle de façon itérative avec 2 lignes de calculs
  - ✓ Disposant du **vecteur paramètre** $\theta = \begin{bmatrix} a \\ b \end{bmatrix}$
  - ✓ Et du **vecteur gradient** $\frac{\partial J(\theta)}{\partial \theta}$
  - ✓ L'algorithme de Descente de Gradient se résume à :
    - $\theta = \theta - \alpha \times \frac{\partial J(\theta)}{\partial \theta}$ qui est évidemment 1 vecteur de dimension (n+1) x 1

ML : Régression Linéaire

Travaux Pratiques N°2

- **Reprendre l'exemple des arbres de la forêt**
  - ✓ Implémenter une Machine Learning sous forme Matricielle en Python
  - ✓ Implémenter une Machine Learning sous forme Matricielle en R
  - ✓ Implémenter une Machine Learning sous forme Matricielle dans un autre langage

ML : Régression Polynomiale

Polynôme

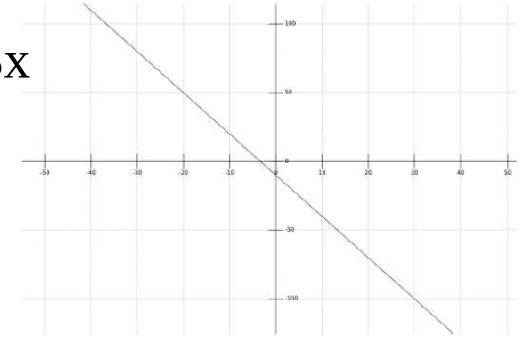
- **Qu'est ce qu'un monôme?**
  - ✓ Expression de la forme : αx^n
    - α :un nombre réel (ou complexe) appelé le **coefficient du monôme**
    - **n** : un entier naturel représentant le **degré du monôme**
  - ✓ Exemple
    - $3x^2$ est un monôme du second degré et de coefficient 3
    - $3 = 3x^0$ est un monôme de degré 0 et de coefficient 3
- **Qu'est ce qu'un polynôme?**
  - ✓ C'est la somme de plusieurs monômes. degré d'un polynôme **puissance la plus élevée de ses monômes**.
  - ✓ Exemple : $y = -5 + 3x$ est un polynôme de degré 1

ML : Régression Polynomiale

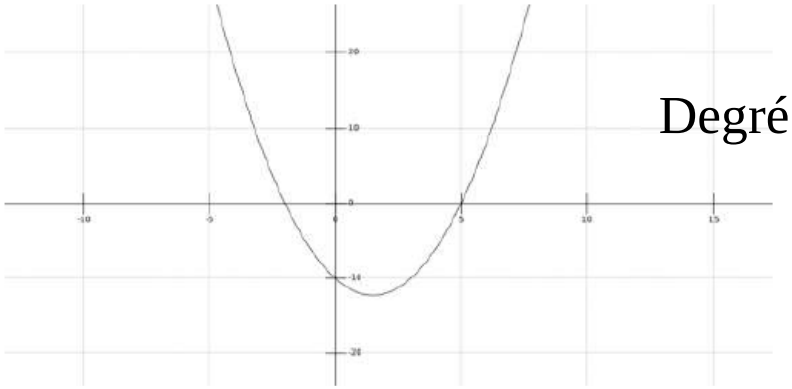
Polynôme : représentation graphique

- **Représentation graphique de polynômes :**

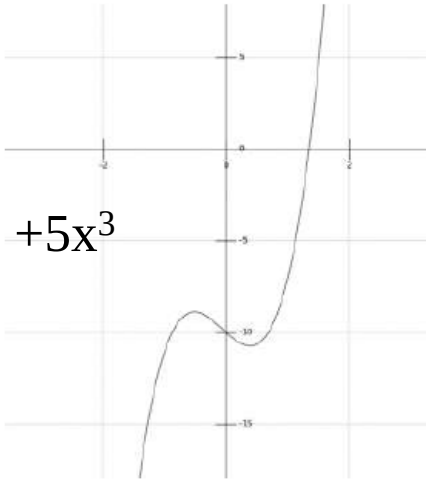
Degré 1 : $y = -10 -3x$



Degré 2 : $y = -10 -3x +x^2$



Degré 3 : $y = -10 -3x +x^2 +5x^3$



ML : Régression Polynomiale

Polynôme : fonction de prédiction

- **Dans la régression linéaire multivariée**
  - ✓ La fonction de prédiction est de la forme :
    - $F(X) = \varepsilon + \alpha x_1 + \beta x_2 + \gamma x_3 + \ldots + \omega x_n$
    - fonction de prédiction reste un polynôme de premier degré
- **En régression polynomiale**
  - ✓ On évalue chaque variable prédictive en l'associant à tous les degrés polynomiaux de 1 à k. Par exemple, si on a 2 variables x_1 et x_2, un modèle de second degré s'écrira comme suit :
    - $F(X) = \varepsilon + \alpha x_1 + \beta x_2 + \gamma x_1^2 + \omega x_2^2$
    - ε : une constante et α, β, γ …: coefficients de la fonction prédictive F(X)
    - X : est un vecteur/tableau de variables prédictives
    - x_i : représente la i^{eme} variable prédictive

ML : Régression Polynomiale

Exemple 1 en python

- **problème**
  - ✓ Soit un site de e-commerce souhaitant savoir si le temps de rechargement des pages web de son site impacte le montant du panier moyen de l'internaute
  - ✓ On souhaite trouver une fonction de prédiction qui modélisera cette corrélation
- **Solution**
  - ✓ **Les données** : ne disposant pas d'un vrai jeu de données, on va créer de manière aléatoire un

ML : Régression Polynomiale

Exemple 1 en python

- **Solution**

✓ **Les données** : le code

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3  #initialisation du générateur
4  np.random.seed(2)
5  # génération de 1000 nombres aléatoires distribués selon la loi normale X ~ N(3, 1)
6  #en d'autre terme, les pages chargeront en moyenne en 3 secondes et un ecart-type de 1 seconde.
7  tempsChargementPages = np.random.normal(3.0, 1.0, 1000)
8  génération aléatoires de montants d'achat corrélés aux temps de chargement
9  montantAchat = np.random.normal(50.0, 10.0, 1000) / (tempsChargementPages * tempsChargementPages)
10 # le temps de chargement de la page est notre variable prédictive
11 x = np.array(tempsChargementPages)
12 # montantAchat est la variable cible (qu'on cherche à prédire)
13 y = np.array(montantAchat)
```

✓ **Visualisation des données** : le code

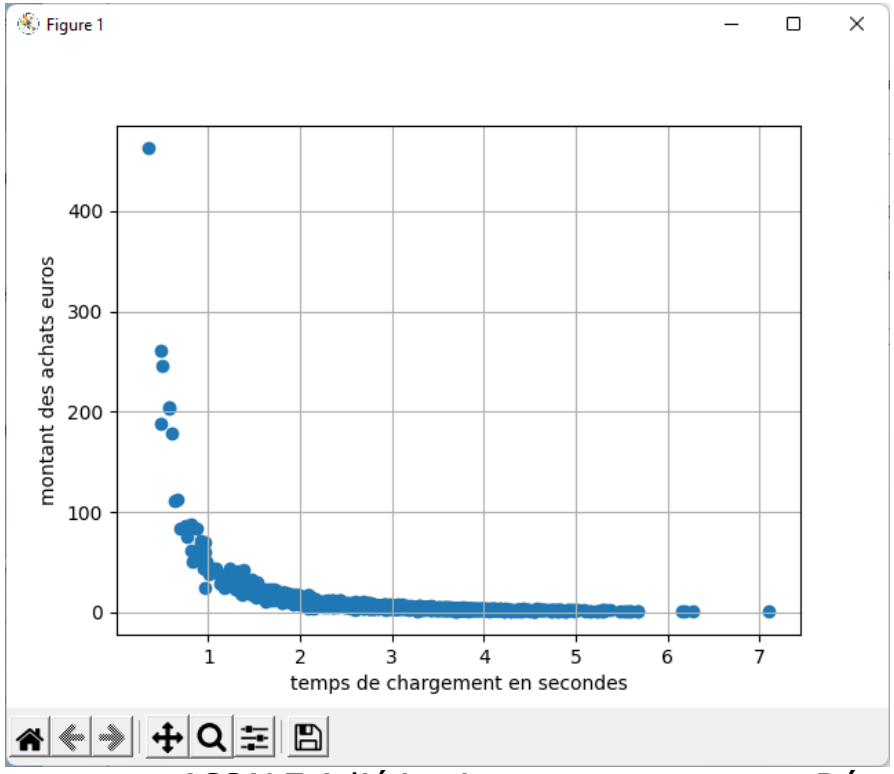
```
15 axes = plt.axes()
16 axes.grid()
17 plt.xlabel('temps de chargement en secondes')
18 plt.ylabel('montant des achats euros')
19 plt.scatter(tempsChargementPages, montantAchat)
20 plt.show()
```

ML : Régression Polynomiale

Exemple 1 en python

- **Solution**

✓ **Visualisation des données** : le graphique



On voit que la corrélation entre le prix du panier et le temps de chargement des pages du site **n'est pas linéaire**

ML : Régression Polynomiale

Exemple 1 en python

- **Solution**
  - ✓ **Modèle polynomial**: la fonction **polyfit** de numPy permet de construire un modèle polynomial de degré **n**. Le problème est de pouvoir déterminer le bonne valeur de n. on va essayer un polynôme de degré 4.

```
p4 = np.poly1d(np.polyfit(x, y, 4))
print(p4)
```

- Python trouve le polynôme suivant

```
          4         3         2
1.71 x - 27.23 x + 153.3 x - 362.8 x + 310.6
```

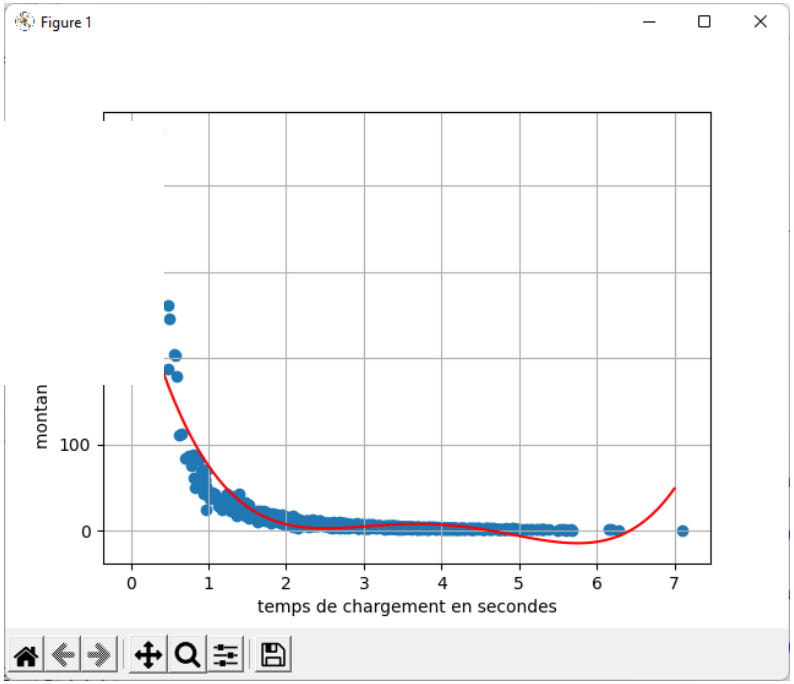
- ***F(X) = 1.71x⁴ -27.23x³ +153.3x² -362.8x +310.6***

ML : Régression Polynomiale

Exemple 1 en python

- **Solution**
  - ✓ **vérification**: vérifions à l'aide d'un graphique que notre fonction est bien adaptée

```
xp = np.linspace(0, 7, 100)
plt.plot(xp, p4(xp), c='r')
plt.show()
```



ML : Régression Polynomiale

Exemple 2 en python

- **Problème**
  - ✓ On va générer un jeu de données où la relation entre les variables explicatives et expliquées n'est pas linéaire
    - Code

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(color_codes=True)
plt.rcParams["figure.figsize"] = [12,12]

#plt.figure(figsize=(12,12))


np.random.seed(0)
#jeu de données sous la forme y = f(x)  avec f(x) = x^4 + bx^3 + c

x = np.random.normal(10, 2, 500)
y = x ** 4 + np.random.uniform(-1, 1,500)*(x ** 3) + np.random.uniform(0, 1,500)

plt.scatter(x,y)
plt.show()
```

ML : Régression Polynomiale

Exemple 2 en python

- **Problème**
  - Représentation graphique