Exercice 1

Exercice 2

L'entreprise Firrara de vente de T-shirt en ligne souhaite analyser les données de navigation de ses clients (Visiteurs) en observant chaque jour le nombre de visiteurs et le nombre de commandes. Le tableau suivant présente les résultats :

Nb visiteurs (xi)	10	11	12	15	16	20
Nb commandes (yi)	5	6	6	10	12	15

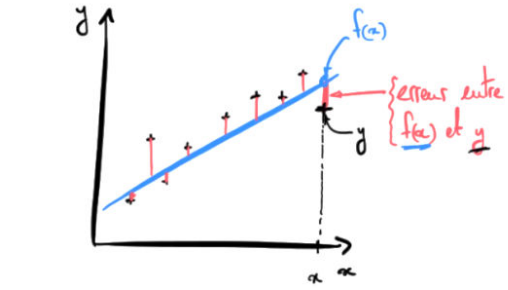
1. Présenter le nuage de points associé.
2. Calculer le barycentre (Point moyen) de cette série.
3. Calculer la covariance (Nombre visiteurs, Nombre commandes) et interpréter le résultat.
4. Calculer le coefficient de corrélation et déduire l'intensité de corrélation entre les deux variables.
5. Déterminer l'équation de la droite de régression linéaire $Y = aX + b$.
6. Le gérant fixe un objectif de recevoir 30 commandes par jour, estimer le nombre de visiteurs nécessaire permettant au gérant d'atteindre son objectif.

Problème

On peut représenter un nuage de points avec plusieurs droites. Alors comment choisir la meilleure droite?

Les erreurs de notre modèle - la Fonction Coût

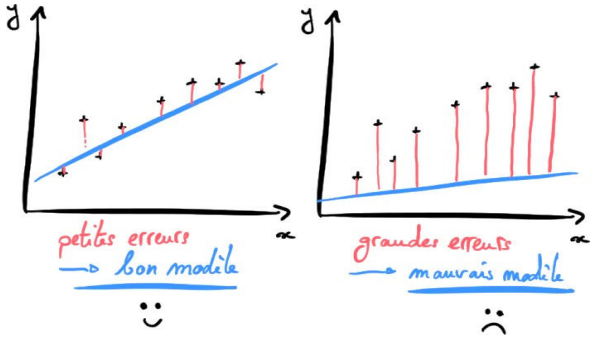
Autre chose à noter est qu'un modèle nous retourne des erreurs par rapport à notre Dataset. On appelle **Fonction Coût** l'ensemble de ces erreurs (le plus souvent on prend la moyenne quadratique des erreurs).



Fonction Coût = l'ensemble des erreurs.

Les erreurs de notre modèle - la Fonction Coût

Alors avoir un bon modèle, c'est avoir un modèle qui nous donne de petites erreurs, donc une **petite Fonction Coût**.



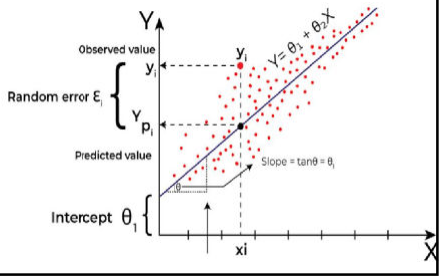
Principe de la régression

L'objectif de la régression linéaire est de trouver les paramètres w qui permettent au modèle f(x) de prédire des valeurs y aussi proches que possible des valeurs réelles.

Pour ajuster correctement w, nous devons :

1. **Définir une fonction d'erreur ou de perte** qui mesure l'écart entre les prédictions du modèle et les valeurs réelles.

2. **Minimiser cette fonction d'erreur** afin de trouver les meilleurs paramètres w.



Méthode: LES 4 ÉTAPES INDISPENSABLES DANS L'APPRENTISSAGE SUPERVISÉ

On doit suivre 4 étapes fondamentales pour résoudre un problème de l'apprentissage supervisé :

1. Utiliser ou importer un Dataset: ou jeux de données qui contient 2 types de variables x (features) et y (target),
2. Développer et Choisir un Modèle avec ses paramètres: la machine va apprendre ces paramètres en utilisant le Dataset.
3. Développer une Fonction coût associée à ce modèle : minimiser les erreurs,
4. Développer un Algorithme d'apprentissage pour minimiser la fonction coût (Hyper paramètres).

Méthode: LES 4 ÉTAPES INDISPENSABLES DANS L'APPRENTISSAGE SUPERVISÉ

1. Collecte les données

Imaginez que plusieurs agences immobilières vous aient fourni des données sur des appartements à vendre, notamment le prix de l'appartement (y) et la surface habitable (x). En Machine Learning, on dit que vous disposez de m exemples d'appartements.

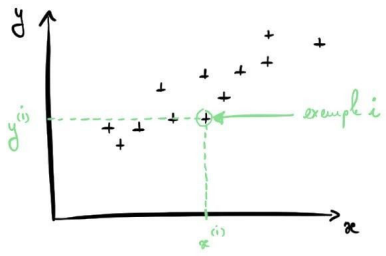
On désigne :

$X^{(i)}$ la surface habitable de l'exemple i

$y^{(i)}$ le prix de l'exemple i

Méthode: LES 4 ÉTAPES INDISPENSABLES DANS L'APPRENTISSAGE SUPERVISÉ

En visualisant le **Dataset**, vous obtenez le nuage de points suivant :



Méthode: LES 4 ÉTAPES INDISPENSABLES DANS L'APPRENTISSAGE SUPERVISÉ

2. Créer un modèle linéaire

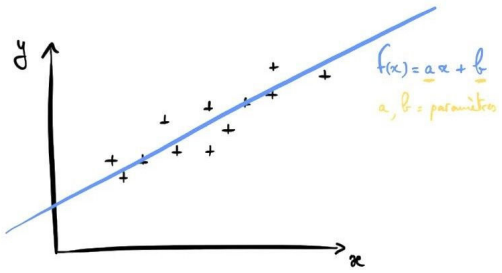
A partir de ces données, on développe un **modèle linéaire** $f(x) = ax + b$

où a et b sont les **paramètres** du modèle.

Un bon modèle donne de **petites** erreurs entre ses prédictions $f(x)$ et les exemples (y) du Dataset.

Nous ne connaissons pas les valeurs des paramètres a et b, ce sera le rôle de la machine de les trouver, de sorte à tracer un modèle qui s'insère bien dans notre nuage de point comme ci-dessous :

Méthode: LES 4 ÉTAPES INDISPENSABLES DANS L'APPRENTISSAGE SUPERVISÉ



Une Variable

$F(x) = ax + b$

Multi Variables

$F(x) = a_1x_1 + a_2x_2 + a_3x_3 + \ldots + b$

6

Méthode: LES 4 ÉTAPES INDISPENSABLES DANS L'APPRENTISSAGE SUPERVISÉ

3. Définir La Fonction Coût

Pour la régression linéaire, on utilise la **norme euclidienne** pour mesurer les **erreurs** entre $f(x)$ et (y)**.**

Concrètement, voici la formule pour exprimer l'erreur i entre le prix $y^{(i)}$ et la prédiction faites en utilisant la surface $x^{(i)}$:

$$erreur^{(i)} = (f(x^{(i)}) - y^{(i)})^2$$

7

Méthode: LES 4 ÉTAPES INDISPENSABLES DANS L'APPRENTISSAGE SUPERVISÉ

3. Définir La Fonction Coût

Par exemple, le 10ième exemple de votre Dataset soit un appartement de $x^{(10)}$ = 80 m^2 dont le prix s'élève à $y^{(10)}$ = 100,000 € et que votre modèle prédise un prix de $f(x^{(10)})$=100,002 €. L'erreur pour cette exemple est donc:

$$erreur^{(10)} = (f(x(10)) - y(10))^2$$

$$erreur^{(10)} = (100{,}002 - 100{,}000)^2 = (0{,}002)^2$$

Chaque prédiction s'accompagne d'une erreur, on a donc $\boldsymbol{m}$ erreurs.

On définit la **Fonction Coût** $(\boldsymbol{a}, \boldsymbol{b})$ comme étant la **moyenne** de toutes les erreurs :

8

Méthode: LES 4 ÉTAPES INDISPENSABLES DANS L'APPRENTISSAGE SUPERVISÉ

3. Définir La Fonction Coût

On définit la **Fonction Coût** $(\boldsymbol{a}, \boldsymbol{b})$ comme étant la **moyenne** de toutes les erreurs :

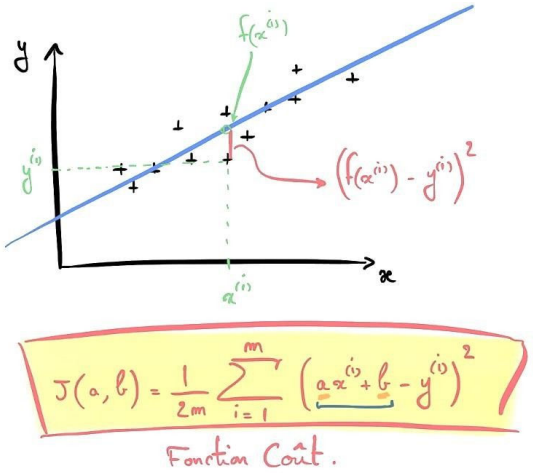
$$j(a,b) = \frac{1}{2m}\sum_{i=1}^{m} erreuri$$

$$j(a,b) = \frac{1}{2m}\sum_{i=1}^{m} \left(f(x^{(i)}) - y^{(i)}\right)^2$$

cette fonction est **l'erreur quadratique moyenne** (*Mean Squared Error*)

Méthode: LES 4 ÉTAPES INDISPENSABLES DANS L'APPRENTISSAGE SUPERVISÉ

3. Définir La Fonction Coût



Méthode: LES 4 ÉTAPES INDISPENSABLES DANS L'APPRENTISSAGE SUPERVISÉ

4. Trouver les paramètres qui minimisent la Fonction Coût

il s'agit de laisser la machine apprendre quels sont les paramètres qui **minimisent** la Fonction Coût, c'est-à-dire les paramètres qui nous donnent le meilleur modèle.

Pour trouver le minimum, on utilise un algorithme d'optimisation qui s'appelle **Gradient Descent** (la descente de gradient).

Méthode: LES 4 ÉTAPES INDISPENSABLES DANS L'APPRENTISSAGE SUPERVISÉ

Comprendre le Gradient Descent (la descente de gradient)

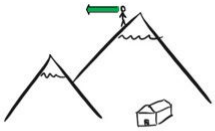
Imaginez-vous perdu en montagne. Votre but est de rejoindre le refuge qui se trouve au point le **plus bas** de la vallée. Vous n'avez pas pris de carte avec vous donc vous ne connaissez pas les coordonnées de ce refuge, vous devez le trouver tout seul.

Méthode: LES 4 ÉTAPES INDISPENSABLES DANS L'APPRENTISSAGE SUPERVISÉ

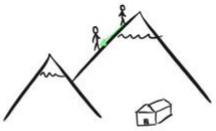
La descente de gradient

Pour vous en sortir, voici une stratégie à adopter :

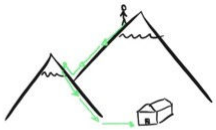
1. Depuis votre position actuelle, vous partez en direction de là où **la pente descend le plus fort.**
2. Vous avancez une certaine distance en suivant cette direction (même si ça implique de remonter une pente)
3. Une fois cette distance parcourue, vous répétez les 2 premières opérations en boucle, jusqu'à atteindre le point le plus bas de la vallée.



Etape 1: Trouver la pente la plus forte



Etape 2: Marcher une certaine distance dans cette direction



Etape 3: Répéter les étapes1 et 2 en boucle

Méthode: LES 4 ÉTAPES INDISPENSABLES DANS L'APPRENTISSAGE SUPERVISÉ

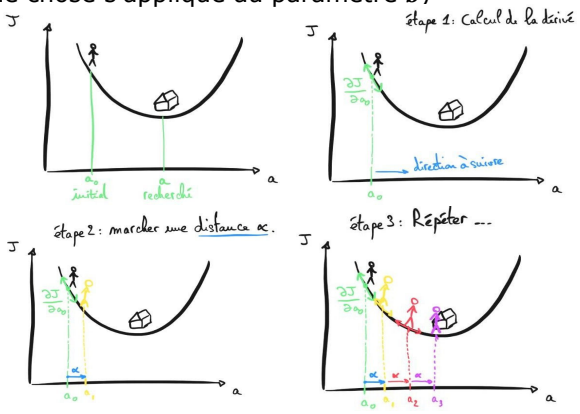
La descente de gradient

Les étapes 1, 2 et 3 forment ce qu'on appelle l'algorithme de **Gradient Descent.**

Cet algorithme vous permet de trouver le **minimum** de la Fonction Coût ($\boldsymbol{a}$, $\boldsymbol{b}$) (le point le plus bas de la montagne) en partant de coordonnées $\boldsymbol{a}$ et $\boldsymbol{b}$ **aléatoires** (votre position initiale dans la montagne) :

1. Calculer la **pente** de la Fonction Coût, c'est-à-dire la **dérivée** de $\boldsymbol{J(a, b)}$.
2. **Evoluer** d'une certaine **distance** $\propto$ dans la direction de la pente la plus forte. Cela a pour résultat de modifier les paramètres $\boldsymbol{a}$ et $\boldsymbol{b}$
3. Recommencer les étapes 1 et 2 jusqu'à atteindre le minimum de $\boldsymbol{J(a, b)}$.

La figure ci-dessous montre la recherche du paramètre $\boldsymbol{a}$ idéal (la même chose s'applique au paramètre $\boldsymbol{b}$)



Comment utiliser l'algorithme de Gradient Descent

- Pour rappel, nous avons jusqu'à présent créé un Dataset, développer un modèle aux paramètres **inconnus**, et exprimé la fonction Coût *J*(a,b) associée à ce modèle.
- Notre objectif final : Trouver les paramètres a et b qui **minimisent** *J*(a,b)
- Pour cela, nous allons choisir a et b au **hasard** (nous allons nous perdre en montagne) puis allons utiliser **en boucle** la descente de gradient pour mettre à jour nos paramètres dans la direction de la Fonction Coût la **plus faible**.

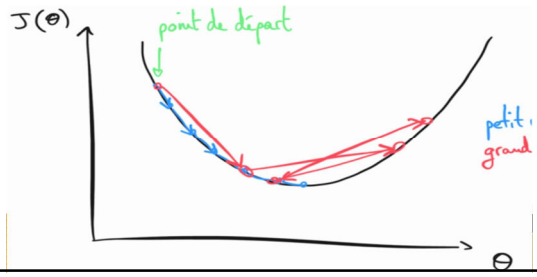
- Répéter en boucle

$$a = a - \propto \frac{\partial J(a,b)}{\partial a}$$

$$b = b - \propto \frac{\partial J(a,b)}{\partial b}$$

On appelle **α la vitesse d'apprentissage (Learning rate: hyper-paramètre)**.

Si la vitesse est trop lente, le modèle peut mettre longtemps à être entraîné, mais si la vitesse est trop grande, alors la distance parcourue est trop longue et le modèle peut ne jamais converger. Il est important de trouver un juste milieu. Le dessin ci-dessous illustre mes propos.



Utilisation des matrices et des vecteurs

- Dans la pratique, on exprime notre Dataset et nos paramètres sous forme matricielle, ce qui simplifie beaucoup les calculs. On créer ainsi un vecteur $\boldsymbol{\theta} = \binom{a}{b} \in \mathbb{R}^{n+1}$ qui contient tous les paramètres pour notre modèle, un vecteur $\boldsymbol{y} \in \mathbb{R}^{m\times 1}$ et une matrice $X \in \mathbb{R}^{m\times n}$ qui inclut toutes les features $\boldsymbol{n}$, Dans la régression linéaire $\boldsymbol{n}$=1.

$$X = \begin{bmatrix} x^{(1)} & 1 \\ \dots & \dots \\ x^{(m)} & 1 \end{bmatrix} \quad m \times (n+1) \qquad Y = \begin{bmatrix} y^{(1)} \\ \dots \\ y^{(m)} \end{bmatrix} \quad m \times 1 \qquad \theta = \begin{bmatrix} a \\ b \end{bmatrix} \quad (n+1) \times 1$$

Résumé des étapes de la Régression Linéaire

Les étapes pour développer un programme de Régression Linéaire

1. Dataset: Récolter des données $(\boldsymbol{x}, \boldsymbol{y})$ avec $\boldsymbol{m}$ exemples, et $\boldsymbol{n}$ variables tq:
$\boldsymbol{y} \in \mathbb{R}^{m\times 1}$, $\boldsymbol{x} \in \mathbb{R}^{m\times n}$
$\boldsymbol{y}$: target
$\boldsymbol{x}$: features

$$X = \begin{bmatrix} x^{(1)} & 1 \\ \dots & \dots \\ x^{(m)} & 1 \end{bmatrix} \quad m \times (n+1) \qquad Y = \begin{bmatrix} y^{(1)} \\ \dots \\ y^{(m)} \end{bmatrix} \quad m \times 1 \qquad \theta = \begin{bmatrix} a \\ b \end{bmatrix} \quad (n+1) \times 1$$

2. Donner à la machine un modèle linéaire: $\boldsymbol{f(x) = ax + b}$
$\boldsymbol{f(x) = x.\theta}$ où: $\theta = \begin{bmatrix} a \\ b \end{bmatrix}$

Résumé des étapes de la Régression Linéaire

Les étapes pour développer un programme de Régression Linéaire

1. **Dataset: Récolter des données** (x, y) avec m exemples, et n variables tq: $y \in \mathbb{R}^{m\times 1}$, $x \in \mathbb{R}^{m\times n}$

2. **Donner à la machine un modèle linéaire:** $f(x) = ax + b$

3. **Créer la Fonction Coût J:**

$$j(a,b) = \frac{1}{2m}\sigma_{i=1}^{m}\left(f(x^{(i)}) - y^{(i)}\right)^2$$

(Erreur Quadratique Moyenne)

4. **l'algorithme de minimisation:** Calculer le gradient et utiliser l'algorithme de **Gradient Descent**:

$$a = a + \alpha\frac{\partial J(a,b)}{\partial a}$$

$$b = b + \alpha\frac{\partial J(a,b)}{\partial b}$$

Exemple

Exemple de Dataset sur des appartements

Target y | Features x_1 x_2 x_3

Prix	Surface	Qualité	Adresse postale
313,000	90	3	95000
720,000	110	5	93000
250,000	40	4	44500
290,000	60	3	67000
190,000	50	3	59300
...	...	...	...

m (nombre de lignes) — n (nombre de colonnes)

$x_3^{(2)}$

Par convention:
m: nombre d'exemples
n: nombre de features

Par convention, on note: $x_{feature}^{(exemple)}$

Exemple

$Dataset\ (x, y)$

y	x_1	x_2	x_3	...	x_n
$y^{(1)}$	$x_1^{(1)}$	$x_2^{(1)}$	$x_3^{(1)}$	...	$x_n^{(1)}$
$y^{(2)}$	$x_1^{(2)}$	$x_2^{(2)}$	$x_3^{(2)}$	...	$x_n^{(2)}$
$y^{(3)}$	$x_1^{(3)}$	$x_2^{(3)}$	$x_3^{(3)}$	...	$x_n^{(3)}$
...	...	...	...	...	...
$y^{(m)}$	$x_1^{(m)}$	$x_2^{(m)}$	$x_3^{(m)}$	...	$x_n^{(m)}$

$vecteur\ target\ y \in \mathbb{R}^{m\times 1}$

$$y = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \dots \\ y^{(m)} \end{pmatrix}$$

$matrice\ features\ X \in \mathbb{R}^{m\times n}$

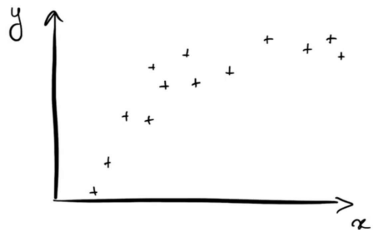
$$X = \begin{pmatrix} x_1^{(1)} & \dots & x_n^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(m)} & \dots & x_n^{(m)} \end{pmatrix}$$

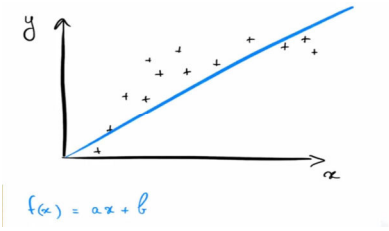
Régression Polynômiale

- Si vous achetez un stylo à 1DH, combien vous coûteront 100 stylos ? 100 DH ? Faux !
- Nous vivons dans un monde régit par des lois souvent non-linéaires et où une infinité de facteurs peuvent influencer nos résultats.Par exemple, si vous achetez 100 stylos, vous aurez peut-être une réduction à 90 €. Si en revanche il y a une pénurie de stylos, ce même stylo qui coûtait 1 € pourrait valoir 1.50 €
- C'est là qu'Excel ne pourra plus rien faire pour vous et que le Machine Learning trouve son utilité dans le monde réel.

Régression Polynômiale

En regardant le nuage de point correspendant, on peut voir qu' on ne peut le representer par une droite ou un modele lineare.

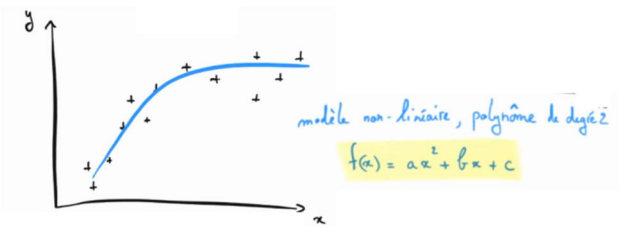




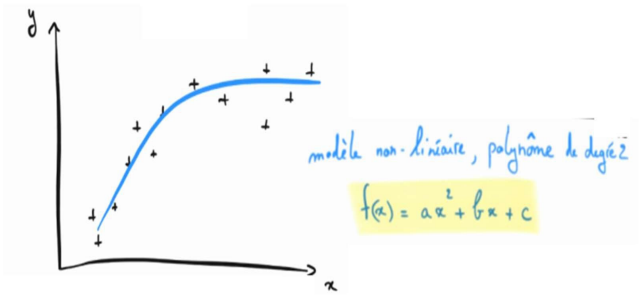
Régression Polynômiale à plusieurs variables

- Pour le nuage de point ci-dessous, il semblerait judicieux de développer un modèle polynômial de degré 2:

$$f(x) = ax^2 + bx + c$$



Régression Polynômiale à plusieurs variables



Ce modèle, plus **complexe** que le modèle linéaire précédent, va engendrer des calculs algébriques plus intenses, notamment le calcul des dérivées

Régression linéaire multiples

Le modèle de régression multiple est une généralisation du modèle de régression simple lorsque les variables explicatives sont en nombre fini.

Exemple de Dataset sur des appartements

Target y | Features x_1 x_2 x_3

Prix	Surface	Qualité	Adresse postale
313,000	90	3	95000
720,000	110	5	93000
250,000	40	4	44500
290,000	60	3	67000
190,000	50	3	59300
...	...	...	...

m (nombre de lignes) — n (nombre de colonnes)

Par convention:
m: nombre d'exemples
n: nombre de features

Quand utiliser la régression multiple

- Pour estimer la relation entre une variable dépendante (Y) et plusieurs variables indépendantes (X_1, X_2, …)
- **Exemples**
  - Expliquer les ventes d'un magasin par le marché total, le prix, l'investissement, la publicité,…
  - Expliquer la consommation des véhicules par le prix, la cylindrée, la puissance et le poids.

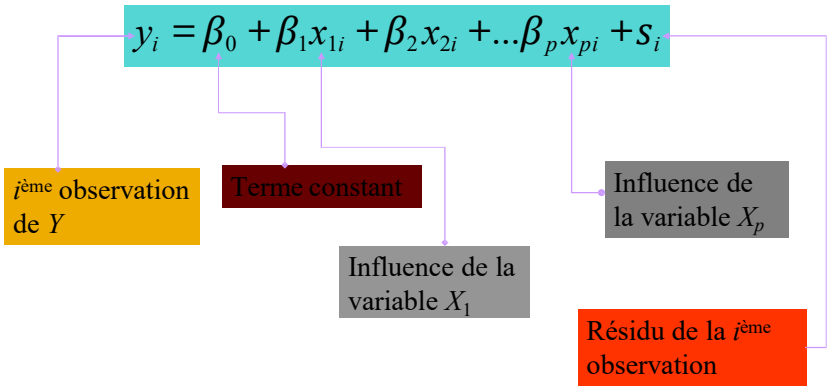
Le modèle général de régression multiple

- Equation de régression multiple
  Cette équation précise la façon dont la variable dépendante est relié $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... \beta_p X_p + s$

où $\beta_1, \beta_2, \ldots, \beta_p$ sont les paramètres et e est un bruit aléatoire représentant le terme d'erreur.

Le modèle général de régression multiple

- Les termes de l'équation



Le résidu de la i-ème observation est la différence entre la valeur réelle observée pour cette observation et la valeur prédite par le modèle.

Le modèle général de régression multiple

- Ecriture matricielle du modèle

$$\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_{1,1} & \cdots & x_{1,p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{1,n} & \cdots & x_{n,p} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

$$y = X\beta + \varepsilon$$

Votre premier programme de Machine Learning

L'environnement de ML

Il existe deux façons d'installer les outils de Machine Learning sur votre ordinateur : La bonne et la mauvaise.

En installant Anaconda, vous vous épargnez des heures de souffrance à taper des commandes à installer les packages, librairies et éditeur de texte indispensables pour faire du Machine

L’environnement de ML

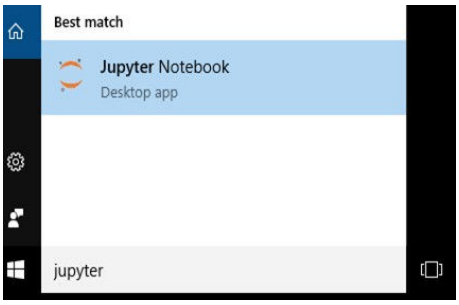
- Anaconda contient tous les outils et librairies dont vous avez besoin pour faire du Machine Learning : **Numpy**, **Matplotlib**, **Sklearn**, etc.

ANACONDA.

- Commencez par télécharger puis installer Anaconda depuis le site officiel : https://www.anaconda.com/distribution/#download-section

L’environnement de ML

Une fois Anaconda installé, vous pouvez lancer l’application **Jupyter Notebook** depuis votre barre de recherche Windows/Mac/Linux.



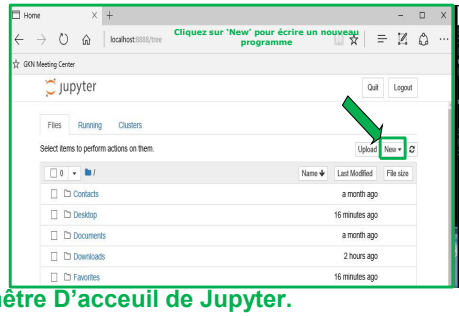
L'environnement de ML: Utilisation de Jupyter Notebook pour vos projets

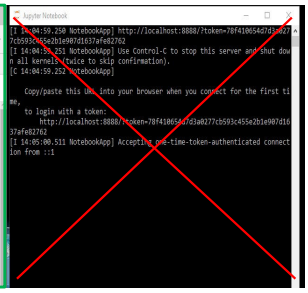
- Jupyter Notebook est une application Web qui permet de créer et de partager des codes Python.

- C'est une application Web, mais il n'est pas nécessaire d'avoir une connexion Internet pour vous servir de Jupyter. Aussi, vos données/codes ne sont à aucun moment stockés sur Internet (ce n'est pas du Cloud).

L'environnement de ML: Utilisation de Jupyter Notebook pour vos projets

Lorsque vous démarrez Jupyter, il est possible que 2 fenêtres s'ouvrent, auquel cas ne vous occupez pas de la fenêtre noire (la console) et surtout ne la fermez pas (ceci déconnecterait Jupyter de votre disque dur).





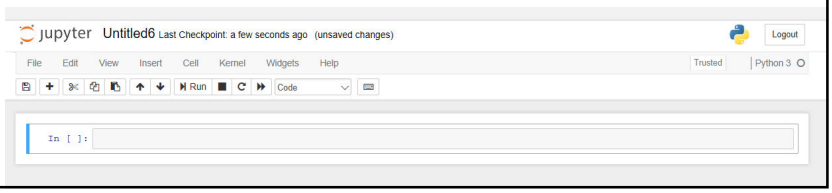
Fenêtre D'acceuil de Jupyter. Vous y trouvez les fichiers stockés sur votre disque dur.

Ne vous occupez pas de cette fenêtre. Ne la fermez surtout pas!

L'environnement de ML: Utilisation de Jupyter Notebook pour vos projets

La fenêtre principale de Jupyter n'est ni plus ni moins qu'un explorateur de fichier relié à votre disque dur, vous pouvez ouvrir tous vos fichiers, Dataset, codes, etc. depuis cette fenêtre.

Cliquez sur le bouton '*New*' situé en haut à droite de cette fenêtre pour commencer à écrire un nouveau programme (puis cliquez sur Python 3). La fenêtre suivante s'affiche alors : Vous êtes prêts à coder !



Atelier

Développer votre premier programme de Machine Learning

- Les étapes pour programmer une Régression Linéaire

Etape 1 : Importer les librairies:

1. **Numpy** pour manipuler notre Dataset en tant que matrice
2. **Matplotlib.pyplot** pour visualiser nos données
3. La fonction **make_regression** de **Sklearn** pour générer un nuage de point (ici on va simuler des données)
4. **SGDRegressor** (qui signifie Stochastic **Gradient Descent**
5. Regressor) et qui contient le calcul de la **Fonction Coût**, des **gradients**, de l'**algorithme** de **minimisation**, bref.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_regression
from sklearn.linear_model import SGDRegressor
```

Etape 2 : Créer un Dataset

Pour ce premier code, nous n'allons pas importer de données personnelles. Plutôt, nous allons générer un tableau de données (x,y) aléatoires, Pour cela, la fonction **make_regression** est très utile. La fonction prend comme **arguments** (c'est le mot pour désigner inputs) le nombre d'échantillons à générer, le nombre de variables et le bruit puis nous retourne deux vecteurs **x** et **y**

Pour maitriser l'aléatoire, on écrit la ligne **np.random.seed(0).**

Finalement, pour visualiser nos données on utilise la fonction **plt.scatter(x, y).**

```
np.random.seed(0)
x, y = make_regression(n_samples=100, n_features=1, noise=10)
plt.scatter(x, y)
```

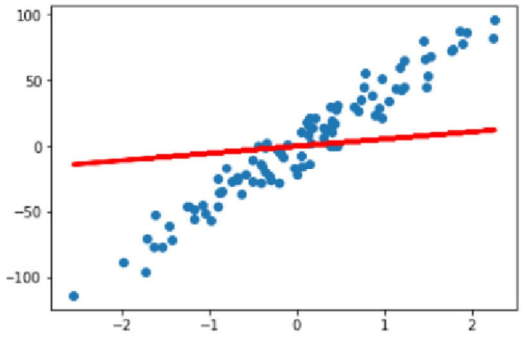
Etape 3 : Développer le modèle et l'entraîner

- Pour développer et entraîner un modèle, il a fallu beaucoup de maths: Entre la Fonction Coût, les dérivées, l'algorithme de Gradient Descent…
- Dans **Sklearn**, tout cela est déjà fait pour vous !
- Il vous suffit de définir une variable model depuis le générateur **SGDRegressor** en entrant le nombre d'itérations que le Gradient Descent doit effectuer ainsi que le Learning Rate.
- Une fois le modèle définit, il vous faut **l'entraîner**. Pour cela, il suffit d'utiliser la fonction **fit**.

```
model = SGDRegressor(max_iter=100, eta0=0.0001)
model.fit(x,y)
```

Nous pouvons maintenant observez la précision de notre modèle en utilisant la fonction **score** qui calcule le **coefficient de détermination** entre le modèle et les valeurs ***y*** de notre Dataset.
On peut aussi utiliser notre modèle pour faire de nouvelles prédictions avec la fonction **predict** et tracer ces résultats avec la fonction **plt.plot** :

```
print('Coeff R2 =', model.score(x, y))
plt.scatter(x, y)
plt.plot(x, model.predict(x), c='red', lw = 3)
```

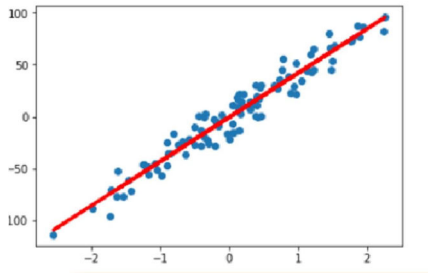


- Notre modèle semble vraiment mauvais. C'est parce que nous ne l'avons pas entraîné suffisamment longtemps et parce que le **Learning rate** était trop faible. Aucun problème, il est possible de le ré-entraîner avec de meilleurs hyper-paramètres.

- En Machine Learning, les valeurs qui fonctionnent bien pour la plupart des entraînements sont :

Nombre d'itérations = 1000

Learning rate = 0.001

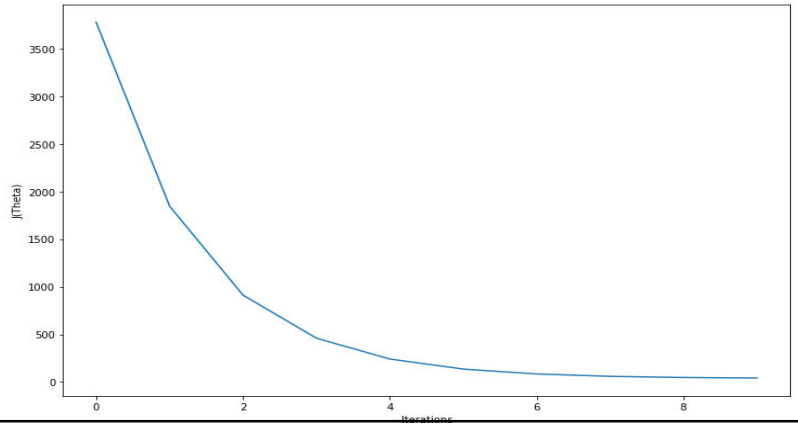


Les courbes d'apprentissage

En Machine Learning, on appelle courbe d'apprentissage (Learning curves) les courbes qui montrent l'évolution de la Fonction Coût au fil des itérations de Gradient Descent. Si votre modèle apprend, alors sa Fonction Coût doit diminuer avec le temps,

```
fig,ax = plt.subplots(figsize=(12,8))

ax.set_ylabel('J(Theta)')
ax.set_xlabel('Iterations')
_=ax.plot(range(iterations),cost_history)
```



À RETENIR

- La régression en apprentissage supervisé est une technique statistique qui vise à modéliser la relation entre une **variable dépendante** (**la variable à prédire**) et une ou plusieurs **variables indépendantes** (**les prédicteurs**).
- Le but est de construire un modèle mathématique qui permet de prédire la valeur de la variable dépendante à partir des valeurs des variables indépendantes.
- Le choix du modèle de régression dépend de la nature des données, de la relation entre les variables et de l'objectif de l'analyse.

À RETENIR

Voici les points clés à retenir :

- **Variable dépendante:** La variable que l'on cherche à prédire.
- **Variables indépendantes:** Les variables qui influencent la variable dépendante.
- **Modèle mathématique:** Une équation qui décrit la relation entre les variables.
- **Estimation des paramètres:** Le processus de détermination des meilleurs coefficients pour le modèle.
- **Prédiction:** L'utilisation du modèle pour estimer de nouvelles valeurs de la variable dépendante.

À RETENIR

- **Fonction de coût:** Une fonction qui mesure l'erreur entre les prédictions du modèle et les valeurs réelles.
- **Optimisation:** Le processus d'ajustement des paramètres du modèle pour minimiser la fonction de coût.
- **Algorithmes d'optimisation:** Des algorithmes comme la descente de gradient sont utilisés pour optimiser les paramètres.

Comment évaluer les performances d'un algorithme de Régression

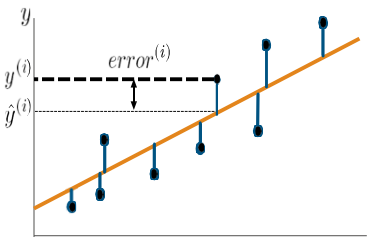
Métriques d'Évaluation des modèles de Régression

Elle permet de mesurer la performance du modèle sur un ensemble de données de test, ce qui nous aide à comprendre sa capacité de généralisation à de nouvelles données.

Une métrique est une valeur numérique permet de **quantifier la qualité des prédictions d'un modèle**.

Évaluation des performances d'un modèle de Régression

- On se base sur le calcul du résidus ou erreurs (residuals).
- Pour chaque observation du dataset on calcule la distance entre la valeur réelle (étiquette) et la valeur prédite.



$y^{(i)}$: Valeur réelle

$\hat{y}^{(i)}$: Valeur prédite

n : Nombre d'enregistrements

Évaluation des performances d'un modèle de Régression

Plusieurs métriques peuvent être calculer pour évaluer l'erreur du modèle de régression. Les plus courants comprennent:

- **Erreur absolue Moyenne** (Mean absolute Error **MAE**)

$$MAE = \frac{1}{n}\sum_{i=0}^{n}|y^{(i)} - \hat{y}^{(i)}|$$

Mesure la moyenne de l'ordre de grandeur des erreur de prédiction

Une MAE proche de 0 signifie que le modèle s'ajuste bien aux données

Plus la valeur est faible, mieux c'est

Évaluation des performances d'un modèle de Régression

n

- **Erreur quadratique moyenne** (Mean Squared Error **MSE**): Moyenne des carrés des erreurs entre les valeurs prédites et les valeurs réelles

$$MSE = \frac{1}{n}\sum_{i=0}^{n}(y^{(i)} - \hat{y}^{(i)})^2$$

Le carré entraîne des pénalités plus importantes pour les erreurs larges.

Évaluation des performances d'un modèle de Régression

- **La racine de la MSE** (Root mean squared error **RMSE** ou RMS): Racine carrée de la MSE

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=0}^{n}(y^{(i)} - \hat{y}^{(i)})^2}$$

Pour que l'erreur soit avec la même unité que la variable cible y

- Une autre métrique pour mesurer la précision des prédictions
- Même unité que la variable dépendante

Plus la valeur est faible, mieux c'est

Évaluation des performances d'un modèle de Régression

- **R-Squared R^2** (Coefficient de détermination)

$$R^2 = 1 - \frac{\sum_{i=0}^{n}(y^{(i)} - \hat{y}^{(i)})^2}{\sum_{i=0}^{n}(y^{(i)} - \bar{y})^2}$$

Mesure populaire qui représente la proportion de la variance de y qui est prédictible par le modèle.

La valeur du R^2 est comprise entre 0-1 :

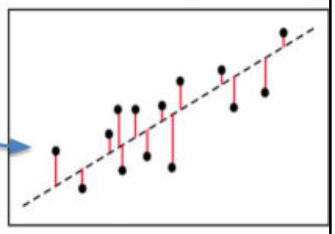
- R^2 proche de 1: le modèle s'ajuste bien
- R^2 proche de 0: pas mieux que de deviner la valeur moyenne

Plus elle est élevée, mieux c'est

Exemple

Obs	Height in Inches, X	Act Weight in Pounds, Y	Predicted Weight $\hat{Y}$	Residual/ Error $e_i = Y_i - \hat{Y}_i$	Residual^2/ Error^2 $e_i^2 = (Y_i - \hat{Y}_i)^2$
1	63	127	120.1	6.900	47.61
2	64	121	126.3	-5.300	28.09
3	66	142	138.5	3.500	12.25
4	69	157	157.0	0.000	0
5	69	162	157.0	5.000	25
6	71	156	169.2	-13.200	174.24
7	71	169	169.2	-0.200	0.04
8	72	165	175.4	-10.400	108.16
9	73	181	181.5	-0.500	0.25
10	75	208	193.8	14.200	201.64
				0.000	597.28

Sum of Squared Residuals



$$R^2 = 1 - \frac{\frac{1}{n}\sum_{i=1}^{n}(y_i - yhat_i)^2}{\frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - yhat_i| \qquad RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - yhat_i)^2}$$

Fin