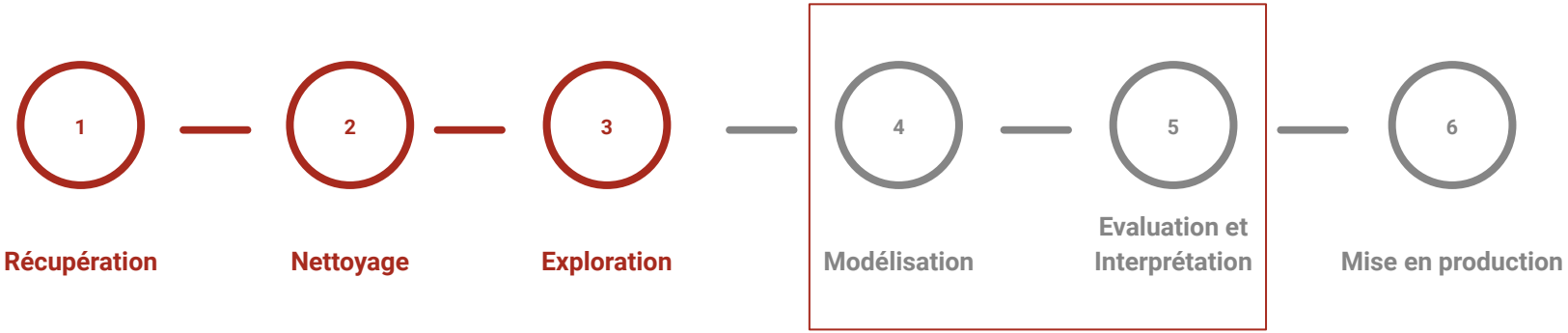
Introduction au Machine Learning

Partie 1 - La régression linéaire

Octobre 2020

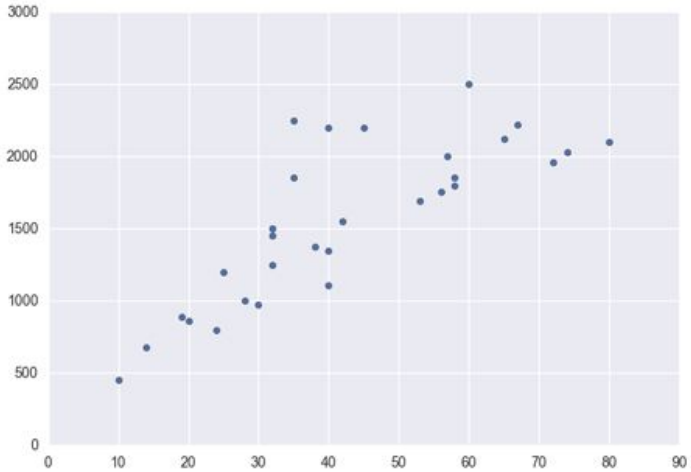
Cycle de travail du développeur IA



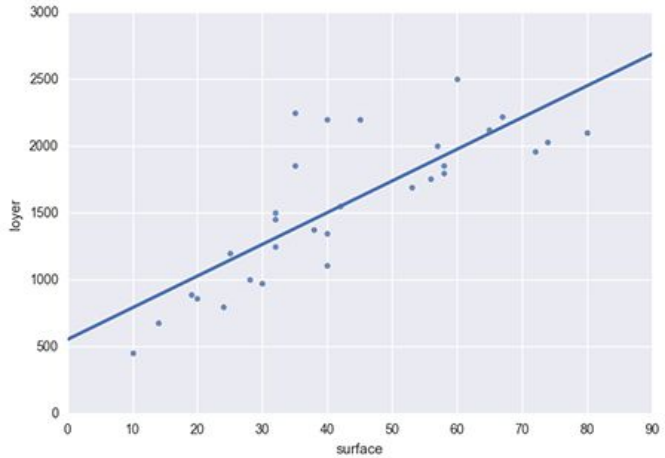
Modélisation et évaluation:
En machine learning, l'objectif est de trouver un modèle du phénomène à l'origine des données. C'est-à-dire qu'on considère que chaque donnée observée est l'expression d'une variable aléatoire générée par une distribution de probabilité.

Le principe de la modélisation

En machine learning, l'objectif est de trouver un modèle du phénomène à l'origine des données. C'est-à-dire qu'on considère que chaque donnée observée est l'expression d'une variable aléatoire générée par une distribution de probabilité.

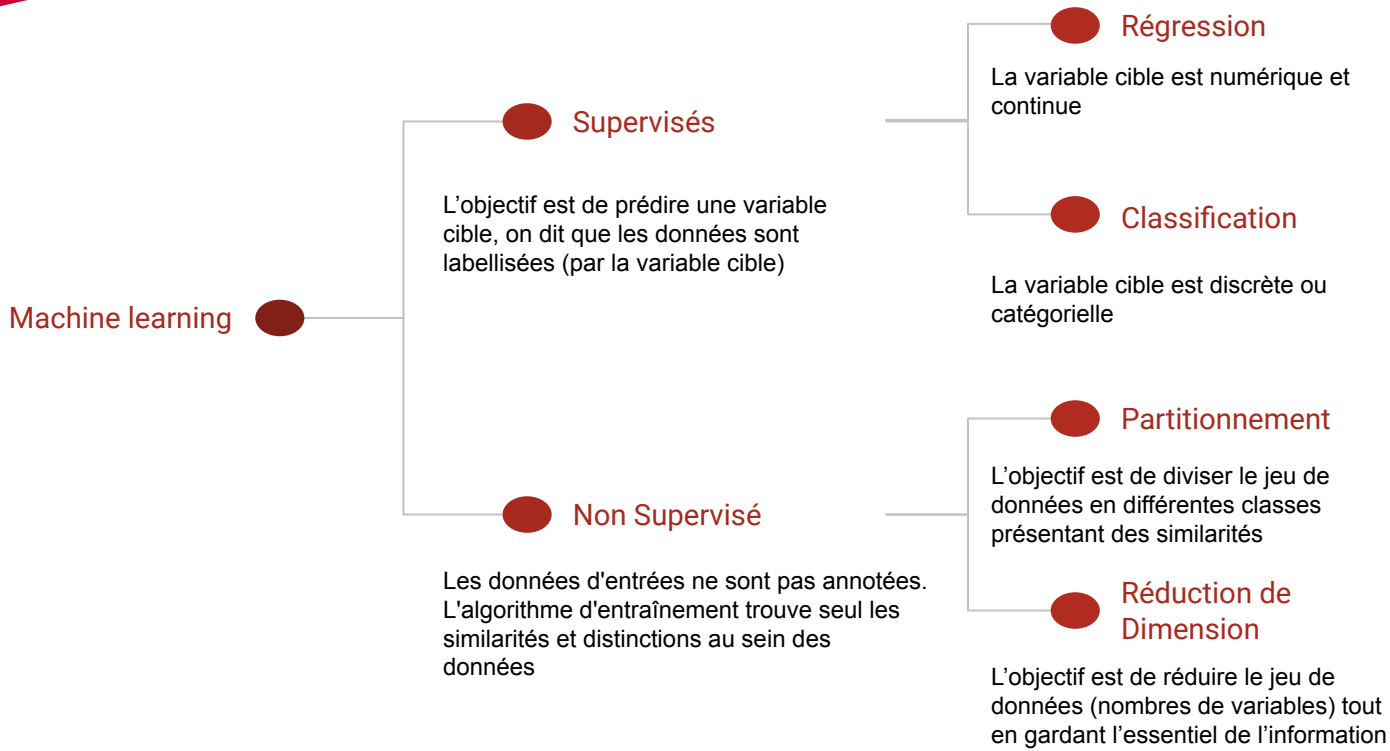


On a représenté ici le loyer d'un appartement en fonction de sa superficie



Une première modélisation simple du phénomène (le prix du loyer) serait donc de considérer la droite la plus "proche" de l'ensemble des points.

Les grands types de problèmes du ML



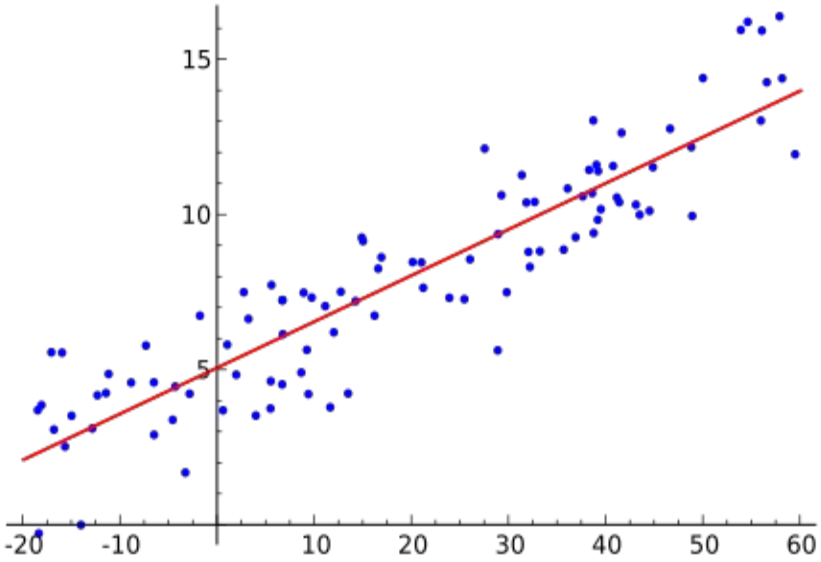
Les principaux algorithmes

Algorithmes	Problèmes correspondant
Régression Linéaire: Cherche à établir une relation linéaire entre deux variables ou davantage	Régression (Supervisé)
Régression logistique (LOGIT): permet d'étendre la régression linéaire aux données discrètes et catégorielles.	Classification (Supervisé)
Classification and Regression Trees (CART): Labellise une instance en suivant le processus d'un arbre de décision. Ces arbres peuvent être combiné de manière séquentielle (**XGboost**) ou non (**Random Forest**)	Régression et Classification (Supervisé)
K nearest neighbors (KNN): Prédire une valeur cible à partir des valeurs des instances qui lui ressemble	Régression et Classification (Supervisé)
Support Vector Machine (SVM): Généralisation des **classificateurs linéaires** qui cherchent à diviser un jeu de données en fonction de leur ressemblance (les multiples divisions ne sont pas séquentielles)	Régression et Classification (Supervisé)
Réseaux de Neurones: Analyse de la données à la fois de manière parallélisée et séquentielles	Régression et Classification (Supervisé) Réduction dimension et partitionnement
K mean: divise un jeu de données en k groupes, souvent appelés clusters, de façon à minimiser une certaine fonction	Partitionnement (Non Supervisé)
Analyse en composantes principales (ACP): transforme des variables corrélées en nouvelles variables décorrélées les unes des autres	Réduction de dimension (Non Supervisé)

La régression linéaire

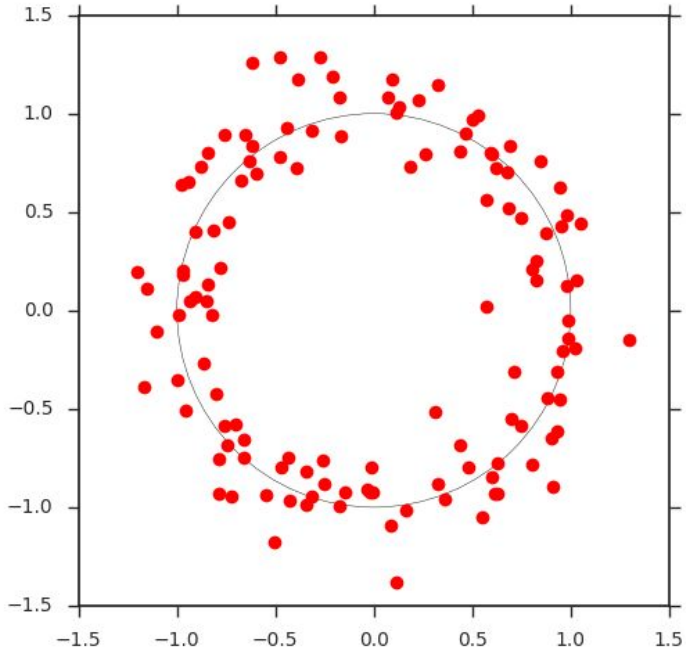
On considère que toute donnée (point bleu) est la combinaison d'un modèle sous jacent (droite rouge) et d'un certain bruit ou aléa indépendant. (écart entre le point et la droite)

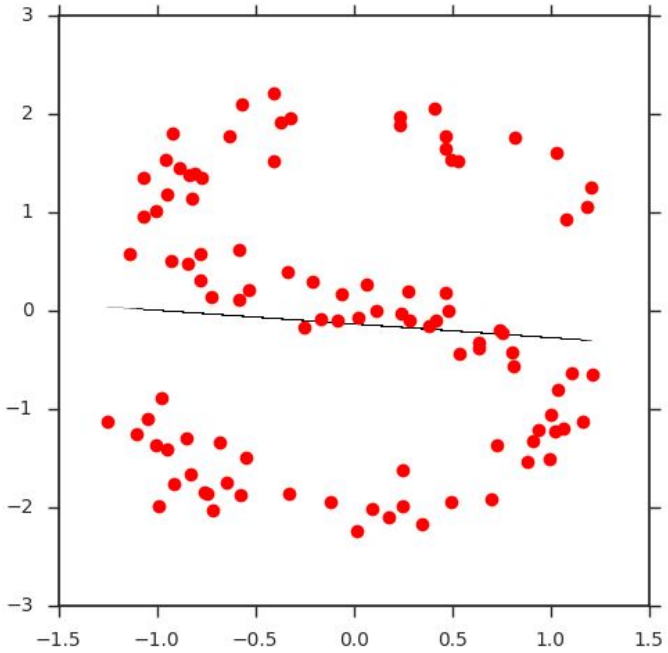
Dans le cas de la régression linéaire, le modèle sous jacent est une droite.



La régression linéaire

Il est important de choisir le modèle sous jacent adapté aux données. Dans les cas suivants il serait impossible de résumer convenablement les nuages de points à l'aide d'une droite:





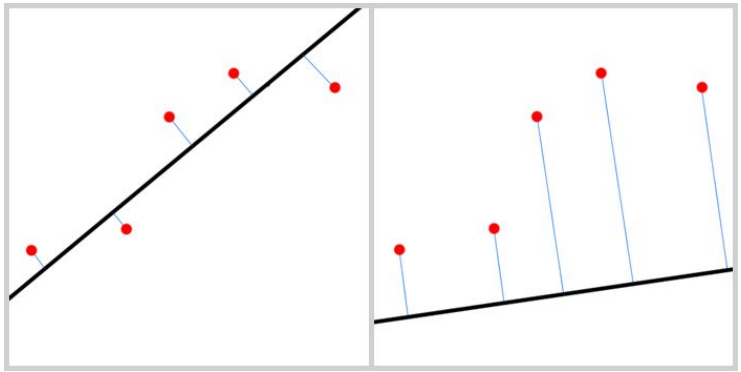
Fonction loss

En apprentissage supervisé, la notion principale est celle de **perte d'information** (loss) due à l'approximation. Elle détermine à quel point notre modélisation du phénomène perd de l'information par rapport à la réalité observée à travers les données d'exemple.

L'apprentissage se résume souvent à une méthode itérative qui converge vers un minimum de cette fonction. Plus la perte d'information diminue, plus on se rapproche de la réalité et meilleur est notre modèle.

Exemple de fonction de perte : L'erreur quadratique

C'est la distance euclidienne (trait bleu) entre la donnée (point rouge) et le modèle (**ligne noire**)



Régression linéaire: aspect théorique

On a un jeu de N données, portant sur le loyer “y” et la surface “x” d’appartements. On cherche un modèle permettant d’expliquer le loyer “y” par la surface ‘x”

On cherche donc une droite qui ferait le lien entre x et y. L’équation d’une droite s’écrit: $y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$

Minimiser l’écart entre nos données réelles et cette droite, revient à écrire: $\text{Argmin}_{\beta_0,\beta_1} \sum_{i=1}^{n} (y_i - \beta_1 x_i - \beta_0)^2$

On cherche alors le point où la dérivée (selon β0 et β1) et on obtient après simplification:

$$\begin{cases} \hat{\beta}_1 = \dfrac{\text{cov}(X, Y)}{\text{var}(X)} \\ \hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} \end{cases}$$

Empiriquement β0 et β1 se calculent à partir d’une estimation de la matrice de variance covariance.

Régression linéaire: évaluation de la modélisation

Trois indicateurs clefs (vrais au delà de la régression linéaire)

La variation expliquée par la régression (Sum of Squares Explained [par la régression]). C'est la somme des différences entre la valeur prédite et la valeur moyenne observée.

$$\mathrm{SSE} = \mathrm{SCE} = \sum_{i=1}^{n} (\hat{y}_i - \bar{y})^2$$

La variation expliquée par les résidus (Sum of Squared Residuals). C'est la différence entre la valeur prédite et la valeur théorique.

$$\mathrm{SSR} = \mathrm{SCR} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

La variation totale (*Sum of Squares Total*) correspond à la variance de y*n. D'après le calcul:

SST= $\sum_{i=1}^{n} (y_i - \bar{y})^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \sum_{i=1}^{n} (\hat{y}_i - \bar{y})^2 + \sum_{i=1}^{n} 2(\hat{y}_i - \bar{y})(y_i - \hat{y}_i).$

Or dans le cas de la régression le linéaire simple, le dernier élément est souvent nul. On a donc:

SST= SSE + SSR

Une partie de la variance est expliqué par le modèle, une partie reste résiduelle.

Régression linéaire: évaluation de la modélisation

On comprend que plus un modèle sera performant plus la proportion de la variance sera forte et plus celle expliquée par le résidu sera faible.

C'est tout le principe du **Coefficient de détermination R2**

$$R^2 = \frac{\text{SSE}}{\text{SST}} = \frac{\text{SST} - \text{SSR}}{\text{SST}} = \frac{\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Le coefficient de détermination varie entre 0 et 1. Lorsqu'il est proche de 0, le pouvoir prédictif du modèle est faible et lorsqu'il est proche de 1, le pouvoir prédictif du modèle est fort.

Puisque le calcul du R2 passe par celui du SST pourquoi est ce qu'on ne se sert pas directement du SST (distance entre la valeur prédite et la valeur réelle) directement comme étiquette? Le problème c'est que cette valeur augmente avec le nombre de données, il faut donc la retravailler un peu. C'est le principe de **RMSE (racine de l'erreur quadratique moyenne)**:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(f(x_i) - y_i)^2}$$

Plus le RMSE est faible, meilleure est la prédiction. Cependant le RMSE dépend de l'ordre de grandeur du jeu de données, il peut donc être plus difficile à interpréter et il ne rend pas possible les comparaisons entre deux jeux de données qui n'ont pas le même ordre de grandeur

Régression linéaire: évaluation de la modélisation

L'entraînement d'un modèle revient à mesurer l'erreur de la sortie de l'algorithme avec les données d'exemple et chercher à la minimiser.

Un premier piège à éviter est donc d'évaluer la qualité de votre modèle final à l'aide des mêmes données qui ont servi pour l'entraînement. En effet, le modèle est complètement optimisé pour les données à l'aide desquelles il a été créé. L'erreur sera précisément minimum sur ces données. Alors que l'erreur sera toujours plus élevée sur des données que le modèle n'aura jamais vues !

Pour minimiser ce problème, la meilleure approche est de séparer dès le départ notre jeu de données en deux parties distinctes :

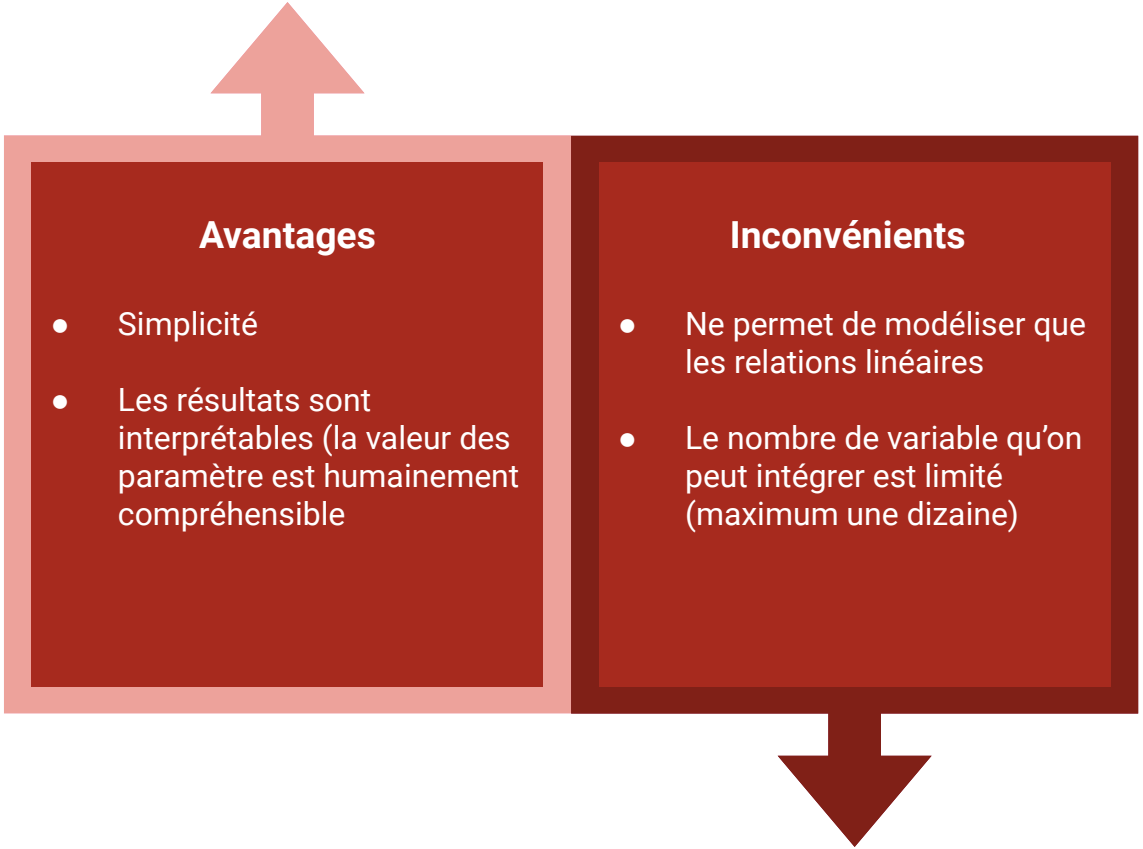
- **Le training set,** qui va nous permettre d'entraîner notre modèle et sera utilisé par l'algorithme d'apprentissage. C'est celui dont on a parlé depuis le début.

- **Le testing set**, qui permet de mesurer l'erreur du modèle final sur des données qu'il n'a jamais vues. On va simplement passer ces données comme s'il s'agissait de données que l'on n'a encore jamais rencontrées (comme cela va se passer ensuite en pratique pour prédire de nouvelles données) et mesurer la performance de notre modèle sur ces données.

C'est à vous de définir la proportion du dataset que vous souhaitez allouer à chaque partie. En général, les données sont séparées avec les proportions suivantes : **80 % pour le training set et 20 % pour le testing set.**

(**Le validation set** – qui permet de mesurer l'erreur de prédiction pour choisir entre plusieurs modèles – est aussi souvent utilisé. Nous l'étudierons dans les prochains cours.)

Avantages et inconvénients de la régression linéaire



Les étapes d'un projet de machine learning (détaillé)

1. Importation des données
2. Nettoyage des données
3. Exploration des données (statistiques univariées et bivariées)
4. Choix du modèle de machine learning
5. Préparation du jeu de données en vue du machine learning
    a. échantillonnage (si besoin) en vue de l'apprentissage
    b. sélection de la variable cible et des variables à explicatives
    c. standardisation et normalisation (si besoin) en fonction du modèle ML choisi
6. Division du jeu de données en training/validation/testing sets
7. Définition des paramètres (si besoin)
8. Apprentissage sur le training set
9. Comparaison des différents modèles sur le validation set
10. Evaluation du modèle sur le testing set
11. Mise en production

Les étapes grisées ne sont pas nécessaires pour les modèles linéaires

Sources

Openclassroom: initiez vous au ML:

https://openclassrooms.com/fr/courses/4011851-initiez-vous-au-machine-learning/5869331-decouvrez-le-domaine-de-la-data-science

Wikipédia

https://www.wikiwand.com/fr/R%C3%A9gression_lin%C3%A9aire

Kaggle: Intro to Machine learning

https://www.kaggle.com/learn/intro-to-machine-learning