# [Initiez-vous au Machine Learning](https://openclassrooms.com/fr/courses/4011851-initiez-vous-au-machine-learning)

Table des matières

[Initiez-vous au Machine Learning 1](#_Toc57616036)

[**Découvrez le domaine de la Data Science** 6](#_Toc57616037)

[**Qu’est-ce que la data science ?** 6](#_Toc57616038)

[**Tout commence par des données** 6](#_Toc57616039)

[**Une problématique bien définie** 7](#_Toc57616040)

[**Data Science et analyse de données** 8](#_Toc57616041)

[**En résumé** 9](#_Toc57616042)

[Plongez-vous dans la peau d’un Data scientist 10](#_Toc57616043)

[Appréhendez le cycle de travail du data scientist 10](#_Toc57616044)

[Récupérez les données 10](#_Toc57616045)

[Nettoyez les données 12](#_Toc57616046)

[Explorez les données 13](#_Toc57616047)

[Modélisez les données à l'aide du machine learning 14](#_Toc57616048)

[Évaluez et interprétez les résultats 16](#_Toc57616049)

[Déployez le modèle en production 17](#_Toc57616050)

[En résumé 18](#_Toc57616051)

[Identifez les différentes étapes de modélisation 19](#_Toc57616052)

[Démystifions le machine learning 19](#_Toc57616053)

[Avant la modélisation 20](#_Toc57616054)

[L'étape de modélisation 21](#_Toc57616055)

[Les données 23](#_Toc57616056)

[La tâche à accomplir 25](#_Toc57616057)

[L'algorithme d'apprentissage 26](#_Toc57616058)

[La mesure des performances 26](#_Toc57616059)

[En résumé 26](#_Toc57616060)

[Identifiez les possibilités du Machine Learning 28](#_Toc57616061)

[Compétences évaluées 28](#_Toc57616062)

[ Question 1 28](#_Toc57616063)

[ Question 2 28](#_Toc57616064)

[ Question 3 29](#_Toc57616065)

[ Question 4 29](#_Toc57616066)

[ Question 5 29](#_Toc57616067)

[ Question 6 30](#_Toc57616068)

[ Question 7 30](#_Toc57616069)

[ Question 8 31](#_Toc57616070)

[Transformez des besoins métiers en problèmes de Machine Learning 32](#_Toc57616071)

[Affectez un score à un client 32](#_Toc57616072)

[Prédisez la rentabilité d’une campagne marketing 32](#_Toc57616073)

[Identifiez les événements rares 33](#_Toc57616074)

[Affectez une catégorie à un produit 34](#_Toc57616075)

[Segmentez les visiteurs d’un site 35](#_Toc57616076)

[Recommandez un produit à un client 35](#_Toc57616077)

[En résumé 38](#_Toc57616078)

[**Sélectionnez les outils de Data Science appropriés** 39](#_Toc57616079)

[**"R ou Python ?"** 39](#_Toc57616080)

[**La récupération des données** 40](#_Toc57616081)

[**Le nettoyage et l'exploration des données** 41](#_Toc57616082)

[**La phase de modélisation (et l'évaluation)** 42](#_Toc57616083)

[**Le déploiement et la mise en production** 42](#_Toc57616084)

[**Passez par une API pour gagner en simplicité** 43](#_Toc57616085)

[**En résumé** 44](#_Toc57616086)

[Identifiez les techniques et outils du Machine Learning 45](#_Toc57616087)

[Compétences évaluées 45](#_Toc57616088)

[ Question 1 45](#_Toc57616089)

[ Question 2 45](#_Toc57616090)

[ Question 3 45](#_Toc57616091)

[ Question 4 46](#_Toc57616092)

[ Question 5 46](#_Toc57616093)

[ Question 6 46](#_Toc57616094)

[ Question 7 47](#_Toc57616095)

[ Question 8 47](#_Toc57616096)

[Identifiez les techniques et outils du Machine Learning 48](#_Toc57616097)

[Compétences évaluées 48](#_Toc57616098)

[ Question 1 48](#_Toc57616099)

[ Question 2 48](#_Toc57616100)

[ Question 3 48](#_Toc57616101)

[ Question 4 49](#_Toc57616102)

[ Question 5 49](#_Toc57616103)

[ Question 6 49](#_Toc57616104)

[ Question 7 50](#_Toc57616105)

[ Question 8 50](#_Toc57616106)

[Construisez un modèle statistique 51](#_Toc57616107)

[Construisez un modèle statistique 51](#_Toc57616108)

[Utilisez la fonction *loss*, ou la perte d’information 53](#_Toc57616109)

[Une histoire d'optimisation numérique 55](#_Toc57616110)

[En résumé 56](#_Toc57616111)

[Programmez votre première régression linéaire 57](#_Toc57616112)

[Installez les logiciels utilisés 57](#_Toc57616113)

[Définissez la problématique et ses données d'entraînement 57](#_Toc57616114)

[Reformulez le problème dans l'espace d’hypothèse : une droite 58](#_Toc57616115)

[Définissez la fonction *loss* 59](#_Toc57616116)

[Apprentissage : trouvez le θ optimal 60](#_Toc57616117)

[Utilisez le modèle pour effectuer des prédictions 62](#_Toc57616118)

[En résumé 62](#_Toc57616119)

[Allez plus loin : le “vrai” travail de modélisation 63](#_Toc57616120)

[**Exploitez votre jeu de données** 64](#_Toc57616121)

[**Echantillonnez les données** 64](#_Toc57616122)

[**Mettez de côté une partie des données pour tester votre modèle** 65](#_Toc57616123)

[**En résumé** 66](#_Toc57616124)

[Entraînez votre premier k-NN 67](#_Toc57616125)

[Comment fonctionne le modèle k-NN ? 67](#_Toc57616126)

[C'est un peu plus compliqué que ça… 69](#_Toc57616127)

[Utilisez k-NN sur un vrai jeu de données 70](#_Toc57616128)

[En résumé 76](#_Toc57616129)

[Entraînez-vous à entraîner un algorithme de Machine Learning ! 77](#_Toc57616130)

[À vous de jouer ! 77](#_Toc57616131)

[Votre mission 77](#_Toc57616132)

[Livrable 77](#_Toc57616133)

[Vérifiez votre travail 78](#_Toc57616134)

[Familiarisez-vous avec les limites des algorithmes 79](#_Toc57616135)

[Le théorème "No Free Lunch" 79](#_Toc57616136)

[Les problèmes insolubles 80](#_Toc57616137)

[En résumé 81](#_Toc57616138)

[Trouvez le bon compromis entre biais et variance 82](#_Toc57616139)

[Tout est affaire d'équilibre ! 82](#_Toc57616140)

[Quel est le lien avec l'erreur d'un modèle ? 84](#_Toc57616141)

[Comment trouver le bon compromis ? 85](#_Toc57616142)

[En résumé 86](#_Toc57616143)

[**Généralisez votre modèle** 87](#_Toc57616144)

[**L'overfitting et l'underfitting** 87](#_Toc57616145)

[**Comment prendre en compte ce problème ?** 89](#_Toc57616146)

[**En résumé** 90](#_Toc57616147)

[Gérez le fléau de la dimension 91](#_Toc57616148)

[Qu'est-ce que le fléau de la dimension ? 91](#_Toc57616149)

[Comment résoudre ce problème ? 93](#_Toc57616150)

[En résumé 95](#_Toc57616151)

[Appréhendez les limites du Machine Learning 96](#_Toc57616152)

[Compétences évaluées 96](#_Toc57616153)

[ Question 1 96](#_Toc57616154)

[ Question 2 96](#_Toc57616155)

[ Question 3 97](#_Toc57616156)

[ Question 4 97](#_Toc57616157)

[ Question 5 98](#_Toc57616158)

[ Question 6 98](#_Toc57616159)

[ Question 7 99](#_Toc57616160)

[ Question 8 99](#_Toc57616161)

Vous êtes intéressé par la **Data Science** et vous cherchez une porte d'entrée vers ce domaine en plein essor ? Ce cours d'initiation au Machine Learning est fait pour vous !

Le **Machine Learning** est un ensemble de techniques utilisées par les Data Scientists qui a grandement fait parler de lui ces dernières années. Car ses applications sont variées et très prometteuses !

Une fois que le Data Scientist a effectué son travail de collecte, de nettoyage et d’exploration des données, il peut passer à la partie "**modélisation"**. C’est ce processus que nous allons explorer ensemble dans ce cours d'initiation au Machine Learning.

Vous allez découvrir un ensemble de techniques puissantes permettant de créer, à partir de données, des modèles **prédictifs qui apprennent par eux-mêmes !**

Je vous propose d'aborder cela avec moi, étape par étape, en restant au plus proche des problématiques actuelles que permet de résoudre la Data Science.



Ce cours a été créé en partenariat avec l'École CentraleSupélec.

**Objectifs pédagogiques :**

* Resituer le Machine Learning au sein de la Data Science
* Identifier les possibilités du Machine Learning
* Identifier les techniques et outils du Machine Learning
* Entraîner un algorithme de régression linéaire
* Identifier les limites du Machine Learning

**Prérequis :**

Ce cours se situe au croisement des mathématiques et de l'informatique. Si vous ne possédez pas les prérequis ci-dessous, n'hésitez tout de même pas à consulter ce cours, qui vous donnera un bon aperçu de l'écosystème et des méthodes de Data science.

Pour profiter pleinement du cours, n'hésitez pas à vous rafraîchir la mémoire sur :

* [**Python pour le calcul numérique**](https://openclassrooms.com/courses/decouvrez-les-librairies-python-pour-la-data-science) ;
* **quelques notions d'algèbre** linéaire, telles que manipulation de vecteurs, multiplication de matrices, normes ;
* **quelques notions de**[**probabilités**](https://openclassrooms.com/fr/courses/4525296-maitrisez-les-bases-des-probabilites)**et**[**statistiques**](https://openclassrooms.com/fr/courses/4525306-initiez-vous-a-la-statistique-inferentielle), telles que distribution de loi de probabilité et variance.

**Découvrez le domaine de la Data Science**

La **data science** (ou science des données en français) et le **machine learning** (ou apprentissage automatique) sont deux mots très en vogue lorsque l'on parle de la **révolution Big Data**, de prédiction des comportements ou tout simplement de la transformation numérique des entreprises. Et comme pour tous les domaines innovants, il est parfois **difficile de s'y repérer**.

C'est pourquoi, avant de rentrer dans le vif du sujet, je vous propose de faire un tour rapide du domaine de la science des données et en quoi elle est devenue une source de valeur ajoutée pour les entreprises.

Qui dit nouveau métier dit aussi « anglicismes ». Eh oui, vous trouverez très souvent dans vos recherches de nombreux termes techniques non traduits, bien plus communs que leurs équivalents dans la langue de Molière. À commencer par "machine learning". Ne vous étonnez donc pas de trouver de nombreux mots anglais dans ce cours. Je vous conseille vivement de vous familiariser avec eux.

**Qu’est-ce que la data science ?**

Pour démarrer, voici une première définition de la data science :

Le premier objectif du data scientist est de produire des méthodes (automatisées, autant que possible) de tri et d'analyse de données, afin d'en extraire des informations utiles.

**Le besoin d'un data scientist est apparu pour trois raisons principales :**

* l'explosion de la quantité de données produites et collectées par les humains ;
* l'amélioration et l'accessibilité plus grande des algorithmes de traitement des données ;
* l'augmentation exponentielle des capacités de calcul des ordinateurs.

Pour reformuler, l’objectif est de **récupérer des données de plusieurs sources différentes** et d’**en extraire des informations qui vont servir l’entreprise**, notamment l’aide à la décision (“data-driven decision”). Une entreprise qui a bien intégré la data science sera capable de pondérer les intuitions humaines à l’aide des nouvelles informations suggérées par les données qu’elle possède.

Comme une mode s'est créée autour de ces métiers, un premier réflexe des entreprises est de penser que la data science constitue une sorte de baguette magique qui va pouvoir prédire beaucoup de choses de façon précise. 🔮

Mais attention à ne pas tomber dans le fantasme ! Deux composantes sont nécessaires avant de se demander si la data science peut, oui ou non, apporter de la valeur et aider à la résolution d'un problème : **des données** et **une problématique bien définie.**

**Tout commence par des données**

Cela paraît évident, **les données constituent la ressource principale pour qu'un data scientist puisse effectuer son travail correctement**. Après tout, c'est tout de même le “data” de "data science".

Donc, si dès le départ vous ne voyez aucune manière de récupérer des données liées au problème que vous cherchez à résoudre, considérez que vous ne pourrez n’être d’aucune aide sur ce problème, en tant que data scientist.

**Un autre frein** est que les données ne sont **pas toujours utilisables d’un point de vue éthique ou pour des raisons de sécurité.** Pensez à vous assurer que vous pouvez bien exploiter les données pour votre travail.

Une expression souvent utilisée pour désigner cette situation où l'on commence sans données est le “**cold start problem**”. Trouver comment travailler avec pas ou peu de données est un domaine de recherche très actif.

La valeur d'un data scientist réside aussi dans sa capacité à trouver des manières innovantes de récupérer des données auxquelles on ne penserait pas au premier abord. Faites appel à votre esprit de hacker et essayez de bien réfléchir à la façon de récupérer des données utiles, même s'il ne semble pas y en avoir à première vue. 🤓

**Une problématique bien définie**

**Il existe un spectre assez large de problématiques que l'on peut résoudre en data science**. Mais vous pouvez aussi facilement en exclure un certain nombre, surtout si vous vous trouvez dans un milieu d'entreprise, avec des contraintes de temps fortes pour produire des résultats.

**Estimer la faisabilité d'un projet est toujours compliqué en data science**, et il est normal que cela vous paraisse flou si vous débutez ! Ce n'est qu'en pratiquant, en développant votre propre expérience dans le domaine, que vous arriverez à affiner votre jugement et à redéfinir précisément des problématiques mal définies ou irréalistes, rentables ou non. À la fin de ce cours, vous aurez déjà acquis quelques réflexes à ce sujet !

Une manière de préciser cette problématique en milieu professionnel, c’est de passer par une phase de **prototypage** qui permet de tester la viabilité et la solidité du projet. Créer un prototype permet rapidement de détecter s’il y a une opportunité car les plus gros obstacles sont écartés : la récupération des données, la formulation du problème à résoudre, une estimation des coûts nécessaires, les difficultés d’implémentation hardware ou software, etc.

Pouvez-vous donner des exemples de problématiques “bien définies” ?

Bien sûr ! Voici quelques exemples que l'on peut estimer réalistes :

* Prédire les ventes d'une campagne marketing.
* Identifier si une image est déjà présente dans une banque d'image existante.
* Segmenter les utilisateurs d'un site en plusieurs groupes en fonction de leur comportement sur le site.

**Data Science et analyse de données**

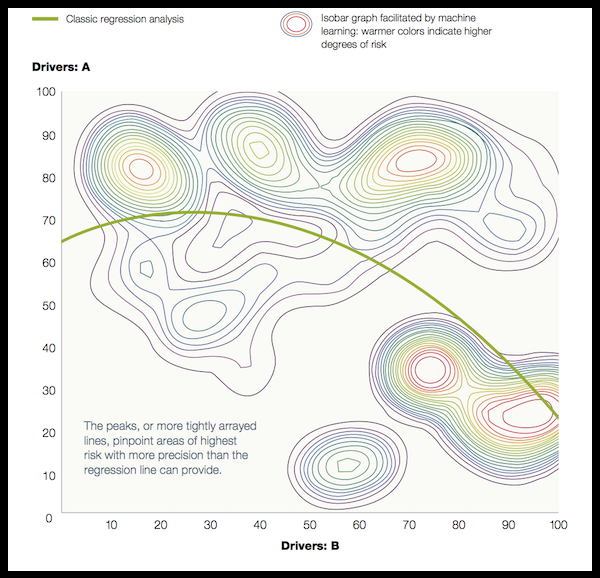
En quoi la data science se distingue-t-elle de l’analyse de données classique ? A-t-on vraiment besoin de ce nouveau domaine ?

Le machine learning se distingue dans un premier temps par l’approche utilisée pour résoudre la problématique - **la notion d’apprentissage** - que l'on va étudier dans la suite de ce cours.

**L’analyse de données (“data analysis”), elle, utilise des méthodes issues des statistiques classiques** comme les estimateurs, les analyses de corrélations ou encore la régression linéaire.

Pour le data scientist, l’objectif est d’aller plus loin dans les algorithmes d’analyse afin d'obtenir un degré supérieur d'information, notamment grâce au machine learning.

Dans l’exemple ci-dessous, on représente le risque de désabonnement de clients selon deux critères, Drivers A et Drivers B. Là où une analyse de données statistiques classique (la courbe) nous donne simplement une frontière de classification, on voit qu’un algorithme de machine learning (les isobares) peut aller beaucoup plus loin et dépasser les contraintes de linéarité pour fournir une réponse plus précise, avec une carte de risques.



Modélisation du risque de désabonnement de clients d'une entreprise de télécommunication

**En résumé**

La data science est un nouveau domaine de travail, qui augmente les capacités d’analyse classique, afin d’aider les entreprises à prendre des décisions informées. Elle s’appuie pour cela sur des données utiles et ne peut s’appliquer que dans certaines problématiques précises.

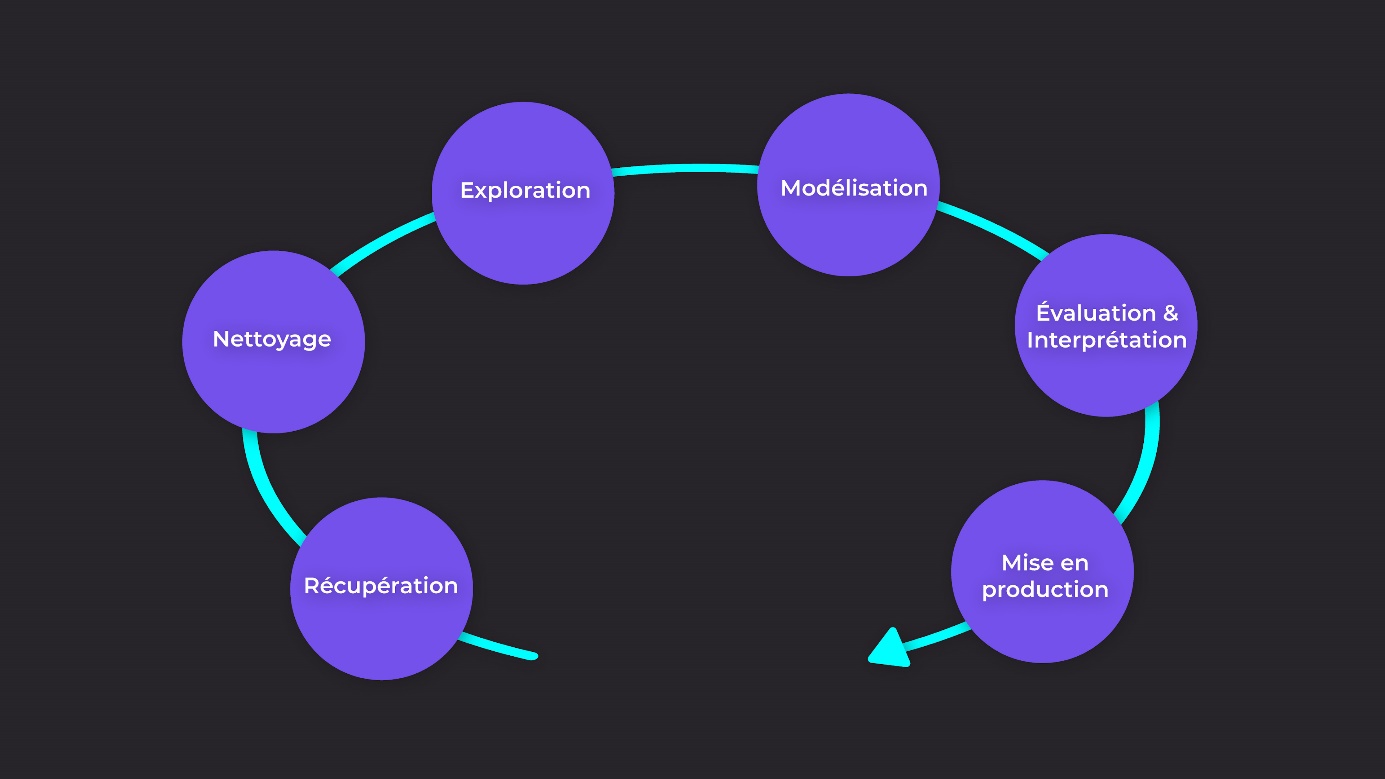
## Plongez-vous dans la peau d’un Data scientist

Le machine learning ne désigne en réalité qu’une partie du travail d’un data scientist. C'est pourquoi avant de rentrer dans le vif du sujet et de ne parler que de la partie machine learning, je vous propose de **faire un tour rapide du métier de data scientist**, afin de se situer.

Dans ce chapitre, nous allons prendre un peu de hauteur et observer en quoi consiste le **cycle habituel de travail des data scientists**, pour comprendre à quelle étape intervient le machine learning. C’est parti !

### Appréhendez le cycle de travail du data scientist

Le cycle de travail du data scientist peut se résumer par le schéma ci-dessous. Pour faire simple, on part de la réalité, on récupère les données, on les nettoie, on les explore, puis on utilise nos algorithmes pour créer de l’intelligence (artificielle) qui aide à la décision. Dans la suite, nous allons détailler ces différentes étapes et voir quels sont les différents métiers sur la chaîne de traitement de la donnée.



Cycle de travail du data scientist

### Récupérez les données

Une fois que vous êtes décidé à attaquer un problème, la première chose à faire est d'explorer toutes les pistes possibles pour récupérer les données. En effet, les données constituent l'expérience, les exemples que vous allez fournir à votre algorithme de machine learning, afin qu'il puisse apprendre et devenir plus performant.

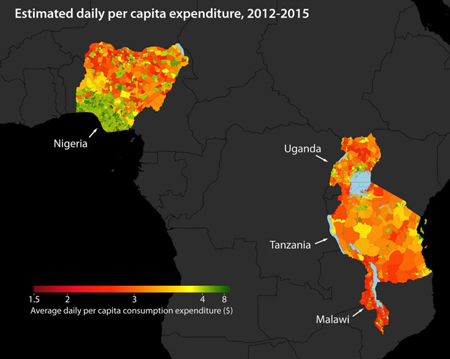
Dans la suite du cours, j’appellerai les données étudiées, destinées à alimenter un algorithme de machine learning, indifféremment **dataset** ou **jeu de données**.

**Tout doit passer au crible !** Les bases de données existantes, des données brutes alternatives (image, son, etc.), et même la création de nouveaux canaux d'acquisition de données. Essayez de trouver l'ensemble des variables qui impactent de près ou de loin le phénomène qui vous intéresse.

Vous trouverez ci-dessous quelques exemples, où les data scientists ont redoublé d'ingéniosité pour récupérer et utiliser leurs données de manière originale.

#### **Les images satellites pour évaluer le niveau de pauvreté**

Des chercheurs ont utilisé le machine learning pour pouvoir cartographier les zones de pauvreté de manière automatique, simplement à partir [d'images satellites](https://news.stanford.edu/2016/08/18/combining-satellite-data-machine-learning-to-map-poverty/) !

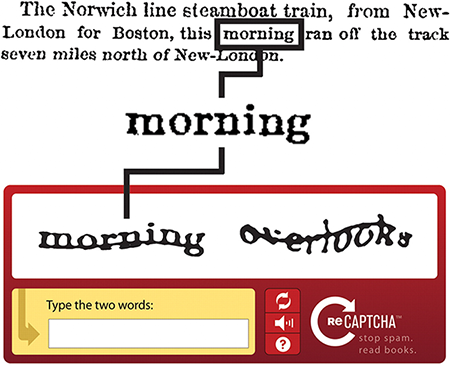


Une cartographie de l'estimation de la consommation moyenne quotidienne (crédits : Neal Jean et al.)

#### **Les CAPTCHAs pour la digitalisation automatique de livres**

Luis von Ahn, entrepreneur et chercheur, a créé un célèbre système de reCAPTCHA qui permettait à la fois aux sites web de valider que les formulaires étaient bien remplis par des humains, et qui alimentait en même temps la base de données d'un algorithme de digitalisation de livres. Grâce aux nombreux exemples renseignés directement par des humains, l'algorithme a fini par avoir suffisamment de données d'exemples pour réussir ensuite seul à retranscrire en texte des images scannées de livres, avec un taux d'erreur très faible.

Pour en savoir plus, vous pouvez consulter [**le site du projet**](https://www.cylab.cmu.edu/partners/success-stories/recaptcha.html).

Exemple de reCAPTCHA

#### **Détectez l'illettrisme par l'utilisation du smartphone**

Un chercheur norvégien a utilisé plusieurs types de données mobiles (tels que les SMS, le nombre de contacts, etc.) pour détecter les personnes illettrées dans les pays en voie de développement.

Pour en savoir plus, vous pouvez consulter [**cet article de 2016 du site MIT Technology Review**](https://www.technologyreview.com/s/601895/mobile-phone-data-reveals-literacy-rates-in-developing-countries/).

#### Croisez les différentes sources de données

Dans beaucoup de cas, **l’innovation** en data science dans une entreprise vient de l’**originalité de l’utilisation des données** et du **croisement de différentes sources de données**. Pour cela, il faut dans l’idéal posséder une politique de gestion des données dans son entreprise la plus transparente possible. Pour les données, c’est comme pour les ressources humaines : les différents départements organisés en silos communiquent moins et innovent moins par rapport à un environnement ou la transversalité est favorisée. Alors essayez d'éviter les **data-silos** !

### Nettoyez les données

Une fois les données trouvées, il faut passer à l'étape de nettoyage. Pour ne rien vous cacher, ce n'est pas l'étape la plus agréable du travail, mais ça ne la rend pas moins indispensable.

Nettoyer les données, c'est s'assurer qu'elles sont **consistantes**, sans **valeurs aberrantes** ni **manquantes**.

Pour aller plus loin dans cette étape, consultez le cours [**Décrivez et nettoyez votre jeu de données**](https://openclassrooms.com/fr/courses/4525266-decrivez-et-nettoyez-votre-jeu-de-donnees).

Une autre étape nécessaire, en général, est l’aggrégation de ces données dans un **data lake**. Nettoyer les données signifie donc qu’elles sont toutes sous le même format, accessibles au même endroit et au bon moment.

Lorsque ces questions deviennent complexes, il faut faire appel au **data architect** qui, lui, possède une maîtrise technique pour réaliser ces différentes tâches. Ces ingénieurs des Big Datas sont responsables de la création et de l'administration de tous les systèmes techniques qui vont permettre la bonne exploitation des données.

Si vous souhaitez en savoir plus sur cet aspect technique de la data science, OpenClassrooms propose un parcours de formation de [**Data Architect**](https://openclassrooms.com/fr/paths/64-data-architect).

L'important, c'est de bien préparer le terrain pour les étapes suivantes. Ces étapes seront grandement simplifiées si ce travail fastidieux est bien effectué en amont.

### Explorez les données

Les données bien propres peuvent maintenant commencer à être explorées. Cette étape vous permet de **mieux comprendre les différents comportements** et de **bien saisir le phénomène sous-jacent**.

C'est vraiment une étape à ne pas négliger. Les meilleurs data scientists ne sont pas ceux qui connaissent les algorithmes les plus complexes, mais **ceux qui ont une très bonne connaissance des données**et ont préparé le terrain avec soin en amont.

À la fin de l’exploration, vous devrez être en mesure de :

* Proposer plusieurs hypothèses sur les causes sous-jacentes à la génération du dataset : "suite à l'exploration, il y a clairement une relation entre X et Y".
* Proposer plusieurs pistes de modélisation statistique des données, qui vont permettre de résoudre la problématique de départ considérée.
* Proposer si nécessaire de nouvelles sources de données qui aideraient à mieux comprendre le phénomène.

C'est dans les phases de nettoyage et d'exploration des données que les data scientists passent le plus clair de leur temps.

Lorsque l'on a simplement besoin de comprendre ses données et les explorer, on peut faire appel à un **data analyst**. Ou bien un data analyst peut effectuer des études préliminaires avant de laisser le travail de modélisation au data scientist. Si vous souhaitez vous former à ce métier, OpenClassrooms propose un parcours de formation de [**Data Analyst**](https://openclassrooms.com/fr/paths/65-data-analyst).

### Modélisez les données à l'aide du machine learning

Nous pouvons enfin rentrer dans la partie la plus intéressante du métier, c’est-à-dire la création du modèle statistique associé aux données qui nous intéressent ! C'est ce qu'on appelle le **machine learning** (ou apprentissage automatique).

Mais ça veut dire quoi “modélisation statistique des données” ?

En machine learning, et en data science plus généralement, l'objectif est de trouver un modèle (stochastique ou déterministe) 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**.

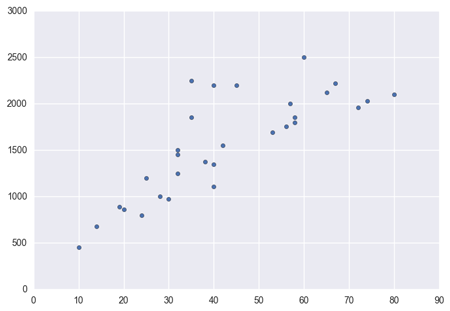
Si vous n'avez jamais entendu parler de **variable aléatoire** ou de **distribution de probabilité**, vous allez avoir des difficultés à suivre le reste du cours. Je vous conseille de suivre en amont un cours d'introduction aux probabilités et aux statistiques. Vous en trouverez dans le parcours [**Data Analyst**](https://openclassrooms.com/fr/paths/65-data-analyst) sur OpenClassrooms.

Le mieux pour expliquer ce que ça signifie est de prendre un petit exemple simple. Imaginez que vous voulez savoir si vous payez trop cher votre loyer. Vous avez récupéré sur un site de location une trentaine de prix des locations disponibles, ainsi que la surface associée :

| **loyer mensuel (en €)** | **surface (en**m2m2**)** |
| --- | --- |
| 1500 | 32 |
| 2120 | 65 |
| 2500 | 60 |
| ... | ... |
|  |  |

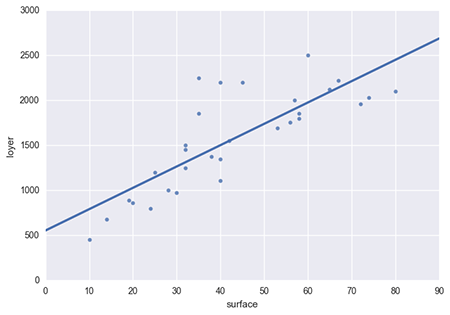
Bien sûr, en réalité d’autres paramètres seraient probablement à prendre en compte (parties communes, voisinage, évolution des loyers au cours du temps, etc). Le but est ici d’appréhender un modèle simplifié afin de comprendre rapidement ce que veut dire "modéliser un phénomène".

Si l'on affiche maintenant ces différents points sur un graphe qui représente le montant du loyer en fonction de la surface, on obtient le graphique suivant :



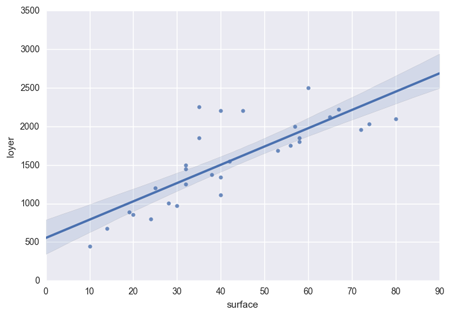
Loyer mensuel en fonction de la surface du logement

Comme on pouvait s’y attendre, on remarque une augmentation relativement **linéaire** du loyer par rapport à la surface de l’appartement. Une première **modélisation** simple du phénomène (le prix du loyer) serait donc simplement de considérer la droite la plus “proche” de l’ensemble des points.



La droite de régression correspondant à la modélisation du nuage de points

La droite représente donc notre **modèle** du phénomène, auquel nous pouvons ajouter l'intervalle de confiance dans lequel on pense que se trouve la droite.



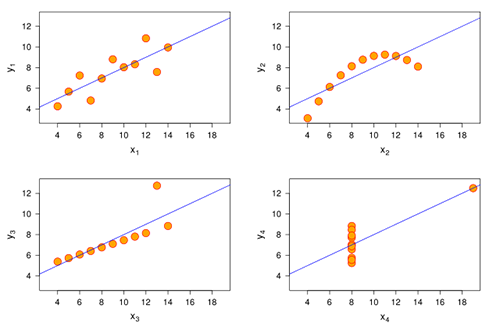
L'intervalle de confiance (à 90 %)

Pour résumer, le travail de modélisation consiste à trouver le bon modèle statistique (ici la droite et son intervalle de confiance) qui colle le mieux aux données d'exemple. Le machine learning en particulier intervient pour trouver ce modèle de manière automatisée.

### Évaluez et interprétez les résultats

Une fois un premier travail de modélisation effectué, la suite de l’étude s’effectue par **l’évaluation de la qualité de notre modèle**, c’est-à-dire sa capacité à représenter avec exactitude notre phénomène, ou a minima sa capacité à résoudre notre problématique.

Une représentation connue qui souligne la nécessité de l'évaluation est le **quartet d'Anscombe**. Il permet de montrer visuellement que pour 4 jeux de données très différents, on obtient la même droite de régression.



Le quartet d'Anscombe illustre bien le fait que si l'on n'examine pas assez les données, et qu'on ne mesure pas de la bonne manière l'erreur de son modèle, on peut facilement arriver à des aberrations de modélisation.

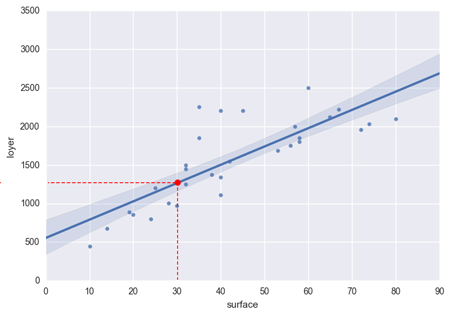
Il y a parfois clairement un problème dans notre modèle, qui ne capture pas l'essence du phénomène. Pour nous aider à évaluer les résultats, mesurer l’erreur de notre modélisation vis-à-vis de nos données d’exemple constitue un premier indicateur de qualité. Dans les cas ci-dessus, il faudrait clairement changer le modèle d’une droite que nous avions décidé au départ !

C’est donc un jeu d’allers-retours entre modélisation et évaluation qui s’effectue pour obtenir les performances les plus satisfaisantes possibles. Il est même possible, dans certains cas, de remettre en question certaines hypothèses de départ et de repartir dans une phase d’exploration pour mieux comprendre les données.

### Déployez le modèle en production

Une fois qu’on est satisfait de la qualité des performances de notre modèle, on va pouvoir passer à l’étape suivante, qui est le rendu de nos résultats et le potentiel déploiement du modèle en production. Imaginez que vous trouvez que votre modèle d’évaluation des loyers est très performant et mériterait d’être partagé à plus de monde. Vous décidez donc de le déployer sur un serveur où tout le monde pourra obtenir une estimation de son loyer selon votre modèle, et ainsi déterminer s'il paie plus ou moins que les prix du marché ! Cela l'aidera sûrement dans sa décision de déménager. 😬

Comment cela fonctionne-t-il en pratique ? C’est assez simple, il vous suffit de récupérer les paramètres de votre modèle et de faire passer la surface de l'appartement en entrée du modèle, afin d’obtenir le loyer associé en sortie, en suivant la droite.



Imaginez qu'un appartement a une surface de 30 mètres carrés (point en rouge), une estimation légitime du loyer se situerait aux alentours de 1300 euros selon notre modèle.

Pour des modèles plus complexes, le fonctionnement reste le même. Si vous voulez appliquer votre travail à de nouvelles données, il vous suffit de passer les nouvelles entrées dans votre modèle (qui est en principe un ensemble de transformation des valeurs d’entrées) afin d’obtenir une sortie.

Là encore, si ce passage en production est complexe, que ce soit en termes d’échelle, de contrainte de rapidité de calcul ou de sortie de résultats, il faut faire appel à un data architect qui sera responsable d’industrialiser le prototype que vous lui fournirez.

### En résumé

**La data science est un nouveau domaine de travail** qui augmente les capacités d’analyse classique, afin d’aider les entreprises à prendre des décisions plus informées. Elle s’appuie pour cela sur des données utiles et ne peut s’appliquer que dans certaines problématiques précises qui gagnent à utiliser ce type de méthodes.

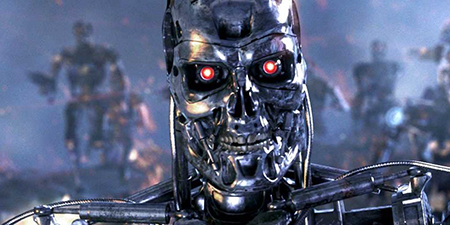
Au sein du cycle de travail du data scientist, le **machine learning** désigne l'ensemble des méthodes de **modélisation statistique à partir des données.**

## Identifez les différentes étapes de modélisation

Dans le chapitre précédent, vous avez découvert le cycle de travail du data scientist. Nous allons maintenant parler spécifiquement de la partie qui nous intéresse dans ce cours, c'est-à-dire la **modélisation**. Et pour modéliser les données, un vrai data scientist utilise son arme secrète de ninja : le fameux **machine learning**.

### Démystifions le machine learning

Lorsqu'on entend parler de machine learning – ou plus généralement de l'intelligence artificielle, dont le machine learning est un sous-domaine – on pense généralement à ça :



Terminator... Ce n'est pas pour tout de suite !

Mais les experts du domaine sont formels : malgré toutes les inquiétudes évoquées dans les médias, le machine learning, et de manière plus générale l'intelligence artificielle, ne constituent pas une réelle menace. En l'état actuel, on est vraiment loin d'avoir atteint un niveau d'intelligence suffisant chez les machines pour avoir de quoi s'inquiéter.

Dans ce cours, nous n'allons pas nous attarder sur l'aspect « éthique » du machine learning.

Pour en savoir plus sur le sujet, je vous conseille [**la rubrique dédiée sur le site future of life**](http://futureoflife.org/background/benefits-risks-of-artificial-intelligence/).

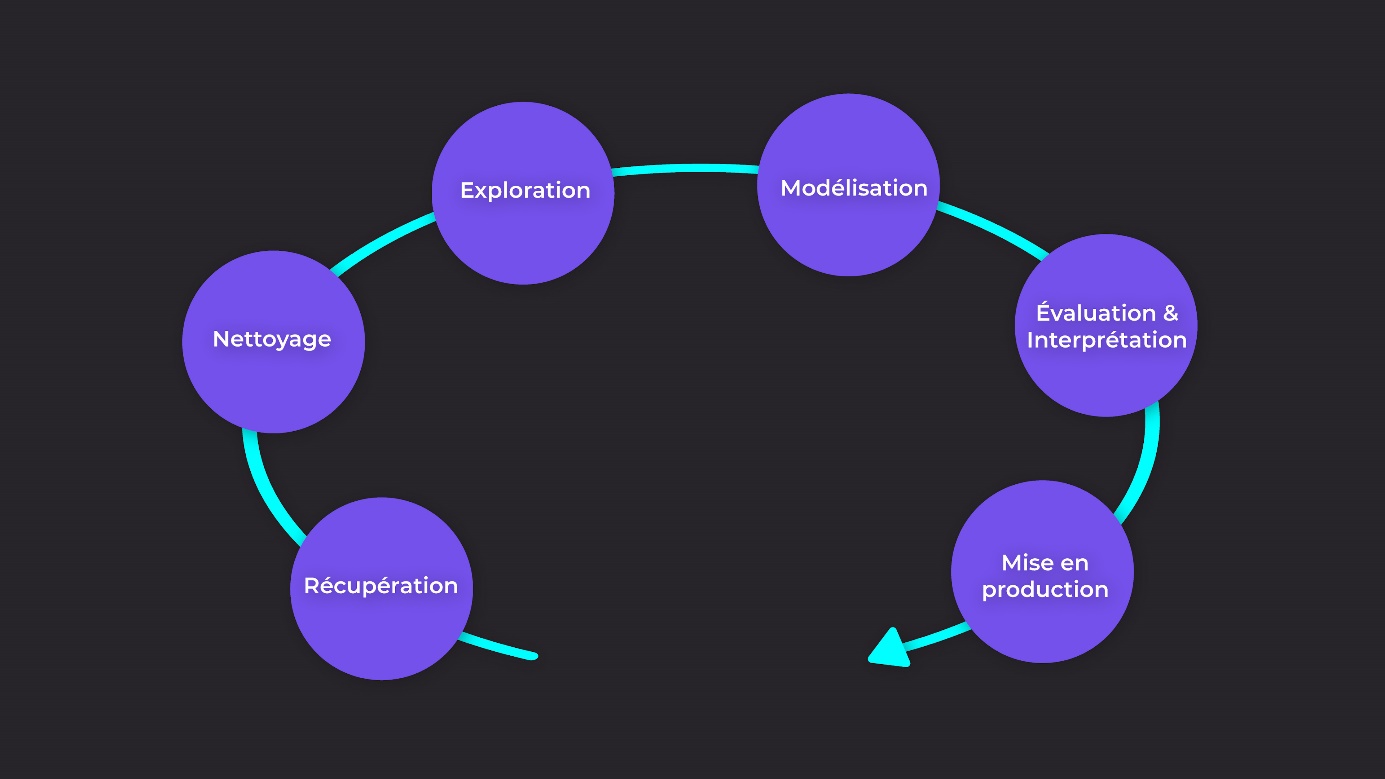
Une expérience récente effectuée par le MIT sur les voitures autonomes aborde la question de la moralité, faisant appel à l'intelligence collective pour entraîner la morale des machines : [**moralmachine.mit.edu**](http://moralmachine.mit.edu/).

Le **machine learning** constitue, comme on l'a vu dans le chapitre précédent, une manière de modéliser des phénomènes, dans le but de prendre des décisions stratégiques.

Même si c'est un outil très puissant quand il est bien utilisé, le machine learning n'est pas une baguette magique. Il est important d'être clair sur ce point avec les collaborateurs avec qui vous travaillez, afin de tempérer leurs attentes sur les résultats que vous allez obtenir.

### Avant la modélisation

Comme nous l'avons vu dans la partie précédente, les étapes de travail d'un data scientist sont les suivantes :



Cycle de travail du data scientist

J'insiste, mais il est important de noter que le travail du data scientist comprend bien toutes les étapes, de la récupération au déploiement. Une **majorité** du travail quotidien du data scientist s'effectue même plus sur la récupération, le nettoyage et l'exploration de données que sur la modélisation en elle-même. Il est donc sous-entendu dans la suite du cours que ces étapes auront déjà été effectuées sur les données utilisées.

#### Un exemple concret

Un data scientist qui travaille dans le web a souvent la possibilité de récupérer les analytics du site pour lequel il travaille (des informations sur les visites et le comportement des utilisateurs sur le site). À partir de ces données brutes, il va **sélectionner**, **nettoyer** et **transformer** les données pertinentes. Par exemple, au lieu de récupérer directement le nombre de visites journalières, il va plutôt récupérer leurs variations d'un jour sur l'autre.

Il va ensuite explorer ces données pré-traitées, c'est-à-dire visualiser les différentes variables, essayer de comprendre les comportements (les valeurs extrêmes ou aberrantes, les corrélations, etc.). Une fois qu'il a une bonne idée de ce à quoi il a affaire, il peut définir une problématique plus précise à laquelle répondre.

Pour amener de la valeur ajoutée à partir de ces données, le data scientist pourrait effectuer un classement des utilisateurs les plus susceptibles de revenir sur le site, ce qui permettrait ensuite de cibler les utilisateurs les plus engagés par des campagnes marketing ciblées.

Une fois l'exploration faite et la question de recherche définie, le data scientist peut passer à l'étape de modélisation.

### L'étape de modélisation

Imaginez que vous êtes un data scientist. Vous êtes maintenant à l'aise avec l'ensemble des données récupérées pour vos analyses. Vous avez une connaissance des objectifs principaux de l'entreprise, ce qui vous a aidé à synthétiser les différentes variables qui interviennent, ainsi de visualiser les différents comportements et corrélations présents au sein de ces données.

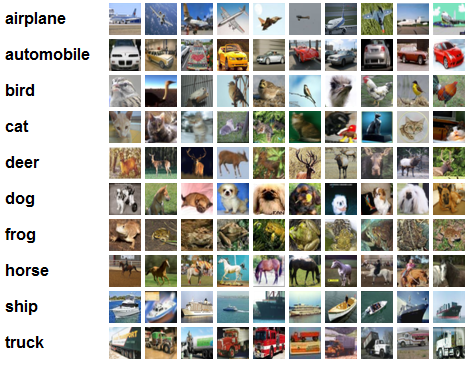
Le problème de machine learning constitue l'étape suivante et permet à un ordinateur de modéliser les données qui lui sont fournies.

"**Modéliser**" signifie dans ce cas représenter le comportement d'un phénomène, afin de pouvoir aider à la résolution d'un problème concret de l'entreprise.

Mais pourquoi parle-t-on d'**apprentissage** ?

En machine learning, l'algorithme se construit une "représentation interne" afin de pouvoir effectuer la tâche qui lui est demandée (prédiction, identification, etc.). Pour cela, il va d'abord falloir lui entrer un jeu de données d'exemples afin qu'il puisse **s'entraîner** et s'améliorer, d'où le mot **apprentissage**. Ce jeu de données s'appelle le **training set**. On peut appeler une entrée dans le jeu de données une **instance** ou une **observation.**

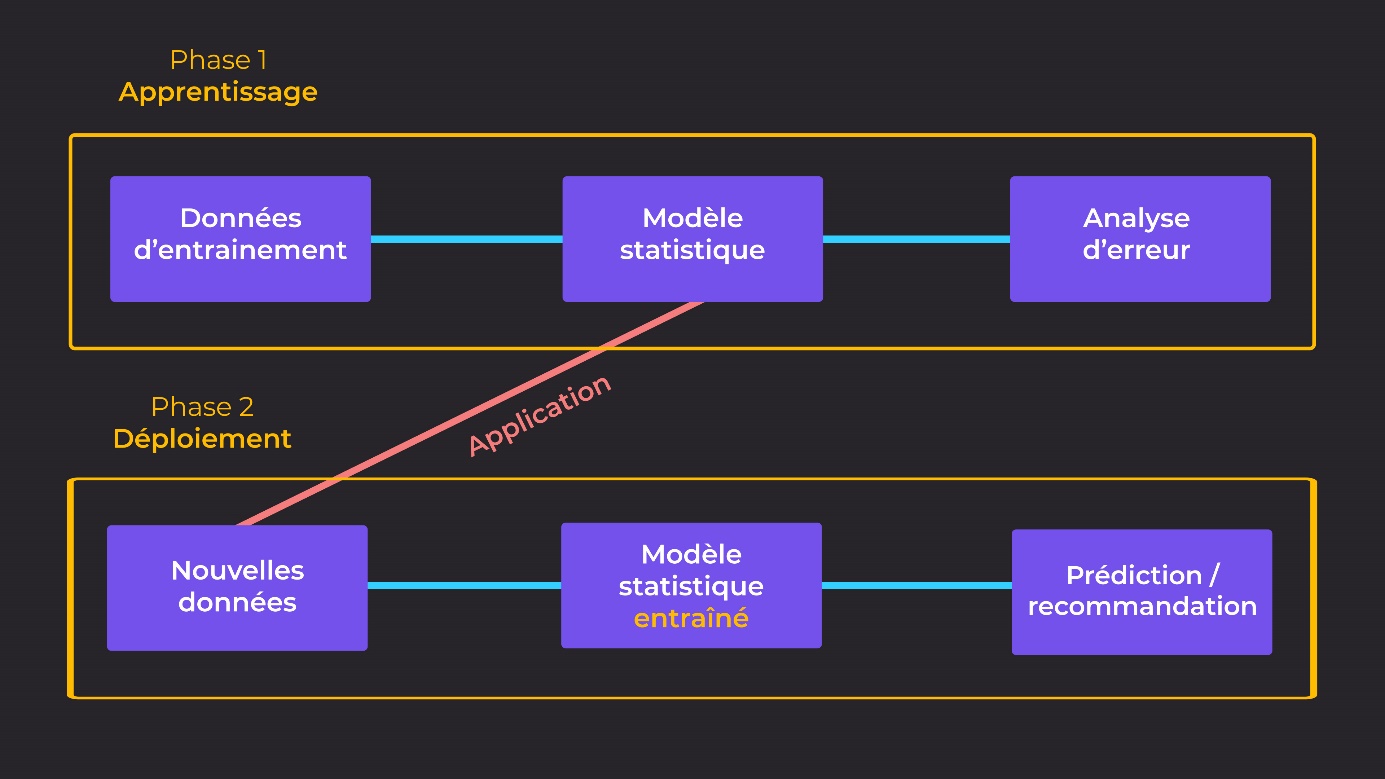
Voici un exemple de jeu de données images classique qui permet d'entraîner un modèle de machine learning. Chaque image constitue une observation du data set.

Source : karpathy.github.io

Vous allez donc être responsable dans une première phase du choix et de l'entraînement de **l'algorithme d'apprentissage** du modèle, mais le traitement de la tâche spécifique sera appris à partir du **training set**et ensuite effectué par l'algorithme lui-même dans une **seconde phase**.

Dans le schéma ci-dessous, vous pouvez voir les différentes étapes qui interviennent dans l'utilisation d'un algorithme de machine learning. Vous serez responsable de la phase 1et vous pourrez ensuite simplement passer les données dans le modèle durant la phase 2.

 Évidemment, ce n'est pas aussi séquentiel (on fait par exemple des allers-retours entre apprentissage et analyse d'erreur).



Un détail des deux phases du process de machine learning

En résumé, le travail du data scientist en machine learning consiste à sélectionner les bonnes **données** test, choisir et entraîner le bon **algorithme**, en vérifiant grâce à l'**analyse d'erreurs** que le modèle devient de plus en plus performant et robuste. Si les performances s'améliorent lorsqu'on lui fournit les données d'entraînement, on dit alors que la machine "apprend".

Une fois le modèle correctement paramétré sur les données d'entraînement, le data scientist peut ensuite le déployer afin qu'il traite de nouvelles données, pour accomplir **la tâche spécifique** poursuivie (prédiction, recommandation, décision...).

Comme on peut le voir sur le schéma ci-dessus, un problème d'apprentissage machine comporte ainsi différents éléments spécifiques :

* **les données** (les données d'entraînement mais aussi les nouvelles données) ;
* **la tâche spécifique** à accomplir (prédire, recommander, décider quelque chose, etc.) ;
* **l'algorithme d'apprentissage** en lui-même ;
* **l'analyse d'erreur** (ou **mesure des performances** du modèle).

Laissez-moi maintenant détailler un peu plus chacun de ces éléments dans les prochaines sections !

### Les données

Nous l'avons déjà dit, les **données** constituent littéralement le nerf de la guerre de la **data** science. 😉

Plus vous aurez une bonne compréhension de vos données, plus vous serez à même de pouvoir les utiliser à bon escient lors de la phase d'entraînement de votre modèle statistique. Nous allons rapidement donner une vue d'ensemble du type de données habituelles rencontrées en machine learning.

#### Les bases de données

Cela coule de source, mais les bases de données constituent la source principale de récupération de données par les data scientists. Ces bases de données peuvent comprendre différents types d'information, une bonne partie généralement spécifique à l'activité de l'entreprise.

À titre d'exemple et de manière non exhaustive :

* les logs d'un serveur web ;
* le catalogue produits d'un site de e-commerce ;
* les transactions bancaires ;
* les comportements des utilisateurs d'un site.

#### Les données brutes

D'autres données brutes, souvent plus complexes et nécessitant des pré-traitements spécifiques pour les rendre manipulables par les algorithmes, peuvent servir de sources pour un problème de modélisation.

Le **deep learning**, dont vous avez peut-être déjà entendu parler, regroupe les algorithmes et modèles assez gros et complexes pour pouvoir traiter les données brutes directement, sans pré-traitement. Pour les autres modèles, on devra souvent réduire la complexité des données par des méthodes spécifiques à chaque type de donnée brute.

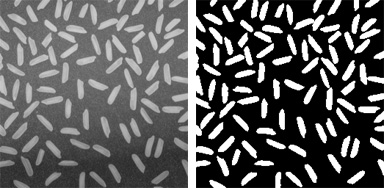
##### Le texte

Le texte libre, rédigé en langage naturel (humain), est ainsi une autre source de données principale pour le travail de data scientist. Cela comporte tous les types de texte auxquels on peut penser naturellement (articles, livres, messages, etc), mais aussi d'autres types de textes tels que du code HTML ou encore des séquences d'ADN.

Le traitement du texte (appelé NLP comme **N**atural **L**anguage **P**rocessing) constitue un domaine de recherche à part entière.

##### Les images (et vidéos)

Les images sont aussi une des sources de captation de l'environnement souvent utile sur des problématiques d'entreprise. Beaucoup d'entreprises ont des banques d'images à traiter pour les classer par type ou autre.



Un exemple de pré-traitement d'image appelé seuillage, qui permet de simplifier ensuite l'apprentissage d'un modèle statistique (crédits : The MathWorks, Inc.)

Le traitement des images et vidéos constitue un domaine de recherche à part entière (computer vision en anglais).

##### IoT (Internet des objets)

Les objets connectés sont une autre source de données brutes, qui récupèrent un grand nombre de données grâce à leurs capteurs. Un bon exemple est l'entreprise [Nest](https://nest.com/), qui a utilisé la data science pour créer un thermostat intelligent qui optimise la consommation d'électricité en surveillant à la fois la température, la présence des habitants, etc.



Un exemple d'objet connecté : le thermostat intelligent de l'entreprise Nest (crédits : Nest)

### La tâche à accomplir

La tâche spécifique à accomplir correspond au problème qu'on cherche à résoudre grâce à la modélisation du phénomène. On peut distinguer un certain nombre de cas qui reviennent souvent dans un environnement business, tels que les **recommandations** de produits par exemple. Je vous ai aussi déjà cité l'**identification** de transactions frauduleuses, la **prédiction** de l'impact d'une campagne marketing sur le taux de conversion ou la prédiction du prix optimal d'un produit pour maximiser le nombre de ventes.

Chaque tâche se traduira différemment et nécessitera, bien sûr, le choix d'algorithmes différents. Nous examinerons dans le dernier chapitre de cette partie quelques problématiques courantes et comment elles se traduisent en problèmes de machine learning.

### L'algorithme d'apprentissage

L'algorithme d'apprentissage constitue la méthode avec laquelle le modèle statistique va se paramétrer à partir des données d'exemple. Il existe de nombreux algorithmes différents ! On choisira un type d'algorithme particulier en fonction du type de tâche que l'on souhaite accomplir et du type de données dont on dispose. En gros, quelle est l'entrée de l'algorithme et quelle est la sortie.

Quelques exemples d'algorithmes de machine learning, dont vous avez peut-être déjà entendu parler :

* la régression linéaire ;
* K-nn ;
* les Support Vector Machine (SVM) ;
* les réseaux de neurones ;
* les random forests.

### La mesure des performances

Mesurer les performances fait partie intégrante du travail de modélisation. Il faut en général déterminer une mesure principale, souvent spécifique à la tâche à accomplir. Le choix de cette métrique est très important ! Vous allez comprendre pourquoi avec un petit exemple simple.

#### Exemple

Imaginez que vous voulez créer un algorithme de détection de fraudes bancaires. Vous voulez mesurer à quel point votre programme est performant. Une manière de faire serait de mesurer la proportion totale de transactions détectées comme fraude. Cependant, on compte ici les transactions qui ne sont pas des fraudes et qui ont quand même été notées comme en étant (appelé "faux positifs"). Donc, avec ce genre de métrique, on n'est pas exigeant sur ce type d'erreur produit par notre algorithme. Il faut peut-être utiliser une autre métrique plus pertinente. Par exemple, préciser la proportion de "vraies fraudes" détectées par rapport au total de transactions détectées comme frauduleuses.

On voit donc qu'il faut bien réfléchir à une métrique plus pertinente pour être sûr de mesurer correctement la qualité de son algorithme.

### En résumé

Le machine learning est l'apprentissage d'un modèle statistique par la machine grâce à des données d'entraînement. Un problème de machine learning comporte plusieurs éléments spécifiques :

* des données ;
* une tâche à accomplir ;
* un algorithme d'apprentissage ;
* une mesure des performances.

En une phrase, un ordinateur **apprend** à partir de **données** pour résoudre une **tâche** en faisant attention à **mesurer les performances.**S'il **améliore** les performances sur cette tâche lorsqu'on lui fournit les données d'entraînement, on dit alors qu'il **apprend.**

# Identifiez les possibilités du Machine Learning

Bravo ! Vous avez réussi cet exercice !

### Compétences évaluées

* Identifiez les possibilités du Machine Learning
* Resituez le Machine Learning au sein de la Data Science

### Question 1

**Le machine learning se distingue de l’analyse de données classiques, car…**

* + 

c’est un domaine bien plus difficile à comprendre.

* + 

le machine learning nécessite beaucoup plus de données pour résoudre un problème.

* + 

le machine learning peut représenter plus efficacement la complexité des phénomènes étudiés.

*En effet, le machine learning peut traiter des données dynamiques et non-structurées comparativement à l’analyse statistique classique. Ceci lui permet de représenter efficacement la complexité des phénomènes étudiés.*

### Question 2

**Parmi les objectifs suivants, quels sont ceux qui vous semblent réalistes pour un data scientist ?**

*Attention, plusieurs réponses sont possibles.*

* + 

Prédire l'horaire optimal pour diffuser du contenu sur les réseaux sociaux à partir des données de visite d'utilisateurs.

* + 

Dire si une phrase est humoristique ou pas.

* + 

Catégoriser automatiquement une banque d'images.

* + 

Déterminer le segment de marché d'un utilisateur d'un site à partir de son comportement d'achat.

*La notion d'humour est subjective et difficile à caractériser. On ne s'aventurera pas sur ce genre de terrain si l’on a des contraintes de temps ou de rentabilité.*

### Question 3

**Un data architect...**

* + 

est quasiment l’équivalent d’un data scientist

* + 

est quasiment l’équivalent d’un data analyst

* + 

s’occupe d’une partie spécifique du traitement des données

*Le data architect est la personne qui s’occupe de mettre en place l’architecture software et hardware nécessaire au bon transit et stockage des données. Bien que ses tâches recoupent celles du data scientist, ils ne remplissent pas la même fonction au sein d’un projet.*

### Question 4

**Quelle est l'étape de travail pour laquelle le data scientist passe en général le plus de temps ?**

* + 

La récupération des données

* + 

L'exploration des données

* + 

La modélisation du phénomène

* + 

La mesure des résultats

*La qualité des variables (features) que vous donnez à l'algorithme d'apprentissage influence directement ses performances. Il faut non seulement qu'elles soient propres et représentatives du phénomène, mais aussi qu'elles soient utiles et pertinentes à la résolution de la tâche à accomplir. Ces critères sont essentiels et sont la raison pour laquelle on passe énormément de temps dans l'exploration.*

### Question 5

**Quels éléments essentiels caractérisent un problème de machine learning ?**

*Attention, plusieurs réponses sont possibles.*

* + 

Une tâche précise à accomplir

* + 

Des données relatives au phénomène évoqué

* + 

Une mesure de la qualité des données

* + 

Une mesure de performance du modèle

*La mesure de la qualité des données ne fait pas partie du domaine du machine learning, mais plutôt de la phase de préparation et pré-traitement de données du cycle de travail du Data Scientist.*

### Question 6

**Complétez la phrase suivante :**

**"Le machine learning est un ensemble de méthodes qui permettent aux ordinateurs d'apprendre à traiter des tâches de manière automatique…**

* + 

supervisée par des humains."

* + 

non supervisée par des humains."

*Le traitement de la tâche est effectuée automatiquement par l'algorithme sans supervision.*

### Question 7

**Qu'est-ce que la notion d'apprentissage pour un ordinateur ?**

* + 

Se baser sur des règles pré-construites pour effectuer des prédictions.

* + 

Améliorer ses performances sur une tâche donnée à partir d'exemples.

*La notion d'apprentissage se définit comme le fait d'améliorer ses performances sur une tâche à partir de données d'exemple.*

### Question 8

**Un problème d’apprentissage supervisé, signifie que...**

* + 

les données d’entraînement sont totalement annotées de la sortie désirée.

* + 

les données d’entraînement doivent être au moins en partie annotées.

* + 

les données d’entraînement ne sont pas annotées du tout.

*Les données d’entraînement sont annotées en apprentissage supervisé. Quand les données sont en partie annotées c’est du semi-supervisé. Si les données d’entraînement ne sont pas du tout annotées, c’est une forme d’apprentissage non supervisé.*

## Transformez des besoins métiers en problèmes de Machine Learning

Pour débuter cette deuxième partie, nous allons faire le pont entre les différentes catégories de problèmes de machine learning que vous venez de découvrir et les différentes problématiques métier auxquelles sont souvent confrontées les entreprises.

Faire ce lien doit devenir une seconde nature pour vous ! 💪

Voyons ensemble quelques exemples concrets.

### Affectez un score à un client

Le **scoring** est une technique répandue dans le domaine du marketing. Le score obtenu traduit généralement la probabilité qu'un individu réponde à une sollicitation marketing ou appartienne à la cible recherchée (par exemple, la probabilité qu'un prospect devienne un client payant, ou qu'un client fasse défaut – c'est-à-dire ne rembourse pas un crédit dans le cas d'une banque ou d'une assurance).

Le scoring est le plus souvent associé aux méthodes de régression, supervisée ou non supervisée selon les données d'entraînement.

Autrefois à la charge de statisticiens, c'est maintenant souvent au data scientist qu’il incombe la charge de noter les clients. Ce type de scoring s'effectue en général grâce à un certain nombre de caractéristiques qui définissent le client (âge, genre, ville, etc.).

Pourquoi n'est-ce pas resté à la charge des statisticiens ? Qu'est-ce que la data science peut faire de plus ?

Une phrase résume très bien la distinction entre statisticien et data scientist :

Data Scientist : Une personne qui est meilleure en stat qu'un développeur, et meilleure en programmation qu'un statisticien.

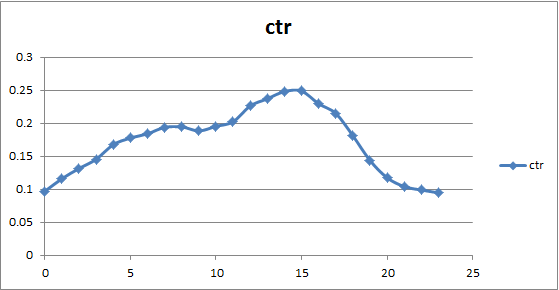
Disons que le débat est toujours ouvert mais que, pour faire simple, ce qui peut distinguer un data scientist d'un statisticien c'est sa capacité à mieux écrire du code et déployer des algorithmes en production, et à l'inverse son attachement moins strict à la rigueur statistique sous-jacente aux algorithmes.

### Prédisez la rentabilité d’une campagne marketing

Les entreprises ont souvent besoin d’évaluer le ROI (retour sur investissement) d’une campagne – ou simplement d'en comparer plusieurs –  avant de la lancer, afin d’estimer si le jeu en vaut la chandelle. C’est aussi une méthode utilisée par les régies publicitaires afin de pouvoir ajuster leurs métriques et prix en temps réel (tels que le CPC ou CPM).

**CPC**(coût par clic) :le coût d'un clic sur une publicité

**CPM** (coût pour mille) : le coût pour mille impressions d'une publicité



Prédiction du CTR (% de visiteurs qui cliquent un lien) selon l'heure de la journée. Source : Kaggle

Ce type de prédiction fait aussi appel à des méthodes de **régression**, puisqu'on essaie là encore de prédire une valeur numérique.

### Identifiez les événements rares

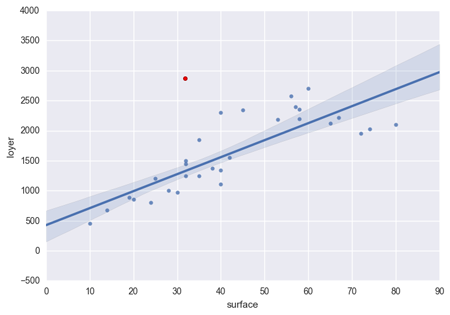
Un premier type de catégorisation automatique est l’identification d’événements appelés événements rares : les spams d’e-mail, les transactions frauduleuses, les textes injurieux, etc.

On peut traiter ce problème comme un problème de **classification supervisée** en annotant les données d'entraînement ("événement rare" et "événement normal" par exemple).

Dans la littérature, ce genre d'événement/anomalie est appelé **outlier**. Nous allons les désigner ainsi dorénavant. 🙃

Mais en fait, on peut aussi effectuer une première modélisation (supervisée ou non supervisée) qui va permettre de modéliser le comportement habituel. Ensuite, on va simplement fixer un critère de **distance** qui permet de déterminer si une entrée est trop éloignée de la modélisation, c'est-à-dire du comportement attendu. La difficulté réside ensuite dans le choix du bon critère de distance spécifique au problème traité.

Sur le graphe ci-dessous, on peut dire que le point rouge est trop éloigné de la droite de modélisation et qu'il peut donc être considéré comme un **outlier.**

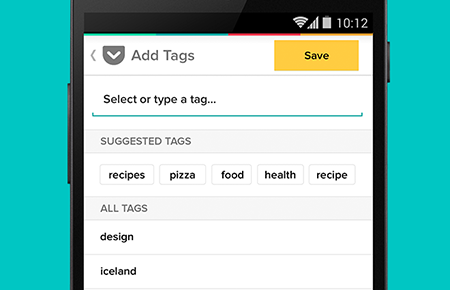


Comment détecter des outliers comme le point rouge ?

### Affectez une catégorie à un produit

Lorsque le catalogue de produits d'un distributeur grandit, la problématique de catalogage se fait ressentir, car elle demande un travail minutieux et donc un temps homme conséquent. Avec les bonnes entrées d’entraînement, l’automatisation de cette tâche peut se faire avec des algorithmes de **classification**.

En effet, nous sommes typiquement ici dans une problématique de classification supervisée : les produits déjà classés peuvent faire office de données d’entraînement et les caractéristiques de produits, voire même directement les photos, peuvent servir de variable d'entrée pour notre algorithme de classification.



Des applications comme Pocket utilisent la classification automatique de nouveaux liens entrants afin de proposer à l'utilisateur des suggestions de tags.  Source : getpocket.com

### Segmentez les visiteurs d’un site

À partir d’une étude préalable ou bien d'une connaissance qualitative de la clientèle d’une entreprise, vous souhaitez catégoriser automatiquement vos clients et les assigner à différents segments (plus ou moins susceptibles de réaliser un achat par exemple).

En pratique, ce genre de segmentation automatique est très utile pour une qualification des visiteurs d’un site, afin de leur recommander les produits appropriés notamment.

On peut choisir :

* d'effectuer une classification supervisée en annotant manuellement des segments qui nous intéressent sur une base client d'entraînement ;
* d'essayer de prédire à l'aide d'une régression la susceptibilité de conversion d'un client et ensuite effectuer une segmentation sur ce critère (< 1 % vs > 1 % susceptibles de convertir par exemple) ;
* d'effectuer une classification non supervisée afin de détecter de nouveaux groupes d'intérêts qui peuvent être targetés individuellement ;
* etc.

La difficulté à utiliser ces algorithmes, notamment concernant les **événements rares**, est de rassembler assez de données d’entraînement, contenant de plus un maximum de **variété**. À vous de jouer avec les données que vous avez et surtout à vous de bien valider les performances du modèle pour vous rendre compte de sa qualité avant de le déployer en production !

### Recommandez un produit à un client

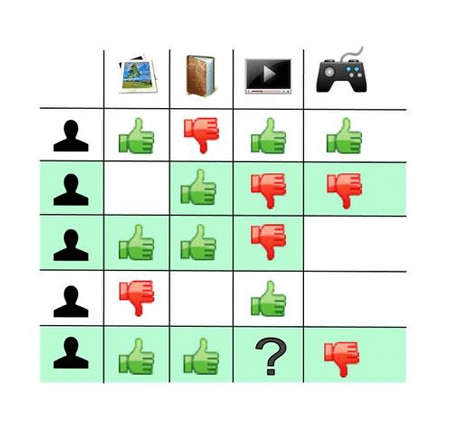
La recommandation est une problématique qui revient très souvent pour les data scientists : suggérer d'autres produits à acheter sur Amazon, des films à regarder sur Netflix, des musiques à écouter sur Spotify, etc. C'est vraiment utile, à la fois pour l'utilisateur final et pour l'entreprise qui peut ainsi proposer le contenu le plus pertinent.

Une recommandation, c'est une proposition de contenu **similaire** aux produits qu'a déjà aimé l'utilisateur. C'est la notion de **similarité** qui est donc à traduire.

Mais du coup, c'est de la classification ? de la régression ? supervisée ? non supervisée ?

Bonnes questions ! En réalité, ce problème peut être formulé d'énormément de manières différentes. Une technique largement répandue est le "**collaborative filtering**", qui se base sur des similarités entre utilisateurs, ou bien des similarités entre produits. Dans ces deux cas, c'est un problème **non supervisé** : on procure toutes nos données à l'algorithme et on le laisse essayer de déterminer les relations entre les différentes entités.

Sur l'image ci-dessous, on regarde par exemple ce qu'ont voté les utilisateurs similaires, c'est-à-dire ceux qui ont déjà voté la même chose sur d'autres produits (surlignés en vert). On peut alors prédire ce qu'aurait voté notre utilisateur sur le produit cherché, et ne proposer que les produits sur lesquels il aurait mis un pouce vert.



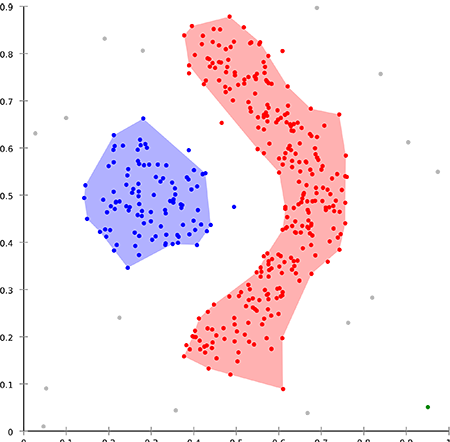
Les utilisateurs similaires (en vert) n'ont pas aimé le produit que notre utilisateur n'a pas encore noté.

L'algorithme aura donc tendance à prédire une mauvaise note et à ne pas recommander le produit ici.

Il existe d'autres méthodes de recommandation qui dépendent du problème considéré.

#### Le clustering

Le clustering désigne les méthodes de regroupement automatique de données qui se **ressemblent** le plus en un ensemble de "nuages", appelés **clusters**. Un ensemble d'algorithmes non supervisés peut réaliser cette tâche. Ceux-ci mesurent donc de manière automatique la **similarité** entre les différentes données. Par exemple, les points sur le graphe ci-dessous peuvent être considérés comme similaires s'ils sont proches en termes de distance.



L'objectif du clustering est de retrouver les différents clusters de données, c'est-à-dire de regrouper les données similaires entre elles.

C'est bien beau, mais à quoi ça sert en pratique ?

En pratique, le clustering permet de détecter des grandes catégories au sein des données. Une utilisation courante en marketing, par exemple, est le partitionnement automatique de consommateurs en différents segments. Vous vous en doutez maintenant, c'est aussi souvent utilisé pour les moteurs de recommandation automatique (e.g., recommander des instances appartenant au même cluster que l'instance considérée).

Une fois que vous avez ces segments, rien ne vous empêche ensuite d'appliquer d'autres algorithmes, supervisés cette fois : par exemple, classer les nouveaux utilisateurs sur ces segments, ou bien encore, identifier les outliers car ils n'appartiennent raisonnablement à aucun cluster.

### En résumé

Vous avez vu quelques exemples de solutions de machine learning courantes à des problématiques d'entreprises :

* affecter un score à un client ;
* prédire la rentabilité d’une campagne marketing ;
* identifier les événements rares ;
* affecter une catégorie à un produit ;
* segmenter les visiteurs d’un site ;
* recommander un produit à un client.

Une fois que vous aurez acquis l'expérience nécessaire, vous arriverez de plus en plus facilement à effectuer la traduction d'une problématique réelle en problème de machine learning !

**Sélectionnez les outils de Data Science appropriés**

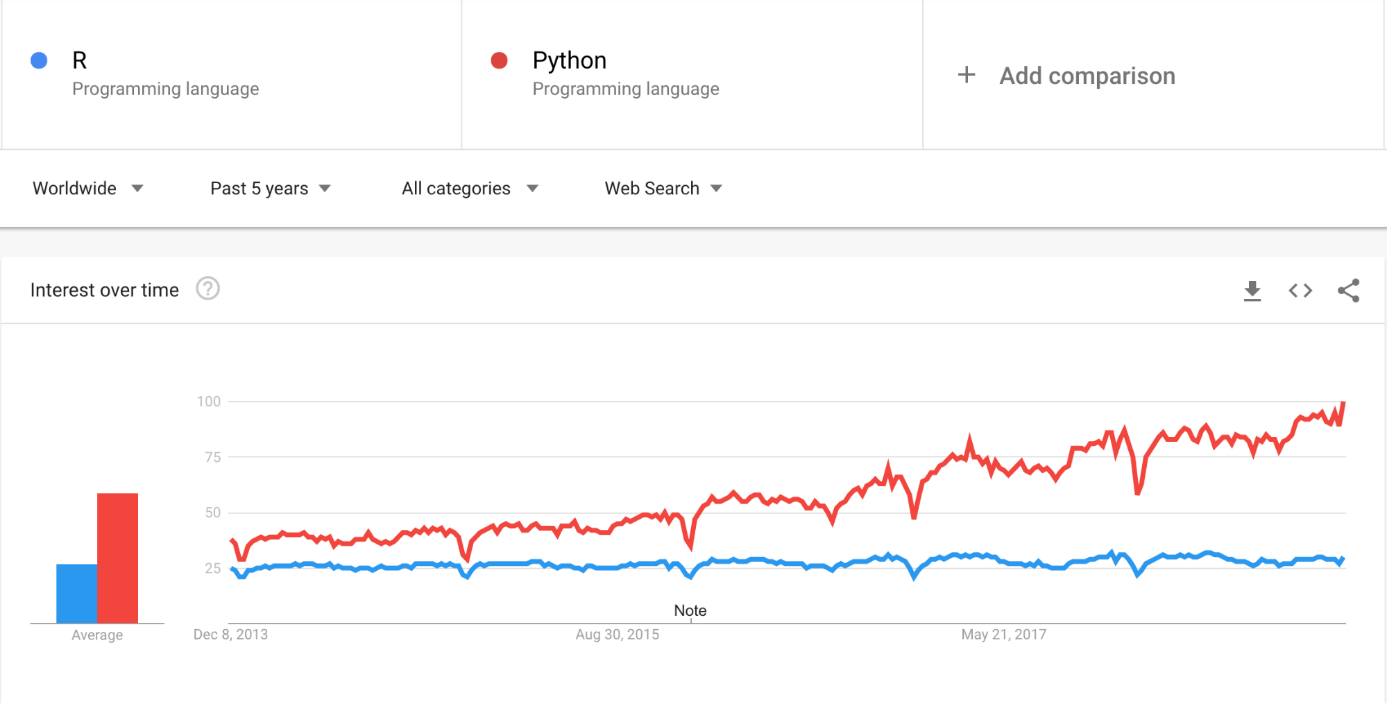
Chacune des étapes du travail du Data Scientist possède des outils spécifiques associés. Ce chapitre a pour vocation de vous présenter les choix les plus populaires pour chacun des domaines, et surtout des critères qui vous permettront de choisir selon votre propre problématique et vos contraintes.

**"R ou Python ?"**

Vous en avez sans doute déjà entendu parler, mais la guerre fait rage entre ces deux langages pour déterminer lequel sera dominant sur le marché de la Data Science.

Pour résumer, **R**, créé en 1993, est **le langage historique des statisticiens**, métier qui représente l’ancêtre de la Data Science. La transition s’est ainsi faite naturellement et vous trouverez votre bonheur dans ce langage en terme de fonctionnalités utiles au bon développement de votre projet et des interfaces pour (quasiment) tous les outils que nous évoquerons par la suite.

**Python**est aussi un “vieux” langage (1991). Il est devenu plus récemment **le langage de référence pour tous les ingénieurs** qui veulent effectuer rapidement des implémentations d’algorithmes mathématiques, entre autre de Machine Learning.



Quel est le langage le plus recherché sur Google ?

**Une guerre sans pitié sévit sur internet** depuis quelque temps pour déterminer lequel des deux langages il faut utiliser, notamment pour l’efficacité, mais aussi (et surtout) pour des raisons pratiques.

Avec l’expérience, voilà ce que je peux vous dire pour **vous aider à faire votre choix** :

* Si vous voulez vous orienter vers des professions qui sont fortement orientées en **Machine Learning** (Machine Learning Engineer, Data Scientist à dominante Machine Learning, etc.), **faites du Python** ! Tous les frameworks existants, les nouveaux algorithmes, sont écrits en Python. La communauté est aussi beaucoup plus importante sur internet pour ces thématiques.
* Si vous vous orientez vers un métier où les **statistiques** jouent un plus grand rôle (pour faire des analyses exploratoires, des corrélations, etc.), il est plus judicieux de travailler sur **R** qui permet d’être très efficace sur ce terrain. Notamment pour un **data analyst,** R constitue un très bon environnement de travail.
* Dernier conseil, si vous savez dans quel secteur et activité vous souhaitez travaillez, faites une recherche sur les demandes d’emploi dans ce métier et les préférences quant au langage précisé dans les descriptions. Vous aurez une bonne idée de la demande à ce sujet.

Et pour les cours OpenClassrooms, quel langage est utilisé ?

Ce cours (et les autres du parcours Data Scientist) étant orienté principalement Machine Learning, on utilisera le langage Python. R intègre la plus grande partie des algorithmes que nous allons évoquer dans les prochains cours du parcours et des wrappers quand ce n’est pas le cas (par exemple pour le deep learning). M**ais dans une optique métier à la fin du parcours, on favorisera l’utilisation de Python**.

Notez que le [**parcours Data Analyst**](https://openclassrooms.com/fr/paths/65-data-analyst) d’OpenClassrooms laisse le choix du langage à l’étudiant : R ou Python.

**La récupération des données**

La première étape est donc de récupérer les données. Vous vous en doutez, cela dépend fortement de votre problématique : une voiture autonome générera des données à partir de ses capteurs, une entreprise de trading récupérera les données sur des outils de marché financier, etc. Pas de recette toute faite donc. Voici quelques pistes un peu générales que je peux vous donner à titre d'inspiration :

* Pour récupérer des données texte depuis des pages internet, vous devrez utiliser un outil de scraping, comme scrapy en Python par exemple.
* Si vous avez accès à des données structurées comme SQL, elles ont en général leur propre syntaxe de récupération. C'est le cas pour la majorité des boîtes.
* Il existe des hubs de jeux de données publics qui peuvent parfois être utiles, a minima pour s'entraîner - [allez faire un tour sur ce site pour avoir une idée](https://github.com/awesomedata/awesome-public-datasets).  
  De manière générale, si vous travaillez dans une boîte très centrée sur les données, ce ne sera pas votre travail de consolider cet accès et la récupération des données, mais celui du **data architect**, qui vous créera un datalake comme point d'entrée unique.

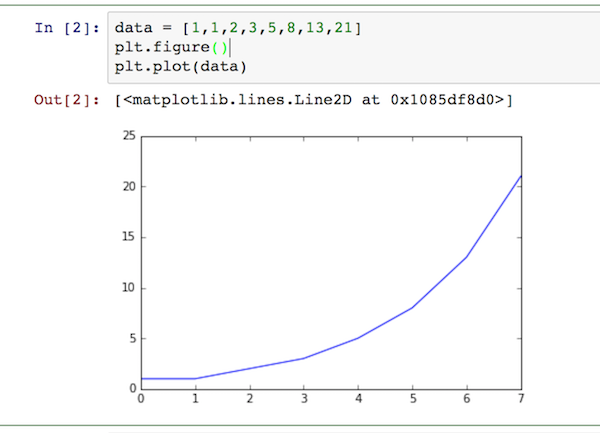
En revanche, dans tous les cas, gardez l’œil ouvert pour tout nouveau canal de récupération de données qui peut aider à résoudre votre problème, quitte à ajouter de nouveaux capteurs par exemple, ou être original sur le type de données générées, comme présenté dans le chapitre précédent.

**Le nettoyage et l'exploration des données**

En Python, l’écosystème **Scipy** est universellement utilisé avec ses librairies :

* **pandas** pour créer des tableaux (ou "Dataframe") à partir de vos données brutes ;
* **numpy** pour gérer des matrices ;
* **matplotlib** pour générer des graphiques ;
* **iPython** pour les feuilles de calculs…

Bien sûr, il en existe encore d'autres !



Utilisation de la librairie Maplotlib pour générer un graphique

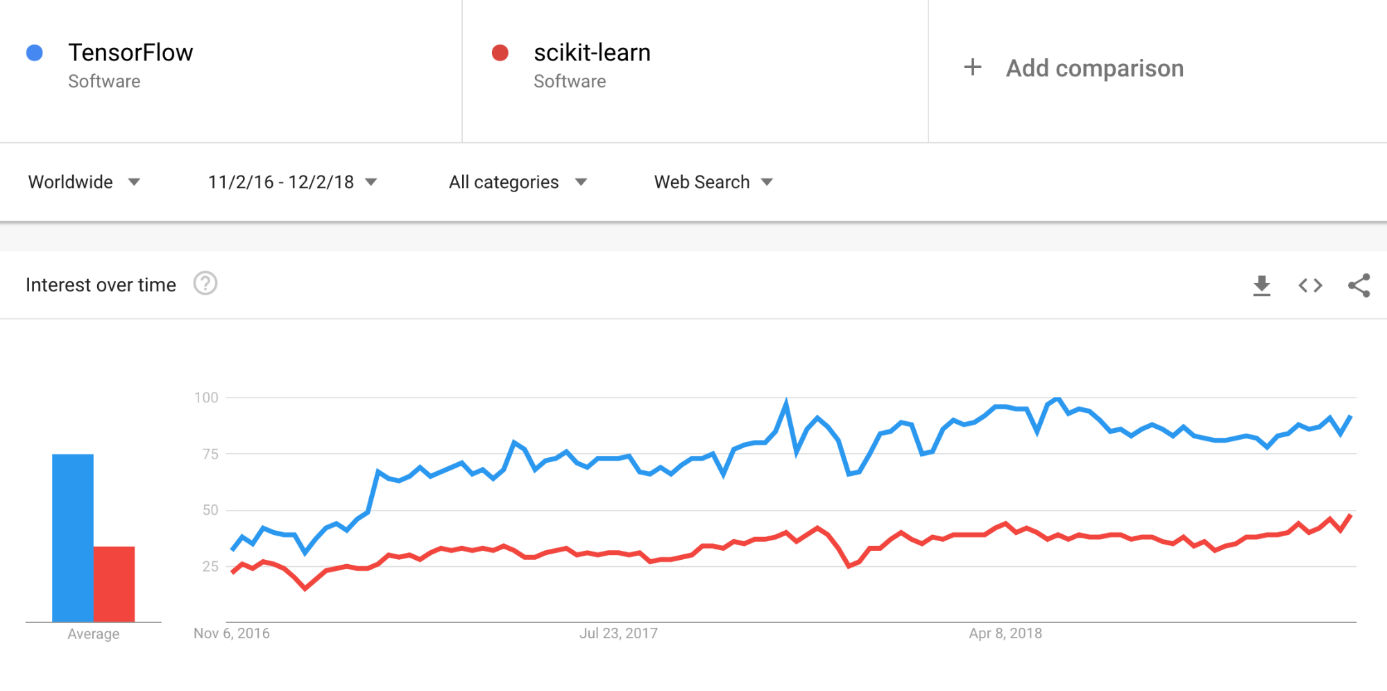
Cette famille de librairies est très utilisée et sert en réalité de base à beaucoup de frameworks de Machine Learning que nous allons utiliser par la suite. Nous allons les employer pour les cours du parcours Data Scientist d'OpenClassrooms.

Si vous n’êtes pas familier avec ces librairies, je vous invite à consulter le cours [**Découvrez les librairies Python pour la Data Science**](https://openclassrooms.com/fr/courses/4452741-decouvrez-les-librairies-python-pour-la-data-science).

**La phase de modélisation (et l'évaluation)**

Ici, on rentre dans le vif du sujet pour le Data Scientist et pour ce cours, c’est à dire le Machine Learning et le travail de modélisation.

À nouveau, la guerre fait rage. Torch, Theano, Caffe, mais surtout **Tensorflow** et **Scikit-Learn** sont les librairies les plus utilisées pour la modélisation. Dans ce cours, le choix a été fait d’utiliser **Scikit-learn**, car il est plus facile d’accès et implémente directement et de manière didactique les différents algorithmes d’apprentissage automatique que nous allons étudier.



Recherche de "TensorFlow" et Scikit-Learn sur Google

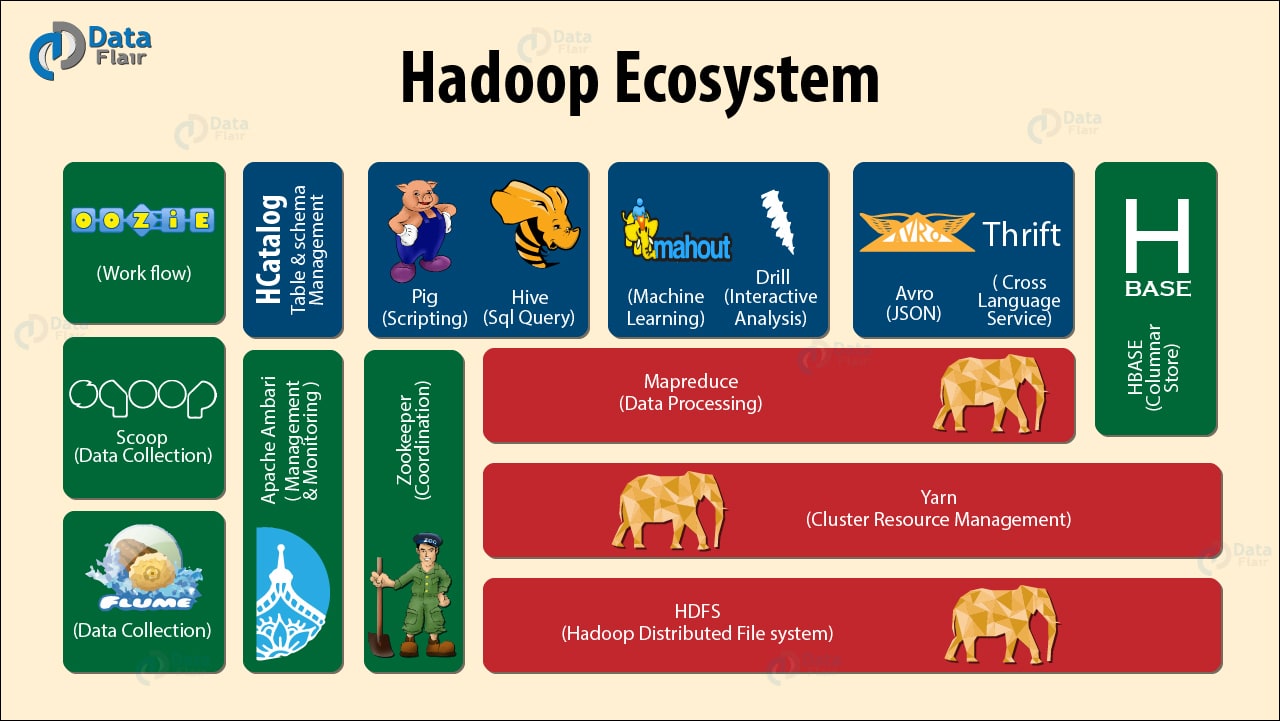
On dirait que **Tensorflow** semble plus populaire et possède une communauté plus active. Pourquoi ne pas avoir utilisé ce framework dans les cours ?

En effet, **Tensorflow** est plus populaire, mais **Scikit-Learn** est beaucoup plus simple d’accès. Dans un objectif pédagogique, c’est le choix évident pour aborder nos différentes thématiques. Mon conseil est de **se familiariser avec les deux**. Scikit-Learn permet de rapidement tester les modèles de bases utilisés en Data Science. Tensorflow permet plus de flexibilité dans l’implémentation et permet d’aller plus loin, notamment grâce à Keras, pour la construction d’algorithmes de Deep Learning.

**Le déploiement et la mise en production**

La manière la plus simple d’effectuer une mise en production est d’exporter notre modèle final et de le rendre accessible sous forme **d’API** sur un serveur.  
  
Cependant, comme évoqué précédemment, cette partie peut rapidement s’avérer laborieuse lorsqu’on travaille dans un environnement réel. La quantité de données, la rapidité d’exécution des calculs, la tolérance aux pannes sont autant d’exigences qui nécessitent de s’appuyer sur **des outils robustes et spécialisés**.

L’image ci-dessous représente une partie de l’écosystème **Hadoop** avec les différentes librairies qui répondent chacune à une utilisation spécifique de mise en place d’architecture Big Data.



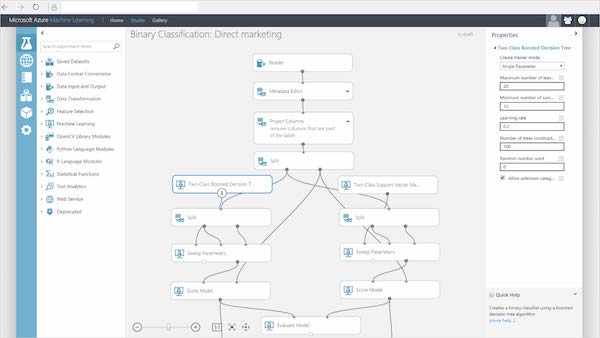
L'éco-système Hadoop est très fourni et répondra à tous vos besoins pour la mise en production

Et pour les cours OpenClassrooms, quelles sont les librairies utilisées ?

Ce cours est centré sur le travail de modélisation du Data Scientist – le Machine Learning – et non sur la mise en place d’une architecture en production. Pour plus d’informations sur ce sujet, je vous invite à consulter le [**parcours DataArchitect**](https://openclassrooms.com/fr/paths/64-data-architect), qui s'occupe en détail de la plupart des questions concernant la création du data lake et de la mise en production sur des Big Data.

**Passez par une API pour gagner en simplicité**

Il existe une alternative si vous êtes **pressé par le temps**, si vous voulez**tester rapidement la viabilité d’un projet** ou encore si vous n’avez**pas besoin de performances énormes,** c'est de passer par une **API**. Les plus reconnues sont Google Cloud AI, Microsoft Azure Machine Learning et AWS Machine Learning. Ces logiciels, hébergés dans le cloud, permettent directement de travailler sans code sur tous les aspects évoqués plus haut, notamment la partie modélisation et la mesure de performance.



Aperçu de l'interface d'Azure Machine Learning

Alors, je peux suivre les cours d’OpenClassrooms sans utiliser de librairie de Machine Learning, simplement à l’aide d’une API ?

Bonne question ! La majorité du parcours Data Scientist se concentre sur le fonctionnement des algorithmes qui sont présentés et non sur leur implémentation. Vous pourrez donc suivre les cours sans utiliser de framework code. Mais si vous souhaitez faire le parcours, vous devrez utiliser les outils spécialisés pour implémenter vous même les algorithmes !

**En résumé**

Pour chaque étape du travail du Data Scientist, il est important de se munir d’**outils efficaces**, mais surtout avec lesquels **vous êtes à l’aise**. Retenez aussi que le Machine Learning et la Data Science sont avant tout composés de concepts à appréhender correctement, indépendamment des outils de travail choisis.

# Identifiez les techniques et outils du Machine Learning

Bravo ! Vous avez réussi cet exercice !

### Compétences évaluées

* Identifiez les techniques et outils du Machine Learning

### Question 1

**Une entreprise veut prédire le prix d’un plein de carburant, connaissant les caractéristiques d'une voiture. Est-ce un problème de régression ou de classification ?**

* + 

C'est un problème de régression.

* + 

C'est un problème de classification.

*Le prix d'un plein est un nombre, c'est donc une régression.*

### Question 2

**J’ai une grande banque de textes que je veux catégoriser automatiquement par thématique. Aucun des documents n’a de thématique associée pour le moment. Comment vais-je entraîner mon modèle ?**

* + 

De manière supervisée

* + 

De manière non-supervisée

*Il n'y a pas de label (ou target) associé aux observations (les différents documents de texte), donc on est bien dans un cas non supervisé.*

### Question 3

**En effectuant une tâche de catégorisation sur mes textes, suis-je en train d’effectuer une régression ou une classification ?**

* + 

J'effectue une régression.

* + 

J'effectue une classification.

*On essaie de trouver une "thématique" aux différents textes ; c'est une catégorie discrète, pas un nombre, donc on est dans un cas de classification.*

### Question 4

**Lorsqu'on cherche à détecter les outliers, que faut-il faire ?**

* + 

Rechercher la bonne mesure de similarité d'une observation par rapport au modèle.

* + 

Utiliser la mesure de similarité qui est toujours utilisée dans la détection d'événements rares.

*Il n'y a pas une seule manière de mesurer l'éloignement d'une observation par rapport au modèle. C'est dépendant du contexte : si on travaille en marketing, on accepte plus les fluctuations qu'en médecine par exemple, donc il faut des mesures d'éloignement différentes.*

### Question 5

**Pour trancher entre R et Python quant au choix de la technologie à utiliser pour devenir Data Scientist et trouver un métier, l’idéal est de…**

* + 

Commencer par python puis basculer sur R si nécessaire.

* + 

Ne pas faire de R du tout, car ce logiciel est réservé aux statisticiens.

* + 

Se baser sur les descriptions d’offres d’emplois du secteur dans lequel on souhaite travailler.

*Comme décrit dans le cours, il faut simplement s’intéresser à “la niche” de data science sur laquelle vous voulez concentrer votre activité, pour voir quels logiciels et technologies sont employés.*

### Question 6

**Comment choisir les frameworks et librairies utilisés pour la partie modélisation ?**

* + 

Le choix est définitif, vous ne pourrez pas vous adapter et changer de framework.

* + 

Le choix dépend de la problématique de modélisation rencontrée, ils ont leur pour et leur contre.

* + 

Ils sont tous plus ou moins bons, mais Scikit-learn domine en terme de performances.

*Encore un choix à effectuer lorsqu’on a une meilleure idée des contraintes liées à la problématique rencontrée, afin de sélectionner la meilleure option.*

### Question 7

**Pour la partie de mise en production…**

* + 

c'est le travail uniquement du data engineer qui saura se débrouiller tout seul.

* + 

tant qu’il ne rencontre pas de contrainte majeure d’industrialisation, le data scientist peut faire ce travail.

* + 

il faut aussi apprendre à développer en Java pour livrer des programmes informatiques robustes en production.

*La majorité des problèmes que vous allez rencontrer ne nécessitent pas de technologies avancées particulières.*

### Question 8

**Les API telles que AWS ML ou Microsoft Azure ML...**

* + 

ne fonctionnent pas encore très bien, mieux vaut s’en tenir au languages classiques.

* + 

sont de bonnes alternatives pour outrepasser les problématiques de performance habituelles, et permettent d’obtenir de meilleurs résultats.

* + 

permettent d’opérer des solutions de machine learning avec une grande puissance de calcul et des algorithmes pré-entrainés mais sont limitées en terme de flexibilité.

*Ces API permettent de résoudre une grande partie des problèmes simples en terme de contraintes et de complexité. Mais on est vite limité lorsqu’on est confronté à des problèmes réels.*

# Identifiez les techniques et outils du Machine Learning

Bravo ! Vous avez réussi cet exercice !

### Compétences évaluées

* Identifiez les techniques et outils du Machine Learning

### Question 1

**Une entreprise veut prédire le prix d’un plein de carburant, connaissant les caractéristiques d'une voiture. Est-ce un problème de régression ou de classification ?**

* + 

C'est un problème de régression.

* + 

C'est un problème de classification.

*Le prix d'un plein est un nombre, c'est donc une régression.*

### Question 2

**J’ai une grande banque de textes que je veux catégoriser automatiquement par thématique. Aucun des documents n’a de thématique associée pour le moment. Comment vais-je entraîner mon modèle ?**

* + 

De manière supervisée

* + 

De manière non-supervisée

*Il n'y a pas de label (ou target) associé aux observations (les différents documents de texte), donc on est bien dans un cas non supervisé.*

### Question 3

**En effectuant une tâche de catégorisation sur mes textes, suis-je en train d’effectuer une régression ou une classification ?**

* + 

J'effectue une régression.

* + 

J'effectue une classification.

*On essaie de trouver une "thématique" aux différents textes ; c'est une catégorie discrète, pas un nombre, donc on est dans un cas de classification.*

### Question 4

**Lorsqu'on cherche à détecter les outliers, que faut-il faire ?**

* + 

Rechercher la bonne mesure de similarité d'une observation par rapport au modèle.

* + 

Utiliser la mesure de similarité qui est toujours utilisée dans la détection d'événements rares.

*Il n'y a pas une seule manière de mesurer l'éloignement d'une observation par rapport au modèle. C'est dépendant du contexte : si on travaille en marketing, on accepte plus les fluctuations qu'en médecine par exemple, donc il faut des mesures d'éloignement différentes.*

### Question 5

**Pour trancher entre R et Python quant au choix de la technologie à utiliser pour devenir Data Scientist et trouver un métier, l’idéal est de…**

* + 

Commencer par python puis basculer sur R si nécessaire.

* + 

Ne pas faire de R du tout, car ce logiciel est réservé aux statisticiens.

* + 

Se baser sur les descriptions d’offres d’emplois du secteur dans lequel on souhaite travailler.

*Comme décrit dans le cours, il faut simplement s’intéresser à “la niche” de data science sur laquelle vous voulez concentrer votre activité, pour voir quels logiciels et technologies sont employés.*

### Question 6

**Comment choisir les frameworks et librairies utilisés pour la partie modélisation ?**

* + 

Le choix est définitif, vous ne pourrez pas vous adapter et changer de framework.

* + 

Le choix dépend de la problématique de modélisation rencontrée, ils ont leur pour et leur contre.

* + 

Ils sont tous plus ou moins bons, mais Scikit-learn domine en terme de performances.

*Encore un choix à effectuer lorsqu’on a une meilleure idée des contraintes liées à la problématique rencontrée, afin de sélectionner la meilleure option.*

### Question 7

**Pour la partie de mise en production…**

* + 

c'est le travail uniquement du data engineer qui saura se débrouiller tout seul.

* + 

tant qu’il ne rencontre pas de contrainte majeure d’industrialisation, le data scientist peut faire ce travail.

* + 

il faut aussi apprendre à développer en Java pour livrer des programmes informatiques robustes en production.

*La majorité des problèmes que vous allez rencontrer ne nécessitent pas de technologies avancées particulières.*

### Question 8

**Les API telles que AWS ML ou Microsoft Azure ML...**

* + 

ne fonctionnent pas encore très bien, mieux vaut s’en tenir au languages classiques.

* + 

sont de bonnes alternatives pour outrepasser les problématiques de performance habituelles, et permettent d’obtenir de meilleurs résultats.

* + 

permettent d’opérer des solutions de machine learning avec une grande puissance de calcul et des algorithmes pré-entrainés mais sont limitées en terme de flexibilité.

*Ces API permettent de résoudre une grande partie des problèmes simples en terme de contraintes et de complexité. Mais on est vite limité lorsqu’on est confronté à des problèmes réels.*

## Construisez un modèle statistique

Maintenant que nous avons posé les bases du machine learning et que vous avez une meilleure idée de quand et comment l'utiliser… il est temps de rentrer dans le concret ! 🤓 Que diriez-vous d'entraîner vos premiers algorithmes ? C'est précisément ce que nous allons attaquer dans cette deuxième partie du cours. Mais avant de nous lancer dans le code, développons dans ce chapitre notre intuition de la notion **d’apprentissage** et ce qu’elle représente lorsqu’on parle de machine learning – c’est à dire **d’algorithmes** qui peuvent **améliorer leurs performances à partir d’expériences**.

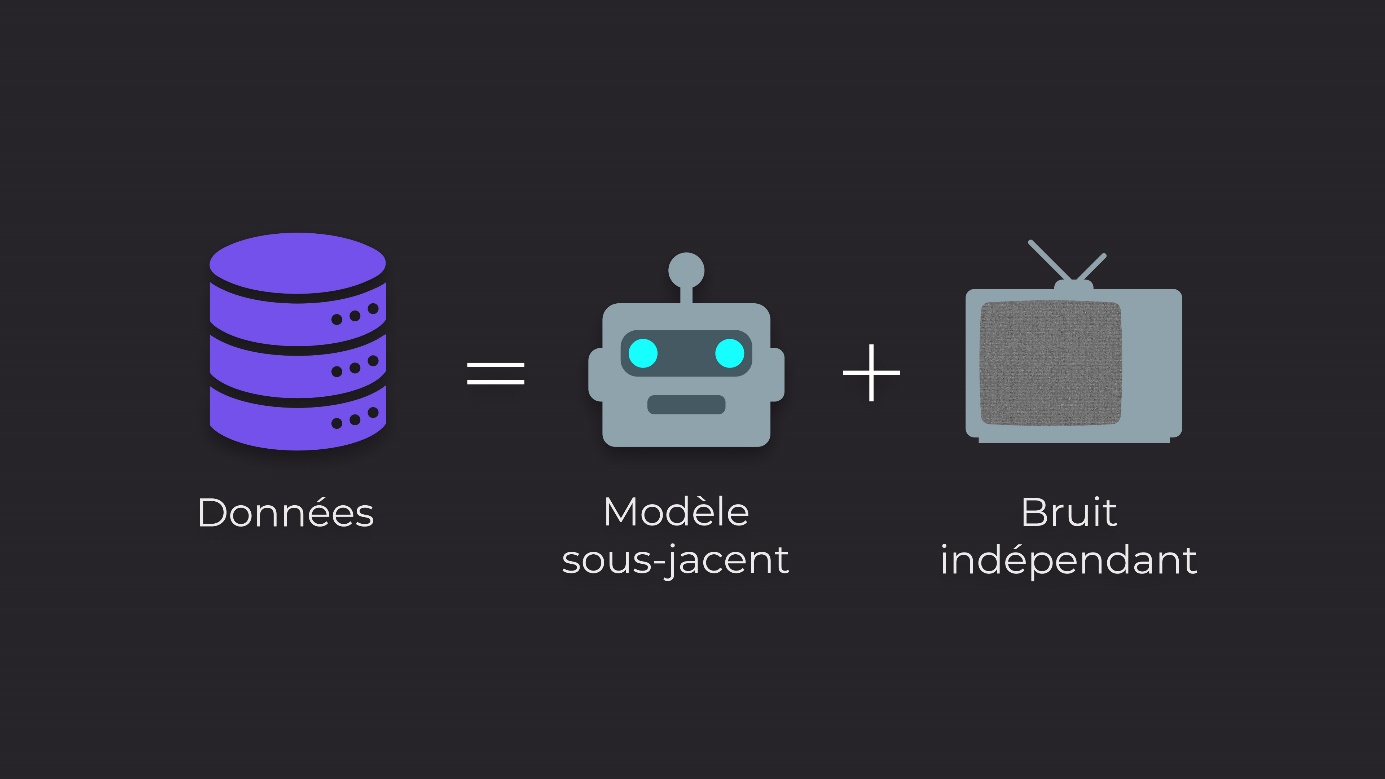
### Construisez un modèle statistique

Qu'est-ce qu'on essaie de faire déjà ?

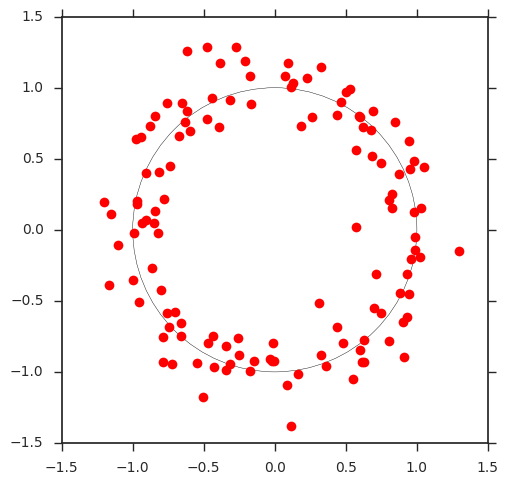
Rappelez-vous : l’objectif du machine learning est de**trouver un modèle** qui effectue une **approximation** de la réalité (le phénomène à l’origine des données), à l’aide de laquelle on va pouvoir effectuer des prédictions.

Et forcément, parce qu’on fait une approximation, on a une perte d’information qui est un bruit non modélisé et qu’on estime indépendant (c’est-à-dire non représentatif du phénomène).

La formule se résume à l'équation suivante :



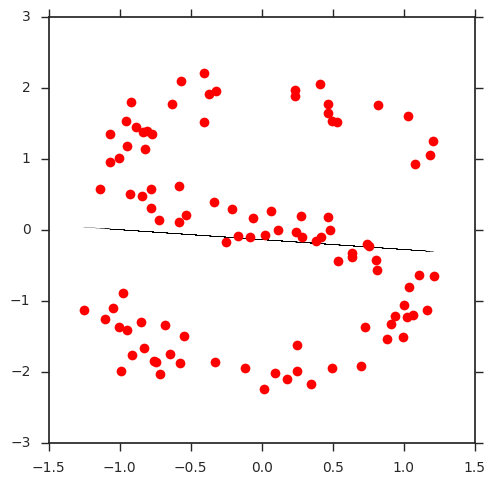
Données = Modèle sous-jacent + bruit indépendant



Ici, on voit facilement qu'on peut approximer le phénomène à l'origine des données par un cercle.

Le **modèle sous-jacent (statistique)** représente une **contrainte** de “forme” non dépendante des données.

Par exemple, si je décide d’effectuer une régression linéaire, je contrains mon modèle à avoir la forme d’une droite, pas d’un cercle. Malgré tous mes efforts, je ne pourrais alors pas obtenir d’autre forme qu'une droite pour modéliser mes données, c’est donc un choix très important ! Cette notion de complexité/contrainte/flexibilité est primordiale dans votre travail de modélisation. 🙂



Comme on peut le voir sur ce genre de données, c'est difficile (impossible) de faire fitter une droite ! Le type de modèle est mal choisi.

Une fois ce choix effectué et les contraintes posées, la question d’apprentissage est alors équivalente à construire le fameux modèle qui se rapprochera le plus des données d’exemple.

Les modèles sont le plus souvent représentés par un ensemble de **paramètres** qu'on mettra dans un vecteur θ. Par exemple, une droite peut être représentée par l'équation   .

On parle alors de modèle paramétrique, et l'apprentissage du modèle revient dans ce cas à trouver la valeur optimale de θ.

### Utilisez la fonction loss, ou la perte d’information

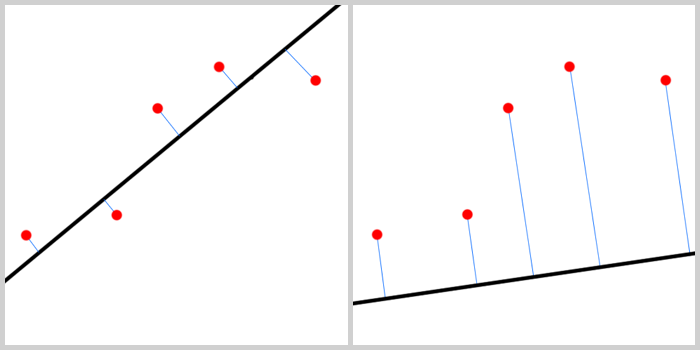
En apprentissage supervisé, la notion principale est celle de **perte d’information** (loss en anglais) due à l'approximation dont je viens de parler. Elle détermine à quel point notre modélisation du phénomène, qui est une approximation de la réalité, perd de l’information par rapport à la réalité observée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

La grande majorité des algorithmes d’apprentissage supervisé utilisent cette fonction de perte ! L'apprentissage se résume en fait 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.

On peut retrouver beaucoup de types de perte. Je vais vous en présenter deux ici, pour vous donner une idée des types d'approche qu'on peut avoir.

#### Minimisez l’erreur du modèle

Une première manière de représenter cette perte se fait par ce qu’on appelle **l’erreur** ou le **risque**, qui traduit l’éloignement des données par rapport à la prédiction effectuée par le modèle considéré.



À gauche, on ne perd pas trop d'information. À droite, par contre, on est trop éloigné de la réalité représentée par les points.

La distance la plus utilisée pour mesurer cet éloignement est l’**erreur quadratique** (la distance euclidienne entre un point et le modèle).

Souvent, on ne peut pas calculer directement l’erreur mais on va utiliser une approximation à partir des données qui sont notre seule ressource. 😐 On va ainsi sommer sur toutes nos données d’exemples l’erreur effectuée du modèle. On appelle cette erreur le **risque empirique**.

Lors de l'apprentissage de notre modèle, on cherche alors à **converger vers le minimum** de risque empirique.

Le risque empirique peut paraître sorti du chapeau. Ce n'est pas entièrement faux, car il représente un proxy pour une erreur réelle qu'on aimerait bien pouvoir mesurer. 😢

Si ça vous intéresse, il existe tout un domaine d'étude appelé statistical learning theory qui permet de creuser un peu plus le fonctionnement de ce genre d'approximation d'erreur.

Cette erreur du modèle utilisée pour l'optimisation est à ne pas confondre avec celle utilisée lors de l'évaluation des performances, qui utilisera d'autres mesures pour étudier la qualité finale du modèle.

#### Maximisez la vraisemblance du modèle

Un autre type de fonctionnement des algorithmes qu’on retrouve beaucoup en machine learning sur des modèles paramétriques, et qui nous vient des statistiques, est la maximisation de la **vraisemblance** (maximum likelihood estimation).

Je rappelle que la vraisemblance d'un jeu d'observations (x1...xN) par rapport à un modèle en statistiques est la fonction suivante :  .

Cette valeur représente la probabilité d'avoir le jeu d'observations étant donnés les paramètres θ.

Dans ce cas-là, l’objectif pour l’algorithme sera de **converger vers le maximum** de la fonction de vraisemblance du phénomène considéré, en trouvant le bon θθ, toujours à partir de nos observations de départ.



On note  avec un accent circonflexe lorsqu'on parle d'un estimateur et pas de la valeur réelle.

Mais c’est une maximisation, pas une minimisation ! Où est la fonction de perte dont vous parliez dans ce cas-là ?

En effet, elle est un peu cachée, mais on peut trouver mathématiquement que maximiser la vraisemblance est en fait équivalent à minimiser une fonction de perte. Et c’est une approche très souvent utilisée, qui représente des familles d’algorithmes à part entière.

Pour ceux qui veulent en savoir plus sur ce lien entre maximiser la vraisemblance et minimiser une erreur, vous pouvez aller voir [**cet article**](https://quantivity.wordpress.com/2011/05/23/why-minimize-negative-log-likelihood/) (en anglais).

### Une histoire d'optimisation numérique

Au final, l'apprentissage est souvent une histoire d'optimisation numérique ! Les fonctions de perte que je viens de vous présenter sont des exemples illustratifs de l'approche qu'on développe pour construire un algorithme de machine learning. Une grande partie des algorithmes s’appuient donc sur des **méthodes d’optimisation numérique**, c’est-à-dire des méthodes qui vont rechercher un maximum ou un minimum d’une fonction déterminée (vous entendrez parler de loss, entropy, energy, likelihood, etc.) de manière **exacte** ou **approximée**.

Ce domaine des mathématiques, appelé optimisation convexe, est très intéressant. Il peut vous aider, lorsque vous travaillez avec des algorithmes, à anticiper si les méthodes utilisées ont des chances de converger – mais c’est aussi une des pierres angulaires de la création de nouveaux algorithmes. Pour aller plus loin, je vous conseille notamment ce [**cours d'optimisation orienté apprentissage statistique**](https://www.di.ens.fr/~aspremon/ENSM1.html). Attention, solides pré-requis en maths conseillés. 🤓

D'accord mais, en pratique, qu'est-ce que j'aurai à faire en tant que data scientist ?

En pratique, les algorithmes d’optimisation sont déjà implémentés dans les librairies que vous utiliserez (par exemple dans Scikit-Learn que j'utilise dans ce cours) et sont intégrés au modèle que vous voudrez créer. Toutes les notions abordées dans ce chapitre sont donc encapsulées dans des fonctions, que vous aurez juste à appeler en leur passant vos données d’entraînement.

Cependant, vous aurez souvent besoin de paramétrer ces algorithmes d’optimisation – ou même les hyperparamètres du modèle – c’est donc important de savoir dans quoi on met les mains. 😉

### En résumé

* Pour construire un modèle, on part d'une hypothèse de départ qui représente l'ensemble des formes que peut prendre notre modélisation (une courbe par exemple, mais ça peut être bien d'autres choses encore).
* En apprentissage supervisé, on cherche ensuite à trouver le modèle optimal à l'aide des données d'entraînement. Cela consiste à faire converger une mesure appelée loss function (fonction de **perte**) en utilisant des techniques d'optimisation numérique.
* Deux exemples de fonctions de perte souvent utilisées en apprentissage supervisé sont **le risque empirique** et le **maximum de vraisemblance**.
* Les algorithmes de machine learning sont en fait une combinaison de plusieurs domaines d'études : les statistiques (statistical learning theory), l'optimisation numérique, l'informatique théorique, etc.

## Programmez votre première régression linéaire

La **régression linéaire** est un premier exemple simple de la manière dont un algorithme peut apprendre un modèle. Pour franchir le pas, je vous propose maintenant de l'aborder en pratique. Donc, à vos claviers !

### Installez les logiciels utilisés

Vous aurez besoin d’installer [scikit-learn](https://openclassrooms.com/fr/courses/2304731-learn-python-basics-for-data-analysis/6009031-get-started-with-python-and-anaconda) pour coder l’exemple en même temps que moi.

Je vous recommande aussi d’utiliser un [notebook Jupiter,](https://openclassrooms.com/fr/courses/2304731-learn-python-basics-for-data-analysis/6009061-code-using-a-jupyter-notebook) pratique pour afficher vos résultats au fur et à mesure directement dans un navigateur web.

### Définissez la problématique et ses données d'entraînement

Reprenons notre problématique des loyers abordée dans le premier chapitre de ce cours. La question qu'on essaie de résoudre est :

Étant donné les caractéristiques de mon appartement, **combien** devrais-je normalement payer mon loyer ?

Imaginons pour l'instant que la seule caractéristique dont nous disposons est la surface de l'appartement. Notre training set est un ensemble de N = 545 observations de surface et des loyers associés :  (x,y)=(surface,loyer).

Ce sont des données réelles que j'ai récupéré sur des sites de location pour quelques arrondissements parisiens. J'ai enlevé quelques points aberrants (notamment sur les surfaces trop grandes). On peut donc se concentrer sur la modélisation.

[**Vous pouvez télécharger ce training set ici**](https://static.oc-static.com/prod/courses/files/initiez-vous-au-machine-learning/house.csv).

Commençons par charger et afficher les données d'entraînement, juste pour avoir une meilleure idée de ce à quoi on a affaire :

# On importe les librairies dont on aura besoin pour ce tp

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

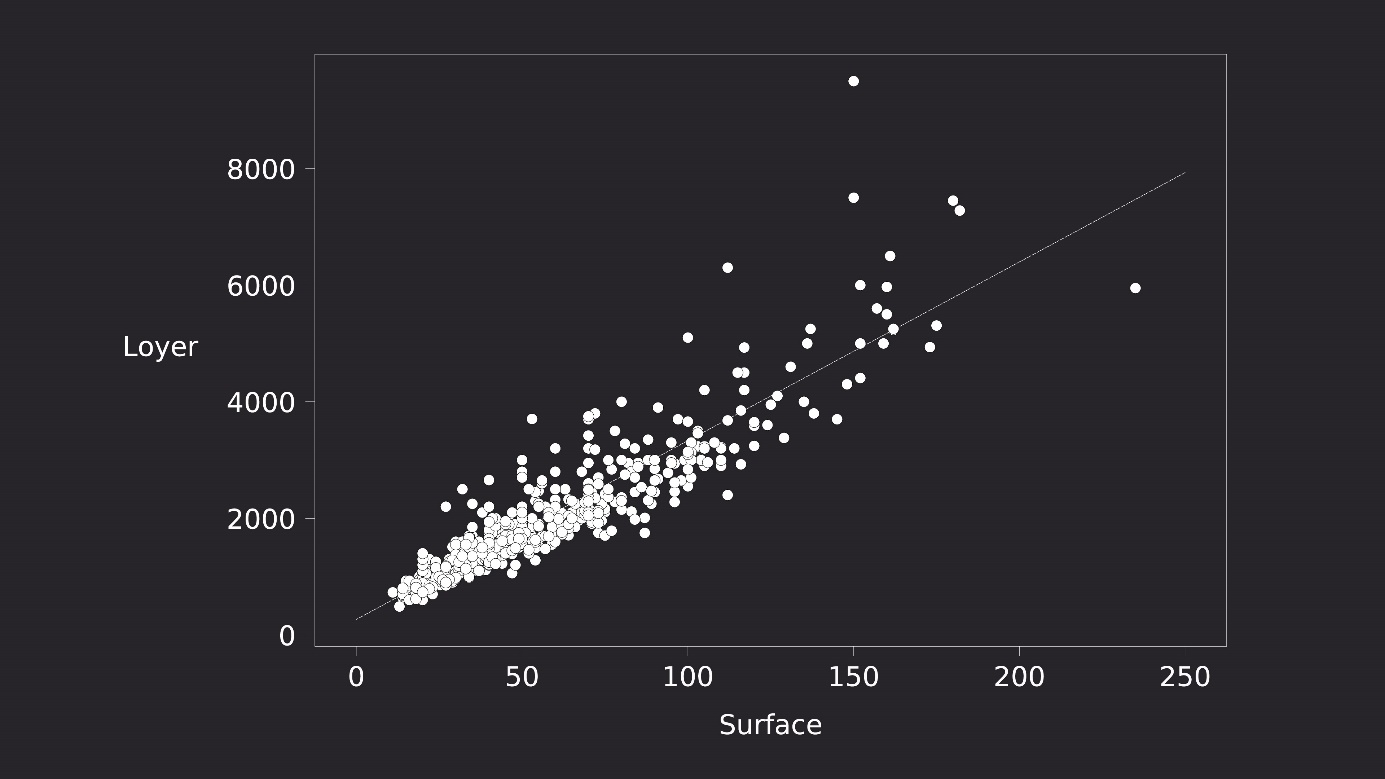
# On charge le dataset

house\_data = pd.read\_csv('house.csv')

# On affiche le nuage de points dont on dispose

plt.plot(house\_data['surface'], house\_data['loyer'], 'ro', markersize=4)

plt.show()



Répartition des loyers sous forme de droite

Clairement, d'après la visualisation, on peut se dire **que le montant du loyer dépend de manière linéaire de la surface du logement**. On peut donc émettre une hypothèse de modélisation qui est que le phénomène possède la forme d'une droite.

Aussi, on peut voir que lorsque la surface devient un peu trop grande, les données semblent devenir moins modélisables facilement, il y a plus de variabilité. On va considérer pour l'instant résoudre le problème de prédiction pour les loyers inférieurs à 10,000€, afin de conserver une robustesse du modèle à ces données plutôt anormales, qui correspondent peut-être à un autre modèle distinct ou à un traitement comme outliers.

house\_data = house\_data[house\_data['loyer'] < 10000]

Dans notre exemple, nous utilisons un dataset **unidimensionnel** en entrée (où les observations dépendent seulement de la variable x = surface) pour un souci de confort, car c’est plus facile à représenter visuellement. En réalité, on a bien plus de **dimensions** (features) lorsqu’on travaille avec de "vraies" données !

### Reformulez le problème dans l'espace d’hypothèse : une droite

La régression linéaire s’appuie sur l’hypothèse que les données proviennent d’un phénomène qui a la forme d'une droite, c’est-à-dire qu’il existe une relation linéaire entre l’entrée (les observations) et la sortie (les prédictions).

Nous avons donc notre contrainte de modèle sous-jacent qui doit être sous la forme 

où N est le nombre d'observations à notre disposition (id est le nombre d'appartements considérés dans notre cas).

On met avec un chapeau pour le distinguer des observations réelles. On parle de l’estimation ici donnée par le modèle.

En général, une observation a plusieurs variables qui la caractérisent (un appartement est par exemple caractérisé par une surface et un nombre de pièces). Donc une observation est caractérisée par un vecteur avec D le nombre de variables caractérisant l'observation.

La constante 1 est ajoutée à chaque observation pour représenter l'ordonnée à l'origine et avoir une écriture vectorielle plus compacte.

Dans notre cas, puisqu’on est en une dimension,   , on peut écrire pour une observation 

où   représente la surface pour du  appartement de notre échantillon de taille   .

Je rappelle que les données sont ainsi générées :

données = modèle sous-jacent + bruit irréductible

C'est-à-dire ici :

données =   + bruit irréductible

Notre objectif est donc de trouver la droite paramétrée par qui fitte le mieux aux données d’entraînement.

### Définissez la fonction loss

Pour effectuer une régression linéaire, on doit ensuite choisir une fonction de perte dont on a parlé dans le chapitre précédent. On ne va pas prendre de risque, on va prendre la distance euclidienne. 😉

**La distance euclidienne**d'une observation   vis-à-vis de notre modèle   est :



**Le risque empirique** dont on parlait dans le chapitre précédent est donc, pour N observations :



Et on va donc chercher à trouver le θθ qui minimise cette fonction de perte (qu’on note E) :

****

### Apprentissage : trouvez le θ optimal

Pour la régression linéaire, la solution de l'équation de minimisation (juste au-dessus) est exacte :



**Détails de la solution**

On cherche à minimiser la fonction E en fonction de theta. La fonction est convexe (c’est une somme de carrés), donc elle possède un optimum global (en l’occurrence un minimum), qui se trouve à l’endroit où la dérivée première en theta est nulle. Pour les détails de la démonstration, il en existe pléthore en ligne, [**par exemple ici**](http://eli.thegreenplace.net/2014/derivation-of-the-normal-equation-for-linear-regression) (en anglais).

Pourtant, vous parliez dans le chapitre précédent d'algorithme itératif qui convergeait vers une solution, non ? Là, on a une solution exacte directement, en fait ?

Alors, en l’occurrence on a de la chance. 😅 Il existe bien une solution exacte dans ce cas précis, qu’on vient de donner. Mais on peut aussi utiliser un algorithme appelé descente de gradient pour trouver une approximation de la solution. C'est en particulier utile lorsque l'on a beaucoup de données d'exemple, car c'est assez long pour un ordinateur de calculer la solution exacte ci-dessus (on calcule un inverse de matrice, ce qui n'est pas gratuit en temps de calcul !).

Bon en tout cas, on peut maintenant calculer les paramètres directement :

# On décompose le dataset et on le transforme en matrices pour pouvoir effectuer notre calcul

X = np.matrix([np.ones(house\_data.shape[0]), house\_data['surface'].values]).T

y = np.matrix(house\_data['loyer']).T

# On effectue le calcul exact du paramètre theta

theta = np.linalg.inv(X.T.dot(X)).dot(X.T).dot(y)

print(theta)

On trouve donc la valeur numérique de θ pour nos données :

[[ 266.45460292]

[ 30.66119596]]

Notre modèle final qui colle bien aux données sera donc dans notre cas (approximativement) :



On peut représenter graphiquement la droite qu'on a trouvée, pour vérifier qu'elle colle bien aux données :

plt.xlabel('Surface')

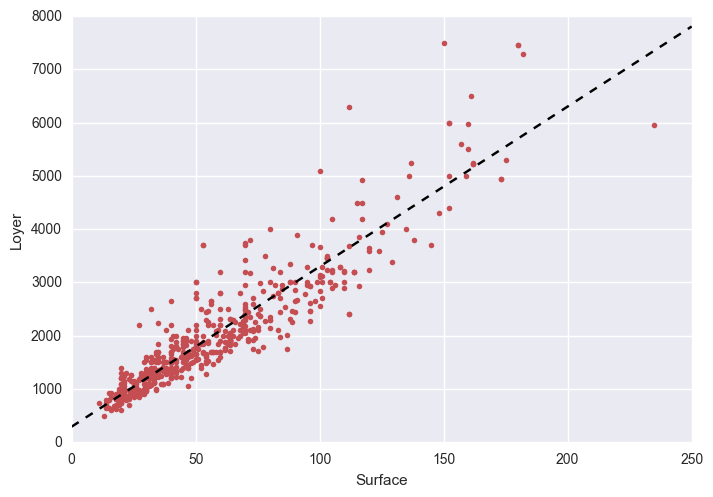
plt.ylabel('Loyer')

plt.plot(house\_data['surface'], house\_data['loyer'], 'ro', markersize=4)

# On affiche la droite entre 0 et 250

plt.plot([0,250], [theta.item(0),theta.item(0) + 250 \* theta.item(1)], linestyle='--', c='#000000')

plt.show()



Notre régression linéaire

### Utilisez le modèle pour effectuer des prédictions

Maintenant qu’on a notre paramètre **θ**, c’est-à-dire qu’on a trouvé la droite qui colle le mieux à nos données d’entraînement, on peut effectuer des prédictions sur de nouvelles données, c'est-à-dire prédire le loyer en fonction de la surface qu’on nous donne en entrée, en appliquant directement la formule du modèle ci-dessus.

Par exemple, si on l’applique pour une surface de 35 mètres carrés :

theta.item(0) + theta.item(1) \* 35

On obtient une estimation du loyer :

1339.646166

Essayez avec n'importe quelle surface !

On vient ici de décomposer l’entraînement de la régression linéaire. En réalité, vous vous doutez bien que cette régression est déjà implémentée dans le package scikit-learn. On peut l'utiliser directement de la manière suivante :

from sklearn import linear\_model

regr = linear\_model.LinearRegression()

regr.fit(X, y)

regr.predict(<des données de test>)

**Bonne nouvelle !** Les prochains modèles que nous utiliserons auront leurs algorithme d'entraînement et de prédiction directement implémentés dans scikit-learn, nous aurons juste à appeler directement leurs fonctions associées. 😎

### En résumé

* À partir d'une problématique et d'un dataset, nous avons considéré une hypothèse de travail pour contraindre le modèle : ici nous nous sommes placés dans le cas d'une régression linéaire, qui signifie contraindre la forme du modèle à une droite.
* Nous avons décomposé l'entraînement de ce modèle sur les observations, afin de déterminer le paramètre (pente et ordonnée à l'origine) de la droite optimale pour ces données. C'est cette partie que l'on appelle apprentissage du modèle.
* À l'aide du modèle ainsi trouvé, nous avons effectué des prédictions de montant de loyer à partir de n'importe quelle surface donnée.

### Allez plus loin : le “vrai” travail de modélisation

Maintenant qu’on a trouvé un premier modèle, il serait possible de tester plein d’hypothèses différentes pour aller plus loin et améliorer ses performances.

Que se passe-t-il si :

* on change l’hypothèse de linéarité (une droite) et qu’on en prend une autre (un polynôme du second degré par exemple) ?
* on teste le modèle avec d’autres types d’erreurs que la distance euclidienne ?
* on ajoute des features (dimensions) supplémentaires en entrée ?
* au fur et à mesure que la surface augmente, les données ont l'air d'être de plus en plus "éparses" ? Comment intégrer ce comportement dans ma modélisation ?

Toutes ces questions peuvent (et devraient !) être testées. Trouver la configuration la plus pertinente revient, du coup, à tester les différents modèles qui en découlent (régression linéaire ou non linéaire, dimensions différentes, et tous les autres algorithmes que l'on verra par la suite) et trouver celui qui convient le mieux en termes de performance.

Pas de panique, un [**cours entier**](https://openclassrooms.com/fr/courses/4297211-evaluez-et-ameliorez-les-performances-dun-modele-de-machine-learning) est dédié à l’évaluation et l’amélioration des performances du modèle.

**Exploitez votre jeu de données**

Le jeu de données (*dataset*) dont vous disposez constitue une ressource précieuse. Il faut pouvoir l’utiliser à bon escient, afin de pouvoir à la fois **choisir** un modèle et l'**entraîner** (ce que nous avons fait dans le chapitre pratique précédent), mais aussi de pouvoir **tester** la qualité de ce modèle.

Voyons cela de plus près !

**Echantillonnez les données**

La première question à se poser est : est-ce que l'on va utiliser toutes les données d'exemple dont on dispose ?

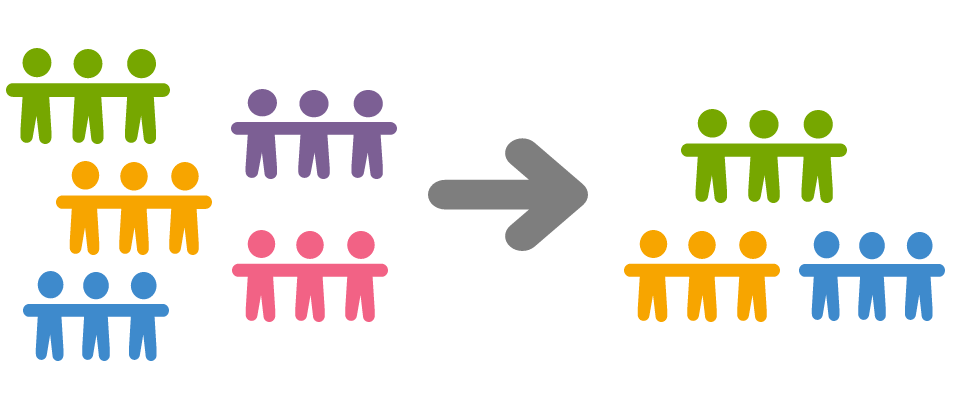
En effet, s'il s’avère qu’on a beaucoup de données d’entraînement et/ou que l’algorithme d’apprentissage est lourd, il est possible qu’utiliser toutes les données prenne énormément de temps et/ou de ressources hardware. Dans ce cas, il faut naturellement **échantillonner** et ne récupérer qu’un petit pourcentage du dataset qui servira au travail de modélisation pour aller plus vite. On parle d'étape de *sampling* en anglais.

Ça ne sert à rien d'utiliser toutes les données pour tester des modèles et se retrouver à attendre à chaque fois plusieurs minutes au lieu de quelques secondes, n'est-ce pas ? 😇

On peut quand même utiliser tout le dataset pour entraîner le modèle *final.* Mais pour explorer et pour le travail quotidien, il vaut en général mieux tester sur une petite partie des données.

Le problème lorsque l'on effectue un échantillonnage, c'est que l’on doit être bien sûr que cet échantillon est **représentatif** de toutes les données. En effet, dans l’exemple des loyers par exemple, si je ne récupère que des données qui proviennent d'Ile-de-France, je serai forcément biaisé pour mes prédictions de loyer qui proviennent d'autres régions. Il faut donc s’assurer de piocher les données de manière uniforme sur toute la population que l'on souhaite considérer.

Soyez vraiment vigilant lors du sampling. C'est certes très utile, mais si on ne sélectionne pas de manière bien distribuée, on augmente le **biais** et notre modèle devient moins représentatif de la réalité.

Si on ne fait pas attention, on peut mal sélectionner au sein de la population et introduire un biais !

Pour effectuer un échantillon, c'est assez simple. Une technique est de récupérer des identifiants tirés aléatoirement grâce à la fonction de la librairie Python "Numpy"  randint .

sample = np.random.randint(data\_size, size=int(data\_size\*0.1) )

sampled\_data = data[sample]

**Mettez de côté une partie des données pour tester votre modèle**

Comme nous l’avons vu, 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. On appelle aussi cela *held-out data*, pour souligner que ce sont des données auxquelles on ne va pas toucher avant la toute fin pour pouvoir être bien sûr que le modèle fonctionne.

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.**

On peut cette fois utiliser la fonction de scikit-learn  train\_test\_split qui prend en paramètre la proportion désirée :

from sklearn.model\_selection import train\_test\_split

xtrain, xtest, ytrain, ytest = train\_test\_split(X, y, train\_size=0.8)

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.

Pourquoi ne pas utiliser toutes les données pour l’entraînement et réutiliser ensuite toutes les données pour tester le modèle ?

En fait, il s’avère que si l'on entraîne le modèle avec des données, il va naturellement être plus performant sur ces données-là. Ce qui nous intéresse, c’est de mesurer sa performance sur des données qu’il n’a jamais vues, puisque c’est ce qui va se passer en pratique. Cette performance est appelée la **généralisation** du modèle : sa capacité à effectuer des prédictions de qualité sur des situations jamais rencontrées. On étudiera ça plus en détail dans la prochaine partie de ce cours. 😀

**En résumé**

On a maintenant une meilleure idée de la manière dont on peut correctement exploiter nos données :

* S’il y en a beaucoup, on travaille d'abord uniquement avec un **échantillon** représentatif de la population pour pouvoir aller plus vite.
* On sépare dès le départ en deux parties notre jeu de données : un **training set** pour créer le modèle et un **testing set** pour tester la qualité du modèle.

## Entraînez votre premier k-NN

Dans ce chapitre, nous allons utiliser un algorithme aussi répandu que la régression linéaire qui s'appelle le k-NN. On va s'en servir pour illustrer le traitement du jeu de données vu au chapitre précédent.

Le k-NN est le diminutif de***k Nearest Neighbors***. C’est un algorithme qui peut servir autant pour la classification que pour la régression. Il est surnommé « nearest neighbors » (plus proches voisins, en français) car le principe de ce modèle consiste en effet à choisir les **k** données les plus proches du point étudié afin d’en prédire sa valeur.

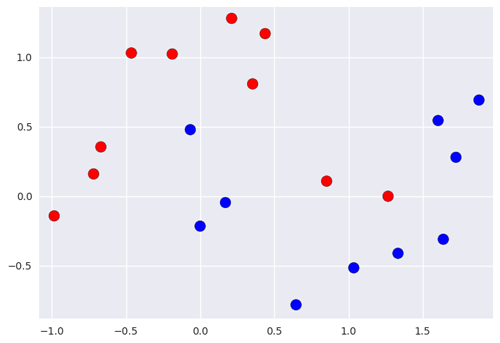
J’ai choisi volontairement cet algorithme car il est assez simple d’un point de vu conceptuel et illustre parfaitement les problématiques classiques qui en découlent. Mais attention, en pratique cet algorithme est assez peu utilisé dans sa forme première, car coûteux en puissance de calcul.

### Comment fonctionne le modèle k-NN ?

Avant d'utiliser un vrai jeu de données, on va prendre un petit exemple visuel afin de comprendre le fonctionnement de l'algorithme.

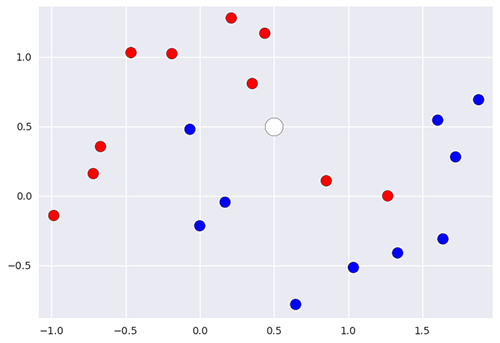
Ci-dessous j'ai représenté un jeu de données d’entraînement avec deux classes, rouge et bleu. L'input est donc bidimensionnel ici, et la target est la couleur à classer.

On a donc une dimension supplémentaire par rapport à l'exemple sur la régression linéaire (qui avait une seule dimension en entrée : la surface du logement).



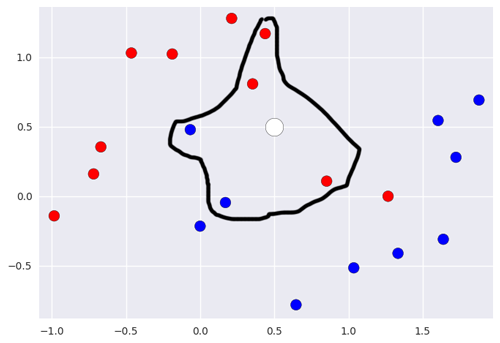
Notre nuage de points de test

Si l'on a une nouvelle entrée dont on veut prédire la classe, comment pourrait-on faire ?



Le point blanc est une nouvelle entrée.

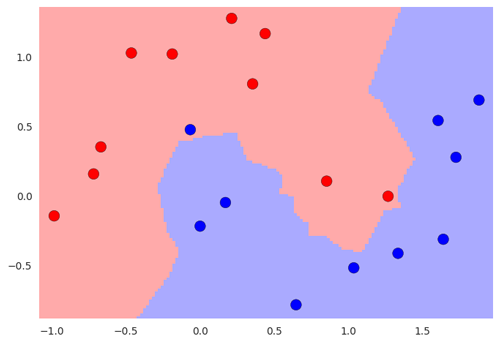
Et bien on va simplement regarder les **k** voisins les plus proches de ce point et regarder quelle classe constitue la majorité de ces points, afin d'en déduire la classe du nouveau point. Par exemple, ici, si on utilise le 5-NN, on peut prédire que la nouvelle donnée appartient à la classe **rouge** puisqu'elle a 3 rouges et 2 bleus dans son entourage.



Les 5 points les plus proches du point que l'on cherche à classer

On a utilisé ici comme mesure de similarité la distance euclidienne. On peut bien sûr utiliser d’autres normes.

On peut facilement en déduire les zone rouge et bleue, où les points qui se situeront dans la zone seront respectivement classés comme rouge ou bleu.



Les deux zones qui séparent l'espace pour la décision à prendre sur la classification de nouvelles entrées avec le modèle 5-NN

Voilà, ce n’est pas plus compliqué que ça !

### C'est un peu plus compliqué que ça…

Vous parliez, dans les chapitres précédents, de modèles statistiques auxquels on voudrait faire correspondre les données. C’est quoi le modèle ici ?

En fait, le k-NN est un type spécial d’algorithme qui n’utilise pas de modèle statistique. Il est "non paramétrique" et il se base uniquement sur les données d’entraînement. Ce type d’algorithme est appelé memory-based. A contrario, la régression linéaire est paramétrique, de paramètre **θ** et ne va donc pas avoir besoin de conserver toutes les données pour effectuer des prédictions, mais seulement **θ**.

C'est d'ailleurs un autre inconvénient de l'algorithme k-NN, il doit conserver toutes les données d'entraînement en mémoire (memory-based) et donc convient aux problèmes d'assez petite taille.

Il va s'agir maintenant de choisir le bon nombre de voisins k, en fonction de l’erreur de généralisation du modèle.

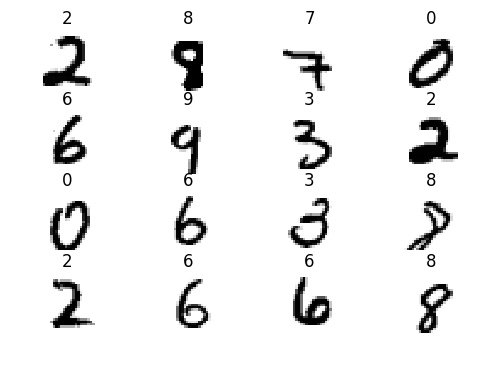
**k** n'est pas un paramètre mais un **hyperparamètre**, c'est-à-dire que contrairement aux paramètres classiques, il ne va pas pouvoir être appris automatiquement par l'algorithme à partir des données d'entraînement. Les hyperparamètres permettent de caractériser le modèle (complexité, rapidité de convergence, etc). Ce ne sont pas les données d'apprentissage qui vont permettre de trouver ces paramètres (en l'occurrence ici, le nombre de voisins **k**), mais bien à nous de l'optimiser à l'aide du jeu de données test.

### Utilisez k-NN sur un vrai jeu de données

OK, on peut maintenant passer aux choses sérieuses !

#### Les données et la problématique

D'abord, parlons du jeu de données que nous allons utiliser. C'est un dataset très célèbre, appelé MNIST. Il est constitué d'un ensemble de 70000 images 28x28 pixels en noir et blanc annotées du chiffre correspondant (entre 0 et 9). L'objectif de ce jeu de données était de permettre à un ordinateur d'apprendre à reconnaître des nombres manuscrits automatiquement (pour lire des chèques par exemple). Ce dataset utilise des données réelles qui ont déjà été pré-traitées pour être plus facilement utilisables par un algorithme.



Un extrait du type d'images que l'on trouve dans le dataset MNIST

Notre objectif sera donc d'entraîner un modèle qui sera capable de reconnaître les chiffres écrits sur ce type d'images. Par chance, ce jeu de données est téléchargeable directement à partir d'une fonction scikit-learn. 😇 On peut donc directement obtenir ce dataset via un appel de fonction :

from sklearn.datasets import fetch\_openml

mnist = fetch\_openml('mnist\_784', version=1)

Le dataset peut prendre un certain temps à se charger, soyez patients.

Si le téléchargement des données échoue, vous pouvez télécharger à la main le fichier   mnist-original.mat  et le placer dans un dossier qui s'appelle  mldata  . Ensuite, demandez à sklearn de le charger depuis un emplacement local avec  mnist = fetch\_mldata('MNIST original', data\_home=location)  , où location est l'adresse du répertoire où est placé le répertoire mldata.

L'objet  mnist contient deux entrées principales,  data  et  target. On peut les afficher :

# Le dataset principal qui contient toutes les images

print (mnist.data.shape)

# Le vecteur d'annotations associé au dataset (nombre entre 0 et 9)

print (mnist.target.shape)

(70000, 784)

(70000,)

* data contient les images sous forme de tableaux de 28 x 28 = 784 couleurs de pixel en niveau de gris, c'est-à-dire que la couleur de chaque pixel est représentée par un nombre entre 0 et 16 qui représente si celle-ci est proche du noir ou pas (0 = blanc, 16 = noir).
* target qui contient les annotations (de 1 à 9) correspondant à la valeur "lue" du chiffre.

On dit ici que le nombre de features (ou dimensions) en entrée est de 28 x 28 x 1 = 784. Dans le cas où l'on aurait utilisé des images couleur et pas en niveaux de gris, on serait passé à 3 composantes couleurs par pixel (rouge, vert, bleu) et donc le nombre de features aurait été : 28 x 28 x 3 = 2352.

#### Échantillonner pour faciliter le travail

Le dataset est relativement petit mais, pour le modèle k-NN, il est déjà trop gros pour obtenir rapidement des résultats. On va donc effectuer un sampling et travailler sur seulement 5000 données :

sample = np.random.randint(70000, size=5000)

data = mnist.data[sample]

target = mnist.target[sample]

Cette manière de "sampler" le jeu de données est dans l'absolu inexacte. En effet, le tableau d'indices générés par  randint  peut contenir des doublons, ce qui nous conduit à avoir des doublons dans nos données. Comme nous ne prenons que 5000 échantillons sur 70000, le risque est plutôt faible. Pour faire ce travail de manière rigoureuse, regardez la fonction  sklearn.utils.resample.

#### Séparez training / testing set

Une fois notre dataset chargé, comme nous l'avons vu dans le chapitre précédent, nous allons séparer le jeu de données en training set et testing set.

J'ai ici appelé les images d'exemple "X" et les annotations cibles "y" :

from sklearn.model\_selection import train\_test\_split

xtrain, xtest, ytrain, ytest = train\_test\_split(data, target, train\_size=0.8)

Je le rappelle, on va utiliser uniquement le training set pour entraîner notre modèle et on garde le testing set pour plus tard. J'ai mis la répartition classique 80/20 entre training et testing set.

#### Le k-NN

On peut créer un premier classifieur 3-NN, c'est-à-dire qui prend en compte les 3 plus proches voisins pour la classification. Pour cela, on va utiliser l'implémentation de l'algorithme qui existe dans la librairie scikit-learn :

from sklearn import neighbors

knn = neighbors.KNeighborsClassifier(n\_neighbors=3)

knn.fit(xtrain, ytrain)

Comme je l'ai dit plus haut pour le k-NN, l'algorithme ici n'effectue aucune optimisation mais va juste sauvegarder toutes les données en mémoire. C'est sa manière d'apprendre en quelque sorte.

Testons à présent l’erreur de notre classifieur. La méthode  score  effectue exactement ça : tester les performances de prédiction d'un classifieur dans lequel on passe un jeu de données annoté — dans notre cas le jeu de données de test. Il renvoie ainsi le pourcentage de prédiction véridique trouvée par le classifieur.

error = 1 - knn.score(xtest, ytest)

print('Erreur: %f' % error)

qui nous donne en sortie :

Erreur: 0.06999999999999995

Pas mal déjà !

La performance exacte de notre modèle dépend du jeu de sélection. N'hésitez pas à lancer plusieurs fois ce traitement en changeant les jeux de données d'apprentissage et de test pour vous en rendre compte.

Comme je l'ai dit, le k (nombre de voisins) est l'hyper-paramètre que l’on va chercher à optimiser pour minimiser l’erreur sur les données test.

#### Optimisation du score sur les données test

Pour trouver le k optimal, on va simplement tester le modèle pour tous les k de 2 à 15, mesurer l’erreur test et afficher la performance en fonction de k :

errors = []

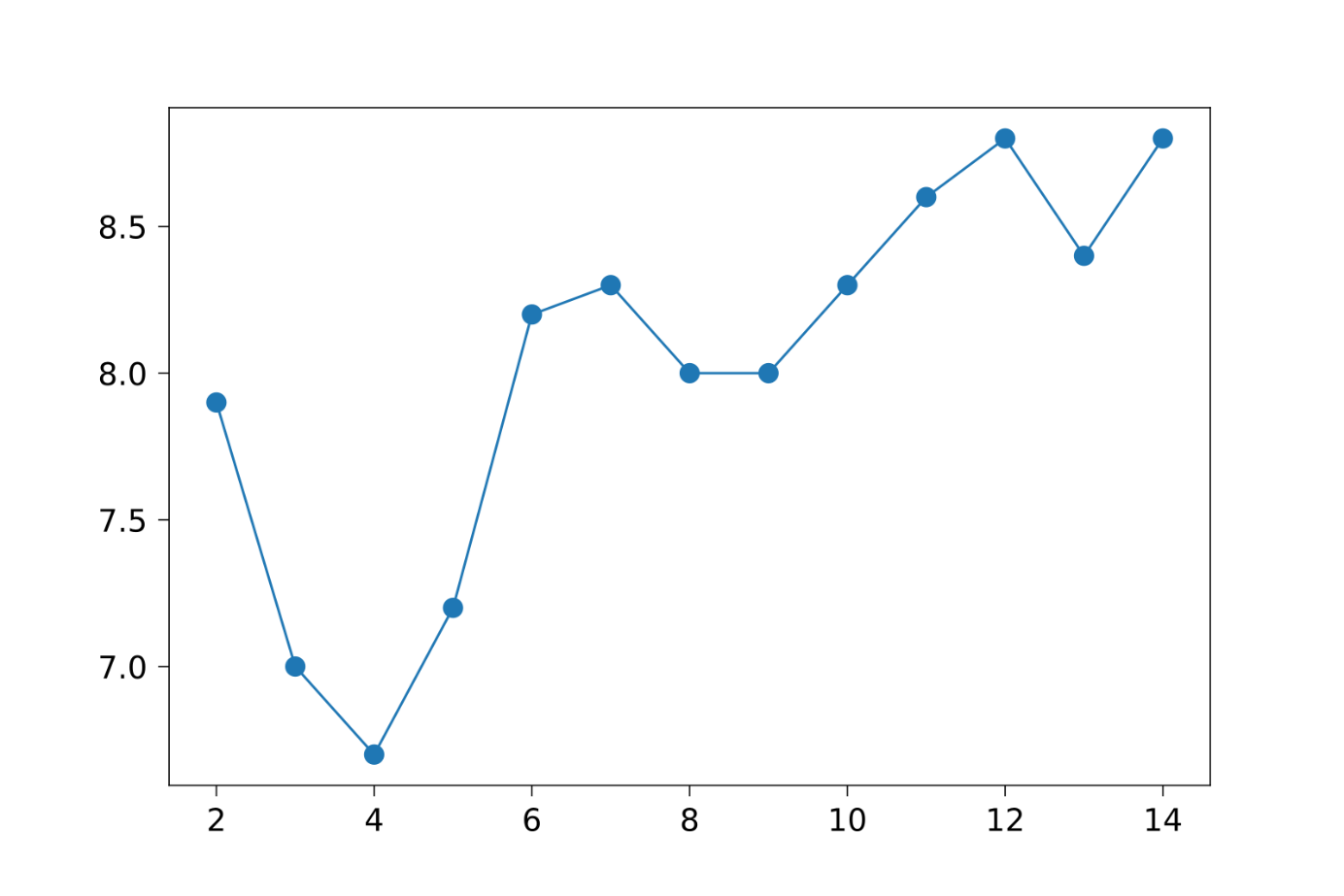
for k in range(2,15):

knn = neighbors.KNeighborsClassifier(k)

errors.append(100\*(1 - knn.fit(xtrain, ytrain).score(xtest, ytest)))

plt.plot(range(2,15), errors, 'o-')

plt.show()



L'erreur en pourcentage pour les différents classifieurs

Comme on peut le voir, le k-NN le plus performant est celui pour lequel k = 4. On connaît donc notre classifieur final optimal : 4-nn. Ce qui veut dire que c'est celui qui classifie le mieux les données, et qui donc dans ce cas précis reconnaît au mieux les nombres écrits à la main.

À titre d'exemple, vous pouvez afficher les prédictions du classifieur sur quelques données.

# On récupère le classifieur le plus performant

knn = neighbors.KNeighborsClassifier(4)

knn.fit(xtrain, ytrain)

# On récupère les prédictions sur les données test

predicted = knn.predict(xtest)

# On redimensionne les données sous forme d'images

images = xtest.reshape((-1, 28, 28))

# On selectionne un echantillon de 12 images au hasard

select = np.random.randint(images.shape[0], size=12)

# On affiche les images avec la prédiction associée

fig,ax = plt.subplots(3,4)

for index, value in enumerate(select):

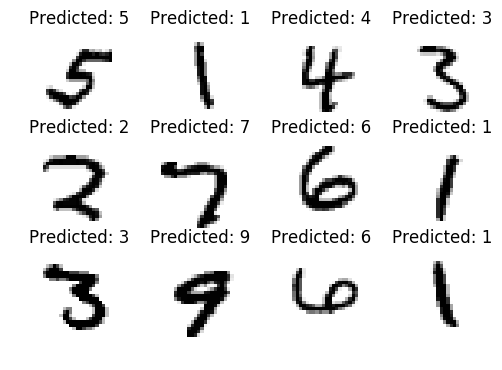
plt.subplot(3,4,index+1)

plt.axis('off')

plt.imshow(images[value],cmap=plt.cm.gray\_r,interpolation="nearest")

plt.title('Predicted: {}'.format( predicted[value]) )

plt.show()



Les prédictions du classifieur associées aux images

Pour pouvoir un peu mieux comprendre les erreurs effectuées par le classifieur, on peut aussi afficher un extrait des prédictions erronées :

# on récupère les données mal prédites

misclass = (ytest != predicted)

misclass\_images = images[misclass,:,:]

misclass\_predicted = predicted[misclass]

# on sélectionne un échantillon de ces images

select = np.random.randint(misclass\_images.shape[0], size=12)

# on affiche les images et les prédictions (erronées) associées à ces images

for index, value in enumerate(select):

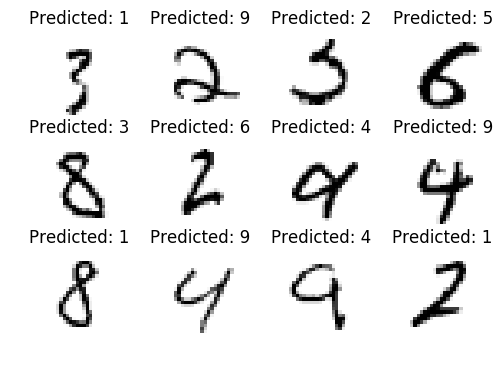
plt.subplot(3,4,index+1)

plt.axis('off')

plt.imshow(misclass\_images[value],cmap=plt.cm.gray\_r,interpolation="nearest")

plt.title('Predicted: {}'.format(misclass\_predicted[value]) )

plt.show()



Quelques exemples d'images mal classées

### En résumé

On vient de voir sur un algorithme simple comment l'entraîner sur des données pour ensuite optimiser les paramètres de cet algorithme à l’aide d’un jeu de données test.

N’hésitez pas à entraîner ce modèle à votre tour sur d’autres jeux de données !

## Entraînez-vous à entraîner un algorithme de Machine Learning !

### À vous de jouer !

Pour vous entraîner, réalisez cet exercice étape par étape. Une fois terminé, vous pouvez comparer votre travail avec les pistes que je vous propose.

Vous êtes consulté par une agence immobilière pour prédire les loyers des différents arrondissements de Paris, afin de les aider à prendre des décisions d'achat d'appartements.

Pour ce faire, vous allez améliorer le modèle de prédiction de loyer étudié dans un chapitre précédent, à l'aide d'une feature (variable) supplémentaire en entrée, pour obtenir un modèle plus performant.

Vous pouvez télécharger le **[dataset](https://static.oc-static.com/prod/courses/files/initiez-vous-au-machine-learning/house_data.csv" \t "_blank)** de l'activité. Il possède une nouvelle variable : l'arrondissement de l'appartement.

### Votre mission

Vous allez reprendre le code que vous avez appris à faire dans le chapitre [Programmez votre première régression linéaire](https://openclassrooms.com/fr/courses/4011851/parts/4121986/edit) et tester plusieurs manières d'améliorer la modélisation à l'aide de cette nouvelle observation (= ce nouveau dataset).

Vous allez effectuer :

* une **séparation en training / testing set**;
* **deux propositions d'amélioration** du modèle qui obtiennent de meilleures performances que la "baseline" (la régression linéaire avec une seule feature) ;
* une **sélection d'un modèle final** à partir des performances.

Pour traiter la variable supplémentaire arrondissement, réfléchissez bien à la nature de cette variable et pensez à la traiter comme une variable catégorielle.

Vous pourrez, par exemple, vous inspirer de cette [approche proposée sur stackoverflow](https://stackoverflow.com/questions/34007308/linear-regression-analysis-with-string-categorical-features-variables).

À vos claviers !

### Livrable

Un fichier python ou un notebook avec l'ensemble des tests effectués, ainsi que les outputs (images & graphes).

Pensez à bien inclure des explications sur les points d'attention dans les commentaires, ainsi que vos choix d'implémentation. Savoir communiquer ses résultats de façon claire fait partie intégrante du travail de data scientist, ne le négligez pas.

### Vérifiez votre travail

Voici un [**exemple**](https://s3-eu-west-1.amazonaws.com/course.oc-static.com/courses/4011851/allez-plus-loin-dans-la-prediction-de-loyer_exemple-2019-01-03T144839.zip) pour vous permettre de vérifier votre travail.

## Familiarisez-vous avec les limites des algorithmes

Vous vous en doutiez, la pratique de la data science est longue et laborieuse, car semée d'embûches. Mais j'espère que cela ne va pas vous arrêter pour autant ! Et comprendre dès maintenant les difficultés classiques, que vous allez forcément rencontrer si vous continuez à utiliser des techniques de machine learning, vous aidera à mieux aborder les problèmes qui surviendront.

Dans ce chapitre, je vous présente deux limites principales du machine learning :

* une limite **théorique** sur la capacité d'un algorithme à résoudre différentes tâches ;
* une limite **pratique** sur la capacité des ordinateurs à gérer la complexité des problèmes à traiter.

### Le théorème "No Free Lunch"

Le théorème du "No Free Lunch" (qu'on pourrait traduire par : "pas de déjeuner gratuit") est la raison pour laquelle on va encore avoir besoin des data scientists pour un bon bout de temps ! 😀

En essence, ce théorème statue qu'aucun modèle ou algorithme ne fonctionne bien pour tous les problèmes. En d'autres termes, si un algorithme de machine learning fonctionne bien sur un type de problème particulier, ça veut dire qu'il le paiera ailleurs et sera donc moins performant en moyenne sur le reste des problèmes.

En effet, un modèle étant une**approximation de la réalité**, il repose sur un certain nombre **d'hypothèses** de départ pour exister. Ces hypothèses sont **dépendantes du contexte** (c.-à-d. du problème considéré). Les hypothèses étant différentes pour chaque type de problème, on doit considérer différents modèles pour différents problèmes.

C'est un peu une prise de tête inutile, non ?

Peut-être un peu, mais c'est juste pour vous rappeler que personne ne trouvera un modèle "ultime" qui résoudra tout avec efficacité. Vous devrez bien toujours tester plusieurs modèles pour chaque problématique, afin de trouver à chaque fois le modèle optimal en termes de précision, temps de calcul et complexité algorithmique. Il n'y aura jamais un algorithme magique (une golden rule) qui fonctionnera mieux que les autres.

Ce qu'il faut également retenir de cette règle, c'est que vous allez construire un modèle efficace basé sur vos hypothèses de départ. **Il faut donc que ces hypothèses servent votre problématique.**

#### Exemple

Examinons de plus près les moteurs de recommandation de type "collaborative filtering" que j'ai présentés dans un chapitre précédent. L'hypothèse principale est qu'un utilisateur qui aime un film va aimer un film similaire. C'est cette **supposition préalable** qui est à la base de cet algorithme. Imaginez un monde où vous avez des goûts répartis de manière aléatoire entre les différents films, sans lien entre eux... les recommandations ne fonctionneront pas ! Cet algorithme est donc utile et performant uniquement dans le cas où cette hypothèse est vérifiée.

Un conseil à retenir : **votre travail est d'être attentif aux hypothèses sur lesquelles vous construisez votre modèle.** Ce n'est pas quelque chose qu'on peut vraiment apprendre, c'est une compétence qu'on développe avec l'expérience et la connaissance des fonctionnements des algorithmes qu'on utilise.

### Les problèmes insolubles

Appelée intractability en anglais, cette notion désigne les problèmes qui ne peuvent être résolus **dans un temps raisonnable**, qui explosent en termes de complexité algorithmique.

Comme vous le savez maintenant, grand nombre d'algorithmes de machine learning cherchent en fait à **converger** vers l'optimum d'une fonction (maximum de vraisemblance, minimum d'erreur empirique, etc). Or, il est très fréquent que ce type de fonctions soit très difficile à calculer de manière exacte, dès lors qu'on a un grand nombre de variables à traiter. Les modèles de représentation utilisés dans les algorithmes sont approximés, ou alors on renforce/relaxe des hypothèses de départ afin de s'assurer que l'algorithme va bien trouver une solution.

Pour ceux qui ont fait un peu d'algorithmique, on parle ici des classes de problèmes qui terminent dans un temps fini en théorie, mais raisonnablement beaucoup trop long pour être considéré en pratique. En machine learning, on retrouve en majorité des problèmes de classe EXPTIME, c'est-à-dire la classe des problèmes décidés en **temps exponentiel** par une machine déterministe.

Merci, mais à quoi ça va me servir en pratique ?

Le premier intérêt, c'est que quand vous verrez dans une documentation d'algorithme le terme intractable, vous saurez ce que ça signifie. 😉

Le second, c'est que vous savez à présent que bon nombre d'algorithmes ne convergent pas exactement vers la solution optimale pour cette raison. Vous pouvez donc comprendre pourquoi on utilise beaucoup d'approximations en machine learning et pourquoi on va relaxer certaines hypothèses pour éviter cette complexité supplémentaire, non nécessaire en pratique.

Enfin, même si les développeurs des différentes librairies de machine learning que vous utiliserez ont fait le maximum pour créer des outils performants et efficaces dans la plupart des circonstances, il arrivera que vos algorithmes soient malheureusement trop complexes pour être raisonnablement intéressants. Une première solution quand on est confronté à un algorithme qui met trop de temps à converger, c'est de simplifier le problème au maximum en minimisant la perte d'information. On peut par exemple chercher à :

* réduire le **nombre** de variables en entrée ;
* réduire la **taille** du dataset au maximum ;
* réduire la **complexité** des hypothèses de modélisation.

### En résumé

Dans ce chapitre, l'objectif était d'encadrer les problèmes posés lors de la construction d'un modèle par un data scientist, en énonçant ces deux limites à garder en tête :

* **Il n'existe pas d'algorithme et modèle "ultime"**, applicable pour tous les problèmes. Vous devez donc aborder chaque nouveau problème avec un œil neuf et veiller à tester plusieurs algorithmes afin de le résoudre, en formulant des hypothèses spécifiques à ce problème.
* Il arrive souvent de se trouver confronté à des algorithmes et modèles très puissants, mais trop compliqués pour être utilisés directement. Pour éviter ces situations **d'intractability**, il ne faut pas avoir peur d'effectuer des approximations qui permettent de gagner en efficacité.

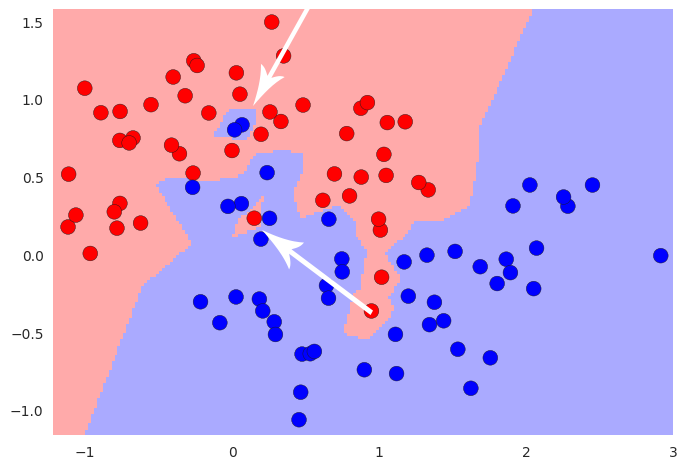
## Trouvez le bon compromis entre biais et variance

Après avoir démystifié certains aspects du travail de modélisation dans les précédents chapitres, je vais ici vous présenter un des plus gros points d'attention du data scientist lors de l'entraînement d'un modèle de machine learning : le compromis entre **biais** et **variance**.

### Tout est affaire d'équilibre !

Pour mieux comprendre ce problème et en développer une intuition visuelle, reprenons le modèle k-NN que nous avons utilisé dans le TP de la partie précédente.

Que se passe-t-il lorsqu'on utilise le 1-NN ? C'est-à-dire que l'on ne prend en considération qu'un seul voisin pour prédire la classe d'un point ?

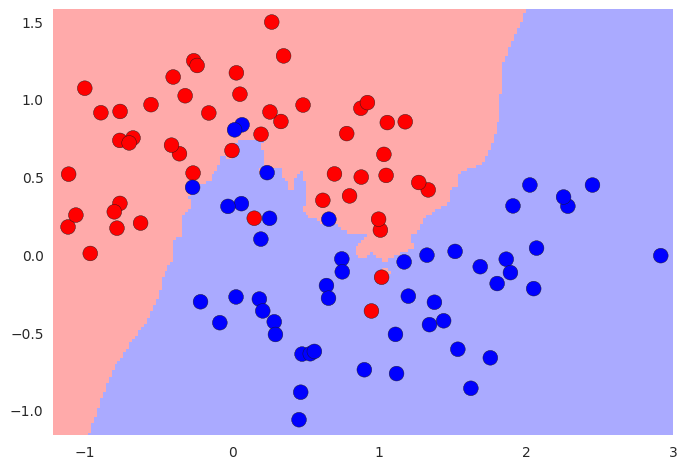


Les zones désignées par les flèches ne vont probablement pas être bien classées...

Comme on peut le voir, le classifieur considéré "colle" trop aux données d'entraînement. En d'autres termes, changer même très légèrement le training set, en y ajoutant un point par exemple, aurait déjà une conséquence non-négligeable.

On voit donc ici une **dépendance** très forte au training set, autrement dit une variation très forte de la décision en fonction des données d'entraînement. Cette variabilité est appelée, à juste titre, **variance** du modèle. 😉

En réalité, on souhaite plutôt obtenir quelque chose de plus "stable", qui ne va pas autant dépendre de notre sélection des données d'entraînement. Pour réduire cette dépendance, on peut dans notre cas utiliser une moyenne de plusieurs voisins, c'est-à-dire augmenter le "k" du k-NN.



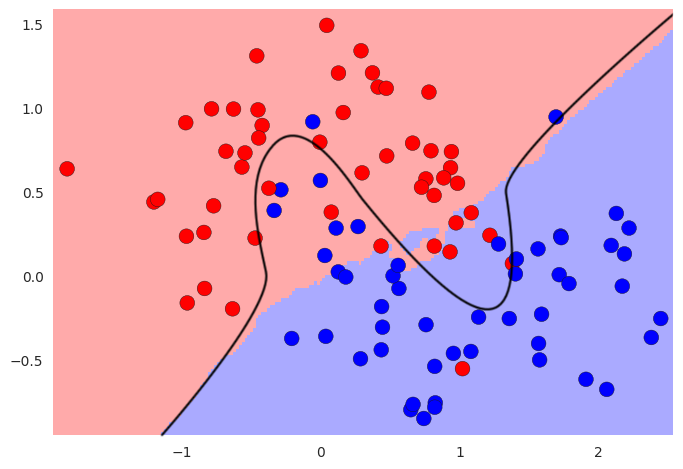
Le classifieur 5-nn sur le même jeu de données est beaucoup plus efficace.

Top ! On est déjà moins dépendant du training set dans ce cas. Augmenter le nombre k de voisins sur lesquels on effectue une moyenne pour prédire la classe d'appartenance permet donc de "lisser" notre prédiction, de la rendre moins dépendante des variations du training set et donc, comme on l'a dit, de **diminuer la variance**.

Je me rappelle que dans le TD, l'erreur du modèle augmentait énormément quand on augmentait le nombre de voisins k, non ?

Oui, effectivement ! Augmenter le k ne va malheureusement pas tout résoudre, car on crée par la même occasion un autre type d'erreur, qui s'appelle le **biais**.

Le biais correspond en quelque sorte à quel point on vise à côté de la "vraie" valeur d'un point considéré. Pour revenir à notre exemple, si je considère à présent le classifieur 10-NN, la frontière de décision, bien que "lisse", devient très éloignée de la zone "idéale".



Parce qu'on a choisi un k trop grand, la zone de classification (frontière rouge/bleue) est trop lisse par rapport à la complexité du modèle en noir, plus proche de la réalité.

Le problème du biais est lié à la complexité du modèle choisi. Comme on peut le voir, si on prend un modèle trop simple par rapport au phénomène que l'on veut modéliser, on va avoir un biais élevé. En prenant un modèle plus complexe, on va **diminuer le biais**.

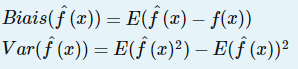
Cette balance à trouver entre la variance trop forte et le biais trop élevé est appelé **compromis biais-variance**, ou ***bias-variance tradeoff***en anglais.

### Quel est le lien avec l'erreur d'un modèle ?

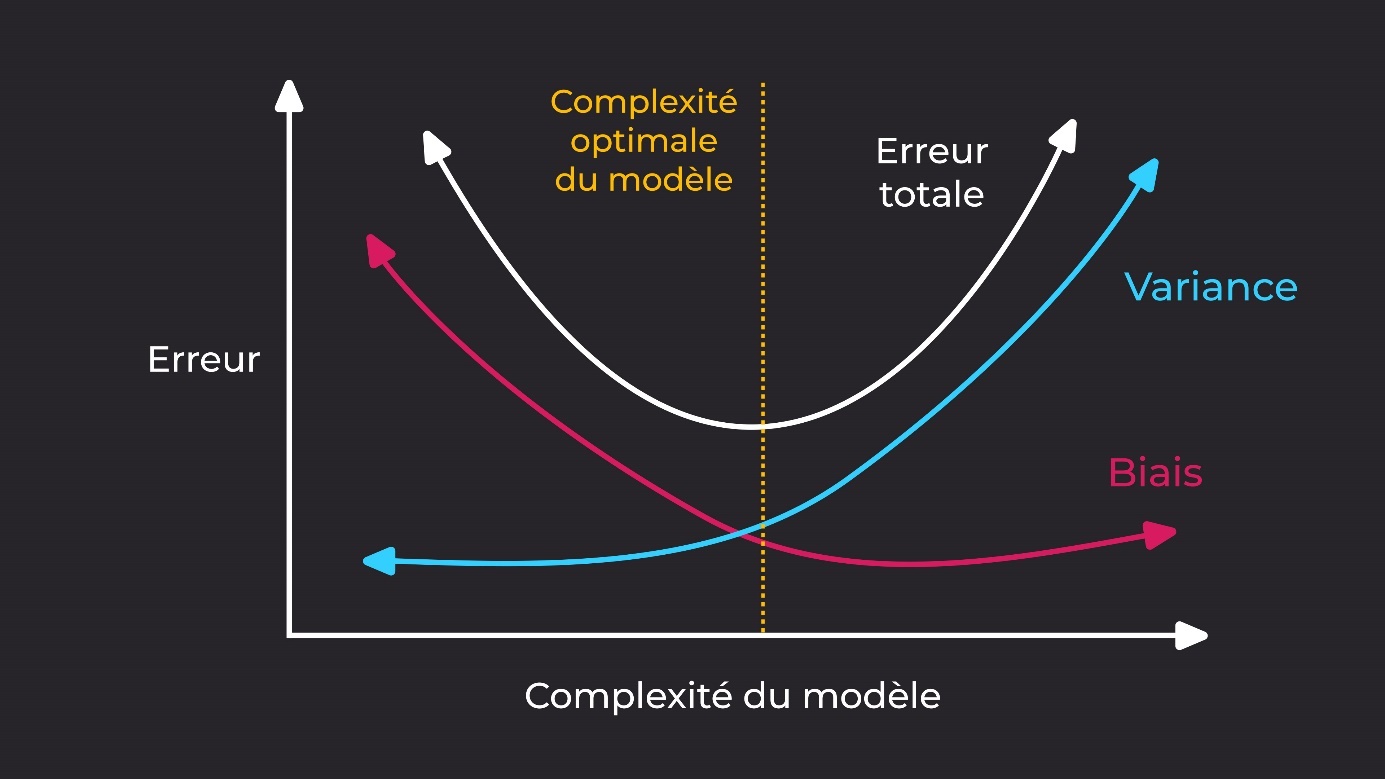
On peut décomposer l'erreur de prédiction en  de la manière suivante :



Pour rappel, en probabilité, le **biais** et la **variance** d'un modèle  sur une variable x :



Tout le jeu va donc être de minimiser cette erreur, en prenant en compte les comportements opposés du **biais** et de la **variance** qui dépendent de la complexité du modèle :



On cherche à se placer au minimum de l'erreur totale.

Le juste équilibre entre biais et variance se trouve pile au minimum de l'erreur totale, qui est une combinaison du biais et de la variance.

### Comment trouver le bon compromis ?

#### Réduire les dimensions

Quand on se trouve dans le cas d'une variance excessive (trop forte dépendance au data set d'entraînement), réduire le nombre de dimensions du modèle (c.-à-d. le nombre de variables en entrée) permet de diminuer la variance en simplifiant la complexité du modèle.

#### Sélectionner et entraîner le bon modèle

Il existe plusieurs **méthodes de sélection de modèle** qui permettent de trouver la complexité optimale pour faire la balance entre le biais et la variance. La manière **d'entraîner le modèle** est primordiale aussi et il existe des méthodes pour minimiser la prise en compte de variances non-représentatives du modèle présentes dans le dataset.

Un peu de patience, nous étudierons ces méthodes d'évaluation et d'entraînement de modèles dans un prochain cours...

#### Utiliser des méthodes ensemblistes

Il faut aussi savoir qu'il existe toute une famille d'algorithmes appelés **les méthodes d'ensembles** qui se basent sur la combinaison de plusieurs modèles à haute variance et les agrègent (p. ex. en les moyennant) pour réduire la variance finale.

### En résumé

* Le principe de **compromis entre biais et variance** est une des problématiques à laquelle vous serez confronté lors de votre travail quotidien !
* En utilisant un modèle comportant une **trop grande complexité**– dit "**à haute variance**" – on peut mal capturer le phénomène sous-jacent et devenir trop dépendant aux données d'entraînement et aux petites fluctuations aléatoires, non représentatives du phénomène.
* A contrario, il ne faut pas choisir un modèle **trop "simple"** qui biaise le résultat et ne parvient pas à capturer toute la complexité du phénomène.

**Généralisez votre modèle**

Rappelez-vous, j'ai déjà parlé de la **généralisation** du modèle quand j'ai abordé la séparation entre training et testing set. En effet, la généralisation du modèle représente sa capacité, une fois entraîné, à effectuer des prédictions sur des données qu'il n'a jamais vues ; d'où la création d'un testing set pour évaluer cette capacité.

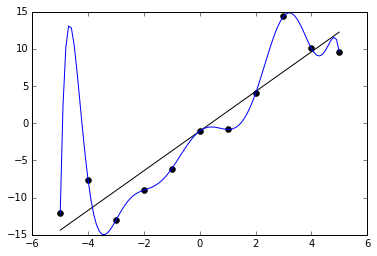
La **généralisation** désigne la capacité du modèle à pouvoir effectuer des prédictions robustes sur des nouvelles données.

En réalité, c'est un aspect du machine learning assez complexe... mais c'est aussi le plus intéressant ! Puisque quand on parle d'*apprentissage* de l'algorithme, on vise essentiellement sa capacité à généraliser, extrapoler les connaissances, et non pas juste à reproduire et ressortir des choses déjà vues.

**L'overfitting et l'underfitting**

La notion d'**overfitting** (quelquefois traduite par **surapprentissage** en français) désigne le fait que le modèle que vous avez choisi est trop collé aux données d'entraînement. C'est un problème classique de data science, lorsqu'on choisi un modèle trop "flexible", c'est-à-dire avec une complexité trop élevée qui prend aussi en compte le **bruit** du phénomène. C'est en fait ce qui arrive aux méthodes à haute variance dont je parlais dans le chapitre précédent.

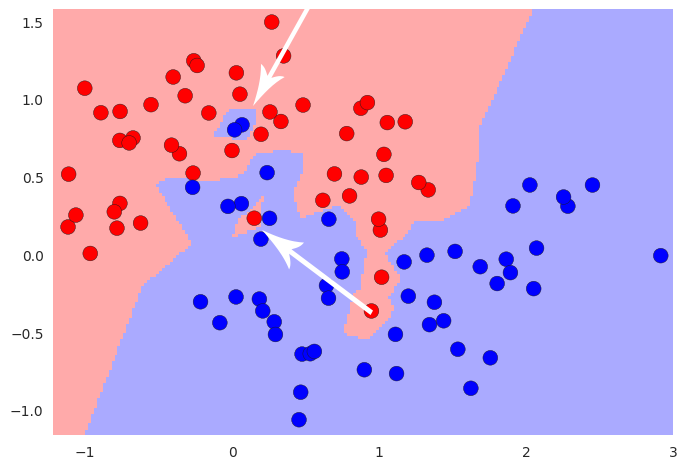
Regardez le graphique ci-dessous :



Comparaison entre un modèle en overfitting (bleu) et un modèle plus réaliste du phénomène (droite noire)

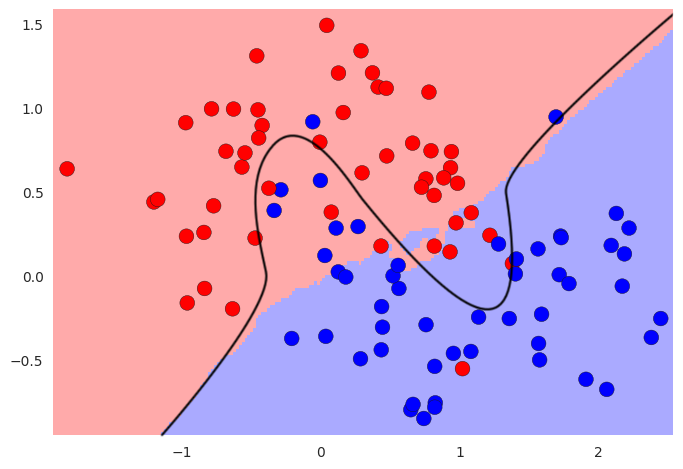
Comme on peut s'en douter, on préférerait évidemment travailler sur un modèle qui ressemble à la droite noire, contrairement à la courbe bleue qui pourtant passe par tous les points. Si on est amené à prédire la valeur pour un nouveau point, on sait que celui-ci a plus de chance de se trouver proche de la droite noire que proche de la courbe bleue (surtout à gauche 😜).

Un autre bon exemple du problème d'overfitting est le 1-NN que nous avons vu précédemment. J'ai désigné avec des flèches blanches les endroits où on *overfit* les données d'entraînement.



Un exemple classique d'overfitting

A contrario, l'**underfitting** (ou **sous-apprentissage**) désigne une situation où le modèle n'est pas du tout assez complexe pour capturer le phénomène dans son intégralité.



Un exemple d'underfitting comparé avec la zone de séparation idéale en noir

Donc pour faire le lien avec le chapitre précédent, les méthodes à haute variance ont tendance à *overfitter facilement* sur du bruit non représentatif du modèle sous-jacent. À l'inverse, les algorithmes avec un biais élevé ont tendance à *underfitter plus facilement* les données d'entraînement, et donc à rater des informations importantes sur le phénomène.

Lorsqu'on veut généraliser un modèle, on a besoin qu'il n'*overfit* pas et qu'il n'*underfit* pas, qu'il soit pile entre les deux.

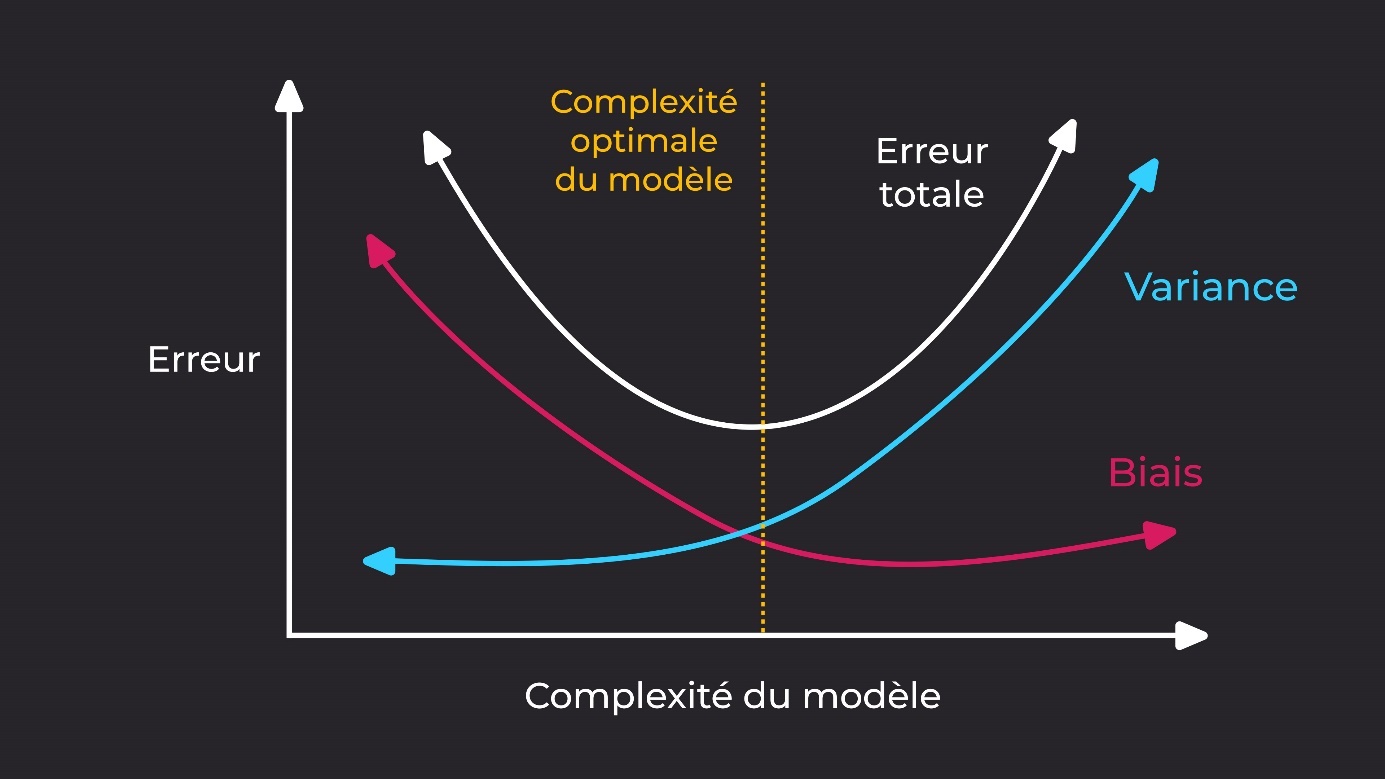
On considère qu'un algorithme est particulièrement puissant lorsqu'il possède cette capacité de généralisation, et qu'il peut effectuer les prédictions les plus performantes possibles avec le moins de données possibles.

**Comment prendre en compte ce problème ?**

Évidemment, les mêmes méthodes que celles citées dans le précédent chapitre pour le compromis biais-variance restent valables ici :

* bien sélectionner le modèle de départ ;
* bien choisir la manière d'entraîner le modèle ;
* utiliser des méthodes ensemblistes.

Mais, rappelez-vous également, dans le chapitre précédent, j'avais utilisé ce graphique :

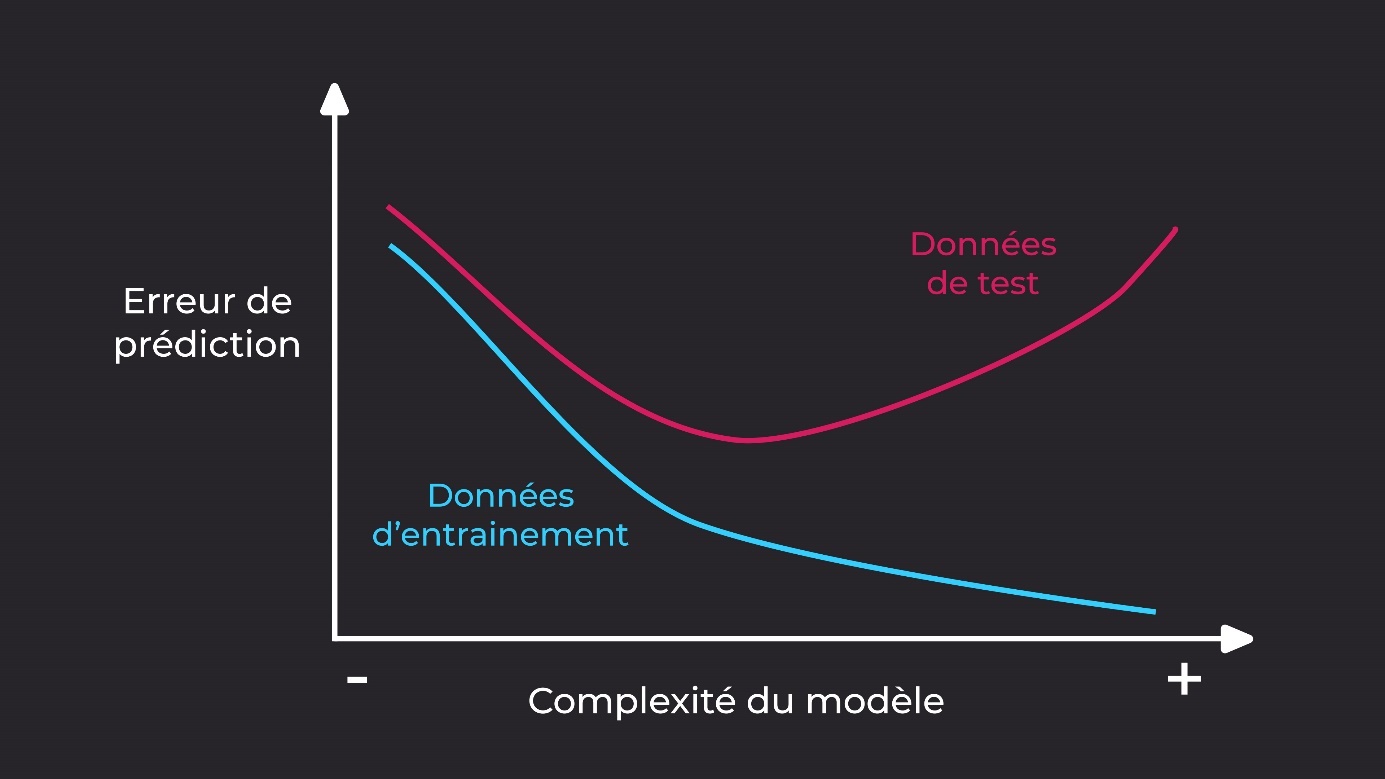


Le compromis entre biais et variance

En réalité, ce graphique n'est pas exact !

Quoi ? Ce n'est pas vrai !?

Non, pas vraiment, mais la courbe d'erreur totale n'est valable *que* sur le **testing set**. En réalité, si l'on mesure l'erreur sur le **training set**, on obtient en fait la courbe bleue suivante :



Lorsque la complexité du modèle augmente, l'erreur totale a tendance à diminuer sur les données d'entraînement, contrairement aux données test.

Augmenter la complexité du modèle va en fait diminuer l'erreur sur le training set (logique, on va coller de plus en plus aux données du training set). Si on ne se basait que sur cette erreur, on n'aurait qu'à prendre le modèle le plus complexe possible pour diminuer l'erreur de prédiction ! Et on risquerait alors de se retrouver en situation d'overfitting lors de la généralisation du modèle. Cela prouve encore une fois l'importance de bien séparer son testing set du reste des données, sous peine de ne pas pouvoir **généraliser** son modèle.

C'est donc une première solution pour assurer la capacité de généralisation du modèle.

Encore une fois, il faut donc être très vigilant à bien séparer un jeu de données test lors de l'entraînement du modèle, afin d'éviter ensuite l'overfitting.

**En résumé**

* L'équilibre entre over- et underfitting est encore une composante à prendre en compte lors du choix et lors de l'entraînement du modèle.
* L'over- ou l'underfitting se détecte grâce aux données test qu'on peut utiliser pour mesurer l'erreur du modèle, contrairement aux données d'entraînement qui pervertissent cette mesure.

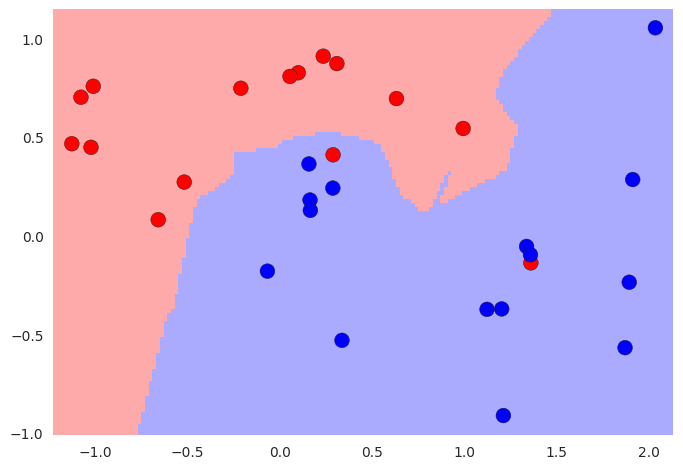
## Gérez le fléau de la dimension

Une dernière difficulté dont je veux que vous ayez conscience avant d'aborder votre prochain problème de machine learning est le **fléau de la dimension**, ou **curse of dimensionality** en anglais.

Cette expression désigne une problématique majeure qui apparaît souvent en data science, plus précisément lorsqu'on se retrouve avec un grand nombre de **features**relativement au nombre d'observations.

### Qu'est-ce que le fléau de la dimension ?

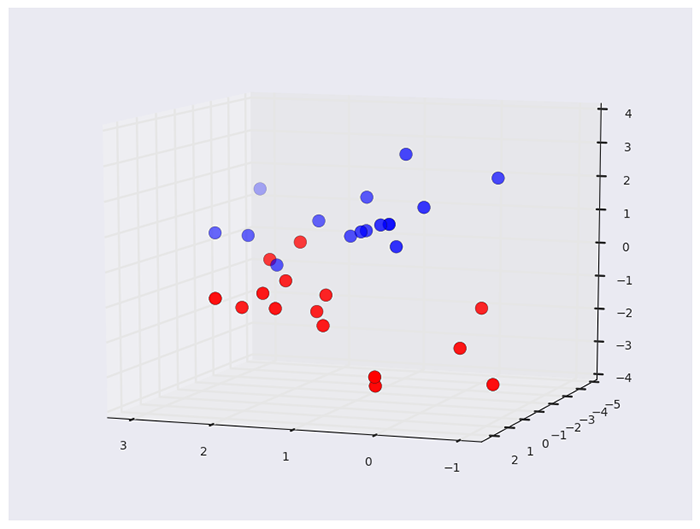
À nouveau, reprenons notre exemple maintenant bien connu du modèle 5-NN pour développer une intuition de ce problème.



Modèle 5-NN appliqué à un jeu de données

En fait, on a ici deux features en entrée qui permettent de prédire la classe rouge ou bleue : la position (x, y), déterminée par les coordonnées x et y sur le graphe.

Si on passe sur un exemple à 3 features, avec **le même nombre de données d'entraînement**, on obtient par exemple le nuage tridimensionnel ci-dessous :



Jeu de données en 3 dimensions

Si on essaye à nouveau d'effectuer une classification avec un 5-NN sur ce jeu de données, on remarque instinctivement qu'il y a moins de points d'exemple disponibles pour effectuer la prédiction relativement à la taille de l'espace à couvrir. L'algorithme devra donc aller chercher plus loin les 5 plus proches voisins pour effectuer la prédiction.

En fait, si on augmente encore le nombre de **features**, il devient de plus en plus difficile d'avoir **assez de données d'entraînement** aux alentours pour pouvoir effectuer une prédiction correcte.

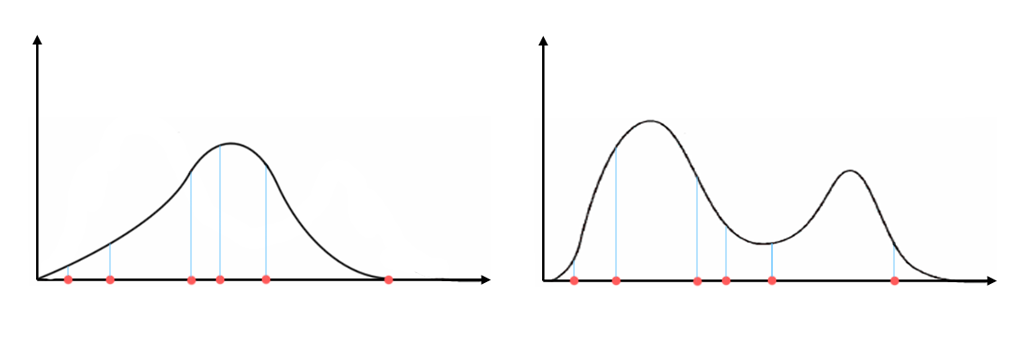
En pratique, il est assez courant d'avoir plusieurs centaines, voire plusieurs milliers de features, donc on imagine facilement que c'est difficile d'avoir assez d'exemples représentatifs de tout l'ensemble…

**Le manque de données nécessaires à l'apprentissage du modèle explose très vite** lorsqu'on augmente la dimension des données en entrée, c'est pourquoi on surnomme ce phénomène ***curse of dimensionality***.

Pour visualiser ce phénomène autrement, imaginez que vous devez traiter un problème qui possède N features et chacune peut posséder 10 valeurs différentes. Notre algorithme d'apprentissage doit pouvoir distinguer parmi  configurations différentes. On voit rapidement que plus N grandit (la dimension des données), plus un algorithme aura besoin d'exemples pour pouvoir apprendre correctement le phénomène. C'est pour cette raison qu'on dit qu'augmenter la dimension des données d'entrée est un fléau pour le machine learning. 💀

En réalité, le problème ne concerne pas exactement la **dimension**, même si c'est plus facile intuitivement à comprendre exprimé comme cela, mais plutôt la **complexité** du phénomène.

En effet, s'il y a beaucoup de features en entrée qui peuvent prendre beaucoup de valeurs différentes, mais qu'en réalité le comportement est facilement généralisable, alors on a besoin de moins d'exemples d'entraînement, puisqu'on peut déduire ce qu'il se passe entre les exemples par extrapolation. Il nous faut quand même un algorithme assez intelligent pour comprendre le modèle et effectuer ces généralisations. 😉



Exemple de complexité plus ou moins importante

Sur l'exemple ci-dessus, à partir des données d'exemple désignées par des points rouges, on imagine qu'il sera beaucoup plus facile de créer un modèle à gauche, alors qu'à droite il manquera des données pour représenter la seconde bosse (qu'on appelle mode). C'est une complexité qui demande d'avoir plus de données, alors qu'on a le même nombre de dimensions pour les deux graphiques (une seule en l'occurrence).

### Comment résoudre ce problème ?

Comment faire donc, quand on se retrouve dans une situation où il y a trop de features par rapport au nombre de données qu'on possède ?

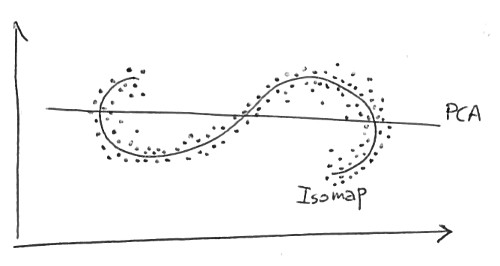
#### Réduire les dimensions

La première solution est bien sûr de réduire le nombre de dimensions des données en entrée. De nombreuses techniques **non supervisées** existent pour cela, même si ce n'est pas l'objet de ce cours d'introduction. Il existe deux manières de réduire le nombre de dimensions d'un problème de machine learning :

* retrouver les **dimensions principales** d'un phénomène, qui ne sont pas forcément celles qu'on observe directement mais peuvent être, notamment, une combinaison linéaire de celles-ci ;
* retrouver la **variété** sous-jacente (ou **manifold** en anglais)— une variété est un objet mathématique qui, grossièrement, représente une surface topologique de dimension inférieure au problème de grande dimension traité. Ce n'est pas très important de comprendre la théorie liée à ce type de traitement, l'essentiel est de comprendre la différence avec la réduction de dimension linéaire simple (qui est en fait un cas particulier de **manifold**).

C'est un peu compliqué cette histoire, non ?

Effectivement ! L'essentiel, c'est juste de faire la distinction entre un espace qui peut simplement être représenté par ses dimensions principales, versus des espaces complexes qui auront besoin d'algorithmes de manifold learning. Il vous faudra bien penser à tester les deux. Ces méthodes seront étudiées plus en détail dans un prochain cours.

Exemple de stack exchange

Dans le dessin ci-dessus, les données sont représentées par des points. La droite légendée "PCA" représente une **réduction linéaire** des points par un algorithme appelé **PCA** (Principal Component Analysis). La courbe représente une **réduction non linéaire** effectuée par un algorithme appelé **Isomap**. Comme on peut le voir ici, une réduction linéaire va écraser les données, alors qu'une réduction non linéaire va mieux conserver la forme du phénomène qu'on cherche à simplifier. [J'ai récupéré l'exemple ci-dessus ici.](https://stats.stackexchange.com/questions/124534/how-to-understand-nonlinear-as-in-nonlinear-dimensionality-reduction)

Ces systèmes de réduction de dimension sont aussi utilisés pour la **visualisation des données**, puisqu'on peut réduire leur représentation à deux, voire trois dimensions. C'est donc très pratique car vous avez dû remarquer que nous, humains, avons bien du mal à nous représenter des objets qui ont plus de 3 dimensions... 🤖

#### Choisir le bon algorithme

Pour en revenir au travail de modélisation, votre travail ici consiste à choisir un modèle assez **contraint** pour pouvoir supporter les variations dimensionnelles sans augmenter en complexité. C'est une balance qui est à trouver.

Par exemple, une régression linéaire contraint le modèle sous forme de droite. Si les données sont linéaires de cette manière, c'est pas mal. Si ce n'est pas le cas, on risque de perdre de l'information importante qui représente le phénomène. Finalement, on se retrouve dans le même genre de dilemme que d'habitude. C'est juste qu'en plus grande dimension, on doit faire encore plus attention à cette sélection. 😀

Attention à ne pas tomber dans l'underfitting ! La tendance naturelle de sélection consiste à prendre des modèles trop contraints et laisser passer des variations importantes dans le modèle.

### En résumé

* Le **fléau de la dimension** représente simplement le fait qu'il y a une explosion dans la taille du jeu de données nécessaire à l'apprentissage du modèle lorsqu'on augmente le nombre de features en entrée.
* De ce fait, on doit **choisir avec encore plus d'attention notre modèle**, sa complexité et son apprentissage.
* On peut aussi réduire les dimensions de l'entrée en amont afin de simplifier encore le traitement du problème.

# Appréhendez les limites du Machine Learning

Bravo ! Vous avez réussi cet exercice !

### Compétences évaluées

* Identifiez les limites du Machine Learning

### Question 1

**Que signifie le théorème du no free lunch ?**

*Attention, plusieurs réponses sont possibles.*

* + 

En moyenne sur l'ensemble des problèmes de machine learning, aucun algorithme n'aura de meilleure performance qu'un autre.

* + 

En moyenne sur l'ensemble des problèmes de machine learning, il existe quelques algorithmes qui surpassent tous les autres.

* + 

Chaque problème nécessite que l'on construise des hypothèses de modélisation spécifiques afin de le résoudre avec efficacité.

*Le théorème du no free lunch est bien un rappel qu'aucun algorithme ne peut avoir (dans un temps fini) de bonnes performances sur l'ensemble des problèmes que le data scientist sera amené à rencontrer. Il faut donc adapter ses algorithmes à chaque problématique spécifique !*

### Question 2

**Que doit-on faire quand on se retrouve face à un problème qui semble intractable ?**

*Attention, plusieurs réponses sont possibles.*

* + 

Changer les hypothèses de modélisation, notamment les simplifier.

* + 

Réduire le nombre de variables en entrée.

* + 

Attendre encore en espérant que la situation change.

* + 

Abandonner le problème, la data science ne permettra pas de résoudre ce problème.

*Un problème*intractable *signifie en pratique qu'un ordinateur aura énormément de mal à trouver une solution. Pour cette raison, on doit effectuer des approximations et simplifications supplémentaires.*

### Question 3

**Comment peut être résolu le compromis biais-variance ?**

*Attention, plusieurs réponses sont possibles.*

* + 

En diminuant la variance du modèle grâce à des méthodes ensemblistes

* + 

En essayant de trouver l'optimum de l'erreur global (biais+variance) en fonction de la complexité

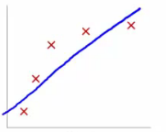
* + 

En entraînant l'algorithme avec plus de données

*Plus de données d'entraînement ne changeront pas le problème si un algorithme est trop variable ou trop biaisé.*

### Question 4

**Quel est le problème du modèle suivant ?**



* + 

Il est en overfitting.

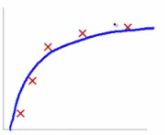
* + 

Il est en underfitting.

* + 

Je ne vois pas de problème avec ce modèle.

*Il est en underfitting, puisque l’on a une perte d'information sur le phénomène qu'on cherche à modéliser qui a la forme d'une courbe. Une modélisation plus performante ressemblerait plutôt à ça :*

**

### Question 5

**Qu'est-ce que la généralisation ?**

* + 

La capacité d'un algorithme à pouvoir extrapoler ses connaissances et prédire des résultats sur des données qu'il n'a jamais rencontrées

* + 

La capacité d'un algorithme à s'ajuster aux données d'entraînement

* + 

La capacité d'un algorithme à pouvoir être efficace sur des observations qui ne correspondent pas au phénomène modélisé à la base

*La généralisation, c'est la capacité d'un algorithme à pouvoir bien "comprendre" le phénomène modélisé pour pouvoir extrapoler ses connaissances sur des données qu'il n'a jamais vues.*

### Question 6

**Qu'est-ce que le fléau de la dimension ?**

*Attention, plusieurs réponses sont possibles.*

* + 

Le fait qu'on a souvent besoin de plus de données si le nombre de features augmente

* + 

Le fait qu'étant donné un dataset de taille fixe, la capacité de prédiction d'un algorithme diminue si on augmente le nombre de dimensions

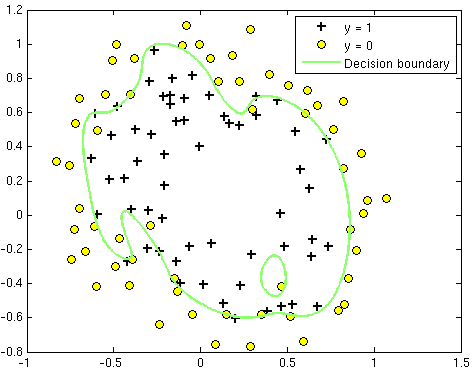
* + 

Le fait que la dimension d'un algorithme empêche de modéliser un phénomène pas assez complexe

*Le fléau de la dimension désigne la diminution des performances d'un algorithme lorsqu'on augmente le nombre de dimensions.*

### Question 7

**La ligne de séparation n'est pas idéale pour classer les '+' des 'o'. Pourquoi ?**

****

* + 

Le modèle semble légèrement en underfitting.

* + 

Je ne vois pas de problème avec cette ligne de séparation.

* + 

Le modèle semble légèrement en overfitting.

*Il ne faut pas chercher à être autant collé aux données. Ici, on a trop pris en compte la variance des observations pour effectuer la modélisation. Ce n'est pas représentatif du phénomène modélisé, et celui-ci est donc en overfitting.*

### Question 8

**Un algorithme à haute variance va plus facilement...**

* + 

être en overfitting.

* + 

être en underfitting.

*Haute variance signifie potentiellement capturer plus de bruit sur les observations et donc "surapprentissage" (overfitting).*