# Machine Learning avec Scikit-Learn

Chez le même éditeur

*Big Data et Machine Learning*
3e édition
Pirmin Lemberger, Marc Batty, Médéric Morel, Jean-Luc Raffaëlli
272 pages
Dunod, 2019

*Introduction au Machine Learning*
Chloé-Agathe Azencott
240 pages
Dunod, 2018

*Python pour le data scientist*
Emmanuel Jakobowicz
304 pages
Dunod, 2018

O'REILLY®

# Machine Learning avec Scikit-Learn

## Mise en œuvre et cas concrets

2e édition

Aurélien Géron

*Traduit de l'anglais par Anne Bohy*

DUNOD

Authorized French translation of material from the English edition of
*Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2E*
ISBN 9781492032649


Conception de la couverture : Karen Montgomery
Illustratrice : Rebecca Demarest



© Dunod, 2017, 2019
11 rue Paul Bert, 92240 Malakoff
www.dunod.com
ISBN 978-2-10-079065-4

# Table des matières

# Avant-propos

### *La révolution du Machine Learning*

Seuls ceux qui ont vécu dans une caverne ces dix dernières années ont pu ignorer l'incroyable révolution de l'apprentissage automatique, ou *Machine Learning* (ML). Il ne se passe plus une semaine sans qu'il ne fasse parler de lui : cela a commencé par de formidables progrès en reconnaissance d'images, puis en analyse de la voix, le programme Watson d'IBM est ensuite devenu champion du jeu de Jeopardy, on a vu les premières voitures autonomes de Google sillonner les routes, puis le programme AlphaGo de DeepMind a vaincu le champion du monde du jeu de go, le logiciel Libratus de l'université Carnegie Mellon a écrasé des champions de poker, des patients paralytiques ont pu contrôler le mouvement de leurs membres par la pensée, grâce à un programme qui avait appris à déchiffrer certaines de leurs ondes cérébrales… bref, les succès s'enchaînent et ne se ressemblent pas. Il y a dix ans, de telles intelligences artificielles n'existaient que dans les romans de science-fiction.

### *Le Machine Learning en entreprise*

Au-delà de ces exemples qui font la une des journaux, le Machine Learning envahit plus discrètement les systèmes informatiques de toutes les entreprises. D'ores et déjà, lorsque vous parcourez Internet ne serait-ce que quelques minutes, une horde de systèmes d'apprentissage automatique s'activent : certains analysent votre personnalité pour vous proposer les produits qui vous correspondent le mieux, d'autres sélectionnent les publicités qui attireront votre attention, et d'autres encore analysent votre comportement pour s'assurer que vous n'êtes pas un fraudeur. Mais le Machine Learning n'est pas réservé aux géants du web : qu'il s'agisse de prédire des séries temporelles (comme les cours de la Bourse), de détecter des anomalies de production, d'optimiser des centres d'appel, d'analyser les profils des clients, ou encore de classer automatiquement des documents, l'apprentissage automatique s'est révélé d'une grande utilité dans une multitude de domaines. Si votre entreprise n'utilise pas encore le Machine Learning, elle risque fort de se faire devancer rapidement par ses concurrents.

### *Data Scientist, une espèce rare*

Le marché du Machine Learning croît si rapidement que le nombre d'experts en analyse de données (*data scientist*) a bien de la peine à suivre. Malgré un nombre d'étudiants en forte hausse en sciences des données (*data science*), il est aujourd'hui difficile pour une entreprise de trouver suffisamment de profils compétents. Face à cette pénurie d'expertise, la meilleure solution est probablement de former certains employés au Machine Learning. Les mieux placés pour cela sont naturellement les ingénieurs en informatique, car ils maîtrisent déjà la programmation et bien souvent aussi les bases mathématiques que requiert le Machine Learning. En outre, ils sont souvent très demandeurs, à la fois car ils savent que la compétence est rare, mais aussi et surtout car le sujet est absolument passionnant !

### *Objectif et approche*

Ce livre a donc pour objectif de vous former au Machine Learning. Vous apprendrez les concepts fondamentaux et les outils nécessaires pour créer des systèmes capables d'apprendre à partir d'exemples, qu'il s'agisse de classer des images, d'estimer la valeur d'un bien, etc. Nous aborderons un grand nombre de techniques et d'algorithmes, depuis de simples systèmes linéaires, aux arbres de décision et aux forêts aléatoires en passant par les machines à vecteurs de support, et bien d'autres encore. Vous apprendrez aussi à utiliser Scikit-Learn[1], l'une des librairies open source les plus simples et néanmoins les plus puissantes disponibles aujourd'hui, que vous pourrez directement utiliser dans vos systèmes en production.

Hormis le premier chapitre, qui offre une vision d'ensemble du Machine Learning et des principaux concepts, le reste de cet ouvrage est résolument axé sur la pratique : le but n'est pas juste de vous enseigner la théorie, mais aussi que vous sachiez concrètement développer vos propres systèmes de ML. Tous les exemples de code sont disponibles en ligne sous la forme de notebooks Jupyter à l'adresse suivante : *https://github.com/ageron/handson-ml2*. Bien que vous puissiez lire ce livre sans allumer votre ordinateur, vous êtes fortement encouragé(e) à mettre la main à la pâte au cours de la lecture, en particulier en faisant les exercices proposés à la fin de chaque chapitre, et dont les solutions sont disponibles à la fin de l'ouvrage.

### *Prérequis*

Bien que ce livre ait été écrit plus particulièrement pour les ingénieurs en informatique, il peut aussi intéresser toute personne sachant programmer et ayant quelques bases mathématiques. Il ne requiert aucune connaissance préalable sur le Machine Learning mais il suppose les prérequis suivants :

- vous devez avoir un minimum d'expérience de programmation ;
- sans forcément être un expert, vous devez connaître le langage Python, et si possible également ses librairies scientifiques, en particulier NumPy, pandas et Matplotlib ;

1. Cette librairie a été créée par David Cournapeau en 2007, et le projet est maintenant dirigé par une équipe de chercheurs à l'Institut national de recherche en informatique et en automatique (Inria).

- enfin, si vous voulez comprendre comment les algorithmes fonctionnent (ce qui n'est pas forcément indispensable, mais est tout de même très recommandé), vous devez avoir certaines bases en mathématiques dans les domaines suivants :
  – l'algèbre linéaire, notamment comprendre les vecteurs et les matrices (par exemple comment multiplier deux matrices, transposer ou inverser une matrice),
  – le calcul différentiel, notamment comprendre la notion de dérivée, de dérivée partielle, et savoir comment calculer la dérivée d'une fonction.

Si vous ne connaissez pas encore Python, il existe de nombreux tutoriels sur Internet, que nous vous encourageons à suivre : ce langage est très simple et s'apprend vite. En ce qui concerne les librairies scientifiques de Python et les bases mathématiques requises, le site *github.com/ageron/handson-ml2* propose quelques tutoriels (en anglais) sous la forme de notebooks Jupyter. De nombreux tutoriels en français sont disponibles sur Internet. Le site *fr.khanacademy.org* est particulièrement recommandé pour les mathématiques.

### *Plan du livre*

- Le premier chapitre présente une vue d'ensemble du Machine Learning ainsi que les concepts fondamentaux : qu'entend-on exactement par *apprentissage automatique* ? Quels problèmes le Machine Learning peut-il résoudre ? Quelles sont les principales catégories d'algorithmes et les principales difficultés que l'on peut rencontrer ? Qu'est-ce que le surajustement et le sous-ajustement ?
- Le deuxième chapitre attaque le vif du sujet en présentant un projet de Machine Learning de A à Z (en introduisant Scikit-Learn au passage) : d'abord analyser le problème (en l'occurrence estimer le prix de l'immobilier), puis obtenir les données, les nettoyer, choisir une fonction d'évaluation, sélectionner un modèle, l'entraîner sur les données d'entraînement, évaluer le système sur un jeu de données préalablement mis de côté, rechercher les meilleurs hyperparamètres grâce à la validation croisée, etc.
- Le chapitre 3 analyse les difficultés particulières liées aux problèmes de classification, en particulier les diverses façons d'évaluer un classificateur, et comment le régler au mieux selon ses besoins.
- Le chapitre 4 est le premier à ouvrir les boîtes noires, pour expliquer comment fonctionnent les algorithmes. Il explique en particulier les principaux modèles linéaires et montre comment les entraîner en ajustant progressivement leurs paramètres à l'aide de l'algorithme de descente de gradient. Il montre aussi diverses techniques de régularisation qui permettent d'éviter le piège du surajustement.
- Le chapitre 5 se concentre sur les machines à vecteur de support (SVM), des modèles très puissants, particulièrement prisés pour les problèmes complexes pour lesquels on ne dispose que d'assez peu de données.
- Le chapitre 6 détaille les arbres de décision. Il s'agit de modèle simples et polyvalents, particulièrement faciles à interpréter.

- Le chapitre 7 présente les forêts aléatoires, qui reposent sur une multitude d'arbres de décision, et il explore plus généralement diverses méthodes ensemblistes, qui permettent de combiner plusieurs modèles de Machine Learning.
- Le chapitre 8 détaille plusieurs algorithmes de réduction de dimensionnalité, permettant de réduire le volume d'un jeu de données pour échapper au fléau de la dimensionnalité et ainsi accélérer l'apprentissage, ou encore à des fins de visualisation des données.
- Enfin, le chapitre 9, apparu dans cette deuxième édition et ayant pour objet l'apprentissage non supervisé, traite notamment du partitionnement avec la méthode des *k*-moyennes ou l'algorithme DBSCAN, des modèles de mélange gaussien, de l'inférence bayésienne et de l'utilisation des modèles de mélange pour le partitionnement, l'estimation de densité, la détection d'anomalies ou de nouveautés. Il fait également un tour d'horizon des algorithmes de partitionnement et de détection d'anomalies ou de nouveautés.

### *Le Deep Learning*

Pour traiter des problèmes particulièrement complexes, tels que la reconnaissance de la parole ou encore les traductions automatiques, on utilise généralement des réseaux de neurones, qui requièrent souvent de gros volumes de données et une immense puissance de calcul. Presque toutes les applications du Machine Learning mises en avant par la presse reposent sur des réseaux de neurones profonds (c'est-à-dire organisés en de nombreuses couches successives) : on parle alors de *Deep Learning* (apprentissage profond).

Ce livre n'aborde pas le Deep Learning, mais il vous donne les bases nécessaires pour le faire. Si à l'issue de ce livre vous souhaitez poursuivre vers le Deep Learning, nous vous recommandons le livre *Deep Learning avec Keras et TensorFlow*, des mêmes auteur et éditeur. Vous y apprendrez comment utiliser la librairie TensorFlow pour créer diverses architectures de réseaux de neurones profonds – des réseaux de neurones convolutionnels (très utilisés pour l'analyse d'images), des réseaux de neurones récurrents (utiles pour l'analyse de séquences de tailles arbitraires, telles que des séries temporelles ou la langue naturelle), des auto-encodeurs (capables de reconnaître des motifs et de les imiter) – et comment entraîner et déployer ces réseaux de neurones sur de nombreux serveurs grâce à TensorFlow. Enfin, ce deuxième livre présente le Deep Reinforcement Learning : il s'agit des techniques développées par DeepMind pour créer un système capable d'apprendre tout seul à jouer à des jeux Atari sans en connaître les règles à l'avance.

### *Conventions*

Les conventions typographiques suivantes sont utilisées dans ce livre :

*Italique*

Indique un nouveau terme, une URL, une adresse email ou un nom de fichier.

`Largeur fixe`

Utilisé pour les exemples de code, ainsi qu'au sein du texte pour faire référence aux éléments d'un programme, tels que des instructions, des mots clés, des noms de variables, de fonctions, de base de données, de types de données ou encore de variables d'environnement.

**`Largeur fixe et gras`**

Affiche des commandes ou d'autres textes qui doivent être saisis littéralement par l'utilisateur.

Ce symbole indique une astuce ou une suggestion.

Ce symbole indique une précision ou une remarque générale.

Ce symbole indique une difficulté particulière ou un piège à éviter.

### *Exemples de code*

Il existe une série de notebooks Jupyter contenant de la documentation supplémentaire, tels que des exemples de code et des exercices, que vous pouvez télécharger à l'adresse suivante : *github.com/ageron/handson-ml2*.

Certains exemples de code dans le livre omettent des sections répétitives ou des détails évidents ou sans rapport avec le Machine Learning. Cela permet de rester concentré sur les parties importantes du code et de garder de la place pour couvrir davantage de sujets. Si vous voulez les exemples de code complets, ils sont tous disponibles dans les notebooks Jupyter.

Notez que lorsque les exemples de code affichent des sorties, ceux-ci sont affichés avec les invites Python (>>> et …), comme dans un shell Python, afin de distinguer clairement le code des sorties. Par exemple, ce code définit la fonction `square()` puis calcule et affiche le carré de 3 :

```
>>> def square(x):
...     return x ** 2
...
>>> result = square(3)
>>> result
9
```

Lorsque le code n'affiche rien, les invites ne sont pas utilisées. Cependant, le résultat peut parfois être affiché sous forme de commentaire comme ici :

```
def square(x):
    return x ** 2

result = square(3)  # le résultat est 9
```

### *Remerciements*

Jamais, dans mes rêves les plus fous, je n'aurais imaginé que la première édition de ce livre rencontrerait un public aussi vaste. J'ai reçu de nombreux messages de lecteurs, avec beaucoup de questions, certains signalant gentiment des erreurs et la plupart m'envoyant des mots encourageants. Je suis extrêmement reconnaissant envers tous ces lecteurs pour leur formidable soutien. Merci beaucoup à vous tous ! N'hésitez pas à me contacter si vous voyez des erreurs dans les exemples de code ou simplement pour poser des questions (*https://homl.info/issues2*) ! Certains lecteurs ont également expliqué en quoi ce livre les avait aidés à obtenir leur premier emploi ou à résoudre un problème concret sur lequel ils travaillaient. Ces retours sont incroyablement motivants. Si vous trouvez ce livre utile, j'aimerais beaucoup que vous puissiez partager votre histoire avec moi, que ce soit en privé (par exemple, via *https://www.linkedin.com/in/aurelien-geron/*) ou en public (par exemple dans un tweet ou par le biais d'un commentaire Amazon).

Je suis également extrêmement reconnaissant envers toutes les personnes qui ont pris le temps d'examiner mon livre avec autant de soin. En particulier, je voudrais remercier Ankur Patel, qui a examiné chaque chapitre de cette deuxième édition et m'a fourni un excellent retour d'informations, en particulier dans le chapitre 9, qui traite des techniques d'apprentissage non supervisées. Il pourrait écrire un livre entier sur le sujet… oh, attendez, il l'a fait[2] !

Un grand merci au personnel fantastique d'O'Reilly, en particulier Nicole Taché, qui m'a fait des commentaires perspicaces, toujours encourageants et utiles : je ne pouvais pas rêver d'un meilleur éditeur. Merci également à Michele Cronin, qui a été très efficace (et patiente) au début de cette deuxième édition, et à Kristen Brown, responsable de la production pour la deuxième édition, qui a suivi toutes les étapes (elle a également coordonné les correctifs et les mises à jour pour chaque réimpression de la première édition). Merci également à Rachel Monaghan et à Amanda Kersey pour leur révision complète (respectivement pour les première et deuxième éditions), et à Johnny O'Toole qui a géré la relation avec Amazon et répondu à beaucoup de mes questions. Merci à Marie Beaugureau, Ben Lorica, Mike Loukides et Laurel Ruma d'avoir cru en ce projet et de m'avoir aidé à le définir. Merci à Matt Hacker et à toute l'équipe d'Atlas pour avoir répondu à toutes mes questions techniques concernant AsciiDoc et LaTeX, ainsi qu'à Nick Adams, Rebecca Demarest, Rachel Head, Judith McConville, Helen Monroe, Karen Montgomery, Rachel Roumeliotis et tous les autres membres d'O'Reilly qui ont contribué à la rédaction de ce livre.

2. Ankur Patel, *Hands-On Unsupervised Learning Using Python: How to Build Applied Machine Learning Solutions from Unlabeled Data* (O'Reilly, 2019) : *https://homl.info/patel*.

Je tiens également à remercier mes anciens collègues de Google, en particulier l'équipe de classification des vidéos YouTube, de m'avoir énormément appris à propos du Machine Learning. Je n'aurais jamais pu commencer la première édition sans eux. Un merci spécial à mes gourous personnels du ML : Clément Courbet, Julien Dubois, Mathias Kende, Daniel Kitachewsky, James Pack, Alexander Pak, Anosh Raj, Vitor Sessak, Wiktor Tomczak, Ingrid von Glehn et Rich Washington. Et merci à tous ceux avec qui j'ai travaillé chez YouTube et dans les formidables équipes de recherche Google de Mountain View.

Je n'oublierai jamais les personnes qui ont aimablement relu la première édition de ce livre, notamment David Andrzejewski, Lukas Biewald, Justin Francis, Vincent Guilbeau, Eddy Hung, Karim Matrah, Grégoire Mesnil, Salim Sémaoune, Iain Smears, Michel Tessier, Ingrid von Glehn, Pete Warden et bien sûr mon cher frère Sylvain.

Je souhaite également remercier Jean-Luc Blanc, des éditions Dunod, pour avoir soutenu et géré ce projet, et pour ses relectures attentives. Je tiens aussi à remercier vivement Anne Bohy qui a traduit cet ouvrage en français bien mieux que je n'aurais su le faire. Dotée d'une grande expérience en développement informatique et d'un doctorat en mathématiques, elle était sans doute la traductrice idéale pour ce livre. Enfin, je remercie chaleureusement Brice Martin, des éditions Dunod, pour sa relecture extrêmement rigoureuse, ses excellentes suggestions et ses nombreuses corrections.

Pour finir, je suis infiniment reconnaissant à ma merveilleuse épouse, Emmanuelle, et à nos trois enfants, Alexandre, Rémi et Gabrielle, de m'avoir encouragé à travailler dur pour ce livre. Je les remercie également pour leur curiosité insatiable : expliquer certains des concepts les plus difficiles de ce livre à ma femme et à mes enfants m'a aidé à clarifier mes pensées et à améliorer directement de nombreuses parties de ce livre. J'ai même eu droit à des biscuits et du café, comment rêver mieux ?

# 1

# Vue d'ensemble du Machine Learning

Lorsqu'on leur parle de « Machine Learning », que l'on traduit en français par « apprentissage automatique », la plupart des personnes s'imaginent un robot : un serviteur fiable ou un Terminator redoutable, selon l'interlocuteur. Mais ce Machine Learning n'est pas un rêve futuriste : c'est déjà une réalité. À vrai dire, il est présent depuis des décennies dans certaines applications spécialisées telles que la *reconnaissance optique de caractères* ou OCR (Optical Character Recognition). La première application ML ayant véritablement touché un large public, améliorant le quotidien de centaines de millions de personnes, s'est imposée dans les années 1990 : il s'agit du *filtre anti-spam*. S'il n'est pas vraiment présenté comme tel, il possède pourtant techniquement les caractéristiques d'un système d'apprentissage automatique (d'ailleurs, il a si bien appris que vous n'avez que rarement à signaler un e-mail comme indésirable). Il a été suivi par des centaines d'applications ML intégrées désormais discrètement dans des centaines de produits et fonctionnalités que vous utilisez régulièrement, depuis les recommandations jusqu'à la recherche vocale.

Où commence l'apprentissage automatique, et où s'arrête-t-il ? Lorsqu'on dit qu'une machine *apprend* quelque chose, qu'est-ce que cela signifie ? Si je télécharge Wikipédia sur mon ordinateur, ce dernier a-t-il vraiment « appris » quelque chose ? Est-il soudainement devenu plus intelligent ? Dans ce chapitre, nous allons commencer par clarifier ce qu'est l'apprentissage automatique et les raisons qui peuvent vous pousser à l'utiliser.

Puis, avant d'entreprendre d'explorer ce vaste territoire, nous en examinerons la carte, découvrirons les principales régions et étudierons leurs particularités les plus notables : apprentissage supervisé / apprentissage non supervisé, apprentissage en ligne / apprentissage groupé, apprentissage à partir d'observations / apprentissage à partir d'un modèle. Après quoi, nous nous intéresserons au déroulement d'un projet

ML type, nous évoquerons les principales difficultés que vous pourriez rencontrer et expliquerons comment évaluer et régler finement votre système.

Ce chapitre introduit un grand nombre de concepts fondamentaux (et de jargon) que chaque spécialiste du traitement des mégadonnées doit connaître par cœur. Il s'agira d'une présentation très générale (c'est le seul chapitre qui ne comporte pratiquement pas de code) ; l'ensemble est assez simple, mais assurez-vous que tout soit parfaitement clair pour vous. Préparez-vous un café, et commençons !

Si vous avez déjà toutes les connaissances de base en apprentissage automatique, vous pouvez passer directement au chapitre 2. En cas de doute, essayez de répondre à toutes les questions figurant à la fin de ce chapitre.

## 1.1 QU'EST-CE QUE L'APPRENTISSAGE AUTOMATIQUE ?

L'apprentissage automatique est la science (et l'art) de programmer les ordinateurs de sorte qu'ils puissent *apprendre à partir de données*.

Voici une définition un peu plus générale :

> « [*L'apprentissage automatique est la*] *discipline donnant aux ordinateurs la capacité d'apprendre sans qu'ils soient explicitement programmés*. »
>
> Arthur Samuel, 1959

En voici une autre plus technique :

> « *Étant donné une tâche T et une mesure de performance P, on dit qu'un programme informatique apprend à partir d'une expérience E si les résultats obtenus sur T, mesurés par P, s'améliorent avec l'expérience E*. »
>
> Tom Mitchell, 1997

Votre filtre anti-spam est un programme d'apprentissage automatique qui peut apprendre à identifier les e-mails frauduleux à partir d'exemples de pourriels ou « spam » (par exemple, ceux signalés par les utilisateurs) et de messages normaux (parfois appelés « ham »). Les exemples utilisés par le système pour son apprentissage constituent le *jeu d'entraînement* (en anglais, *training set*). Chacun d'eux s'appelle une *observation d'entraînement* (on parle aussi d'*échantillon*). Dans le cas présent, la tâche T consiste à identifier parmi les nouveaux e-mails ceux qui sont frauduleux, l'expérience E est constituée par les *données d'entraînement*, et la mesure de performance P doit être définie (vous pourrez prendre par exemple le pourcentage de courriels correctement classés). Cette mesure de performance particulière, appelée *exactitude* (en anglais, *accuracy*), est souvent utilisée dans les tâches de classification.

Si vous téléchargez simplement Wikipédia, votre ordinateur dispose de beaucoup plus de données, mais cela ne le rend pas soudainement meilleur pour quelque tâche que ce soit. Par conséquent, il ne s'agit pas d'apprentissage automatique.

## 1.2 POURQUOI UTILISER L'APPRENTISSAGE AUTOMATIQUE ?

Voyons comment vous écririez un filtre de spam en utilisant des techniques de programmation traditionnelles (voir figure 1.1) :

1. Tout d'abord, vous examineriez à quoi ressemble en général un courrier frauduleux. Vous remarqueriez peut-être que certains mots (tels que « pour vous », « carte bancaire », « gratuit », « bravo »…) semblent apparaître souvent dans l'intitulé du sujet. Vous remarqueriez peut-être aussi quelques autres détails caractéristiques dans le nom de l'expéditeur, le corps du message et d'autres parties du message.
2. Après quoi, vous écririez un algorithme de détection de chacune des caractéristiques repérées, et votre programme marquerait comme indésirables tous les messages dans lesquels un certain nombre de ces caractéristiques ont été détectées.
3. Vous testeriez alors le programme et répéteriez les étapes 1 et 2 jusqu'à obtenir un résultat satisfaisant.

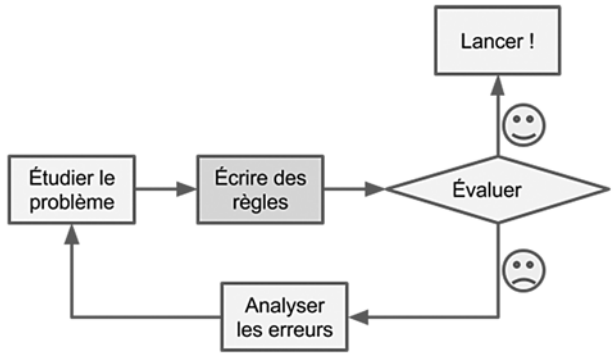


**Figure 1.1** – Approche traditionnelle

Le problème étant compliqué, votre programme deviendra vraisemblablement une longue liste de règles complexes assez difficiles à maintenir.

Par contraste, un filtre anti-spam basé sur des techniques de Machine Learning apprend automatiquement quels sont les mots et les phrases qui constituent de bons prédicteurs de courriels frauduleux en détectant des associations de mots aux fréquences inhabituelles dans des exemples de courriels frauduleux comparés à des exemples de courriels licites (voir figure 1.2). Le programme est beaucoup plus court, plus facile à maintenir et a plus de chances de fournir de bons résultats.

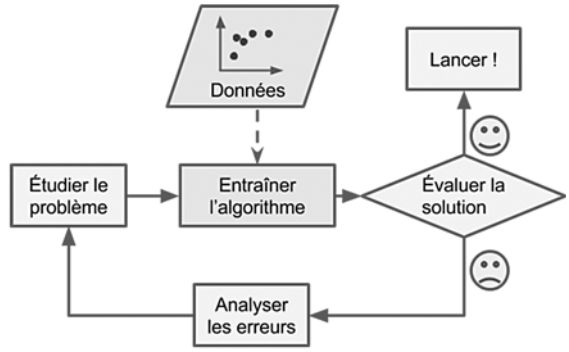


**Figure 1.2** – L'approche du Machine Learning

Si les pirates remarquent que leurs e-mails contenant le mot « Pour vous » sont bloqués, ils peuvent décider d'écrire « For you » à la place. Un filtre anti-spam utilisant des techniques de programmation traditionnelles nécessiterait une mise à jour pour pouvoir intercepter les messages « For you ». Si les pirates continuent à tenter de contourner votre filtre anti-spam, vous devrez sans arrêt écrire de nouvelles règles.

Par contre, un filtre anti-spam basé sur des techniques de Machine Learning repère automatiquement que « For you » est devenu inhabituellement fréquent dans les messages indésirables signalés par les utilisateurs et il commence à en tenir compte sans votre intervention (voir figure 1.3).

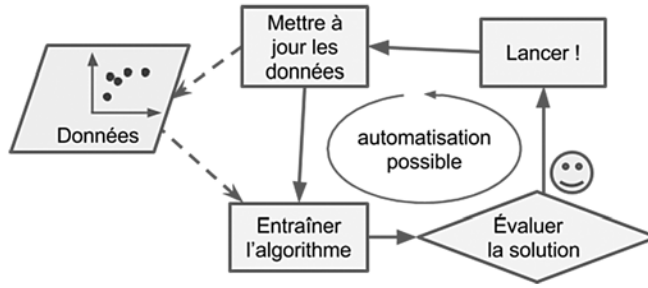


**Figure 1.3** – Adaptation automatique aux évolutions

Un autre domaine dans lequel l'apprentissage automatique fait merveille est celui des problèmes pour lesquels il n'existe pas d'algorithme connu ou dont la complexité déroute. Considérons par exemple la reconnaissance vocale. Supposons que vous souhaitiez débuter par quelque chose de simple, en écrivant un programme capable de distinguer les mots « Un » et « Deux ». Vous pourriez remarquer que le mot « Deux » débute par un son de tonalité élevée (« D »), ce qui pourrait vous conduire à coder en dur un algorithme mesurant l'intensité des sons de tonalité élevée et à l'utiliser pour distinguer les « Un » et les « Deux ». Cependant, cette technique s'adaptera manifestement mal à des milliers de mots prononcés par des millions de personnes très différentes dans des environnements bruyants et dans

des dizaines de langues distinctes. La meilleure solution (du moins jusqu'à maintenant) consiste à écrire un algorithme qui apprend par lui-même, à partir de nombreux enregistrements pour chaque mot.

Enfin, l'apprentissage automatique peut aider les humains à apprendre (voir figure 1.4) : ses algorithmes peuvent être inspectés afin de découvrir ce qu'ils ont appris (même si, pour certains algorithmes, cela peut s'avérer difficile). Ainsi, une fois que le filtre anti-spam a été entraîné sur suffisamment de courriers frauduleux, on peut l'inspecter aisément pour découvrir la liste des mots et combinaisons de mots qu'il considère être les meilleurs prédicteurs de spam. Parfois, cela révélera des corrélations insoupçonnées ou de nouvelles tendances, permettant ainsi d'avoir une meilleure compréhension du problème.

Appliquer des techniques d'apprentissage automatique pour explorer de gros volumes de données peut permettre d'y découvrir certains éléments de structuration qui n'étaient pas immédiatement apparents. C'est ce qu'on appelle l'*exploration des données* (en anglais, *data mining*).

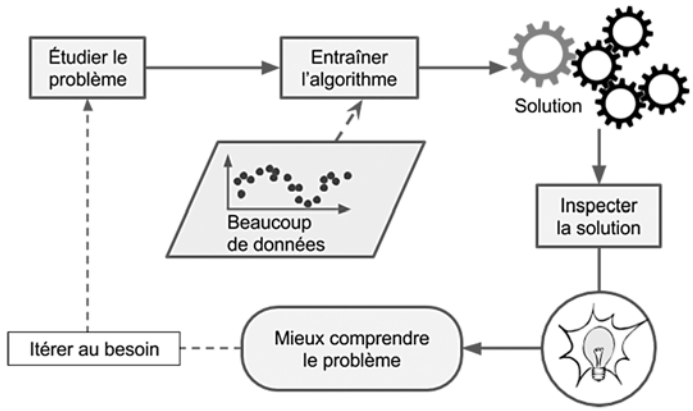


**Figure 1.4** – L'apprentissage automatique peut aider les humains à apprendre

En résumé, l'apprentissage automatique est excellent pour :

- les problèmes pour lesquels les solutions existantes requièrent beaucoup d'ajustements fins ou de longues listes de règles : un algorithme d'apprentissage automatique peut souvent simplifier le code et donner de meilleurs résultats que l'approche traditionnelle ;
- les problèmes complexes pour lesquels il n'existe aucune bonne solution si l'on adopte une approche traditionnelle : les meilleures techniques d'apprentissage automatique peuvent peut-être trouver une solution ;
- les environnements fluctuants : un système d'apprentissage automatique peut s'adapter à de nouvelles données ;
- l'exploration des problèmes complexes et des gros volumes de données.

## 1.3 EXEMPLES D'APPLICATIONS

Voyons maintenant quelques exemples concrets de tâches d'apprentissage automatique, ainsi que les techniques particulières nous permettant de les mener à bien :

- Analyse d'images de produits sur une chaîne de production pour les classifier automatiquement : il s'agit d'une classification d'images, effectuée en général à l'aide de réseaux de neurones convolutifs (en anglais, *convolutional neural networks* ou CNN)[3].
- Détection de tumeurs sur des scanners cérébraux : il s'agit d'une segmentation sémantique dans laquelle chaque pixel de l'image est classifié (étant donné que nous souhaitons déterminer l'emplacement exact et la forme des tumeurs), également à l'aide de CNN en général.
- Classification automatique d'articles de presse : il s'agit de traitement automatique du langage naturel (ou TALN) pour lequel on utilise en général des réseaux de neurones récurrents (ou RNR), des CNN ou des transformateurs[3].
- Identification automatique de commentaires inappropriés dans les forums de discussion : il s'agit aussi de classification de texte à l'aide des mêmes outils de TALN.
- Synthèse automatique de longs documents : il s'agit d'une branche de TALN appelée résumé automatique de texte et utilisant les mêmes outils.
- Création d'un agent conversationnel (alias chatbot) ou d'un assistant personnel : une telle tâche fait appel à de nombreux composants de TALN, parmi lesquels des modules de compréhension de langage naturel (CLN) et de réponse aux questions.
- Prévision des futurs résultats financiers d'une entreprise, en fonction de différentes métriques de performances : il s'agit d'une tâche de régression (consistant à prédire des valeurs) pouvant être effectuée à l'aide d'un modèle de régression linéaire ou polynomiale (voir chapitre 4), à l'aide d'une régression SVM (voir chapitre 5), d'une forêt aléatoire (voir chapitre 7) ou d'un réseau de neurones artificiels[3]. Si vous voulez prendre en compte des séquences de métriques de performances antérieures, vous utiliserez peut-être des réseaux de neurones récurrents (RNR), des réseaux de neurones convolutifs (CNN) ou des transformateurs[3].
- Développement d'une interface de commande vocale pour votre appli. Il s'agit de reconnaissance vocale, impliquant le traitement d'échantillons audio : s'agissant de séquences longues et complexes, on utilise en général des RNR, des CNN ou des transformateurs[3].
- Détection d'utilisations frauduleuses de cartes bancaires : il s'agit de détection d'anomalies (voir chapitre 9).
- Segmentation de clients en fonction de leurs achats, de façon à pouvoir concevoir une stratégie marketing différente pour chaque segment : il s'agit de partitionnement (voir chapitre 9).

3. Pour plus d'informations, voir l'ouvrage *Deep Learning avec Keras et TensorFlow*, A. Géron, Dunod (2e édition, 2020).

- Élaboration d'une représentation graphique claire et explicite à partir d'un jeu de données complexe de grande dimension : il s'agit de visualisation de données mettant fréquemment en œuvre des techniques de réduction de dimension (voir chapitre 8).
- Recommandation d'un produit pouvant intéresser un client, compte tenu de ses achats antérieurs : il s'agit d'un système de préconisation. On peut fournir à un réseau de neurones artificiels[3] les achats antérieurs et autres informations concernant ce client et lui demander en sortie quel sera l'achat suivant le plus probable. Ce réseau de neurones sera en général entraîné sur les séquences d'achats antérieures de tous les clients.
- Réalisation d'un programme-robot intelligent pour un jeu. On utilise souvent pour cela l'apprentissage par renforcement (en anglais, *reinforcement learning* ou RL)[3], une technique d'apprentissage automatique consistant à entraîner des agents (tels que des robots) à choisir les actions maximisant leurs récompenses à long terme (p. ex., un robot peut recevoir une récompense chaque fois qu'un joueur perd des points de vie) dans un environnement donné (tel un jeu). Le fameux programme AlphaGo, qui a battu le champion du monde de go, utilisait des techniques de RL.

On pourrait allonger cette liste, mais cela devrait déjà vous donner un aperçu de l'extrême diversité et complexité des tâches auxquelles l'apprentissage automatique peut s'attaquer, et des diverses techniques utilisables dans chaque cas.

## 1.4 TYPES DE SYSTÈMES D'APPRENTISSAGE AUTOMATIQUE

Il existe tellement de types de systèmes d'apprentissage automatique différents qu'il est utile de les classer en grandes catégories :

- selon que l'apprentissage s'effectue ou non sous supervision humaine (apprentissage supervisé, non supervisé, semi-supervisé ou avec renforcement),
- selon que l'apprentissage s'effectue ou non progressivement, au fur et à mesure (apprentissage en ligne ou apprentissage groupé),
- selon qu'il se contente de comparer les nouvelles données à des données connues, ou qu'il détecte au contraire des éléments de structuration dans les données d'entraînement et construise un modèle prédictif à la façon d'un scientifique (apprentissage à partir d'observations ou apprentissage à partir d'un modèle).

Ces critères ne sont pas exclusifs, vous pouvez les combiner comme vous le souhaitez. Ainsi, un filtre anti-spam dernier cri peut apprendre au fur et à mesure en s'appuyant sur un modèle de réseau neuronal profond dont l'apprentissage s'effectue sur des exemples de messages indésirables ou non : ceci en fait un système supervisé d'apprentissage en ligne à partir d'un modèle. Examinons maintenant plus soigneusement chacun de ces critères.

### 1.4.1 Apprentissage supervisé / non supervisé

Les systèmes d'apprentissage automatique peuvent être classés en fonction de l'importance et de la nature de la supervision qu'ils requièrent durant la phase d'entraînement. Il existe quatre catégories majeures : l'apprentissage supervisé, l'apprentissage non supervisé, l'apprentissage semi-supervisé et l'apprentissage avec renforcement.

#### *Apprentissage supervisé*

Dans l'*apprentissage supervisé*, les données d'entraînement que vous fournissez à l'algorithme comportent les solutions désirées, appelées *étiquettes* (en anglais, *labels*), comme sur la figure 1.5.

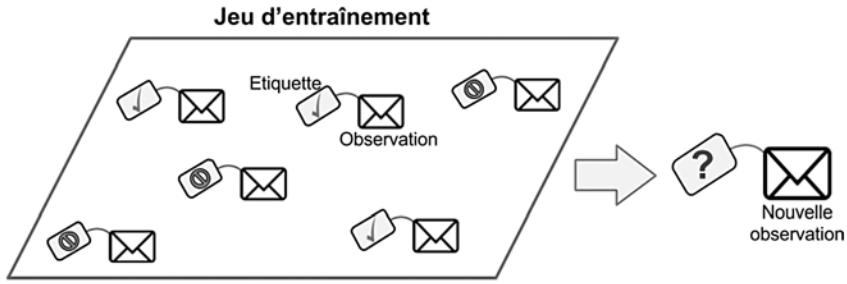


**Figure 1.5** – Un jeu d'entraînement étiqueté pour la classification de spam (un exemple d'apprentissage supervisé)

Un exemple classique de tâche d'apprentissage supervisé est la *classification*. Le filtre de spam en constitue un bon exemple : son apprentissage s'effectue à partir de nombreux exemples d'e-mails accompagnés de leur *classe* (spam ou normal), à partir desquels il doit apprendre comment classer les nouveaux e-mails.

Une autre tâche classique consiste à prédire une valeur numérique *cible* (en anglais, *target*) telle que le prix d'une voiture à partir des valeurs d'un certain nombre d'*attributs* ou *variables*. Ces valeurs sont appelées les *caractéristiques* (en anglais, *features*) d'une observation. Ces variables, comme le kilométrage, l'âge, la marque, etc., sont appelées *variables explicatives* ou encore *prédicteurs*. Une tâche de ce type est une *régression*[4] (voir figure 1.6). Pour entraîner le système, vous devez lui donner beaucoup d'exemples de voitures, en y intégrant à la fois les variables explicatives et la variable à expliquer (c.-à-d. en fournissant les caractéristiques et les étiquettes).

4. Pour la petite histoire, ce nom curieux est un terme statistique introduit par Francis Galton alors qu'il étudiait le fait que les gens de grande taille ont tendance à avoir des enfants plus petits qu'eux. Les enfants étant plus petits, il a nommé cela régression vers la moyenne. Ce nom fut alors donné aux méthodes qu'il utilisait pour analyser les corrélations entre variables.

En apprentissage automatique, un attribut est un type de données (ex. `kilométrage`) tandis qu'une caractéristique peut avoir plusieurs significations selon le contexte, mais désigne en général un attribut et sa valeur (ex. `kilométrage = 15000`). Cependant, beaucoup de personnes utilisent attribut et caractéristique de manière interchangeable.

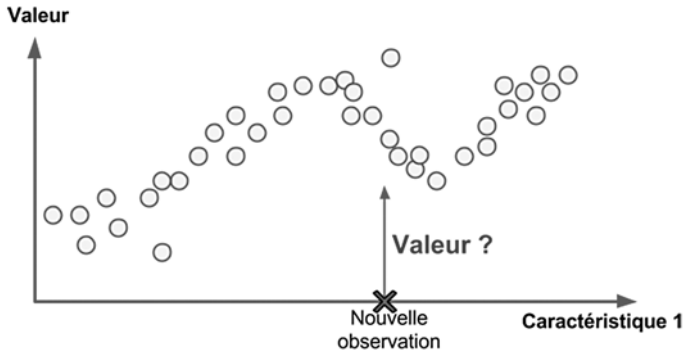


**Figure 1.6** – Un problème de régression : prédire une valeur en fonction d'une variable en entrée (il y a généralement plusieurs variables en entrée, et parfois plusieurs valeurs en sortie)

Notez que certains algorithmes de régression peuvent être utilisés également en classification, et inversement. Par exemple, la *régression logistique* s'utilise couramment en classification, car elle peut fournir une valeur correspondant à la probabilité d'appartenance à une classe donnée (par exemple, 20 % de chances qu'un courriel soit du spam).

Voici quelques-uns des plus importants algorithmes d'apprentissage supervisé (présentés dans ce livre) :

- K plus proches voisins
- Régression linéaire
- Régression logistique
- Machines à vecteurs de support
- Arbres de décision et forêts aléatoires
- Réseaux neuronaux[5]

### *Apprentissage non supervisé*

Dans l'*apprentissage non supervisé*, comme vous vous en doutez, les données d'apprentissage ne sont pas étiquetées (voir figure 1.7). Le système essaie d'apprendre sans professeur.

5. Certaines architectures de réseaux neuronaux peuvent être non supervisées, par exemple les auto-encodeurs ou les machines de Boltzmann restreintes. Elles peuvent aussi être semi-supervisées, comme dans le cas des réseaux de croyance profonds ou lorsqu'on utilise un pré-apprentissage non supervisé.