PYTHON
POUR LE
DATA SCIENTIST

Des bases du langage au machine learning

Emmanuel Jakobowicz

DUNOD

retirage corrigé avec nouvelle présentation (2019)
11 rue Paul Bert, 92240 Malakoff
www.dunod.com
ISBN 978-2-10-080162-6

TABLE DES MATIÈRES

DEUXIÈME PARTIE

La préparation et la visualisation des données avec Python

TROISIÈME PARTIE
Python, le machine learning et le big data

AVANT-PROPOS

◆ *Pourquoi un ouvrage sur Python en data science ?*

Python s'impose aujourd'hui dans de nombreux domaines comme un langage de programmation de référence. Dans le cadre de la science des données (data science), il s'avère être un outil de premier plan permettant d'articuler des projets complexes avec un langage « universel ». Il est aujourd'hui l'outil idéal pour mettre en place des prototypes et un grand allié du big data, du machine learning et du deep learning.

Cet ouvrage vous guidera afin d'aborder ce langage simple, et d'approfondir son utilisation en tant que data scientist.

L'objectif est de vous donner les outils nécessaires à la compréhension et à l'utilisation de Python en data science. Si vous êtes ou voulez devenir data scientist, la maîtrise de Python est aujourd'hui un avantage important et tend à devenir un prérequis.

Ce livre va vous permettre de prendre en main ce langage et d'en comprendre toutes les subtilités afin d'être capable de développer en Python des processus de data science.

Notre objectif n'est pas ici de présenter des méthodes de manière théorique mais plutôt d'illustrer l'application de méthodes de data science grâce au langage Python et à toutes ses spécificités.

◆ *À qui s'adresse ce livre ?*

Comme son titre l'indique, cet ouvrage s'adresse aux data scientists ou aux futurs data scientists. De manière plus large, il intéressera tous ceux qui, au quotidien, traitent des données et tentent de mettre en place des traitements de données dans des cadres concrets.

Plus spécifiquement, il est destiné :

- ✓ Aux professionnels de la Business Intelligence qui désirent analyser plus finement leurs données grâce à un nouveau langage.
- ✓ Aux statisticiens (notamment ceux qui utilisent SAS ou R) qui ont besoin de mettre en place des processus automatisés et qui désirent s'initier à Python.
- ✓ Aux développeurs qui souhaitent acquérir une compétence en data science avec l'utilisation de Python.
- ✓ Aux étudiants de niveau master en informatique ou en statistique qui désirent s'initier et se perfectionner au langage Python en travaillant sur des cas pratiques.
- ✓ Aux analystes et consultants amenés à gérer des projets et des analyses de données en Python dans le cadre de leurs activités.

Cet ouvrage ne demande pas de connaissances particulières. Néanmoins, la compréhension des notions de programmation et des connaissances rudimentaires en statistique faciliteront l'apprentissage.

◆ *Qu'est-ce que la data science et que fait le data scientist ?*

Les termes de data science et de data scientist sont assez récents. La data science ou science des données consiste à résoudre des problèmes complexes en utilisant de grands volumes de données.

Le data scientist est celui qui mettra en place et fera fonctionner des processus de data science. Son rôle est de faire le lien entre des problèmes métiers et des solutions par le biais d'algorithmes et d'infrastructures modernes. À ce titre, il devra avoir une vision assez large du cadre opérationnel, technique et théorique. On demande aujourd'hui au data scientist d'être très technique mais il faut garder en tête que sa principale plus-value sera sa capacité à résoudre des problèmes qui n'avaient pas de solutions opérationnelles jusqu'alors. Bien entendu, il doit avoir de bonnes connaissances en statistique, en machine learning, mais aussi des compétences techniques en développement et une bonne appréhension des environnements de traitement.

◆ *Comment lire ce livre ?*

Ce livre est organisé en trois grandes parties qui peuvent être lues séparément suivant les connaissances et les objectifs du lecteur.

La première partie propose d'acquérir les bases du langage Python et de découvrir les outils et les principes du langage. Elle se divise en deux chapitres. Le premier vous permettra de comprendre l'histoire de Python et tous les outils qui vous aideront à coder en Python (Anaconda, Jupyter notebooks, iPython…). Le second est une initiation au langage pour maîtriser ou réviser les bases de Python.

Dans la seconde partie, tous les outils pour traiter des données avec Python sont détaillés. En premier lieu, les structures de NumPy et Pandas pour le traitement des données sont introduites. Puis, nous étudierons la préparation des données et les statistiques élémentaires. Finalement, nous détaillerons l'utilisation des outils de visualisation des données (notamment Matplotlib, Seaborn et Bokeh).

La dernière partie présente des traitements plus complexes en data science, basés sur des exemples concrets. L'utilisation du machine learning ou la communication avec les environnements big data y sont abordés. Dans cette partie, nous traiterons des cas de machine learning avec Scikit-Learn, de deep learning avec Keras et de traitement de données non structurées. Le dernier chapitre de ce livre s'intéresse à la communication avec les infrastructures big data, notamment avec Apache Spark.

◆ *Les supports supplémentaires*

Cet ouvrage ne se limite pas à sa version papier, tous les contenus et des contenus supplémentaires sont disponibles en ligne.

L'ensemble des exemples présentés dans cet ouvrage sont disponibles sous forme de Jupyter notebooks. Ils sont accessibles directement à l'adresse :

https://www.github.com/emjako/pythondatascientist

À cette adresse, vous trouverez de nombreuses informations qui complètent la lecture de cet ouvrage.

Afin de les lire et de tester vos connaissances, nous vous conseillons d'installer Anaconda ou Python 3 sur votre ordinateur ou d'utiliser un service du type MyBinder.

Tous les détails sur les notebooks, et les environnements à installer sont détaillés dans le premier chapitre. Un processus d'installation pas à pas est disponible dans les annexes du livre.

◆ *Remerciements*

Je voudrais en premier lieu remercier Olivier Decourt sans lequel ce projet n'aurait pas pu voir le jour. Merci à mes relecteurs, tout spécialement Jean-Paul Maalouf, à l'équipe de Dunod et à mon éditeur Jean-Luc Blanc pour sa relecture minutieuse. Par ailleurs, je voudrais aussi remercier tous ceux qui m'ont permis de construire cet ouvrage au fil des années, tous ceux que j'ai pu former à Python avec Stat4decision qui m'ont fourni d'incessants challenges. Merci aussi à la communauté Python orienté data pour ses échanges passionnants, ses conférences innovantes et ses Meetups de grande qualité. Et pour terminer, je tiens tout spécialement à remercier mon épouse, Nathalie, et mes deux fils qui m'ont soutenu tout au long du long processus d'écriture de cet ouvrage.

PREMIÈRE PARTIE

Les fondamentaux du langage Python

Python est un langage simple qui doit vous permettre de vous concentrer sur la mise en place de processus automatisés en data science. Néanmoins, il est nécessaire de bien maîtriser son environnement. Cette première partie vous permettra avant tout de comprendre les spécificités du langage Python au sens large, ses subtilités et ses bases. Elle vous permettra aussi de mettre en place un environnement de travail orienté data science afin de maîtriser tous les aspects de Python en tant que langage pour traiter des données et automatiser des processus.

Python, ses origines et son environnement

Objectif

Dans ce chapitre, nous allons découvrir toutes les pièces nécessaires à la compréhension et à l'utilisation de Python. Vous y trouverez des explications sur les installations nécessaires, sur les bonnes pratiques et sur les outils développés pour vous simplifier la vie lors de votre utilisation du langage Python.

1. 1 HISTOIRE, ORIGINES ET ÉVOLUTION : DE LA NAISSANCE À LA VERSION 3.7

Python est né à la fin des années 1980, période extrêmement riche en création de langages de programmation. C'est Guido van Rossum qui a créé ce langage et c'est lui qui en parle le mieux (en 1996) :

« *Over six years ago, in December 1989, I was looking for a "hobby" programming project that would keep me occupied during the week around Christmas. My office... woul… closed, but I had a home computer, and not much else on my hands. I decided to write an interpreter for the new scripting language I had been thinking about lately: a desce ndant of ABC that would appeal to Unix/C hackers. I chose Python as a working title for the project, being in a slightly irreverent mood (and a big fan of Monty Python's Flying'Circus).* »[1]

1. Texte de Guido van Rossum écrit en 1996. Traduction : « Il y a six ans, en décembre 1989, je cherchais un projet de programmation pour mes loisirs sachant que mon bureau était fermé pendant la période de Nöel . J'avais un ordinateur à la maison et pas grand-chose d'autre. J'ai donc décidé de créer un interpréteur pour un nouveau langage de script. J'y pensais déjà depuis un moment : il fallait un héritier de ABC qui serait attractif pour les hackers utilisateurs de Unix/C. J'ai choisi Python comme nom pour ce projet car j'étais dans une humeur plutôt provocatrice (et aussi car j'adore le Monty Python's Flying Circus). »

1.1.1 Les origines et l'évolution du langage

Dès sa création, Python a été conçu pour mettre à la disposition du plus grand nombre un outil de développement simple et intuitif pour créer des scripts.

Les cadres de développement de Python

CWI : Le Centrum voor Wiskunde en Informatica d'Amsterdam est le berceau du langage Python qui se développe comme langage de script pour le projet Amoeba. Dans le cadre du CWI, la première version officielle de Python est publiée (0.9.0) en février 1991. La version 1.0 est aussi publiée dans cette structure.

CNRI : Le Corporation for National Research Initiatives à Reston dans l'état de Virginie est le second lieu dans lequel le langage Python s'est développé, à partir de 1995. Python devient un langage d'apprentissage dans le cadre du projet Computer Programming for Everybody.

BeOpen : Après la sortie de la version 1.6 en 2000, l'équipe de développement de Python change de base et crée l'équipe PythonLabs de BeOpen avec la version 2.0 de Python.

Python Software Foundation : Cette association à but non lucratif est créée en 2001. Elle va héberger le développement du langage Python à partir de la version 2.1. Elle est aussi à l'origine de la licence d'utilisation du langage ; la Python Software Foundation Licence. En 2008, l'équipe de développement de Python décide de publier la version 3.0, qui n'est pas rétro-compatible avec la version 2.

La Python Software Foundation héberge toujours les codes sources du langage, de nombreuses documentations, le répertoire des packages (PyPi) et gère la licence d'utilisation.

On voit bien que Python s'est cherché durant toutes les premières années de développement et que son orientation vers le traitement de la donnée n'a jamais été central dans son développement au cours de ces trente années.

1.1.2 La rencontre entre Python et la data science

L'émergence de la data science est récente, et ces nouveaux usages de la donnée ont souvent eu des difficultés à trouver des outils adaptés. En effet, le data scientist se doit d'être un bon développeur tout en restant un bon analyste de données. Il a fallu opter pour un outil qui permette de combiner cette demande de plus en plus forte de développement et d'automatisation (d'autant plus avec l'arrivée de l'intelligence artificielle et des objets connectés), avec la nécessité d'avoir une boîte à outils adaptée aux applications data.

De nombreuses pistes ont été explorées, notamment avec le logiciel R qui continue d'être une référence en data science mais qui pouvait paraître trop orienté vers la statistique pour des data scientists plus dirigés vers le développement. De nombreux

autres outils permettant de mettre en place des processus de data science ont été développés (la plupart propriétaires tels que Matlab ou SAS), mais il s'avère que Python (qui combine un langage puissant et extrêmement simple) a réussi à tirer son épingle du jeu.

La première avancée réelle a été la création du package **NumPy** (Numerical Python), qui est toujours aujourd'hui la pierre angulaire de l'écosystème Python pour la data science. D'autre part, la mise en place d'environnements Python orientés data avec notamment **Anaconda** a aussi permis à toute une communauté de se créer autour de Python et de la data. Finalement, IPython et ses notebooks (devenus Jupyter aujourd'hui) ont parachevé l'écosystème afin de proposer à des data scientists un langage simple mais extrêmement complet pour la data science. Cet environnement global a abouti au développement de nombreux packages et de nombreuses API, qui font aujourd'hui de Python le meilleur outil pour automatiser des traitements de data science.

1.1.3 L'évolution actuelle

Depuis quelques années, Python a pris une place extrêmement importante dans le monde du traitement des données. Alors qu'au début des années 2010, il semblait clair que dans le monde des outils open source de traitement de données, le logiciel R allait se tailler la part du lion, un changement important a eu lieu depuis quelques années. L'émergence de Python en tant que langage lié à la data science, au machine learning, au deep learning et à l'intelligence artificielle est extrêmement rapide. Grâce à une communauté extrêmement active sous la bannière PyData et à des évènements fréquents et nombreux (les PyCon, les PyData, les JupyterCon, les SciPyCon…), le développement du langage prend une tournure inattendue. Alors qu'on pouvait entendre en 2015 des développeurs dire que du point de vue du machine learning le développement de Python se calquait sur celui de R avec quelques mois de retard. Aujourd'hui, c'est R qui commence à calquer ses développements dans le domaine du machine learning, du deep learning et du big data, sur les packages développés en Python. En 2018, KDnuggets, un blog influent dans le monde de la data science, a même effectué un sondage auprès de milliers de data scientists dans le monde entier qui, pour la première fois, montre plus d'utilisateurs de Python que de R.

L'aventure de Python en data science est donc récente mais ne fait que commencer car c'est un langage qui s'adapte parfaitement à l'approche menée par un data scientist, lequel serait : « *meilleur en programmation qu'un statisticien et meilleur en statistique qu'un développeur.* »

1.1.4 Le futur de Python

Le futur proche de Python, c'est avant tout l'abandon de la version 2 et la généralisation de la version 3.

L'autre grand développement actuel concerne l'utilisation d'interfaces interactives avec Python comme langage de communication avec des API de plus en plus

évoluées. Nous parlerons un peu plus loin des widgets de Jupyter qui permettent de développer de manière interactive et de construire des interfaces en quelques lignes.

Python est de plus en plus utilisé comme un langage pour communiquer avec d'autres environnements. Ainsi Python permet de communiquer avec Apache Spark par l'intermédiaire de PySpark, ou avec des écosystèmes de deep learning tels que TensorFlow. Les calculs ne se font plus dans le moteur Python mais dans des moteurs bien plus puissants utilisant des infrastructures distribuées ou des calculs basés sur des GPU (Graphical Process Units). Cette tendance n'en est qu'à ses débuts avec la massification des données et les demandes de traitements en temps réels toujours plus fréquents.

Python ne peut pas répondre seul à ces challenges mais, combiné à d'autres outils, il permet au data scientist de gérer tout l'écosystème des traitements de la donnée qu'elle soit big, small, smart…

1. 2 PYTHON vs R vs LE RESTE DU MONDE

Si vous lisez cet ouvrage, vous avez forcément entendu parler d'autres outils en data science. On trouve aujourd'hui un écosystème extrêmement développé dans ce domaine avec des langage tels que R et Python mais aussi des logiciels plus classiques comme DSS de Dataiku, IBM-SPSS Modeler, SAS Entreprise Miner, Azure machine learning de Microsoft… En tant que data scientist, vous pourrez être amené à croiser certains de ces outils dans vos missions. Il est donc important de comprendre où se situe la force de chacun.

Nous nous concentrons ici sur Python et son utilisation par les data scientists. Alors pourquoi Python gagne du terrain sur la concurrence ?

Depuis quelques années, la tendance globale s'oriente vers plus de code dans les processus de traitement et Python répond parfaitement à cette demande. Ainsi, les outils du data scientist se différencient de plus en plus de ceux de l'analyste BI (business intelligence) qui eux sont de plus en plus intuitifs. Dans ce contexte, deux outils open source prennent les devants : Python et R.

Concernant les outils propriétaires, une tendance se généralise. Il s'agit de l'utilisation de langages tels que Python ou R à l'intérieur de ces outils dès qu'on aura besoin d'effectuer des opérations complexes. Ainsi, DSS de Dataiku, RapidMiner ou KNIME intègrent des modules pour développer en Python ou en R. Vos compétences en Python pourront donc être valorisées aussi dans le cadre de l'utilisation de ces outils.

1.2.1 R

R et Python sont aujourd'hui les bases indispensables du data scientist. De plus, l'évolution rapide des deux langages aboutit à une forme de compétition saine permettant de les rendre de plus en plus complets pour le traitement des données. Il existe néanmoins des différences qui sont notables et qui orienterons vos choix lors de la décision du langage à utiliser pour un projet.

R est basé sur un langage créé pour des statisticiens par des statisticiens, il s'appuie avant tout sur une approche descriptive et de modélisation des données. Les sorties de R sont des sorties dignes des logiciels de statistiques « classiques » avec de nombreux détails. Python n'est pas basé sur cette approche, c'est un langage de programmation visant à automatiser des processus en ne recherchant qu'à calculer le strict minimum. Dans le cadre des approches actuelles d'application de la data science, Python est adapté à d'autres besoins que R.

D'autre part, Python est un langage de programmation extrêmement simple se basant sur une syntaxe très lisible. Sa compréhension par de nombreux utilisateurs est facilitée, ce qui permet une meilleure interaction entre les différents métiers liés aux systèmes d'information (responsables, développeurs, data scientists, data ingeneer…).

Le tableau suivant s'intéresse à quelques points clé de vos traitements et tente de comparer l'utilisation que l'on peut en faire avec Python et R. Il s'agit d'une vision subjective et qui peut être souvent ajustée par l'utilisation de packages spécifiques.

Propriété	R	Python
Lisibilité du code	Moins lisible au premier abord	Plus lisible, notamment à cause de l'indentation
Rapidité	Langage interprété donc peu rapide (possibilité d'intégrer d'autres langages plus rapides Rcpp)	Langage interprété donc peu rapide (possibilité d'intégrer d'autres langages plus rapides C, cython…)
Préparation des données	Peu pratique avec sa version de base mais extrêmement efficace avec l'environnement Tidyverse	Efficace grâce au package Pandas
Capacités big data	De nombreuses API vers les infrastructures big data (parfois un peu de retard au développement)	De très nombreuses API vers les infrastructures big data avec un développement rapide
Gestion des données non structurées	Surtout spécialisé sur les données structurées	Très bien adapté notamment avec des packages comme NumPy ou Scikit-Image
Traitements statistiques	Clairement le meilleur outil	Des possibilités avec SciPy et Statsmodels mais pas son point fort
Analyse de données multivariées	Très bien adapté avec notamment des packages comme FactoMineR	Toutes les méthodes sont présentes mais manque d'outils de visualisation adaptés

Propriété	R	Python
Machine learning	Toutes les méthodes sont présentes mais manque de cohérence (existence du package caret pour créer de la cohérence)	Clairement en avance grâce au package Scikit-Learn
Deep learning	Rattrape actuellement son retard sur ce sujet	La combinaison des traitements de données non structurées et d'API vers les principaux environnements deep learning lui donne un réel avantage
Visualisation	Capacités fortes grâce à ggplot2	Bonnes capacités avec Matplotlib et Seaborn
Construction d'applications « web »	En avance grâce au package shiny	Quelques packages se mettent en place avec Bokeh et Dash
Mise en production de code	Parfois un peu complexe mais de plus en plus simplifié	Extrêmement simple grâce aux environnements et à la simplicité du code
La licence d'utilisation	GPL v3	PSF (proche BSD)

Pour résumer, Python est le langage de l'automatisation qui s'intègre parfaitement dans un cadre plus large de service informatique et qui s'adapte aux contextes de l'intelligence artificielle (données non structurées, environnements complexes). Il reste néanmoins plus faible que R pour un statisticien qui recherche un logiciel de statistique.

R et Python sont aujourd'hui complémentaires et vous seront nécessaires très fréquemment pour vos traitements.

Remarque – Comme vous avez pu le voir dans le tableau ci-dessus, Python et R ne sont pas basés sur la même licence d'utilisation. Ce sont bien entendu tous les deux des langages open source pour lesquels vous pouvez utiliser ces outils dans vos activités personnelles et professionnelles de manière libre. Il faut néanmoins garder en tête une grande différence. La licence de Python est une licence libre « classique », elle vous permet de réutiliser du code source, de le modifier, de le commercialiser et d'en faire tout usage sans obligation d'ouverture de votre code. Il s'agit du type de licence utilisée classiquement pour les langages de programmation. L'ensemble de l'écosystème Python pour la data est basé sur cette licence. D'un autre côté, R est basé sur une licence plus contraignante, il s'agit de la licence GPL v3. Celle-ci vous donne des responsabilités par rapport à la communauté de développement. En effet,

si vous utilisez du code source R et que vous le modifiez pour le distribuer, vous serez forcé de rendre ce code accessible aux utilisateurs (open source). En allant encore plus loin, cette licence est « contaminante », c'est-à-dire que si vous intégrez du code sous licence à votre code, votre code passe sous licence et doit être open source. Ce point peut effrayer certains développeurs qui se tournent parfois vers Python. Cette différence entre les deux langages que sont R et Python traduit une différence de développement du langage entre les deux communautés. Les développeurs de R sont plus dans une idée de « forcer » les utilisateurs avancés à contribuer au projet alors que les développeurs Python parient sur une utilisation très large du langage qui aboutira à plus de contributions pour le développement du langage.

1.2.2 Outils de traitement de flux

Les autres outils de la data science sont pour la plupart des facilitateurs, ainsi la majorité de ces outils se basent sur la création de flux (DSS de Dataiku, RapidMiner, KNIME, IBM-SPSS Modeler…) en utilisant votre souris. Ils simplifient la vie des utilisateurs et, suivant vos besoins, pourront vous faire gagner du temps. Cependant, tout ce que peuvent faire ces outils peut être fait directement avec Python et ils intègrent tous des moyens d'ajouter du code en Python dans les flux.

Ces outils vont vous permettre de créer des analyses, allant de l'acquisition à l'analyse des données en quelques clics. Par exemple, vous pourrez aller chercher des données dans différents formats dans un premier niveau, vérifier et fusionner ces données au niveau suivant, transformer les données, les découper et appliquer et valider des modèles prédictifs sur ces données. Tout ceci est inclus dans un seul flux, comme nous pouvons le voir dans la figure 1.1 (il s'agit ici de KNIME).

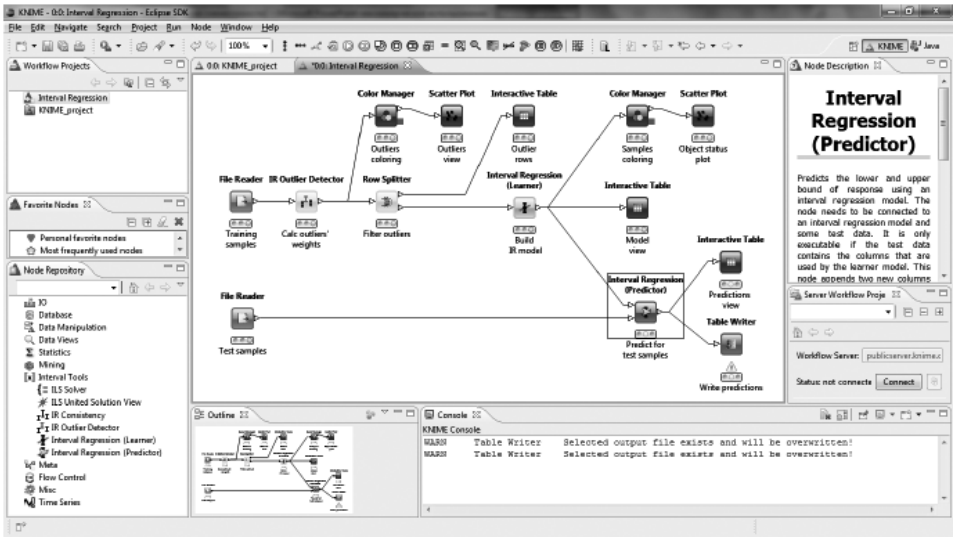

Figure 1.1 – Exemple de traitement de flux en data science.

Python sera donc très différent de ce traitement visuel mais permettra assez facilement de reproduire ce type de flux sous forme de code. De plus, dès que les « briques » de vos traitements deviennent plus complexes, l'utilisation de Python à l'intérieur de chaque brique devient nécessaire.

1.2.3 SAS

Nous allons nous attarder ici sur un point spécifique car il concerne de nombreux utilisateurs dans le monde professionnel. Depuis quelque temps, de nombreuses entreprises décident de migrer leurs infrastructures de l'historique SAS aux nouveaux venus (R et Python). Il est sans doute utile de clarifier quelques points à ce sujet.

SAS est un logiciel propriétaire spécialisé dans le traitement de données. Il cumule près de quarante ans d'expérience dans ce domaine et ne peut pas être remplacé de manière simple dans des entreprises qui se sont souvent basées sur cet outil pour le traitement de leurs données. Néanmoins, le passage vers un outil comme Python peut se justifier pour plusieurs raisons :

- ✓ D'un point de vue économique : c'est souvent la première raison invoquée. En effet, les licences d'utilisation de SAS coûtent très cher. Mais le changement aussi va coûter cher, il demande de modifier les façons de travailler et une plus grande prise en charge des infrastructures qu'auparavant.
- ✓ D'un point de vue technologique : c'est le point le plus important. Le passage à Python va permettre d'accéder à des méthodes de machine learning, de deep learning et de traitement de données non structurées bien plus puissantes qu'avec SAS.

Il faut garder en tête que ce changement va amener un certain nombre d'inconvénients. Principalement le fait que Python est un langage qui va charger les données en mémoire dans votre machine alors que SAS utilisait un système intermédiaire de tables stockées sur de la mémoire physique. Il vous faudra donc modifier vos habitudes et passer par des requêtes plus élaborées vers vos bases de données. Ensuite, le processus de transformation et de traitement des données sera largement simplifié. Vous pouvez voir beaucoup de détails sur la documentation de Pandas, l'une des composantes centrales de Python, à l'adresse suivante :
https://pandas.pydata.org/pandas-docs/stable/comparison_with_sas.html

Python va donc vous apporter une flexibilité importante mais il faudra modifier votre approche de la gestion des données et du développement de code avec Python.

1.2.4 Les autres langages

Nous avons comparé Python a des outils de data science mais une autre comparaison intéressante peut se faire par rapport à d'autres langages de programmation. Aujourd'hui de nombreux langages sont disponibles. On pourra citer dans l'univers de la data : Julia, MatLab, C/C++, Scala. Ces langages ont tous leurs spécificités, nous pourrons les classer dans deux catégories :

- ✓ Les langages interprétés tels que MatLab, Julia et Scala qui sont des alternatives crédibles à Python dans un cadre orienté data science.

- ✓ Les langages compilés tels que C, C++, Java qui entrent dans une autre catégorie et qui s'adressent à des développeurs plus chevronnés.

Dans certains cas, ils sont plus efficaces que Python mais ils ne possèdent pas un environnement de packages et d'API aussi développé que celui de Python.

1.3 COMMENT DÉVELOPPER EN PYTHON ?

C'est une question récurrente : « quel *logiciel* utiliser pour coder en Python ? » La réponse n'est pas simple et nous devons d'abord déconstruire certains préjugés.

Python n'est pas un logiciel, c'est un langage de programmation ! Donc si vous voulez coder en Python, il suffit de l'installer, de lancer un terminal de commande et de taper le mot python. Vous pouvez commencer à coder. Bien entendu, ça n'est pas ce que vous allez faire au quotidien.

La solution la plus simple pour débuter avec Python est d'installer Anaconda (http://www.anaconda.com). Il s'agit d'un environnement de développement multi-plateforme (Linux, Mac, Windows), intégrant un interpréteur Python mais aussi de multiples fonctionnalités, notamment l'interpréteur IPython, l'IDE Spyder, les notebooks Jupyter mais aussi des centaines de packages permettant de débuter rapidement à coder en Python. Si vous commencez votre voyage avec Python, je ne peux que vous conseiller de partir avec Anaconda. Je reviendrai en détail sur les procédures d'installation de ces différentes composantes dans la dernière partie de ce chapitre.

Autre point fondamental : lorsque nous faisons du Python, nous développons des programmes informatiques. Que votre objectif soit l'analyse d'une petite base de données ou la construction d'un programme complexe d'intelligence artificielle, il faudra vous poser un certain nombre de questions liées au développement informatique. Ces points ne sont pas spécifiques à Python mais sont centraux dans votre processus de travail en tant que data scientist.

Développer des outils en Python est un processus de développement simplifié qui permet une réelle agilité dans le développement mais nécessite tout de même des principes fondamentaux qui vont s'intégrer dans le cadre spécifique de la data science.

Le processus de traitement d'un problème en data science passe par un certain nombre d'étapes :

1. La définition du problème métier
2. La transformation du problème métier en un problème data science
3. La recherche et la découverte des sources de données
4. La préparation et la mise en forme des données
5. La construction et l'ajustement de modèles
6. La validation des modèles
7. La réponse à la question et la mise en production des modèles

Toutes ces étapes sont centrales dans le processus de travail du data scientist. Le développement va intervenir à différentes étapes et Python sera un outil qui vous accompagnera au mieux lors de toutes ces étapes.

Votre façon de développer dépendra surtout des points (1) et (7). Votre processus de développement dépendra :

- ✓ Du **commanditaire** ou du porteur de projet : c'est lui à qui il faudra répondre. En fonction de son rôle, de son niveau d'expertise, de ses attentes, vous ne développerez pas de la même manière. Imaginons que la problématique soulevée émane du CEO de votre entreprise, pour répondre à cette question, il faudra mettre à sa disposition des outils qui seront le moins technique possible, il pourra s'agir d'un rapport « classique » ou d'un outil interactif. Si cette question vient d'un collègue data scientist, des scripts ou des notebooks seront souvent suffisants pour fournir une réponse satisfaisante.
- ✓ Du **livrable attendu** : s'il s'agit d'une simple demande ponctuelle sur une analyse à effectuer, il vous suffira de développer un script simple et de construire un rapport « figé ». En revanche, si l'attendu (ce qui est de plus en plus le cas) est un outil interactif ou la mise en production d'un algorithme avancé de machine learning, vous devrez mettre en place un processus extrêmement strict de développement.

Pour bien gérer votre projet de data science au niveau du développement, vous allez donc vous baser sur un certain nombre de points :

1. Détermination des besoins et des objectifs :

   On va essayer de répondre lors de cette étape à un certain nombre de questions : que doit faire le logiciel ? Dans quel cadre va-t-il servir ? Qui seront les utilisateurs types ?

   À partir des réponses, la rédaction d'un cahier des charges avec le commanditaire du logiciel est nécessaire.

2. Conception et spécifications :

   On va alors rentrer dans le détail des attentes des utilisateurs. Quelles sont les fonctionnalités du logiciel ? Avec quelle interface ?

   C'est souvent à cette étape qu'on choisit le langage de programmation approprié. Il peut s'agir d'une combinaison de langage (pour le front, le back…).

3. Programmation :

   C'est la partie qui nous intéresse dans cet ouvrage mais ce n'est pas la plus importante du processus global. Si les étapes précédentes sont mal menées alors nous ne pourrons pas aboutir aux résultats attendus.

4. Tests :

   Un code a toujours des bugs ! Que ce soit des bugs de codage ou des cas que l'on n'avait pas prévu lors du codage, il faut tester son code. Moins les utilisateurs sont techniques, plus il faut tester son code.

5. Déploiement :

   On n'a pas toujours la nécessité de déployer son code ailleurs que sur son ordinateur. Mais beaucoup de projets de data science impliquent le passage