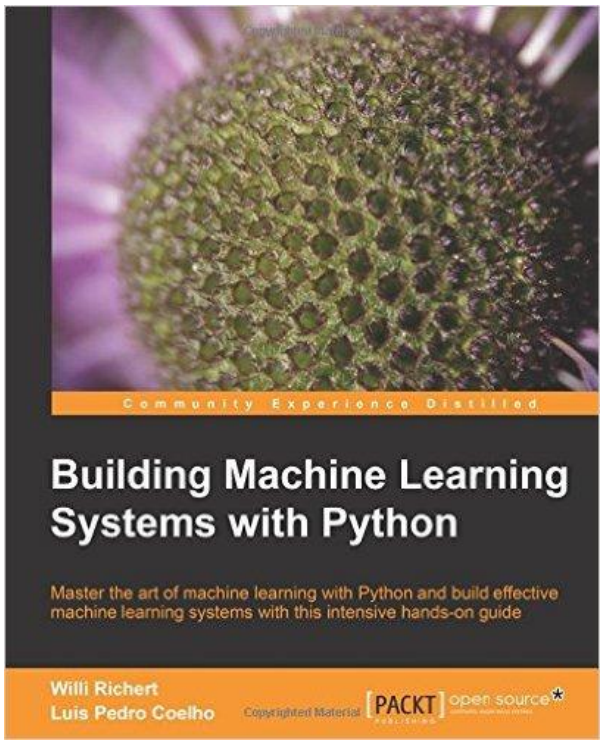
Building Machine Learning Systems with Python (2nd Edition)

Luis Pedro Coelho, Willi Richert

Packt Publishing, Mars 2015.



J'ai toujours eu des réticences à acheter et conseiller à mes étudiants des ouvrages consacrés à des outils. Généralement, ils prétendent couvrir un très large panel d'approches en quelques centaines de pages. A la sortie, on se rend compte qu'ils traitent de manière très superficielle les méthodes sous-jacentes qu'ils essaient d'illustrer. Et, de toute manière, on trouvera toujours sur le web des tutoriels en français ou en anglais, qui décriront les opérations et les sorties des logiciels de manière autrement plus approfondie. De plus, la diffusion croissante de nombreuses vidéos de démonstration sur la plate-forme d'échange YouTube modifie la donne concernant ce type de document à vocation pédagogique.

Pourquoi alors faire une fiche de lecture sur « Building Machine Learning Systems with Python » qui s'inscrit finalement dans cette lignée des ouvrages centrés sur les outils ?

Tout simplement parce qu'il nous permet de cerner le champ des possibles en matière de Machine Learning sous Python. Le livre ne prétend pas à l'exhaustivité. Il ne constitue

certainement pas un ouvrage sur l'apprentissage automatique. L'objectif serait plutôt de titiller la curiosité du lecteur. Pour ma part, j'y ai surtout vu une source d'inspiration intéressante me permettant de faire évoluer mes Cours / Travaux Dirigés pour mes enseignements de Data Mining - Data Science en Master Statistique et Informatique.

De fait, une fois assimilé ce cadrage, la tonalité pragmatique adoptée par les auteurs me convient parfaitement. Ils annoncent dans la préface rechercher le juste équilibre entre l'informaticien (le programmeur adepte de Python) et l'adepte du Machine Learning (le statisticien adepte des techniques exploratoires) : « *This book is for Python programmers who want to learn how to perform machine learning using open source libraries… This book is also for machine learners who want to start using Python to build their systems* ». Ca tombe bien. La recherche du juste équilibre entre les statistiques au sens large (incluant l'apprentissage statistique, le machine learning, le data mining) et l'informatique a toujours été un de mes crédos.

(1) Le chapitre 1, « **Getting Started with Python Machine Learning** », introduit la démarche Machine Learning via une étude de cas. La manipulation des vecteurs et matrices - des structures de données fondamentales pour la suite - du package **NumPy** tient une place importante. Pas assez à mon goût ceci dit. Les auteurs enchaînent ensuite avec la recherche de la « meilleure » régression (librairie **SciPy**) sur un jeu de données. L'exemple montre surtout que la modélisation statistique est un processus itératif où l'on procède souvent par essai-erreur (**Note :** principalement par essai-erreur ? je n'irai pas jusque là, il ne s'agit pas non plus de tester tout et n'importe quoi).

(2) Le chapitre suivant, « **Classifying with Real-world examples** », aborde la classification supervisée (ou classement). On commence avec l'incontournable base des IRIS de Fisher. Après une représentation graphique (package **PyPlot**) des données où les trois classes apparaissent nettement dans certaines situations, les auteurs proposent une approche intuitive fondée sur les frontières déduites des positions relatives des nuages de points dans un premier temps. La démarche n'est pas très usuelle, surtout pour un premier contact. Mais le discours se tient, et des notions importantes telles que l'évaluation des performances à l'aide de la validation croisée sont introduites.

On entre dans le vif du sujet dan un deuxième temps. Les auteurs rappellent que la modélisation peut se décomposer en trois tâches en machine learning. (1) Le choix de la structure du modèle (*representation bias* en anglais) correspond à la fonction avec laquelle nous représentons la liaison entre les variables prédictives et la variable cible. (2) La procédure de recherche (*search bias*) retrace le mode de recherche de la solution « optimale » sur un ensemble de données. Ainsi, deux modèles peuvent reposer sur le même système de représentation (ex. classifieur linéaire), mais reposer sur des valeurs de paramètres différents (ex. un SVM linéaire et une régression logistique produiront des coefficients estimés différents sur le même échantillon d'apprentissage). (3) La fonction de gain ou de perte permet d'évaluer la qualité de la modélisation. Les auteurs évoquent notamment la catégorisation d'e-mail où classer en spam un message sain (le serveur élimine le message, vous ne saurez jamais que vous avez gagné au loto par ex.) « coûte » différemment de laisser passer un message délictueux (vous l'ouvrez par mégarde, la liste de vos dernières commandes sur un site olé olé se retrouvent du jour au lendemain sur le mur Facebook de votre mère, il va falloir vivre avec ça après…).

Les données SEEDS illustrent la démarche. Il s'agit d'identifier des variétés de blés à l'aide de la méthode des plus proches voisins (package **scikit-learn**). L'analyse est somme toute très classique. Je note quand même l'utilisation de l'objet validation croisée de la classe KFold() de scikit-learn (page 44). L'outil s'avère très pratique pour générer les échantillons d'apprentissage et de test utilisés durant le processus.

(3) Le 3ème chapitre, « **Clustering - Finding Related Posts** », aborde la classification non supervisée (clustering), en mixant la présentation avec le traitement des données textuelles (text mining). De fait, dans une première étape, l'ouvrage s'appuie sur un petit ensemble de données jouet (traduction maison de ''toy dataset'') pour montrer comment transformer une collection de documents en une table documents x termes (individus x variables) avec la représentation en sac de mots (bag of words). Les classes dédiées de scikit-learn fait merveille. Des éléments de nettoyage des termes sont examinés (ex. élimination des stopwords, le stemming avec la librairie **NLTK**, le choix de la pondération, etc.). Couplé avec l'étude des différentes mesures de distances entre documents (euclidienne, cosinus, etc.), cet exemple m'a fourni des pistes intéressantes pour monter un TD « amusant » de text mining sous Python.

Dans une seconde étape, les auteurs présentent l'algorithme K-Means de classification automatique (clustering). Le texte est intuitif mais particulièrement lapidaire (4 pages, avec des graphiques principalement). Heureusement qu'on a été prévenu à l'avance que l'ouvrage ne développait pas en profondeur les méthodes…

Bon, ils se rattrapent dans une troisième étape avec un exemple on ne peut plus réaliste, où on nous propose de traiter les données « newsgroup » avec 18.826 posts issus de 20 différents groupes de discussion. Les données sont directement intégrées dans le package scikit-learn. Mais nous pouvons également les charger à partir du web : http://mlcomp.org/datasets/379. L'ouvrage s'en tient à des opérations basiques. Mais l'on y perçoit d'excellents sujets de TD potentiels. Il y a matière à faire des choses attrayantes.

(4) Le chapitre 4 traite du « **Topic Modeling** ». Là également, peu de place est accordée aux aspects théoriques. Tout juste comprend-t-on qu'à partir d'une matrice documents-termes, on cherche à rassembler ces derniers dans des « topics » (thèmes). Un terme peut appartenir à plusieurs thèmes à différents degrés. On peut par la suite représenter les documents dans l'espace des topics. Quel intérêt me direz-vous ? D'une part, l'interprétation des résultats est améliorée parce qu'on peut donner un sens aux topics après analyse des mots qui les composent. D'autre part, nous bénéficions d'une réduction drastique de la dimensionnalité. En effet, si le nombre de termes peut atteindre plusieurs milliers dans une matrice documents-termes, le nombre de topics en revanche dépasse rarement quelques centaines. Les calculs seront plus rapides sur la matrice documents-topics. Ils seront également plus stables pour les opérations courantes de text mining (ex. calculs de distances entre documents, mise en œuvre des techniques de data mining, etc.).

Pour montrer la puissance de l'approche, les auteurs proposent d'analyser tout Wikipedia (*rien que ça !*) dont l'image est téléchargeable en ligne (http://dumps.wikimedia.org/). Il apparaît à l'issue des traitements (*c'est dont possible !*) que le thème musical est le plus populaire sur les articles de Wikipedia, en anglais tout du moins (5.5% de tous les mots sur Wikipedia sont relatifs à la musique, page 91).

Le « topic modeling » ne se limite pas au texte. Dans la conclusion du chapitre, les auteurs indiquent que l'approche peut être étendue à l'analyse d'image. Les évolutions et les applications sont nombreuses.

(5) Le chapitre 5 s'intitule « **Classification - Detecting Poor Answers** ». Il se présente comme une étude de cas. L'objectif est de discerner automatiquement les bonnes et mauvaises réponses sur les sites de « Questions - Réponses » (Q&A - *Question and Answer*) tels que StackOverflow, en s'appuyant sur les appréciations exprimées par les internautes. Les données sont accessibles en ligne (https://archive.org/details/stackexchange). On s'intéresse en particulier au fichier « stackoverflow.com-Posts.7z » (7.9 Go sur le site de téléchargement à ce jour - 19/02/02016).

La première étape consiste bien sûr à préparer et nettoyer convenablement le fichier pour le rendre exploitable. La variable cible notamment est binarisée en traitant le score attribuée à une réponse : elle vaut 1 lorsque le score est supérieur strictement positif, 0 dans le cas contraire (page 100). C'est un choix comme un autre. Il faut qu'elle soit simplement bien comprise lorsque l'on devra lire les résultats. On aurait pu également utiliser directement le score fourni par les internautes. Mais nous serions sortis du cadre de la classification supervisée binaire dans ce cas.

Les auteurs démarrent ensuite un processus incrémental où ils commencent avec une seule variable prédictive (nombre de liens) avec un k-plus proches voisins. Ils font évoluer la modélisation en rajoutant d'une part de nouveaux descripteurs, et en modifiant le nombre de voisins. Une réflexion intéressante sur l'identification de la source de l'erreur (biais ou variance) à travers les courbes comparées des erreurs en test et en apprentissage lorsque la taille de l'échantillon augmente est menée. A la sortie, on switche sur une régression logistique où l'optimisation porte sur la constante d'apprentissage (**Note :** la régression logistique de scikit-learn s'appuie sur une descente du gradient).

En fin de chapitre, les auteurs essaient de dépasser le simple taux de succès en s'intéressant au rappel et la précision issus de la matrice de confusion. On peut améliorer l'un ou l'autre en jouant sur le seuil d'affectation (page 119).

(6) Le chapitre 6 est consacré à l'analyse des sentiments sur twitter (« **Classification II - Sentiment Analysis** »), un des sujets à la mode du moment. La base des tweets étiquetés manuellement par Nick Sanders (http://www.sananalytics.com/lab/twitter-sentiment/) est utilisée. Il a défini 4 niveaux de messages : non pertinent, négatif, neutre et positif. A l'instar du chapitre précédent, l'objectif est toujours une catégorisation par le contenu.

Les auteurs souhaitent s'appuyer sur le modèle bayésien naïf (modèle d'indépendance conditionnelle cette fois-ci) qui est décrit assez longuement (par rapport aux autres méthodes). Ils essaient d'améliorer les performances en jouant notamment sur la longueur des n-grammes (nombre de mots à prendre en compte simultanément - https://fr.wikipedia.org/wiki/N-gramme), sur la pondération, etc.

On passe au niveau au-dessus dans la seconde partie du chapitre. Les messages sur twitter sont assez particuliers, avec de nombreuses abréviations. Le nettoyage des textes est très important (page 146). Tenir compte des informations linguistiques l'est également, en distinguant la nature des mots (verbe, nom, etc.) (page 148), en attribuant une note de tonalité aux mots (page 150, en s'appuyant sur les associations disponibles sur SentiWordNet - http://sentiwordnet.isti.cnr.it/). Mixer tout ce savoir faire passe par la définition d'une nouvelle classe LinguisticVectorizer() dont le code source est décrit dans le livre (**Note :** on perçoit les grandes lignes du code, mais franchement aller dans le détail demande quand même un certain courage…). Franchement, les gains ne me paraissent pas très tranchants. Le processus a surtout le mérite de nous montrer ce qu'il est possible de faire en la matière. Je connaissais de nom SentiWordNet. A la lecture de cette section, je me suis intéressé à la notion de « sentiment score » en opinion mining.

(7) On redescend un peu sur terre (pour un statisticien en tous les cas) dans le chapitre suivant consacré à la « **Regression** ». On nous présente très succinctement les régressions simples et multiples. Les auteurs embrayent par la suite sur la régression pénalisée (ridge, lasso et elasticnet [un mix de ridge et lasso]) pour appréhender la forte dimensionnalité.

L'affaire prend un tour concret (page 168) lorsqu'il s'agit de traiter des données correspondant à une matrice document - termes avec 16.807 observations et (surtout) 150.360 descripteurs. Lancer une régression basée sur les algorithmes usuels d'économétrie n'est pas vraiment raisonnable dans ce contexte. La descente du gradient implémentée dans scikit-learn fait merveille ici. Les auteurs traduisent la modélisation en une démarche de recherche de paramètres optimaux (les paramètres de régularisation d'elasticnet) par validation croisée (**Note :** ce n'est pas parce qu'on travaille en validation croisée que nous sommes à l'abri du sur-apprentissage, il faut toujours être prudent dans ce genre de processus). Pour ma part, je suis surtout impressionné par les capacités de calcul du couple Python et scikit-learn. Appréhender une telle base n'est pas une mince affaire. On peut de

surcroît paramétrer le nombre de processeurs à exploiter sur les machines multi-processeurs (ou multi-cœurs ? il faut que je vérifie cela).

(8) Le chapitre 8 est consacré aux « **Recommandations** ». Le but est de produire un système de recommandations personnalisées sous forme de suggestions (de produits à acheter par ex.) à partir des comportements (les achats effectués) ou des avis exprimés par les internautes (les notes attribuées). Ce thème est également appelé « Filtrage collaboratif » dans la littérature, dans le sens où les utilisateurs (internautes) collaborent avec le système pour améliorer l'offre de service (page 175).

Dans une première partie, les auteurs s'intéressent à la notation des produits. Le point de départ est une matrice avec en ligne les utilisateurs, en colonne les produits. A l'intersection ligne-colonne, nous observons la note assignée par l'utilisateur à un produit. La matrice est forcément constellée de données manquantes puisque les utilisateurs n'achètent pas tous les produits. On souhaite calculer la note d'un individu pour un produit afin d'évaluer son appétence. La première approche est très fruste. Il s'appuie sur la recherche de profils similaires et, pour un individu à prédire, le système lui assigne la moyenne des notes des individus qui lui ressemblent sur les achats déjà effectués. La seconde approche, plus évoluée, utilise la régression. Notons que l'analyse peut être centrée sur les utilisateurs (qui ressemble à qui) ou sur les produits (quels sont les produits qui récoltent des notes liées entre elles, on travaille sur la matrice transposée dans ce cas).

La seconde partie du chapitre s'intéresse aux règles d'association. Elles visent à produire des règles que l'on voit souvent sur des sites de vente en ligne : « les individus qui ont acheté tels ou tels livres ont également acheté tel livre ». La présentation est toujours très succincte mais, fait très curieux, comme la méthode n'est pas présente dans scikit-learn ou *les packages étudiés dans l'ouvrage*, les auteurs se proposent de la programmer eux-mêmes à l'aide de quelques lignes de codes bien sentis sous Python. La base « retail » de Tom Brijs est utilisée pour illustrer l'approche (http://fimi.ua.ac.be/data/).

(9) Le chapitre 9 s'intitule « **Classification - Music Genre Classification** ». Il nous fait sortir de notre zone de confort (« *This chapter wil show how we can come up with a decent classifier in a domain that is outside our comfort zone* », page 199). En effet, on souhaite catégoriser

automatiquement en genres musicaux (classique, jazz, etc.) de fichiers de musique anonymisés. L'objectif est de pouvoir déployer le modèle en rattachant automatiquement une nouvelle musique (un individu supplémentaire) à son genre. Ce ne sont pas des applications que l'on rencontre souvent dans le data mining (**Note :** je savais que ce genre d'applications existait, mais c'est bien la première fois que j'ai le loisir de compulser une analyse menée de bout en bout) même si, finalement, on sait à peu près ce qu'il faut faire : extraire des vecteurs de caractéristiques à partir du fichier de musique, puis les utiliser comme descripteurs dans la modélisation prédictive où la variable cible est le genre musical.

Le chapitre commence par l'appréhension des données de musique. Les fichiers « GTZAN dataset » accessibles en ligne (http://opihi.cs.uvic.ca/sound/) sont convertis en fichiers WAV lisibles directement par une procédure de la librairie **SciPy**. Les signaux musicaux sont traduits en vecteurs numériques via une transformation de Fourier rapide (FFT - https://fr.wikipedia.org/wiki/Transformation_de_Fourier_rapide). Je le dis souvent à mes étudiants, cette étape est nécessairement présente lors du traitement des données non structurées. L'enjeu réside dans la perte d'information induite par cette opération. Dans le cas présent, le premier taux de reconnaissance global des 6 genres musicaux est assez mitigé. Les auteurs passent ensuite (la justification n'est quand même pas très étayée même si l'on comprend bien qu'ils entendent s'intéresser à la structure de l'erreur) à une formulation binaire (un genre contre les autres, 6 classifieurs à construire) et expriment la qualité des modèles via une courbe ROC et le calcul de l'indicateur AUC. Mis à part la reconnaissance du genre « classique », les modèles sont visiblement perfectibles (page 213).

Les premiers descripteurs ne sont pas très performants. Nous passons dans une deuxième phase à l'extraction de nouvelles caractéristiques : le MFC (Mel Frequency Cepstrum - https://en.wikipedia.org/wiki/Mel-frequency_cepstrum). Franchement, je n'y connais rien. Mes goûts et connaissances dans le domaine sont assez basiques. La notion qu'il faut avant tout retenir est que nous essayons d'extraire de nouvelles caractéristiques dans les données musicales afin d'améliorer le pouvoir prédictif des modèles. Un musicien saura nous dire le type d'information que nous essayons de capter ce faisant. Les connaissances métiers s'avèrent essentielles à ce stade.

Force est de constater que l'opération s'est avérée fructueuse puisque la qualité prédictive fait un bon en avant spectaculaire (pages 216 et 217). Le gain montre bien - si besoin était -

que l'extraction des descripteurs est une tâche essentielle dans le traitement des données non structurées.

(10) Nous nous attaquons à l'analyse d'images dans le chapitre 10 : « **Computer Vision** ». La trame est la même que pour toute analyse de données non structurées : appréhender les fichiers (librairie **mahotas**), les préparer avec des primitives de traitement d'images (ex. thresholding, gaussian blurring, …), extraire les caractéristiques, mettre en œuvre les techniques de data mining. Ici aussi, la performance prédictive dépend fortement de la qualité des informations que nous arrivons à soutirer aux données originelles. Les auteurs utilisent une base jouet de 30 images x 3 classes (immeubles, paysages et images de textes). Les descripteurs sont extraits avec les objets ''features'' de la librairie Mahotas (http://mahotas.readthedocs.org/en/latest/api.html). Le premier taux de reconnaissance est de 81.1%.

Les auteurs comptent améliorer le système en intégrant leur propre fonction d'extraction de descripteurs, les histogrammes de couleurs en l'occurrence. La puissance de Python s'exprime pleinement dans ce cadre puisque quelques lignes de code suffisent pour obtenir les nouveaux vecteurs de caractéristiques (page 231). L'opération est payante avec un nouveau taux de 95.6%.

Lorsque l'on traite des problèmes plus complexes où les informations de texture et de couleur ne suffisent plus à faire la différence. Une piste d'amélioration possible consiste à s'intéresser aux caractéristiques locales des images c.-à-d. calculées sur certaines zones de l'image (SURF - Speeded Up Robust Features, page 235).

Le chapitre est conclu par une énumération des librairies pour traitement d'images pouvant s'interfacer avec Python : mahotas que nous avons déjà cité, mais aussi **Skimage** (scikit-image) et **OpenCV**. Tous produisent des structures de données NumPy. Combiner les caractéristiques qu'elles génèrent enrichit forcément l'analyse.

(11) Nous revenons à des aspects plus méthodologiques dans le chapitre 11 consacré à la réduction de la dimensionnalité (« **Dimensionality Reduction** »). Deux approches sont mises en avant : la sélection de variables (selecting features) et la construction de variables (feature extraction). La problématique est d'importance dans les domaines explorés dans cet

ouvrage. En effet, nous constatons qu'il est possible d'extraire des caractéristiques en masse à partir des données non structurées, qu'il s'agisse de texte, de son ou d'image. Nombre d'entre eux sont redondants ou tout simplement non pertinents. Il faut traiter l'excès de dimensionnalité parce qu'il peut perturber les techniques de data mining, il mène au sur apprentissage (le nombre de paramètres à estimer explose), et il grève inutilement les temps de calculs (page 241).

Pour la sélection de variables, les auteurs effleurent les techniques de filtrage. Il faut évacuer d'une part les variables redondantes (mesurée à l'aide de la corrélation ou de l'information mutuelle), d'autre part les variables non pertinentes (non liées à la variable cible dans une démarche prédictive). Ils parlent un peu également des approches wrappers. Dans une seconde partie, la construction de variables est abordée. L'objectif est de produire un nombre réduit de descripteurs synthétisant au mieux les variables initiales en utilisant des techniques factorielles par exemple.

Je suis forcément déçu à la lecture de ce chapitre. Ce sont là des domaines dans lesquels j'ai beaucoup œuvré. Nombre de raccourcis faits par les auteurs parsemés de code Python ici et là me hérissent forcément. Il faut surtout retenir l'idée que la très grande dimensionnalité est un problème potentiel, d'autant plus prégnant que les descripteurs sont générés en grand nombre de manière automatique.

(12) Le dernier chapitre est consacré à la gestion de la volumétrie, « **Bigger Data** ». L'écueil de la question « à partir de quand peut-on parler de big data ? » est évacuée d'emblée : « *The expression ''big data''… means that the data has been growing faster than processing power* ». Ce qui implique une adaptation des méthodes existante de data mining, soit méthodologiquement, soit dans leur implémentation. La gestion des données devient un problème intrinsèque de la tâche de modélisation.

Les auteurs mettent l'accent sur l'outil **jug** que l'un d'entre eux a développé. Il sait stocker sur disque les informations déjà calculées pour éviter le recalcul d'une exécution à l'autre (je ne connaissais pas le terme ''mémoïsation'' - https://fr.wikipedia.org/wiki/Mémoïsation), et dispatcher les calculs lorsqu'il est placé dans le contexte d'une machine multi-processeur ou d'un cluster de machines. La mise en œuvre sur un petit exemple jouet est décrite dans un premier temps. Les analyses du chapitre 10 « Computer Vision » sont repris dans un second

temps. Une réécriture du code Python décomposant les traitements sous forme de tâches est nécessaire. Le programme imprimé dans le livre n'est pas vraiment limpide à mon goût. Il faudra faire plusieurs essais pour comprendre les éléments clés du dispositif.

Le service Amazon (Amazon Web Services) est ensuite présenté. Il nous propose des capacités de calcul en ligne avec un espace de stockage. On s'intéresse en particulier au dispositif EC2 (Elastic Compute Cloud). On nous montre comment installer les machines virtuelles et, surtout, comment aménager notre environnement de travail Python dans le cloud. La description est relativement claire, la jonction avec l'outil jug décrit dans la première partie du chapitre est assurée. Une dernière section nous indique comment automatiser la construction, la configuration et la gestion de nos machines virtuelles sur EC2 avec StarCluster (http://star.mit.edu/cluster/).

En **conclusion**, je dirais que mon opinion sur les ouvrages généralistes consacrés aux outils n'a pas changé. On ne peut pas parler de tout en analysant en profondeur les méthodes et, dans le même temps, détailler le code permettant de réaliser les analyses. Sauf à produire un livre qui a la taille d'un bottin (certains l'ont fait). Celui-ci n'échappe pas à la règle. Il va très vite sur certains sujets. Quelques chapitres sont parfois un peu confus. On saute d'un thème à l'autre dans l'allégresse générale sans que l'on perçoive réellement la trame sous-jacente. En réalité, chaque chapitre peut quasiment constituer un ouvrage à part entière.

Mais il n'en reste pas moins qu'il est très intéressant (si je n'avais pas aimé, je n'aurai pas fait de fiche de lecture de toute manière, c'est plus simple) car il nous ouvre l'esprit et nous apprend beaucoup sur les « nouvelles » applications et les pratiques actuelles. Libre à nous d'approfondir les thèmes en nous documentant par ailleurs. Moi qui suis très centré sur les techniques de data mining et de statistique, j'y ai trouvé de nombreux exemples et thématiques qui me permettront d'enrichir mes enseignements de Text Mining, que je vais coupler avec le Web Mining. J'avais un peu regardé vite fait ces différents sujets à un moment ou un autre, sans tout saisir. Là pour le coup, j'ai pris le temps de comprendre. Le livre m'y a aidé. De ce point de vue, la lecture de « Building Machine Learning Systems with Python » a été tout bénéfice. **Nous pouvons par la suite approfondir nos connaissances en récupérant les codes sources Python et les données sur le site de l'éditeur**. Ce que je me suis empressé de faire bien évidemment.