Chapitre 3
La pile technologique en Python

1. Les outils de la Data Science

Un bon artisan a besoin de bons outils. C'est également vrai en Data Science et en Machine Learning.

Il existe trois grandes catégories d'outils :

– Des outils « intégrés » qui contiennent ce qui est nécessaire pour un projet : charger les données, les analyser, créer des modèles, les évaluer, les déployer, créer des rapports…

– Des outils « Auto ML » qui simplifient le processus pour les non-experts, en automatisant au maximum les différentes phases.

– Des outils de « développement » avec lesquels il est possible de faire plus de choses, à condition de tout coder, par exemple en Python.

1.1 Les outils intégrés

Il existe de nombreux outils dans la première catégorie, et chacun nécessite un apprentissage particulier permettant de s'en servir au mieux. De plus, ayant chacun leurs points forts et points faibles, il est important de pouvoir choisir le bon logiciel selon le but recherché.

Il est possible de citer : Dataiku DSS, SAS, QlikView, Tableau, Power BI, Snowflake, etc. Certains sont très orientés statistiques (SAS), d'autres visualisation de données (Tableau), mais tous couvrent au moins une partie du processus. Il s'agit principalement d'applications téléchargeables et accessibles via une interface web.

Remarque

Attention, la plupart de ces logiciels sont payants. Il existe bien souvent des versions gratuites, mais qui sont bridées dans leurs fonctionnalités et qui limitent souvent leur utilisation à un usage personnel. Pour être en règle, il est donc important de se rapprocher des principaux éditeurs.

1.2 L'auto ML

Les outils d'auto ML sont des outils généralement accessibles via une interface web. L'utilisateur y rentre ses données brutes et la tâche voulue : classification, régression, etc. L'application se charge ensuite du reste : choix de la préparation des données à effectuer, choix des modèles, choix des paramètres. Le meilleur modèle créé est alors retourné à l'utilisateur.

Ces outils ont l'énorme avantage d'être utilisables sans connaissance en Machine Learning. Cependant, ils manquent de souplesse et ne sont souvent pas très aimés des Data Scientists qui préfèrent avoir le contrôle de leur processus de modélisation.

Ils permettent en revanche d'avoir un premier modèle très rapidement, et un Data Scientist peut ensuite consacrer du temps à d'autres modèles et à l'optimisation des résultats de l'auto ML. La majorité des acteurs du cloud ont leurs propres outils.

1.3 Les outils de développement

Il est tout à fait possible de coder les algorithmes de Machine Learning dans tous les langages de programmation. De nombreux frameworks ou librairies existent, prêts à être incorporés dans des projets.

Les outils de développement classiques sont donc tout à fait utilisables. Tous les éditeurs de code comme Atom, Visual Studio Code ou encore Sublim Text peuvent servir pour un projet.

De plus, le code a l'avantage de plus facilement être partagé et synchronisé entre plusieurs personnes grâce aux gestionnaires de versions comme Git. Par ailleurs, les outils de CI/CD (Intégration continue/Déploiement continu) comme GitLab CI, Jenkins ou Ansible peuvent faciliter le déploiement et la mise à jour. Là encore, chaque cloud provider propose ses outils supplémentaires, comme la série des services Amazon Web Services (AWS) dont le nom commence par « Code » : CodeCommit, CodeBuild... ou encore Azure DevOps.

Ils sont de plus en plus utilisés aujourd'hui.

2. Langage Python

2.1 Présentation

Python est un langage de programmation dont la première version est sortie en 1991. C'est donc un langage mature, connu depuis longtemps des développeurs.

Il possède plusieurs caractéristiques :

– **Interprété** : cela signifie qu'il n'y a pas de compilation avant de pouvoir exécuter le code. Il est donc facile à déboguer mais cela le rend moins efficace en termes de temps d'exécution que des langages compilés comme C/C++. Cependant, la majorité des librairies dites « bas niveau » étant codée en C, le code reste performant.

- **Multiplateforme** : un code en Python ne dépend pas de la plateforme sur laquelle il s'exécute, ce qui le rend très portable et facile à partager. Il est en particulier compatible avec Windows, Unix, Mac OS, Android, iOS.
- **Multiparadigme** : Python permet différentes formes de programmation et s'adapte donc à la majorité des développeurs. Il est ainsi possible d'écrire :
  - En programmation impérative, forme de programmation la plus ancienne
  - En programmation orientée objet, forme aujourd'hui préférée des développeurs tous langages confondus
  - Et en programmation fonctionnelle, permettant une évaluation de fonctions mathématiques et la manipulation de listes de manière simplifiée
- **Lisible** : le langage Python ne contient pas de délimiteurs de blocs contrairement à d'autres langages, comme des accolades en Java ou des mots-clés begin – end en Pascal. La structure se définit par l'indentation des blocs. De cette façon, il est plus court et lisible et, avantage non négligeable, forcément bien indenté.
- **Facile à apprendre** : que vous soyez débutant en programmation ou que vous connaissiez un autre langage, il est facile d'apprendre à coder en Python, même si maîtriser complètement le langage demande plusieurs années d'expérience.

Toutes ces caractéristiques en font un très bon langage pour la Data Science, bien qu'il ne soit pas le seul utilisable.

2.2 Brève présentation de R

R est un langage de programmation destiné aux statistiques et à la Data Science. Il s'agit d'un langage libre et open source. Ce langage a été fortement développé depuis 1997. C'est la force de sa communauté et les nombreux packages disponibles (plus de 15000) qui en font un langage si apprécié.

En effet, quel que soit l'analyse ou le calcul souhaité sur des données, il existe sûrement un package R permettant de le faire.

Sa syntaxe est cependant particulière, avec par exemple pour l'affectation le symbole <-, mais les tableaux et matrices sont très bien gérés et le code optimisé.

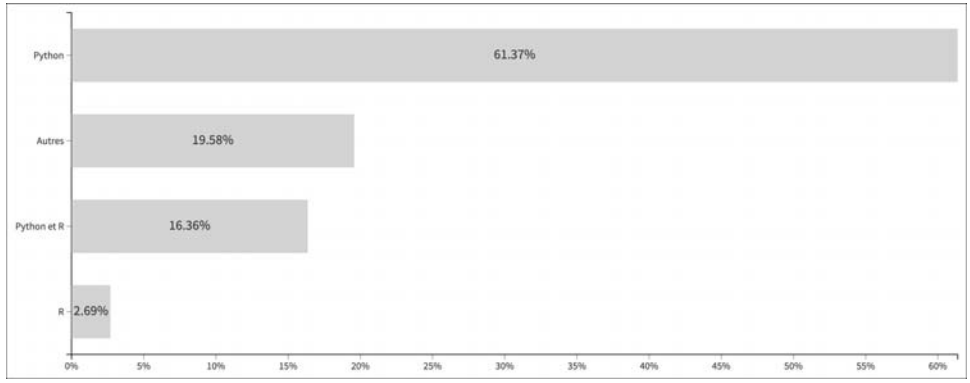
Tout comme Python, il est interprété, multiplateforme et lisible.

2.3 Python ou R ?

Python et R sont devenus les deux langages préférés des Data Scientists. Les deux permettent de coder facilement la majorité des algorithmes de Machine Learning et disposent de librairies pour accéder aux principaux formats ou sources de données.

En tant que langage de programmation, Python est particulièrement apprécié des Data Engineers, et tous les algorithmes existent dans des frameworks Python, souvent codés en C pour des questions d'optimisation.

D'après l'enquête de Kaggle sur les langages utilisés par les Data Scientists (« Data Science and Machine Learning Survey »), Python devance R depuis 2017. Le sondage réalisé fin 2022 donne les résultats suivants, ce qui fait de Python le langage préféré par trois quarts des Data Scientists :



2.4 Python 2 vs Python 3

Il existe deux versions majeures de Python :

– Python 2, supportée jusqu'à fin 2019, et dont la dernière version est la version 2.7, sortie en 2010

– Python 3, actuellement en version 3.12, sortie en octobre 2023

Les différences entre la version 2 et la version 3 sont importantes, rendant souvent la compatibilité entre du code 2.7 et 3.x impossible. De plus, de nombreuses librairies ne sont compatibles qu'avec une seule des versions.

Depuis début 2020, la version 2 n'est plus supportée et ne doit donc plus être utilisée. Il existe cependant quelques cas exceptionnels, comme du code commencé en 2.7 qui doit être maintenu en attendant une migration vers 3.x, ou du code faisant appel à une librairie non disponible en 3.x (ce qui est de moins en moins le cas).

Attention aussi au choix de la version en 3.x. La version 2.7 ayant été très longtemps la version mise en production, les éditeurs de librairies n'ont pas forcément tous créé des versions compatibles avec toutes les versions 3.x. Par exemple TensorFlow, librairie la plus utilisée pour le Deep Learning, n'est compatible avec Python 3.8 que depuis Tensorflow 2.2 et Python 3.9 que depuis sa version 2.5. À l'heure où nous écrivons ces lignes, Tensorflow n'est officiellement pas supporté pour les versions de Python supérieures à 3.9.

De même, il peut exister des contraintes sur les versions disponibles sur un système donné en particulier si le déploiement doit avoir lieu sur du matériel de faible puissance, comme des objets connectés.

Il est donc important de définir les librairies externes et les capacités des systèmes utilisés pour le déploiement en amont du projet, afin de ne pas faire de choix bloquants au moment du déploiement.

3. Jupyter

3.1 Caractéristiques de Jupyter

Jupyter est un logiciel qui propose de créer des « notebooks ». Chaque notebook est en réalité une page de taille non fixée, dans laquelle vont s'enchaîner des cellules.

Chaque cellule peut être de différents types :

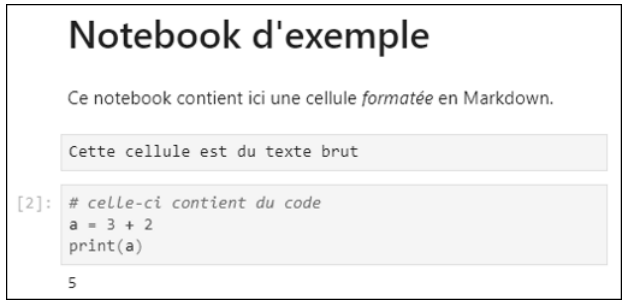
– Code, qui pourra être exécuté et dont le résultat s'affichera immédiatement sous la cellule à l'exécution.
– Texte brut, non formaté et affiché tel quel.
– Texte en Markdown, pour de la mise en page et même des formules en LaTeX.

Remarque

*Le Markdown est un langage de formatage léger créé en 2004 et utilisé dans de nombreux logiciels. Par exemple, les titres de niveau 1 commencent par #, ceux de niveau 2 par ##, etc. Les textes sont mis en italique et en gras en les entourant d'astérisques : *italique*, **gras**.*

Le LaTeX est quant à lui un langage de composition de documents. Fortement utilisé dans les milieux académiques, il permet de dissocier l'écriture du texte de sa mise en page. En effet, celle-ci est calculée lors d'une compilation du texte brut, en respectant les contraintes typographiques et le modèle voulu. Le langage LaTeX est très réputé pour sa capacité à écrire facilement des équations complexes.

Voici un exemple de rendu d'un notebook :



Chaque notebook doit être associé à un kernel. Il s'agit d'un environnement contenant le langage de programmation principal du notebook ainsi que les packages installés. Cela permet d'avoir en parallèle différentes versions d'un même langage.

Il est ainsi possible d'avoir dans le même Jupyter un kernel en Python 3.6 et un en Python 3.9, avec différents modules. En effet, chaque kernel peut avoir ses propres librairies ou des versions différentes de celles-ci, ce qui évite en partie les conflits potentiels.

Actuellement, Jupyter supporte plus de quarante langages dont Python, R, Julia, Scala… De plus, il s'intègre avec Spark, qui permet de gérer des gros volumes de données partitionnés sur plusieurs nœuds.

Le format d'enregistrement des notebooks est un fichier .ipynb qui est en réalité un fichier JSON contenant toutes les informations nécessaires : contenu des cellules et des retours s'ils sont enregistrés. Ce notebook peut ensuite être exporté dans les principaux formats d'échange, dont PDF, HTML, EPS…

Le code JSON du notebook montré précédemment est le suivant :

```
{
"cells": [
 {
  "cell_type": "markdown",
  "metadata": {},
  "source": [
   "# Notebook d'exemple\n",
```