Pourquoi Choisir Python pour la Modélisation Analytique ?

par Visual Numerics (Livres blancs)

Date de publication : Février 2009

Dernière mise à jour :

Cet article explore en détail la modélisation analytique et le déploiement en environnement de production d'applications d'analyse et explique dans quelles mesures ces étapes sont fondamentalement différentes en termes de pré-requis, d'objectifs et de choix d'outils. Les pré-requis concernant l'élaboration du prototype sont calqués sur les fonctionnalités du langage dynamique Python.

Il sera discuté comment réaliser le passage entre la modélisation du prototype et le développement du code de production et il sera montré comment PyIMSL Studio en adresse plusieurs points-clé. PyIMSL Studio est le premier et unique environnement de développement commercial pour le développement d'applications d'analyse numérique conçu pour le déploiement de prototypes mathématiques et statistiques en application de productions.

Cet article concerne principalement les personnes qui sont intéressées par Python comme langage de prototypage mais qui ne sont pas familières quant à son utilisation et à la manière dont il peut parfaitement répondre aux exigences de prototypage/mise en production d'application numériques.

I - Prototyper / Mettre en Production....3
I.1 - Prototyper le Modèle Analytique....3
I.2 - Déploiement en Production de la partie Analytique....3
II - Les Concepteurs de Prototypes et Leurs Besoins....4
II.1 - Pourquoi Python ?....5
III - Les Défis du Déploiement en Environnement de Production....5
IV - La Solution : PyIMSL Studio....5
V - Conclusion....6

I - Prototyper / Mettre en Production

De nombreuses sociétés développant des applications analytiques suivent un processus consistant d'abord à la création d'un prototype de modélisation avant de développer l'application de production. Pour mieux appréhender ces deux aspects, il convient de bien comprendre les objectifs de chacune de ces activités.

I.1 - Prototyper le Modèle Analytique

La modélisation est une activité dans laquelle les données et les algorithmes sont sélectionnés afin d'obtenir les résultats numériques désirés. Cela peut prendre de nombreuses formes différentes en fonction du secteur d'activité, mais voici quelques objectifs communs à tous :

- Identifier les besoins en matière d'analyse pour l'application de production. Les investigations commencent souvent par quelques idées simples à propos des techniques disponibles, mais les besoins concrets pour le déploiement d'une application de production resteront flous jusqu'à ce que des données effectives soient collectées et examinées, et qu'une étude exhaustive des techniques d'analyse adéquates soit réalisée.
- Apporter la preuve qu'une approche analytique spécifique adresse les objectifs identifiés du projet. Par exemple, le calcul de prévisions de ventes n'est pas un objectif en soi, pas plus que l'optimisation d'une chaîne logistique. Le but est plutôt de démontrer que les prévisions calculées et l'efficacité de l'optimisation apportent une réelle valeur commerciale mesurable. Bien sûr, des objectifs similaires s'appliquent pour d'autres industries et d'autres ensembles de problématiques.
- Identifier chacun des problèmes de performance et de scalabilité qui sont des aspects particulièrement cruciaux dans le cadre d'un déploiement d'applications de production.
- Obtenir l'accord des responsables pour passer à la phase de déploiement et de mise en production - traditionnellement au moyen de démonstrations et documentations des prototypes.

I.2 - Déploiement en Production de la partie Analytique

Le but du déploiement en production est de "migrer" le modèle analytique vers un environnement où les parties calculatoires peuvent être efficacement utilisées pour produire des résultats exploitables. Ceci prend également différentes formes en termes de paramétrage, parmi lesquelles :

- Intégrer des algorithmes, des calculs statistiques ou une logique à une application existante utilisée par un groupe, un département ou l'intégralité d'une société. L'application intégrée peut proposer une interface-utilisateur, éventuellement basée sur le web, et est bien souvent conçue pour être utilisée par des personnes dont le domaine d'expertise ne relève pas de l'analytique, mais plutôt de la répétition de différentes tâches ou de la prise de décision sur des données différentes.
- Utiliser le code pour opérer sur des problèmes de calculs intensifs, mettant bien souvent en jeu de conséquents volumes de données, ou lancer des simulations intensives utilisant les modèles développés, éventuellement dans un environnement de calcul haute performance (HPC).
- Traiter les données en différé (batch) pour effectuer des analyses fréquentes ou pour effectuer l'analyse sur plusieurs jeux de données différents. Par exemple, les prévisions de ventes concernant de nombreux produits basées sur un modèle commun de technique prédictive, ou la classification de nouvelles données au fur et à mesures qu'elles apparaissent.
- Intégrer la partie analytique dans un produit commercial destiné à la revente.

Notons qu'il existe d'importants défis à relever lorsqu'on souhaite porter un code d'analyse en production, ce qui distingue les activités et problématiques du déploiement en production de celles liées à la phase de prototypage. Il est généralement risqué de déployer directement et simplement le code du prototype en environnement de production. Parmi ces défis, on peut citer :

- l'amélioration des performances, en écrivant généralement la partie calculatoire intensive dans un langage de plus bas niveau, comme le C.
- la collecte planifiées ou "à la demande" des données, leur mise en forme (filtrage, nettoyage□). Ces opérations sont généralement faites "à la main" lors de la création du prototype. Cette collecte et ce

prétraitement de données peuvent se retrouver à différents niveaux de l'application de production : collecte depuis une base de données, durant la phase d'ETL (Extraction Translation and Loading) ou dans le code de calcul lui-même.
- Gestion et remontée sans faille des erreurs. Il est particulièrement important d'intercepter et de faire remonter les anomalies ou erreurs analytiques plutôt que de renvoyer des résultats potentiellement corrompus.
- Test et contrôle de la qualité sur la précision des résultats analytiques obtenus, afin de s'assurer qu'ils sont identiques à ceux déterminés par le prototype.

Disposer des bons outils numériques pour atteindre cet objectif est primordial, et la parité de ces outils numériques utilisés à la fois pour le prototypage et la production est une considération prépondérante.

II - Les Concepteurs de Prototypes et Leurs Besoins

Qui fait le prototypage ?
Les concepteurs de prototypes sont des analystes qui "explorent" les données et les algorithmes pour parvenir à un résultat numérique désiré. Bien souvent, ils ne sont pas formés au développement logiciel mais sont plutôt des experts dans leur domaine : statisticiens, mathématiciens, experts en business intelligence, analystes quantitatifs, scientifiques ou ingénieurs.
Généralement, comment mènent-ils ces investigations ?
Ils utilisent souvent des "outils sur étagère" pour leur flexibilité et leurs capacités en matière de prototypage rapide. . Les problématiques du déploiement en production sont bien différentes de la facilité à manipuler, filtrer et transformer les données; à appliquer ou paramétrer des algorithmes numériques et à créer des rapports ou des graphiques intermédiaires. Lorsque les résultats produits par les prototypes sont jugés satisfaisants, son code est souvent transmis à une équipe de développement logiciel qui doit trouver un moyen de "répliquer" les méthodes numériques dans un environnement de production en utilisant des outils différents, car l'outil de prototypage est très généralement inadapté à cet environnement.
Quels outils sont requis pour le prototypage ?

- L'outil le plus important consiste en un langage facile à utiliser où les détails des algorithmes sont clairement exposés et faciles à comprendre. Des langages de plus bas niveau comme C/C++, Java et C# peuvent "noyer" cette lisibilité du fait de leur constitution syntaxique et ils pêchent par le manque de fonctionnalités en matière d'opérations de haut-niveau pour la manipulation de tableaux. Idéalement, le code du prototype devrait ressembler aux formules algébriques qu'il implémente.
- Des librairies ou des outils analytiques sont requis pour développer le modèle. Tandis que certains composants sont développés par l'analyste, ces composants s'appuient souvent sur une couche de fonctionnalités d'analyse ou sur une librairie numérique sous-jacente.
- La visualisation graphique est largement utilisée afin d'afficher des résultats intermédiaires et/ou finaux lors de la mise au point d'un modèle, même si cet aspect graphique n'est pas nécessairement requis par l'application de production.
- Certains concepteurs de prototypes préfèrent travailler à partir d'une ligne de commande interactive. D'autres vont favoriser un environnement de développement intégré, plus formel, disposant de puissantes fonctionnalités comme la coloration syntaxique, la complétion de commandes, la mise en forme du code source. Les outils de débogage permettant de suivre l'évolution de la valeur d'une variable sont également une ressource de grande valeur.

Quel(lle)s plateformes/environnements sont utilisé(e)s pour le pour le prototypage ?
Les plateformes de prototypage sont traditionnellement des PC de bureaux fonctionnant sous Microsoft Windows, Linux ou MacOS. Elles sont généralement différentes des plateformes de production qui exigent parfois un plus grand spectre d'environnements matériels et/ou de systèmes d'exploitation, bien souvent des serveurs et non des PC de bureau standard.
Quel est le format typique des données utilisées pour le prototypage ?

- L'accès aux données pour le prototypage peut impliquer un accès formel à une base de données mais dans de nombreux cas, il est question d'échantillons de données ou de sous-ensembles de jeux de données sous la forme d'un simple fichier ASCII tabulé ou d'une feuille Excel. Ceci est très rarement le cas pour une application de production.

- Le filtrage et le prétraitement des données dans une application de production peuvent être effectués lors de l'accès à la base de données ou par d'autres modules composant l'architecture de l'application, mais pour le concepteur du prototype, il est nécessaire qu'il ait à sa disposition un large éventail d'outils faciles à utiliser de manière à ce qu'il accomplisse rapidement et efficacement ces tâches.

II.1 - Pourquoi Python ?

Python est un langage dynamique open source de tout premier ordre, parfaitement adapté au prototypage d'applications d'analyse pour un certain nombre de raisons :

- C'est un langage permettant à la fois une approche procédurale et une approche orientée objet. D'autres langages dynamiques sont souvent plus spécialisés, avec des fonctionnalités adressant des besoins spécifiques, mais ils n'offrent pas autant de compromis que Python s'agissant de la programmation généraliste.
- Python n'est pas un langage propriétaire, ce qui permet un très large partage des outils et des codes d'analyse parmi une très vaste communauté d'utilisateurs. Il y a de très nombreux toolkits open-source pour la modélisation analytique avec Python et ceci est le résultat de plus de 10 ans d'une forte adoption et de larges contributions de la part de la communauté scientifique.
- C'est un langage faiblement typé, avec une syntaxe très simple et très intuitive, ce qui le rend facile à lire et à comprendre.
- Le package NumPy - un standard - pour Python le transforme en un langage dédié aux opérations sur les tableaux, adéquat pour un stockage efficace et pour la manipulation de grands tableaux multidimensionnels. NumPy propose une syntaxe simple pour indexer, sous-échantillonner et manipuler les tableaux. De plus, NumPy est efficace en matière de gestion de la mémoire et de performances.
- Alors qu'il existe de nombreux outils et librairies numériques open-source disponibles pour Python, les wrappers PyIMSL (inclus dans l'environnement PyIMSL Studio) offrent l'ensemble le plus cohérent, le plus riche de techniques mathématiques et statistiques avancées pour, à la fois, la conception de prototypes et le déploiement en applications de production, indépendamment de l'environnement matériel et du système d'exploitation finaux.

III - Les Défis du Déploiement en Environnement de Production

Certains bénéfices du prototypage en langage dynamique deviennent un obstacle pour le déploiement en application de production. Par exemple, les performances peuvent parfois constituer une barrière. Ce point peut devenir critique pour le prétraitement répétitif de larges volumes de données, pour de nombreux jeux de données.
L'architecture de déploiement peut rendre l'intégration d'un moteur de calcul en langage dynamique difficile. Les applications existantes dans lesquelles des composants d'analyse vont être ajoutées définissent le langage à utiliser. Le déploiement web et la scalabilité peuvent requérir quant à eux des solutions différentes de celles proposées par un langage dynamique.
Python peut être utilisé pour le déploiement d'applications de production car il existe de nombreux composants permettant de créer des interfaces-utilisateurs de grande qualité, pour le développement d'applications monopostes comme orientées web. Pourtant, pour de nombreuses applications de production, les performances et la flexibilité optimales sont atteintes en déployant du code écrit en langage C. En effet, du code écrit en C peut être optimisé pour différentes combinaisons de déploiement spécifiques de matériels / systèmes d'exploitation / compilateurs.

IV - La Solution : PyIMSL Studio

PyIMSL Studio combine le langage Python ainsi qu'une sélection d'outils Python robustes avec les fonctionnalités d'analyse avancée de la librairie numérique IMSL C, le tout dans un ensemble facile à installer et avec un support technique de tout premier ordre. Les manques en termes d'entrées/sorties et de "nettoyage" des données sont comblés par des fonctionnalités supplémentaires implémentées par Visual Numerics. Les composants de PyIMSL Studio fournissent aussi bien les fonctionnalités requises par les concepteurs de prototypes que les fonctionnalités analytiques des librairies en C nécessaires au déploiement en environnements de production.
Au cur de PyIMSL Studio se trouve la libraire IMSL C, un ensemble cohérent et complet d'algorithmes mathématiques et statistiques que les développeurs peuvent intégrer à leurs applications logicielles.

Le nombre d'algorithmes disponibles les librairies IMSL avoisine le millier, proposant aux développeurs de nombreuses combinaisons afin de créer des applications d'analyse uniques, compétitives et de qualité industrielle.
Au sein de PyIMSL Studio, ces fonctions mathématiques et statistiques sont accessibles au programmeur Python pour le prototypage rapide et au développeur C pour la réalisation d'applications de production. Le plus important est que les algorithmes sous-jacents sont les mêmes pour les deux environnements. Les professionnels interrogés estiment qu'en utilisant les mêmes algorithmes durant la phase de prototypage et durant la phase de mise en production, cela peut réduire de manière significative le temps de développement par la suppression de recherches additionnelles, de réécriture et de tests.
Avec PyIMSL Studio, Visual Numerics a intégré et packagé une collection pratique et complète d'outils Python fiables pour la modélisation analytique. Ces outils sont testés, documentés et supportés par Visual Numerics et ils s'installent très facilement par l'intermédiaire d'une seule procédure standard. Ce jeu de composants comprend :

- **Python**
- **NumPy**-un ensemble de modules pour la manipulation puissante et efficace de tableaux de données. Il s'agit d'un standard de facto pour l'algèbre matricielle et les tableaux en Python.
- **Des composants d'I/O et de transformations de données**-des utilitaires pour filtrer et transformer les données, parmi lesquels :
  - Un utilitaire pour lire des fichiers de données ASCII, disponible en Python et en code de production C.
  - Un utilitaire d'identification et de substitution de valeurs manquantes, disponible en Python et en code de production C.
  - PyODBC-Un module Python d'accès aux bases de données pour Windows et Linux.
  - xlrd-Un module Python pour importer des données depuis une feuille Excel.
- **matplotlib/pylab**-Des composants graphiques Python.
- **IPython**-Une interface de type "ligne de commandes" pour le développement interactif et l'exploration de données en Python.
- **Eclipse/Pydev**-Un Environnement complet de Développement Intégré (EDI) pour Python.

Les Wrappers PyIMSL
Lorsqu'on effectue une modélisation analytique, le plus important pour les concepteurs est de disposer des fonctionnalités mathématiques et statistiques nécessaires à cette analyse. Dans PyIMSL Studio, cette fonctionnalité est fournie par les wrappers PyIMSL, un ensemble de wrappers Python aux algorithmes de la librairie IMSL C.
Les wrappers PyIMSL permettent d'"exposer" l'intégralité des fonctionnalités de la libriairie IMSL C dans un style qui reflète parfaitement la philosophie du langage Python. Les fonctions qui requièrent des tableaux peuvent être utilisées avec tout ce qui se comporte comme un tableau en Python. La gestion des erreurs utilise la gestion standard des exceptions en Python. Utiliser PyIMSL permet de produire un code minimaliste et lisible, "pythonique" est le terme utilisé à cet effet par la communauté des utilisateurs Python. Pour le concepteur de la modélisation, les wrappers PyIMSL sont un moyen d'accéder aux fonctions fiables, robustes et performantes de la librairie IMSL C sans avoir à programmer en C.
La **documentation des wrappers PyIMSL** décrit comment utiliser l'intégralité des fonctions mathématiques et statistiques de la librairie et est téléchargeable depuis le site web de Visual Numerics.
Parce que les wrappers PyIMSL offrent une interface directe à chacune des fonctions de la librairie IMSL C, les personnes qui souhaitent passer d'un prototype Python à un code de déploiement en C n'auront aucun problème pour déterminer la routine et les paramètres de l'algorithme en C de l'application de production.

V - Conclusion

Ce document a souligné qu'il existe des différences significatives entre la conception d'un prototype et la mise en production du modèle résultant. Python est désigné comme étant un excellent langage de prototypage. Mais dans bien des cas pourtant, le déploiement en application de production implique une ré-implémentation de la partie analytique en un autre langage. PyIMSL Studio fournit non seulement l'environnement de prototypage complet mais aussi la partie analytique pour les deux parties : la conception du prototype et sa mise en production.