TECH TRENDS

LES PRODUITS DATA SCIENCE

La Data Science, de l'idée à la production

Xebia THIGA publicis sapient

Depuis la création de Thiga en 2014, nous avons à coeur de partager notre savoir-faire et notre expérience.

Dans ce livre publié avec notre partenaire Xebia, nous nous sommes concentrés sur les produits Data Science. Ils nous semblaient mériter un traitement particulier pour deux raisons. Tout d'abord, parce que la nature exploratoire, presque expérimentale, d'un Produit Data Science **nécessite des adaptations fortes** de la démarche Produit classique. Deuxièmement, la création de ces produits sollicite des équipes plus larges que les produits numériques traités dans nos autres ouvrages. Au trio Product Manager, développeur, designer s'ajoutent les Data Scientists, Data Engineers ou Data Ops qui constituent une **Product Team Data Science**.

Nous conversons toujours le même objectif : vous aider à **créer des produits qui rencontrent le succès !**

Bonne lecture !

Hugo Geissmann, CEO et co-fondateur de Thiga

EXPLORER

S'ORGANISER

FLUIDIFIER

Corto et le Data Lake perdu

Corto est chef de projet SI d'une grande entreprise française, Murano. Après avoir mené à bien plusieurs projets SI avec une habileté certaine, il vient d'être promu responsable d'un chantier hautement stratégique : la transformation Data de Murano.

Corto ne connaît pas grand chose à la Data Science mais, comme il a travaillé sur le développement du dernier MainFrame, il est de loin le plus qualifié pour le poste. À force de travail et de lecture, il définit un plan d'action en accord avec ce que sa direction souhaite :

- 1ère étape : **créer un Data Lake qui centralise TOUTES les données** de l'entreprise. C'est bien connu, pour faire de la Data Science, il faut de la Data. Et plus il y en a, mieux c'est. Nous sommes en 2019, une entreprise comme Murano se doit d'avoir son Data Lake !
- 2ème étape : **créer un Data Lab**, avec une vaste équipe de Data Scientists de pointe. Ces petits génies des mathématiques sont capables de trouver des relations insoupçonnées entre les sources de données et de créer des modèles qui révolutionnent la manière de travailler des équipes métier.
- 3ème étape : **livrer les solutions et les modèles finalisés aux équipes métier**. Corto connaît bien la maison et il sait que les projets IT laissent dubitative une grande partie de ces équipes. Il faudra donc arriver avec une solution exhaustive et, une fois convaincues, elles deviendront à n'en pas douter des utilisatrices passionnées du Data Lake.

Corto met alors son plan à exécution. Il a un budget conséquent pour mener à bien ce projet, et recrute une quinzaine de Data Engineers organisée en **Feature Teams** pour constituer le Data Lake. Atterrissage prévu dans 3 mois !

Mais rapidement, les problèmes apparaissent. Obtenir les droits pour accéder à certaines bases de données devient extrêmement chronophage. La **diversité des technologies** présentes dans le legacy est telle que chaque source de données nécessite une solution spécifique. Et lorsque les flux imaginés sont compatibles avec les technologies des systèmes actuels, ce sont les équipes de production qui en bloquent l'utilisation car elles n'ont pas les qualifications pour en assurer la maintenance. Les délais s'allongent progressivement : 6 mois, 9 mois, 1 an …

Le projet a déjà pris du retard et le budget prévu pour cette étape est largement dépassé. La pression monte chez Murano. Il faut enclencher l'étape 2. Tant pis si le Data Lake n'est pas terminé. Après tout, plusieurs téraoctets de données y ont déjà été déversés, les Data Scientists devraient pouvoir commencer à travailler.

L'équipe Data Lab est créée, formée de Data Scientists triés sur le volet, juniors pour la plupart, mais formés dans les meilleures écoles.

Malheureusement, et contre toute attente, la magie n'opère pas. Il faut dire que, parmi les données déjà récupérées dans le Data Lake, la plupart ne sont d'aucune utilité ou sont tellement parcellaires qu'elles sont inutilisables, alors que d'autres sources cruciales n'ont toujours pas été injectées. Dans d'autres cas, ce sont la **fréquence de mises à jour ou le format d'import qui sont incompatibles** avec les cas d'usage imaginés. Mais surtout, les équipes ne savent pas vraiment ce qu'elles doivent chercher.

Corto essaie de les aider et les met sur une piste : la détection de fraude. Il met l'équipe en contact avec un membre de la DSI qui connaît ce métier "sur le bout des doigts". Les Data Scientists demandent aussi à rencontrer d'autres personnes en charge du traitement de la fraude afin de **mieux comprendre leur métier et leurs besoins**, mais Corto insiste : son contact a toute la connaissance nécessaire. Il sait surtout qu'en interne le projet perd du crédit et qu'il ne faut pas aller voir les équipes métier sans avoir quelque chose qui fonctionne parfaitement, au risque de définitivement se décrédibiliser. Il demande donc à l'équipe de **faire un Proof of Concept (PoC) pour démontrer la valeur du Use Case**.

Comme il s'agit d'un cas complexe, l'équipe Data Lab teste des dizaines d'approches, améliore le modèle, intègre pléthore de nouvelles caractéristiques (features). Les performances du modèle s'améliorent petit à petit, jusqu'à être, 6 mois plus tard, jugées satisfaisantes : il est temps de mettre en production !

Pour rattraper le temps perdu, Corto décide d'accélérer le tempo, et d'envoyer **le prototype des Data Scientists directement sur le Data Lake de production**. En effet, nous sommes dans un monde agile, inutile de s'embêter avec les processus lourds de qualification d'un projet classique. Et là, nouvelle déconvenue. Les librairies utilisées pour créer le modèle ne sont pas compatibles avec les conditions de production, le temps de calcul est beaucoup trop long pour une utilisation intelligente et les performances se dégradent au fur et à mesure que l'on adapte le modèle à ces nouvelles contraintes. Deux nouveaux mois perdus.

Lorsqu'enfin le modèle est en production et présenté à l'équipe en charge de la détection de fraude, les retours sont mitigés. Certes, le modèle prédit mieux les fraudes que leur outil existant, mais il ne distingue pas les différents types de fraude qui appellent pourtant chacun une réponse distincte. **Le modèle est peut-être une réussite scientifique, mais il n'est en réalité pas utilisable**.

Plus d'un an après son lancement et en dépit d'investissements colossaux, le Data Lake de Corto n'aura apporté aucune valeur.

INTRODUCTION

Vous avez probablement déjà croisé Corto dans vos couloirs, mais que déduire de cette expérience malheureuse ? Certes, la Data Science et l'Intelligence Artificielle ont fait des avancées phénoménales, mais elles ont souvent été faites au prix d'investissements colossaux savamment orchestrés par les géants du secteur. Face à cette démonstration de force qui promet de bouleverser tous les domaines, nombre d'entreprises, poussées par un mélange de peur d'être dépassées et d'excitation face à un tel potentiel, se lancent à corps perdu vers ce nouvel eldorado. Mais **chaque nouvelle ruée vers l'or amène son lot d'excès et d'erreurs**.

Mauvaise gestion du Data Lake, PoCs interminables qui repoussent sans cesse une éventuelle mise en production, ou encore, manque de lien avec la réalité opérationnelle : les causes d'échec d'un projet Data Science sont nombreuses. Une première réaction pourrait être de se dire qu'il aurait fallu mieux planifier, mieux anticiper les problèmes de mise en production, mieux prévoir les incompatibilités de technologies.

Nous ne pensons pas que ce soit la solution. La complexité inhérente à la Data Science est telle qu'il y aura toujours un élément inattendu pour venir enrayer cette mécanique que l'on pensait bien huilée.

C'est l'approche elle-même qui doit être repensée dans son intégralité. Le Data Lake n'est pas une fin en soi, ce n'est qu'un moyen. Il en est de même pour les modèles de Machine Learning. La seule obsession des équipes doit être le **diptyque Use Case critique - solution créative** pertinente.

L'approche que nous préconisons implique de relever trois défis :

- Explorer : identifier la problématique la plus pertinente, la cadrer et définir une vision Produit qui permet l'alignement des acteurs.
- S'organiser : rassembler l'équipe adéquate et mettre en place le cadre de travail et les bonnes pratiques qui permettront d'atteindre l'objectif.
- Fluidifier : mettre en place l'usine logicielle qui apportera la flexibilité nécessaire à l'équipe pour explorer, expérimenter et mettre en production.

Ce TechTrends a pour but de vous guider pas à pas dans cette démarche et de vous aider à faire de votre Produit Data Science un succès.

Nous vous souhaitons une excellente lecture !

PRÉAMBULE

Avant d'entrer dans le détail de cette démarche, clarifions quelques termes.

Qu'entend-on par Data Science ?

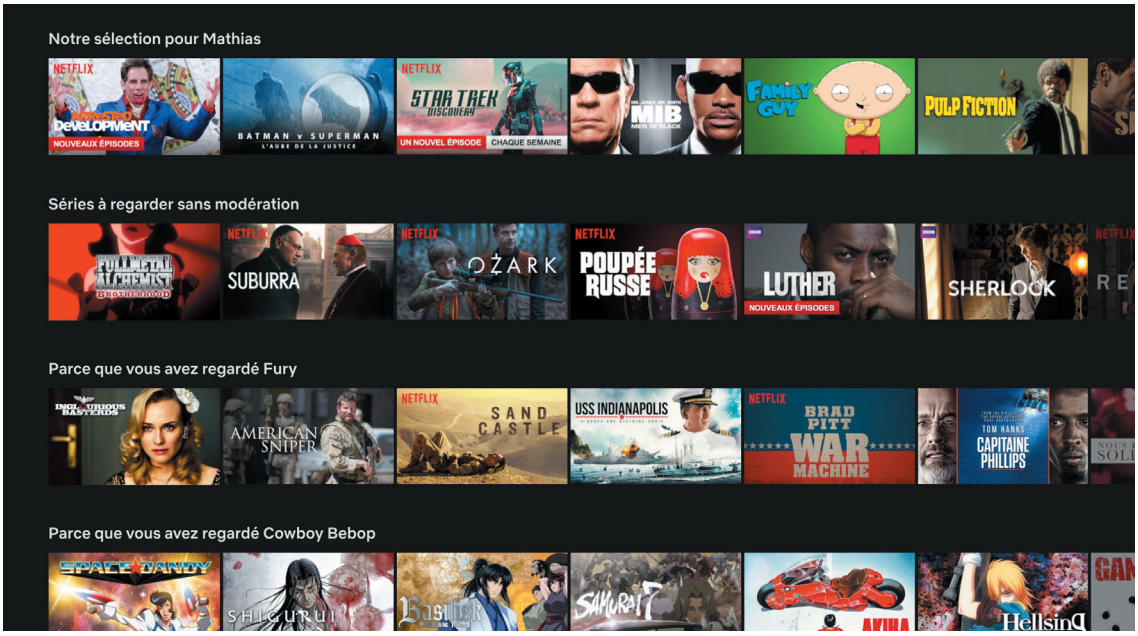
Pris au sens large, la Data Science est la discipline qui consiste à extraire de la connaissance et des informations actionnables à partir de données brutes.

Le Machine Learning, qui est une sous-discipline de la Data Science, est la science qui a pour objectif de créer des algorithmes capables d'identifier des patterns dans des données brutes pour déduire de nouvelles connaissances. Ce type d'algorithme est appelé modèle.

En guise d'illustration, prenons un modèle de reconnaissance d'images qui a pour but d'identifier si une photo représente une voiture ou une moto. Pour atteindre ce résultat, on commence par entraîner un modèle en lui présentant un nombre important de photos labellisées, c'est-à-dire avec l'information "moto" ou l'information "voiture". Au fur et à mesure de cet entraînement, le modèle va identifier des structures dans les photos qui caractérisent les motos et les voitures : c'est ce que l'on appelle des patterns. Une fois ces patterns identifiés, on peut présenter de nouvelles photos non-labellisées au modèle qui sera alors en mesure de reconnaitre s'il s'agit d'une photo de moto ou de voiture. On appelle cela de l'apprentissage supervisé.

Qu'est-ce qu'un Produit Data Science ?

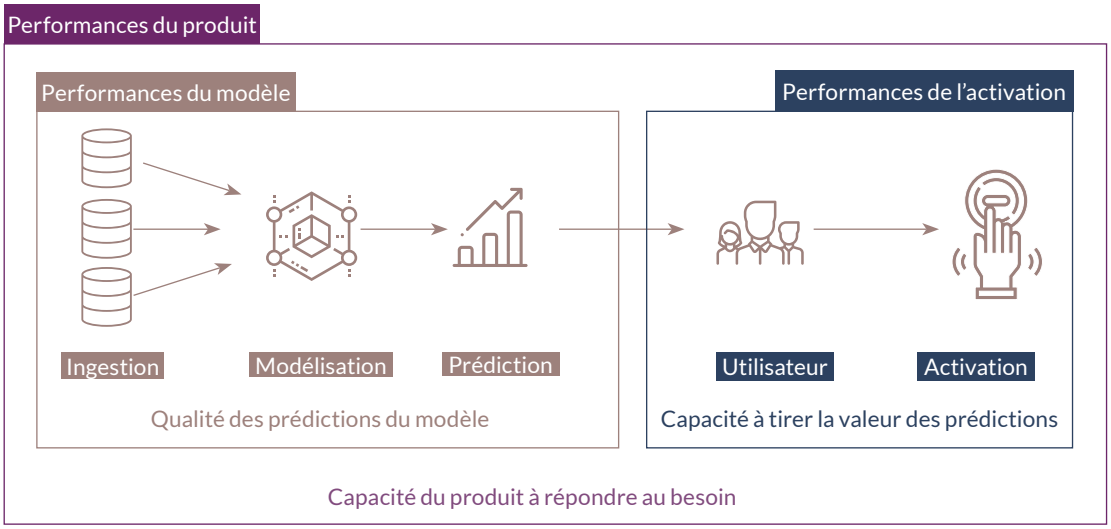
Un Produit Data Science est un produit dont la capacité à apporter une solution à un problème repose principalement sur la Data Science. Il ne se limite donc pas au modèle mais englobe également la mise à disposition des résultats aux utilisateurs pour qu'ils puissent en tirer de la valeur. La performance d'un Produit Data Science mesure donc la résolution du problème et ne se limite pas à la performance du modèle.



La feature "Notre sélection pour vous" de Netflix est un Produit Data Science.

Un Produit Data Science est donc composé de deux éléments :

- le modèle, de l'ingestion de la donnée à l'algorithme, qui apprend en se basant sur un historique de données ;
- la mise à disposition des résultats auprès des utilisateurs.



C'est l'intégration intelligente de ces deux éléments qui permet d'apporter une réponse sans équivalent au problème de l'utilisateur.

La démarche que nous vous proposons dans ce TechTrends permet d'intégrer de manière efficace les aspects modélisation et utilisation dans la création de vos Produits Data Science.

EXPLORER

Les clients ne porteront aucun intérêt à une technologie quelle qu'elle soit, à moins qu'elle n'apporte une solution supérieure à un problème donné.

> *Customers won't care about any particular technology unless it solves a particular problem in a superior way.*
>
> *Peter Thiel, Zero to One*

Cela peut sembler évident, mais cette évidence est trop souvent oubliée lorsque l'on parle de Data Science. Voulant prendre part à cette révolution, de nombreux acteurs se précipitent vers la mise en place d'une solution, consistant principalement à transposer des cas d'usage classiques observés ailleurs à leur situation.

Cette mécanique fait courir le risque d'apporter une solution à un problème qui n'existe pas au détriment d'une opportunité bien plus pertinente. Ainsi, nous avons vu ces dernières années des initiatives autour des chatbots se multiplier, malgré des résultats mitigés : le nouveau parcours utilisateur est souvent plus complexe et moins intuitif qu'une navigation classique sur le site .

Pour éviter cet écueil, il faudra **cadrer rigoureusement le problème que votre Produit Data Science va résoudre**.

Explorer les points de douleur

Objectifs stratégiques

Avant toute chose, il est important de vous questionner sur **l'objectif stratégique dans lequel votre démarche s'insère**. À ce stade, il est recommandé d'être aussi large que possible afin de **considérer une problématique dans son ensemble** et de ne pas vous diriger vers une sous-partie du problème.

Dans l'histoire présentée en introduction, Corto a trop rapidement convergé vers la sous-partie du problème qui lui semblait la plus pertinente, à savoir la détection de fraude. Malheureusement, il n'a pas pris en compte l'objectif plus global recherché qui est d'empêcher que des transactions frauduleuses ne passent entre les mailles du filet.

Le problème peut se décomposer en deux sous-objectifs :

- empêcher les transactions frauduleuses avant qu'elles ne soient passées ;
- détecter les transactions frauduleuses passées et mener les actions nécessaires à leur annulation.

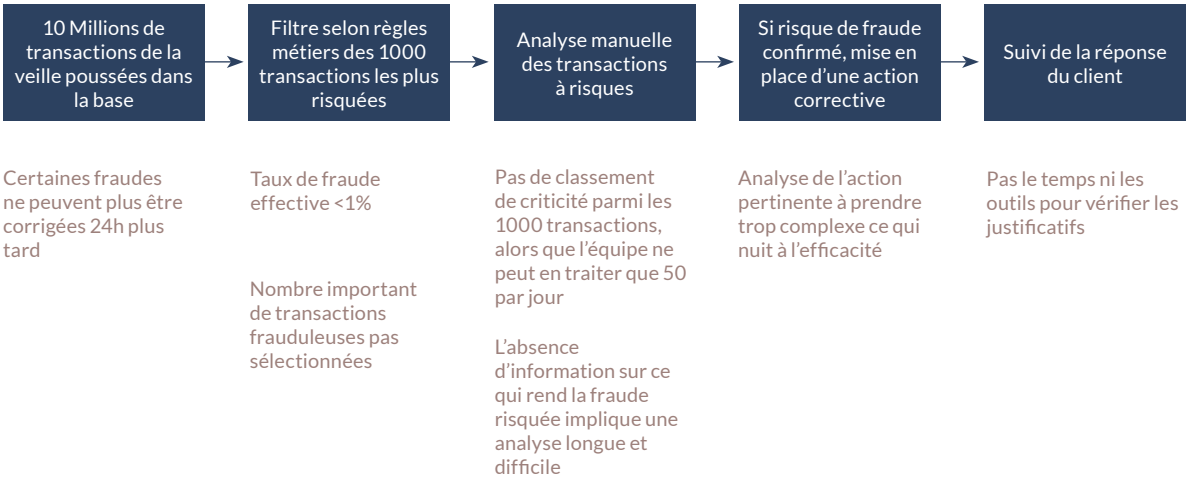
Attention à ne pas tomber dans l'excès inverse. L'objectif "Augmenter le chiffre d'affaires" est par exemple beaucoup trop large pour permettre une réflexion efficace sur les solutions à implémenter.

Définir le bon niveau de problématique est un critère clé de succès pour la suite. Si vous sentez que l'idéation patine, essayez de remonter au cran supérieur (par exemple avec un atelier des "5 Pourquoi"[1]), ou au contraire, de décomposer en sous-problématiques.

1. *https://workshopbank.com/5-whys-root-cause-analysis*

Cartographier le processus / le parcours utilisateur

Une fois le bon niveau d'objectif défini, il est intéressant de **mapper le processus qui permet de l'atteindre** (s'il s'agit d'un outil à utilisation interne) ou **le parcours utilisateur** (si votre Produit Data Science a vocation à être utilisé par vos utilisateurs/clients). À chacune des étapes de ce parcours/processus, listez les points de douleur, c'est-à-dire les étapes chronophages, qui sont sous-performantes ou qui génèrent de la frustration chez vos utilisateurs.



Parcours de traitement de la fraude

Idéation

L'objectif stratégique, le processus et les points de douleur sont la matière brute qui va vous permettre de faire naître des idées de solutions que votre produit va apporter.

Pour l'exploiter, nous recommandons **d'organiser des ateliers d'idéation faisant intervenir des profils hétérogènes** : Managers, Data Scientists, Experts métier, Data Product Manager, etc.

Une fois l'ensemble des participants alignés sur le fruit des étapes précédentes, vous pouvez passer en **phase d'idéation**. De manière collaborative, le groupe va chercher à imaginer des solutions permettant de résoudre tout ou partie des points de douleur cartographiés. À ce stade, priorité à la quantité ! On ne se soucie ni de la pertinence des solutions proposées, ni de leur faisabilité technique.

Lorsque la créativité des participants est épuisée et que plus aucune nouvelle idée n'émerge, combinez les solutions puis sélectionnez-les en fonction de leur pertinence, de leur faisabilité ou de tout autre critère adapté à votre organisation. Nous vous recommandons de **sélectionner trois idées**. Ainsi, vous pourrez explorer plusieurs pistes, tout en restreignant suffisamment le champ des solutions envisagées pour permettre d'approfondir chacune d'elles.

Voici les solutions qui pourraient être envisagées pour Corto :

- **Catégorisation des fraudes :** à chaque transaction, associer une probabilité d'appartenir à chaque catégorie de fraude connue avec une mise en avant des critères explicatifs.
- **Détection de nouvelles fraudes :** identifier les patterns de transactions anormaux et les critères qui rendent chaque transaction anormale pour permettre de détecter des fraudes encore jamais observées.
- **Bloqueur de transaction en direct :** détecter en temps réel les transactions frauduleuses pour permettre de les bloquer avant qu'elles ne soient acceptées.

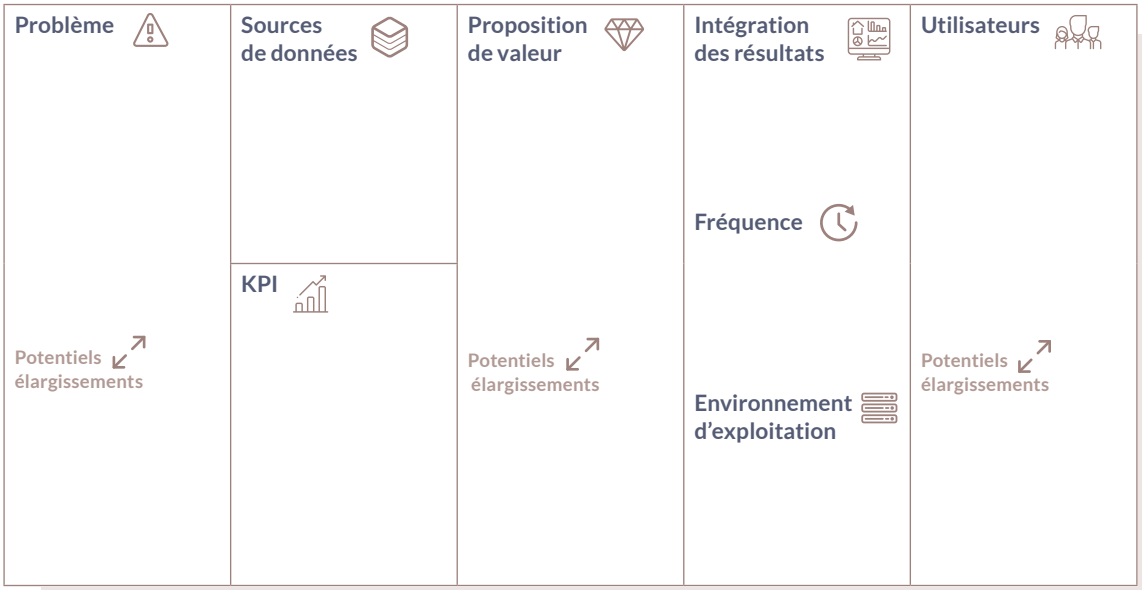
Si le processus que vous avez cartographié est significativement modifié par l'implémentation d'une des solutions imaginées, nous vous recommandons de le mettre à jour. Cela permet d'aligner l'ensemble des participants sur la perception de l'utilisation de la solution et de mettre en lumière de premières limites dans l'utilisation de votre produit.

Etablir une vision produit

Les idées de solution que vous avez fait émerger jusqu'ici sont volontairement floues : l'objectif étant de stimuler la créativité et de considérer des pistes qui sortent des sentiers battus.

Avant de mobiliser une équipe pour réaliser votre produit, vous allez devoir approfondir ces idées et définir une **vision produit**. Cette vision a pour objectif de formaliser les éléments structurants afin de les partager à l'ensemble des parties prenantes, mais également d'identifier les incertitudes qui pèsent sur votre produit.

Pour synthétiser cette vision, nous vous recommandons d'utiliser le **Data Science Product Canvas.**



Le Data Science Product Canvas

Nous vous recommandons de remplir ce Canvas pour chacune des 3 idées de Produit que vous avez identifiées.

Compléter les premiers éléments : Problème, Proposition de valeur et Utilisateurs

À ce stade, vous avez déjà identifié 3 éléments de ce Canvas :

- **Problème :** le ou les points de douleur de votre parcours/process que votre solution doit aborder.
- **Proposition de valeur :** la valeur que votre produit apporte.
- **Utilisateurs :** les personnes qui effectuent les actions dans votre processus.

Définir les KPIs associés à votre produit

Ce sont les indicateurs qui définissent la performance de votre produit sur le plan métier. Ils permettront de mesurer la résolution du problème et l'apport de valeur de votre produit. Si vous ne parvenez pas à définir ces indicateurs en les détachant des indicateurs techniques (taux d'erreur, temps de prédiction, performance du modèle, etc.), ceci est le symptôme d'une proposition de valeur bancale.

Identifier les sources de données

Afin de remplir la case « sources de données », nous vous conseillons un exercice en 3 temps :

1. Oubliez les contraintes de votre entreprise et identifiez toutes les données que vous pourriez utiliser pour réaliser le Produit.

2. Classez ces données en 3 catégories :
    a. **Absolument nécessaires** : sans ces données, il est impossible d'apporter la valeur attendue.
    b. **Importantes** : ces données apporteront l'incrément de précision qui produira la valeur.
    c. **Secondaires** : ces données pourraient avoir un impact positif sur les performances du modèle, mais à la marge ou au prix d'une expérimentation importante.
3. Pour toutes les données des deux premières catégories, identifiez les données que vous avez à disposition ou que vous pourriez acquérir et, le cas échéant, un moyen de gérer l'absence de cette donnée.

Cette catégorisation des données se base sur votre connaissance, encore partielle, du produit ; elle évoluera au fil de l'exploration, des expérimentations et des développements. Néanmoins, cette première vision est cruciale pour vous permettre de définir un point de départ et un angle d'attaque.

Par souci de lisibilité, vous pouvez à ce stade n'inscrire que les données absolument *nécessaires* dans le Canvas.

Se projeter sur l'activation

Ici sont synthétisées les 3 dimensions de « la mise à disposition des résultats » de votre produit :

Intégration des résultats

C'est le vecteur par lequel le résultat de votre modèle sera délivré à votre utilisateur. Est-ce que les résultats ont vocation à s'interfacer avec un outil existant ou est-ce une nouvelle interface ? Est-ce une API ? Un envoi de mail automatique ?

Fréquence

C'est la fréquence à laquelle le modèle doit pousser les résultats. Est-ce un batch une fois par heure, par jour, par semaine, par mois ? Doit-il pouvoir être appelé en temps réel ? À la demande ?

Environnement d'exploitation

Cet élément fait référence aux éventuels changements d'infrastructure qu'il faut envisager entre l'entraînement d'un modèle et son utilisation. Cela peut avoir un impact sur le choix des modèles, leur gestion et les outils à utiliser. Y a-t-il des contraintes supplémentaires sur cet autre environnement ? Quelles adaptations sont à envisager ?

Voir plus loin avec les potentiels élargissements

Vous pouvez enfin vous demander comment vous pourriez élargir plus tard le champ d'application de votre produit, que ce soit par la résolution d'un nouveau problème, la création d'une proposition de valeur enrichie ou encore l'application de votre solution à un autre type d'utilisateurs.

Cette analyse vous permet de vous projeter sur le potentiel à plus long terme de votre produit.

Consolider sa vision produit

Le travail sur tous les éléments du Canvas proposé vous permettra de consolider une vue d'ensemble de votre Produit Data Science et de valider définitivement son intérêt business.

<table>
<tr>
<td rowspan="2">Problème
Absence d'information sur ce qui rend la fraude risquée, implique une analyse longue et difficile

Potentiels élargissements
Analyse de l'action pertinente à prendre trop complexe, ce qui nuit à l'efficacité</td>
<td>Sources de données
Historique des transactions labellisées.
Données clients</td>
<td rowspan="2">Proposition de valeur
Catégoriser les risques de fraude pour suggérer les actions d'analyse

Potentiels élargissements
Catégoriser les couples clients/fraudes pour suggérer une action à prendre</td>
<td rowspan="2">Intégration des résultats
Ajout d'une feature dans l'outil de gestion des fraudes

Fréquence
1 fois par jour

Environnement d'exploitation
Environnement de production de l'outil de gestion de fraude</td>
<td rowspan="2">Utilisateurs
Agent de gestion de fraude

Potentiels élargissements
Equipe de détection de fraudes internes</td>
</tr>
<tr>
<td>KPI
Temps d'analyse d'une transaction suspecte
Nombre de fraudes détectées</td>
</tr>
</table>

Illustration du Canvas pour l'une des trois solutions retenues : catégorisation des fraudes.

Identifier et lever les incertitudes

Afin de définir une vision cohérente malgré une connaissance parcellaire à ce stade de la réalité de votre produit, vous avez émis des hypothèses structurantes dont découlent des incertitudes.

Avant de passer en phase de réalisation et de mobiliser une équipe complète, cherchez à identifier les incertitudes les plus risquées et effectuez un ou plusieurs **prototypes** pour permettre de les lever à moindre coût. Ces **prototypes ne sont pas des éléments ayant vocation à aller directement en production**. Ils permettent de lever les principaux doutes concernant la faisabilité et l'utilité du produit afin d'éviter un investissement inutile.

Incertitude 1 : Fluidité de l'expérience utilisateur de votre produit

L'interface utilisateur est le lien entre l'intelligence apportée par le modèle et son activation par votre utilisateur. Quelle que soit la qualité de votre brique Data Science, si l'interface utilisateur n'est pas adaptée, votre produit créera de nouveaux points de douleur et vous aurez échoué à délivrer la valeur espérée.

Pour éviter cela, nous vous conseillons de veiller à ce que votre produit soit **IPAP** :

- **Intuitif** : les informations qu'il affiche sont facilement compréhensibles.
- **Pratique** : les bonnes informations sont présentées au bon moment et le produit offre un parcours fluide qui s'intègre bien avec les autres outils nécessaires à la réalisation du processus.
- **Actionnable** : le lien entre l'information affichée et l'action qu'il permet de déclencher pour apporter de la valeur est clair.
- **Personnalisable :** chaque utilisateur peut personnaliser les éléments affichés pour satisfaire son besoin.

Le meilleur moyen de s'en assurer est de réaliser un **prototype d'interface** grâce à des outils comme Sketch, Adobe XD ou Figma et de le faire tester par vos utilisateurs : demandez-leur de simuler la réalisation d'un scénario en utilisant votre outil. Récoltez leurs retours et itérez sur votre maquette jusqu'à ce qu'elle soit satisfaisante.

L'interface utilisateur sera parfois un simple dashboard qui a vocation à remonter quelques informations clés, ou bien constituera un simple appel d'API. Dans d'autres cas, votre Produit Data Science présentera un parcours complexe qui nécessite plusieurs actions d'un ou plusieurs utilisateurs et dans ce cas, la qualité de votre interface sera un critère clé de succès. N'hésitez donc pas à vous faire aider par des UX Designers et à mettre en place un Design Sprint[2] pour augmenter vos chances de réussite.

Incertitude 2 : Faisabilité technique de votre produit

Dans un Produit Data Science, il existe deux types d'incertitudes techniques :

- **Incertitude autour de la donnée** : les données nécessaires sont-elles présentes et accessibles ? Sont-elles de qualité ? Est-il possible de croiser les différentes sources ? Leur pertinence statistique est-elle suffisante pour répondre aux besoins ? Une première modélisation simple permet-elle de se projeter sur un niveau de performance adéquate ?
- **Incertitude autour de l'infrastructure** : l'infrastructure actuelle est-elle en capacité de supporter les outils nécessaires à la réalisation du produit ? Les fréquences attendues pour apporter de la valeur sont-elles soutenables au regard du legacy ?

Pour lever ces incertitudes, nous allons réaliser des prototypes techniques.

Attention ! Si un **MVP** (Minimum Viable Product) est une version industrialisée minimaliste d'un produit, un prototype n'a rien d'industriel et n'a pas vocation à l'être. **Le fameux "PoC to Prod" ne devrait même pas exister.**

Ai-je besoin d'un prototype ?

Un prototypage n'est pas une étape obligatoire. Si le Use Case a un intérêt business et que sa réalisation est certaine, n'attendez pas pour démarrer la phase industrielle. Ceci est généralement possible quand le Use Case dérive d'un Use Case antérieur ou que les connaissances sur la donnée et le problème à résoudre sont suffisamment fiables. Ne vous réjouissez pas trop vite, c'est rarement le cas ! Un bon prototypage reste un gage de sécurité pour le développement de votre produit.

2. *http://blog.thiga.fr/innovation-digitale/design-sprint-idee-prototype-5-jours/*

Objectifs et règles du jeu d'un prototypage

Durant la conception d'un Produit Data Science il y aura toujours une part d'inconnu. Cela fait partie du jeu lorsque l'on travaille sur la donnée. Cependant, certaines hypothèses sont plus critiques que d'autres, et peuvent conditionner le choix d'investir ou non sur le développement du produit. **Le but de la phase de prototypage est de lever une à une ces hypothèses critiques**, en un temps restreint, afin d'éviter un accident industriel (comme arrêter un projet 6 mois après son démarrage).

> *If an image worth a 1000 words, a prototype worth a 1000 meetings*
> *IDEO*

Nous vous proposons les règles du jeu suivantes :

Faire fi des contraintes Dans une phase de prototypage, et uniquement durant celle-ci, l'équipe peut s'autoriser à s'abstraire des questions d'architecture et de mise en production. Cette phase se focalise sur la faisabilité et le potentiel du Produit. Il ne s'agit pas de le développer jusqu'au bout sans contrainte.	**Impliquer rapidement le métier** Le métier est souvent le plus à même de connaître les données pertinentes à explorer et de comprendre les explorations qui sont faites. Certaines règles métier sont souvent cachées dans les données et il est important de s'en rendre compte rapidement sans perdre de temps. Grâce à cette interaction, il sera possible de faire ressortir les impacts positifs et négatifs de la donnée, comme le manque d'information.
2 semaines, pas plus ... Plus on passe de temps à explorer sans contrainte, plus la difficulté sera grande de passer à une phase industrielle. Sur une durée de 2 semaines (l'équivalent d'un sprint classique), l'équipe va chercher à surmonter les obstacles les plus critiques qui pourraient compromettre la bonne réalisation du produit. C'est une durée courte, mais en général suffisante pour se forger des convictions fortes sur la faisabilité du Use Case et sur son potentiel business.	**... à renouveler une fois maximum** Un deuxième sprint de prototypage est envisageable pour tester une nouvelle hypothèse critique. Néanmoins seuls les cas les plus complexes et les plus critiques doivent pouvoir en bénéficier. Ne tombez pas dans le travers d'enchaîner des sprints de prototypage. Posez-vous plutôt la question : "L'incrément que j'attends en investissant sur un autre sprint de prototypage me permettra-t-il de prendre une décision finale, ou bien suis-je déjà convaincu de la faisabilité et du potentiel de mon Use Case ?".

Converger vers la solution cible

Vous avez maintenant levé les incertitudes sur vos 3 idées de produit, tant sur le plan de la capacité à apporter de la valeur aux utilisateurs que sur le plan de la faisabilité technique. Pour chacune de vos idées de produit, vous avez une vision clairement définie ainsi que tous les éléments pour estimer la complexité technique de leur réalisation. Vous êtes en capacité de prendre une décision raisonnée et éclairée sur le produit le plus pertinent à construire.

Cependant, n'oubliez pas une chose. L'un des intérêts de cette démarche, largement inspirée des méthodes de **Lean Start-up**, est de vous aider à **identifier les principaux réservoirs de valeur** mais également de **mettre en évidence rapidement et à moindre coût les manquements et complexités dans votre idée de produit**. Évaluez donc attentivement l'intérêt de poursuivre et n'hésitez pas à tuer une idée plutôt que de pousser le développement d'un produit qui ne parviendra pas à apporter la valeur espérée.

> *Un MVP (Minimum Viable Product) est une version industrialisée minimaliste d'un produit, un prototype n'a rien d'industriel et n'a pas vocation à l'être.*

TAKE AWAY PRODUITS DATA SCIENCE

EXPLORER

- Concentrez-vous sur vos objectifs stratégiques avant même de penser à la réalisation en explorant vos points de douleur les plus forts ;
- Menez des ateliers d'idéation avec des profils hétérogènes afin d'identifier les idées les plus porteuses de valeur ;
- Itérez sur votre vision produit, en vous aidant d'un Data Science Product Canvas ;
- Prototypez pour lever les plus grosses incertitudes avant de vous lancer dans la réalisation d'un MVP.

S'ORGANISER

Il est maintenant temps de passer de la vision à la réalisation. Trois points sont essentiels pour cela :

- Constituer une équipe Produit Data Science autonome ;
- Définir une trajectoire permettant de mettre une première version rapidement entre les mains de vos utilisateurs pour l'améliorer itérativement ;
- Mettre en place un cadre agile pour maximiser la valeur apportée.

Constituer son équipe

L'équipe Produit Data Science idéale

Lorsque vous constituez votre équipe Produit Data Science, vous devez avoir une obsession : avoir **l'ensemble des compétences nécessaires pour réaliser le Produit Data Science de bout en bout.**

Concrètement, l'équipe doit être en mesure de :

- gérer l'infrastructure ;
- créer les pipelines de traitement de données ;
- réaliser des modélisations ;
- industrialiser ces modélisations ;
- mettre les résultats à disposition des utilisateurs via le moyen que vous avez défini précédemment.

Cette autonomie a deux vertus. Tout d'abord, elle permet d'établir un sentiment de responsabilité commune au sein de l'équipe vis-à-vis du succès du produit. **Chaque membre est responsable du succès du produit dans sa globalité** et non de la performance d'une sous-partie dont il aurait la charge (par exemple la performance du modèle pour un Data Scientist). La deuxième vertu réside dans la capacité de réalisation de l'équipe. Comme toutes les compétences nécessaires sont réunies, il devient **facile de pousser régulièrement des améliorations en production**, d'itérer et de pivoter si cela est nécessaire.

Deux idées reçues ont la vie dure lorsqu'on parle d'équipes Produit Data Science.

Idée reçue n°1 : il faut recruter des profils qui maîtrisent l'ensemble des compétences

Cette idée reçue date des premières définitions du profil de Data Scientist. Il était décrit comme un profil maîtrisant parfaitement à la fois les enjeux métiers, le développement logiciel, les mathématiques, les statistiques, l'algorithmie et les infrastructures Big Data.

Ce profil n'existe pas. Si vous essayez de constituer une équipe avec ce type de profil vous risquez fort de vous retrouver avec une équipe ayant une connaissance parcellaire de chaque sujet et manquant d'expertise.

Idée reçue n°2 : il vaut mieux séparer Data Scientists et Data Engineers

D'après cette idée reçue, le travail des Data Scientists et des Data Engineers (ou développeurs) est trop différent pour que l'on puisse les rassembler en une seule équipe. Les premiers feraient de la recherche et devraient pouvoir explorer longuement avant de mettre le modèle ultime entre les mains de Data Engineers qui eux se chargeraient d'industrialiser les modèles.

Procéder ainsi est une garantie d'inefficacité. Modèles travaillés en chambre pendant des mois, industrialisation coûteuse, perte de performance, ressentiment entre les équipes, etc. Les points négatifs sont légion !

A contrario, nous avons identifié 8 champs de compétences que votre équipe doit avoir pour permettre la réalisation d'un Produit Data Science.

DevOps Continuous Integration / Continuous Delivery Containerisation Version Management Monitoring Orchestration Serving de modèles Sécurité Networking	**Infrastructure / Architecture** Bases de données NoSQL Architectures Cloud Architectures Event-Driven Architectures Hadoop Data Lake	**Data Engineering** Frameworks de stockage et traitement parallèle / distribué Outils de requêtage de données Concepts et outils pour le temps réel Outils d'Extract Transform Load (ETL)
Développement logiciel Développement Back-End Développement Front-End Développement d'APIs Pratiques de Software Craftsmanship	**Exploration / Statistiques** SQL Modélisation de données Statistiques descriptives Notebooks Analyse de données textuelles Data Visualisation	**Machine Learning** Librairies de Machine Learning Algorithmes de Machine Learning (basiques et avancés) Bonnes pratiques de projets Data Science
Business Expertise métier Compréhension des enjeux stratégiques et business Recherche / Connaissance utilisateurs	**Product Management** Leadership Alignement des stakeholders Communication sur l'avancée des résultats Vision Produit Pont entre équipes techniques et métier Animation d'ateliers	

Les 8 champs de compétences d'une équipe Produit Data Science

L'équipe idéale pour la réalisation d'un Produit Data Science doit alors répondre à deux caractéristiques :

- Les membres de l'équipe couvrent collectivement les 8 champs de compétences ;
- Chaque membre a une expertise sur un ou plusieurs champs de compétences qu'il souhaite partager avec les autres et des connaissances éparses sur les autres champs qu'il veut approfondir.

À toutes ces compétences spécifiques viennent s'ajouter **un socle commun d'organisation agile et d'adhésion à ses valeurs**. Tous les membres doivent aimer travailler en équipe, collaborer, apprendre des autres et transmettre leurs connaissances. Ces qualités personnelles sont trop souvent négligées lors de la constitution d'une équipe au profit d'expertises purement techniques. Pourtant, l'environnement de travail dynamique et stimulant qui doit prévaloir dans ce type d'équipe ne peut se faire sans ce socle commun.

Les profils au coeur d'une équipe Produit Data Science

Il existe une grande variété de profils Data, chacun ayant une coloration due à ses expériences précédentes, à sa formation ou à son intérêt personnel. On voit cependant certains titres apparaître régulièrement dans la littérature : Data Scientist, Data Engineer, Data Ops, Machine Learning Engineer ou encore Data Product Manager.

Gardez à l'esprit qu'en dépit d'un poste sur un CV, chacun peut avoir poussé davantage une compétence au détriment d'une autre. Il n'y a ainsi **pas un profil type pour un Data Scientist, mais bien toute une multitude,** allant du profil statisticien avec un doctorat en Mathématiques appliquées au profil école d'ingénieur ayant travaillé sur des techniques avancées de Deep Learning et de reconnaissance d'images, en passant par celui ayant déjà travaillé sur la mise en production d'un modèle.

Plutôt que de rechercher des intitulés de poste, nous vous recommandons de construire votre équipe pour qu'elle complète tout le spectre des compétences nécessaires.

Bien entendu, au fil de la vie du produit, et en fonction des besoins de chaque phase, la proportion de chaque profil ou expertise dominante sera amenée à évoluer.

Afin de vous aider à y voir plus clair, nous avons détaillé les principaux champs de compétences nécessaires à chaque profil. N'hésitez pas à demander aux membres de votre équipe de se positionner. Plus que des intitulés de poste, cherchez à identifier s'il vous manque des expertises.

Ces expertises sont évaluées sur 3 niveaux en fonction de leur centralité dans le quotidien de la personne :

1. **Débutant** : compréhension des concepts, les sujets relatifs à cette expertise sont périphériques au travail de la personne ;
2. **Avancé** : maîtrise des concepts et des enjeux, les sujets relatifs à cette expertise sont fortement liés au travail de la personne ;
3. **Expert** : au coeur du travail du membre de l'équipe, ses expériences et connaissances des sujets en font LE référent sur cette expertise.

Nous pouvons aussi associer une mission type à chaque profil :

Data Scientist

Sa mission : créer le moteur "intelligent" du produit, en prenant en compte les contraintes autour des données, des enjeux métier, ainsi que les contraintes techniques en vue d'une mise en production.

Data Engineer / Data Architect

Sa mission : garantir les bons choix d'outillage et la robustesse de l'industrialisation du cas d'usage.

Machine Learning Engineer

Sa mission : renforcer le lien en Data Scientists et Data Engineers. Sa compréhension des enjeux des deux mondes lui permet de prendre en considération les contraintes de chacun pour aider à trouver les compromis et faire les choix les plus pertinents.

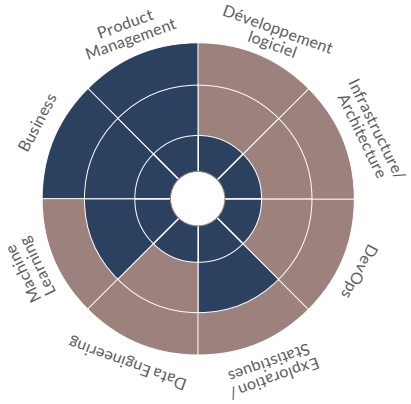
Data Ops

Sa mission : s'assurer de l'alignement de l'ensemble des équipes du SI sur un objectif commun de mise en production des cas d'usage et de leur gestion au quotidien.

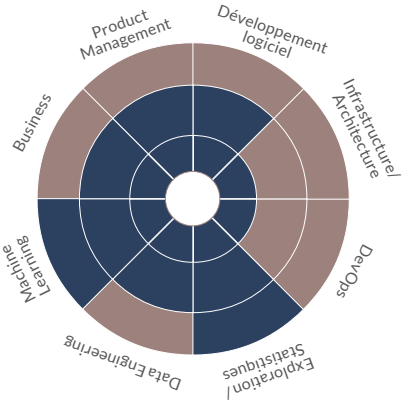
Data Product Manager

Sa mission : maximiser la valeur apportée pour le métier et faire le pont entre vision métier, vision technique et vision produit.

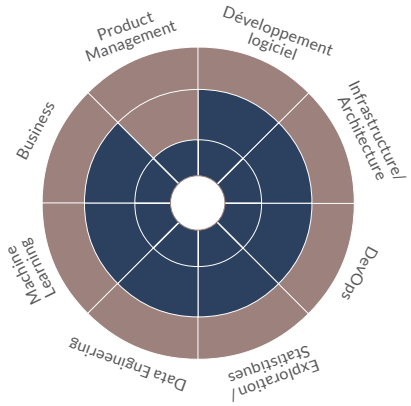
Data product Manager



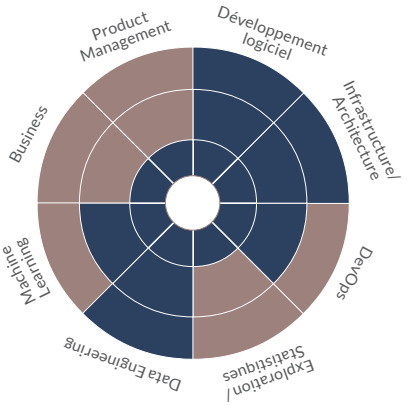
Data Scientist



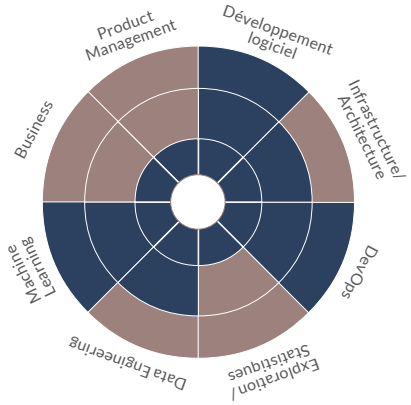
ML Engineer



Data Engineer



Data Ops



Expertises associées à chaque profil classique d'une équipe Produit Data Science

Les profils complémentaires

Ces 5 profils présentés sont au coeur du Produit Data Science : ensemble, ils recouvrent les compétences clés qui permettent de réaliser un modèle dans des conditions de production et d'en pousser le résultat aux utilisateurs.

En fonction de la nature de votre Produit Data Science, il peut être intéressant d'intégrer d'autres profils pour assurer l'autonomie de l'équipe et maximiser la valeur créée pour l'utilisateur.

Si l'apport de valeur nécessite une nouvelle interface utilisateur qui va évoluer tout au long de la vie du produit, il est recommandé d'intégrer un **développeur front-end** dans l'équipe et de bien penser à l'analytics nécessaire pour monitorer les parcours utilisateurs. Si ces derniers sont complexes, vous pouvez également renforcer l'équipe avec des **UX Designers.**

Dans certains cas, votre produit va manipuler des données extrêmement sensibles ou va nécessiter des traitements qui peuvent présenter des failles de sécurité. Si c'est un sujet central tout au long de la vie du produit et qui aura un impact sur le développement de chaque itération, l'équipe peut être renforcée d'un **expert Sécurité**. Il peut aussi être nécessaire, surtout lors des démarrages de projets, de faire appel à un profil régulièrement nommé **Data Architect,** expert en architectures et infrastructures (Big) Data. Une fois les choix structurants faits, l'expertise revient au Data Engineer ou au Data Ops. Afin de garantir le cadre agile dans l'équipe, un Scrum Master est souvent nécessaire, principalement dans les équipes fraîchement constituées.

Enfin, lorsque l'analyse des performances du produit est complexe et doit être croisée avec d'autres éléments externes pour permettre de tirer des conclusions sur les pistes les plus pertinentes à adresser, il est intéressant d'intégrer un **Data Analyst,** qui aura un regard business et métier plus approfondi.

Attention cependant à ne pas dépasser une taille d'équipe critique. Une équipe de **5 à 7 personnes** est idéale. Vous pouvez dans certains cas aller jusqu'à 10 personnes, mais au delà, vous prenez le risque de rajouter de la friction organisationnelle et rendre plus complexe le partage des connaissances.

Constituer une équipe Produit Data Science est un jeu d'équilibriste : il faut réussir à réunir les expertises nécessaires au développement tout en conservant une équipe de taille raisonnable.

Une équipe autonome ne veut pas dire une équipe isolée

Si l'équipe doit être autonome dans la réalisation de son Produit Data Science, elle fait partie d'une organisation plus large et doit s'interfacer de manière régulière et pertinente pour gérer les enjeux communs.

Quel est le coût d'une refonte imposée pour non respect des normes GDPR ou de sécurité ? Quel est l'impact d'une migration non anticipée ? Combien d'heures perdues à cause d'une source de données ignorée ou mal interprétée ? L'isolement de l'équipe peut coûter très cher !

Listez donc dès le départ les équipes satellites qui peuvent avoir un impact sur le développement du Produit. Les équipes classiques sont les équipes métier, les équipes de Data Mining, le service client, les équipes de sécurité, les équipes de production ou encore le département légal.

Une fois ces équipes identifiées, définissez avec elles un mode de collaboration et d'échange satisfaisants pour tous : participation au grooming, à la démonstration, point mensuel, note d'information, etc.

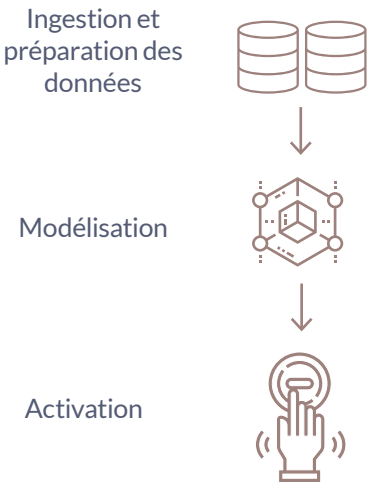
Définir une trajectoire

Le MVP d'un Produit Data Science

La notion de MVP est maintenant largement démocratisée pour les produits web et mobiles. Définir les premiers parcours et fonctionnalités indispensables pour un coût maîtrisé est un exercice pratiqué par la majorité des Products Managers. Mais comment appliquer le même principe à un Produit Data Science ?

Un modèle naïf, mais activé

Dans sa version la plus simple, un Produit Data Science est constitué de données prises en entrée (préparation), d'un algorithme qui les combine et les transforme en données de sortie (modélisation). Cette donnée est ensuite délivrée à l'utilisateur final de telle sorte qu'il puisse en tirer de la valeur (activation).



MVP : Une chaîne de traitement simple, de bout en bout

Dans le cadre d'un MVP, l'objectif sera de simplifier au maximum ces trois briques.

Ingestion et préparation de la donnée

Lorsque vous avez imaginé votre Produit Data Science, vous avez probablement vu grand. Vous imaginiez déjà le croisement de dizaines de sources de données de format et de nature différents. Dans le cadre de la réalisation de votre MVP, vous allez vous **contenter de l'ingestion des sources absolument nécessaires**. Les autres pourront être ajoutées au gré des améliorations ultérieures.

Modélisation

Concernant la modélisation, cherchez à vous abstraire de modèles de Machine Learning complexes. Des algorithmes plus complexes pourront être mis en place par la suite mais, dans un premier temps, contentez-vous d'**algorithmes dits naïfs**. Cela signifie qu'ils appliqueront des règles métier sous la forme de règles mathématiques simples comme des moyennes ou des briques de Machine Learning pré-implémentées.

Non seulement cette approche permet **d'obtenir rapidement un modèle en production**, mais elle a de plus l'avantage d'obliger l'équipe à acquérir la connaissance métier pour réaliser cet algorithme, ce qui sera fort utile pour la suite.

Activation

Il reste maintenant à délivrer le résultat de ce premier modèle à l'utilisateur final. Si nous avons vu que l'interface utilisateur cible de notre produit doit être IPAP, dans le cadre d'un MVP, nous nous concentrerons sur les aspects **Intuitif** et **Actionnable**, et nous laisserons de côté les aspects **Pratique** et **Personnalisable**.

Pour cela, **mettons automatiquement à disposition les résultats à notre utilisateur sous la forme la plus simple**. Cette action lui permettra de comprendre les résultats du modèle et de les actionner. Les subtilités dans les résultats et leur mise en perspective avec d'autres éléments via des visualisations complexes n'est pas à l'ordre du jour, pas plus que l'interfaçage avec les outils du quotidien. Le seul objectif ici est de permettre à l'utilisateur de tester votre produit en conditions réelles pour qu'il puisse vous orienter sur sa pertinence.

Tester en conditions réelles

Par définition, le MVP est imparfait. Des prédictions ou recommandations risquent d'être erronées et l'activation peut ne pas être appropriée à certaines situations.

Dans quelques rares cas, la valeur apportée par le MVP est telle que l'on peut s'en accommoder. Malheureusement, les problèmes traités par un Produit Data Science sont souvent critiques et il est nécessaire de prendre quelques dispositions pour permettre de **tester votre MVP** en conditions réelles et de bénéficier de retours utilisateurs tout en limitant l'impact de ces imperfections.

Garder une activation manuelle

Si en vision cible, le produit sera entièrement indépendant et automatisé, il peut être intéressant de garder dans un premier temps une part manuelle pour l'activation. Dans le cas d'un outil de type chatbot dédié à la relation client, plutôt que de proposer directement un moteur de conversation totalement automatisé, le MVP consisterait à aider les membres du service client (qui continueraient à piloter les conversations eux-mêmes) en leur suggérant des réponses appropriées. On apporte ainsi de la valeur (gain de temps) tout en récoltant un retour sur la qualité de la réponse (sélectionnée ou non).

Sur un sujet aussi critique que la prédiction de défaillance de pièces d'un avion, Air France a mis en place un système de tests ingénieux. L'outil devait prédire la défaillance potentielle d'une pièce présente en plusieurs exemplaires sur un avion : une principale, en position critique, dont la panne empêche l'aéronef de partir, et trois autres dont la défaillance permet de décoller sous conditions. Ainsi, lorsque le modèle prédisait une défaillance de l'équipement sur la position critique, celle-ci était permutée avec une autre située sur une position moins critique. De cette manière, le risque de faux positif par mauvaise prédiction était écarté (pas de pièce changée inutilement), tout en permettant d'observer le comportement de la pièce pour confirmer la défaillance (récolter des retours utilisateurs). Cela a permis de commencer à apporter de la valeur : si la pièce déplacée en position moins critique rencontrait effectivement une défaillance, l'annulation serait évitée et l'avion pourra partir sous tolérance.

A/B Testing

Cette solution est utile lorsqu'il s'agit de remplacer un outil existant. Une partie des choix est réalisée par l'ancien outil et une partie par le nouveau. Dans le cas où il y a un nombre d'occurrences suffisant, on peut orienter un faible pourcentage de choix vers l'outil test afin de garder une significativité statistique tout en limitant l'impact sur l'utilisateur.

Restreindre le périmètre

Lorsque le produit doit gérer une grande diversité de situations, il est intéressant de **se focaliser sur une sous-partie du problème**. Prenons l'exemple d'un Produit Data Science ayant pour objectif de prédire les ventes d'une chaîne de magasins de fruits et légumes. Pour éviter de courir le risque de confier la totalité de l'approvisionnement à votre MVP, vous pouvez commencer par tester sur quelques magasins ou certains fruits spécifiques.

Trois stratégies s'offrent alors à vous :

Stratégie	Exemple	Avantages	Inconvénients
Périmètre où le métier est déjà performant avant l'utilisation de votre produit	Le métier parvient à prédire les ventes de carottes. Le périmètre d'utilisation de votre MVP sera la prédiction de vente de carottes.	Permet de tester votre produit de bout en bout à moindre risque : votre modèle naïf se base sur des règles métier, il devrait donc être performant sur ce périmètre.	L'apport de valeur à l'utilisateur est faible. Les retours ne vous permettront pas de cibler intelligemment les améliorations futures.
Périmètre où le modèle est performant	Les performances théoriques semblent bonnes sur les magasins de Nice. Le périmètre du MVP sera la ville de Nice.	Apporte de la valeur à l'utilisateur, permet de confronter de bonnes performances théoriques et des performances réelles.	Les retours utilisateurs risquent de vous amener à spécialiser votre produit sur un périmètre donné au détriment de l'élargissement futur.
Périmètre aléatoire, mais représentatif	Certains produits ou magasins sont choisis aléatoirement comme périmètre.	Maximise la pertinence des retours utilisateurs et permet d'éviter la spécialisation.	Ceci représente la solution la plus risquée en terme d'impacts utilisateurs négatifs.

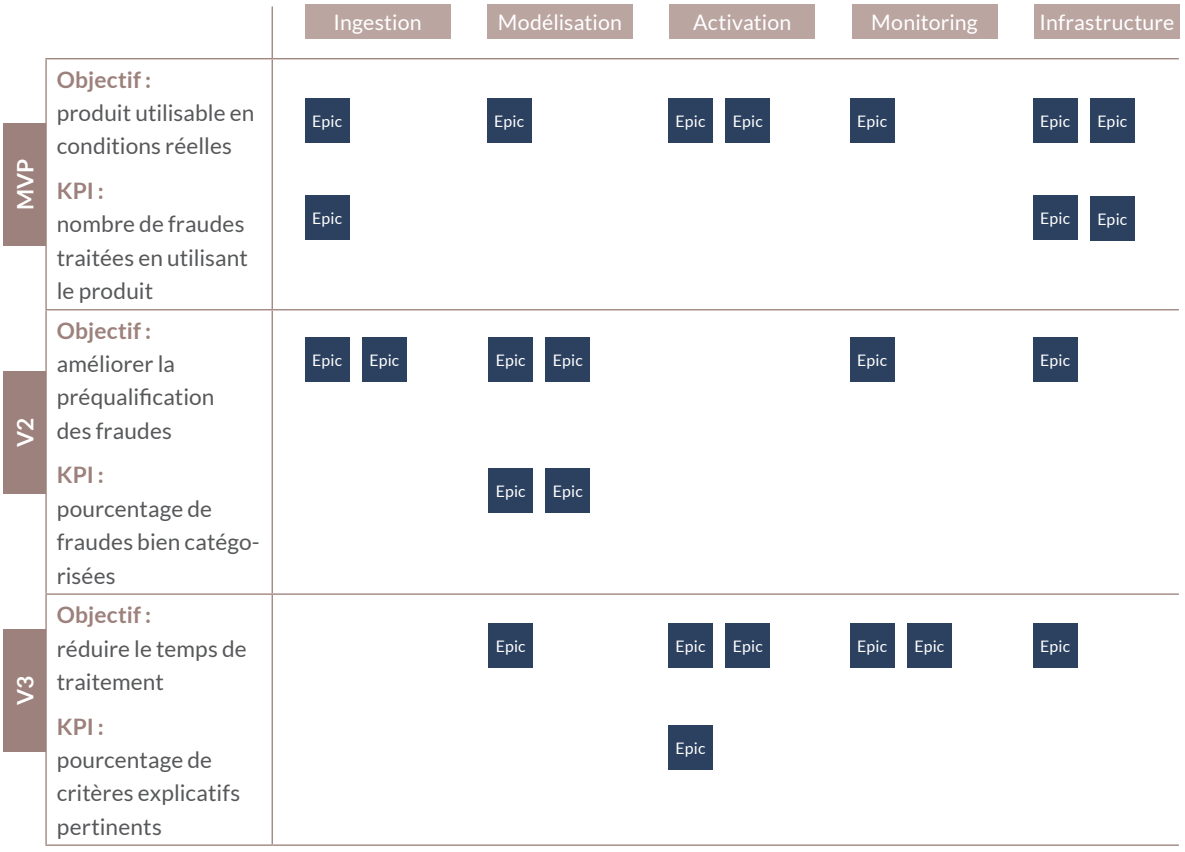
Quelle que soit la stratégie adoptée, l'enjeu sera par la suite de gérer l'élargissement progressif de votre produit au fur et à mesure que ses performances s'amélioreront.

La Story Map pour une vision long terme

Votre MVP remplit sa mission : il apporte de la valeur à l'utilisateur et vous permet de récolter du feedback. Le chemin à parcourir pour atteindre votre vision cible reste néanmoins conséquent. Il est fortement recommandé de tracer ce chemin dès le début de votre projet grâce à une **Story Map, qui est une visualisation en deux dimensions de votre backlog**. Sur l'axe vertical, on trouvera les versions successives. Sur l'axe horizontal, on placera les grandes catégories de développement. Pour ce dernier, commencez par séparer les 5 grandes catégories de votre Produit Data Science : Ingestion, Modélisation, Activation, Monitoring et Infrastructure. Vous pourrez, en fonction des spécificités de votre produit, séparer ces grandes catégories en sous-catégories. Par exemple, si votre brique de modélisation est constituée de plusieurs modèles distincts, il peut être intéressant d'allouer une colonne à chacun. De même, en fonction de la méthode d'activation, vous pouvez distinguer l'interface utilisateur et l'activation automatique.

Pour chaque version de votre produit, associez un objectif orienté utilisateur à un KPI permettant de le mesurer. Par exemple, vous pouvez avoir comme objectif “Réduire le temps de traitement d’une transaction frauduleuse” et associer le KPI “Pourcentage de critère explicatifs pertinents”.

Complétez ensuite votre Story Map en indiquant les grands incréments produit (Epic) à réaliser pour chaque catégorie afin d’atteindre l’objectif de la version. La Story Map doit être **un outil qui évolue tout au long de la vie du produit au gré des découvertes**, des retours utilisateurs ou des nouvelles opportunités qui apparaissent et modifient le chemin initial. Maintenue à jour, c’est un formidable outil, non seulement pour l’équipe, en s’assurant que les développements sont bien en accord avec les objectifs et la vision, mais aussi pour les parties prenantes à qui elle apporte visibilité et clarté sur l’avancement.



Exemple d’implémentation de Story Map

Data Science Agile

Principes clés

De par sa nature exploratoire, un Produit Data Science est particulièrement bien adapté à une démarche agile. Rappelons quelques principes de l'agilité et voyons comment ils s'adaptent aux Produits Data Science.

Accueillez positivement les changements de besoins, même tard dans la vie du Produit

Les processus agiles encouragent le changement pour donner un avantage compétitif au produit. La Data Science est avant tout une suite de rebonds d'une découverte à l'autre, permettant l'obtention d'un logiciel adapté aux réelles problématiques des utilisateurs.

Livrez fréquemment un logiciel opérationnel

Il est nécessaire de s'imposer une livraison régulière pour mettre la solution dans les mains des utilisateurs et obtenir des retours. Les cycles de livraison sont idéalement de quelques semaines, au pire de quelques mois.

L'imperfection d'un modèle peut faire peur, pourtant il est primordial de toujours livrer un incrément utilisable du système afin de le confronter à la réalité et d'obtenir un feedback constructif. Des livraisons d'indicateurs au travers d'un support, type PowerPoint, ne peuvent pas être considérées comme un logiciel opérationnel.

Les utilisateurs et les développeurs doivent travailler ensemble quotidiennement tout au long de la réalisation du produit

Intégrer les utilisateurs au fil des réalisations est primordial pour faire un produit utile. De plus, cette collaboration permettra de démystifier ce domaine technique et favorisera l'appropriation par les utilisateurs. Si une présentation lors des revues de sprint est le minimum, nous encourageons des rencontres plus régulières pour échanger sur les découvertes et arbitrer en fonction des connaissances métier et de la donnée.

Une attention continue à l'excellence technique et à une bonne conception renforce l'agilité

Les Data Engineers et Data Scientists doivent travailler ensemble pour se tirer mutuellement vers le haut. Faire des revues de code quasi quotidiennes est une bonne pratique. Ces deux profils aux parcours souvent bien différents ont tendance à être séparés. Ils ont pourtant un objectif commun. La collaboration entre ces équipiers permettra d'avoir un produit performant, robuste et facilement déployable.

La simplicité est essentielle

Le mieux étant l'ennemi du bien, il faut savoir s'arrêter même si l'on n'est pas pleinement satisfait.

L'amélioration continue

À intervalles réguliers, l'équipe réfléchit aux moyens de devenir plus efficace, puis règle et modifie son comportement en conséquence. Les rétrospectives sont un moment privilégié pour cela.

Alimenter le backlog

La Story Map offre la **vision macro à un instant T** des chantiers à réaliser et de leur ordre de priorité. Comme pour tout produit, il est nécessaire d'affiner au cours de l'avancement cette Story Map pour en obtenir un backlog de stories réalisables par l'équipe.

> *Un projet en échec : un projet au cours duquel personne n'a eu de meilleure idée que ce qui était initialement prévu.* *Mike Cohn*

Il n'est pas question de s'enfermer dans la vision portée par cette Story Map. Tout le sens de la démarche proposée ici est de rester en alerte vis-à-vis des signaux qui pourraient l'amener à évoluer, que ce soit par un changement de priorité ou l'émergence d'un nouveau sujet qui n'avait pas été anticipé. Il existe au moins trois canaux à maintenir actifs tout au long de la vie d'un Produit Data Science afin d'alimenter cette réflexion : **les retours utilisateurs, l'analyse de la donnée et le monitoring.**

Interactions avec le métier et les utilisateurs

La fréquence et la qualité des interactions avec les métiers est un facteur clé de succès. Pendant la démonstration mais aussi tout au long du sprint, l'équipe doit pouvoir poser des questions et demander des explications sur une donnée ou un comportement étonnant qu'elle observe. Prenons à titre d'exemple une équipe travaillant sur une prédiction de revenus et qui s'arrache les cheveux pendant toute la durée d'un sprint : leur modélisation marche très bien, sauf pour une partie du mois de janvier. L'équipe a tenté de résoudre l'énigme durant tout un sprint, sans succès. Lorsqu'elle a expliqué ceci lors de la démonstration, le métier a immédiatement réagi et expliqué que cela était dû à un événement spécifique qui se déroulait à cette date.

Le métier et les utilisateurs ne doivent pas être en simple réaction aux demandes de l'équipe. Ils doivent **faire part de leurs intuitions et des pistes** qui leur semblent intéressantes à explorer, communiquer les informations pertinentes et surtout donner leurs retours sur l'utilisation de la solution.

Il est grand temps de briser ce mythe qui consiste à penser qu'il suffit de fournir des données à des Data Scientists pour qu'ils trouvent comme par magie la solution. **La connaissance métier mise à la disposition de l'équipe est primordiale**. Elle permet une **meilleure priorisation et un gain de temps considérable**.

Exploration et documentation des données

Tout au long de la vie du produit, l'équipe va acquérir une meilleure connaissance des sources de données fournies. Des analyses plus poussées provenant des différents acteurs travaillant sur la donnée peuvent permettre d'identifier des zones de sous-performance du modèle ou des potentiels encore inexploités.

De tous ces éléments naissent des hypothèses et des intuitions qu'il va falloir vérifier. Nous recommandons vivement de **mettre en place un outil répertoriant les hypothèses faites** et, lorsque c'est le cas, le test qui a été mis en place pour les démontrer ainsi que les conclusions de celui-ci. L'objectif est double. Premièrement, il s'agit **d'éviter les doublons dans la vérification d'hypothèse**. Deuxièmement, en ajoutant la nature du test réalisé on peut juger de sa complétude et le cas échéant **reprendre un pan de l'analyse qui aurait été négligé**. Bien qu'il soit possible de créer cette mécanique avec ses propres outils, des projets tels que **Knowledge Repo**[3], rendu Open Source par Airbnb, peuvent vous aider à la mettre en place simplement.

3. https://github.com/airbnb/knowledge-repo

Monitoring

En plus de permettre de réagir rapidement en cas de problème, le monitoring mesure la performance atomique de chacune de vos briques (d'ingestion, de modélisation, d'activation et d'infrastructure). En décomposant les performances de votre produit sur plusieurs niveaux, il permet **d'identifier de manière fine les axes d'amélioration du produit** afin de prioriser les développements. Un monitoring efficace s'organise sur différents niveaux d'indicateurs, que nous allons détailler.

Indicateur de performance du produit

Pour piloter le développement de votre produit, vous devez rattacher l'objectif de celui-ci, tel que défini dans la phase d'exploration, à un **indicateur principal**. Ainsi, le développement de votre produit sera piloté par cet indicateur. Il sera suivi au quotidien par l'équipe pour faire les arbitrages. S'il y a une hésitation entre deux solutions, privilégiez celle qui optimise cet indicateur. Les anglo-saxons l'appellent "the One Metric That Matters" , c'est-à-dire l'unique indicateur qui compte.

Indicateurs métiers secondaires

La proposition de valeur de votre produit peut être décomposée en plusieurs sous-objectifs. L'atteinte de ces sous-objectifs, perceptible par les utilisateurs, se mesure à l'aide des indicateurs métiers secondaires. On s'en sert régulièrement pour **mesurer la performance d'une version dans une Story Map**. Dans le cas du Produit Data Science de catégorisation des fraudes, on peut suivre des indicateurs secondaires tels que le pourcentage de fraudes correctement catégorisées, la pertinence des facteurs explicatifs ou encore le temps de traitement d'une fraude.

Indicateurs techniques

La dernière couche de monitoring couvre les indicateurs techniques, c'est-à-dire ceux qui ne sont pas directement perceptibles par les utilisateurs mais qui contribuent indirectement à l'apport de valeur. Il peut s'agir **d'indicateurs de performance du modèle ou de performance de l'infrastructure** (temps d'entraînement, temps de prédiction, temps de chargement des données, etc.).

Les éléments initialement définis dans votre Story Map et les nouvelles opportunités que vous avez identifiées avec ces méthodes vous offrent un vaste éventail de pistes pour améliorer votre Produit Data Science. Il faut maintenant affiner ses pistes pour les transformer en éléments clairs et concrets qui pourront être développés par l'équipe.

Affiner le backlog

Les pistes que vous avez identifiées sont souvent trop larges pour être adressées en un sprint, ce sont des **Epics**. La première étape consiste à **les découper en plus petits incréments** : les **User Stories**. Ce sont ces User Stories que nous allons affiner grâce au **Backlog Grooming** (ou **Backlog Refinement**).

Backlog Grooming

Cette cérémonie doit être l'occasion de **clarifier le périmètre de la User Story** que vous cherchez à réaliser. Le Data Product Manager présente les éléments qui l'ont amené à considérer cette User Story comme intéressante (analyse, retour métier, etc.) à l'ensemble de l'équipe et répond aux questions pour clarifier le périmètre. C'est un moment d'échange qui permet notamment d'arbitrer sur le niveau de robustesse attendu.

Une fois le périmètre clairement défini, un exercice intéressant consiste à diviser l'équipe en deux et à demander à chaque sous-équipe de **découper cette User Story en tâches techniques**. L'une des deux équipes présente ses réflexions et l'autre ajoute les points qui lui semblent manquer.

Toute l'équipe est alors alignée sur le périmètre et sur la manière de réaliser la User Story, elle peut à présent en **estimer la complexité**, à l'aide d'un *Poker Planning*.

Si l'équipe ne parvient pas à découper ou estimer une User Story (parce qu'elle n'a jamais traité de User Story similaire auparavant ou s'attaque à une nouvelle technologie), il est alors logique de passer par une étape de défrichage, souvent appelée **Spike**. Un Spike est intégré au sprint de l'équipe comme le serait une User Story, mais avec deux particularités :

- il n'est pas estimé en points de complexité mais à un nombre d'heures ou de jours qui lui sont dédiés ;
- son objectif n'est pas de réaliser un incrément produit mais de lever les incertitudes.

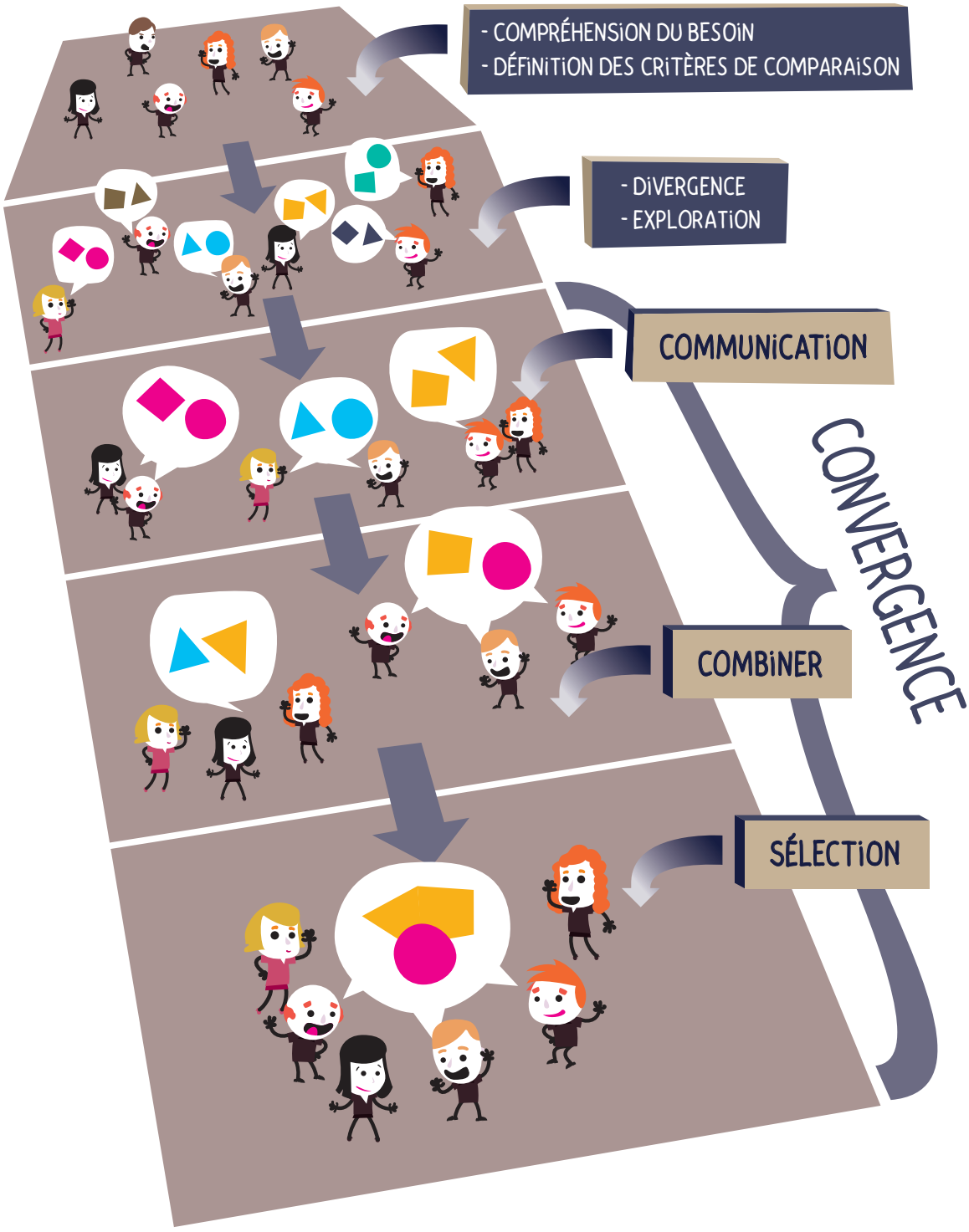
Pour certains choix structurants, cette méthode ne s'applique pas. C'est le cas lorsqu'il s'agit de changer le modèle de l'une des briques clés du produit : plusieurs approches sont possibles, mettre en place chacune d'elles demande un effort conséquent et on ne dispose pas d'éléments nous permettant de choisir la meilleure a priori. Comment identifier alors l'approche la plus adaptée ?

Set Based Concurrent Engineering

Le Set Based Concurrent Engineering (SBCE) est une méthode d'innovation pour système complexe mise en place par Toyota. Le SBCE se base sur l'exploration de pistes multiples pour résoudre un problème et sur leur combinaison pour parvenir à une solution optimale.

Cette méthode peut être appliquée à la Data Science de la manière suivante :

- J1 : le Data Product Manager expose la problématique et définit avec l'équipe les contraintes, le périmètre du test et les critères qui permettront de faire le choix.
- J1 à J7 : en parallèle du développement du sprint, chaque membre de l'équipe réfléchit individuellement sur les solutions qui lui semblent appropriées.
- J7 : chaque membre de l'équipe présente une ou plusieurs approches de son choix. Un vote est fait et l'équipe sélectionne autant de pistes qu'il y a de Data Scientists.
- J8 à J14 : chaque Data Scientist prototype une des approches sur une semaine. Tous les jours, le travail de chacun doit être revu par un autre Data Scientist.
- J14 : chaque Data Scientist présente les résultats et enseignements de son prototype. L'équipe définit alors deux approches hybrides combinant les bonnes idées de chaque prototype.
- J15 à J21 : l'équipe prototype les solutions hybrides.
- J21 : l'équipe sélectionne la solution la plus prometteuse, avec éventuellement une nouvelle hybridation des solutions.



Set Based Concurrent Engineering

De bonnes pratiques pour développer une responsabilité commune

Une équipe pluridisciplinaire a été mise en place pour permettre une grande autonomie et le développement d'un sentiment de responsabilité commune : nous ne sommes pas dans un schéma avec un membre qui soit responsable de la modélisation et un autre de la mise en place de pipeline de transformation de données, mais bien au sein d'une équipe qui a la responsabilité d'atteindre l'objectif final.

Pour éviter que cela ne demeure qu'un voeu pieu et que des silos internes à l'équipe ne se forment, nous vous recommandons de mettre en place quelques bonnes pratiques.

User Stories orientées utilisateur

La tentation est grande lors de la réalisation d'un Produit Data Science de séparer en User Stories distinctes les tâches de modélisation et d'industrialisation. Nous ne pouvons que vous mettre en garde contre ceci. D'une part, cela implique que certaines stories n'apportent aucune valeur à l'utilisateur (quel est l'intérêt d'un modèle s'il n'est pas mis à disposition de l'utilisateur ?). D'autre part, vous courrez le risque d'avoir des Stories spécifiques à chaque profil et voir les membres de l'équipe cesser de travailler ensemble.

Nous vous conseillons donc de garder des User Stories qui apportent qui apportent une valeur tangible au produit et de vous assurer que tous les profils travaillent sur l'ensemble des tâches.

Pair Programming et Pair Review croisés

Toujours dans cette optique de renforcer la collaboration et la montée en compétence des différents profils de l'équipe, encouragez le Pair Programming et les Pair Reviews croisés.

Concrètement cela se traduit par l'instauration de deux règles qui se sont révélées très efficaces :

- Chaque story commence par du Pair Programming entre deux membres aux profils différents ;
- Le code de chaque membre doit être revu par un membre au profil différent de celui qui a rédigé le code.

TAKE AWAY PRODUITS DATA SCIENCE

S'ORGANISER

- Constituez une équipe pluridisciplinaire pour qu'elle puisse être autonome dans la réalisation du produit ;
- Définissez un MVP qui permettra de récolter rapidement des retours utilisateurs et de prioriser vos prochains développements ;
- Matérialisez le chemin pour atteindre la vision cible grâce à une Story Map ;
- Mettez en place les interactions et outils qui permettent un pilotage éclairé des itérations ;
- Adoptez les spikes et SBCE pour gérer l'incertitude.

FLUIDIFIER

La mise en production rapide du MVP et des itérations successives sont des critères clés de succès pour votre Produit Data Science.

Qui dit mise en production, dit aussi mise en place d'une **usine logicielle**. Elle englobe l'ensemble des outils et frameworks permettant de développer et de déployer les incréments d'un produit. Elle est a minima composée d'outils de versionning, de repositories de packages, de scripts de déploiement et d'éléments de monitoring.

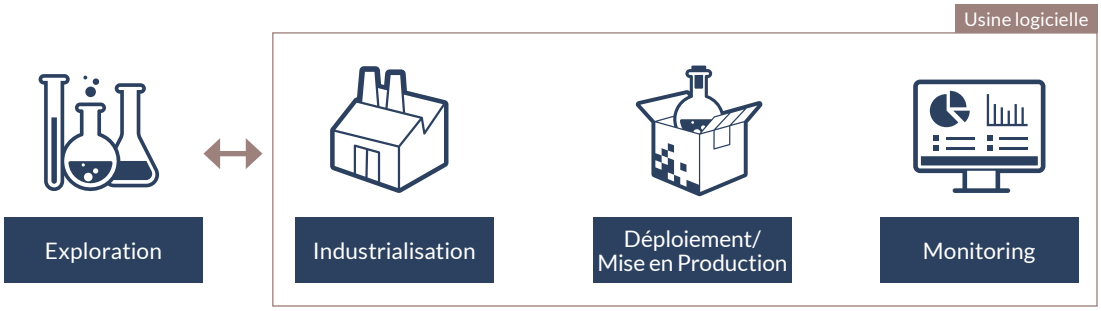
Dans le cadre de la conception d'un Produit Data Science, l'usine logicielle **fait bénéficier d'une mise en production rapide, tout en favorisant l'amélioration et l'innovation**.

Concrètement, la qualité d'une usine logicielle doit être évaluée selon deux axes :

- La **robustesse**, qui mesure la capacité à pousser en production les développements sans compromis sur la qualité ;
- La **souplesse**, qui mesure la facilité d'intégration entre l'exploration et l'ajout itératif de complexité au produit.

Si l'équipe redoute la mise en production, souvent perçue comme une contrainte qui ralentit l'innovation, c'est probablement que l'usine logicielle n'est pas adaptée.

Nous allons parler tout au long de cette partie de «projet», un terme qui porte une certaine ambiguïté. Nous ne faisons pas ici référence à la gestion de projet, puisque nous recommandons justement d'adopter une démarche produit en opposition à une démarche projet. Nous ferons plutôt référence au projet au sens technique du développement logiciel qui est l'ensemble des outils et travaux effectués par les développeurs pour réaliser un logiciel.



Usine logicielle Data Science

Industrialisation et packaging ne sont pas les ennemis des notebooks

Dans le cadre du développement de Produits Data Science, on ne part que très rarement d'une feuille blanche : il existe une base de code, issue de la phase de prototypage ou d'exploration. Toutefois, elle n'est pas exploitable en l'état car constituée de notebooks inadaptés à un environnement industriel.

Il est donc nécessaire de reprendre ce code, de l'industrialiser et de le packager pour son exploitation. Cette étape peut être longue et fastidieuse. Elle est généralement synonyme d'un ralentissement voire d'un arrêt des phases exploratoires visant à améliorer les performances.

Pourtant, les phases d'industrialisation et de packaging peuvent contribuer au déploiement d'un projet stable et robuste, tout en permettant l'accélération des prochaines explorations.

Bien structurer son projet

La première étape vers l'industrialisation d'un Produit Data Science est de se munir d'un environnement de travail adéquat. Un **IDE, tel qu' IntelliJ**[4] **ou Atom**[5], est souvent préconisé par les Data Engineers.

L'environnement de travail configuré, il s'agit alors de définir la structure de son projet. Cette partie est généralement simple dans la mesure où un Produit Data Science peut être découpé en étapes bien définies, qui vont correspondre aux différents modules du projet :

- **Ingestion et nettoyage** des données ;
- **Feature Engineering** : dérivation de caractéristiques explicatives à partir de données brutes ;
- **Modélisation** : définition d'un modèle (simples heuristiques ou bien modèle de Machine Learning) qui sera entraîné et utilisé pour faire des prédictions ;
- **Prédiction** : application du modèle entraîné sur de nouvelles données pour fournir des prédictions.

Contrôler et rendre le code lisible via des tests et du refactoring

Une fois la structure posée, il devient plus aisé de reprendre le code de la phase exploratoire et de le remettre en forme. Néanmoins, passer à un code de qualité industrielle induit plus que de simples copier/coller. Une phase de refactoring sera obligatoire. C'est à ce moment qu'une **approche Test Driven Development**[6] peut avoir du sens. En plus de favoriser l'observabilité du code produit, elle permet de sécuriser la migration de celui-ci.

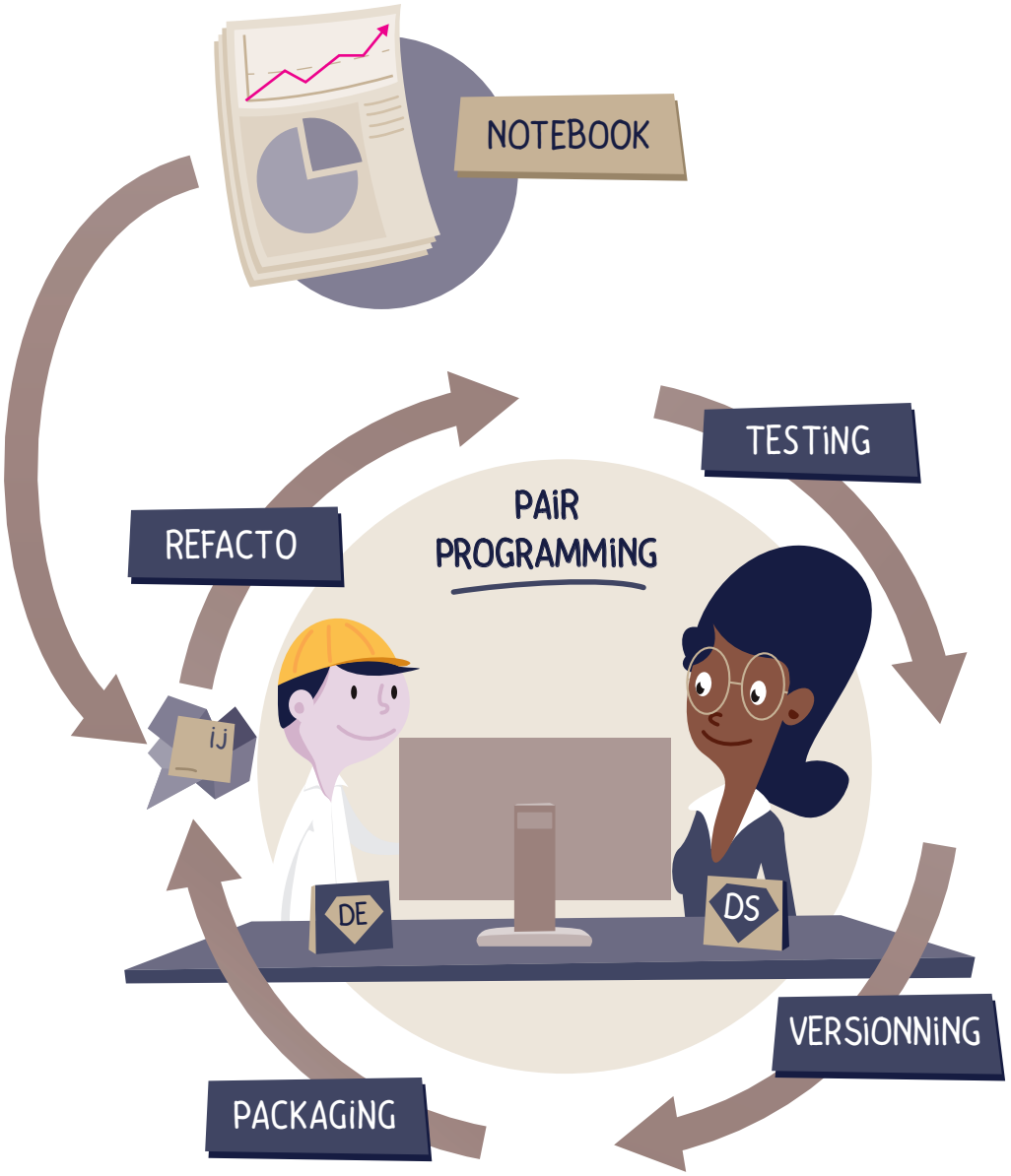
---

4. https://www.jetbrains.com/idea/

5. https://ide.atom.io

6. https://fr.wikipedia.org/wiki/Test_driven_development

Ne nous le cachons pas, le code généré sur un notebook par un Data Scientist est souvent difficile à appréhender sans la présence de ce dernier. L'objectif de la reprise du code pour l'industrialisation est de le rendre compréhensible, afin **d'obtenir rapidement une vision d'ensemble sur ce qui est fait et d'accéder facilement aux détails si besoin**. Le pair programming entre Data Scientists et Data Engineers est un élément déterminant dans la bonne réussite de cette industrialisation.



Industrialisation d'un projet Data Science

Dans cette optique le code n'appartient plus à son auteur, il est partagé et sa qualité relève de la responsabilité de tous. C'est **le principe du collective ownership**. Des formations et exercices sur les principes du clean code permettront de propager cette culture de la qualité.

En plus du refactoring du code, **l'incorporation de tests unitaires et d'intégration est primordiale** pour avoir un produit digne de confiance. Nous attirons votre attention sur le fait qu'une couverture de code à 100% est irréaliste.

La phase de **Feature Engineering est facilement testable unitairement**. En effet, correspondant à un ensemble de fonctions pour la suppression de doublons, la gestion des valeurs manquantes, l'extraction d'informations à partir des dates ou la création de nouvelles caractéristiques, elle ne représente souvent qu'un enchaînement de transformations élémentaires de la donnée.

Concernant la phase de modélisation, c'est en revanche une autre histoire. On ne peut pas tester simplement l'entraînement des modèles dans la mesure où ce dernier implique de l'aléatoire et l'utilisation d'une grande quantité de données. Le but du test n'est pas de valider le fonctionnement du modèle. Les fournisseurs du framework s'en sont chargés pour vous. Votre objectif est la performance du modèle sur un jeu de données défini. Cette mesure vous permettra de conclure sur la véracité du modèle pour certaines catégories ou sur sa capacité à atteindre un seuil de performance cible. Dans cette optique, **tester la performance de la modélisation relève plus du test d'intégration**.

Faire des choix de langages qui maximisent l'efficacité de l'équipe

Durant une phase d'industrialisation, se pose souvent la question d'un changement de langage. En effet, Data Scientists et Data Engineers ont des backgrounds différents. En termes d'outillage, les premiers tendent à préférer des langages comme **Python ou R**, tandis que les Data Engineers développent généralement en **Java ou Scala**.

Une éventuelle migration de langage va souvent se jouer sur les considérations suivantes :

- Les **contraintes des équipes de production** : une sélection restreinte de langages peut être définie par la production sans possibilité d'en ajouter.
- La **complexité du produit Data Science implémenté** : le temps de développement pour passer d'un langage à l'autre peut être un réel frein.
- La **disponibilité des fonctions et modèles** utilisés pour répondre à la problématique dans le langage cible : les Data Scientists font aussi le choix d'outils et frameworks en fonction de la présence d'algorithmes leur permettant de répondre à la question posée.

Il arrive aussi fréquemment d'avoir plus d'un langage composant le projet Data Science. Voici deux cas classiques :

- Toutes les phases de chargement de données et de Feature Engineering sont implémentées en Scala/Spark pour des questions de performance et les briques de Machine Learning sont codées en Python pour une utilisation optimale des algorithmes.
- Toute la partie amont (Feature Engineering et entraînement de modèles) est réalisée en Python sur un cluster avec les données de production. Les parties « chargement de modèle » et « prédiction » sont implémentées en Java / Scala pour répondre aux contraintes de l'environnement tiers, pour la prédiction.

Nous voyons ici tout l'intérêt de limiter drastiquement le temps de prototypage en dehors des contraintes de production. Dès que la faisabilité technique est confirmée, un travail avec un focus sur la mise en production doit être réalisé. Au fur et à mesure, **la phase d'exploration doit être faite avec les outils directement exploitables en production**.

Les équipiers doivent garder en tête l'importance de fluidifier le passage de l'exploration à l'incrément de valeur à chaque action qu'ils entreprennent.

Aujourd'hui, les **infrastructures basées sur les conteneurs** sont une réponse de plus en plus courante pour lever les barrières relatives aux langages. Leur combinaison avec des environnements proposés par les fournisseurs de Cloud accélère fortement le travail de chacun.

Packager et déployer le projet

Structurer, refactorer et tester votre code a permis de le transformer en un Produit réutilisable et maintenable. Une fois versionné à chaque nouvel incrément, ce code doit passer entre les mains d'un outil de build qui fera le packaging et potentiellement le déploiement dans un environnement cible pour son utilisation.

Le packaging et le déploiement dépendent fortement des langages utilisés. Scala et Java disposent de **Maven, SBT** ou **Gradle**. Python ne propose pas d'équivalents aussi complets pour le packaging de projets, mais **distutils** et **setuptools** proposent des alternatives intéressantes.

Au delà des outils de build, les adhérences au système cible ne sont pas les mêmes selon le langage. Là où Java et Scala ne dépendent que de la seule JVM, les autres langages seront plus sensibles à l'environnement sur lequel ils seront déployés. Là encore, les environnements basés sur les conteneurs faciliteront grandement la tâche de l'équipe.

Créer ses packages en fonction du besoin

Nous l'avons vu, la structure d'un projet Data Science contient en général des modules pour la transformation de la donnée, l'entraînement d'un modèle et la prédiction. En pratique ces trois éléments sont rarement utilisés de concert : l'entraînement d'un modèle de Machine Learning est une tâche souvent réalisée de manière décorrélée de la prédiction sur de nouvelles données.

Malgré cela, il arrive régulièrement qu'il n'y ait qu'un seul package qui soit déployé, incluant toutes les étapes du projet dans un seul et même artefact. Cet artefact sera utilisé par les différents scripts (pré-traitement, entraînement de modèles, prédiction). Cette manière de procéder simplifie le packaging et le déploiement, mais a l'inconvénient que les scripts fassent appel à un package contenant des parties inutiles, augmentant ainsi le risque de bug.

Il est plutôt recommandé de **créer un package spécifique pour chaque étape du projet**. Voici un exemple de découpage :

- **un package pour le Feature Engineering**, qui contient toutes les étapes de transformation amont de la donnée qui sont le point d'entrée de l'entraînement du modèle et de la prédiction ;
- **un package pour l'entraînement de modèles**, qui sera appelé par un script prenant en entrée un jeu de données conséquent et fournissant en sortie un modèle entraîné qui sera sauvegardé sur un environnement cible pour réutilisation ;
- **un package pour la prédiction**, qui sera appelé par un script prenant en entrée des données et une version de modèle entraîné et fournira en sortie une prédiction.

L'approche par package spécifique fait notamment sens lorsque le modèle doit être utilisé sur un environnement différent de celui ayant servi à faire l'entraînement.

L'industrialisation comme accélérateur de l'innovation

Les phases d'exploration et d'industrialisation sont trop souvent décorrélées, ce qui ralentit inexorablement le rythme de livraison d'incréments de valeur à l'utilisateur. Cette trop forte séparation alimente l'idée reçue selon laquelle l'industrialisation ne fait que freiner l'exploration et donc l'innovation. Pourtant, **une industrialisation bien réalisée permet au contraire de les accélérer**.

Dans l'optique de simplifier le code à réaliser dans les phases exploratoires et de l'optimiser pour les prochaines tâches, une bonne pratique consiste notamment à rendre **vos packages fraîchement déployés disponibles pour vos notebooks**. Utilisés comme n'importe quels autres packages par le Data Scientist, ce dernier pourra profiter des développements et transformations déjà implémentés lorsqu'il devra tester une nouvelle feature. Cette pratique permet ainsi de s'assurer que la même base de code est utilisée en exploration comme en production.

Les équipes les plus performantes et avancées se créent régulièrement leur propre **librairie interne** pour que toute l'équipe ait accès aux fonctions classiques de transformation de la donnée et de visualisation de résultats. **L'utilisation de cette librairie interne permet aussi d'harmoniser les développements de chaque Data Scientist** et de faciliter les relectures et compréhensions mutuelles du code réalisé.

En poussant un cran plus loin, nous vous recommandons d'industrialiser et de mettre à disposition, sous forme de librairies, les étapes d'évaluation de différents modèles, de calcul d'hyperparamètres optimaux, d'analyse des erreurs et de comparaison des résultats.

Automatiser et gérer ses différents workflows

Afin d'éviter d'avoir à lancer manuellement des scripts dans un environnement de production, pratique dangereuse et faisant perdre du temps, il est nécessaire d'automatiser leur lancement. **Il y a au moins deux phases à automatiser : l'entraînement de vos modèles et le déclenchement des phases de prédiction**, avec des dépendances et des fréquences différentes.

Un Produit Data Science ne se limite généralement pas à un ou deux scripts à lancer mais présente tout un workflow de transformation de la donnée. En phase de développement des projets, ces différentes étapes sont souvent lancées une par une à la main afin de vérifier les résultats intermédiaires (écriture d'une table, présence de nouvelles caractéristiques, etc.). Une automatisation du lancement du workflow permettra de garder cet enchaînement d'étapes tout en vérifiant les avancements intermédiaires.

Un **workflow** classique ressemble à ceci :

- parallélisation des tâches de préparation de différentes sources de données ;
- jointure de ces dernières ;
- agrégation et dérivation de nouvelles caractéristiques ;
- déclenchement d'un entraînement de modèle ou d'une prédiction.

Exemple de workflow d'un projet Data Science

Cette orchestration peut se faire via les outils déjà présents dans l'entreprise, mais qui sont généralement assez bas niveau. Il existe alors des outils de gestion de workflow spécialisés dans la Data, qui facilitent grandement l'exécution d'un enchaînement de tâches interdépendantes.

Apache Airflow[7] est parmi les plus complets, et fonctionne sous la forme de Directed Acyclic Graphs (DAG). En plus du lancement automatique des tâches, il permet de visualiser les dernières exécutions à travers une interface web et ainsi de détecter les problèmes survenus. Vous pourrez aussi le connecter à des systèmes d'alerting de type email, bot Slack ou system as-a-Service comme **Pager Duty**[8].

Développé par Spotify, **Luigi**[9] présente des caractéristiques similaires bien que légèrement moins complet. Il a l'avantage d'être configurable très rapidement et s'installe via une simple commande Python.

Enfin, il ne faut pas oublier **Apache Oozie**[10] qui reste l'un des standards de l'écosystème Hadoop.

Exporter et mettre à disposition son modèle

Votre package a été déployé et votre modèle entraîné. Reste maintenant la question de la mise à disposition de ce dernier à un système tiers qui va l'utiliser pour produire un résultat : c'est ce que l'on appelle le **serving**. La manière de procéder sera différente en fonction de l'utilisation du modèle : via des jobs en batch, dans une application streaming ou via des environnements différents de celui utilisé pour l'entraînement.

Cette question est cruciale et doit être posée au plus tôt dans la conception du produit. En effet, à quoi bon entraîner des centaines d'arbres de décision si le modèle final est trop long à charger alors que le besoin requiert un temps de réponse minimal ? La mise en production d'un MVP au plus tôt permet de prendre en compte tous les prérequis du produit afin d'éviter de devoir réimplémenter d'autres modèles qui leur correspondent mieux.

Lorsque les outils pour la phase d'entraînement diffèrent de ceux dédiés à la prédiction, un terrain d'entente est à trouver pour faire communiquer les deux mondes.

7. https://airflow.apache.org/

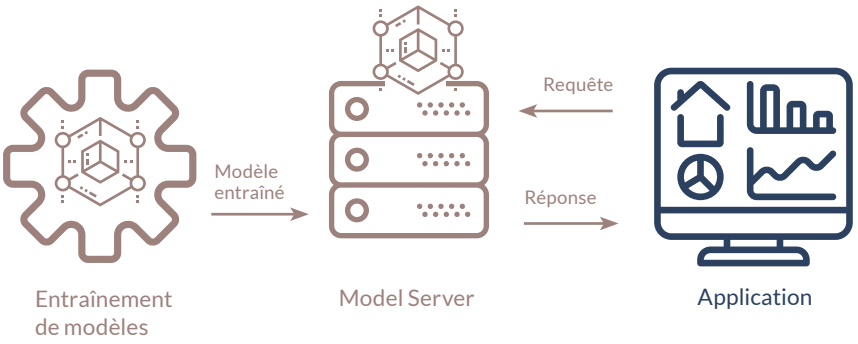
8. https://www.pagerduty.com/

9. https://luigi.readthedocs.io/en/stable/

10. http://oozie.apache.org/

Deux notions sont alors à prendre en compte :

- **L'export du modèle** dans un format cible, c'est-à-dire son format de sauvegarde / sérialisation, qui peut être dépendant ou non du framework utilisé pour l'entraînement.
- Le **serving du modèle** sérialisé à proprement parler, c'est-à-dire sa mise à disposition pour un système tiers.



Serving d'un modèle de Machine Learning

Exporter son modèle

La solution la plus immédiate pour l'export d'un modèle est d'**utiliser le format de sérialisation associé au framework** ayant été employé pour l'entraînement. En effet, tout bon framework de Machine Learning propose un ou plusieurs moyens d'exporter les modèles entraînés.

Cette approche est la plus simple et la plus directe, mais ne garantit pas la portabilité si rien n'existe de l'autre côté pour importer le format d'origine.

Nous observons alors une tendance grandissante à passer par un **format de sérialisation pivot** afin de découpler l'environnement d'entraînement de celui de prédictions. Ceci s'observe souvent lorsqu'il s'agit de basculer vers un système tournant sur une JVM alors que l'entraînement a été réalisé avec une autre stack logicielle (Python pour ne pas le nommer).

Même si la promesse de la portabilité est attrayante, il faut toutefois souligner que ce n'est pas toujours possible pour tous les types de modèles. Bien souvent, on constate que les frameworks ne possèdent pas encore de convertisseur ad-hoc ou que le support n'est pas complet.

	Format standard du framework				Format pivot indépendant du framework		
Format	Pickle	Protobuf	HDF5	Parquet	PMML	PFA	ONNX
Utilisation	Scikit-learn	TensorFlow	Keras	Spark	Basé sur du XML	Basé sur du JSON	Pour l'interopérabilité entre les modèles de Deep Learning

Formats de sérialisation et dépendances aux frameworks

Le serving de modèles

Dans de nombreux cas de figure, le choix de l'équipe se porte sur **le développement d'une API pour chaque framework utilisé** (et donc chaque format de stockage), laissant la possibilité aux Data Scientists d'utiliser l'outil de leur choix pour le développement et l'entraînement de leurs modèles.

Cette manière de faire est probablement la plus répandue aujourd'hui dans la mesure où le nombre de frameworks utilisés reste en général restreint au sein d'une même équipe. La souplesse d'utilisation finale du modèle (*batch/streaming*) est ainsi conservée. C'est aussi la seule solution s'offrant à nous dans un cas particulier où l'entraînement et la prédiction se font à la volée (le modèle se met à jour au fur et à mesure de l'arrivée de la donnée, c'est ce que l'on appelle communément du *Online Learning*).

Il arrive parfois que le Use Case et l'infrastructure permettent de conserver les mêmes outils pour l'entraînement comme pour la création des prédictions, mais il arrive régulièrement qu'il soit nécessaire de réimplémenter toute une brique spécifique pour la prédiction, dans un système tiers. Les applications temps réel exigent souvent de le faire quand sont impliqués des bus de messages tels que **Kafka**. Les sérialisations intermédiaires peuvent alors avoir de l'intérêt. Elles permettent, par exemple, de passer d'un modèle entraîné via Spark à un modèle mis à disposition via un topic Kafka spécifique, sans avoir à utiliser Spark pour charger le modèle.

Afin de pallier la difficulté que représente le serving de modèles, certains frameworks et plateformes proposent **des solutions d'utilisation de ces modèles de bout en bout, de leur entraînement à leur déploiement**.

Outil	Caractéristiques
TensorFlow Serving	Utilisé en production chez Google
Sérialisation de pipelines avec Spark	Serving de pipelines complets, réutilisables dans tout job Spark ou bien via Spark Streaming
MXNet Model Server	Créé par AWS, compatible avec Gluon
MLeap	Format de sérialisation et API Server Supporte Spark, Scikit-learn et TensorFlow
clipper.ai	Simplification du déploiement et intégration de modèles dans des outils tierces via une interface REST
Services managés	Gestion complète du cycle de vie d'un modèle Ex: AWS SageMaker, Google Cloud ML

Quelques outils pour le serving de modèles de Machine Learning

Que choisir ?

Devant cet éventail de possibilités, l'idée directrice à conserver est la suivante : **laisser à chacun la possibilité de travailler avec des outils qui lui permettent d'exprimer au mieux ses compétences.**

En ce qui concerne l'export et le serving de modèles, cela revient à établir un contrat d'interface entre le modèle sauvegardé et son utilisation par un système tiers.

Le choix d'une solution plutôt qu'une autre **va aussi être conditionné par l'utilisation cible du modèle**. Les déploiements et les besoins seront en effet différents si l'utilisation du modèle se fait par batch ou dans des applications temps réel.

On comprend l'importance d'une réflexion en amont sur la mise en production sous peine de devoir réimplémenter la quasi totalité de ce dernier s'il n'est pas utilisable dans les conditions cibles souhaitées.

Monitorer tout le workflow pour réagir au plus tôt

Tout projet doit être monitoré efficacement, ne serait-ce que pour des besoins de reporting et d'alerting. Et nous ne parlons pas uniquement de logs techniques, les informations à faire remonter sont nombreuses :

- **Performances dans le temps** du modèle déployé pour s'assurer qu'il n'y ait pas de dégradation ;
- **Erreurs faites par le modèle déployé** sur différents axes (temps, catégories, etc.) ;
- **Qualité des données** ingérées en entrée.

Nous avons vu dans la partie précédente qu'un monitoring efficace permet le pilotage du produit. Mais d'un point de vue technique, il peut avoir d'autres vertus : **le développement d'une stratégie de choix des modèles et l'automatisation de la gestion de ces derniers.**

Développer des stratégies de choix de modèles

Sans une mesure adéquate de la cible de votre produit, il sera impossible de choisir la meilleure stratégie et le meilleur modèle. Le monitoring doit donc être intégré dès l'implémentation du MVP de votre produit.

Pour le déploiement d'un système de recommandation, l'équipe souhaitera probablement tester les performances de plusieurs types de modèles. Ce choix pourra être fait via des méthodes d'**A/B Testing**[11] ou des techniques comme les **multi-armed bandits**[12] pour converger itérativement vers la solution optimale.

11. https://fr.wikipedia.org/wiki/Test_A/B

12. https://en.wikipedia.org/wiki/Multi-armed_bandit

Automatiser la gestion des modèles

Grâce à la mise en place d'un système d'alerting et de suivi des performances sous différents angles de votre modèle, vous serez en mesure d'automatiser des tâches qui habituellement se font trop tardivement. Passer sous un seuil de performance défini provoquera le **déclenchement d'un ré-entraînement du modèle actuel**.

De même, lors du déploiement d'un nouveau modèle en production, il est crucial de monitorer les performances afin d'avoir une **stratégie de rollback vers une ancienne version** ou un mode dégradé au plus tôt afin d'éviter des pertes qui peuvent être conséquentes.

Les outils pour le monitoring

En termes d'outillage, les logiciels de dashboarding comme **Grafana**[13], **Kibana**[14] ou **Apache Superset**[15] sont généralement de bonnes options.

Vous pouvez aussi vouloir utiliser des outils de type notebook pour faire votre monitoring puisqu'ils seront nativement liés à vos données pour vos tâches exploratoires. **Apache Zeppelin**[16] est un bon candidat pour cela.

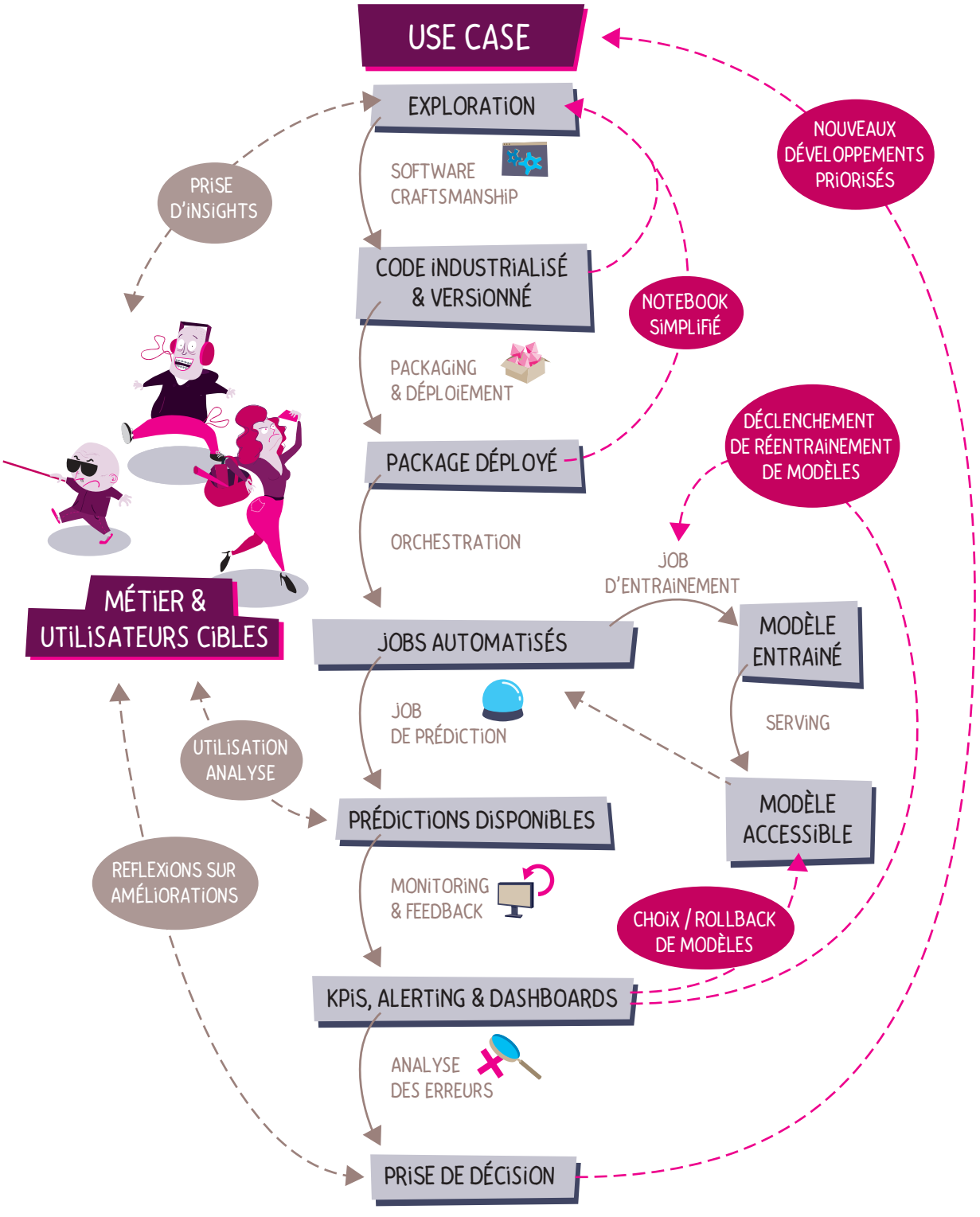
> *Tout projet doit être monitoré efficacement, ne serait-ce que pour des besoins de reporting et d'alerting.*

13. https://grafana.com/

14. https://www.elastic.co/fr/products/kibana

15. https://superset.incubator.apache.org/

16. https://zeppelin.apache.org/



Cycle de vie d'un Produit Data Science

Gérer la jungle des modèles et expérimentations menées

Au fur et à mesure que votre produit évolue et que votre équipe grandit, de nouvelles problématiques apparaissent. Citons-en quelques unes :

- Les Data Scientists ont déjà travaillé sur plusieurs dizaines de notebooks différents et il devient impossible de s'y retrouver ni de se rappeler de la raison d'être des notebooks les plus anciens ;
- Le nombre de modèles testés et passés en production, qui était faible et gérable au début, devient trop grand et il est impossible de savoir à quoi un modèle fait référence (sur quelles données il a été entraîné, avec quelles caractéristiques, avec quelles performances, etc.) ;
- Une nouvelle personne dans l'équipe se lance dans l'exploration d'une hypothèse sans se rendre compte qu'une expérimentation similaire a déjà été menée et donc que quelqu'un a la connaissance.

Nous voyons apparaître un fort besoin d'harmonisation, de capitalisation sur la connaissance et de création d'un journal de bord des modèles créés et des expériences menées. Ce sont des points clés pour établir une véritable stratégie d'utilisation des modèles.

Il peut être intéressant de se tourner vers les nouveaux concepts de **Model Repository** et de **Machine Learning Platform**, qui commencent à faire leur apparition dans l'écosystème Data Science. Cependant, ces outils ne remplaceront jamais des discussions d'équipe sur l'amélioration continue et le partage.

Cette idée de Machine Learning Platform, permettant de gérer tout le cycle de vie d'un Produit Data Science, a rapidement intéressé de grandes entreprises comme Uber avec sa plateforme **Michelangelo**[17] ou encore Google avec **TFX**[18] (TensorFlow eXtended).

17. https://eng.uber.com/michelangelo/

18. https://www.tensorflow.org/tfx/

Il existe aussi de plus en plus de projets Open Source qui permettent d'adresser cette problématique. Citons parmi eux :

- **DVC**[19] (Data Science Version Control) : cette librairie permet à la fois de gérer le contrôle de versions des projets de Machine Learning, le management des expériences et le déploiement en production ;
- **MLFlow**[20] : développé par Databricks, cet outil offre un cadre de développement complet pour les projets Data Science. Bien qu'encore en version alpha à la date d'écriture de ce TechTrends, il devrait s'imposer rapidement comme un standard. Il est possible grâce à MLFlow de packager et réutiliser ses modèles à sa guise sur des plateformes différentes, de traquer et partager les expérimentations faites et de déployer les modèles sur une des nombreuses plateformes prévues via des connecteurs. Il permet aussi de travailler avec des langages et des frameworks différents.

Une usine logicielle Data Science se construit au fur et à mesure de l'évolution du produit. Elle doit permettre d'accélérer l'innovation et de fluidifier le passage en production.

> *Une usine logicielle Data Science doit permettre d'accélérer l'innovation et de fluidifier le passage en production.*

19. https://dvc.org/

20. https://mlflow.org/

TAKE AWAY PRODUITS DATA SCIENCE

FLUIDIFIER

- Industrialisez et packagez votre projet afin de rendre le produit robuste et utilisable en toute confiance ;
- Accélérez les phases exploratoires grâce à des librairies internes et à la réutilisation de code industrialisé ;
- Automatisez la gestion de votre workflow afin de contrôler chacune des étapes et leurs interdépendances ;
- Réfléchissez dès le départ à la manière dont sera utilisé un modèle en production pour avoir une stratégie d'export et de serving adéquate ;
- Priorisez les prochains développements et déclenchez des ré-entraînements ou des rollbacks de modèle en cas de problème grâce au monitoring des données et des performances ;
- Pensez à l'utilisation de *Machine Learning Platform* pour avoir une réelle stratégie de gestion de vos modèles et éviter la confusion.

Pour aller plus loin : passer à l'échelle dans la conception de Produits Data Science

Vos premiers Produits Data Science sont maintenant réalisés dans les règles de l'art et sont des succès. Vous vous posez maintenant la question de passer à l'échelle : augmenter la taille de l'équipe et traiter plusieurs Use Cases en parallèle. De nouvelles problématiques vont se dresser sur votre chemin. Nous vous proposons quelques questions qu'il est bon de se poser pour aller dans cette direction.

Comment capitaliser sur les connaissances acquises sur la donnée ?

Les membres de l'équipe, au fur et à mesure de leurs explorations et développements, ont acquis une connaissance non négligeable sur la donnée, ce qu'elle représente, ses pièges et sa qualité. Cette connaissance mérite d'être partagée, transmise et répertoriée afin qu'une nouvelle personne, travaillant sur les mêmes données, n'ait pas à parcourir le même chemin pour sa compréhension et son utilisation optimale.

Quelques idées :

- Organisez régulièrement des sessions de partage de la connaissance ;
- Mettez en place des communautés de pratiques ;
- Mettez en place un **Knowledge Repository**[21] pour versionner et partager la connaissance acquise lors des explorations ;
- Mettez en place et enrichissez un **Data Catalog**.

Comment capitaliser sur les développements déjà faits ?

Si les Use Cases ont été réalisés selon les **principes du Software Craftsmanship**, vous devriez avoir à votre disposition un ensemble de transformations élémentaires testées et une structure de projet éprouvée. Ces éléments ont de fortes chances de ne pas être spécifiques à vos précédents Use Cases et pourront être réutilisés dans les prochains. Pensez donc à réutiliser la structure de vos précédents projets Data Science et en faire un template réutilisable à chaque nouveau projet.

21. https://knowledge-repo.readthedocs.io/en/latest/

Comment capitaliser sur les Use Cases réalisés ?

Lorsque vous avez investi beaucoup de temps et d'efforts pour créer un Produit Data Science, plutôt que de passer à un nouveau sujet complètement différent, cherchez à **prioriser des Use Cases similaires qui seront réalisables en un temps restreint**.

Une première idée : réfléchissez aux « quick-wins » qui pourraient dériver des Use Cases déjà implémentés.

Comment établir une gouvernance pour l'arbitrage entre les Use Cases ?

Plus votre équipe accumulera des Use Cases qui auront répondu au besoin, plus les demandes entrantes s'accumuleront. L'équipe Data va s'agrandir, mais, inévitablement, va devenir un goulot d'étranglement et ne pourra pas développer tous les produits potentiels. Il faudra alors **établir une gouvernance pour arbitrer entre les Use Cases**.

Quelques idées :

- Mettez en place un processus de Continuous Discovery : industrialisez la démarche décrite dans la partie « Explorer » pour identifier les potentiels Use Cases puis évaluez leur valeur et leur faisabilité.
- Mettez en place des KPIs de gains marginaux sur les produits en cours de développement pour repérer les plateaux (i.e. les moments où les développements additionnels apportent de moins en moins de valeur).
- Créez une gare de triage et des instances de pilotage inspirées des directions produit.

CONCLUSION

La conception d'un Produit Data Science est un parcours semé d'embûches. Sans s'être posé les bonnes questions au démarrage, les probabilités d'échec sont grandes.

Nous parlons bien de **Produit Data Science, et non de projet**. Cette notion fait toute la différence. La vie du produit ne s'arrête pas après sa livraison. Au contraire, elle évolue constamment et s'adapte au fur et à mesure de sa confrontation à la réalité de l'utilisation. En prenant en compte dès le départ les critères nécessaires pour répondre au besoin, de l'identification du problème et de ses interlocuteurs jusqu'à l'activation adéquate des résultats, vous vous assurez d'avoir un **MVP solide qui constitue une bonne base**. Il vous faudra pour cela mettre en place **une usine logicielle Data Science, une équipe dédiée, un cadre agile et des éléments de pilotage pertinents**.

Nous ne le répéterons jamais assez : un modèle de Machine Learning n'est pas une fin en soi. C'est un composant d'un produit complet. En bons ingénieurs, nous avons tendance à vouloir trouver une solution technique à un problème technique. L'essor de l'Intelligence Artificielle amène à penser (à tort) que l'algorithme est l'unique problématique à résoudre autour de la Data. C'est bien entendu faire fausse route lorsque l'on cherche à concevoir un produit activable et ayant une réelle plus-value pour l'entreprise.

Grâce à un travail minutieux d'identification et de conception du produit dans sa globalité, vous serez en mesure de faire de vos objectifs initiaux des réussites, sur lesquelles vous pourrez capitaliser. Si le focus sur la réponse au besoin est toujours présent, et que l'objectif de mise en production des développements est défini au plus tôt, alors, à ce moment-là, vous pourrez **faire le choix des bons outils et des bons algorithmes pour résoudre le problème**, et bénéficier pleinement des compétences de chaque membre de votre équipe.

Tout au long de ce TechTrends, nous vous avons fourni une démarche et des conseils, tant organisationnels que méthodologiques et techniques, pour construire vos Produits Data Science de bout en bout. Nous espérons que vous en ferez bon usage !

TAKE AWAY PRODUITS DATA SCIENCE

EXPLORER

- Concentrez-vous sur vos objectifs stratégiques avant même de penser à la réalisation en explorant vos points de douleur les plus forts ;
- Menez des ateliers d'idéation avec des profils hétérogènes afin d'identifier les idées les plus porteuses de valeur ;
- Itérez sur votre vision produit, en vous aidant par exemple d'un Data Science Product Canvas ;
- Prototypez pour lever les plus grosses incertitudes avant de vous lancer dans la réalisation d'un MVP.

S'ORGANISER

- Constituez une équipe pluridisciplinaire pour qu'elle puisse être autonome dans la réalisation du produit ;
- Définissez un MVP qui permettra de récolter rapidement des retours utilisateurs et de prioriser vos prochains développements ;
- Matérialisez le chemin pour atteindre la vision cible grâce à une Story Map ;
- Mettez en place les interactions et outils qui permettent un pilotage éclairé des itérations ;
- Adoptez les spikes et SBCE pour gérer l'incertitude.

FLUIDIFIER

- Industrialisez et packagez votre projet afin de rendre le produit robuste et utilisable en toute confiance ;
- Accélérez les phases exploratoires grâce à des librairies internes et à la réutilisation de code industrialisé ;
- Automatisez la gestion de votre workflow afin de contrôler chacune des étapes et leurs interdépendances ;
- Réfléchissez dès le départ à la manière dont sera utilisé un modèle en production pour avoir une stratégie d'export et de serving adéquate ;
- Priorisez les prochains développements et déclenchez des ré-entraînements ou des rollbacks de modèle en cas de problème grâce au monitoring des données et des performances ;
- Pensez à l'utilisation de Machine Learning Platforms pour avoir une réelle stratégie de gestion de vos modèles et éviter la confusion.


TAKE AWAY
PRODUITS
DATA SCIENCE
TRENDS
TAKE AWAY

Remerciements

Anne Beauchart, Damien Baron, Alexandre Brebant, Aurore De Amaral, Christophe Heubès, Christophe Julien, Cynthia Muzet, Guillaume Albini, Sandra Pietrowska, Sylvain Lequeux, Anne-Sophie Girault, Hamza Bachar, Clément Rochas, Diana Ortega, Julien Smadja, Elhadi Cherifi, Guillaume Matthieu, Isabelle Roques, Klervi Falip, Laetitia Janné, Pauline Tirman, Raphaël Matusiak, Sameh Ben Fredj, Sinan Serdaroglu, Soufiane El Alami, Simon Joliveau-Breney, Vanessa Guilloteau, Tristan Boulfroy, Céline Winant-Pateron, Victor Dulieu, Amélie Darcy, Diane Paul, Marine Petitpas, Solène Mensah, Victor Cocagne, Adeline Delort, Cyrielle Jourdren, Stéphan Babou, Cindy Rodrigues, Romain Sagean, Romain Ardiet et Nicolas Laille.

Les auteurs Xebia

Pablo
Lopez

Yoann
Benoit

Nelson
Dufossé

L' auteur Thiga

Nathan
Chauliac

Xebia

Software Development **done right**

THIGA

Product Management. Redefined

Xebia France
156 bd Haussmann, 75008 Paris
xebia.fr

Thiga
23 rue taitbout, 75009 Paris
thiga.co