Mettre les technologies cloud au service de la production statistique

Romain Avouac
Insee
romain.avouac@insee.fr

Thomas Faria
Insee
thomas.faria@insee.fr

Frédéric Comte
Insee
frederic.comte@insee.fr

2025-02-08

Résumé French

1 Introduction

L'exploitation de nouvelles sources de données en complément des enquêtes traditionnelles est une orientation majeure du Système Statistique Européen (SSE) pour améliorer les processus de production statistique. Cette évolution s'accompagne d'innovations à la fois méthodologiques et dans les systèmes d'information afin de tirer parti du potentiel de ces sources — plus grande disponibilité, résolution spatio-temporelle accrue, etc. — tout en faisant face à leur complexité et à leurs limites. Parmi ces innovations figurent les méthodes d'apprentissage automatique et leurs applications prometteuses dans les domaines du codage et de la classification, des redressements et de l'imputation [1]. Les multiples défis auxquels font face les instituts statistiques dans ce contexte d'évolution sont abordés dans le *Mémorandum de Bucarest sur les statistiques publiques dans une société numérisée*, qui anticipe que « la variété des nouvelles sources de données, paradigmes computationnels et outils nécessitera des adaptations de l'architecture métier statistique, des processus, des modèles de production, des infrastructures informatiques, des cadres méthodologiques et de qualité, ainsi que des structures de gouvernance correspondantes », et invite en conséquence le SSE à évaluer les adaptations requises et à les prioriser [2]. Cette évolution est également largement visible dans le cadre du service statistique public (SSP), dont elle constitue l'une des lignes directrices de la stratégie à horizon 2025 [3].

Face à ces transformations, de nombreux travaux ont été menés dans le cadre de projets innovants visant à qualifier l'utilisation de sources de données non-traditionnelles dans la production de statistiques officielles. Dans le cadre des projets ESSnet Big Data I (2016-2018) et II (2018-2020), les instituts statistiques nationaux (INS) ont travaillé sur une large gamme de thématiques (offres d'emploi en ligne, transactions financières, traces GPS, etc.) afin de constituer les briques nécessaires pour intégrer ces sources dans les processus de production et identifier leurs limites [4]. A l'Insee, les travaux sur l'exploitation des données mobiles [5] ou des données de caisse [6] ont

permis d'illustrer le potentiel de ces sources pour construire de nouveaux indicateurs ou raffiner des indicateurs existants. Néanmoins, si un travail conséquent a été consacré au développement de cadres méthodologiques [7], [8], de lignes directrices sur la qualité [9], ainsi qu'à la conception de processus sécurisant l'acquisition de données dans le cadre de partenariats avec des acteurs privés [10], les infrastructures informatiques et les compétences nécessaires pour gérer ces nouveaux objets sont restées peu abordées dans la littérature.

On désigne généralement par « *big data* » les données qui se distinguent par leur volume (souvent de l'ordre de plusieurs centaines de Go voire du To), leur vélocité (vitesse de génération, parfois proche du temps réel) ou leur variété (données structurées mais aussi non structurées, telles que du texte ou des images). Cette caractérisation s'applique très naturellement aux données massives générées de manière automatique par les comportements individuels (données mobiles, données de caisses) ou encore aux données récupérées depuis internet via des méthodes de *web scraping*. Mais elle est également pertinente pour caractériser certaines sources de nature administrative déjà utilisées pour la production statistique. Le projet Résil en est une bonne illustration, qui repose sur l'appariement de multiples sources administratives volumineuses (DSN, POTE, etc.) devant être accueillies avec différentes temporalités et dans des formats hétérogènes, plus ou moins structurés [11]. Dans les deux cas, l'intégration de telles sources dans un processus de production statistique pose des défis qui se situent au confluent de la méthodologie statistique et de la technique informatique, et relèvent ainsi du domaine de la *data science*. Dans ses multiples acceptions, le terme *data scientist* reflète en effet l'implication croissante des statisticiens dans le développement informatique et l'orchestration de leurs opérations de traitement des données, au-delà des seules phases de conception ou de validation [12]. Hors si l'on observe un nombre croissant de statisticiens publics formés aux méthodes de *data science*, leur capacité à tirer pleinement parti des sources non-traditionnelles pour la production statistique se heurte à plusieurs défis.

Un premier défi réside dans l'absence d'infrastructures informatiques adaptées aux nouvelles sources de données auxquelles les INS ont désormais accès, ainsi qu'au besoin croissant de nouvelles méthodes statistiques. Par exemple, les sources *big data* nécessitent de très grandes capacités de stockage et s'appuient souvent sur des infrastructures et des méthodes de calcul distribués pour être traitées en temps raisonnable [13]. De même, l'adoption de nouvelles méthodes statistiques basées sur des algorithmes d'apprentissage automatique requiert des capacités informatiques — en particulier des GPU (processeurs graphiques) dans le cadre du traitement du texte ou de l'image — pour paralléliser massivement les calculs [14]. De telles ressources sont rarement disponibles dans les infrastructures informatiques traditionnelles. Lorsque des infrastructures de calcul adaptées sont disponibles, comme les supercalculateurs (HPC) utilisés dans certains domaines de recherche, elles nécessitent des compétences spécifiques, notamment pour leur mise en place et leur maintenance, qui sont rarement disponibles au sein des INS. Pour lever cette barrière, il est nécessaire d'adopter des infrastructures informatiques qui reflètent les besoins des projets de *data science* actuels en permettant de découpler stockage et traitement de la donnée, afin de s'adapter rapidement à l'évolution des besoins.

Un autre défi majeur pour les statisticiens est de disposer d'environnements de développement flexibles leur permettant d'expérimenter plus librement. Cette agilité est limitée lorsque les envi-

ronnements de calcul dépendent excessivement des départements informatiques pour provisionner des ressources ou installer de nouveaux logiciels. Dans les configurations traditionnelles où les statisticiens effectuent leurs calculs sur des ordinateurs personnels ou des bureaux virtuels sur des architectures centralisées[1], les départements informatiques privilégient généralement la sécurité et la stabilité du système là où l'innovation réside dans la capacité à intégrer rapidement de nouveaux outils. De plus, la rigidité de ces environnement rend difficile la mise en œuvre de bonnes pratiques de développement, comme le travail collaboratif qui nécessite des environnements permettant de partager facilement des expérimentations avec ses pairs, ou encore la reproductibilité qui n'est pleinement possible que dans des environnements dont les statisticiens peuvent eux-mêmes spécifier les caractéristiques (version du langage statistique, version des packages, librairies système nécessaires, etc.). L'enjeu ici est donc de dépasser l'opposition traditionnelle entre sécurité et innovation en adoptant des infrastructures propices à la seconde sans compromettre la première.

Un troisième défi concerne la difficulté de passer des expérimentations innovantes à des solutions en production. Même lorsque les statisticiens ont accès à des environnements « bac à sable » leur permettant de développer une application de recette ou d'entraîner un modèle, la transition vers le déploiement en production de tels objets est coûteuse. Les environnements de production diffèrent souvent des environnements de développement, ce qui peut entraîner des coûts de développement supplémentaires importants pour passer d'une preuve de concept à une solution industrialisée qui rend du service à des utilisateurs dans la durée. Par exemple, dans le cas des projets d'apprentissage automatique, les modèles déployés nécessitent un suivi rigoureux pour s'assurer qu'ils conservent leur précision et leur utilité au fil du temps, et requièrent généralement des améliorations périodiques ou continues. Ainsi, notre choix doit s'orienter vers des infrastructures informatiques qui d'une part permettent de mettre à disposition des environnements de développement qui ressemblent au maximum aux environnements de production, et qui favorisent une collaboration plus continue entre statisticiens et équipes informatiques dans la gestion du cycle de vie des projets de *data science*.

Ces différents défis ont un thème sous-jacent commun : le besoin d'une plus grande autonomie. La capacité des méthodes de *data science* à améliorer la production des statistiques officielles dépend crucialement de la capacité des statisticiens à intégrer de nouvelles sources et de nouvelles méthodes de traitement de la donnée dans les processus statistiques. Et cette capacité dépend à son tour de la disponibilité d'environnements de calcul adaptés aux besoins de la *data science* moderne (capacités de stockage, infrastructures distribuées, disponibilité de GPUs) qui permettent l'application des bonnes pratiques de développement et facilitent ainsi le passage en production des projets statistiques. Cet article vise à montrer comment les technologies *cloud* permettent de répondre à ces différents défis et peuvent être mises à profit pour favoriser l'innovation dans la production statistique.

[1] `AUSv3` est un exemple d'une telle infrastructure. Les utilisateurs y accèdent via leur poste de travail, qui sert de point d'accès à un bureau virtuel qui « reproduit » l'expérience habituelle du poste de travail. Néanmoins, les calculs qui sont lancés — via `R` ou `Python` par exemple — sont effectués sur des machines virtuelles (VM) de calcul dédiées, et non sur le poste de travail lui-même.

La section 2 offre une analyse approfondie des derniers développements de l'écosystème de la donnée, mettant en lumière les choix technologiques qui ont façonné le développement d'un environnement de calcul moderne à l'Insee, adapté aux besoins spécifiques de la *data science*. Nous montrons comment certaines technologies dites *cloud-native*, comme la conteneurisation et le stockage objet, permettent de créer des environnements évolutifs et flexibles qui favorisent l'autonomie tout en promouvant la reproductibilité des projets statistiques. Malgré leurs atouts, la disponibilité de ces technologies n'implique pas leur adoption dans l'organisation, dans la mesure où elles s'avèrent complexes à configurer du point de vue informatique et nécessitent une adaptation de leurs pratiques de la part des statisticiens pour les exploiter dans le cadre de projets statistiques. Dans la section 3, nous détaillons comment le projet `Onyxia`, développé à l'Insee, a permis de mettre les technologies *cloud* au service des statisticiens grâce à une interface dynamique et un catalogue étendu de services de *data science* prêts à l'emploi. Enfin, la section 4 illustre l'application pratique de ces technologies à un projet innovant de l'Insee : la classification des activités des entreprises françaises (APE) à l'aide de méthodes d'apprentissage automatique. Ce retour d'expérience vise à montrer comment l'utilisation de ces technologies permet de faciliter et fiabiliser la mise en production de modèles d'apprentissage en permettant d'appliquer les bonnes pratiques issues du *MLOps*.

2 Principes pour construire une architecture de données moderne et flexible pour les statistiques publiques

Avec l'émergence de sources de données massives et de nouvelles méthodologies prometteuses pour améliorer le processus de production des statistiques publiques, les statisticiens formés aux techniques de *datascience* sont désireux d'innover. Cependant, leur capacité à le faire est limitée par plusieurs défis. L'un des principaux défis réside dans le besoin d'une plus grande autonomie — qu'il s'agisse de dimensionner la puissance de calcul en fonction des chaînes de productions statistiques, de déployer des preuves de concept avec agilité et de manière collaborative, etc. Dans ce contexte, notre objectif était de concevoir une plateforme de *datascience* qui non seulement gère efficacement les données massives, mais qui renforce également l'autonomie des statisticiens. Pour y parvenir, nous avons étudié l'évolution de l'écosystème des données afin d'identifier les tendances significatives susceptibles de surmonter les limitations mentionnées précédemment[2]. Nos conclusions indiquent que l'adoption des technologies *cloud*, en particulier les conteneurs et le stockage objet, est essentielle pour construire des infrastructures capables de gérer des ensembles de données volumineux et variés de manière flexible et économique. De plus, ces technologies améliorent considérablement l'autonomie, facilitant ainsi l'innovation et favorisant la reproductibilité dans la production des statistiques publiques.

[2]En préambule à cette analyse, il convient de noter que, bien que nous ayons fait de notre mieux pour ancrer nos réflexions dans la littérature académique, une grande partie de nos observations provient de connaissances informelles acquises grâce à une veille technologique assidue et continue. Dans l'écosystème des données, qui est en constante évolution, les articles de recherche traditionnels cèdent de plus en plus la place aux billets de blog en tant que principales références pour les développements de pointe. Ce changement s'explique en grande partie par le rythme soutenu auquel les technologies et méthodologies liées aux données massives progressent, rendant souvent le processus de publication formelle trop long pour la diffusion de connaissances et d'innovations.

2.1 Limites des architectures traditionnelles pour les big data

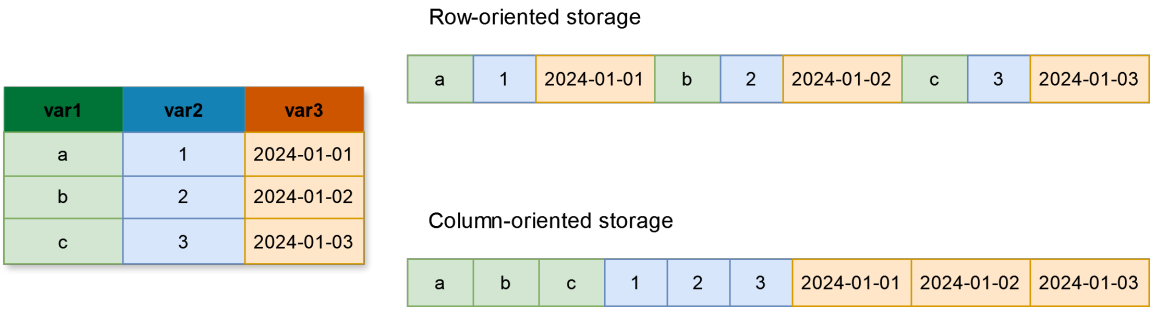
Au cours de la dernière décennie, le paysage des *big data* s'est transformé de manière spectaculaire. Suite à la publication des articles fondateurs de Google introduisant le *MapReduce* [15]–[17], les systèmes basés sur Hadoop sont rapidement devenus l'architecture de référence dans l'écosystème des données massives, salués pour leur capacité à gérer d'importants ensembles de données grâce aux calculs distribués. L'introduction d'Hadoop a marqué une étape révolutionnaire, permettant aux organisations de traiter et d'analyser des données à une échelle sans précédent. En substance, Hadoop offrait aux entreprises des capacités complètes pour l'analyse de *big data* : des outils pour la collecte, le stockage des données (HDFS) et des capacités de calcul (notamment Spark) [18], expliquant ainsi son adoption rapide dans les industries.

À la fin des années 2010, les architectures basées sur Hadoop ont connu un net déclin de popularité. Dans les environnements Hadoop traditionnels, le stockage et le calcul étaient co-localisés par construction : si les données sources étaient réparties sur plusieurs serveurs (scalabilité horizontale), chaque section des données était directement traitée sur la machine hébergeant cette section, afin d'éviter les transferts réseau entre serveurs. Dans ce paradigme, la mise à l'échelle de l'architecture impliquait souvent une augmentation linéaire à la fois des capacités de calcul et de stockage, indépendamment de la demande réelle. Dans un article volontairement provocateur et intitulé « Big Data is Dead » [19], Jordan Tigani, l'un des ingénieurs fondateurs de Google BigQuery, explique pourquoi ce modèle ne correspond plus à la réalité de la plupart des organisations centrées sur les données. Premièrement, parce que « dans la pratique, la taille des données augmente beaucoup plus rapidement que les besoins en calcul ». Alors que la quantité de données générées et nécessitant donc d'être stockées peut croître de manière linéaire au fil du temps, il est généralement vrai que nous n'aillons besoin d'interroger que les portions les plus récentes, ou seulement certaines colonnes et/ou groupes de lignes. Par ailleurs, Tigani souligne que « la frontière du *big data* ne cesse de reculer » : les avancées dans les capacités des serveurs et la baisse des coûts du matériel signifient que le nombre de charges de travail ne tenant pas sur une seule machine — une définition simple mais efficace du *big data* — a diminué de manière continue. En conséquence, en séparant correctement les fonctions de stockage et de calcul, même les traitements de données substantiels peuvent finir par utiliser « beaucoup moins de calcul que prévu […] et pourraient même ne pas avoir besoin d'un traitement distribué du tout ».

Ces observations concordent fortement avec nos propres constats à l'Insee au cours des dernières années. Par exemple, une équipe de l'Insee a mis en place un cluster Hadoop en tant qu'architecture alternative à celle déjà utilisée pour traiter les données des tickets de caisse dans le cadre du calcul de l'indice des prix à la consommation. Une accélération des opérations de traitement des données pouvant aller jusqu'à un facteur 10 a été obtenue, pour des opérations qui prenaient auparavant plusieurs heures [20]. Malgré cette amélioration des performances, ce type d'architectures n'a pas été réutilisé par la suite pour d'autres projets, principalement parce que l'architecture s'est révélée coûteuse et complexe à maintenir, nécessitant une expertise technique spécialisée rarement disponible au sein des Instituts Nationaux de Statistiques (INS) [21]. Bien que ces nouveaux projets puissent encore impliquer des volumes de données massifs, nous avons observé que des traitements efficaces pouvaient être réalisés à l'aide de logiciels conventionnels (R, Python) sur des systèmes à nœud unique, en tirant parti des récentes innovations importantes de l'éco-

système des données. Tout d'abord, en utilisant des formats de stockage efficaces tels qu'Apache Parquet [22], dont les propriétés — stockage en colonnes [23] (voir Figure 1), optimisation pour les analyses « écrire une fois, lire plusieurs fois », possibilité de partitionner les données, etc. — le rendent particulièrement adapté aux tâches analytiques comme celles généralement effectuées dans les statistiques publiques [16]. Ensuite, en effectuant des calculs optimisés en mémoire tels qu'Apache Arrow [24] ou DuckDB [25] le proposent. Également basés sur une représentation en colonnes — travaillant ainsi en synergie avec les fichiers Parquet — ces deux logiciels améliorent considérablement les performances des requêtes de données grâce à l'utilisation de l'"évaluation paresseuse" (*lazy evaluation*) : au lieu d'exécuter de nombreuses opérations distinctes (par exemple, sélectionner des colonnes et/ou filtrer des lignes, puis calculer de nouvelles colonnes, puis effectuer des agrégations, etc.), ils les traitent toutes en une fois de manière plus optimisée. En conséquence, les calculs se limitent aux données effectivement nécessaires pour les requêtes, permettant le traitement de données beaucoup plus importantes que la mémoire disponible sur des machines classiques à nœud unique.

Figure 1. – Représentation orientée ligne et orientée colonne d'un même jeu de données.



Note: De nombreuses opérations statistiques sont analytiques (OLAP) par nature : elles impliquent la sélection de colonnes spécifiques, le calcul de nouvelles variables, la réalisation d'agrégations basées sur des groupes, etc. Le stockage orienté ligne n'est pas bien adapté à ces opérations analytiques, car il nécessite de charger l'ensemble du jeu de données en mémoire afin d'effectuer une requête. À l'inverse, le stockage orienté colonne permet de ne lire que les colonnes de données pertinentes, ce qui réduit considérablement les temps de lecture et de traitement pour ces charges de travail analytiques. En pratique, les formats colonnes populaires tels que Parquet utilisent une représentation hybride : ils sont principalement orientés colonne, mais intègrent également un regroupement astucieux basé sur les lignes pour optimiser les requêtes de filtrage.

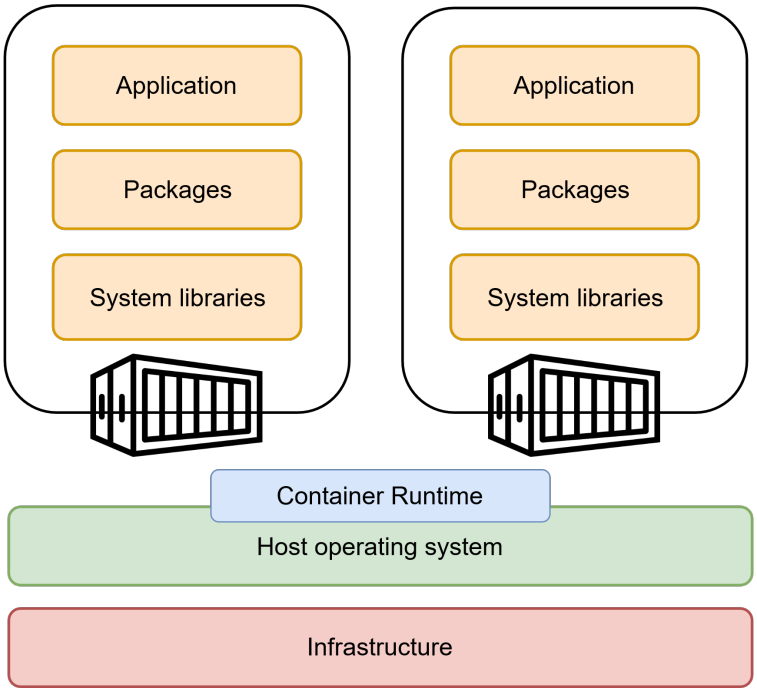
2.2 Adopter les technologies cloud

Suite à cette évolution de l'écosystème des big data, on observe un virage notable ces dernières années dans l'industrie vers des architectures plus flexibles et faiblement couplées. L'avènement des technologies *cloud* a joué un rôle déterminant dans cette transition. Contrairement à l'époque où Hadoop dominait, la latence réseau est devenue une préoccupation bien moindre, rendant le modèle traditionnel de solutions de stockage et de calcul sur site et co-localisées moins pertinent. Concernant la nature des données à traiter, on observe une évolution que certains ont qualifiée de passage « du big data aux données flexibles ». Les infrastructures modernes doivent non seulement

être capables de traiter de grands volumes, mais aussi être adaptables sur de multiples dimensions. Elles doivent pouvoir prendre en charge diverses structures de données (allant des formats structurés et tabulaires aux formats non structurés comme le texte et les images), assurer la portabilité des données dans des environnements *multi-cloud* et *cloud* hybride, et prendre en charge une large gamme de calculs computationnels (des calculs parallèles aux modèles d'apprentissage profond nécessitant des GPU, ainsi que le déploiement et la gestion d'applications) [26]. Ces dernières années, deux technologies ont émergé comme des éléments fondamentaux pour atteindre cette flexibilité dans les environnements *cloud* : la conteneurisation et le stockage d'objets.

Dans un environnement *cloud*, l'ordinateur de l'utilisateur devient un simple point d'accès pour effectuer des calculs sur une infrastructure centrale. Cela permet à la fois un accès ubiquitaire et une scalabilité des services, car il est plus facile de mettre à l'échelle une infrastructure centrale — généralement de manière horizontale, c'est-à-dire en ajoutant davantage de serveurs. Cependant, ces infrastructures centralisées présentent deux limites bien identifiées qui doivent être prises en compte : la concurrence entre utilisateurs pour l'accès aux ressources physiques et la nécessité d'isoler correctement les applications déployées. Le choix de la conteneurisation est fondamental, car il répond à ces deux enjeux [27]. En créant des « bulles » spécifiques à chaque service, les conteneurs garantissent l'isolement des applications tout en restant légers, puisqu'ils partagent le système d'exploitation de support avec la machine hôte (voir Figure 2). Pour gérer plusieurs applications conteneurisées de manière systématique, les infrastructures conteneurisées s'appuient généralement sur un logiciel orchestrateur — le plus connu étant Kubernetes, un projet open source initialement développé par Google pour gérer ses nombreuses charges de travail conteneurisées en production [28]. Les orchestrateurs automatisent le processus de déploiement, de mise à l'échelle et de gestion des applications conteneurisées, coordonnant leur exécution sur différents serveurs. De manière intéressante, cette propriété permet de traiter de très grands volumes de données de manière distribuée : les conteneurs décomposent les opérations de traitement des données massives en une multitude de petites tâches, organisées par l'orchestrateur. Cela minimise les ressources requises tout en offrant une flexibilité supérieure aux architectures basées sur Hadoop [29].

Figure 2. – Architecture d'un environnement conteneurisé.



Note: Un conteneur est un regroupement logique de ressources permettant d'encapsuler une application (par exemple, du code R), les bibliothèques utilisées (par exemple, ggplot, dplyr) et les bibliothèques système (l'interpréteur R, d'autres bibliothèques dépendantes du système d'exploitation, etc.) dans un seul package. Les applications conteneurisées sont isolées les unes des autres grâce à la virtualisation, ce qui permet d'attribuer des ressources physiques spécifiques à chaque application tout en garantissant leur indépendance totale. Contrairement aux machines virtuelles, qui virtualisent également le système d'exploitation (OS), les conteneurs s'appuient sur une forme de virtualisation légère : le conteneur partage l'OS de l'infrastructure hôte via le runtime de conteneur (par exemple, Docker). En conséquence, les conteneurs sont beaucoup plus portables et peuvent être déployés et redistribués facilement.

L'autre choix fondamental dans une architecture de données concerne la nature du stockage de ces données. Dans l'écosystème *cloud*, le « stockage d'objets » est devenu la référence *de facto* [30][3]. Dans ce paradigme, les fichiers sont stockés sous forme d'"objets" composés de données, d'un identifiant et de métadonnées. Ce type de stockage est optimisé pour la scalabilité, car les objets ne sont pas limités en taille et la technologie sous-jacente permet un stockage rentable de fichiers (potentiellement très) volumineux. Le stockage d'objets joue également un rôle clé dans la construction d'une infrastructure découplée comme celle évoquée précédemment : les dépôts de données — appelés *« buckets »* — sont directement interrogeables via des requêtes HTTP standards grâce à une API REST normalisée. Dans un contexte où la latence réseau n'est plus le principal goulot d'étranglement, cela signifie que le stockage et le calcul n'ont pas besoin d'être sur les mêmes machines, ni même dans le même lieu. Ils peuvent ainsi évoluer indépendamment en

[3]Principalement grâce à l'implémentation « S3 » (Simple Storage Service) d'Amazon.

fonction des besoins spécifiques de l'organisation. Enfin, le stockage d'objets est un complément naturel aux architectures basées sur des environnements conteneurisés. Il fournit une couche de persistance — les conteneurs étant par construction sans état (*stateless*) — et une connectivité facile, sans compromettre la sécurité, voire en renforçant celle-ci par rapport à un système de stockage traditionnel [31].

2.3 Exploiter les technologies cloud pour accroître l'autonomie et favoriser la reproductibilité

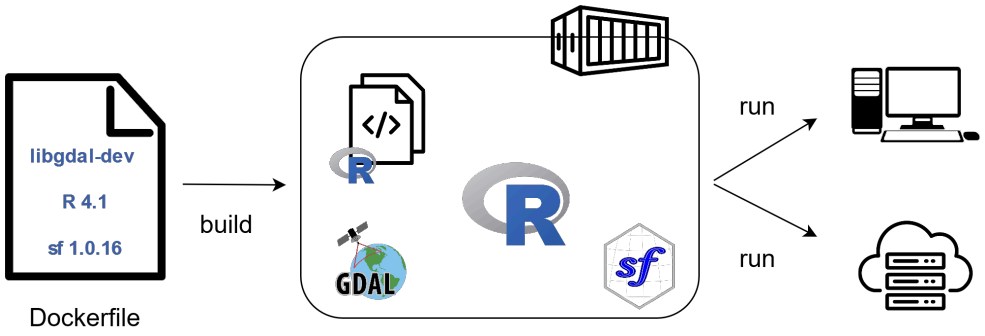
Comprendre comment les choix technologiques décrits dans la discussion technique ci-dessus sont pertinents dans le contexte des statistiques publiques nécessite un examen approfondi des pratiques professionnelles des statisticiens dans leur utilisation des environnements informatiques. À la fin des années 2000, alors que la micro-informatique était à son apogée, une grande partie des ressources techniques utilisées par les statisticiens de l'Insee étaient locales : le code et les logiciels de traitement étaient situés sur des ordinateurs personnels, tandis que les données étaient accessibles via un système de partage de fichiers. En raison de la scalabilité limitée des ordinateurs personnels, cette configuration restreignait considérablement la capacité des statisticiens à expérimenter avec des sources *big data* ou des méthodes statistiques intensives en calculs, et cela impliquait des risques de sécurité liés à la diffusion étendue des données au sein de l'organisation. Pour surmonter ces limitations, une transition a été opérée vers des infrastructures informatiques centralisées, regroupant toutes les ressources — et donc globalement beaucoup plus — sur des serveurs centraux. Ces infrastructures, mises à disposition des statisticiens via un environnement de bureau virtuel partagé pour faciliter leur utilisation, constituent encore la méthode dominante pour réaliser des calculs statistiques à l'Insee au moment de la rédaction de ces lignes.

À travers nos observations et nos discussions avec d'autres statisticiens, il est devenu évident que, bien que l'infrastructure informatique actuelle soutienne adéquatement les activités fondamentales de production statistique, elle restreint de manière notable la capacité des statisticiens à expérimenter librement et à innover. Le principal goulot d'étranglement dans cette organisation réside dans la dépendance des projets statistiques à la prise de décision centralisée en matière d'informatique, notamment en ce qui concerne l'allocation des ressources de calcul, l'accès au stockage partagé, l'utilisation de langages de programmation préconfigurés etc. En outre, ces dépendances conduisent souvent à un phénomène bien connu dans la communauté du développement logiciel, où les priorités des développeurs — itérer rapidement pour améliorer continuellement les fonctionnalités — entrent souvent en conflit avec l'objectif des équipes informatiques de garantir la sécurité et la stabilité des processus. À l'inverse, nous comprenons que les pratiques modernes en *datascience* reflètent une implication accrue des statisticiens dans le développement et l'orchestration informatique de leurs opérations de traitement de données, au-delà de la simple phase de conception ou de validation. Les nouvelles infrastructures de *datascience* doivent donc prendre en compte ce rôle élargi de leurs utilisateurs, en leur offrant plus d'autonomie que les infrastructures traditionnelles.

Nous soutenons que les technologies *cloud* sont une solution puissante pour offrir aux statisticiens une autonomie bien plus grande dans leur travail quotidien, favorisant ainsi une culture de l'innovation. Grâce au stockage d'objets, les utilisateurs obtiennent un contrôle direct sur la couche

de stockage, leur permettant d'expérimenter avec des sources de données diverses sans être limités par les espaces de stockage souvent restreints et alloués par les départements informatiques. La conteneurisation permet aux utilisateurs de personnaliser leurs environnements de travail selon leurs besoins spécifiques — qu'il s'agisse de langages de programmation, de bibliothèques système ou de versions de packages — tout en leur offrant la flexibilité nécessaire pour adapter leurs applications à la puissance de calcul et aux capacités de stockage requises. Par construction, les conteneurs favorisent également le développement d'applications portables, permettant des transitions plus fluides entre les environnements (développement, test, pré-production, production), en garantissant que les applications peuvent être exécutées sans difficulté, évitant ainsi les problèmes liés aux incohérences d'environnement. Enfin, avec des outils d'orchestration tels que Kubernetes, les statisticiens peuvent déployer plus facilement des applications et des API, tout en automatisant l'ensemble du processus de construction. Cette capacité s'aligne avec l'approche DevOps, qui préconise la création de preuves de concept de manière itérative, plutôt que de chercher à développer la solution optimale (mais chronophage) pour un objectif préalablement défini [32].

Figure 3. – Par construction, les conteneurs favorisent la reproductibilité et la portabilité.



Note: Dans un environnement conteneurisé, les applications sont créées à partir de spécifications sous forme de scripts — un paradigme connu sous le nom d'*"infrastructure as code"*. Dans un fichier texte, conventionnellement nommé « Dockerfile », les *data scientists* peuvent spécifier l'environnement de travail de leur application : le code de l'application, les logiciels à inclure (par exemple, R), les packages utilisés pour leurs opérations de traitement (par exemple, le package R pour le calcul géospatial *sf*), ainsi que les bibliothèques système dépendant de l'OS appelées par ces packages (par exemple, GDAL, la bibliothèque qui permet de lire et de traiter les formats d'images géospatiales utilisée par la plupart des packages traitant des données géospatiales). Un point essentiel est que les versions des logiciels et des packages utilisés pour développer l'application peuvent être précisément spécifiées, ce qui garantit la reproductibilité des calculs effectués. Une étape de construction génère ensuite une image associée au Dockerfile, c'est-à-dire une forme empaquetée et compressée de l'environnement de travail de l'application. Les images créées de cette manière sont portables : elles peuvent être facilement distribuées — généralement via un registre de conteneurs — et exécutées de manière reproductible sur n'importe quelle infrastructure disposant d'un runtime de conteneur.

Outre la scalabilité et l'autonomie, ces choix architecturaux favorisent également la reproductibilité des calculs statistiques. Le concept de reproductibilité — à savoir la capacité de reproduire le résultat d'une expérience en appliquant la même méthodologie aux mêmes données — est un critère fondamental de validité scientifique [33]. Il est également très pertinent dans le domaine des statistiques publiques, car il constitue une base pour la transparence, essentielle pour établir et maintenir la confiance du public [34]. Favoriser la reproductibilité dans la production statistique implique de concevoir des solutions de traitement capables de produire des statistiques reproductibles, tout en étant partageables entre pairs [35]. Les infrastructures informatiques traditionnelles — qu'il s'agisse d'un ordinateur personnel ou d'une infrastructure partagée avec un accès à distance — sont insuffisantes à cet égard. Construire un projet ou calculer un simple indicateur statistique dans ces environnements implique généralement une série d'étapes manuelles (installation des bibliothèques système, des binaires du langage de programmation, des packages du projet, gestion des versions conflictuelles, etc.) qui ne peuvent pas être pleinement reproduites d'un projet à l'autre. En comparaison, les conteneurs sont reproductibles par définition, car leur processus de construction implique de définir précisément toutes les ressources nécessaires comme un ensemble d'opérations standardisées, allant de la « machine nue » à l'application en cours d'exécution [36]. De plus, ces environnements reproductibles peuvent être facilement partagés avec des pairs, car ils peuvent être publiés sur des registres ouverts (par exemple, un registre de conteneurs comme DockerHub) avec le code source de l'application (par exemple, sur une forge logicielle publique comme GitHub ou GitLab). Cette approche améliore considérablement la réutilisation des projets de code, favorisant un modèle de développement et d'innovation basé sur la collaboration communautaire.

3 Onyxia : un projet open source pour construire des plateformes de data science sur des technologies cloud

Cette section examine comment Onyxia, un projet open source initié par l'Insee, démocratise l'accès aux technologies *cloud* pour les statisticiens en fournissant des environnements modernes de *data science* favorisant l'autonomie. Nous analysons comment cette initiative s'inscrit dans l'objectif général de création des « connaissances communes » en promouvant et en développant des logiciels facilement réutilisables dans le domaine des statistiques publiques et ailleurs.

3.1 Rendre les technologies cloud accessibles aux statisticiens

Notre veille technologique et notre revue de la littérature ont mis en évidence les technologies *cloud*, en particulier la conteneurisation et le stockage d'objets, comme des éléments clés pour construire une plateforme de *data science* à la fois scalable et flexible. En nous appuyant sur ces enseignements, nous avons mis en place notre premier cluster Kubernetes dans les locaux de l'Insee en 2020, en l'intégrant avec MinIO, un système de stockage d'objets *open source* conçu pour fonctionner de manière fluide avec Kubernetes. Cependant, nos premières expérimentations ont révélé un obstacle majeur à l'adoption généralisée des technologies *cloud* : la complexité de leur intégration. C'est une considération importante lorsqu'il s'agit de construire des architectures de données qui privilégient la modularité — une caractéristique essentielle pour atteindre la flexibi-

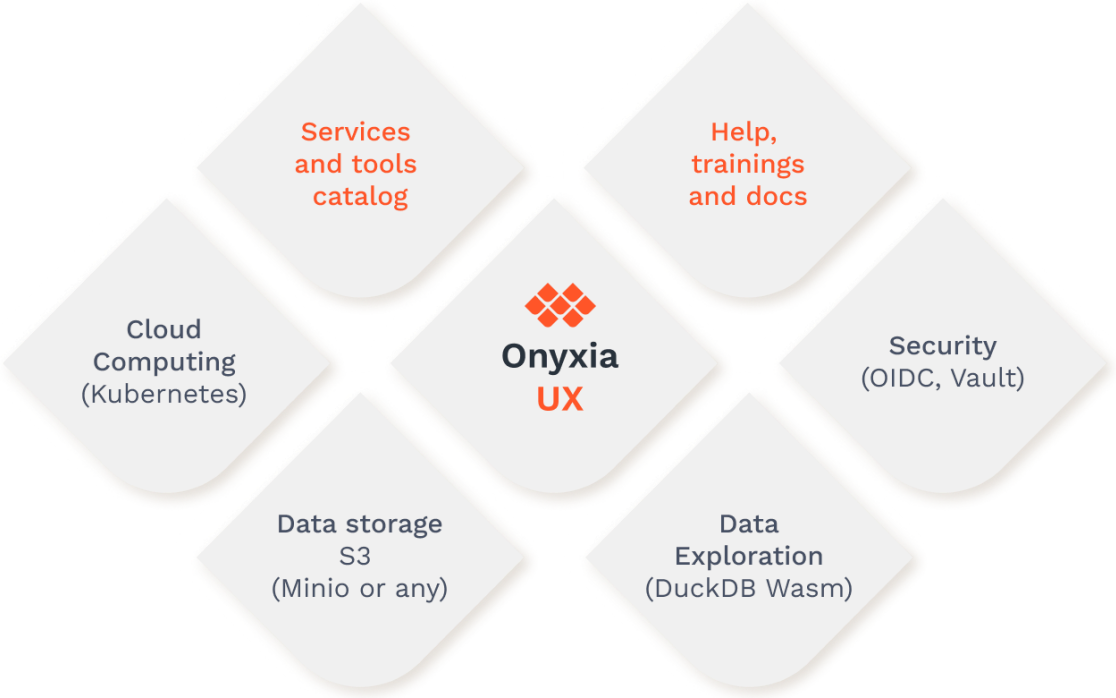
lité que nous visons[4]. Toutefois, la modularité des composants architecturaux implique également que toute application de données lancée sur le cluster doit être configurée pour communiquer avec tous les autres composants. Par exemple, dans un environnement *big data*, la configuration de Spark pour fonctionner sur Kubernetes tout en interagissant avec des ensembles de données stockés dans MinIO nécessite de nombreuses et complexes configurations (définition des points d'entrée, des jetons d'accès, etc.), une compétence qui dépasse généralement l'expertise des statisticiens.

Par exemple, grâce à la compatibilité de MinIO avec l'API S3 d'Amazon, la source de stockage pourrait facilement être remplacée par une solution gérée par un autre fournisseur de *cloud* public, sans nécessiter de modifications substantielles.

Cette idée est véritablement le fondement du projet Onyxia : choisir des technologies qui favorisent l'autonomie ne remplira pas cet objectif si leur complexité constitue une barrière à une adoption généralisée au sein de l'organisation. Ces dernières années, les statisticiens de l'Insee ont déjà dû s'adapter à un environnement en évolution en ce qui concerne leurs outils quotidiens : passer de logiciels propriétaires (SAS®) à des outils open source (R, Python), s'approprier des technologies qui améliorent la reproductibilité (contrôle de version avec Git), consommer et développer des API, etc. Ces changements, rendant leur travail de plus en plus semblable à celui de développeurs logiciels, impliquent déjà des efforts considérables en termes de formation et des modifications des pratiques de travail quotidiennes. Dans ce contexte, l'adoption des technologies *cloud* dépend totalement de leur accessibilité immédiate.

[4]Un exemple révélateur de l'importance de construire une architecture modulaire est la capacité de basculer entre différentes sources de stockage (on-premise, fournisseur de *cloud* public, etc.). La solution de stockage que nous avons choisie, MinIO, est compatible avec l'API S3 d'Amazon, qui est devenue un standard *de facto* dans l'écosystème *cloud* grâce au succès de la solution de stockage S3 d'Amazon AWS. Ainsi, les organisations qui choisissent d'utiliser Onyxia ne sont pas liées à une solution de stockage spécifique : elles peuvent opter pour n'importe quelle solution conforme aux standards définis par l'API S3.

Figure 4. – Onyxia est le lien technique entre les composants modulaires du cloud.



Pour combler cet écart, nous avons développé Onyxia, une application qui agit essentiellement comme une interface entre les composants modulaires qui composent l'architecture (voir Figure 4). Le point d'entrée principal pour l'utilisateur est une application web ergonomique[5] qui lui permet de lancer des services à partir d'un catalogue de *data science* (voir Chapitre 3.3) sous forme de conteneurs exécutés sur le cluster Kubernetes sous-jacent. Le lien entre l'interface utilisateur (UI) et Kubernetes est assurée par une API[6], qui transforme essentiellement la demande d'application de l'utilisateur en un ensemble de manifestes nécessaires pour déployer des ressources Kubernetes. Pour une application donnée, ces ressources sont regroupées sous la forme de *charts* Helm, une méthode populaire pour empaqueter des applications potentiellement complexes sur Kubernetes [37]. Bien que les utilisateurs puissent configurer un service pour l'adapter à leurs besoins, la plupart du temps, ils se contentent de lancer un service prêt à l'emploi avec des paramètres par défaut et commencent à développer immédiatement. Ce point illustre parfaitement la valeur ajoutée d'Onyxia pour faciliter l'adoption des technologies cloud. En injectant automatiquement les informations d'authentification et de configuration dans les conteneurs lors de leur initialisation, nous veillons à ce que les utilisateurs puissent lancer et gérer des services de *data science* dans lesquels ils interagissent sans difficulté avec les données de leur *bucket* sur MinIO, leurs informations sensibles (jetons, mots de passe) dans un outil de gestion des secrets tel que Vault, etc. Cette injection automatique, associée à la pré-configuration des environnements de

[5]https://github.com/InseeFrLab/onyxia-ui
[6]https://github.com/InseeFrLab/onyxia-api

data science dans les catalogues d'images[7] et de *charts* Helm[8] d'Onyxia, permet aux utilisateurs d'exécuter des scripts potentiellement complexes — comme des calculs distribués avec Spark sur Kubernetes à l'aide de données stockées sur S3, ou l'entraînement de modèles d'apprentissage profond utilisant un GPU — sans se heurter aux difficultés techniques liées à la configuration.

3.2 Des choix architecturaux visant à favoriser l'autonomie des statisticiens

Le projet Onyxia repose sur quelques principes structurants, avec un thème central : favoriser l'autonomie, à la fois au niveau organisationnel et individuel. Tout d'abord, au niveau de l'organisation, en évitant l'enfermement propriétaire. Pour obtenir un avantage concurrentiel, de nombreux fournisseurs de *cloud* commerciaux développent des applications et protocoles spécifiques que les clients doivent utiliser pour accéder aux ressources *cloud*, mais qui ne sont pas interopérables, compliquant considérablement les migrations potentielles vers une autre plateforme *cloud* [38]. Sachant cela, une tendance émerge vers l'adoption de stratégies neutres vis-à-vis des *clouds* [39] afin de réduire la dépendance à des solutions spécifiques d'un seul fournisseur. En revanche, l'utilisation d'Onyxia n'est intrinsèquement pas restrictive : lorsqu'une organisation choisit de l'utiliser, elle choisit les technologies sous-jacentes — la conteneurisation et le stockage d'objets — mais pas la solution en elle-même. La plateforme peut être déployée sur n'importe quel cluster Kubernetes, qu'il soit *on-premise* ou sur des *clouds* commerciaux. De même, Onyxia a été conçue pour être utilisée avec MinIO, car il s'agit d'une solution de stockage d'objets *open source*, mais il est également possible de l'utiliser avec les solutions de stockage d'objets proposées par divers fournisseurs de *cloud* (AWS, GCP, etc.).

Onyxia favorise également l'autonomie au niveau des utilisateurs. Les logiciels propriétaires qui ont été intensivement utilisés dans les statistiques officielles — comme SAS ou STATA — induisent également un phénomène d'enfermement propriétaire. Les coûts des licences sont élevés et peuvent évoluer rapidement, et les utilisateurs se retrouvent dépendants de certaines méthodes de calcul, empêchant une montée en compétences progressive. Au contraire, Onyxia aspire à être amovible ; nous souhaitons améliorer la familiarité et le confort des utilisateurs avec les technologies *cloud* sous-jacentes plutôt que de devenir un élément permanent travail quotidien. Un exemple illustratif de cette philosophie est l'approche de la plateforme concernant les actions des utilisateurs : pour les tâches effectuées via l'interface utilisateur, comme le lancement d'un service ou la gestion des données, nous fournissons aux utilisateurs les commandes terminal équivalentes, promouvant ainsi une compréhension plus approfondie de ce qui se passe réellement lors du déclenchement d'une action. De plus, tous les services proposés via le catalogue d'Onyxia sont *open source*.

[7] https://github.com/InseeFrLab/images-datascience
[8] https://github.com/InseeFrLab/helm-charts-interactive-services

Figure 5. – Lancer un service via l'interface web d'Onyxia.

Note: Les services du catalogue d'Onyxia peuvent être utilisés tels quels ou configurés par les utilisateurs pour répondre à leurs besoins spécifiques. Afin de limiter la dépendance des utilisateurs vis-à-vis d'Onyxia, chaque action effectuée par l'utilisateur via l'interface utilisateur est accompagnée de la commande exacte exécutée sur le cluster Kubernetes.

Naturellement, la manière dont Onyxia rend les statisticiens plus autonomes dans leur travail dépend de leurs besoins et de leur familiarité avec les compétences informatiques. Les statisticiens qui souhaitent simplement accéder à des ressources de calcul importantes pour expérimenter avec de nouvelles sources de données ou méthodes statistiques pourront, en quelques clics, accéder à des environnements de *data science* préconfigurés et faciles à utiliser, leur permettant de commencer à expérimenter immédiatement. Cependant, de nombreux utilisateurs souhaitent aller plus loin et développer de véritables prototypes d'applications de production pour leurs projets : configurer des scripts d'initialisation pour adapter les environnements à leurs besoins, déployer une application interactive offrant des visualisations de données aux utilisateurs de leur choix, ou encore déployer d'autres services que ceux disponibles dans nos catalogues. Pour permettre à ces utilisateurs avancés de continuer à repousser les limites de l'innovation, Onyxia leur donne accès au cluster Kubernetes sous-jacent. Cela signifie que les utilisateurs peuvent ouvrir librement un terminal sur un service interactif et interagir avec le cluster — dans les limites de leur *namespace* — afin d'appliquer des ressources personnalisées et de déployer des applications ou services personnalisés.

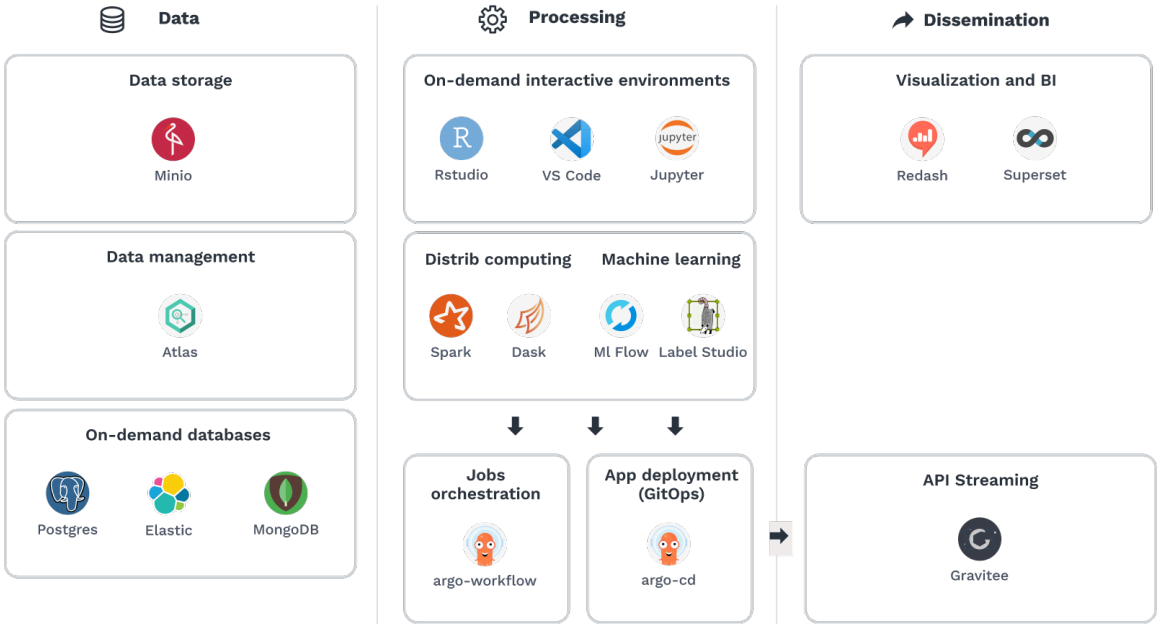
Au-delà de l'autonomie et de la scalabilité, les choix architecturaux d'Onyxia favorisent également la reproductibilité des calculs statistiques. Dans le paradigme des conteneurs, l'utilisateur doit apprendre à gérer des ressources qui sont par nature éphémères, puisqu'elles n'existent qu'au moment de leur mobilisation effective. Cela encourage l'adoption de bonnes pratiques de développement, notamment la séparation du code — hébergé sur une forge interne ou *open source* telle que GitLab ou GitHub —, des données — stockées sur une solution de stockage spécifique, comme MinIO —, et de l'environnement de calcul. Bien que cela impose un coût d'entré non négligeable aux utilisateurs, cela les aide également à concevoir leurs projets sous forme de *pipelines*, c'est-à-dire une série d'étapes séquentielles avec des données en entrées et productions finales bien définies (semblables à un graphe orienté acyclique, ou DAG). Les projets développés de cette manière sont généralement plus reproductibles et transposables — ils peuvent fonctionner sans

problème sur différents environnements de calcul — et sont ainsi plus facilement partageables avec leurs pairs.

3.3 Un catalogue exhaustif de services pour couvrir l'ensemble du cycle de vie des projets de data science

Lors du développement de la plateforme Onyxia, notre intention était de fournir aux statisticiens un environnement complet conçu pour accompagner le développement de bout en bout des projets de *data science*. Comme illustré dans Figure 6, la plateforme propose une vaste gamme de services couvrant l'ensemble du cycle de vie d'un projet de *data science*.

Figure 6. – Le catalogue d'Onyxia vise à couvrir l'ensemble du cycle de vie des projets de data science



L'utilisation principale de la plateforme est le déploiement d'environnements de développement interactifs (IDE), tels que RStudio, Jupyter ou VSCode. Ces IDE sont équipés des dernières versions des principaux langages de programmation *open source* couramment utilisés par les statisticiens publics (R, Python, Julia), ainsi que d'une vaste collection de librairies fréquemment employées en *data science* pour chaque langage. Afin de garantir que les services restent à jour et cohérents entre eux, nous maintenons nos propres images Docker sous-jacentes et les mettons à jour chaque semaine. Ces images sont entièrement *open source*[9] et peuvent donc être réutilisée en dehors d'Onyxia.

Comme discuté dans les sections précédentes, la couche de persistance de ces environnements interactifs est principalement assurée par MinIO, la solution de stockage d'objets par défaut d'Onyxia. Étant basé sur une API REST standardisée, les fichiers peuvent être facilement interrogés depuis R ou Python à l'aide de librairies de haut niveau. Cela représente en soi une étape im-

[9]https://github.com/InseeFrLab/images-datascience

portante pour garantir la reproductibilité : les données ne sont pas sauvegardés localement, puis spécifiés via des chemins propres à une infrastructure ou un système de fichiers particulier. Au contraire, les fichiers sont spécifiés sous forme de requêtes HTTP, rendant la structure globale des projets bien plus extensible. D'après notre expérience, le paradigme du stockage d'objets répond très bien aux besoins de la plupart des projets statistiques que nous accompagnons. Cependant, des services de bases de données supplémentaires, tels que PostgreSQL et MongoDB, sont disponibles pour les applications ayant des besoins spécifiques, notamment celles nécessitant des capacités de traitement transactionnel en ligne (OLTP) ou un stockage orienté documents.

Comme Onyxia a été développée pour permettre l'expérimentation avec des sources de données volumineuses et des méthodes d'apprentissage automatique, nous proposons également des services optimisés pour passé à l'échelle facilement. Par exemple, des frameworks comme Spark et Trino, qui permettent d'effectuer des calculs distribués au sein d'un cluster Kubernetes. Ces services sont préconfigurés pour s'intégrer parfaitement avec le stockage S3, facilitant ainsi la création de *pipelines* de données intégrés et efficaces.

Au-delà de la simple expérimentation, notre objectif est de permettre aux statisticiens de passer des phases de test à des projets de qualité proche de celle requise en production afin de réduire le coût lors de la transmission d'un projet d'une équipe de production vers une équipe informatique. Conformément aux principes de l'approche DevOps, cela implique de faciliter le déploiement de prototypes et leur amélioration continue au fil du temps. À cette fin, nous proposons un ensemble de services *open source* visant à automatiser et industrialiser le processus de déploiement d'applications (ArgoCD, Argo-Workflows, MLflow). Pour les projets exploitant des modèles d'apprentissage automatique, les statisticiens peuvent exposer leurs modèles via des API, les déployer en utilisant les outils susmentionnés et gérer leur cycle de vie grâce à un gestionnaire d'API (par exemple, Gravitee). La Section 4 illustrera comment ces outils, en particulier MLflow, ont joué un rôle central dans la mise en production de modèles d'apprentissage automatique à l'Insee, en lien avec les principes de MLOps.

Dans la Chapitre 3.2, nous avons souligné qu'un des principes fondamentaux de conception d'Onyxia était d'éviter l'enfermement propriétaire. Dans cette optique, les organisations qui instancient Onyxia sont libres de personnaliser les catalogues pour répondre à leurs besoins spécifiques, ou même de créer leurs propres catalogues indépendamment des offres par défaut d'Onyxia. Cette flexibilité garantit aux organisations de ne pas être limité à une solution ou à un fournisseur unique, et qu'elles peuvent adapter la plateforme à l'évolution de leurs besoins.

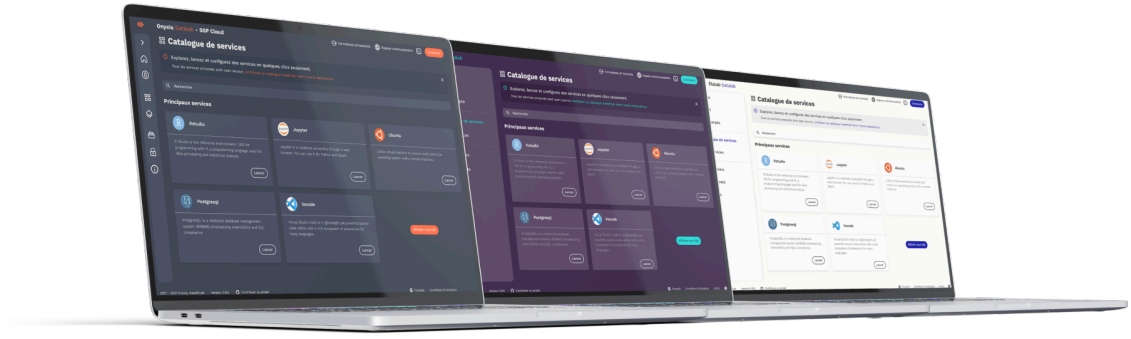
3.4 Construire des biens communs : un projet open source et une plateforme d'innovation ouverte

En tant qu'initiative entièrement *open source*, le projet Onyxia vise à construire des « connaissances communes » en promouvant et en développant des logiciels facilement réutilisables dans les statistiques publiques et ailleurs [40]. Cela concerne, tout d'abord, les composants sur lesquels repose Onyxia : à la fois ses briques technologiques (Kubernetes, MinIO, Vault) et l'ensemble des services du catalogue, qui sont *open source*. Plus important encore, tout le code du projet est dis-

ponible publiquement sur GitHub[10]. Associée à une documentation détaillée[11], cette transparence facilite grandement la possibilité pour d'autres organisations de créer des instances de plateformes de *data science* basées sur le logiciel Onyxia et de les adapter à leurs besoins spécifiques (voir Figure 7). Cela a permis au projet d'attirer une communauté croissante de contributeurs issus des statistiques publiques (Statistique Norvège), des ONG (Mercator Ocean[12]), des centres de recherche et même de l'industrie, favorisant ainsi une transition progressive vers une gouvernance plus décentralisée du projet.

Dans les prochaines années, l'implication des INS (Instituts Nationaux de Statistique) du système statistique européen devrait augmenter, puisque le SSPCloud a été choisie comme plateforme *data science* de référence dans le cadre du projet AIML4OS[13].

Figure 7. – Un projet, de multiples instances : l'interface web est adaptable à l'identité graphique de chaque organisation



Une autre manière majeure de construire des communs est le développement et le maintien d'une instance de démonstration du projet Onyxia, le SSP Cloud [41]. Cette plateforme, équipée de ressources de calcul extensives et évolutives[14], est conçue comme un bac à sable pour expérimenter avec les technologies cloud et les nouvelles méthodes de science des données. Le catalogue complet des services d'Onyxia est disponible sur cette plateforme, permettant aux utilisateurs motivés d'aller au-delà de la simple expérimentation en produisant des « preuves de concept » avec une autonomie totale concernant la configuration et l'orchestration de leurs services.

Au-delà de ses capacités techniques, le SSP Cloud incarne les principes de l'innovation ouverte [42]. Déployé sur internet[15], il est accessible non seulement aux employés de l'Insee, mais également, plus largement, aux agences gouvernementales françaises, aux universités françaises et aux autres INS européens. Il est dédié à l'expérimentation des méthodes de *data science* en utilisant des données ouvertes. Ainsi, les projets menés sur cette plateforme mettent en lumière l'abondance

[10]https://github.com/InseeFrLab/onyxia

[11]https://docs.onyxia.sh/

[12]Lien vers l'instance Onyxia de Mercator Ocean : https://datalab.dive.edito.eu/

[13]Plus d'informations à propos de ce projet disponibles à https://cros.ec.europa.eu/dashboard/aiml4os

[14]Sur le plan matériel, le SSP Cloud est constitué d'un cluster Kubernetes d'environ 20 serveurs, pour une capacité totale de 10 To de RAM, 1100 processeurs, 34 GPU et 150 To de stockage.

[15]https://datalab.sspcloud.fr/

croissante des jeux de données publiés en libre accès par les organisations publiques ou privés, faisant écho à la loi pour une République numérique de 2016. La nature fondamentalement collaborative du SSP Cloud s'est avérée particulièrement bénéfique pour l'organisation d'événements innovants, tels que des hackathons — tant au niveau national qu'international — et dans le domaine académique. Il est devenu une ressource intégrale pour plusieurs universités et Grandes Écoles en France, favorisant l'utilisation d'environnements *cloud* et reproductibles, tout en évitant l'effet d'enfermement propriétaire dû à une dépendance excessive des institutions éducatives envers des solutions *cloud* propriétaires. En conséquence, la plateforme est désormais largement utilisée dans le service statistique publique français et ailleurs, avec environ 1000 utilisateurs uniques par mois début 2025. Ces utilisateurs forment une communauté dynamique grâce à un canal de discussion centralisé[16] ; ils contribuent à améliorer l'expérience utilisateur en signalant des bugs, en proposant de nouvelles fonctionnalités et en participant ainsi directement au projet.

4 Cas d'usage : adopter les pratiques MLOps pour améliorer la codification de l'APE

Ce chapitre vise à illustrer comment l'Insee a réussi à déployer son premier modèle de machine learning (ML) en production. Il propose une description détaillée de l'approche MLOps à laquelle ce projet s'est efforcé d'adhérer, en mettant l'accent sur les différentes technologies employées. En particulier, nous soulignons le rôle crucial des technologies *cloud* qui ont permis la construction du projet de manière itérative, ainsi que la manière dont Onyxia a grandement facilité cette construction en fournissant des environnements de développement flexibles et des outils pour entraîner, déployer et surveiller les modèles des modèles d'apprentissage automatique. De plus, la convergence entamée des environnement de self (LS^3), de développement (KubeDev) et de production (KubeProd) constitue une réelle avancer pour faciliter la mise en production d'autres modèles d'apprentissage automatique (un modèle de codification de la PCS a également été déployé récemment). Le projet présenté est totalement disponible en open source[17] et reste en cours de développement actif.

4.1 Fluidifier la codification de l'APE à l'aide de méthodes d'apprentissage automatique

4.1.1 Motivation

Les tâches de codification sont des opérations bien connues des instituts statistiques, et peuvent parfois être complexes en raison de la taille des nomenclatures. À l'Insee, un outil sophistiqué appelé Sicore a été développé dans les années 1990 pour effectuer diverses tâches de classification [43]. Cet outil repose sur un ensemble de règles déterministes permettant d'identifier les codes corrects à partir d'un libellé textuel en se basant sur un fichier de référence comprenant un certain nombre d'exemples. Chaque libellé d'entrée est soumis à ces règles et, lorsqu'un code correct est

[16] Lien vers les canaux de discussion https://www.tchap.gouv.fr/#/room/#SSPCloudXDpAw6v:agent.finances.tchap.gouv.fr et https://join.slack.com/t/3innovation/shared_invite/zt-19tht9hvr-bZGMdW8AV_wvd5kz3wRSMw

[17] https://github.com/orgs/InseeFrLab/teams/codification-ape/repositories

reconnu, il est attribué au libellé. En revanche, si le libellé n'est pas reconnu, il doit être classer manuellement par un agent de l'Insee.

Deux raisons principales ont motivé l'expérimentation de nouvelles méthodes de codification.

Premièrement, un changement interne est survenu avec la refonte du répertoire statistique des entreprises en France (Sirene), qui liste toutes les entreprises et leur attribue un identifiant unique utilisé par les administrations publiques, le numéro Siren. Les principaux objectifs de cette refonte étaient d'améliorer la gestion quotidienne du répertoire pour les agents de l'Insee et de réduire les délais d'attente pour les entreprises. Par ailleurs, au niveau national, le gouvernement a lancé, dans le cadre de la loi PACTE (n° 2019-486 du 22 mai 2019), un guichet unique pour les formalités des entreprises, offrant aux chefs d'entreprises plus de flexibilité dans la description de leurs activités principales les rendant ainsi plus verbeux que précédemment. Les tests initiaux ont révélé que Sicore n'était plus adapté pour effectuer la codification APE, puisque seulement 30% des liasses d'entreprises étaient automatiquement codées, et donc 70% devait être codés manuellement par des gestionnaires. Les équipes en charge du répertoire Sirene, déjà confrontées à des charges de travail importantes et à de fortes contraintes opérationnelles, ne pouvaient pas voir leur charge augmentée par une re-codification manuelle, une tâche à la fois chronophage et peu stimulante. Ainsi, en mai 2022, la décision a été prise d'expérimenter de nouvelles méthodes pour effectuer cette tâche de codification, avec pour objectif de les utiliser en production dès le 1er janvier 2023, date de lancement du nouveau répertoire Sirene.

Trois parties prenantes étaient donc impliquées dans ce projet : l'équipe métier (division RIAS[18]), responsable de la gestion du répertoire statistique des entreprises ; l'équipe informatique, en charge du développement des applications liés au fonctionnement du répertoire ; et l'équipe d'innovation (l'unité SSP Lab), responsable de la mise en œuvre du nouvel outil de codification.

4.1.2 La tâche de codification

Le projet présenté consiste en un problème classique de classification dans le cadre de traitement de langage naturel. À partir d'une description textuelle de l'activité d'une entreprise, l'objectif est de prédire la classe associée dans la nomenclature APE. Cette classification présente la particularité d'être hiérarchique et comporte cinq niveaux différents[19] : section, division, groupe, catégorie et sous-catégorie. Au total, la nomenclature comprend 732 sous-classes, ce qui correspond au niveau le plus fin de la nomenclature et pour lequel on souhaite réaliser notre codification. La table Table 1 fournit un exemple de cette structure hiérarchique.

[18]Répertoire Interadministratif Sirene

[19]En réalité, il existe cinq niveaux en France, mais seulement quatre au niveau européen.

Table 1. – Nomenclature APE

Niveau	NAF	Libellé	Taille
Section	H	Transports et entreposage	21
Division	52	Entreposage et services auxiliaires des transports	88
Groupe	522	Services auxiliaires des transports	272
Catégorie	5224	Manutention	615
Sous-catégorie	**5224A**	**Manutention portuaire**	**732**

Avec la mise en place du guichet unique, les chefs d'entreprise décrivent désormais leur activité dans un champ de texte libre. Par conséquent, les nouveaux libellés diffèrent fortement des libellés harmonisés précédemment reçus. Il a donc été décidé de travailler avec des modèles d'apprentissage automatique, reconnus pour leur efficacité sur les tâches de classification supervisée de texte [44]. Cela représente un changement de paradigme significatif pour l'Insee, puisque le *machine learning* n'est traditionnellement pas utilisé dans la production des statistiques officielles. De plus, la perspective de mettre le nouveau modèle en production a été envisagée dès le début du projet, orientant de nombreux choix méthodologiques et techniques. Ainsi, plusieurs décisions stratégiques ont dû être rapidement prises, notamment en ce qui concerne la méthodologie, le choix d'un environnement de développement cohérent avec l'environnement de production cible, et l'adoption de méthodes de travail collaboratif.

4.1.3 Méthodologie

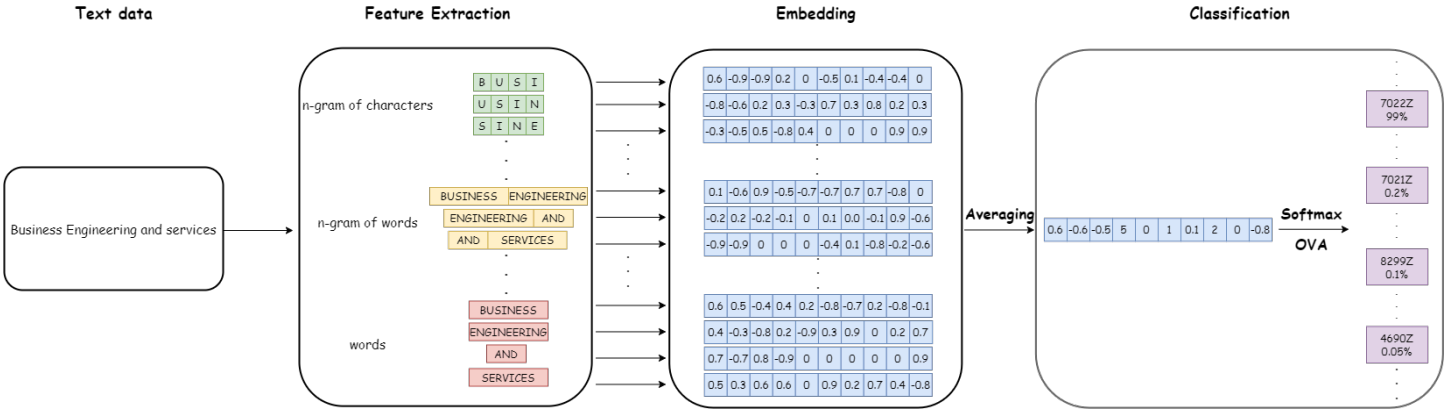
La classification textuelle à partir des champs de texte libre fournis par les chefs d'entreprise est une tâche complexe : les descriptions d'activité sont relativement courtes et contiennent donc peu d'information statistique, peuvent inclure des fautes d'orthographe et nécessitent souvent une expertise métier pour être correctement codées. Pour une telle tâche, les méthodes traditionnelles d'analyse textuelle, comme la vectorisation par comptage ou TF-IDF, sont souvent insuffisantes, tandis que les méthodes d'intégration basées sur des réseaux de neurones tendent à donner de meilleurs résultats [44]. Cependant, ces architectures nécessitent souvent des ressources de calcul importantes, et peuvent exiger du matériel spécifique, comme des GPUs, afin obtenir une latence acceptable lors de l'inférence. Ces contraintes nous ont, dans un premier temps, éloignés des modèles les plus performants, tels que les modèles Transformer, et orientés vers le modèle fastText [45], un réseau de neurone plus simple basé sur des plongements lexicaux. Le modèle fastText est extrêmement rapide à entraîner, et l'inférence ne nécessite pas de GPU pour obtenir un temps de latence faible. En outre, le modèle a donné d'excellents résultats pour notre cas d'usage, qui, compte tenu des contraintes de temps et de ressources humaines, étaient largement suffisants pour améliorer le processus existant. Enfin, l'architecture du modèle est relativement simple, ce qui facilite la communication et l'adoption au sein des différentes équipes de l'Insee.

Le modèle fastText repose sur une approche de sac de mots (*bag-of-words*) pour obtenir des plongements lexicaux et une couche de classification basée sur la régression logistique. L'approche sac de mots consiste à représenter un texte comme un ensemble de représentations vectorielles

de chacun des mots qui le composent. La spécificité du modèle fastText, par rapport à d'autres approches basées sur des plongements lexicaux, est que les plongements lexicaux ne sont pas seulement calculés sur les mots, mais aussi sur des *n-grams* de mots et de caractères, fournissant ainsi plus de contexte et réduisant les biais liés aux fautes d'orthographe. Ensuite, le plongement lexical d'une phrase est calculé comme une fonction des plongements lexicaux des mots (et n-grams de mots et de caractères), généralement une moyenne. Dans le cas de la classification textuelle supervisée, la matrice de plongement et les poids du *classifier* sont appris simultanément lors de l'entraînement par descente de gradient, en minimisant la fonction de perte d'entropie croisée.

Figure 8 présente la pipeline complete des opérations effectuées par fastText sur un exemple de texte en entrée.

Figure 8. – Aperçu simplifié du processus derrière les classifications fastText



4.2 Une approche orientée production et MLOps

Dès le début du projet, l'objectif était d'aller au-delà de la simple expérimentation et de mettre le modèle en production. Par ailleurs, ce projet pilote avait également pour but de servir de modèle pour les futurs projets de *machine learning* à l'Insee. Nous avons donc cherché à appliquer les meilleures pratiques de développement dès les premières étapes du projet : respect des standards de qualité de code de la communauté, utilisation de scripts pour le développement au lieu de notebooks, construction d'une structure modulaire semblable à un *package*, etc. Cependant, par rapport aux projets de développement traditionnels, les projets de *machine learning* présentent des caractéristiques spécifiques qui nécessitent l'application d'un ensemble de bonnes pratiques complémentaire, regroupées sous le nom de MLOps.

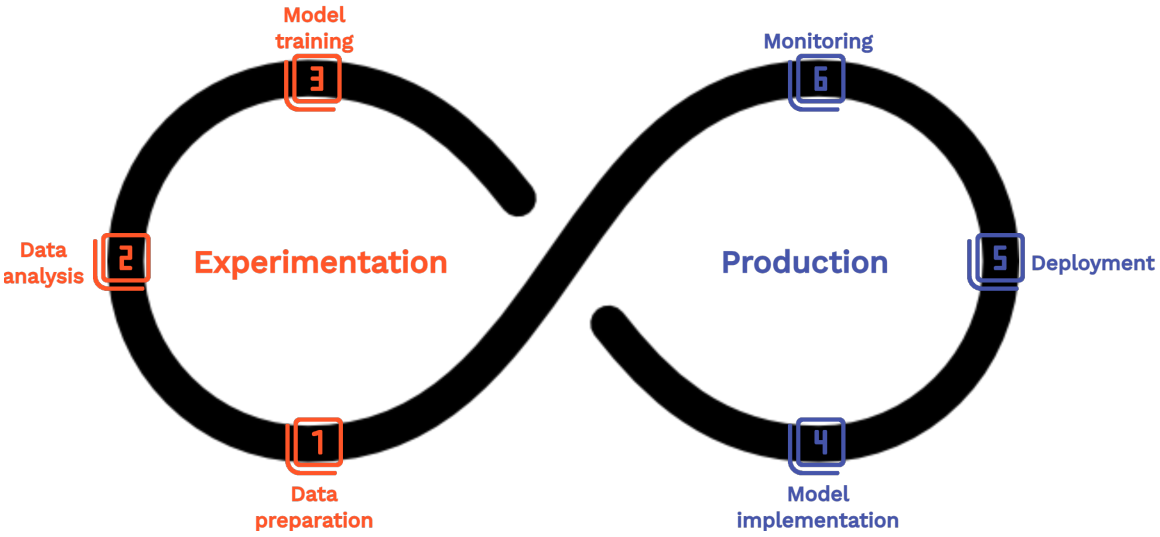
4.2.1 Du DevOps au MLOps

Le DevOps est un ensemble de pratiques conçu pour favoriser la collaboration entre les équipes de développement (*Dev*) et d'opérations (*Ops*). L'idée fondamentale est d'intégrer tout le cycle de vie d'un projet dans un continuum automatisé. Un outil important pour atteindre cette continuité sont les pipelines CI/CD. Avec l'intégration continue (*Continuous Integration* ou CI), chaque *commit* de nouveau code source déclenche un processus d'opérations standardisées, telles que la construction de l'application, son test et sa mise à disposition sous forme de version. Ensuite, le

déploiement continu (*Continuous Deployment* ou CD) consiste en des outils pour automatiser le déploiement du nouveau code et limiter les interventions manuelles, tout en garantissant une supervision appropriée pour assurer la stabilité et la sécurité des processus. Cette approche favorise un déploiement plus rapide et continu des modifications ou ajouts nécessaires de fonctionnalités. En outre, en encourageant la collaboration entre les équipes, le DevOps accélère également le cycle d'innovation, permettant aux équipes de résoudre les problèmes au fur et à mesure qu'ils surviennent et d'intégrer efficacement les retours tout au long du cycle de vie du projet.

L'approche MLOps peut être vue comme une extension du DevOps, développée pour relever les défis spécifiques liés à la gestion du cycle de vie des modèles de ML. Fondamentalement, DevOps et MLOps partagent le même objectif : construire des logiciels de manière plus automatisée et robuste. La principale différence réside dans le fait qu'avec le MLOps, le logiciel inclut également une composante de machine learning. Par conséquent, le cycle de vie du projet devient plus complexe. Le modèle de ML sous-jacent doit être réentraîné régulièrement afin d'éviter toute perte de performance au fil du temps. L'ingestion des données doit également être intégrée au processus, car de nouvelles données peuvent être utilisées pour améliorer les performances. Figure 9 présente les étapes d'un projet de ML en utilisant une représentation continue, comme cela se fait traditionnellement en DevOps. Cela illustre un principe fondamental du MLOps : la nécessité d'une amélioration continue, décrite plus en détail dans Chapitre 4.2.2.

Figure 9. – L'approche MLOps favorise une gestion continue du cycle de vie des projets de ML



4.2.2 Principes du MLOps

Le MLOps repose sur quelques principes fondamentaux qui sont essentiels pour construire des applications de *machine learning* évolutives et prêtes pour le passage en production. Ces principes visent à relever les défis spécifiques associés aux chaînes de production de *machine learning*.

Le principe le plus fondamental du MLOps est l'amélioration continue, reflétant la nature itérative des projets de ML. Lors de la phase d'expérimentation, le modèle est développé à partir d'un ensemble de données d'entraînement, qui diffère généralement des données de production à certains

égards. Une fois le modèle déployé en production, les nouvelles données sur lesquelles le modèle doit effectuer des prédictions peuvent révéler des informations sur ses performances et ses éventuelles lacunes. Ces informations nécessitent un retour à la phase d'expérimentation, où les data scientists ajustent ou redéfinissent leurs modèles pour corriger les problèmes découverts ou améliorer la précision. Ce principe souligne donc l'importance de construire une boucle de rétroaction permettant des améliorations continues tout au long du cycle de vie d'un modèle. L'automatisation, en particulier grâce à l'utilisation de pipelines CI/CD, joue un rôle crucial en rendant la transition entre les phases d'expérimentation et de production plus fluide. La surveillance (*monitoring*) est également une composante essentielle de ce processus : un modèle déployé en production doit être continuellement analysé pour détecter d'éventuelles dérives importantes susceptibles de réduire ses performances prédictives et nécessitant des ajustements supplémentaires, comme un ré-entraînement.

Un autre objectif majeur du MLOps est de promouvoir la reproductibilité, en garantissant que toute expérience de ML puisse être reproduite de manière fiable avec les mêmes résultats. Les outils de MLOps facilitent ainsi une sauvegarde détaillée des expériences de ML, incluant les étapes de prétraitement des données, les hyperparamètres des modèles utilisés et les algorithmes d'entraînement. Les données, modèles et codes sont versionnés, permettant aux équipes de revenir à des versions antérieures si une mise à jour ne donne pas les résultats escomptés. Enfin, ces outils aident à produire des spécifications détaillées de l'environnement informatique utilisé pour produire ces expériences — comme les versions des bibliothèques — et reposent souvent sur des conteneurs pour reproduire les mêmes conditions que celles dans lesquelles le modèle initial a été développé.

Enfin, le MLOps vise à favoriser le travail collaboratif. Les projets basés sur le ML impliquent généralement une gamme plus large de profils : équipes métier et équipes de *data science* d'un côté, développeurs et équipes de production informatique de l'autre. Comme le DevOps, le MLOps met donc l'accent sur la nécessité d'une culture collaborative et d'éviter le travail en silos. Les outils de MLOps incluent généralement des fonctionnalités collaboratives, telles que des stockages centralisés pour les modèles de ML ou les caractéristiques (*features*) de ML, qui facilitent le partage des composants entre les membres des équipes et limitent la redondance.

4.2.3 Implémentation avec MLflow

De nombreux outils ont été développés pour mettre en œuvre l'approche MLOps dans des projets concrets. Tous visent à appliquer, sous une forme ou une autre, les principes fondamentaux décrits précédemment. Dans ce projet, nous avons choisi de nous appuyer sur un outil open-source populaire nommé MLflow[20]. Ce choix ne reflète pas une supériorité inhérente de MLflow par rapport à d'autres outil, mais s'explique par un ensemble de bonnes propriétés associées à MLflow, qui en font une solution particulièrement pertinente pour notre cas d'usage. Tout d'abord, il couvre l'intégralité du cycle de vie des projets de ML, tandis que d'autres outils peuvent être plus spécialisés sur certaines parties seulement. Ensuite, il offre une grande interopérabilité grâce à une bonne interface avec les bibliothèques populaires de ML — telles que PyTorch, Scikit-learn, XGBoost,

[20] https://github.com/MLflow/MLflow

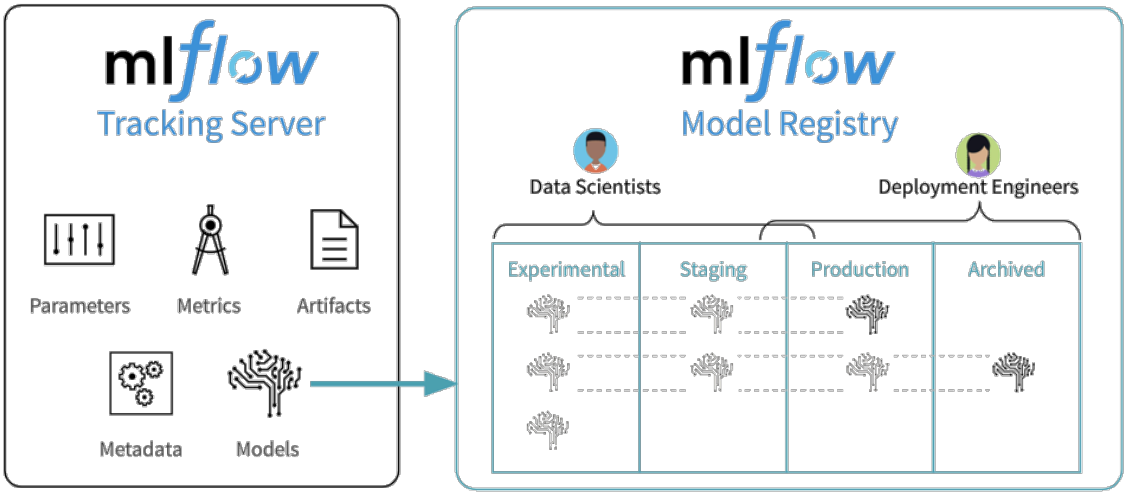
etc. — et prend en charge plusieurs langages de programmation — notamment Python, R et Java, couvrant ainsi le spectre des langages couramment utilisés à l'Insee. Enfin, MLflow s'est révélé très facile d'utilisation, encourageant ainsi son adoption par les membres du projet et facilitant la collaboration continue entre eux.

MLflow fournit un cadre cohérent pour opérationnaliser les principes du MLOps efficacement au sein des projets de ML. Les data scientists peuvent encapsuler leur travail dans des *MLflow Projects* qui regroupent le code ML et ses dépendances, garantissant que chaque projet soit reproductible et puisse être ré-exécuté de manière identique. Un projet s'appuie sur un *MLflow Model*, un format standardisé compatible avec la plupart des bibliothèques de ML et offrant une méthode normalisée pour déployer le modèle, par exemple via une API. Cette interopérabilité et cette standardisation sont essentielles pour soutenir l'amélioration continue du projet, puisque les modèles entraînés avec une multitude de packages peuvent être facilement comparés ou remplacés les uns par les autres sans casser le code existant.

À mesure que les expériences avec différents modèles progressent, le *Tracking Server* enregistre des informations détaillées sur chaque exécution — hyperparamètres, métriques, artefacts et données — ce qui favorise à la fois la reproductibilité et facilite la phase de sélection des modèles grâce à une interface utilisateur ergonomique. Une fois la phase d'expérimentation terminée, les modèles sélectionnés sont rajoutés dans le *Model Registry*, où ils sont versionnés et prêts pour le déploiement. Cet entrepôt sert de « magasin » centralisé pour les modèles, permettant aux différents membres ou équipes du projet de gérer collaborativement le cycle de vie du projet.

Figure 10 illustre les composants principaux de MLflow et la manière dont ils facilitent un flux de travail plus continu et collaboratif au sein d'un projet de ML.

Figure 10. – Composants principaux de MLflow. Source : Databricks.



4.3 Faciliter le développement itératif avec les technologies cloud

Bien que l'amélioration continue soit un principe fondamental du MLOps, elle est également très exigeante. En particulier, elle nécessite de concevoir et de construire un projet sous la forme d'une

pipeline intégrée, dont les différentes étapes sont principalement automatisées, de l'ingestion des données jusqu'à la surveillance du modèle en production. Dans ce contexte, le développement itératif est essentiel pour construire un produit minimum viable qui sera ensuite affiné et amélioré au fil du temps. Cette section illustre comment les technologies *cloud*, via le projet Onyxia, ont été déterminantes pour construire le projet sous forme de composants modulaires interconnectés, renforçant ainsi considérablement la capacité de raffinement continu au fil du temps.

4.3.1 Un environnement de développement flexible

Dans un projet de ML, la flexibilité de l'environnement de développement est essentielle. Premièrement, en raison de la diversité des tâches à accomplir : collecte des données, prétraitement, modélisation, évaluation, inférence, surveillance, etc. Deuxièmement, parce que le domaine du ML évolue rapidement, il est préférable de construire une application de ML sous forme d'un ensemble de composants modulaires afin de pouvoir mettre à jour certains éléments sans perturber l'ensemble de la pipeline. Comme discuté dans la Section 2.2, les technologies *cloud* permettent de créer des environnements de développement modulaires et évolutifs.

Cependant, comme également abordé dans la Section 3, l'accès à ces ressources ne suffit pas. Un projet de ML nécessite une grande variété d'outils pour se conformer aux principes du MLOps : stockage des données, environnements de développement interactifs pour expérimenter librement, outils d'automatisation, outils de surveillance, etc. Bien que ces outils puissent être installés sur un cluster Kubernetes, il est essentiel de les rendre disponibles aux data scientists de manière intégrée et préconfigurée pour faciliter leur adoption. Grâce à son catalogue de services et à l'injection automatique de configurations dans les services, Onyxia permet de construire des projets qui reposent sur plusieurs composants *cloud* capables de communiquer facilement entre eux.

La manière dont l'entraînement du modèle a été réalisé pour ce projet illustre bien la flexibilité offerte par Onyxia pendant la phase d'expérimentation. Tout le code utilisé pour l'entraînement est écrit en Python au sein d'un service VSCode. Grâce à l'injection automatique des identifiants personnels S3 dans chaque service au démarrage, les différents utilisateurs du projet peuvent interagir directement avec les données d'entraînement stockées dans un *bucket* S3 sur MinIO, la solution de stockage d'objets par défaut d'Onyxia. Toutes les expériences menées lors de la phase de sélection du modèle sont consignées dans une instance partagée de MLflow, qui enregistre les données sur une instance PostgreSQL automatiquement lancée sur Kubernetes, tandis que les artefacts (modèles entraînés et métadonnées associées) sont stockés sur MinIO.

Le modèle a été entraîné en utilisant une recherche exhaustive (*grid-search*) pour l'ajustement des hyperparamètres et évalué par validation croisée (*cross-validation*). Cette combinaison, reconnue pour offrir une meilleure évaluation des performances de généralisation du modèle, nécessite cependant d'importantes ressources de calcul en raison de la nature combinatoire du test de nombreuses combinaisons d'hyperparamètres. Dans notre cas, nous avons tiré parti d'Argo Workflows, un moteur de *workflows* open source conçu pour orchestrer des tâches parallèles sur Kubernetes, chaque tâche étant spécifiée comme un conteneur indépendant. Cela a permis de comparer facilement les performances des différents modèles entraînés et de sélectionner le meilleur en utilisant les outils de comparaison et de visualisation disponibles dans l'interface utilisateur de MLflow.

En résumé, la phase d'entraînement a été rendue à la fois efficace et reproductible grâce à l'utilisation de nombreux composants modulaires interconnectés — une caractéristique distinctive des technologies *cloud* — mis à disposition des *data scientists* grâce à Onyxia.

4.3.2 Déploiement d'un modèle

Une fois que les modèles candidats ont été optimisés, évalués et qu'un modèle performant a été sélectionné, l'étape suivante consiste à le rendre accessible aux utilisateurs finaux de l'application. Fournir simplement le modèle entraîné sous forme d'artefact, ou même uniquement le code pour l'entraîner, n'est pas une manière optimale de le transmettre, car cela suppose que les utilisateurs disposent des ressources, de l'infrastructure et des connaissances nécessaires pour l'entraîner dans les mêmes conditions. L'objectif est donc de rendre le modèle accessible de manière simple et interopérable, c'est-à-dire qu'il doit être possible de l'interroger avec divers langages de programmation et par d'autres applications de manière programmatique.

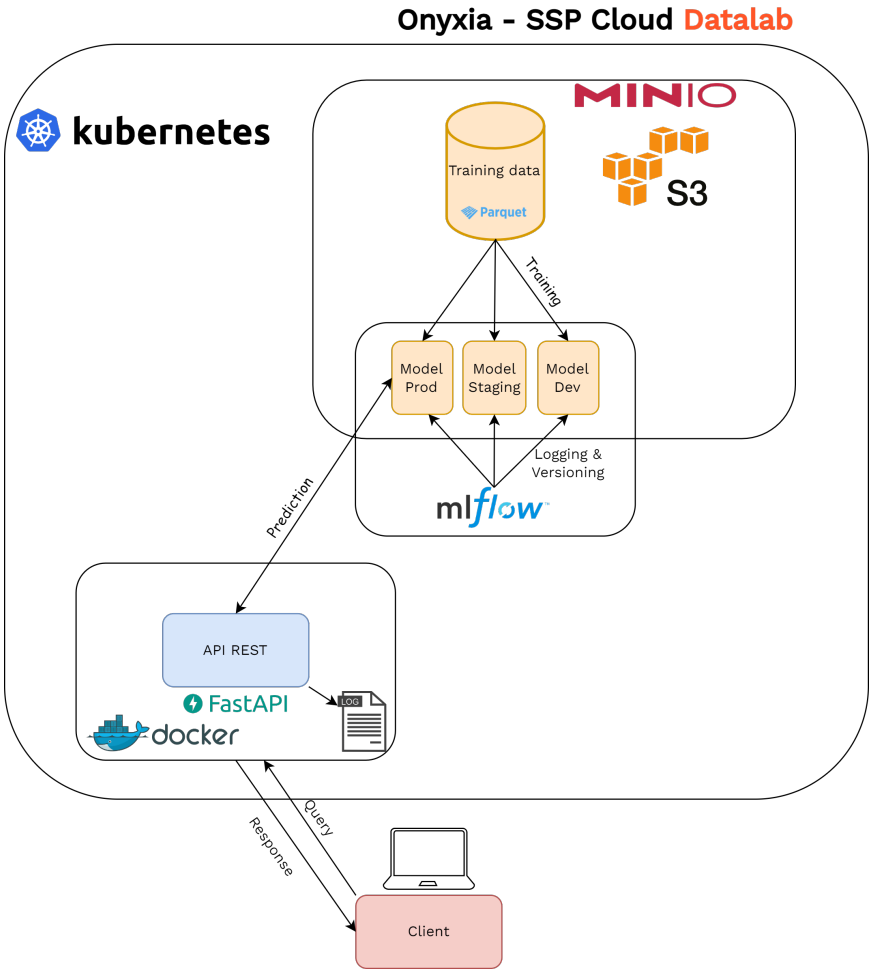
Dans ce contexte, nous avons choisi de déployer le modèle via une API REST. Cette technologie est devenue une norme pour servir des modèles de ML, car elle présente plusieurs avantages. Tout d'abord, elle s'intègre parfaitement dans un environnement orienté *cloud* : comme les autres composants de notre stack, elle permet d'interroger le modèle en utilisant des requêtes HTTP standard, ce qui contribue à la modularité du système. De plus, elle est interopérable : reposant sur des technologies standards pour les requêtes (requêtes HTTP) et les réponses (généralement une chaîne formatée en JSON), elle est largement indépendante du langage de programmation utilisé pour effectuer les requêtes. Enfin, les API REST offrent une grande évolutivité grâce à leur conception sans état (*stateless*)[21]. Chaque requête contient toutes les informations nécessaires pour être comprise et traitée, ce qui permet de dupliquer facilement l'API sur différentes machines pour répartir une charge importante — un processus connu sous le nom de scalabilité horizontale.

Nous avons développé l'API servant le modèle avec FastAPI[22], un framework web rapide et bien documenté pour construire des APIs avec Python. Le code de l'API et les dépendances logicielles nécessaires sont encapsulés dans une image Docker, ce qui permet de la déployer sous forme de conteneur sur le cluster Kubernetes. L'un des avantages majeurs de Kubernetes est sa capacité d'adapter la puissance de l'API — via le nombre de pods d'API effectivement déployés — en fonction de la demande, tout en fournissant un équilibrage de charge automatique. Au démarrage, l'API récupère automatiquement le modèle approprié depuis l'entrepôt de modèles MLflow stocké sur MinIO. Enfin, comme le code de l'application est *packagé* en utilisant l'API standardisée de MLflow — permettant par exemple d'intégrer directement l'étape de prétraitement dans chaque appel API — le code d'inférence reste largement uniforme, quel que soit le framework de ML sous-jacent utilisé. Ce processus de déploiement est résumé dans Figure 11.

[21] La conception sans état (*stateless*) fait référence à une architecture système où chaque requête d'un client au serveur contient toutes les informations nécessaires pour comprendre et traiter la requête. Cela signifie que le serveur ne stocke aucune information sur l'état du client entre les requêtes, ce qui permet de traiter chaque requête indépendamment. Cette conception simplifie l'évolutivité et renforce la robustesse du système, car n'importe quel serveur peut gérer une requête sans dépendre des interactions précédentes.

[22] https://fastapi.tiangolo.com

Figure 11. – Une approche basé sur des technologies cloud pour servir un modèle de ML via une API REST



4.3.3 Construction d'une pipeline intégrée

L'architecture construite à ce stade reflète déjà certains principes importants du MLOps. L'utilisation de la conteneurisation pour déployer l'API, ainsi que celle de MLflow pour suivre les expérimentations pendant le développement du modèle, garantit la reproductibilité des prédictions. L'utilisation de l'entrepôt central de modèles fourni par MLflow facilite la gestion du cycle de vie des modèles de manière collaborative. De plus, la modularité de l'architecture laisse de la place pour des améliorations ultérieures, puisque des composants modulaires peuvent être ajoutés ou modifiés facilement sans casser la structure du projet dans son ensemble. Comme nous le verrons dans les sections suivantes, cette propriété s'est avérée essentielle pour construire le projet de manière itérative, permettant d'ajouter une couche de surveillance du modèle (Chapitre 4.3.4) et un composant d'annotation (Chapitre 4.3.6) afin de favoriser l'amélioration continue du modèle en intégrant « l'humain dans le cycle de vie du modèle de ML » (*human in the loop*).

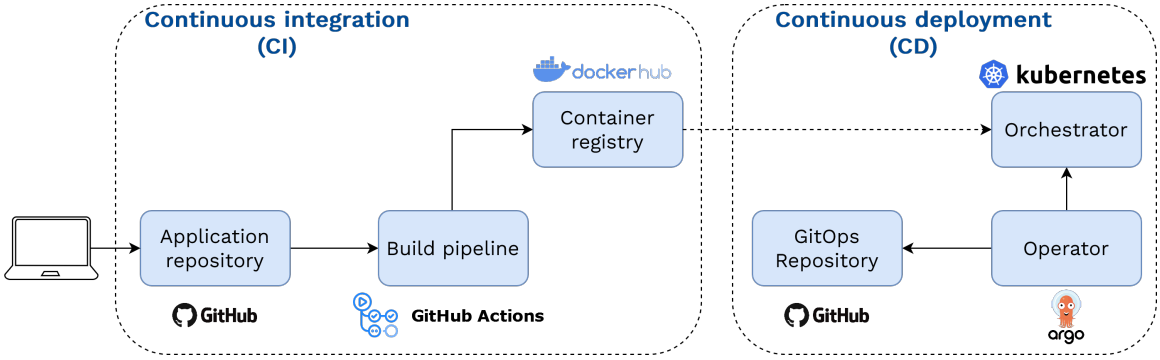
Cependant, la capacité à affiner l'architecture de base de manière itérative nécessite également une plus grande continuité dans le processus. À ce stade, le processus de déploiement implique

plusieurs opérations manuelles. Par exemple, l'ajout d'une nouvelle fonctionnalité à l'API nécessiterait de construire une nouvelle image, de la taguer, de mettre à jour les *manifests* Kubernetes utilisés pour déployer l'API et de les appliquer sur le cluster afin de remplacer l'instance existante avec un temps d'arrêt minimal. De même, un changement de modèle servi via l'API nécessiterait une simple modification du code, mais plusieurs étapes manuelles pour mettre à jour la version sur le cluster. En conséquence, les *data scientists* ne sont pas totalement autonomes pour prototyper et tester des versions mises à jour du modèle ou de l'API, ce qui limite le potentiel d'amélioration continue.

Afin d'automatiser ce processus, nous avons construit une pipeline CI/CD — un concept déjà présenté dans Chapitre 4.2.1 — intégrant ces différentes étapes. Figure 12 illustre notre implémentation spécifique de la pipeline CI/CD. Toute modification du code du dépôt de l'API, associée à un nouveau tag, déclenche un processus de build CI (implémenté avec GitHub Actions) de l'image Docker, qui est ensuite publiée sur un hub public de conteneurs (DockerHub). Cette image peut ensuite être récupérée et déployée par l'orchestrateur de conteneurs (Kubernetes) en spécifiant et en appliquant manuellement de nouveaux manifests pour mettre à jour les ressources Kubernetes de l'API.

Cependant, cette approche présente un inconvénient : elle limite la reproductibilité du déploiement, car chaque ressource est gérée indépendamment par l'orchestrateur, et le cycle de vie du déploiement de l'API dans son ensemble n'est pas contrôlé. Pour pallier cette lacune, nous avons intégré la partie déploiement dans une pipeline CD basée sur l'approche GitOps : les *manifests* des ressources de l'API sont stockés dans un dépôt Git. L'état de ce dépôt « GitOps » est surveillé par un opérateur Kubernetes (ArgoCD), de sorte à ce que toute modification des *manifests* de l'application soit directement propagée au déploiement sur le cluster. Dans cette pipeline intégrée, la seule action nécessaire pour que le *data scientist* déclenche une mise à jour de l'API est de modifier le tag de l'image de l'API indiquant la version à déployer.

Figure 12. – La pipeline CI/CD implémentée dans le projet



4.3.4 Surveillance d'un modèle en production

Une fois la phase initiale de développement du projet terminée — incluant l'entraînement, l'optimisation et le déploiement du modèle pour les utilisateurs —, il est crucial de comprendre que les responsabilités du *data scientist* ne s'arrêtent pas là. Traditionnellement, le rôle du *data scientist* se limite souvent à l'entraînement et à la sélection du modèle à déployer, le déploiement étant

généralement délégué au département informatique. Cependant, une spécificité des projets de ML est que, une fois en production, le modèle n'a pas encore atteint la fin de son cycle de vie : il doit être surveillé en permanence afin d'éviter toute dégradation indésirable des performances. La surveillance continue du modèle déployé est essentielle pour garantir la conformité des résultats aux attentes, anticiper les changements dans les données et améliorer le modèle de manière itérative. Même en production, les compétences du *data scientist* sont nécessaires.

Le concept de surveillance peut avoir différentes significations selon le contexte de l'équipe impliquée. Pour les équipes informatiques, il s'agit principalement de vérifier l'efficacité technique de l'application, notamment en termes de latence, de consommation de mémoire ou d'utilisation du disque de stockage. En revanche, pour les *data scientists* ou les équipes métier, la surveillance est davantage centrée sur le suivi méthodologique du modèle. Cependant, le suivi en temps réel des performances d'un modèle de ML est souvent une tâche complexe, car la vérité terrain (*ground truth*) n'est généralement pas connue au moment de la prédiction. Il est donc courant d'utiliser des proxys pour détecter les signes éventuels de dégradation des performances. Deux types principaux de dégradation d'un modèle ML sont généralement distingués. Le premier est le *data drift*, qui se produit lorsque les données utilisées pour l'inférence en production diffèrent significativement des données utilisées lors de l'entraînement. Le second est le *concept drift*, qui survient lorsqu'un changement dans la relation statistique entre les variables explicatives et la variable cible est observé au fil du temps. Par exemple, le mot « Uber » était habituellement associé à des codes liés aux services de taxis. Cependant, avec l'apparition des services de livraison de repas comme « Uber Eats », cette relation entre le libellé et le code associé a changé. Il est donc nécessaire de repérer au plus tôt ces changements afin de ne pas dégrader la codification.

Dans le cadre de notre projet, l'objectif est d'atteindre le taux le plus élevé possible de libellés correctement classifiés, tout en minimisant le nombre de descriptions nécessitant une intervention manuelle. Ainsi, notre objectif est de distinguer les prédictions correctes des prédictions incorrectes sans avoir accès au préalable à la vérité terrain. Pour y parvenir, nous calculons un indice de confiance, défini comme la différence entre les deux scores de confiance les plus élevés parmi les résultats renvoyés par le modèle. Pour une description textuelle donnée, si l'indice de confiance dépasse un seuil déterminé, la description est automatiquement codée. Sinon, elle est codée manuellement par un agent de l'Insee. Cette tâche de codification manuel est néanmoins assistée par le modèle ML : via une application qui interroge l'API, l'agent visualise les cinq codes les plus probables selon le modèle. Le seuil choisi pour l'indice de confiance est un paramètre que l'équipe métier peut utiliser pour arbitrer entre la charge de travail qu'elle est disposée à assumer pour la reprise gestionnaire et le taux d'erreurs qu'elle est prête à tolérer.

4.3.5 Définition du seuil de codification automatique et surveillance en production

La définition du seuil pour la codification automatique des descriptions textuelles a été une étape cruciale de ce processus, nécessitant un compromis entre un taux élevé de codification automatique et une performance optimale de celle-ci. Pour surveiller le comportement du modèle en production, nous avons développé un tableau de bord interactif permettant de visualiser plusieurs métriques d'intérêt pour les équipes métier. Parmi ces métriques figurent le nombre de requêtes par jour et le taux de codification automatique quotidien, pour un seuil donné pour l'indice de

confiance. Cette visualisation permet aux équipes métier de connaître le taux de codification automatique qu'elles auraient obtenu si elles avaient choisi différents seuils. Le tableau de bord représente également la distribution des indices de confiance obtenus et compare des fenêtres temporelles afin de détecter des changements dans les distributions des prédictions renvoyées par le modèle[23]. Enfin, les indices de confiance peuvent être analysés à des niveaux de granularité plus fins, basés sur les niveaux d'agrégation de la classification statistique, pour identifier les classes les plus difficiles à prédire et celles qui sont plus ou moins fréquentes.

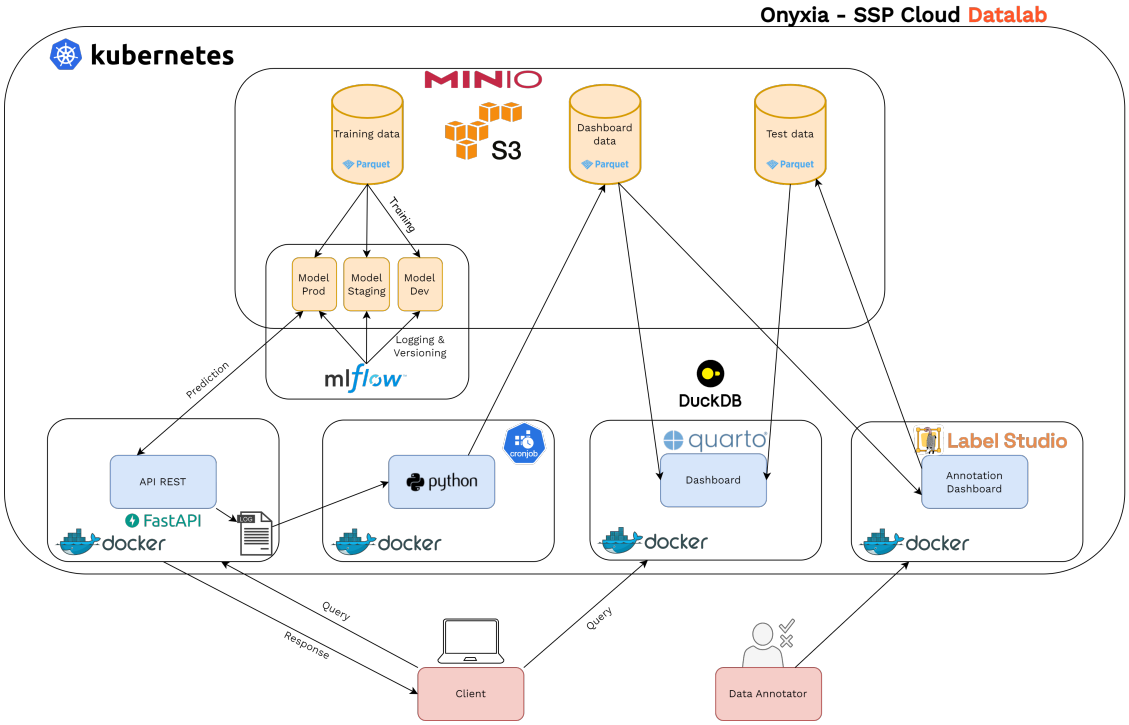
Figure 13 présente les composants ajoutés à l'architecture du projet pour fournir le tableau de bord de surveillance décrit ci-dessus. Tout d'abord, nous avons mis en place un processus simple en Python (deuxième composant de la rangée inférieure), qui récupère quotidiennement les logs de l'API et les transforme en fichiers partitionnés au format Parquet[24]. Ensuite, nous avons utilisé Quarto[25] pour construire un tableau de bord interactif (troisième composant de la rangée inférieure). Pour calculer les diverses métriques présentées dans le tableau de bord, les fichiers Parquet sont interrogés via le moteur optimisé DuckDB. À l'instar de l'API, le tableau de bord est construit et déployé sous forme de conteneur sur le cluster Kubernetes, et ce processus est également automatisé grâce à une pipeline CI/CD. Le composant d'annotation (quatrième composant de la rangée inférieure) est discuté dans la section suivante.

[23]Ces changements de distribution sont généralement vérifiés en calculant des distances statistiques — telles que la distance de Bhattacharyya, la divergence de Kullback-Leibler ou la distance de Hellinger — et/ou en effectuant des tests statistiques — tels que le test de Kolmogorov–Smirnov ou le test du khi-deux.

[24]Idéalement, les frameworks existants devraient être privilégiés par rapport aux solutions sur mesure pour adopter des routines standardisées. Lors de la construction de ce composant du pipeline, nous avons constaté que les frameworks *cloud* existants pour l'analyse des logs présentaient d'importantes limites. Cela constitue une piste d'amélioration pour le projet.

[25]Successeur de R Markdown, Quarto est devenu un outil essentiel. Il unifie les fonctionnalités de plusieurs packages très utiles de l'écosystème R Markdown tout en offrant une prise en charge native de plusieurs langages de programmation, dont Python et Julia en plus de R. Il est de plus en plus utilisé à l'Insee pour produire des documents reproductibles et les exporter dans divers formats.

Figure 13. – Notre implémentation d'une architecture MLOps



4.3.6 Favoriser l'amélioration continue du modèle

La composante de surveillance de notre modèle fournit une vue détaillée et essentielle de l'activité du modèle en production. En raison de la nature dynamique des données de production, les performances des modèles de ML ont tendance à diminuer avec le temps. Pour favoriser l'amélioration continue du modèle, il est donc essentiel de mettre en place des stratégies permettant de surmonter ces pertes de performance. Une stratégie couramment utilisée est le réentraînement périodique du modèle, nécessitant la collecte de nouvelles données d'entraînement plus récentes et donc plus proches de celle observées en production.

Plusieurs mois après le déploiement de la première version du modèle en production, le besoin de mettre en œuvre un processus d'annotation continue est devenu de plus en plus évident pour deux raisons principales. Premièrement, un échantillon de référence (*gold standard*) n'était pas disponible lors de la phase d'expérimentation. Nous avons donc utilisé un sous-ensemble des données d'entraînement pour l'évaluation, tout en sachant que la qualité de la labelisation n'était pas optimale. La collecte continue d'un échantillon de référence permettrait ainsi d'obtenir une vue réaliste des performances du modèle en production sur des données réelles, en particulier sur les données codifiées automatiquement. Deuxièmement, la refonte de la nomenclature statistique APE prévue en 2025 impose aux INS d'adopter la dernière version. Cette révision, qui introduit des changements importants, nécessite une adaptation du modèle et surtout la création d'un nouveau jeu de données d'entraînement. L'annotation de l'ancien jeu de données d'entraînement selon la nouvelle nomenclature statistique est donc indispensable.

Dans ce contexte, une campagne d'annotation a été lancée début 2024 pour construire de manière continue un jeu de données de référence. Cette campagne est réalisée sur le SSP Cloud[26] en utilisant le service Label Studio, un outil open source d'annotation offrant une interface ergonomique et disponible dans le catalogue d'Onyxia. Figure 13 montre comment le composant d'annotation (quatrième composant de la rangée inférieure) a pu être intégré facilement dans l'architecture du projet grâce à sa nature modulaire. En pratique, un échantillon de descriptions textuelles est tiré aléatoirement des données passées par l'API au cours des trois derniers mois. Cet échantillon est ensuite soumis à l'annotation par des experts APE via l'interface de Label Studio. Les résultats de l'annotation sont automatiquement sauvegardés sur MinIO, transformés au format Parquet, puis intégrés directement dans le tableau de bord de surveillance pour calculer et observer diverses métriques de performance du modèle. Ces métriques offrent une vision beaucoup plus précise des performances réelles du modèle sur les données de production, et permet notamment de détecter les cas les plus problématiques.

En parallèle, une campagne d'annotation pour construire un nouveau jeu d'entraînement adapté à la NAF 2025 a également été réalisée. En exploitant à la fois les nouvelles données d'entraînement et les métriques de performance dérivées de l'échantillon de référence, nous visons à améliorer la précision du modèle de manière itérative grâce à des réentraînements périodiques et automatique dès lors que le moteur de codification aura migré sur le cluster Kubernetes de production.

5 Discussion

Le développement des méthodes de *data science* offre un potentiel considérable pour la statistique publique. Cependant, notre capacité à tirer profit de ces nouvelles méthodes dépend essentiellement de notre aptitude à produire des chaînes de production de qualité, robustes et adaptés à leurs objectifs. Cette évolution nécessite une réflexion approfondie sur ce qui constitue une infrastructure moderne et évolutive pour la *data science* dans le domaine des statistiques publiques. Cet article présente le projet Onyxia, une proposition pour une telle plateforme développée à l'Insee. En exploitant des technologies *cloud* devenues des standards dans l'écosystème de la donnée, le projet vise à accroître l'autonomie des statisticiens dans l'orchestration de leurs traitements statistiques, tout en favorisant la reproductibilité des statistiques produites. Les technologies *cloud* étant notoirement difficiles à configurer, la valeur principale d'Onyxia réside dans leur accessibilité pour les statisticiens grâce à une interface ergonomique, simple d'utilisation, et un catalogue de services préconfigurés couvrant les usages les plus courants d'un statisticien public. À travers un projet interne visant à refondre le processus de codification de l'Activité Principale de l'Entreprise (APE) en utilisant des méthodes d'apprentissage automatique, nous illustrons comment Onyxia permet de construire de manière itérative des projets de *machine learning* prêts à passer en production, favorisant l'amélioration continue, un principe fondamental de l'approche MLOps.

Initialement développé comme un projet interne, Onyxia a acquis une reconnaissance dépassant le cadre de l'Insee ou de l'administration française. Convaincues du potentiel des technologies *cloud* pour renforcer l'autonomie et exploiter pleinement le potentiel de la *data science*, plusieurs organisations disposent désormais d'une instance de production d'Onyxia (comme c'est le cas à

[26] https://datalab.sspcloud.fr/

l'Insee avec le déploiement récent de LS^3), et de nombreuses autres sont en phase de test ou d'implémentation. Par ailleurs, le choix d'Onyxia comme plateforme de *data science* de référence dans le cadre du projet AIML4OS devrait encore accroître son adoption au sein du SSE. Cette tendance est naturellement très bénéfique pour le projet Onyxia, qui passe d'un projet développé en *open source* — mais principalement à l'Insee — à un véritable projet *open source* avec une base croissante de contributeurs. Cela, en retour, facilite son adoption par d'autres organisations, en offrant davantage de garanties sur sa pérennité indépendamment de la stratégie de l'Insee. La gouvernance du projet évolue actuellement pour refléter cette tendance, notamment avec l'organisation de réunions communautaires mensuelles et la création d'un canal public et d'une feuille de route pour le projet[27].

Malgré ce succès, nous constatons plusieurs limites à l'adoption généralisée du projet au sein des organisations. Tout d'abord, il est essentiel de rappeler que le choix fondamental fait par les organisations qui adoptent Onyxia ne porte pas sur le logiciel en lui-même, mais sur les technologies sous-jacentes : la conteneurisation (via Kubernetes) et le stockage d'objets. Ces technologies peuvent représenter des coûts d'entrée important pour les organisations, puisqu'elles nécessitent un investissement dans le développement et le maintien de compétences qui ne sont pas toujours présentes dans les INS. Cependant, on constate que les organisations qui manipulent de la donnée tendent de plus en plus à s'orienter vers des solutions *cloud* qui pourrait atténuer ces défis à long terme.

De même, la transition vers les technologies *cloud* impose des coûts d'entrée pour les statisticiens. Tout d'abord, ils sont souvent confrontés à une perte de repères quant à l'endroit où les calculs sont réellement effectués : bien qu'ils soient habitués à effectuer des calculs sur des serveurs centralisés plutôt que sur un ordinateur personnel, le conteneur ajoute une couche d'abstraction qui rend cet emplacement difficile à appréhender au départ. Mais le changement le plus perturbant dans ce paradigme est la perte de persistance des données. Dans les configurations traditionnelles — qu'il s'agisse d'un ordinateur personnel ou d'un serveur accessible via un bureau virtuel — le code, les données et l'environnement de calcul sont souvent mélangés dans une sorte de boîte noire. À l'inverse, les conteneurs, par construction, n'ont pas de persistance. Si le stockage d'objets fournit cette persistance, une utilisation adéquate de ces infrastructures exige une variété d'outils et de compétences : utilisation d'un système de contrôle de version pour le code (e.g. Git), interaction avec l'API de stockage d'objets pour enregistrer les données, gestion de fichiers de configuration et/ou de secrets et variables d'environnement, etc. En un sens, ces coûts d'entrée peuvent être considérés comme le « prix » de l'autonomie : grâce aux technologies *cloud*, les statisticiens ont désormais accès à des environnements évolutifs et flexibles leur permettant d'expérimenter plus librement, mais cette autonomie exige une montée en compétences significative, qui peut être intimidante et, *in fine*, limiter cette adoption. Cependant, notre expérience à l'Insee montre que cet effet peut être largement atténué grâce à une combinaison de formation des statisticiens aux bonnes pratiques de développement et d'accompagnement des projets statistiques lors de leur transition vers des infrastructures *cloud*.

[27] Toutes les informations sont disponibles sur le dépôt GitHub du projet : https://github.com/InseeFrLab/onyxia

Bien qu'Onyxia ait significativement démocratisé l'accès aux technologies *cloud* pour les statisticiens, l'intégration effective des méthodes de *data science* dans la production statistique des INS soulève des défis plus larges, d'ordre organisationnel. Une leçon majeure tirée du déploiement de notre premier modèle de *machine learning* en production est la nécessité de surmonter la segmentation des compétences entre les équipes informatiques, métier et innovation. Par nature, les projets de *machine learning*, pour pouvoir passer en production, impliquent un large éventail de compétences — connaissance du domaine métier, entraînement et amélioration des modèles ainsi que leur déploiement et supervision — et nécessitent donc une collaboration efficace entre des professionnels aux cultures de travail et langages de programmation variés. Notre expérience montre que les technologies *cloud*, en favorisant l'autonomie des *data scientists*, apportent plus de continuité aux projets d'apprentissage automatique et facilitent cette collaboration essentielle entre les différents profils. Toutefois, répondre pleinement à ces défis nécessite des choix qui dépassent le domaine technique. Par exemple, intégrer certaines compétence en *data science* directement au sein des équipes métier, en complément des équipes d'innovation centralisées, pourrait favoriser une meilleure collaboration. De même, recruter des profils qui ne sont pas traditionnellement présents dans les INS, comme les *data engineers* ou les *ML engineers*, pourrait apporter de nouvelles compétences à l'intersection des méthodologies statistiques et des techniques informatiques. Au final, la transition vers une approche axée sur la *data science* dans la production statistique doit s'appuyer sur une stratégie équilibrée qui lie des solutions techniques comme Onyxia avec des ajustements organisationnels et humains, favorisant une culture de collaboration, de formation continue et d'innovation.

Bibliographie

[1] T. Gjaltema, « High-Level Group for the Modernisation of Official Statistics (HLG-MOS) of the United Nations Economic Commission for Europe », *Statistical Journal of the IAOS*, vol. 38, nº 3, p. 917-922, 2022.

[2] DGINS, « Bucharest Memorandum on Official Statistics in a Datafied Society ». 2018.

[3] INSEE, « Horizon 2025 ». 2016.

[4] EUROSTAT, « ESSnet Big Data 2 - Final Technical Report ». 2021.

[5] B. Sakarovitch, M.-P. d. Bellefon, P. Givord, et M. Vanhoof, « Estimating the residential population from mobile phone data, an initial exploration », *Economie et Statistique*, vol. 505, nº 1, p. 109-132, 2018.

[6] M. Leclair, I. Léonard, G. Rateau, P. Sillard, G. Varlet, et P. Vernédal, « Scanner data: advances in methodology and new challenges for computing consumer price indices », *Economie et Statistique*, vol. 509, nº 1, p. 13-29, 2019.

[7] P. Descy, V. Kvetan, A. Wirthmann, et F. Reis, « Towards a shared infrastructure for online job advertisement data », *Statistical Journal of the IAOS*, vol. 35, nº 4, p. 669-675, 2019.

[8] D. Salgado, L. Sanguiao-Sande, S. Barragán, B. Oancea, et M. Suarez-Castillo, « A proposed production framework with mobile network data », in *ESSnet Big Data II - Workpackage I - Mobile Network Data*, 2020.

[9] A. Kowarik et M. Six, « Quality Guidelines for the Acquisition and Usage of Big Data with additional Insights on Web Data », in *4th International Conference on Advanced Research Methods and Analytics (CARMA 2022)*, 2022, p. 269-270.

[10] F. Ricciato, F. De Meersman, A. Wirthmann, G. Seynaeve, et M. Skaliotis, « Processing of mobile network operator data for official statistics: the case for public-private partnerships », in *104th DGINS conference*, 2018.

[11] O. Lefebvre, M. Soulier, et T. Tortosa, « L'accueil des données administratives: un processus structurant », *Courrier des statistiques*, p. 141-142, 2024.

[12] T. H. Davenport et D. Patil, « Data scientist », *Harvard business review*, vol. 90, nº 5, p. 70-76, 2012.

[13] L. Liu, « Computing infrastructure for big data processing », *Frontiers of Computer Science*, vol. 7, p. 165-170, 2013.

[14] A. Saiyeda et M. A. Mir, « Cloud computing for deep learning analytics: A survey of current trends and challenges. », *International Journal of Advanced Research in Computer Science*, vol. 8, nº 2, 2017.

[15] S. Ghemawat, H. Gobioff, et S.-T. Leung, « The Google file system », in *Proceedings of the nineteenth ACM symposium on Operating systems principles*, 2003, p. 29-43.

[16] A. I. Abdelaziz, K. A. Hanson, C. E. Gaber, et T. A. Lee, « Optimizing large real-world data analysis with parquet files in R: A step-by-step tutorial », *Pharmacoepidemiology and Drug Safety*, 2023.

[17] J. Dean et S. Ghemawat, « MapReduce: simplified data processing on large clusters », *Communications of the ACM*, vol. 51, nº 1, p. 107-113, 2008.

[18] B. Dhyani et A. Barthwal, « Big data analytics using Hadoop », *International Journal of Computer Applications*, vol. 108, nº 12, p. 975-8887, 2014.

[19] J. Tigani, « Big data is dead ». [En ligne]. Disponible sur: https://motherduck.com/blog/big-data-is-dead/

[20] M. Leclair et others, « Using Scanner Data to Calculate the Consumer Price Index », *Courrier des statistiques*, vol. 3, p. 61-75, 2019.

[21] S. Vale, « International collaboration to understand the relevance of Big Data for official statistics », *Statistical Journal of the IAOS*, vol. 31, nº 2, p. 159-163, 2015.

[22] A. S. Foundation, « Apache Parquet ». 2013.

[23] D. Abadi, P. Boncz, S. Harizopoulos, S. Idreos, S. Madden, et others, « The design and implementation of modern column-oriented database systems », *Foundations and Trends® in Databases*, vol. 5, nº 3, p. 197-280, 2013.

[24] A. S. Foundation, « Apache Arrow ». 2016.

[25] M. Raasveldt et H. Mühleisen, « Duckdb: an embeddable analytical database », in *Proceedings of the 2019 International Conference on Management of Data*, 2019, p. 1981-1984.

[26] Y. Li *et al.*, « Big data and cloud computing », *Manual of digital earth*, p. 325-355, 2020.

[27] O. Bentaleb, A. S. Belloum, A. Sebaa, et A. El-Maouhab, « Containerization technologies: Taxonomies, applications and challenges », *The Journal of Supercomputing*, vol. 78, nº 1, p. 1144-1181, 2022.

[28] R. Vaño, I. Lacalle, P. Sowiński, R. S-Julián, et C. E. Palau, « Cloud-native workload orchestration at the edge: A deployment review and future directions », *Sensors*, vol. 23, nº 4, p. 2215-2216, 2023.

[29] Q. Zhang, L. Liu, C. Pu, Q. Dou, L. Wu, et W. Zhou, « A comparative study of containers and virtual machines in big data environment », in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, 2018, p. 178-185.

[30] S. Samundiswary et N. M. Dongre, « Object storage architecture in cloud for unstructured data », in *2017 International Conference on Inventive Systems and Control (ICISC)*, 2017, p. 1-6.

[31] M. Mesnier, G. R. Ganger, et E. Riedel, « Object-based storage », *IEEE Communications Magazine*, vol. 41, nº 8, p. 84-90, 2003.

[32] L. Leite, C. Rocha, F. Kon, D. Milojicic, et P. Meirelles, « A survey of DevOps concepts and challenges », *ACM Computing Surveys (CSUR)*, vol. 52, nº 6, p. 1-35, 2019.

[33] M. McNutt, « Reproducibility », *Science*, vol. 343, nº 6168, p. 229-230, 2014.

[34] European Commission, « European Statistics Code of Practice — revised edition 2017 ».

[35] S. Luhmann, J. Grazzini, F. Ricciato, M. Mészáros, J.-M. Museux, et M. Hahn, « Promoting reproducibility-by-design in statistical offices », in *2019 New Techniques and Technologies for Statistics (NTTS) conference*, 2019, p. . doi: 10.5281/zenodo.3240198.

[36] D. Moreau, K. Wiebels, et C. Boettiger, « Containers for computational reproducibility », *Nature Reviews Methods Primers*, vol. 3, nº 1, p. 50-51, 2023.

[37] S. Gokhale *et al.*, « Creating helm charts to ease deployment of enterprise application and its related services in kubernetes », in *2021 international conference on computing, communication and green engineering (CCGE)*, 2021, p. 1-5.

[38] J. Opara-Martins, R. Sahandi, et F. Tian, « Critical analysis of vendor lock-in and its impact on cloud computing migration: a business perspective », *Journal of Cloud Computing*, vol. 5, p. 1-18, 2016.

[39] J. Opara-Martins, M. Sahandi, et F. Tian, « A holistic decision framework to avoid vendor lock-in for cloud saas migration », *Computer and Information Science*, vol. 10, nº 3, 2017.

[40] C. M. Schweik, « Free/open-source software as a framework for establishing commons in science », 2006.

[41] F. Comte, A. Degorre, et R. Lesur, « SSPCloud: a creative factory to support experimentations in the field of official statistics », *Courrier des Statistiques, INSEE*, vol. 7, p. 68-85, 2022.

[42] H. W. Chesbrough, *Open innovation: The new imperative for creating and profiting from technology*. Harvard Business Press, 2003.

[43] E. Meyer et P. Rivière, « SICORE, un outil et une méthode pour le chiffrement automatique à I'INSEE », in *Actes de la 4ème Conférence Internationale des Utilisateurs de Blaise*, mai 1997, p. 280-293. Consulté le: 4 juillet 2023. [En ligne]. Disponible sur: http://www.blaiseusers.org/1997/papers/meyer97.pdf

[44] Q. Li *et al.*, « A survey on text classification: From traditional to deep learning », *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 13, nº 2, p. 1-41, 2022.

[45] A. Joulin, E. Grave, P. Bojanowski, et T. Mikolov, « Bag of Tricks for Efficient Text Classification », *arXiv preprint arXiv:1607.01759*, 2016.