« The Analytical Engine has no pretensions to originate anything. It can do whatever we know how to order it to perform »

Ada Lovelace, créditée pour avoir créé le premier programme informatique

Édito

Ce Tech Radar 100% IA, co-construit lors de plusieurs journées d'ateliers, est l'image des débats qui animent nos équipes de Data Science depuis 7 ans chez Sicara.

Il va bien au-delà du choix d'algorithmes : ce radar reflète l'ensemble des questions que peut être amenée à se poser une équipe de Data Scientists et Machine Learning Engineers, depuis les phases de R&D jusqu'au déploiement et au suivi d'algorithmes en production.

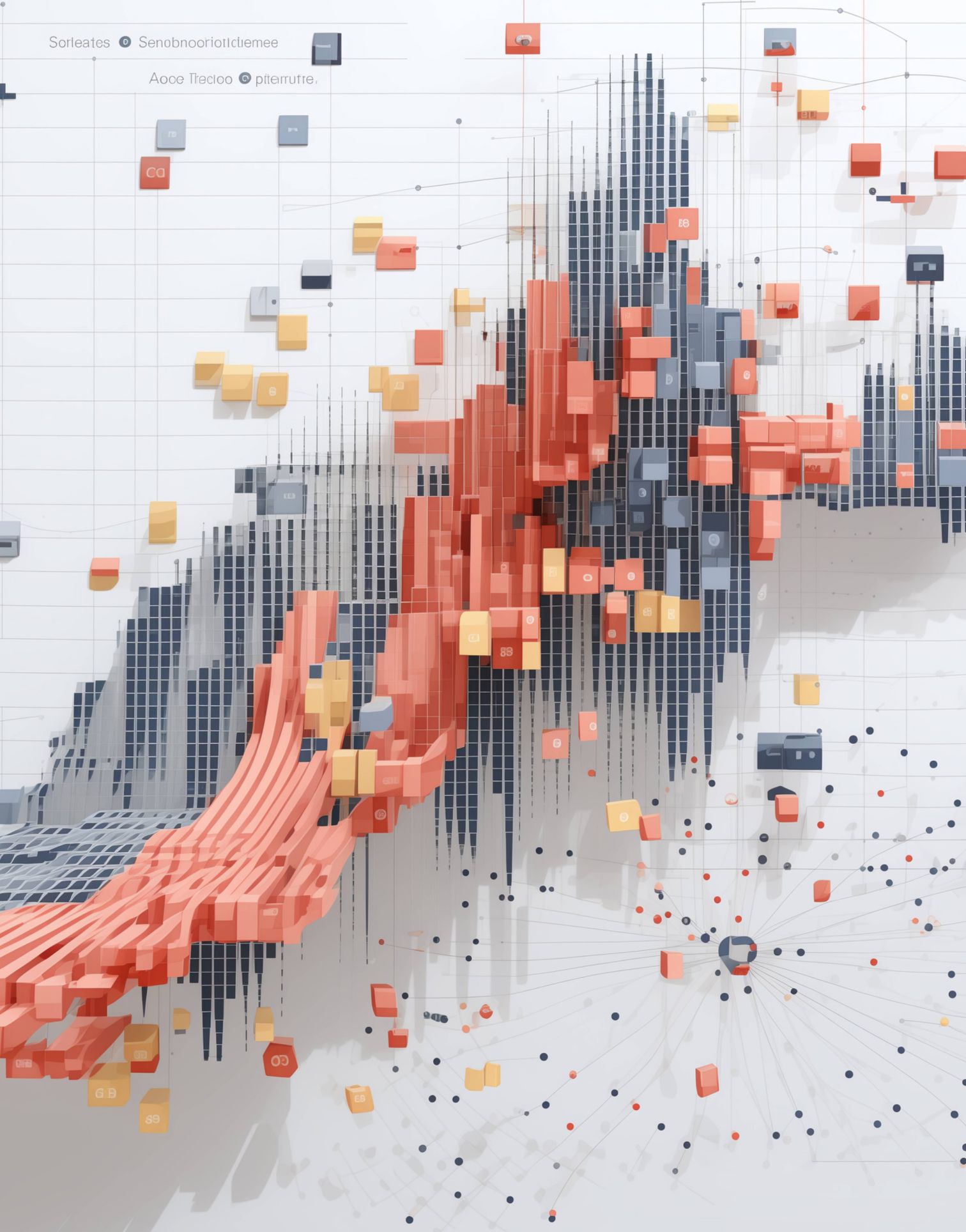
Découpé en quatre quadrants, il reflète l'étendue des outils techniques et méthodologiques nécessaires pour développer des produits d'Intelligence Artificielle :

- **Data :** sans données, les meilleurs algorithmes n'auront aucune utilité. Avec l'émergence de modèles toujours plus puissants, accessibles et généralisables à différents cas d'usage, la donnée spécifique à chaque entreprise devient un différenciant de plus en plus critique. Ce quadrant adresse les solutions de stockage et de gestion de la donnée indispensables aux Data Scientists pour maîtriser leur donnée de bout en bout.
- **Algorithmes :** la brique indispensable spécifique aux produits intégrants de l'IA.
- **Industrialisation :** la question suivante qui se pose est de trouver les bons outils pour entraîner, versionner et déployer ces algorithmes. Nous avons pris le parti d'intégrer quelques solutions très standards dans d'autres écosystèmes, notamment issues du monde du développement ou du data engineering. Notre expérience est que certaines pratiques très standardisées pour des développeurs ou Data Engineers ne le sont pas du tout pour des équipes de Data Scientists.
- Enfin, nous avons dédié un quadrant aux pratiques des équipes travaillant sur des solutions algorithmiques : le quadrant **Méthodes & Enablers.**

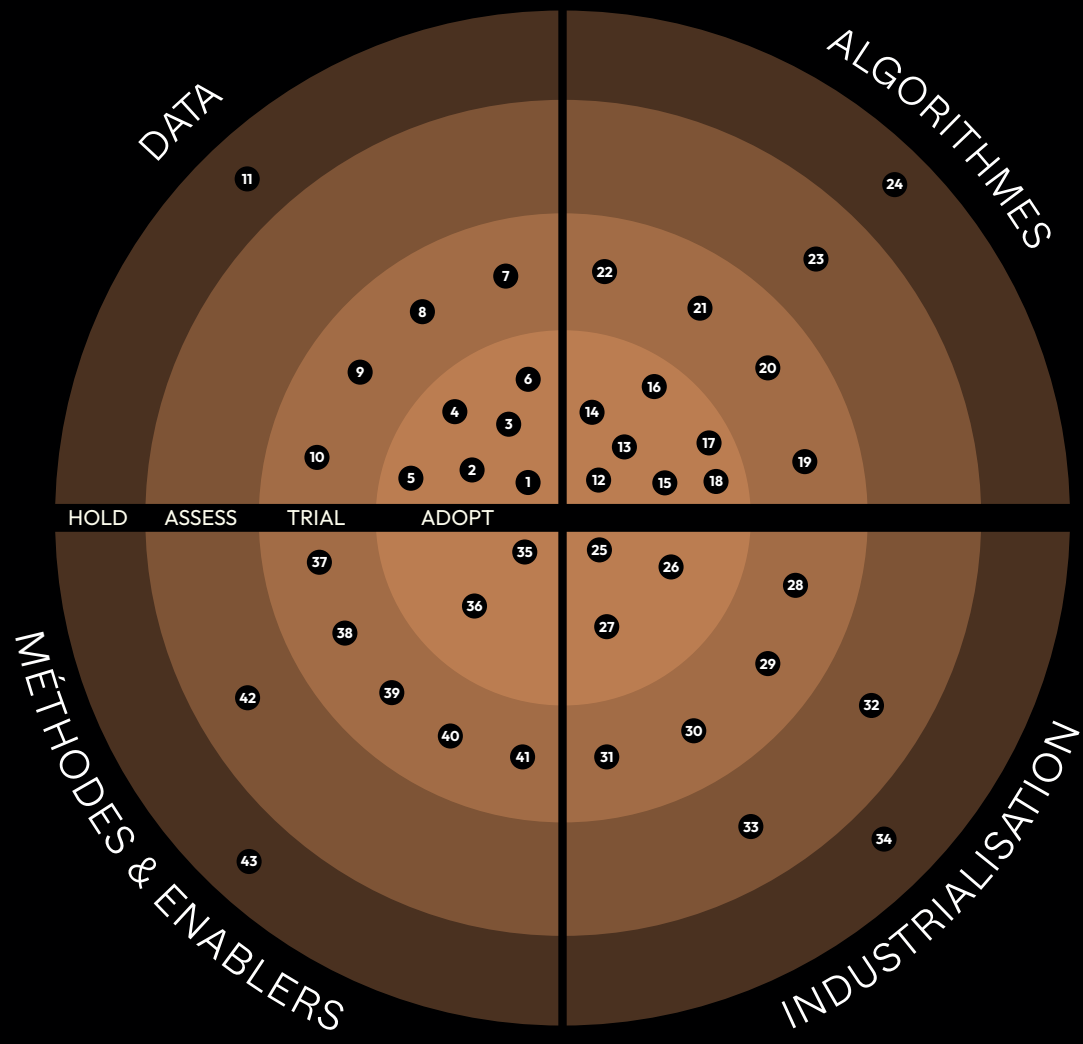
Je vous souhaite une excellente lecture, et au plaisir d'en discuter avec vous !

PIERRE-HENRI CUMENGE

CTO Sicara



Le Tech Radar IA



4 niveaux d'adoption

ADOPT

Nous recommandons chaudement cette technologie, pas ou peu de limitations, les contre-mesures aux limitations sont claires et simples.

TRIAL

Ces technologies sont prêtes à l'emploi, nous recommandons de les explorer malgré quelques limitations bien identifiées.

ASSESS

Usage prudent recommandé : soyez attentifs aux écueils, surtout pour les versions bêta ou les technologies émergentes.

HOLD

Nous avons identifié plusieurs limitations et de meilleures alternatives. Nous ne recommandons pas ce blip dans le cas général, vous pouvez le choisir avec précaution en fonction de votre contexte.

Les blips

DATA

1. Apache Parquet
2. DVC
3. Polars
4. Prodigy
5. Pydantic
6. Streamlit
7. Bases de données vectorielles dédiées
8. Presidio
9. Segments.ai
10. TensorBoard Embedding Projector
11. Pandas avec backend NumPy

ALGORITHMES

12. DINOv2 - comme modèle d'embedding d'image
13. GPT-4
14. PyTorch
15. Retrieval Augmented Generation
16. SHAP
17. SpaCy
18. YOLO
19. Boruta
20. Causal impact
21. OpenAI assistants API
22. LLM Fine-tuning supervisé sur des Q&A
23. LLM Open Source
24. Few-shot learning classique

INDUSTRIALISATION

25. Airflow
26. Infrastructure as Code
27. Poetry
28. LangChain
29. Qarnot
30. vLLM
31. LangSmith
32. Guidance
33. Plateforme de ML end-to-end
34. Dataiku

MÉTHODES & ENABLERS

35. Algorithm Decision Record
36. Métriques business
37. GitHub Copilot
38. Itérations courtes en IA
39. Test-Driven Machine Learning
40. Sicarator
41. Massive Text Embedding Benchmark (MTEB)
42. LLM -as-a-judge
43. Itérations sans métriques sur des modèles basés sur les LLMs

Data

11 BLIPS | 6 ADOPT | 4 TRIAL | 1 HOLD

En IA, l'attention se porte souvent sur le développement et le raffinement des modèles. Pourtant, une grande partie de la qualité des résultats d'un modèle de Machine Learning est imputable à la préparation et la gestion des données.

Ce quadrant met en lumière l'importance cruciale de la data, qui constitue la fondation indispensable de tout projet d'IA réussi. Nous y mettons l'accent sur les outils et pratiques les plus efficaces, des solutions de labellisation aux techniques de visualisation, en passant par des outils de transformation, manipulation et de stockage de données.

ADOPT
1/6 blips

1 Apache Parquet

Stocker ses datasets au bon format est souvent une non-question en data science tellement **le format CSV est un standard**. Pourtant, les limites de ce format sont clairement identifiées :

- Les fichiers CSV peuvent devenir **volumineux et lents** à lire et à écrire avec de grandes quantités de données.
- Les CSV ne sont **pas typés** ; c'est donc au logiciel lisant le fichier d'inférer les types, ce qui peut conduire à différents bugs. Les structures complexes, comme les listes ou les dictionnaires, sont par exemple par défaut interprétées comme des chaînes de caractères.
- Enfin, il est tentant d'éditer les fichiers CSV **manuellement**, ce qui peut entraîner des **erreurs**. Il arrive par exemple qu'on veuille directement renommer des labels erronés dans un CSV d'annotations, sans afficher les images correspondantes pour s'assurer qu'on a bien compris le problème.

Apache Parquet est un **format de données open-source orienté colonnes**. Sa conception le rend particulièrement **efficace** pour les opérations de lecture et d'écriture sur de grands ensembles de données.

Or, le format de fichier **Parquet** résout chacun des 3 points précédents :

- Il est en moyenne **dix fois plus léger** que les fichiers CSV et considérablement **plus rapide** à lire et à écrire.
- Il gère correctement **tous les types de données**, y compris les structures complexes.
- Enfin, le fichier Parquet n'est pas facilement modifiable manuellement, ce qui **encourage l'utilisation de scripts ou d'interfaces graphiques dédiées** (outils de labellisation ou Streamlit (p.19) sur-mesure) pour effectuer les modifications dans de meilleures conditions.

Ce format permet plus d'optimisation comme le partitionnement ou la compression du fichier.

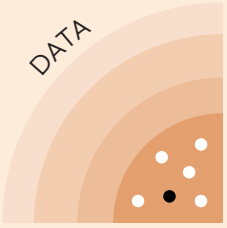
Les librairies de Dataframe comme Polars (p.16) ou Pandas (p.24) gérant nativement le format Parquet (avec des méthodes `read_parquet` et `to_parquet` équivalentes aux méthodes `read_csv` et `to_csv`), il est très **facile de migrer** d'un format à un autre.

L'inconvénient de ce format est qu'il est moins accessible pour un profil non-tech qui pouvait ouvrir un CSV sur un logiciel de tableur.

NOTRE POINT DE VUE

Bien que ce format soit couramment utilisé par les Data Engineers, nous avons choisi de l'intégrer dans nos recommandations pour ce tech radar IA car il reste **sous utilisé par les Data Scientists**.

Pour gagner en **performance** et réduire le **risque de bugs**, nous recommandons aux Data Scientists de **remplacer leurs fichiers CSV** par des fichiers au **format Parquet pour stocker leurs données tabulaires**. Nous l'utilisons en combinaison avec des outils permettant de visualiser et éditer les données en question de manière efficace et rigoureuse, par exemple via Streamlit (p.19)



ADOPT
2/6 blips

2 DVC

Le **versioning du code** est une bonne pratique standard sur la grande majorité des projets de Machine Learning, mais ce n'est malheureusement pas encore le cas du **versioning des *données***. Beaucoup de data scientists ont en effet déjà connu cette frustration de ne pas remettre la main sur tel dataset ou tel modèle si performant. **DVC** (pour **Data Version Control**) est une librairie **Python open-source** lancée en 2017 par **Iterative** pour éviter ce genre de cafouillages.

L'outil permet de **versionner** n'importe quels fichiers de données en capitalisant sur le versioning de **Git**. Il enregistre les **données** sur le **stockage distant** choisi (Google Cloud Storage ou S3 par exemple), et versionne les **métadonnées** correspondantes via Git.

DVC permet également de définir des **pipelines de données**, de les exécuter et de versionner les différents runs. Ainsi, il est possible de retracer les différentes étapes qui ont mené à la création de chacun des fichiers et de les reproduire.

Il existe d'autres systèmes de versioning de données et de pipelines, comme Pachyderm, mais aucun d'entre eux n'égalent DVC en terme de **facilité de mise en place et d'utilisation**.

D'autres outils permettent par ailleurs de **tracker les expériences de ML**, comme **MLFlow** ou **Weights & Biases**. La force de DVC par rapport à ces derniers est de **s'intégrer de manière beaucoup plus transparente** dans le workflow Git et dans la code base du projet :

- L'historique de DVC nativement lié à celui de Git permet d'avoir au même endroit les itérations sur le code, les données et les expériences, facilitant l'exploration du passé.
- Le logging des paramètres, inputs et outputs est fait en dehors du code, évitant ainsi de l'alourdir avec des opérations spécifiques au tracking (comme c'est le cas avec MLFlow).

Pour explorer facilement les expériences dans une interface graphique, plusieurs options sont possibles :

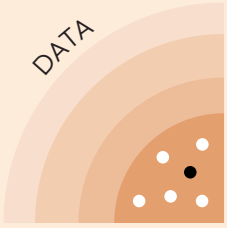
- **Iterative Studio :** une app web développée par l'entreprise derrière DVC, assez complète mais coûtant 50$ par utilisateur par mois à partir de 2 utilisateurs
- **L'extension DVC pour VSCode :** gratuite, mais moins complète en terme de collaboration et seulement disponible sur VSCode
- **Un dashboard sur-mesure de tracking d'expérience** que l'on peut construire en combinant DVC et un outil de visualisation comme Streamlit (p.19), pour explorer les expériences de son projet en affichant exactement les informations dont on a besoin sur le projet

NOTRE POINT DE VUE

À l'instar de Git, une fois qu'on a pris l'habitude d'utiliser DVC, **on se demande comment un jour on a pu travailler sans**.

Pour **visualiser les expériences** trackées par DVC avec un maximum de flexibilité et sans avoir à utiliser de solution payante, nous suggérons de combiner **DVC** avec **Streamlit** (p.19). C'est l'approche que nous avons adoptée chez Sicara et dont **Sicarator** (p.64) facilite la mise en place.

Attention, DVC est conçu pour la gestion du **flux d'experimentation**. Nous ne recommandons pas son usage pour des pipelines en production : un outil comme **Airflow** (p.46) sera plus approprié.



ADOPT
3/6 blips

3 Polars

Pandas s'est imposé ces dernières années pour **l'analyse et la transformation des données tabulaires** en Python. Elle **atteint ses limites sur les gros volumes de données** car ses traitements sont non parallèlisables et la donnée traitée doit entièrement être chargée en mémoire.

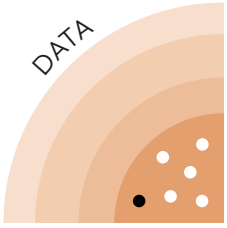
Polars est une bibliothèque open-source de **Dataframes** écrite en Rust, rendue publique en 2021. Elle dépasse les limites de Pandas en rendant possible d'effectuer **des calculs multithreads**. La performance est encore améliorée grâce au **lazy evaluation** pour optimiser les transformations et les **calculs par batch** pour gérer des **volumes de données plus grand que la mémoire**.

Sa syntaxe expressive permet une **grande variété de transformations**, comparables à celles réalisables avec Pandas. De plus, le passage de Pandas à Polars est grandement simplifié car Polars reprend **la syntaxe des fonctions et méthodes de Pandas**.

Cependant, Polars est récent et n'a pas encore la richesse de Pandas. Néanmoins, grâce à l'implémentation du format de données Arrow, **il est facile de transformer un dataframe polars en Pandas** et ainsi de rebasculer vers Pandas pour des traitements spécifiques.

NOTRE POINT DE VUE

Nous recommandons donc d'utiliser **Polars au lieu de Pandas** (p.24) pour tout nouveau projet utilisant des DataFrames, et de ne conserver cette dernière librairie qu'en contexte de code legacy.



ADOPT
4/6 blips

4 Prodigy

Annoter des datasets est une tâche non-trivale : faciliter l'annotation, la sauvegarder, permettre des annotateurs parallèles...

Prodigy est **un outil d'annotation de données textuelles**, telle que la classification, la reconnaissance des entités ou l'analyse des phrases.

Prodigy présente de nombreux avantages. Il intègre :

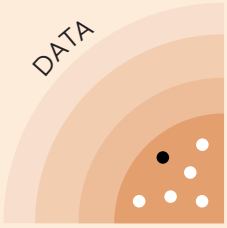
- Une interface user-friendly
- **De l'apprentissage actif (active learning)** intégré à Prodigy améliore l'efficacité du processus d'annotation en dirigeant les utilisateurs vers les exemples les plus pertinents.
- Une pré-annotation des textes pour des tâches telles que la **reconnaissance d'entités nommées (NER)** et la **classification de texte**, en utilisant des LLMs. Attention tout de même à la confidentialité des données dans le cas d'utilisation de LLMs accessibles via des services tiers.
- La possibilité d'être **scripté** pour **personnaliser les workflows d'annotation**
- Un coût compétitif : 490$ pour une license à vie.

Il s'intègre très bien à SpaCy (developpé par la même équipe) (p.35).

Néanmoins, des alternatives open-sources telles que Doccano ou Label Studio pourraient être envisagées.

NOTRE POINT DE VUE

Nous utilisons Prodigy sur nos projets d'annotations textuels et notamment la feature d'active learning. Nous vous recommendons de faire de même en particulier si vous utilisez SpaCy (p.35)



ADOPT
5/6 blips

5 Pydantic

Python étant interprété, les **garanties sur les types de données** sont moins fortes que sur les langages compilés. Par ailleurs, les modèles de langages génèrent des sorties dont la structure n'est pas toujours garantie.

Pydantic est une **bibliothèque Python open source**, permettant **de définir et valider automatiquement une structure de donnée** attendue via des types python décrits par annotations. L'utilisation de Pydantic permet de répondre à la question : Comment **améliorer la fiabilité** et **la précision** des sorties des modèles de langage ?

Pydantic répond à ces interrogations en offrant un **cadre structuré** pour la **définition et la validation des données**. Cette approche est d'autant plus pertinente lorsqu'il s'agit d'intégrer la sortie de modèles de langage dans des applications, où **la cohérence et la précision des données** sont cruciales.

La communauté s'en rend de plus en plus compte et des outils comme **Marvin, Instructor ou Outlines** émergent et utilisent Pydantic pour créer des interfaces robustes avec les modèles. Pydantic est aussi une brique clé dans les output parsers de LangChain (p.49).

Comme Pydantic se base sur les types Python, pour exploiter pleinement ses capacités, il est essentiel d'avoir **une connaissance approfondie de la gestion des types de données en Python** et de leurs limites, ainsi que de comprendre comment ces types peuvent être utilisés et transformés pour modéliser efficacement des données complexes et structurées.

NOTRE POINT DE VUE

Nous utilisions Pydantic depuis plusieurs années pour nos projets de Data Engineering. Avec l'émergence du besoin dans les projets LLMs, Pydantic s'est imposé aussi comme un outil indispensable des projets IA. L'adoption de Pydantic représente un choix stratégique pour garantir **une gestion robuste des données**.

Un effet secondaire bénéfique de son utilisation sur nos projets a été de clarifier les modèles de données attendus, **documentant** et **explicitant** les interfaces entre composants.

Nous recommandons donc pydantic pour toutes les équipes ayant une connaissance suffisante de python.

ADOPT
6/6 blips

6 Streamlit

Les data scientists ont besoin de support visuel pour discuter avec le métier et les autres data scientists. Générer des applications custom rapidement est donc essentiel.

Streamlit est **une librairie Python permettant de créer des applications web** sans nécessité de connaissances en langages de développement web. C'est utile pour partager des rapports d'investigations techniques sur le web d'une manière plus visuel pour une personne du métier qu'un jupyter notebook.

La librairie **s'interface simplement avec DVC** ce qui facilite le flux et permet de comparer rapidement les performances et résultats de plusieurs modèles entraînés.

Pour aller plus loin, il est possible d'installer **des composants créés par la communauté**, dont la taille permet une bonne couverture des besoins.

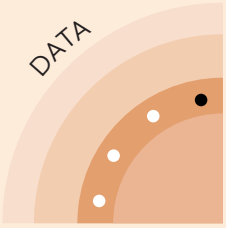
Le partage des pages webs créées est possible **en quelques clics et gratuitement** via le Streamlit Community Cloud. Mais un déploiement manuel sera nécessaire dans le cas d'une application que l'on souhaite garder privée.

Surtout, Streamlit convient pour des **web apps simples et dont la rapidité n'est pas critique**. Dans les autres cas, nous recommandons tout de même de s'intéresser à des frameworks web dédiés pour avoir toute la main sur la création de l'application et pouvoir créer des composants spécifiques plus facilement.

NOTRE POINT DE VUE

À Sicara, nous l'utilisons sur quasi tous les projets pour faire des **proof of concept** et tracks **d'investigation**. Nous développons généralement des solutions customisées pour aller en production.

Nous vous conseillons de l'essayer puisque le coût d'entrée et de création d'une app est très faible et permet de gagner beaucoup de temps sur le long terme.



TRIAL
1/4 blips

7 Bases de données vectorielles dédiées

L'information sémantique de texte ou d'image est habituellement encodée sous formes de vecteurs de taille fixe, appelés **embeddings**. Manipuler et requêter de tels vecteurs demande un outillage spécifique. Dès les années 2010, des librairies de **recherche vectorielle** se développent mais il n'y a pas de stockage associé.

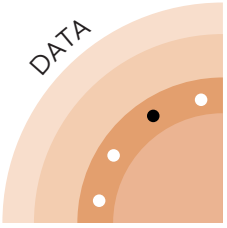
Les bases de données vectorielles dédiées apparaissent en 2019 avec **Milvus et Pinecone**.
Par rapport à une recherche vectorielle avec des **bases de données standards** (comme celles de PostgreSQL ou ElasticSearch), les bases de données vectorielles dédiées sont généralement **plus performantes** et offrent des fonctionnalités **plus spécifiques** : **quantization** des vecteurs, stockage des vecteurs sur disque pour sauver de l'espace en RAM et **minimiser les coûts**, ou même être capable de faire des recherches avec plusieurs vecteurs desquels s'approcher ou s'éloigner. Ce **gain en performance et en fonctionnalités** disponibles est notamment utile pour des cas d'usage avec **beaucoup de données** (à partir de plusieurs centaines de milliers de vecteurs, jusqu'à plusieurs milliards). Par exemple, Grok (le LLM de X) se base sur la base de données vectorielle Qdrant pour son système de RAG.

Malgré ces avantages, **les bases de données vectorielles peuvent être complexes à déployer et maintenir.** Par exemple, Milvus est conçue selon une architecture en micro-services, dont chaque service est complexe à comprendre et à débugger (documentation peu explicite), même avec une équipe dédiée. De plus, comparé à la recherche vectorielle avec des bases de données standards, cela nécessite d'**ajouter une brique supplémentaire à la stack technique**, ce qui est un choix non négligeable dans le cas d'infrastructures complexes. Cela peut aussi poser **des problèmes transactionnels,** comme des erreurs de serialisation, si la base de données vectorielle doit intéragir en temps réel avec une base de données standard.

NOTRE POINT DE VUE

Nous recommandons les **bases de données vectorielles dédiées** pour tout projet nécessitant une recherche sur un grand nombre de vecteurs. En particulier, nous utilisons souvent **Qdrant** qui se distingue par sa flexibilité, sa performance et son interfaçage avec DVC.

Ce choix **dépend de la stack technique existante**, de la complexité de l'intégration d'un nouvel outil et des problèmes de transaction anticipés. Une **base de données standard avec recherche vectorielle** peut être une bonne alternative.



TRIAL
2/4 blips

8 Presidio

Le respect de **la confidentialité des données** est un champ important du monde de la data, que ce soit par **respect des utilisateurs** ou des **contraintes légales** type RGPD ou CCPA. Avec l'apparition des **API externes** de **LLM** et le **shadow IA**, le risque d'envoyer des données personnelles s'accroit. Il est donc nécessaire d'avoir des solutions simples pour **anonymiser** les informations personnelles identifiables (IPI).

Presidio est une librairie python **open source** soutenue par Microsoft permettant la détection et l'anonymisation des données personnelles dans du texte. Il est spécifiquement conçu pour détecter et anonymiser des données personnelles sensibles telles que les **emails**, les **adresses IP** et les **numéros de téléphone**.

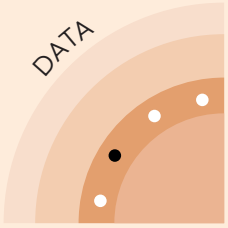
Presidio permet non seulement la **détection**, mais aussi le **remplacement (masking)** des informations sensibles, une caractéristique essentielle pour la **confidentialité** dans le traitement des données.

Sa mise en place est dans notre expérience très simple. Il est possible de choisir le modèle de détection d'entités sous-jacent, que ce soit un modèle sur étagère ou réentraîné.

Néanmoins, l'usage de **Presidio** requiert de définir précisément quelles données doivent être anonymisées, ce qui peut alourdir son usage. Par exemple, si je cherche à anonymiser les interactions d'un utilisateur avec un Chatbot avant de l'envoyer à une API LLM externe, comment faire pour anonymiser une demande qui contient un nom propre attendu comme dans la question suivante : "Dans le cadre de ma demande de prêt pour **mon entreprise X**, pourrais-tu me dire quels sont les taux habituellement pratiqués par <Nom d'une banque Banque> ?". Ici, le nom de la banque est une entité que l'on **ne veut pas anonymiser**, car elle porte une information essentielle pour construire la réponse, alors que le nom de l'entreprise de l'utilisateur doit être anonymisé avant d'être envoyé à l'API.

NOTRE POINT DE VUE

Nous avons utilisé **Presidio** pour **l'anonymisation des données** sur plusieurs projets, en particulier pour les applications impliquant des **LLMs**. Sa mise en place très simple permet de l'utiliser même sur des projets exploratoires avec un budget de mise en place réduit.



TRIAL
3/4 blips

9 Segments.ai

Dans les projets de Data Science, avoir des données **correctement** labellisées est très important mais demande beaucoup d'efforts.

Segments.ai, startup fondée en 2020, s'est donnée pour mission de **supprimer les coûts d'annotation d'image**. Leur plateforme s'est révélée efficace et facile à utiliser pour les tâches d'annotations d'images, avec plusieurs points forts notables :

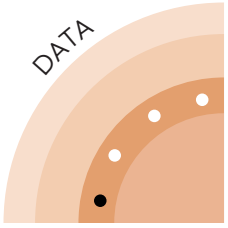
1. **La gestion de Dataset dans nos pipelines MLOps :** SegmentsAI offre la possibilité de **versionner ses datasets** et propose un **SDK Python**, qui permet une intégration simple dans nos scripts Python.
2. **Collaboration :** La plateforme propose **différents rôles** pour gérer le **cycle de vie des annotations**, comme revoir, valider ou rejeter des annotations, ce qui facilite la collaboration au sein des équipes. Cette validation est automatisable via le SDK Python.
3. **Rapidité :** Pour la segmentation, Segments.ai permet d'utiliser des modèles de **Deep Learning pré-entraînés pour pré-annoter** les images. Cette fonctionnalité accélère considérablement le processus d'annotation.
4. **Réactivité :** L'intégration de SAM (Segment Anything Model, de Meta AI) peu de temps après sa sortie démontre la réactivité de Segments.ai à intégrer les nouvelles technologies et fonctionnalités.
5. **Prise en main** : L'interface utilisateur est intuitive, facilitant l'adoption de l'outil même si ça n'est pas la plus simple du marché.

Cependant certaines features manquent de flexibilité. Par exemple, les étapes du flux d'annotation sont figées et ne peuvent pas être customisées, tout comme la gestion des autorisations n'est pas granulaire.

Il y a d'autres alternatives, comme CVAT qui est open-source mais plus limité en termes de features disponibles.

NOTRE POINT DE VUE

On l'a utilisé sur plusieurs projets, **l'expérience est très agréable** mais le coût peut être rédhibitoire. Dans ce cas là nous proposons **CVAT** comme alternative **Open Source.** La solution est aussi relativement jeune, nous attendons d'avoir plus de recul sur sa maturité et sa robustesse avant d'avoir une recommendation définitive.



TRIAL
4/4 blips

10 TensorBoard Embedding Projector

Visualiser les résultats d'un modèle d'embedding peut être complexe. Tensorboard Embedding Projector adresse ce problème en projetant des embeddings à N dimensions (N>>3) dans un espace 3D via des **méthodes de réduction de dimensionalité** (par ex. t-SNE). Cela permet de visualiser de potentiels clusters mais aussi de zoomer pour observer des similarités spécifiques.
Il implémente des features simplifiant la visualisation, comme la possibilité d'associer chaque point à une image (utile dans le cas d'embeddings d'images) ou de colorier par metadata.

Le projecteur se crée en quelques lignes de code avec PyTorch (p.31).

Néanmoins, la projection reste une **approximation de la réalité** (2 points proches dans l'espace 3D sont potentiellement éloigné en réalité). De plus, l'UX de l'outil est relativement mauvaise lorsque l'on essaie de naviguer dans le nuage de points, rendant laborieux le fait de zoomer sur une zone particulière.

Il existe plusieurs alternatives proposant des projections d'embeddings, qui sont cependant en 2D :

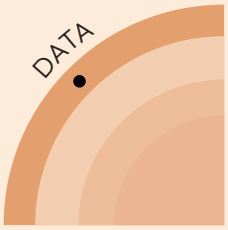
- La projection d'embeddings directement intégrée au dashboard de la base de données vectorielle dédiée Qdrant
- Le projection d'embeddings de l'outil FiftyOne

Pour une démonstration de Tensorboard Embedding Projector :

NOTRE POINT DE VUE

Sa **facilité de mise en place** ainsi que sa capacité à donner une première idée de la performance du modèle pré-entrainé choisi sur la donnée spécifique du projet expliquent pourquoi **tous nos projets de matching d'images via des embeddings** commencent par une exploration dans TensorBoard Embeddings.

Nous n'avons pas encore eu d'utilité pour cet outil pour des embeddings de textes longs, dont la visualisation en nuage de points est plus complexe à exploiter.



HOLD
1 blip

11 Pandas
avec backend NumPy

Avant 2009, le traitement de données en Python se limitait à des outils comme NumPy ou Python natif.

Pandas, en s'appuyant sur NumPy est rapidement devenu **un standard dans l'industrie et la recherche** permettant d'analyser et manipuler **des données tabulaires plus rapidement et efficacement** via sa couche d'abstraction. NumPy est efficace pour les opérations matricielles et permet des performances inatteignables en Python "pur" pour les opérations numériques car il utilise un langage compile comme le C. Malgré cela, la bibliothèque **montre ses limites dans la gestion de grands ensembles de données** ou de tâches complexes parce qu'il a été seulement pensé pour l'in-memory analytics et pas le big data.

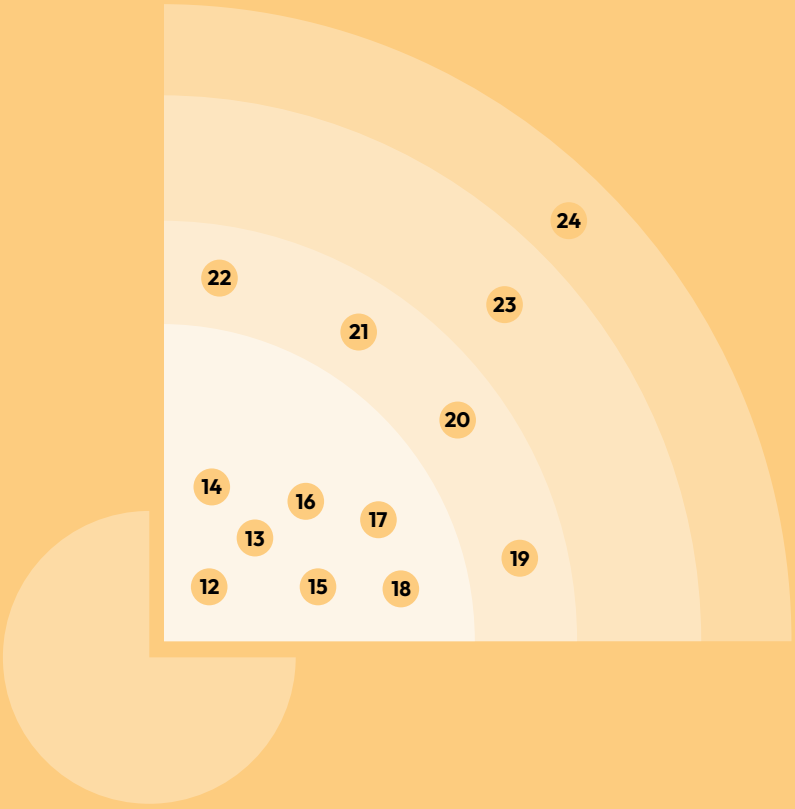
Aujourd'hui, l'écosystème Python pour les DataFrames s'est enrichi, notamment avec des alternatives au backend NumPy de Pandas, comme le backend PyArrow et Polars (une autre bibliothèque de Dataframes). Arrow, et **Polars**, écrit en Rust et axé sur le multi-threading, **offrent des performances nettement supérieures** car tous deux utilisent le format de données Arrow pour optimiser l'utilisation de la mémoire et les calculs.

NOTRE POINT DE VUE

Nous recommandons donc de délaisser le backend NumPy de Pandas. Bien que cette technologie soit complète et bien intégrée avec d'autres outils comme Matplotlib, **les options actuelles comme PyArrow ou Polars offrent des capacités similaires avec des optimisations plus avancées**. Face à l'augmentation des tailles de datasets, privilégier la performance devient essentiel. Les développeurs de Pandas reconnaissent également cette tendance, ayant **établi Arrow comme le backend par défaut** depuis la version 1.4.0 en janvier 2022.

Algorithmes

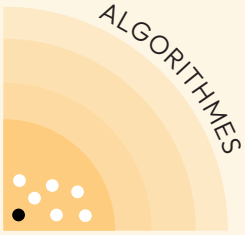
13 BLIPS | 7 ADOPT | 4 TRIAL | 1 ASSESS | 1 HOLD



Dans un projet d'Intelligence Artificielle (IA), la sélection du modèle ou de l'algorithme est une étape décisive de la conception d'un produit performant.

L'explosion en popularité des Large Language Models (LLM) a eu des impacts significatifs sur le monde de l'IA, avec des modèles propriétaires qui démontrent une performance et une accessibilité financière et technique accrue. Ces solutions ont remis les modèles d'IA sur le devant de la scène.

Au delà de la genAI, nous avons englobé dans ce quadrant trois catégories de technologies : celles facilitant la création d'algorithmes, celles qui permettent d'appeler des modèles extérieurs, et les algorithmes eux-mêmes. De l'IA générative aux bibliothèques complètes de modèles, ce quadrant reflète la variété des besoins algorithmiques auxquels nos équipes sont confrontées, couvrant des modèles de vision par ordinateur, des outils d'analyse d'impact sur des séries temporelles, ou des technologies d'interprétabilité.



ADOPT
1/7 blips

12 DINOv2
comme modèle d'embedding d'image

Un besoin assez récurrent en computer vision est d'obtenir de **bonnes représentations vectorielles** (embeddings) d'images pouvant être utilisés pour des tâches spécifiques en aval (comme le clustering, la classification, la détection, la segmentation, etc.). Par exemple, une classification simple peut être composé des étapes suivantes:

1. Calcul de représentation d'un ensemble d'images
2. Stockage et indexation dans **une base de données vectorielle** (p.20)
3. **Détermination** de la classe grâce à un algorithme simple comme k-NN ou un modèle linéaire

Il n'est pas toujours possible de construire un modèle pour calculer de bonne représentation, pour un problème donné pour des raisons de **disponibilité de données d'entraînement** : images annotées non disponibles, chères à produire, ou inexistantes en début de projet. Il est alors intéressant de considérer des modèles génériques pré- entrainés sur de vaste corpus de donnée généraliste. Pendant longtemps, notre référentiel à Sicara furent des CNN (typiquement et dans l'ordre chronologique: VGG, ResNet, EfficientNet) pré-entrainés sur imagenet-1K (~1.2M images).

Depuis 5 ans, plusieurs innovations ont changé la donne :

- le perfectionnement de techniques d'entrainements dites **"self-supervized"** permettant de s'affranchir d'annotation pendant l'entrainement et donc de s'appuyer sur des bases d'images beaucoup plus grandes (e.g., 1.2B images pour DinoV2)
- l'apparition des **transformers**, une nouvelle architecture basée sur les mécanismes d'attention (*Attention is all you need*, 2017) d'abord en NLP (*BERT*) puis en vision (*An Image is Worth 16x16 Words*, 2020) avec les modèles ViT (*Vision Transformers*) qui atteignent et dépassent les CNNs en terme de performance, de flexibilité et de scaling à de grand jeux de donnée.

DinoV2 (2023) est un algorithme

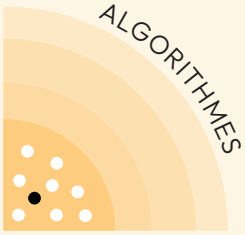
proposé par Meta qui est une évolution de son prédécesseur Dino (2021) qui combine et bénéficie de ces deux approches. Ceux qui s'interessent à son fonctionnement pourront regarder l'excellente analyse de Yannick Kilcher sur Youtube à ce sujet à ce sujet. Outre l'excellence académique du papier, Meta a rendu public les poids des modèles pré-entrainées présentant de nombreux avantages :

- la licence Apache 2.0 permet un usage commercial;
- les poids sont disponibles sur torch hub;
- plusieurs tailles de modèle sont disponibles, entre 21M et 1.1B de paramètres;
- les embeddings obtenus par ces modèles permettent de construire **sans fine-tuning** des classifieurs très performants (83.5% sur imagenet 1K en kNN, 86.7% sur image-net avec une couche linéaire);
- les mécanismes d'attention des ViT permettent une amélioration de l'explicabilité du modèle en permettant de déterminer les pixels de l'image entrant le plus en jeux pour la sortie.

NOTRE POINT DE VUE

Nous recommandons l'usage des modèles DinoV2 (architecture et poids) comme modèle d'embedding sur étagère sans ré-entrainement.

Le fine-tuning ou l'entrainement complet avec DinoV2 est reservé à un public averti et nécessite à la fois une grande puissance de calcul et un gros volume de donnée.



13 GPT-4

ADOPT
2/7 blips

GPT-4, le Large Language Model (LLM) le plus performant actuellement d'OpenAI, est un **modèle de génération de texte auto-régressif développé par OpenAI**. Il est probablement constitué d'une "Mixture of Experts" de plusieurs modèles de plus de 200 milliards de paramètres chacun (comme confirmé par plusieurs leaks). Pouvant aussi accepter des images en entrée, GPT-4 **résout la plupart des tâches de traitement de texte ou d'image** (traduction, synthèse, compréhension, rédaction, etc. - comme on peut s'en rendre compte en utilisant ChatGPT Plus). GPT-4 trouve son utilité dans divers domaines tels que la récupération d'informations textuelles, l'automatisation des flux de travail de support client, et le traitement automatisé de documents. Sur la quasi totalité des tâches, GPT-4 surpasse très largement les alternatives open-source et dépasse nettement les autres modèles propriétaires (Claude, PaIM, etc.).

Le **Function Calling**, intégré à GPT-4, permet d'utiliser des «fonctions» prédéfinies, ou des schémas, dans les appels de modèles. Cela **facilite la génération de sorties structurées**, plutôt que de se limiter au texte brut. Les Fonctions permettent aussi d'interagir avec des outils externes tels que l'exécution de code, l'accès à des API ou l'exécution de commandes shell.

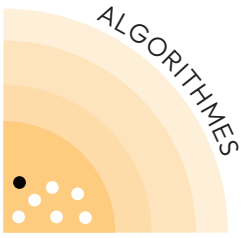
Récemment OpenAI a sorti **GPT-4 Turbo** qui est une itération sur GPT-4 ayant des **coûts ~3x plus faibles**, un temps d'inférence plus rapide et une fenêtre de contexte maximale à **128k tokens** (~40k mots). Il reste relativement lent à inférer - par rapport à GPT-3.5 par exemple, qui est plus adapté pour du temps réel - et cher (~$0.1 pour un appel avec 4k tokens en entrée et 2k tokens en sortie).

Enfin, attention à la fiabilité de la version Turbo, toujours en bêta, dont la précision de la réponse peut s'avérer moins fiable. Dans un contexte ou la qualité de l'output est particulièrement critique, nous recommandons de rester sur la version stable de GPT-4 pour le moment.

NOTRE POINT DE VUE

GPT-4 bat tous les modèles accessibles au public et est particulièrement peu susceptible à l'hallucination. L'intégration du **Function Calling** et OpenAI Assitants API et la récente réduction des coûts avec GPT-4 Turbo **renforcent sa position de leader.**

Si la réduction des coûts de Run ou de la latence est plus importante que la performance brute, alors préférez plutôt GPT-3.5 ou un autre "petit" LLM. Pour des contraintes de sécurité ou de contrôle particulières, il existe aussi les LLM Open Source (p.42).



ADOPT
3/7 blips

14 PyTorch

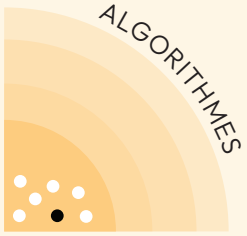
La démocratisation du Machine Learning dans les années 2010 est imputable à plusieurs facteurs, l'un d'entre eux étant l'apparition de packages open-source de calcul de tenseurs et de gradients, comme PyTorch en 2016 (ou TensorFlow en 2015).

Conçu par MetaAI et initialement orienté vers la recherche, il a étendu son influence à l'industrie et permet la **configuration et l'entraînement de modèles de Machine Learning** (notamment de réseaux de neurones), ainsi que le traitement de données qui est essentiel à l'entraînement. PyTorch permet de **détailler précisément les boucles d'entraînement des modèles** tout en gardant une syntaxe relativement facile à prendre en main, et il **existe des sur-couches** telles que PyTorch Lightning qui permettent de simplifier encore la syntaxe et le déploiement.

Un avantage significatif de PyTorch par rapport à ses concurrents réside dans la **richesse des modèles disponibles au sein de la communauté.** Sur des plateformes telles que Papers With Code et HuggingFace, la majorité des modèles sont implémentés en PyTorch, en comparaison avec une présence minoritaire de TensorFlow, par exemple. De plus, PyTorch **offre une bonne rétrocompatibilité**, minimisant les risques lors des mises à jour de versions, ce qui est crucial pour des applications en production.

NOTRE POINT DE VUE

Nous recommandons vivement l'utilisation de PyTorch, principalement pour sa flexibilité et la richesse de sa communauté. Attention aux cas d'usage d'IA embarquée car PyTorch Mobile est encore en Beta. Malgré cela, PyTorch demeure un choix solide pour une variété de cas d'usages, offrant une package complet pour le développement et l'entraînement de modèles de Machine Learning.



ADOPT
4/7 blips

15 Retrieval Augmented Generation

Les Large Language Models (LLMs) ont gagné une grande popularité grâce à leurs compétences généralistes en zero-shot et leur comportement comme une base de connaissances universelle interactive. Or, cette connaissance **ne comprend que des données publiques et s'arrêtent à la date d'entrainement du modèle**.

La Retrieval Augmented Generation (RAG) permet de complémenter ce savoir d'une base de connaissances externe (Notion, Confluence, PDFs, documentation interne, etc.). On peut alors **interroger cette base via du langage naturel**.

L'algorithme fonctionne en deux étapes :

1. **Retrieval :** récupération des documents les **plus proches** de la question en comparant une représentation (i.e. un embedding vectoriel) de la requête à des représentations des documents qui sont stockées dans une base de données (voir Recherche vectorielle avec des Bases de Données standards et Bases de Données vectorielles dédiées (p.20))
2. **Generation :** utilisation d'un Large Language Model **(LLM) pour générer une réponse** à la question à partir des documents récupérés.

Cette méthode permet de s'adapter à sa propre donnée, sans passer par du fine-tuning, ce qui a plusieurs avantages :

- **Éviter des coûts** de GPUs qui peuvent être importants avec des LLMs.
- **Renforcer l'explicabilité :** on sait exactement sur quelle donnée se base la réponse. Cela permet aussi de renvoyer ces sources aux utilisateurs qui ne souhaitent pas attendre le temps de génération du LLM.
- Permettre de s'adapter continuellement à **l'évolution de la base de connaissance**, sans avoir à entraîner à nouveau le modèle.

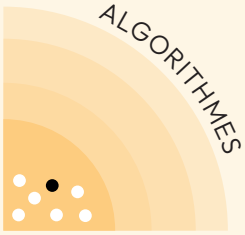
- **Améliorer la sécurité**, en limitant l'accès aux données sensibles aux personnes non autorisées (via l'utilisation des metadatas dans la base de données vectorielle).
- **Limiter les hallucinations du modèle**, en demandant au LLM de s'appuyer sur les documents du prompt.

Cependant, un RAG est un système plus complexe à gérer qu'un modèle unique (si on opte pour le LLM Fine-Tuning supervisé sur des questions / réponses (p.40)) et plus cher à l'inférence (puisque le prompt sera relativement large, pour contenir les documents). Aussi, sans fine-tuning, le modèle d'embedding et le LLM **ne s'adapteront pas au domaine spécifique** de la base de données de connaissances, ce qui rendra la compréhension de certains jargons plus compliquée. Enfin, le Retrieval est souvent le **bottleneck de précision** : si cette étape échoue à trouver de l'information pertinente, le LLM (qui aurait pu répondre correctement dans le cas du fine-tuning) ne pourra pas fournir de réponse pertinente.

NOTRE POINT DE VUE

Malgré ces points, les **RAGs sont généralement à préférer au fine-tuning** lorsqu'un **contrôle sur la donnée** utilisée pour générer la réponse est nécessaire (e.g. pour la renvoyer à l'utilisateur ou la filtrer pour des raisons de sécurité) ou lorsque la **donnée évolue fréquemment** (e.g. une documentation qui évolue au quotidien).

Mettre une V1 de RAG en place pour tester la valeur produit est très accessible, même sans compétence de code via les GPTs de ChatGPT+, l'API Assistants d'OpenAI ou des boiler plates ou frameworks comme LangChain (p.49). Cependant, maximiser la valeur du produit nécessite généralement des itérations spécifiques.



16 SHAP

ADOPT
5/7 blips

Les modèles de Machine Learning complexes sont considérés comme des boites noires peu interprétables. Pour donner confiance dans ces modèles, des méthodes d'explicabilité sont utilisées. **SHAP (SHapley Additive exPlanations)** est une méthode qui en fait partie et qui fournit une vision transparente et compréhensible de la manière dont les prédictions sont faites, **quel que soit le type de modèle utilisé**. Elle est pertinente pour les cas où les données d'entrée sont structurées.

Avant l'avènement de SHAP, l'interprétation des modèles ML complexes reposait largement sur des méthodes **plus simplistes**, telles que la **feature importance**. SHAP, en revanche, utilise **la théorie des jeux coopératifs** pour attribuer à chaque caractéristique sa contribution locale à la prédiction.

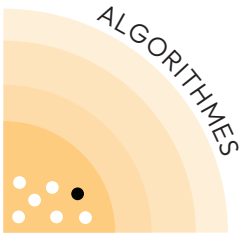
SHAP a l'avantage majeur d'être local par nature et permet donc d'analyser **la contribution de chaque variable en un point unique ou sur un cluster de points de données précis**. Les valeurs de SHAP forment elles-mêmes des distributions statistiques qu'on peut analyser et visualiser pour aller plus loin qu'un simple bar chart.

La principale limitation de SHAP est sa **complexité de calcul**, particulièrement pour les modèles avec beaucoup de paramètres ou sur des jeux de données massifs.

NOTRE POINT DE VUE

Nous appliquons la méthode SHAP systématiquement sur nos projets où **l'explicabilité des décisions prises par les modèles est indispensable,** e.g. dans les secteurs réglementés comme la finance ou la santé. Nous l'utilisons également pour analyser finement nos modèles dans le but de les améliorer, lorsque les données d'entrée sont structurées.

Nous recommandons donc l'adoption de SHAP dans ces applications.



17 SpaCy

ADOPT
6/7 blips

SpaCy est une **librairie open-source de traitement du langage naturel (NLP)** en Python, conçue pour une utilisation en production. Son approche privilégie l'efficacité, offrant des modèles en mode **«boîte noire» pour une mise en production rapide et des résultats fiables.** Contrairement à d'autres outils comme NLTK ou Stanford NLP, orientés vers la recherche, SpaCy vise à fournir **des solutions concrètes, avec une bonne performance et une grande variété de fonctionnalités NLP**, de la reconnaissance d'entités nommées à la classification de texte.

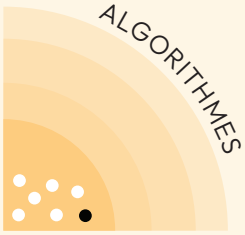
La simplicité d'utilisation de SpaCy est parmi ses atouts majeurs. La possibilité d'entraîner des modèles rapidement à partir **d'un fichier de configuration**, ainsi que **l'intégration des commandes CLI en tant que fonctions Python**, facilitent l'intégration de SpaCy dans divers environnements de développement. Cette facilité d'adaptation, combinée à son **architecture orientée objet,** rend la navigation et l'utilisation de la librairie intuitive, même pour ceux qui sont relativement nouveaux dans le domaine du NLP. En outre, SpaCy supporte nativement tous les modèles disponibles via la librairie Transformers de HuggingFace et des multiples LLM propriétaires via des APIs telles que OpenAI mais aussi des LLM open-source obtenus via HuggingFace (finetuné ou pas).

Cependant, il est important de noter que SpaCy, tout en permettant des itérations sur les données d'entraînement et les hyperparamètres des modèles, offre **une marge de manœuvre limitée pour des modifications en profondeur des modèles** eux-mêmes. Cette caractéristique souligne la philosophie de SpaCy : obtenir des résultats rapides et fiables, tout en simplifiant le processus de développement.

NOTRE POINT DE VUE

SpaCy est un outil historique et représente une valeur sûr. C'est pourquoi nous recommandons fortement l'adoption de SpaCy pour les projets NLP classiques (hors LLM) qui ont des contraintes de robustesse ou de performance.

Nous n'avons pas eu l'opportunité d'utiliser SpaCy-LLM en production donc ne partageons pas de recommandation concernant son utilisation.



18 YOLO

ADOPT
7/7 blips

Les **modèles YOLO (You Only Look Once)** sont des algorithmes de **détection d'objets en temps réel**.

Le premier modèle YOLO, sorti en 2016, se distingue d'autre modèles de détection d'objets (comme le R-CNN/Fast R-CNN) notamment par sa vitesse **d'inférence**. En effet, les modèles jusqu'alors impliquaient généralement des **étapes séparées** pour **générer des propositions de régions** et **les classifier**, alors que les modèles YOLO intègrent ces deux étapes en un seul réseau de neurone dans lequel on ne passe qu'une fois (ce qui lui a valu son nom).

Au fil des années, plusieurs versions de YOLO sont sorties, améliorant les performances : YOLOv4 dépasse largement en 2020 son concurrent Faster RCNN-FPN+ en termes de vitesse (5 fois plus rapide) et de précision (9.4 pts de Box AP en plus sur le benchmark MS COCO).

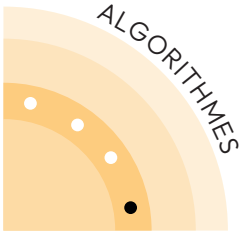
Après YOLOv4, l'entreprise privée **Ultralytics** a repris le flambeau, et passé les nouvelles itérations sous **licence AGPL-3.0**, ce qui rend plus difficile son utilisation à des fins commerciales. Ces versions améliorent légèrement la performance (de quelques points) mais offrent surtout des implémentations avec une bonne qualité de code, ce qui les rend plus facilement exploitables. Aujourd'hui, les dernières versions de YOLO restent l'état de l'art sur des benchmark de détection d'objets en temps réels, et se sont généralisés à des problèmes connexes comme la **segmentation**, le **tracking d'objets** ou la **pose estimation**.

NOTRE POINT DE VUE

Les modèles YOLO sont **un choix robuste** pour les applications nécessitant une détection d'objets en temps réel par leur capacité à fournir des détections **rapides et précises**. Il existe cependant des modèles plus lents, mais plus précis.

Nous avons utilisé YOLOv4 sur de nombreux cas d'application, y compris de l'embarqué (compressé en **tflite**). Bien que la **qualité du code** de ces modèles open source a parfois rendu complexe leur entrainement, cela n'a pas été bloquant.

Nous ne recommendons pas les versions d'Ultralytics (v5 et plus) pour des raisons de license.



19 Boruta

TRIAL
1/4 blips

Lorsqu'on entraîne un modèle de Machine Learning sur des données structurées, un écueil classique est de se retrouver avec un grand nombre de ***features* qui n'améliorent pas le modèle** voire le détériorent. Cela dégrade la qualité, allonge inutilement **le temps d'entraînement et d'inférence**, et **complexifie les analyses et les itérations**.

Boruta est une méthode de **sélection de *features*** pour les modèles de Machine Learning, conçue pour identifier celles qui sont **les plus utiles** pour la prédiction, éliminant ainsi **le bruit et la redondance**.
Le principe :

1. Des ***"shadow features"*** aléatoires sont créées à partir des vraies features en mélangeant aléatoirement leurs valeurs.
2. Un modèle de ***random forest*** est entraîné sur l'ensemble des features, qui sont ensuite classées en fonction de leur ***feature importance*** au sein du modèle. Boruta ne retient alors que les features statistiquement plus importantes que les shadow features

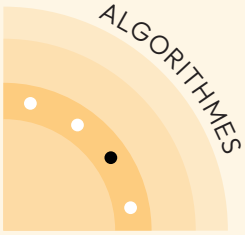
Il existe d'autres méthodes de sélection de features, notamment des méthodes statistiques classiques comme l'analyse en composantes principales (PCA) ou la brique de feature selection intégrée à **scikit-learn**.

Boruta se distingue par sa capacité à capturer l'importance de toutes les variables, y compris celles qui **interagissent de manière complexe** (e.g. haute non-linéarité) avec d'autres. De plus, la méthode **Boruta-Shap** (p.37) permet d'utiliser les SHAP values à la place des features importances pour classer les features et obtenir des résultats plus fiables.

L'algorithme Boruta peut cependant être **coûteux en calcul**, surtout avec de très grands ensembles de données et de variables, et nécessite **d'optimiser certains hyper-paramètres** comme les seuils de signification pour le test d'importance.

NOTRE POINT DE VUE

Nous recommandons donc d'utiliser Boruta principalement comme **un outil en première approche**, afin d'identifier rapidement les features qui sont les plus utiles. En revanche, il est toujours préférable de **tester le modèle avec et sans les features**, et de se baser sur les **performances réelles** pour faire un choix quant à l'exclusion ou non des features.



TRIAL
2/4 blips

20 Causal Impact

Un problème classique dans les modèles de séries temporelles est de mesurer l'impact d'une intervention sur la série. Par exemple, mesurer l'impact d'une campagne marketing sur le volume de ventes. Les approches classiques (A/B Testing, tests randomisés en double aveugle) nécessitent un groupe de contrôle sur lequel aucune intervention n'est faite comme point de comparaison. Il n'est pas toujours simple d'identifier et isoler un groupe de contrôle ayant un comportement similaire.

Causal Impact est une **méthode de mesure d'impact causal d'un événement ("intervention") sur une série temporelle ne nécessitant pas de groupe de contrôle**. Initialement publiée par une équipe de recherche de Google, cette approche utilise une méthode d'inférence causale (par défaut un modèle "Bayesian structural time-series") pour prédire **un contrôle synthétique**. Ce dernier va servir de comparaison avec les données réellement observées post-intervention. Il représente l'évolution prédite de la série temporelle après la date d'intervention, si cette dernière n'avait pas eu lieu. Le modèle nécessite de disposer **de séries temporelles qui ne doivent pas être impactées par l'intervention. Ces features pré-intervention** sont utilisées pour entraîner le modèle, qui sert ainsi à prédire le contrôle synthétique après intervention.

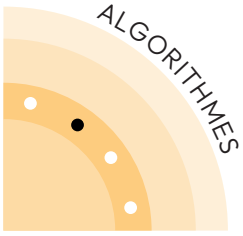
On peut **mesurer la précision du modèle** grâce à la sortie du modèle bayésien, une distribution de probabilité. On peut alors en déduire **la probabilité que cet impact soit dû au hasard.**

Avec la bonne combinaison de modèle et de features prédictives, il est possible d'observer l'impact local de l'intervention isolé d'autres variables, mais **cette combinaison peut être difficile à trouver selon le problème**. Pour faire le meilleur choix, il est conseillé de calculer une mesure d'impact en considérant uniquement la donnée pré-intervention. Dans ce cas, une bonne combinaison de modèle et de features devrait toujours donner un impact nul.

On peut ensuite conserver cette combinaison pour mesurer l'impact de l'intervention sur la courbe à l'étude.

NOTRE POINT DE VUE

Nous recommandons **d'essayer Causal Impact** dans un contexte de **mesure d'impact sur des séries temporelles sans groupe contrôle**, car la méthode apporte une **robustesse statistique** à une mesure de changement post-intervention. Comme le package original de Google est seulement disponible en R, nous conseillons la librairie tfcausalimpact de Willian Fuks pour une implémentation simplifiée en Python.



TRIAL
3/4 blips

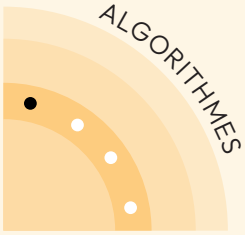
21 OpenAI Assistants API

Les LLMs se base sur le contenu acquis pendant son entraînement pour nous répondre. Or nous cherchons à avoir des réponses nécessitant des connaissances précises, c'est pourquoi, par exemple, nous implémentons des RAG (Retrieval Augmented Generation (p.32)). Cela est coûteux de mettre en place les systèmes nécessaire (comme LangChain pour l'orchestration, Qdrant pour la base de données vectorielle), alors qu'elles sont très communes entre les cas d'usages. L'Assistants API d'OpenAI offre solution clé en main (reposant sur GPT-4 (p.30) ou GPT-3.5) pour accélérer et gérer ces cas d'usages où un LLM intéragit avec des outils externes (base de connaissance, APIs externes, environnement de calcul, etc.) et répondre aux mêmes cas d'usages en quelques clics.

Cette API, actuellement en phase beta, prend en charge trois types d'outils : **Code Interpreter, Retrieval** (utilisation de connaissances externes au modèle) et Function calling (appels de fonction déclenchés par le modèle). L'interaction avec l'API repose sur des objets abstraits (l'Assistants, le Thread, le Message et le Run), laissant la gestion de la fenêtre de contexte et de l'historique de chat au backend OpenAI (du point de vue de l'utilisateur, il n'y a pas de limite sur la taille de l'historique d'un Thread). **Attention néanmoins au coût** des fichiers de la base de connaissances : c'est en fonction de l'espace disque utilisé ($0.20/GB/jour). Donc, si on stocke autre chose que du texte brut, cela peut représenter un coût significatif. De plus, le **lock-in et le manque de maitrise sur le comportement du modèle** (ingestion, chunking, logique de recherche, etc.), sont un revers inévitable d'une API haut-niveau managée.

NOTRE POINT DE VUE

Nous avons l'habitude de développer nous même les solutions comme les RAGs, mais nous réfléchissons de plus en plus à l'utiliser l'Assistants API pour créer des assistants LLM qui accède à une base de connaissance, des outils externes et un environnement Python. Nous attendons d'avoir plus de recul sur son utilisation en production à grande échelle pour avoir une recommendation définitive.



TRIAL
4/4 blips

22 LLM Fine-Tuning

supervisé sur des questions / réponses

Les Large Language Models (LLMs) à l'état de l'art sont des modèles généralistes entrainés sur de la donnée publique. L'enjeu **d'adapter ces modèles à des données propriétaires** est majeur.

Fine-tuner un LLM de manière supervisée consiste à continuer à entraîner un modèle pré-entraîné (e.g. Mixtral, Llama, GPT3.5, ...) avec un set de questions / réponses pour adapter les réponses du modèle et le rendre plus spécifique. **Cela s'avère notamment utile lorsque le prompt engineering seul ne suffit pas**, par exemple pour guider le format de sortie souhaité ou pour incorporer un concept spécifique, par exemple résumer des textes avec du vocabulaire propre à une entreprise.

Le fine-tuning de LLM a été énormément simplifié au cours de 2023: Des techniques comme QLora (**Quantization des poids et Low-Rank Adaption** - consiste à n'adapter qu'un sous-ensemble des poids) permettent d'effectuer ce fine tuning de manière efficiente et donc économique (quelques dizaines d'euros pour un modèle de 7B paramètres). Seulement quelques milliers d'exemples (questions/réponses) de qualité sont nécessaires pour obtenir de bons résultats.

Cependant, le fine tuning est une opération **coûteuse en temps**, nécessitant la préparation d'un **dataset adapté** (qu'on pourra générer via GPT-4 (p.30) et plusieurs **itérations pour obtenir des résultats optimaux**. De plus, si fine-tuné sur une tâche spécifique, le modèle perdra une partie de ses capacités généralistes.

Il y a **plusieurs alternatives**, notamment :

- **Des méthodes sans réentraînement.** Ces méthodes nécessitent souvent des prompts bien plus gros (pour contenir les informations sur lesquelles la réponse doit s'appuyer) et sont donc plus chères et longues à l'inférence. Ces méthodes peuvent être limitées (par ex comprendre du jargon spécifique). Ces méthodes sont :
  - Le **Retrieval Augmented Generation** (p.32) qui permet aussi de s'adapter à de la donnée spécifique et permet notamment d'avoir un meilleur contrôle sur la donnée et d'éviter le complexités du

fine-tuning supervisé (création de dataset de question/réponse, ...).

- Le **prompt engineering** pour préciser le contenu de la tâche. Du few-shot (i.e. montrer dans le prompt quelques exemple de questions/réponses) peut-être utilisé pour guider plus précisément le modèle comparé à une simple explication.

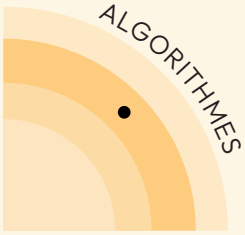
- **Le fine-tuning non supervisé :** cela consiste à apprendre au modèle à compléter des textes. Le modèle perdra probablement ses capacités de question/réponse en conséquence. Cette méthode n'est donc que recommandée pour des cas d'utilisations de complétions (par ex un copilote de rédaction ou de code).

- **Reinforcement-Learning from Human Feedback (RLHF) :** cela consiste à utiliser du feedback humain pour entrainer un reward model à noter l'output de notre LLM et une méthode appelée PPO pour fine-tuner le LLM à partir du reward model. Cependant cette méthode est très couteuse en terme de collection de human feedback et d'entrainement (>$10M en ordre de grandeur). De plus, elle est très instable et nécessite donc de nombreuses itérations.

- **Direct Preference Optimization (DPO) :** cette méthode consiste à fine-tuner un modèle en fournissant des exemples de bonnes et mauvaises réponses pour chaque question (généralement obtenues via du human feedback). Bien que plus simple, plus stable et plus efficiente que le RLHF, elle reste beaucoup plus couteuse que le fine tuning supervisé sur de simples questions/réponses.

NOTRE POINT DE VUE

Privilégiez le prompt engineering d'un modèle puissant tel que GPT-4 au fine-tuning, notamment en phase de POC. Quand de claires limitations sont identifiées et qu'il devient nécessaire de **s'adapter à de la donnée spécifique**, faites le choix entre Retrieval Augmented Generation (p.32) et fine-tuning supervisé sur des questions/réponses. Dès la phase de POC, **mettez votre outil à disposition d'utilisateurs tests** et utilisez des outils comme LangSmith (p.52) pour récolter les questions que posent les utilisateurs : une donnée précieuse pour préparer un dataset de fine-tuning.



ASSESS
1 blip

23 LLM Open Source

Les modèles de langage à l'état de l'art sont historiquement open-source (jusqu'à BERT et GPT-2 en 2018/19), mais cela a changé avec GPT-3 (2020) dont les poids n'ont pas été publiés par OpenAI.

Pour le moment, **les modèles closed-source** (accessibles par API), notamment GPT- 4, **restent bien plus performants** que les alternatives open-source. Avec la sortie de Mixtral (8x7B) on atteint une performance légèrement supérieure à GPT 3.5, relançant le débat sur si les modèles open-source atteindront l'état de l'art dans les prochaines années.

Les modèles open-source offrent cependant **plus de contrôle**, ce qui a plusieurs avantages :

- Disponibilité du modèle: avoir son propre modèle permet d'avoir le contrôle sur le serving et la charge, évitant latences et/ou downtimes sont parfois observés sur les APIs d'OpenAI;
- Flexibilité d'utilisation des tokens en sortie du modèle (e.g. avec Guidance (p.53));
- Efficience du modèle en quantité de calcul et en consommation énergétique en utilisant un petit modèle spécialisé plutôt qu'un gros modèle généraliste;
- Sécurité des données et propriété intellectuelle du modèle dans le cas de fine- tuning

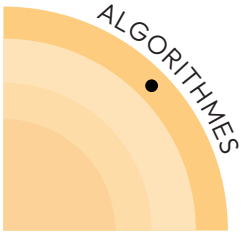
Les coûts sont aussi à prendre en compte : se servir **d'un LLM open-source est relativement cher**, comparé à l'utilisation d'un modèle API où l'on paie quelques centimes par call, surtout si l'utilisation est peu intense. Si la charge est suffisamment élevée la tendance peut s'inverser. De plus, des outils comme vLLM (p.51) viennent faciliter le serving et diminuer ses coûts.

Pour en savoir plus :

NOTRE POINT DE VUE

Pour les organisations ayant les ressources et l'expertise nécessaires, l'adoption d'un LLM open source peut être une stratégie viable, offrant à la fois flexibilité et contrôle sur les modèles.

Pour les applications nécessitant la plus haute fiabilité et les meilleures performances, GPT-4 (p.30) reste actuellement le meilleur choix. Plus généralement, pour un POC démontrant valeur produit et faisabilité technique, nous recommandons **d'initier les projets avec un modèle à l'état de l'art**, avant d'envisager des alternatives open-source.



HOLD
1 blip

24 Few-shot learning classique

Un problème classique pour entraîner des modèles de machine learning est de n'avoir qu'un faible nombre d'exemples annotés par classe. Le plus simple est d'annoter suffisamment de données, mais ce n'est pas toujours possible, pour des raisons de coût, de disponibilité de la data, ou lorsque les toutes les classes ne sont pas connues à l'avance.

Le Few-Shot Learning (FSL) est la branche du ML qui vise à apprendre de nouvelles tâches avec un petit nombre d'exemples d'entraînement. Quelques techniques classiques :

- l'apprentissage de bonne représentation avec du metric learning : apprendre un espace de représentation dans lequel les exemples d'une même classe sont proches entre-eux et éloignés de ceux des autres classes. Par exemple avec des réseaux siamois entrainés avec une fonction de coût contrastive, en montrant des exemples positifs (même classe) et négatifs (autre classe).
- le meta-learning, une approche qui consiste à «apprendre à apprendre», c'est à dire apprendre à un algorithme à s'adapter rapidement à de nouvelles tâches avec peu d'exemples. Par exemple MAML où le modèle apprend une initialisation de poids rapide à fine-tuner pour de nouvelles tâches.

Ces techniques ont été remises en question : notamment l'article "A closer look at few-shot classification" montre que des performances au niveau de l'état de l'art peuvent être obtenues par un pré-entrainement supervisé de représentation puis un fine-tuning. De plus, l'arrivée d'une nouvelle génération de modèle à la suite de la révolution des transformers à partir de 2019 a permis l'émergence de modèles proposant des représentations "universelles" d'une qualité suffisante pour la plupart des uses-cases."universelles" d'une qualité suffisante pour la plupart des uses-cases.

NOTRE POINT DE VUE

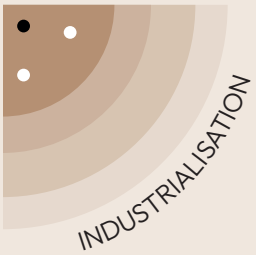
Nous avons régulièrement utilisé des approches few-shot learning dans nos projets, dont les réseaux siamois et les entrainements contrastifs.

Cependant l'expressivité des représentations proposées par les modèles de fondation s'est améliorée : le travail sur la fin de la chaîne algorithmique (ex. : méthode de choix des représentants par classe ou métrique utilisée) devient suffisant. Nous recommandons donc de ne plus commencer par des approches visant à optimiser les représentations pour le few-shot.

Industrialisation

10 BLIPS | 3 ADOPT | 4 TRIAL | 2 ASSESS | 1 HOLD

À Sicara, nous visons plus loin que la création de Proof Of Concept de Data Science, nous cherchons à atteindre la valeur utilisateur et ce n'est possible qu'en exposant nos modèles. Leur exploitation en production passe par des technologies qu'on assimile souvent au Data Engineering et au DevOps. En particulier, nous sommes convaincus que cela doit passer par une attention particulière sur les technologies de monitoring, de gestion de l'infrastructure et de l'orchestration des flux de données.



25 Airflow

ADOPT
1/3 blips

Gérer des flux de données fait partie des tâches du data scientist (préparation de données, lancement de construction de modèle). Or cette gestion est devenue complexe, montrant les limites des outils classiques d'orchestration comme CRON. En 2014, Airbnb a créé Airflow (p.46) en réponse à cette complexité croissante.

Airflow est une librairie Python open-source pour **l'orchestration de tâches** qui permet la création, le déploiement et le suivi de workflows.

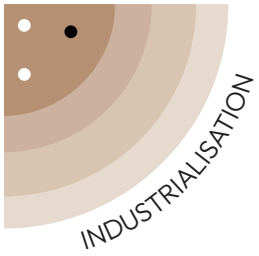
Airflow **modélise les flux de données** complexes sous forme de graphe de tâches. Un ordonnanceur **planifie l'exécution des tâches en fonction de leurs dépendances**. Une interface web offre une vue d'ensemble, et la multitude de types de tâche qu'il est possible d'exécuter offre une flexibilité manifeste. Toutes ces fonctionnalités contribuent à **simplifier l'automatisation des processus de traitement des données**, faisant d'Airflow une technologie très utilisée aujourd'hui.

En data science, on pourrait être tenté de **mettre en place un enchaînement d'étapes** tel que pré-traiter un dataset, lancer l'entraînement d'un modèle, évaluer ses performances, et envoyer les résultats avec des scripts Bash, complexes à maintenir et utiliser. **L'utilisation d'Airflow** pour un tel cas d'usage offre **une meilleure maintenabilité par les outils de monitoring et de gestion des erreurs disponibles**.

Il existe plusieurs alternatives à Airflow pour des problématiques plus spécifiques. Par exemple, **Dagster** permet de communiquer de la donnée entre deux tâches sans passer par un service de sauvegarde de données externes (table, bucket). **Kubeflow Pipelines** est aussi une alternative intéressante, car elle vient avec des opérateurs pré-configurés pour le Machine Learning (un pour l'entraînement des modèles et un pour leur déploiement sur Kubernetes), mais son focus sur les problématiques de ML induit une **communauté plus réduite**. Enfin, **la librairie DVC** (p.14) offre une brique pour définir et exécuter des pipelines, très pratique dans un contexte d'expérimentation pour son intégration avec le tracking d'expériences, mais pas adaptée à une exécution en production.

NOTRE POINT DE VUE

Aujourd'hui, **nous recommandons Airflow** pour une orchestration robuste de tâches hétérogènes, incluant des pipelines de Machine Learning en production. Lors de l'itération sur le modèle pendant le développement, nous préférons des outils comme DVC qui apportent de meilleures features pour le suivi des expérimentations.



ADOPT
2/3 blips

26 Infrastructure as Code

Développer des solutions de Machine Learning nécessite de provisionner des resources (base de données, cluster de calculs...). **Traditionnellement, cela était fait manuellement**, créant un risque d'erreur humaine et ne permettant pas de redéployer rapidement une infrastructure

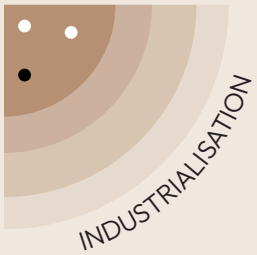
L'**Infrastructure as Code** (IaC) est une approche pour **créer et manager les ressources d'infrastructure** d'un projet. Définie par des fichiers, l'infrastructure est versionnée et automatisée. Cette pratique permet de diminuer le nombre d'erreur et offre aussi la possibilité de **répliquer** des environnements rapidement, de manière **cohérente** tout en pouvant les faire **évoluer** facilement.

L'Infrastructure as Code est une pratique assez standard sur les projets Web et Data Engineering mais elle est beaucoup moins répandue en Machine Learning. Définir au moyen de code les différents services de stockage de vos données, les environnements pour entraîner vos modèles, et l'infrastructure scalable pour la mise à disposition de vos modèles ou des environnements d'entraînements assure une **maîtrise** des composants, des coûts et surtout de leur **évolutivité et leur mise à l'échelle**.

Cette approche nécessite de maîtriser un outil comme **Terraform** et les bonnes pratiques associées. En effet, le code de l'infrastructure doit être **maintenue** avec la même **rigueur et la même qualité** que le code traditionnel.

NOTRE POINT DE VUE

Nous recommandons **l'adoption** de l'Infrastructure as Code (IaC) pour les projets de Machine Learning. Cette approche garantit une gestion d'infrastructure plus agile, scalable et efficiente, permettant des déploiements rapides et une meilleure cohérence. L'IaC renforce également la sécurité et la maintenance, essentielles dans des projets de ML.



27 Poetry

ADOPT
3/3 blips

Pour gérer les dépendances sur un projet Python, on utilise traditionnellement **Pip** via le requirements.txt ou **Conda** qui gèrent seulement les dépendances primaires. Cela crée des problèmes de compatibilité de dépendances, de versions différentes entre les différents environnements (dev, prod, etc...).

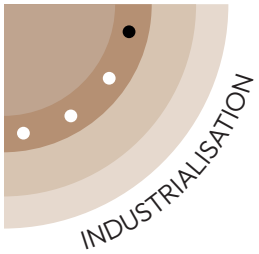
Poetry est un outil de gestion des dépendances et de packaging en Python qui permet de résoudre ces problèmes. Ses principales caractéristiques sont :

1. **Résolution robuste des dépendances :** Poetry utilise un algorithme robuste de résolution de dépendances pour éviter les conflits entre les bibliothèques. Cela évite ainsi des erreurs lors de l'installation et des résolutions manuelles de conflits.
2. **Verrouillage des versions entre les développeurs :** Poetry assure que tous les développeurs travaillant sur un projet utilisent exactement les mêmes versions des bibliothèques, éliminant les «pourtant ça marche sur ma machine» liés aux divergences de dépendances.
3. **Facilité d'utilisation :** Poetry offre une interface en ligne de commande intuitive et un fichier de configuration unique qui simplifie la gestion des dépendances et des configurations de projet. Cela rend Poetry accessible y compris pour des data scientists moins expérimentés en gestion de packages Python.

Poetry permet aussi la gestion des environnements virtuels Python mais cette fonctionnalité est limitée, ne proposant pas d'activation automatique du bon répertoire de dépendances. Nous recommandons plutôt d'utiliser Poetry en tandem avec **Pyenv** et son plugin **pyenv-virtualenv** pour améliorer encore l'expérience de développement.

NOTRE POINT DE VUE

Nous recommandons fortement Poetry et le considérons comme un outil essentiel pour la gestion moderne des dépendances en Python grâce à son approche de **résolution des dépendances**, combinée à la facilité de synchronisation des environnements de développement.



28 LangChain

TRIAL
1/4 blips

Les applications basées sur des **Large Language Models (LLM)** ont souvent de nombreuses briques fonctionnelles en commun. **LangChain** est un framework open source ayant pour but de simplifier leur mise en place et leur orchestration. Il offre une **interface haut niveau** permettant de définir la logique des applications en étant agnostique du LLM et/ou du vector store utilisé.

Malgré la promesse de LangChain de pouvoir facilement **passer d'un LLM à l'autre**, la migration entre différents LLM avec LangChain implique souvent **d'ajuster les prompts et les paramètres** pour maintenir une performance optimale de l'IA.

Actuellement en plein développement, LangChain présente encore quelques inconvénients :

- Une **documentation complexe** et parfois peu intuitive
- Les différents composants **manquent de stabilité** - un problème qui peut être atténué, avec l'isolation récente des abstractions clés dans le module **langchain-core**
- L'intrication de son code et l'utilisation de callbacks peuvent complexifier la navigation dans le code et le debugging.

D'autres frameworks LLM comme LlamaIndex ou Haystack existent et présentent des avantages et inconvénients similaires. Alternativement, il est possible de ne pas utiliser de framework et appeler les différentes briques (LLM, BDD vectorielle, ...) avec du code custom.

NOTRE POINT DE VUE

Nous recommandons l'utilisation de LangChain pour **prototyper un projet LLM**. Permettant d'abstraire l'appel aux LLMs et aux modèles d'embedding, il permet par exemple de développer une pipeline de RAG en quelques lignes de code. Au delà du prototype, rentre en compte le **trade-off** entre valeur ajoutée du framework (souvent en fonction de la formation des développeurs) et complexité apportée.

LangChain est enfin un **bon outil de formation** sur les LLM, permettant d'explorer différents types d'application, de modèles, de stratégies de prompt engineering ou de BDD vectorielles.

29 Qarnot

TRIAL
2/4 blips

Réduire sa consommation carbone devient un enjeux pour les entreprises. Or l**es entrainements de modèles de machine learning** sont de **grands émetteurs de gaz à effet de serre**.

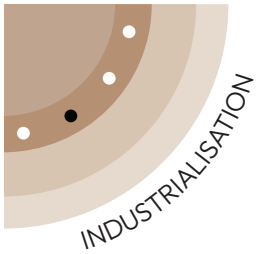
Qarnot propose de limiter ces émissions et fournit du **cloud computing "bas-carbone"** conçu pour les jobs de rendering graphiques et les entrainements de modèles de Deep Learning. Début 2024, Qarnot promet une réduction de 50% de l'empreinte carborne par rapport aux autres centres de données en France et de 90% par rapport à ceux aux États-Unis, grâce à une approche de décentralisation et à la réutilisation presque totale de la chaleur produite. **Pour un entrainement d'une heure** l'économie d'empreinte carbone est d'environ **1kg CO2eq** par rapport à des fournisseurs traditionnels comme AWS/GCP

Qarnot a cependant des limitations en comparaison à ces fournisseurs :

- Il y a un seul type de GPU disponible contrairement au catalogue proposé par AWS ou GCP qui en contiennent plusieurs dizaines.
- Les instances ne sont pas connectées à internet ce qui oblige à utiliser des **Buckets Qarnot d'input / output** (I/O) pour le transfert de données. Un SDK Python (open-source) est disponible pour le transfert de données et le déploiement de tâches sur Qarnot, cependant le transfert est relativement lent. La mutualisation permet de régler ce problème mais en apporte de nouveaux notamment sur **concurrence** des jobs. Par ailleurs, les possibilités de collaboration au niveau de l'équipe / l'organisation sont limitées, seule la facturation est commune..

NOTRE POINT DE VUE

Face à la crise climatique, **nous soutenons la démarche de Qarnot** de réduction d'empreinte carbone des entraînements. L'utilisation de buckets d'I/O, l'automatisation de tâches via Python et la gestion des workflows avec DVC ont été suffisantes pour contourner la plupart de ses limitations. Nous vous recommandons donc de tester l'outil par vous-même. Il est cependant prudent de **continuer à évaluer Qarnot dans des contextes plus variés** avant de l'utiliser à grande échelle.



30 vLLM

TRIAL
3/4 blips

Gérer **l'inférence** et le **serving de modèles** de Deep Learning de manière optimisée est une problématique complexe, et c'est d'autant plus le cas pour les **Large Language Models** (LLM), très gourmands en ressources du fait de leur taille.

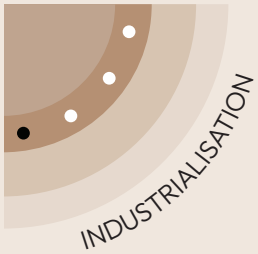
vLLM a été créé en 2023 pour permettre de **servir un LLM et d'optimiser son inférence**. Il permet de déployer un service dédié accessible par une **API** en configurant une image **Docker**. vLLM utilise un algorithme d'attention efficace, ***PagedAttention***, qui permet un **rendement jusqu'à 24x meilleur** qu'avec une approche classique. De plus, vLLM implémente de nombreuses fonctionnalités **facilitant l'utilisation des LLM en production**, comme le continuous batching, permettant de batcher les requêtes des différents appels reçus, afin d'utiliser efficacement les ressources mises à disposition. Cet outil encore très jeune gagne rapidement en popularité, la preuve étant ses 13k étoiles sur GitHub (début 2024) ou le fait que Mistral mette à disposition une image vLLM pour son modèle Mixtral.

Il existe plusieurs alternatives à vLLM :

- **llama.cpp :** cette librairie permet de faire tourner des LLMs en pur C/C++, ce qui a l'avantage de rendre possible l'inférence de modèles sur de petits CPUs. Cependant, cette librairie a été conçue comme un playground et n'est pas faite pour une utilisation en production.
- D'autres outils plus généralistes qui existaient avant vLLM comme Triton inference server ou Tensorflow serving permettent de servir tout type de modèle. Tout comme vLLM, ces outils sont configurables via Docker et implémentent du continous batching. Plus anciens, ils ont pu être testés et approuvés sur beaucoup de cas d'usage mais ne sont cependant pas aussi efficaces que vLLM pour le serving de LLMs.

NOTRE POINT DE VUE

Nous suivons cette technologie depuis sa sortie et l'avons utilisée pour faire tourner l'inférence de LLM sur des projets internes. Utilisez-la sans crainte dans un **contexte d'expérimentation** pour **éviter les coûts de hardware trop élevés** (notamment grâce à PagedAttention). Pour une **utilisation en production,** nous la recommandons aussi, mais gardez en tête que la technologie est **encore jeune** et utilisez-la avec précaution..



31 LangSmith

TRIAL
4/4 blips

Les spécificités des Large Language Models (LLM) se sont accompagnés de besoins spécifiques de **monitoring** et de collection de la donnée. **LangSmith**, développé par **LangChain**, propose de répondre à ces enjeux en loggant les différentes interactions avec les LLMs.

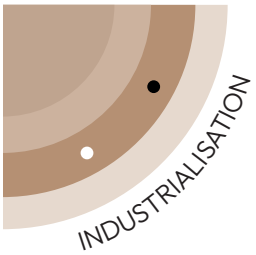
Il intègre de nombreux features autour de ce logging, comme la possibilité de modifier et réexécuter les prompts via une interface de **"playground"**, suivre les **performances et les coûts** ou créer des datasets à partir des logs.

Initialement développer pour s'intégrer à LangChain, il peut aussi être utilisé indépendamment via un sdk.

Cependant, LangSmith est encore en version beta et l'utilisateur n'a pas de contrôle sur les versions (l'outil est mis à jour automatiquement): des **instabilités** sont donc parfois notées (par exemple une régression sur la feature de playground). De plus, **la transition vers un modèle payant**, **potentiellement onéreux,** pourrait restreindre son accessibilité pour certains utilisateurs ou organisations. Face à ces limitations, **LangFuse** offrant des fonctionnalités similaires, **émerge comme une alternative open-source intéressante**.

NOTRE POINT DE VUE

Nous avons utilisé langsmith (via son sdk) sur un projet de **Retrieval Augmented Generation** (p.32). Son intégration s'est faite sans soucis et nous l'utiliserons sur de prochain projets (la question se posera de migrer vers Langfuse si il devient payant). Le **"playground"** a notamment permis d'intégrer le product owner et son savoir métier dans les itérations de prompt engineering. Nous vous recommandons donc d'utiliser langsmith, tout en prenant en compte les limitations spécifiées.



32 Guidance

ASSESS
1/2 blips

Avec l'avènement récent des Large Language Models (LLM), le besoin en outillage à augmenté pour intégrer de manière robuste ces solutions dans nos applications. Le framework open-source Guidance a été créé par **Microsoft** pour permettre un **templating complexe de prompts**. Il permet de s'interfacer avec **plusieurs LLM** (open-source ou non) en plus d'ajouter une boite à outils autour de l'inférence de ces modèles au niveau du token (grammaires, token healing, etc.)

Il se différencie du framework LangChain (dont la communauté est bien plus grande) par les points suivants :

- Une emphase particulière sur le contrôle de la sortie du modèle, c'est-à-dire la possibilité de le **contraindre** à une structure de sortie (ou "grammaire"). Néanmoins cette fonctionnalité clé n'est disponible **que pour les modèles open-source**, car les modèles disponibles via API ne fournissent pas les détails nécessaires pour appliquer ce post-traitement.
- Il est très facile de définir des fonctions pouvant être appelées par le modèle (à la manière du "Function Calling" d'OpenAI (voir OpenAI Assistants API (p.39))
- Le framework est beaucoup moins tourné vers l'utilisation de templates et de chaines pré-construites comme LangChain (la vision semble plus tournée vers le sur-mesure).

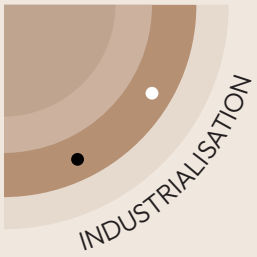
La formulation des prompts se veut très simple d'accès mais nous trouvons que le changement de syntaxe opéré récemment, utilisant maintenant une surcharge de l'opérateur d'addition, peut s'avérer assez dur à lire et à écrire.

Le repository GitHub de Guidance compte plusieurs notebooks d'exemples à suivre, mais vous en trouverez moins que pour un framework plus dominant comme LangChain.

NOTRE POINT DE VUE

Guidance est particulièrement puissant pour le templating et la génération de prompts. **Encore en version bêta**, sa communauté est encore réduite. Des changements majeurs et réguliers sont donc à attendre.

Pour des cas d'usage plus généraux, nous recommandons LangChain. Davantage clé en main, il offre des fonctionnalités plus développées pour la gestion de la mémoire ou les intégrations avec des vector stores.



ASSESS
2/2 blips

33 Plateforme de ML end-to-end

Jusqu'au milieu des années 2010, les projets de Machine Learning étaient une succession **d'opérations manuelles**. Les outils de **MLOps** qui sont progressivement apparus depuis ont permis de structurer et de fluidifier une partie croissante de ces tâches, jusqu'à des plateformes complètes intervenant sur **l'ensemble du cycle de vie des modèles.**

On peut citer **Databricks**, disponible depuis 2015, comme l'un des précurseurs de ces plateformes, ou les services de ML des **3 principaux fournisseurs cloud : Google VertexAI, Amazon Sagemaker** et **Azure ML**.

Celles-ci permettent une **mise en œuvre rapide** des principaux composants d'un projet : une **infrastructure** pour les **entraînements et prédictions, un model registry, l'orchestration de pipelines, le tracking d'expériences**, etc.

Sans rentrer dans le détail de chaque plateforme, on peut identifier plusieurs **problèmes-types communs** à celles-ci :

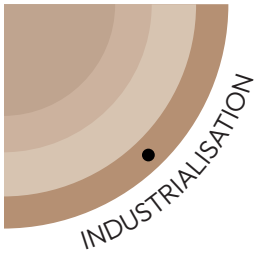
- **Coût des resources :** Il est **toujours plus élevé** qu'avec des solutions moins managées. Par exemple, un entraînement sur Vertex AI coûte ~15% plus cher que via Compute Engine.
- **Rigidité :** Les services de ML tout intégrés sont limités en terme de **personnalisation et d'intégration** avec des outils n'appartenant pas à leur écosystème. Par exemple, on pourra difficilement utiliser DVC pour versionner ses données ou lancer des entraînements bas-carbone sur Qarnot (p.50).
- ***Vendor lock-in*** : Une dépendance à un fournisseur spécifique peut s'avérer d'autant plus frustrante dans le domaine de l'IA, où les technologies évoluent très rapidement.

L'alternative principale est de **combiner des technologies spécifiques et moins managées**, comme DVC (p.14), Streamlit (p.19), Airflow (p.46), ce qui représente un investissement non négligeable en coût de mise en place.

NOTRE POINT DE VUE

Chez Sicara, notre stack par défaut n'utilise pas de plateforme ML end-to-end. Nous privilégions **la combinaison d'outils open-source** adaptés au besoin, minimisant les coûts et évitant le vendor lock-in. Sicarator (p.64), notre **générateur de projet ML open-source**, permet d'accélérer leur mise en place.

Une solution end-to-end reste préférable si l'on ne dispose pas **du temps ou des compétences nécessaires** pour une stack sur-mesure, ou pour une entreprise qui n'a prévu de développer que quelques projets de ML sur le **court terme**.



HOLD
1 blip

34 Dataiku

for industrialized tech teams

La création de pipelines de Data Science nécessite beaucoup de temps et plusieurs équipes pour traiter et rendre la donnée disponible. **Dataiku**, plateforme propriétaire, émerge comme **un accélérateur dans l'univers Data et Machine Learning** (ML), en s'appuyant sur **une interface low-code pour simplifier ces étapes**. Dataiku propose une boîte à outils assez large, interfacée à de nombreuses sources de données. La plateforme permet d'automatiser des pipelines **de transformation de données, des entrainements de modèles**, ainsi que **des déploiements de modèles en production**. Il est également possible de visualiser les données à travers des tableaux de bords personnalisés.

Bien que le no-code facilite la création de projets ML à court terme, cela crée également de nouveaux défis en terme de maintenabilité sur les projets sophistiqués. Par exemple, **la contribution de plusieurs équipes** sur la même pipeline est **difficile** : impossibilité de modifier au même moment, versionnage des changements au stade embryonnaire (rendant difficiles les revues de code, un standard chez Sicara). De plus, élargir l'accès à la plateforme est tentant car elle est intuitive, mais cela peut nécessiter **des coûts de licence importants**.

NOTRE POINT DE VUE

Dataiku est une excellente solution pour explorer puis déployer rapidement des use cases Data Science. Nous avons trouvé que la solution complexifie collaboration et maîtrise de la qualité pour des contextes d'industrialisation avec un fort besoin de collaboration et un bon bagage technique de l'équipe. Nous recommandons alors de se baser sur une stack modulaire avec un investissement plus élevé au départ mais une meilleure évolutivité et flexibilité.Streamlit, etc.) ce qui nécessite certes un investissement plus important au début du projet mais permet une meilleure évolutivité et flexibilité.

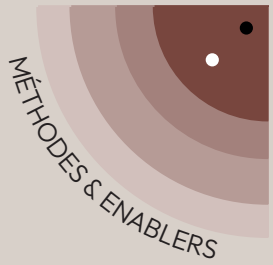
Méthodes & Enablers

9 BLIPS | 2 ADOPT | 5 TRIAL | 1 ASSESS | 1 HOLD

Nous avons itéré depuis 7 ans sur nos pratiques afin, d'une part, de fluidifier les différentes étapes de nos projets IA et, d'autre part, d'intégrer dès les phases de R&D la maîtrise de la valeur du produit auquel les algorithmes contribuent.

Ce quadrant du radar est pour nous l'occasion de partager quelques uns de ces méthodes et "enablers". Cela va du choix des algorithmes à utiliser à leur implémentation, en passant par la mesure de leur performance.

Si certaines pratiques mentionnées sont déjà bien répandues (comme l'utilisation de GitHub Copilot (p.60), qui a franchi la barre du million d'abonnés fin 2023), d'autres comme le Test-Driven Machine Learning (p.62) le sont moins. Plusieurs méthodes et outils présentés ont par ailleurs été directement créés ou adaptées par nos équipes. C'est le cas du principe d'Algorithm Decision Record (p.58), de Sicarator (p.64) : notre générateur de projet de Machine Learning, ou encore de notre approche de projets IA en cycles courts.



ADOPT
1/2 blips

35 Algorithm Decision Record

L'éventail des options algorithmiques à disposition des data scientists est très large : choix de l'algorithme en lui même, de son implémentation, stratégies d'entraînement... En parallèle, les contraintes entre un premier PoC et un algorithme déployé en production sont rarement les mêmes : volume et qualité de données ou hardware à disposition par exemple. Les solutions à disposition évoluent aussi rapidement : à l'échelle de 6 à 12 mois les méthodes à l'état de l'art peuvent changer drastiquement, comme on l'a vu depuis un an avec les modèles d'IA Générative.

Dans des contextes de R&D, il est crucial que les Data Scientists aient une vision claire des **contraintes du produit développé** et des **impacts de chaque choix algorithmique**, à la fois au moment du choix initial, et dans les évolutions ultérieures. Il est aussi important de pouvoir partager simplement avec les autres parties prenantes d'un projet les **arbitrages** liés à ce choix et leurs **impacts**.

Pour répondre à ces enjeux, nous nous sommes inspirés d'un outil standard dans le monde du développement : les **Architecture Decision Record**.

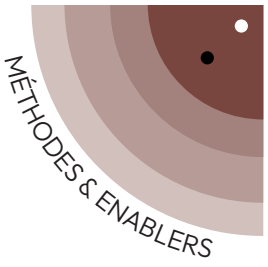
Les éléments clés d'un Algorithm Decision Record sont :

- **Contexte et problème adressé**
- **Critères de décision clé :** critères habituels d'un ADR, ainsi que des questions algorithmiques : y a-t-il **une implémentation existante fiable ?** Y a-t-il des contraintes de **hardware** ? Quelle **donnée** (volume, qualité) puis-je avoir à disposition ?
- **Choix des options et recommandations :** identifier au moins 3 options, e.g. via des aggrégateurs comme Papers With Code ou HuggingFace
- **Impacts** du choix du modèle sur le produit

NOTRE POINT DE VUE

Les Algorithm Decision Records sont devenus la pratique standard de nos équipes de Data Scientists. Un effet de bord positif que nous avons observé est qu'au delà de leur impact sur un projet donné, ils **encouragent les échanges** entre équipes et facilitent le **partage de connaissances**.

Comme pour les Architecture Decision Records, l'usage des Algorithms Decision Records est recommandé en priorité pour des **décisions impactantes**, pour lesquelles le retour en arrière n'est pas trivial.



ADOPT
2/2 blips

36 Métriques business

Sur les projets de Machine Learning, nous construisons des modèles pour **résoudre des problèmes métiers**. Pour évaluer ces modèles, les data scientists ont pour habitude d'utiliser des **métriques issues du monde de la recherche. Ces métriques ne permettent pas toujours de mesurer correctement la valeur de l'outil** pour un contexte métier donnée.

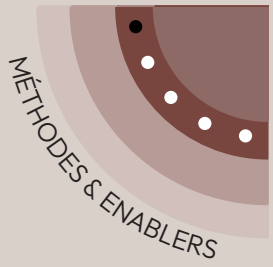
Prenons l'exemple d'un outil censé détecter automatiquement des erreurs sur des factures reçues par e-mail. L'outil va dans un premier temps classifier les pages des documents pour déterminer où se trouvent la facture et le bon de commande, avant de faire les vérifications nécessaires pour trouver des potentiels défauts. Une première manière d'évaluer la performance de cet outil serait d'utiliser des métriques comme l'accuracy globale pour l'étape de classification et un f1-score sur la détection de défauts. Or, dans ce contexte métier, il suffit d'une erreur de l'algorithme sur un e-mail pour nécessiter une intervention humaine. On peut faire l'hypothèse qu'intervenir pour 2 erreurs sur un même e-mail n'est pas tellement plus impactant que d'intervenir pour une seule erreur. Afin de connaitre la valeur "business" de notre outil, il faut donc définir une métrique qui, à la moindre erreur sur un e-mail, estime que l'outil n'a pas été performant dans ce cas-là. Une métrique finale possible est donc le pourcentage d'e-mails traités sans erreurs.

Cette métrique, dite **"business"** permet donc d'évaluer à date si l'outil permet un gain de temps ou pas pour les comptables. On peut même assez facilement estimer le temps gagné grâce à l'outil à partir du pourcentage d'e-mails sans intervention humaine, et **en déduire un réel gain business**.

Pour établir une telle métrique, on passe souvent par des raccourcis. **Une métrique business comprend donc parfois des biais** qu'il est important d'identifier et de rappeler à chaque fois qu'on communique dessus. Dans notre exemple, la performance peut être extrêmement basse par rapport aux performances individuelles de chaque modèle. Si mon détecteur de défauts fait de bonnes prédictions à 80%, il suffit qu'il fasse 1 seule erreur sur chacun des documents pour que ma métrique globale soit pénalisée à chaque fois. D'où l'importance de garder des **métriques plus spécifiques** pour constater les progrès techniques sur les modèles, même si ceux-ci n'apportent pas encore de réelle valeur business.

NOTRE POINT DE VUE

Une **pratique essentielle dans la conception d'un projet de Machine Learning** est de définir cette métrique avec les parties prenantes afin de pouvoir la suivre tout au long du projet et garantir que nos itérations apportent de la valeur au métier.



37 GitHub Copilot

TRIAL
1/5 blips

Plus de 90% des développeurs intègrent de l'IA dans leur environnement de développement, en complétant automatiquement leur **code**. Une des principales technologies utilisées est **GitHub Copilot**.

Lancé en 2021, et s'appuyant sur un modèle de langage dérivé de GPT-3, Copilot vise à être un ***pair-programmer* virtuel**. Il accélère grandement leur travail :

- proposition de la **bonne syntaxe** lors de l'utilisation d'une fonction
- **génération des tests**
- proposition de **refactoring**

La valeur ajoutée ne se situe pas seulement dans le **gain de temps**, mais également dans le **confort de développement** apporté. La qualité de ses suggestions supprime le caractère rébarbatif de ces tâches, pour laisser **le développeur se concentrer sur ce qui importe vraiment** : la logique de développement.

Bien sûr, l'outil a ses limites, et propose parfois du code qui ne répond pas à ce qui était attendu. Les data scientists doivent rester attentifs, et relire systématiquement le contenu proposé. De plus, **les profils moins expérimentés passent moins de temps à comprendre ce qu'ils produisent**, les freinant dans leur apprentissage des bonnes pratiques. Le **Tech Leader** en est renforcé dans son rôle de garant de la **qualité et de la formation de son équipe technique.**

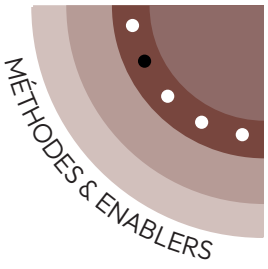
Sans surprise, nous avons observé que Copilot était beaucoup moins pertinent sur des technologies récentes, sur lesquelles peu de données d'apprentissage sont disponibles (par exemple Polars ou LangChain (p.49) en 2023).

Par ailleurs, il n'est pas possible de fonctionner uniquement en local : même si GitHub propose de **ne pas partager** le contenu généré avec GitHub Copilot ni l'utiliser pour entraîner le modèle, les projets les plus stricts en matière de sécurité ne pourront pas utiliser l'outil. Dans ce cas, on pourra s'orienter vers **Tabnine**, qui génère ses suggestions en local ou des modèles open source comme StarCoder.

NOTRE POINT DE VUE

Tous nos développeurs l'ont installé et sont **unanimes quant à son utilité**. Le confort apporté et les gains de temps sont du même ordre que ceux offerts par le passage d'un éditeur de texte à un vrai IDE. Les gains de productivité sont bien supérieurs au prix de la license.

Il nous paraît indispensable d'utiliser GitHub Copilot, notamment pour des projets de Machine Learning en Python. Nous considérons cependant **manquer de recul sur certains effets de bord**,notamment sur la progression des juniors.



TRIAL
2/5 blips

38 Itérations courtes en IA

Dans des projets de développement classiques on travaille généralement sur des fonctionnalités standards dont la complexité est estimable. On peut itérer rapidement afin d'obtenir des retours, et la valeur d'une itération se trouve dans les fonctionnalités développées.

À l'inverse, en IA, les **sujets sont souvent exploratoires** : il faut comprendre la donnée, la traiter, choisir le bon modèle, l'entraîner et enfin l'évaluer pour savoir si l'on va dans la bonne direction. Pour de tels sujets, **comment rester pragmatique et donner régulièrement de la visibilité dans ce tunnel, dont la durée est incertaine et le résultat flou ?**

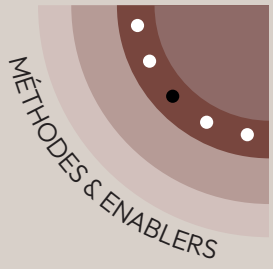
Chez Sicara nous avons expérimenté avec succès la séparation de ces sujets exploratoires en deux types de segments :

- Le **Proof of Concept** (POC) : composé d'une majorité de **tickets d'investigation, dont l'estimation n'est pas en complexité mais est un budget en temps**, et dont le but peut être d'établir une stratégie technique via un Algorithm Decision Record (p.58), d'explorer les données, d'analyser les résultats d'un algorithme, etc. Contrairement à un développement standard où la valeur créée est une fonctionnalité mise à disposition de l'utilisateur final, la **valeur créée par une investigation est un apprentissage** : l'issue d'une "time-box" est une décision : soit d'arrêter d'explorer, soit de continuer à investiguer, soit de passer à l'implémentation.
- L'implémentation : composée de tickets standards à la complexité estimée, elle permet de construire une solution à la lumière de l'apprentissage de l'investigation.

Il faut cependant veiller à ne pas tomber dans des extrêmes : un POC chaotique ou vague, ou à l'inverse une approche bureaucratique documentant chaque choix, même mineur.

NOTRE POINT DE VUE

Nous savons qu'il est complexe et éprouvant de se forcer à découper et de faire des plus courtes itérations, en particulier dans le cadre de projets d'IA. Nous considérons cependant que **le jeu en vaut la chandelle**, et qu'à la clef, le rythme d'apprentissage des équipes et l'adéquation aux enjeux métier en seront accrus. Nous vous recommandons d'essayer et serons ravis d'échanger avec vous à ce sujet.



TRIAL
3/5 blips

39 Test-Driven Machine Learning

La pratique standard en machine learning est de suivre **une métrique de performance** de l'algorithme, mesurée sur un dataset de test. Cependant cette approche ne prend pas en compte l'évolution de la performance spécifique de l'algorithme sur des sous-problèmes : une amélioration de la métrique peut cacher **une baisse de performance sur un sous ensemble** de données critique.

Pour répondre à ce besoin de finesse, nous nous appuyons sur le **Test-Driven Machine Learning (TDML)** (p.62) : une approche de développement d'IA qui s'inspire du Test-Driven Development (TDD).

Le **TDD** est une pratique du développement logiciel dans laquelle les **tests** sont écrits **avant** le code. Le développeur rédige d'abord un test **automatisé** qui correspond aux **spécifications fonctionnelles**, puis écrit le code nécessaire pour faire passer ce test. Puis réitère autant de fois que nécessaire. Ainsi :

- Le développement est guidé par un objectif clair, donc plus efficace.
- Le processus en cycles courts permet de découvrir les problèmes au fur et à mesure.
- Les tests constituent une documentation "vivante" intégrée au code.

Le **TDML** est l'adaptation du **TDD** au monde du **Machine Learning** :

- À la notion de **spécification fonctionnelle**, correspond la notion de **performance** du modèle sur un périmètre délimité via un sous-ensemble d'inputs du modèle. Elle sous-entend la définition de Métriques business (p.59) compréhensibles par le métier. **Un seuil** permet de déterminer si le test associé est **OK ou KO**.
- À la notion de **processus itératif** (ajout de nouveaux tests), correspond le concept de ***slices*** : nous découpons le test set en sous-ensembles appelés slices qui constituent chacun un test. L'ajout d'un nouveau test peut se faire par augmentation du test set global et/ou par un nouveau découpage en slices du test set existant.

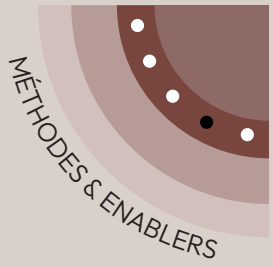
En IA, une partie de la logique fonctionnelle est fournie par un modèle qui n'est pas programmé directement par un humain mais obtenu à travers un processus d'entrainement. Le TDML permet de contrôler la **qualité des outputs et la non- régression** de ces modèles après entrainement.

Le TDML apporte aussi des avantages spécifiques au ML :

- L'ajout de **nouvelles slices** venant de la production peut permettre de détecter une baisse de la performance du modèle dans le temps (**data drift**).
- L'un des principaux challenges lors du développement d'un modèle IA est la **généralisation à un nombre de cas croissant de la vie réelle**. L'ajout de **nouvelles slices** couvrant ces nouveaux cas permet de refléter ce challenge et de guider le travail des data scientists.

NOTRE POINT DE VUE

Bien que nous considérions la pratique du TDML comme saine, nous mettons ce blip en "Trial" de part le **manque de ressources théoriques** disponibles (le TDML est une pratique largement interne à Sicara) et par conséquent le **manque de framework / outils** pour son implémentation et son intégration dans les écosystèmes ML existants. Nous réservons notre recommandation à un public averti. Elle est pertinente dans des phase d'itérations sur un algorithme éprouvé, moins lors de phase de PoC.



TRIAL
4/5 blips

40 Sicarator

Au **lancement d'un projet de Machine Learning**, on a deux options principales en termes de stack technique : la première est d'utiliser une Plateforme de ML end-to-end (p.54), les composants sur l'étagère testés et approuvés sont un gain de temps. Cette solution est cependant accompagnée des inconvénients classiques des solutions managées (**coût**, fonctionnalités **"boîtes noires"**, moins de **personnalisation, intégration limitée** avec d'autres outils, **vendor lock-in**). La seconde option est d'utiliser des **outils open-source** et du **code sur-mesure** pour se constituer sa propre stack. On évite alors les problèmes de la solution managée, au prix d'un **investissement initial** à la fois sur les choix de briques techniques et leur mise en place.

Nous avons développé un générateur de projet, **Sicarator** afin de simplifier cette seconde option : il permet **d'initier en quelques minutes une base de code de qualité** pour un projet de Machine Learning, intégrant des **technologies open-source récentes.** Créé en 2022 pour un usage interne, il a été open-sourcé un an plus tard après avoir prouvé son efficacité sur une vingtaine de projets.

La promesse est de pouvoir, en suivant **une interface en ligne de commande**, générer un projet répondant aux **meilleures pratiques** que nous avons identifiées, comme :

- **Intégration continue** avec plusieurs checks de qualité (tests unitaires, linting, typage)
- **Visualisation des données** avec un dashboard Streamlit (p.19)
- **Tracking et visualisation des données et des expériences** en combinant **DVC** et **Streamlit** (comme expliqué dans le blip DVC (p.14))

Le **code** correspondant est généré avec la **documentation** nécessaire pour l'utiliser. L'outil a été conçu avec une approche **centrée sur le code**, pour donner un maximum de **contrôle**

aux data scientists / ingénieurs ML. L'outil cherche à refléter les bonnes pratiques à mesure que l'écosystème évolue. Par exemple, Ruff a récemment remplacé PyLint et Black comme linter / formateur de code.

Il apportera cependant une réponse **moins complète** que ce que proposent les plateformes les plus avancées, demandant un travail supplémentaire de mise en place.

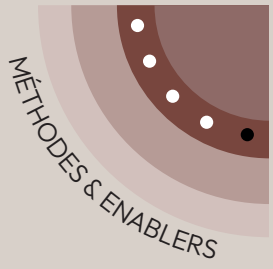
Par exemple, à date, le lancement automatisé d'instances d'entraînement de modèles n'est pas intégré.

NOTRE POINT DE VUE

Ce blip est particulier, car il reflète à la fois l'outil **Sicarator** et **nos convictions sur la stack technique** que ce dernier met en place. Nous serons heureux que vous le testiez et d'échanger avec vous sur les choix faits et les prochaines fonctionnalités à intégrer.

Nous l'utilisons sur n'importe quel projet de ML impliquant du code Python, même ceux utilisant des Plateformes de ML end-to-end (p.54) - dans ces cas précis, on bénéficiera surtout des bonnes pratiques de développement Python incluses dans le générateur. C'est cependant sur des projets avec lesquels on souhaite combiner **des technologies open-source** comme DVC (p.14), Streamlit, FastAPI, etc., que l'outil apportera un maximum de valeur.

Nous conseillons donc Sicarator pour initialiser des projets de ML en Python à toutes les équipes IA avec **une expertise en code** et souhaitant mettre en place un outillage ML orienté open-source.



TRIAL
5/5 blips

41 MTEB

Massive Text Embedding Benchmark

L'information sémantique du texte est habituellement encodée sous formes de vecteurs de taille fixe, appelés embeddings. Des modèles de deep learning spécifiques permettent de calculer ces embeddings à partir du texte brut. Il en existe de nombreux sur étagère et il peut être complexe de sélectionner le plus adapté à son cas d'utilisation.

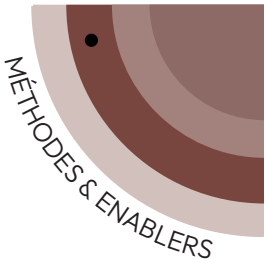
Le Massive Text Embedding Benchmark (MTEB) permet de comparer les différents modèles d'embedding de texte, en agrégeant 129 datasets différents (début 2024) pour mieux généraliser, regroupés en 8 tâches différentes.

Cependant, MTEB présente certaines limitations importantes. Premièrement, **il se concentre uniquement sur trois langues : l'anglais, le chinois et le polonais.** Deuxièmement, il n'y a pas de benchmark cross-language. Les modèles d'embedding comme "text-embedding-ada-2" d'OpenAI ou ceux de Cohere ont pour atout de pouvoir gérer efficacement des datasets multilingues sans nécessiter de traduction préalable (qui résulte souvent en une perte d'information).

Enfin, la **nature open source des datasets** utilisés dans MTEB présente à la fois des avantages et des inconvénients. D'un côté, cela facilite l'accès et l'utilisation par la communauté. D'un autre côté, cela peut permettre aux modèles dont les datasets d'entraînement ne sont pas publics (e.g. Cohere et OpenAI) de s'entraîner spécifiquement sur ces benchmarks pour gagner des points à l'évaluation, sans pour autant refléter une amélioration de leur performance réelle.

NOTRE POINT DE VUE

MTEB est **une référence pour le choix d'un modèle d'embedding** de texte lorsque l'on travaille avec de la donnée monolingue en anglais, chinois ou polonais. Naturellement, si les contraintes du projet le permettent, une solution plus fiable - mais plus coûteuse en temps - reste de **construire un dataset labellisé** représentatif de sa tâche, afin de s'évaluer sur les différents modèles.



42 LLM-as-a-judge

ASSESS
1 blip

L'évaluation des Large Language Models (LLMs) est souvent plus complexe que pour les autres modèles, car il y a généralement un nombre indéfinissable de bonnes réponses, ce qui empêche de faire des métriques automatisées simplement en vérifiant l'égalité de la réponse du modèle et du label attendu.

Le "LLM-as-a-judge" consiste à utiliser un LLM pour évaluer la réponse d'un LLM. Cela a du sens dans 3 cas:

- Si le LLM jugeant est plus performant que le LLM qu'il évalue.
- Si un raisonnement avancé est utilisé (par ex. "chain of thoughts") par le LLM jugeant, qui serait trop couteux en argent et/ou en temps à l'inférence.
- Si la réponse attendue est suffisamment proche d'un label manuel, pour que le LLM jugeant puisse dire si c'est correct ou non.

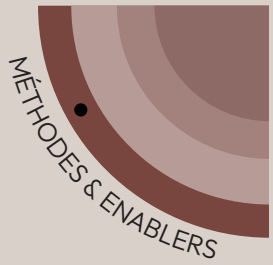
L'avantage de cette méthode est de pouvoir s'évaluer automatiquement sur un large nombre d'exemples. Cependant, ce domaine de recherche est récent et peu mature. Ses principaux inconvénients sont :

- Le fait d'utiliser de l'IA pour s'évaluer : l'outil jugeant la pertinence des réponses est lui-même sujet à des erreurs. Poussé à l'extrême, faut-il une pipeline d'évaluation pour la pipeline d'évaluation ?
- Le fait d'avoir une nouvelle logique à maintenir (si la pipeline évolue, les prompts d'évaluation utilisés le devront aussi)
- Le coût (notamment dans le cas de chain-of-thoughts)

L'alternative principale à cette méthode est de s'évaluer manuellement, ce qui a pour avantage d'être plus fiable et de mieux se familiariser avec la donnée et le comportement du modèle, et pour désavantage d'être chronophage et donc de limiter le nombre d'itérations et d'exemples sur lesquels s'évaluer.

NOTRE POINT DE VUE

Les LLM-as-a-judge permettent d'accélérer le développement de modèles basés sur des LLMs et d'élargir le pannel de réponses évaluées. Cependant, ils comportent de nombreux défauts : nous recommandons donc de l'utiliser avec précaution, et de compléter une évaluation via LLM par des évaluations manuelles régulières pour s'assurer qu'il n'y a pas de dérive et mieux analyser le comportement du modèle.



HOLD
1 blip

43 Itérations sans métriques

sur des modèles basés sur les LLMs

Les métriques sont un **standard** en data science. Cependant, du fait de la simplicité à obtenir des résultats décents avec les Large Language Models (LLMs), il est tentant de **s'abstraire de cette étape clé,** ce qui empêche souvent le produit d'aller jusqu'en **production**

En effet, contrairement aux autres modèles de Deep Learning, les LLMs n'ont (la plupart du temps) pas besoin d'être fine-tuné pour exécuter une tâche précise. La collection d'un dataset d'évaluation n'est donc pas nécessaire pour faire rapidement un **Proof Of Concept (POC).**

Cependant, pour passer en production, la performance initiale ne sera la plupart du temps pas suffisante, ce qui nécessitera des adaptations spécifiques. **Il est alors essentiel de collecter un dataset d'évaluation** (doublé idéalement d'un dataset de test) pour s'assurer que les itérations sont pertinentes et qu'elles généralisent sur l'ensemble de la donnée, mieux prioriser les améliorations à apporter et tout simplement connaître la performance du modèle.

Néanmoins, l'évaluation de LLMs est souvent plus complexe que pour les autres modèles, car il y a un nombre indéfinissable de bonnes réponses, ce qui empêche de faire des métriques automatisées simplement en vérifiant l'égalité de la réponse du modèle et du label attendu. La pratique la plus fiable reste alors l'évaluation manuelle. Cette dernière est cependant très chronophage, ce qui limite le nombre d'évaluations possibles et la quantité de donnée évaluée. Des méthodes plus expérimentales consistent à utiliser les LLMs pour s'évaluer (p 42)

NOTRE POINT DE VUE

Nous encourageons le fait d'innover rapidement sans dataset d'évaluation dans 2 cas : si la performance n'est pas critique (par ex. un outil à usage personnel) ou pour dérisquer les grandes orientations du produit.

Dans les autres cas, s'évaluer est essentiel. Quand l'évaluation automatique via des labels n'est pas possible, nous recommandons a minima l'évaluation manuelle pour sa fiabilité, qui peut être doublée par l'utilisation de LLM-as-a-judge (p.67) pour accélérer les itérations.

Les contributeurs

PIERRE-HENRI CUMENGE

CTO

GASPARD SAGOT

AI Product Manager

YANNICK WOLFF

Lead Data Scientist

LAURE AUDUBON

Growth Marketing Manager

ANTOINE TOUBHANS

Head of Science

NOÉ ACHACHE

Gen AI Leader

YANDI ANDRIAMASY

Data Scientist

MARIA VEXLARD

Lead Data Scientist
& Gen AI Expert

MATHIEU SOUL

Lead Data Scientist

CORENTIN BERTEAUX

Data Scientist

JÉRÉMY MARCHAND

Lead Data Engineer

MAX PERDRIGEAT

Analytics Engineer

ERWAN BENKARA

Data Engineer

JULIEN PERICHON

Lead Data Scientist

EMILIO DE SOUSA

Lead Data Engineer

GUILLAUME DELEPOULE

Lead Data Engineer

JULIE PROST

Lead Data Scientist

PAUL MONNIOT

Data Scientist

VINCENT DOBA

Lead Data Engineer

TYCHO TATITSCHEFF

Principal Technologist BAM

EDMOND VERDIER

Data Scientist

VALENTIN DE LA PERRAUDIÈRE

Lead Data Scientist

TAREK AYED

Lead Data Scientist
& GenAI Expert

À propos de Sicara

Experts data et IA de Theodo depuis 7 ans, Sicara conçoit et construit avec et pour ses clients, des produits sur mesure et durables en Data Engineering et en Intelligence Artificielle, tout en formant leurs équipes.

+200
projets en data et IA

Sicara a développé une réelle expertise en Data Engineering et Data Science plurisectorielles en travaillant aux côtés de grands groupes, PMEs et startup parmi lesquels : **Carrefour, Manpower, Qonto, Pierre & Vacances Center Parcs, TF1, Ornikar**.

+70

Nous sommes aujourd'hui plus de **70 Sicariotes**, répartis au sein des pôles business, développement, produit et tech. Nous sommes tous convaincus que **l'IA révolutionne nos façons de travailler en libérant le potentiel humain**, stimulant la créativité et permettant une collaboration sans précédent, ouvrant ainsi la voie à un avenir où **l'innovation et la compassion guident notre progression collective**.