

Et la QA là dedans ?

Déployer un service utilisant du Machine Learning



François Mockers



QA Lead chez Paylead. Enrichissement des opérations bancaires, en Rust / Python

Maintainer de Bevy Engine. Le futur des moteurs de jeux vidéo, en Rust.



@francoismockers



@mockersf

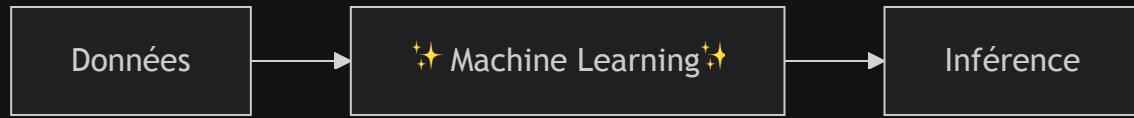
Comment tester le non déterministe ?

- Tests unitaires, fonctionnels, pyramide des tests, comme d'habitude ?
- On fait confiance aux data scientists ?

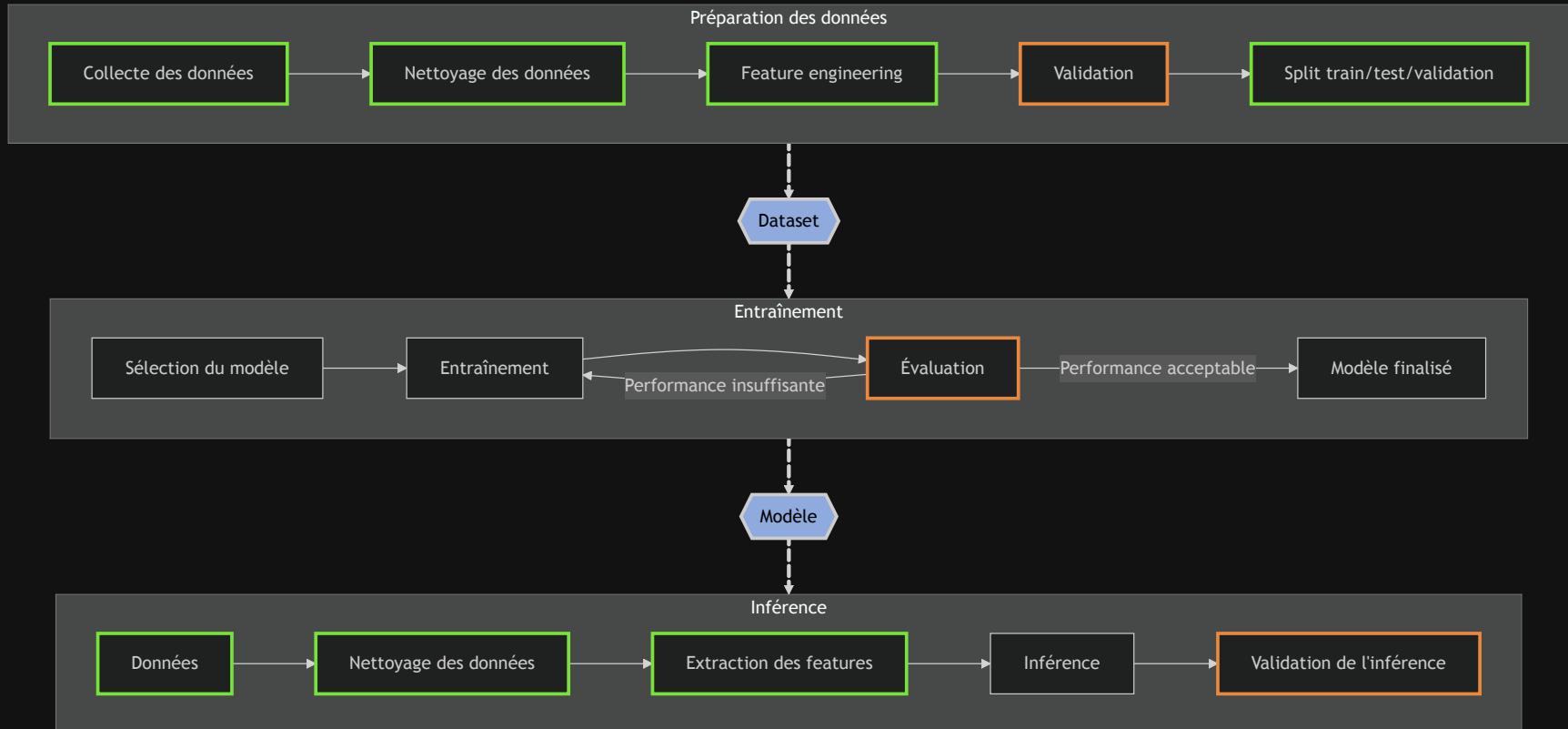
Comment tester le non déterministe ?

- Décomposons notre service de Machine Learning
- Comment tester chaque composant ?
- Comment tester ce qui permet de produire un composant ?

Machine Learning !



Machine Learning...

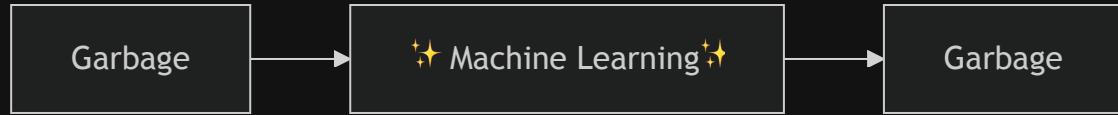


Des Points de Validation

- Validation des datasets
 - Biais, outliers, ...
- Evaluation pendant l'entraînement
 - Accuracy, Precision, Recall, ...
- Validation de l'inférence

Analyse statistique

Importance des Données



- Il faut des données de qualité en (gros) volume pour l'entraînement
- Des transformations sur les données
 - Extraction
 - Format
 - Features

Tout ça peut se tester !

Pipelines de Données

- Pour l'entraînement
 - Création d'un dataset
 - Split pour l'entraînement du modèle
- Pour l'inférence en intégration
 - Contrôle complet des données pour les tests
- Pour l'inférence en production
 - Monitoring
 - Validation de l'inférence en sortie

L'Entraînement : une Boîte Noire ?

- Domaine des data scientists
- Explicabilité
 - SHAP values (SHapley Additive exPlanations)
 - LIME (Local Interpretable Model-agnostic Explanations)
 - ...
- Peu de contrôles pendant l'entraînement

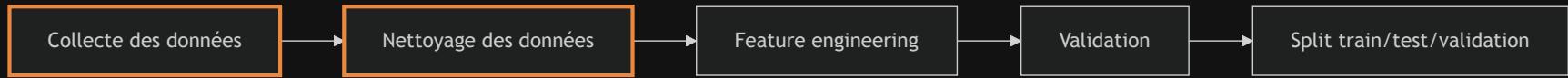
Des Artefacts

- Le dataset
- Le modèle
- Attention au versionning !

Quand, comment utiliser une
nouvelle version ?

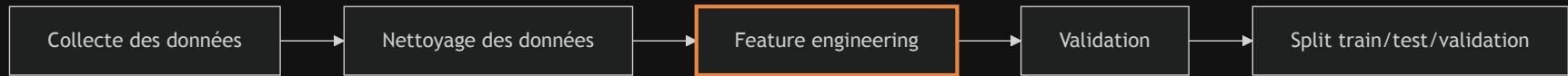
Préparation du Dataset

Préparation du Dataset



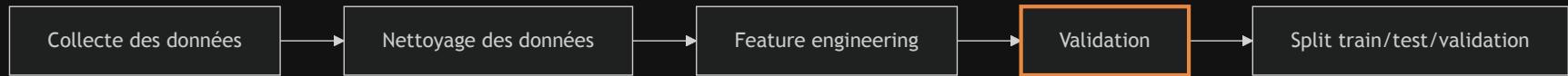
- Validité et consistance des sources
 - Référencer l'origine
- Détection d'anomalies et valeurs manquantes
 - Définir les règles
 - Exclure ou "réparer" les données

Préparation du Dataset



- Sélection des features
 - Informatives
 - Différentiantes
 - Indépendantes
- Extraire les features

Préparation du Dataset



Distribution

- Test de Kolmogorov-Smirnov
- Test du chi-carré
- Tests paramétriques
- Détection de skewness
- ...

Anomalies

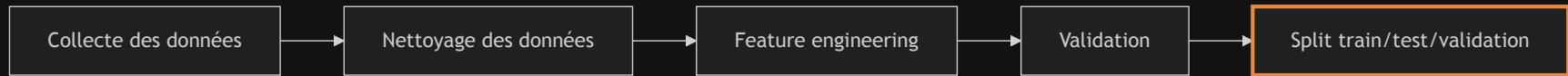
- Z-score
- IQR (écart interquartile)
- Isolation Forest
- DBSCAN clustering
- ...

Corrélations

- Coefficient de Pearson
- Test de Spearman
- Analyse en composantes
- Multicollinéarité (VIF)
- ...

Vérifier que c'est fait !

Préparation du Dataset



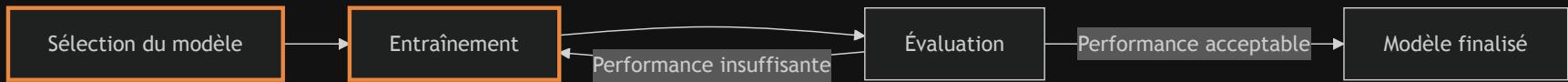
- Split du dataset : Train / Test / Validation
 - Volume de chaque part : 80% / 10% / 10% ?
 - Représentation
 - Validation individuelle
- Comment les stocker
- Comment les partager

Dataset !

- Un Dataset
 - Référence ses origines
 - Liste les transformations qu'il a subit
 - Est versionné
 - Met à disposition ses metadata
- Tester les transformations des données
- Vérifier les analyses statistiques

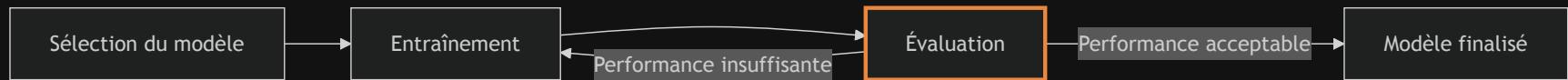
Pendant l'Entraînement

Validation de l'Entraînement



- Choix du type du modèle et de son architecture
- Tuning des hyperparamètres

Validation de l'Entraînement



Classification

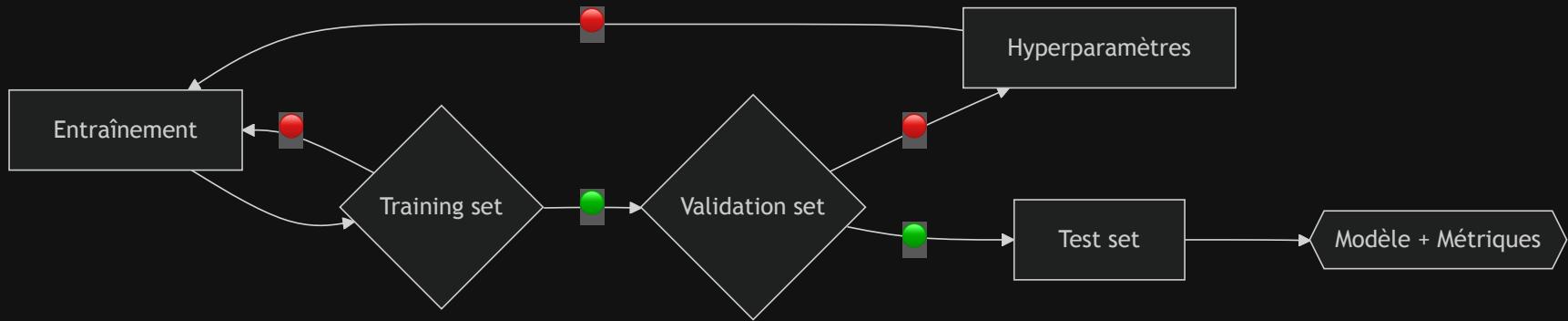
- Accuracy, Precision, Recall
- F1-score
- ROC / AUC
- Confusion Matrix
- ...

Régression

- MSE, MAE, RMSE ((Root) Mean (Squared|Absolute) Error)
- R^2 (coefficient de détermination)
- MAPE (Mean Absolute Percentage Error)
- ...

Vérifier que c'est fait !

Évaluation des Métriques



- Sur le set de Training : continuer l'entraînement
- Sur le set de Validation : changer les hyperparamètres
- Sur le set de Test : ne fait pas partie du cycle de feedback
- Problèmes
 - D'entraînement (under/over training)
 - Mauvaise architecture ou hyperparamètres
 - Mauvais dataset ou features

Modèle !

- Un Modèle
 - Référence le dataset utilisé pour l'entraîner
 - Liste ses hyperparamètres
 - Est versionné
 - Met à disposition ses métriques
- Attention à la sérialisation du modèle

Tester en Intégration

Tester en Intégration



- Tester l'intégration du modèle dans la chaîne data
- Contrôle complet des données
 - Envoyer des données anormales
 - Vérifier le nettoyage
 - Vérifier la préparation des features
- Comment le système réagit aux données inattendues
- Ne pas oublier les cas nominaux

Tester en Intégration



- Modèles de prod : lourds, gourmands, lents
- Modèle custom
 - Rapide et prédictible
 - Faux modèle ? Overtaining ?
- Court circuits
 - Avoir des inputs qui déclenchent chaque "classe" d'output

Tester en Intégration



- Inférences non réalistes
 - Peut dépendre des autres données disponibles sur l'environnement
- Vérifier leur "forme"
- Vérifier leur intégration

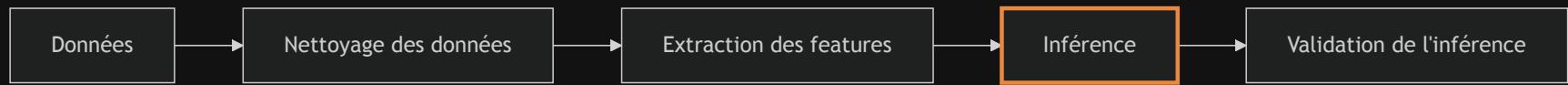
Tester en Production

Tester en Production



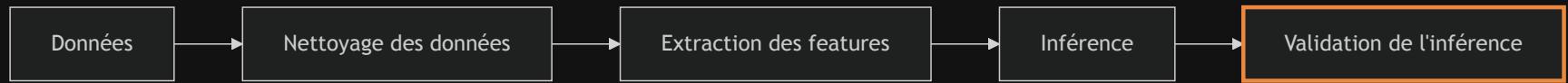
- Données réelles
 - Découverte des vrais cas inattendus
 - Attention aux volumes

Tester en Production



- Peut être coûteux
- Trop de tests peuvent réduire les ressources disponibles pour les vrais utilisateurs
- Métriques de fonctionnement
 - Temps d'inférence
 - Consommation CPU, RAM, GPU, VRAM

Tester en Production



- Comment mesurer la satisfaction des utilisateurs ?
- Demande de feedback rapide
- Est-ce qu'ils interagissent avec le résultat ou l'ignorent ?
- Est-ce que le futur confirme l'inférence ?

Tester en Production

- À intervalle régulier
 - Envoyer des "canaris"
 - Faciles à identifier dans les logs / stats
 - Avec une réponse prévisible
- Alimenter les datasets d'entraînement
 - Avec le feedback utilisateur sur les inférences
 - Attention à anonymiser
 - "Data Flywheel"

Déployer en Production

Déployer en Production

- Premier déploiement
 - Mise en place du monitoring
 - Shadow deployment : ne pas exposer les inférences aux utilisateurs
 - Sinon, mettre un label "beta"
- Prendre le temps de mettre en place les tests en prod
- Si besoin, envoyer le traffic de production par pallier

Déployer en Production

- Déploiement d'une nouvelle version
 - Shadow deployment si possible
 - Blue/green, canary release sinon
- Suivi des métriques pour détecter les régressions
- Comparer les résultats de la nouvelle version avec celle d'avant
- Rollback rapide en cas de problème

Déployer en Production

- Déploiement d'une modification de la pipeline de data
 - Souvent avec changement du modèle
 - Donc changement des métriques
 - Modification des features
- En SemVer, une version majeure

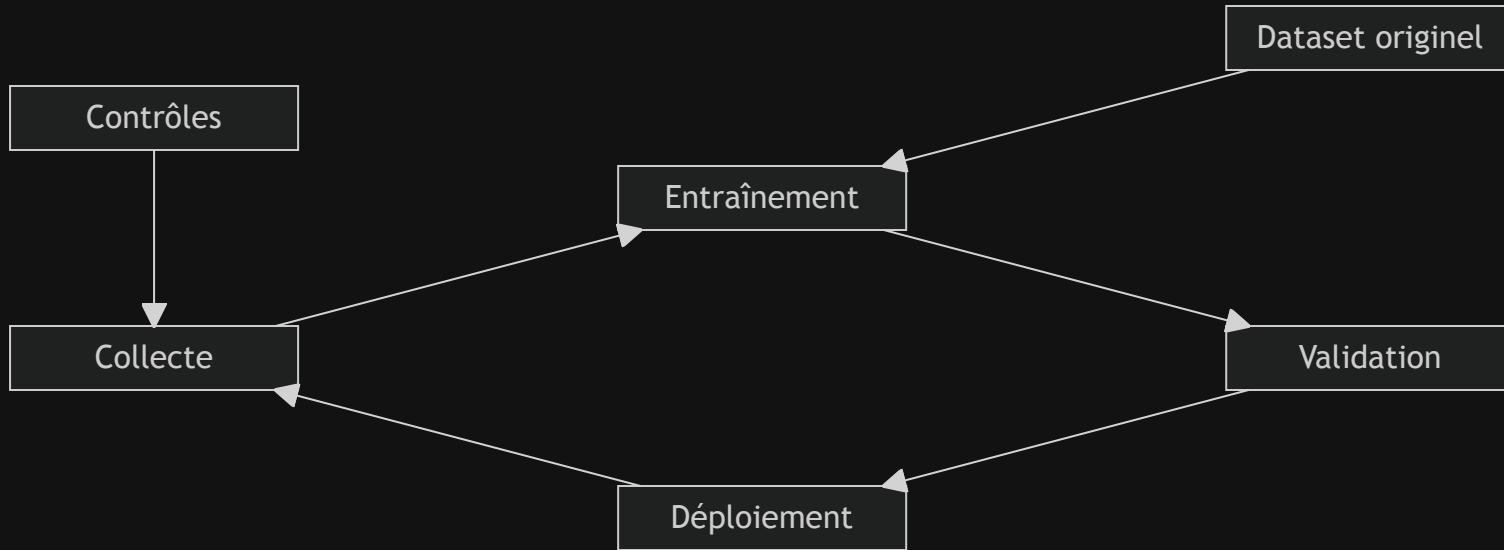
Déployer en Production

- Déploiement d'un changement du modèle
 - Souvent avec changement des métriques
- En SemVer, une version mineure

Déployer en Production

- Déploiement d'une mise à jour des données d'entraînement
 - Entraînement du modèle avec les nouvelles données
 - Surveillance des métriques
 - Automatisation du réentraînement et du déploiement
- en SemVer, une version patch
- Doit devenir un "non-événement"
 - Données périmées, modèle périmé
 - "Data Flywheel"

Data Flywheel

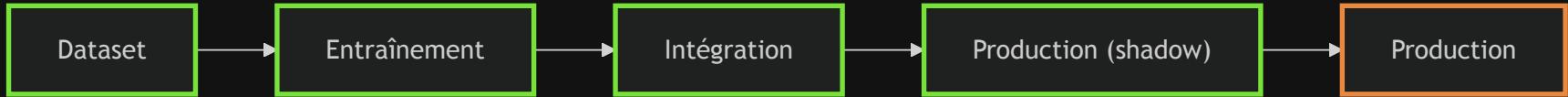


Cycle de Vie d'un Modèle

Cycle de Vie d'un Modèle

- Avant le modèle, le dataset
 - D'où vient-il ?
 - Quelle features expose-t-il ?
 - À quelle date a-t-il été constitué ?
- Tracabilité
- Data Lineage
 - RGPD ?

Cycle de Vie d'un Modèle



Dataset

- Origine
- Transformations
- Statistiques

Entraînement

- Architecture
- Hyperparamètres
- Métriques

Intégration

- Traitement des données

Production

- Shadow
- Monitoring
- Validation

Cycle de Vie d'un Modèle

- Plateformes de MLOps
 - MLFlow
 - Kubeflow
 - Sagemaker
 - Weights & Biases
 - ...
- Va prendre en charge plus où moins d'étapes
- Stockage des metadatas
- Promotion de modèles

Generative AI

Generative AI

- Domaine encore à défricher, au moins pour moi
 - Pas encore professionnellement
- Validation du dataset
 - Devrait rester similaire
- Tests de prompt
 - Protéger contre les injections
- Contexte
 - Vérifier qu'il est correctement rempli

Generative AI

- Fonctions
 - Testable unitairement
- Retrieval Augmented Generation
 - Tester la vectorisation des documents
- Model Context Protocol
 - Tester chacun individuellement
- Agents
 - Chaque agent a ses prompts et son contexte

Generative AI

- Typer la réponse
 - Schéma JSON
 - Validation spécifique à ce qui est attendu
 - Génération de code ? Linter, compiler, ...
- LLM as Judge
 - Le LLM testé donne des réponse libres
 - Elles sont analysées par un LLM Juge qui réduit le domaine de la réponse
 - Plusieurs juges peuvent être utilisés

Takeaway

Takeaway

- Beaucoup du travail des datascientists est du Développement Logiciel
 - Tous les outils habituels s'appliquent
 - Tests
 - Version Control
 - ...
- Pour ce qui l'est moins, il y a des métriques
 - Vérifier leur collecte
 - Tester leur évolution

Takeaway

- Certaines fonctionnalités sont souvent écrites deux fois
 - Pour l'entraînement
 - Pour le déploiement
 - Avec des contraintes différentes de volume, des langages différents, ...
- Ne pas oublier la production !
 - Avoir prévu l'inattendu
 - Monitorer les métriques: résultats de l'inférence, fonctionnement du modèle

Takeaway

- Des déploiements comme les autres
 - Avec des vérifications et des approbations
 - Qui peuvent être complètement automatisés
- Des nouveaux types de livrables
 - Qui peuvent être versionnés
 - Et dont la qualité peut être mesuré

Comment tester le non déterministe ?

En testant sa fabrication

En l'isolant

En le rendant déterministe

En le monitorant