Bloc 3
Pipeline de données

David **RAMBEAU**

Full **S**tack in **D**ata **S**cience

Promotion 2024 -2025

- Sommaire -

- Pipeline de données -

1. Objectif d'un pipeline de données.

Un pipeline de données constitue l'infrastructure permettant le transfert, la transformation et l'exploitation des données au sein d'une organisation moderne.

Son objectif principal est de standardiser le flux de données depuis des sources hétérogènes jusqu'aux systèmes de destination, c'est-à-dire jusqu'à l'exploitation.

Ce procédé de fonctionnement repose sur une architecture garantissant le maintien de la qualité, de la disponibilité et de la conformité réglementaire des données.

Dans notre cas, nous allons nous placer dans le contexte bancaire, et particulièrement celui de La Banque Postale, pour créer un pipeline de données qui pourrait simuler celui que l'on retrouve dans notre structure postale.

Dans l'ordre, nous commencerons par l'ingestion massive et en temps réel de transactions provenant de multiples sources.

Ensuite viendra la phase de transformation et d'enrichissement qui permettra de nettoyer, normaliser et agréger les données brutes.

Dans notre cas d'usage de détection de fraude, le pipeline va permettre de détecter une fraude en fonction d'un modèle qui aura préalablement appris sur des données historiques.

Enfin, le pipeline vise l'optimisation opérationnelle en réduisant les interventions manuelles, standardisant la qualité des données, et en créant un système reproductible.

2. Contexte pour La Banque Postale.

Pour les besoins du cas d'usage, nous nous placerons comme acteur majeur du secteur bancaire français, La Banque Postale.

Comme ses consœurs, La Banque Postale fait face à une augmentation des tentatives de fraude sur les transactions en ligne, notamment les virements et paiements par carte.

Pour renforcer la sécurité de ses 10 millions de clients[1], elle souhaite déployer un système de détection de fraude en temps réel, capable d'analyser chaque transaction rapidement et d'identifier les comportements suspects (montants anormaux, géolocalisation incohérente, horaires inhabituels).

Ce projet s'inscrit dans sa stratégie de protection des données clients et de conformité avec les réglementations DSP2[2] et RGPD[3].

Le système repose sur une architecture entraînée sur des données historiques (montants, profils clients, historiques de fraude) et déployée en production via une API interne.

Ce projet vise à réduire les pertes financières liées à la fraude, tout en minimisant les faux positifs pour éviter de pénaliser l'expérience client.

Il s'intègre aux outils existants de La Banque Postale, comme son système de gestion des risques, et respecte les contraintes de souveraineté des données (hébergement en France).

Une phase pilote sera menée sur les transactions en ligne avant un déploiement progressif à l'ensemble de son parc.

C'est ce cas d'étude qui sera abordé ici.

---

[1] *https://www.labanquepostale.com/a-propos/presentation-chiffres-clefs/chiffres-clefs.html*

[2] *DSP2 : Directive sur les Services de Paiement - https://www.francenum.gouv.fr/guides-et-conseils/developpement-commercial/solutions-de-paiement/paiements-en-ligne*

[3] *RGPD : Réglementation Générale sur la Protection des données*

3. Préparation et transformation des données.

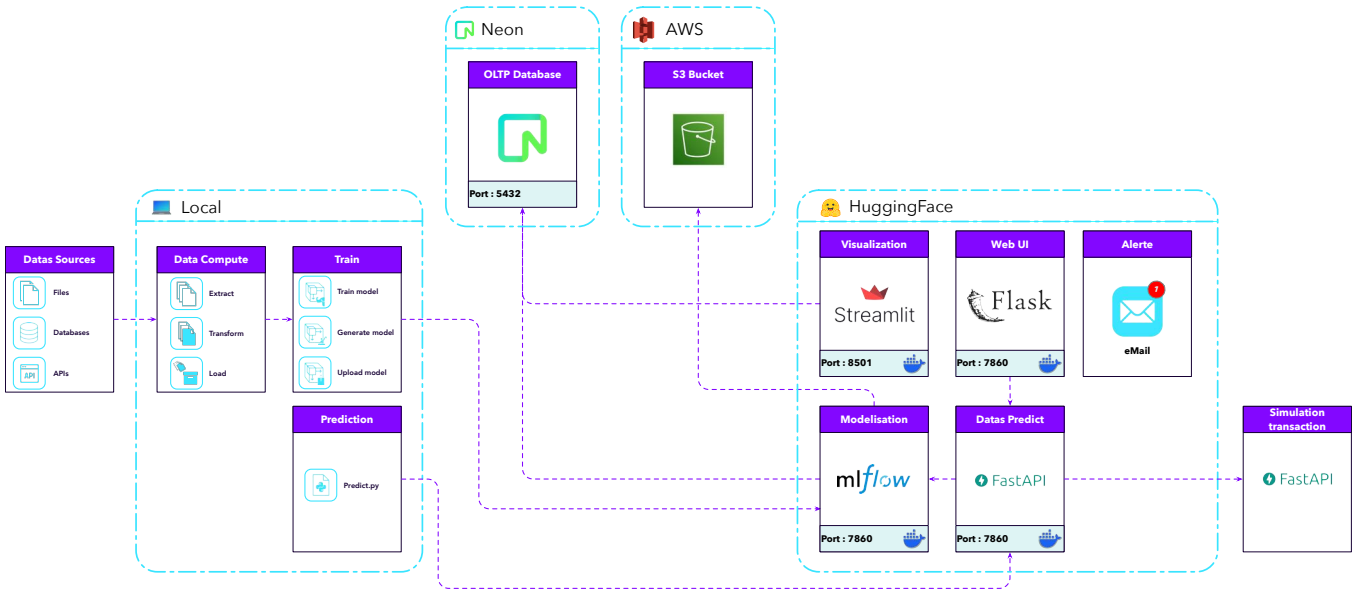
Afin de modéliser les transactions, de comprendre le travail à fournir et également pour nourrir et entraîner le modèle, nous utiliserons le jeu de données de référence suivant :

🔗 ***https://huggingface.co/spaces/sdacelo/real-time-fraud-detection***

3.1. Pipeline de données.

Pour la réalisation de l'architecture de détection de fraude, nous avons besoin de mettre en place tout un « système » de couplage de données, pour à la fois procéder à l'apprentissage depuis le jeu de données de référence, jusqu'à la détection en temps réel.

Pour cela, un schéma d'architecture global cible va permettre de comprendre l'imbrication des flux d'informations et fera l'objet d'une ébauche de cahier des charges par la constitution de ce dossier et des livrables déployés sur GitHub©.



3.2. Inventaire des données.

Avant toute analyse ou modélisation, l'inventaire des données est une étape facilitant la qualité et la pertinence du projet de création d'un modèle d'apprentissage.

Cette phase permet notamment de comprendre la structure et la provenance des données en identifiant les sources et les technologies afférentes.

C'est également le moment de vérifier que les données disponibles répondent aux besoins du projet, et si nous aurons besoin d'agréger plusieurs sources de données pour augmenter le niveau de compréhension.

Cette étape permet de préparer le travail collaboratif, notamment, en documentant les données et en partageant un dictionnaire des ressources.

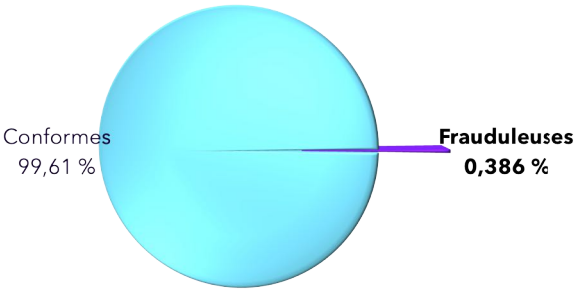
Dans notre cas, voilà à quoi ressemble ce dictionnaire :

#	Nom de la variable	Contenu	Objectifs
1	**trans_date_trans_time**	Date et heure exactes de la transaction	Analyser la temporalité, détecter des schémas suspects, repérer des pics d'activité frauduleuse
2	**cc_num**	Numéro de carte bancaire	Identifier le porteur, regrouper les transactions, repérer des comportements inhabituels
3	**merchant**	Nom du commerçant	Détecter des commerçants à risque, analyser les habitudes du client
4	**category**	Catégorie du commerçant	Identifier les types d'achats, détecter les anomalies dans la typologie des dépenses
5	**amt**	Montant de la transaction	Détecter des montants inhabituels, pondérer la suspicion
6	**first**	Prénom du porteur	Pas utile dans la prédiction, permet de s'affranchir de données personnelles
7	**last**	Nom de famille du porteur	
8	**gender**	Genre du porteur	Voir si il y a des comportements en fonction des genres (caractères discriminatoire)
9	**street**	Adresse – Rue	Connaissance de la localisation du porteur Analyse géographique, détecter les transactions incohérentes Identifier les zones habituelles d'achat Localisation fine du porteur, détection d'incohérence géographique Déterminer l'environnement urbain, corrélations éventuelles
10	**city**	Ville de résidence	
11	**state**	État / Région	
12	**zip**	Code postal	
13	**city_pop**	Population de la ville	
14	**lat**	Latitude du domicile	Calcul de distance domicile → lieu d'achat
15	**long**	Longitude du domicile	Calcul de distance domicile → lieu d'achat
16	**job**	Profession du porteur	Catégorie sociaux pro, faire de la classification (oneHot encoding)
17	**dob**	Date de naissance	Calcul de l'âge
18	**trans_num**	Identifiant unique de transaction	Suivi et traçabilité des transactions, pas utile dans la creation dumodele car unique
19	**unix_time**	Timestamp Unix de la transaction	Calculs temporels rapides
20	**merch_lat**	Latitude du magasin	Calcul distance client → magasin pour détecter anomalies
21	**merch_long**	Longitude du magasin	Calcul distance client → magasin pour détecter anomalies
22	**is_fraud**	Indicateur de fraude (0 = conforme, 1 = fraude)	Variable cible pour l'entraînement du modèle de détection de fraude

3.3. Nettoyage et prétraitement des données.

Avant toute utilisation « froide » des données, il est impératif d'en comprendre la pertinence et la représentativité. Pour cela des méthodologies de traitements existent qui permettent de poser des hypothèses et de définir une stratégie de traitement des variables d'entrées, conservation/élimination, labélisation, standardisation, etc.

3.3.1. Analyse exploratoire des données



Pour comprendre la structure des données, leurs caractéristiques et les anomalies du jeu de données, il est important de :

- ***Identifier*** *les déséquilibres (ex. : rapport des transactions frauduleuses).*
- ***Visualiser*** *les distributions pour repérer les tendances, les valeurs aberrantes.*
- ***Estimer*** *les variables pertinentes et celles à transformer ou exclure.*
- ***Éviter*** *les pièges, comme l'utilisation de données non informatives ou redondantes.*

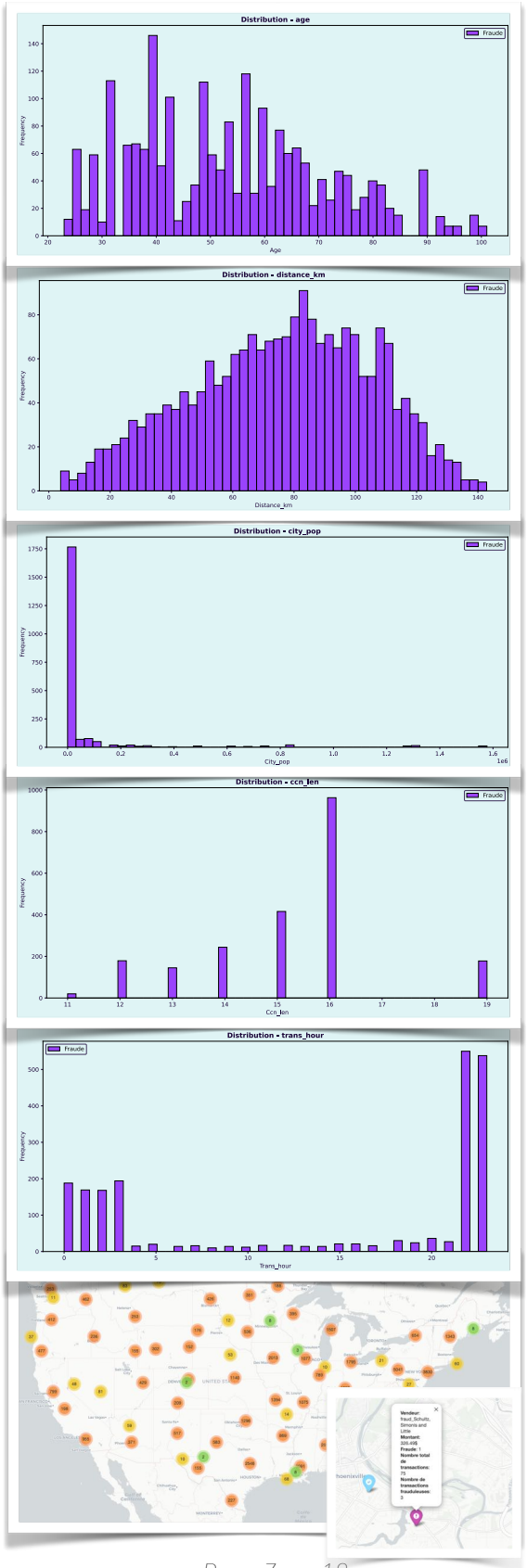
Les exemples ci-contre montrent le résultat des analyses réalisées dans la phase préliminaire de traitement des données.

Ces observations permettent de rapidement mettre en lumière les éléments sur lesquels ils faut se concentrer.

À titre de première constatation, les fraudes surviennent le plus souvent la nuit entre 22h et 3h du matin.

Enfin la répartition géographique ne permet pas de cibler une zone particulière d'étude, la fraude est généralisée sur l'ensemble du territoire.

Par ailleurs, une problématique est identifiée à la lecture du jeu de données. En effet, un nombre limité de transactions frauduleuses dans l'échantillon entraîne un déséquilibre significatif et va rendre l'élaboration de prédictions plus ardue.



3.3.2. Création de nouvelles variables.

Il est intéressant de s'apercevoir qu'en transformant les données brutes, en les enrichissant, en les croisant avec d'autres sources ou en modifiant leur échelle, il devient alors possible de générer de nouvelles informations nous aidant à la compréhension du problème initial.

Ces transformations facilitent l'interprétation des modèles en créant des variables plus intuitives en changeant des coordonnées en distance, par exemple.

Elles permettent également d'améliorer les performances du modèle en capturant des motifs caractéristiques de la fraude, comme des transactions successives effectuées en très peu de temps.

Parmi les données supplémentaires extraites des données, nous avons pu dégager le tableau suivant :

#	Nom de la variable	Contenu	Objectifs
1	**age**	Age de l'acheteur calculé à partir de sa date de naissance.	Permet de repèrer des profils générationnel
2	**distance**	Distance géographique entre le vendeur et le domicile du client	Identifier des comportements inhabituels
3	**trans_hour**	Heure de la transaction	Heure à laquelle la transaction a été effectuée, permet de repérer des pics d'activité frauduleuse.
4	**ccn_len**	Longueur du munero de carte bleue	Validité de la carte selon la provenance par exemple
5	**bin**	Identification de la banque à partir du numéro de carte bleue	Permet de repèrer des banques pour lesquelles les transactions seraient facilités

Ces données viennent maintenant compléter le dataset de départ et également le dictionnaire de ressources.

3.3.3. Feature engineering et sélection des variables

L'efficacité d'un modèle d'apprentissage est liée à la qualité des caractéristiques utilisées pour le former. Le « feature engineering[4] » englobe différentes techniques permettant de créer de nouvelles caractéristiques en combinant ou en modifiant les existantes.

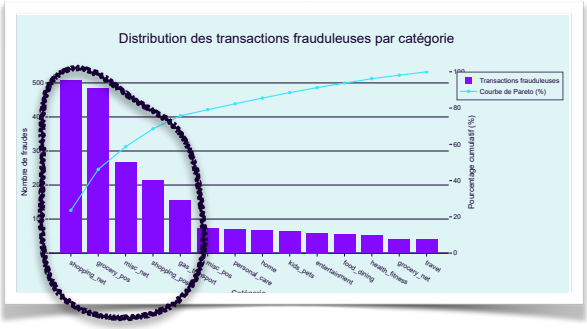
Le feature engineering reposent sur trois techniques majeures :

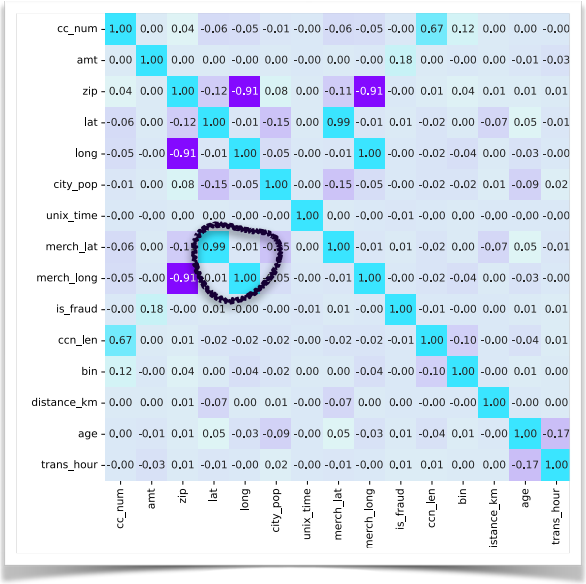
- *Le processus de normalisation, standardisation, encodage catégoriel (one-hot).*
- *Le processus de combinaisons de features, extraction d'informations à partir de dates ou de textes, agrégations temporelles.*
- *La réduction des dimensions vise à simplifier les données tout en préservant leur structure informative.*

Le choix des variables doit rester très sélectif de façon à privilégier celles fortement corrélées avec la cible, non redondantes, robustes aux variations et facilement interprétables.

À l'inverse, il faut écarter les features introduisant du bruit ou de la complexité inutile sans gain réel de performance, afin de limiter le risque de surapprentissage.

Pour nous aider dans cette démarche, des outils comme les histogrammes de distribution (vu plus haut), les diagrammes Pareto ou les matrices de corrélations, nous permettent de comprendre comment s'articulent les données.





En faisant un « Pareto » sur les catégories d'achat, on constate que les fraudes se concentrent majoritairement sur cinq d'entre elles.

Ici, par exemple, la matrice de corrélation nous montre qu'il existe des relations entre les positions (latitude/longitude) entre le marchand et l'acheteur.

En résumé, le feature engineering constitue une étape déterminante qui permet de découvrir comment les variables sont liées et nous donne les pistes de recherche pour les étapes d'apprentissages.

[4] *Ingénierie des caractéristiques*

3.4. Intégration et stockage des données

3.4.1. Organisation des données.

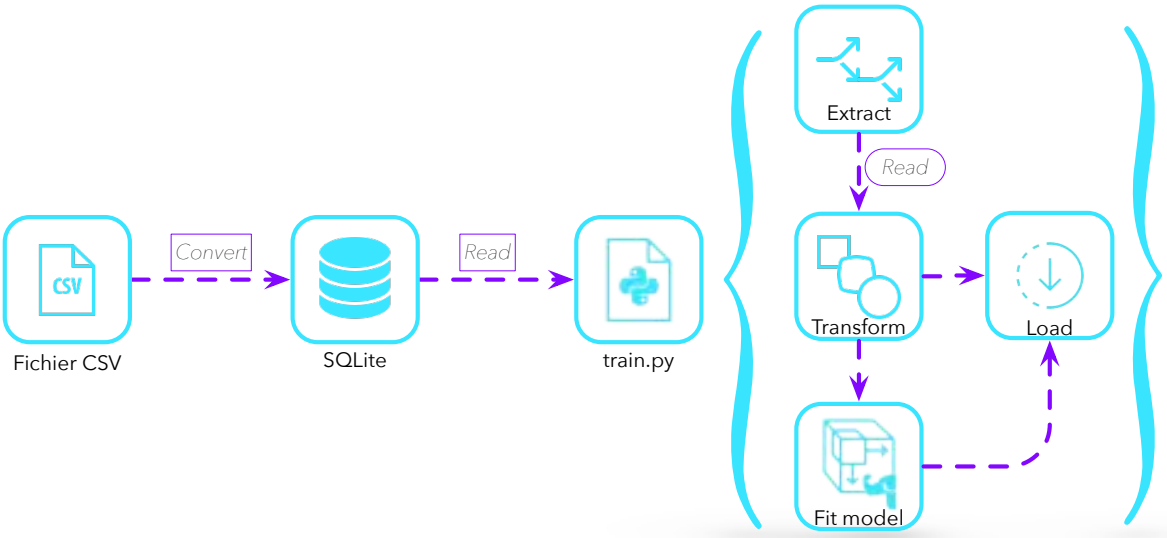
L'intégration et le stockage des données sont des étapes critiques pour garantir que les informations préparées (nettoyées, transformées, et enrichies) soient accessibles, sécurisées et optimisées pour l'entraînement des modèles et le déploiement en production.

Il est impératif de centraliser les données au sein de structures cohérentes, optimisant ainsi leur exploitation par les équipes de data science et métiers.
Il faut par ailleurs assurer la traçabilité des versions (ex. : jeux de données avant/après prétraitement) pour reproduire les analyses ou auditer les résultats.

Cette façon d'organiser les données nous pousse à optimiser les performances en choisissant des solutions technologiques adaptées aux volumes et aux besoins en temps réel.

Dans le cadre d'un projet de détection de fraude pour La Banque Postale, cette phase doit répondre à des enjeux spécifiques de scalabilité (millions de transactions par jour), de sécurité (données sensibles), et d'intégration fluide avec les systèmes existants.

Pour cela, la mise en place d'un flux de traitement de données est nécessaire en amont de notre pipeline qui va permettre de répondre aux enjeux de performance et aux contraintes de qualité jusque-là énoncée.



Sur ce schéma on peut s'apercevoir que les données d'entrée seront d'abord transformées en base de données locales (SQlite) permettant de requêter facilement en cas de besoin rapide dans la phase exploratoire.
Une fois cette transformation effectuée, nous pourrons commencer le traitement, ici dans un fichier python, pour s'appuyer sur un ETL et entraîner le modèle.

3.4.2. Stockage des données

Parmi l'ensemble des données à conserver, nous pouvons, dans notre cas, les distinguer en deux catégories, les données de transactions et les données « techniques » d'apprentissage du modèle.

Concernant les données de transactions, une base de données relationnelle constitue un moyen efficace de garantir les relations solides qui existent entre les tables.

D'autre part, les données issues du modèle se répartissent en deux types : les métadonnées qui ont servi à l'apprentissage et les données constitutives du modèle.

Pour cela nous utiliserons deux espaces de stockage différents en fonction de ces contraintes.

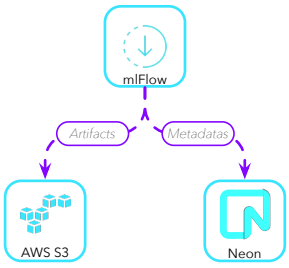
La base de données relationnelle va permettre le stockage des l'ensemble des transactions, mais également des expériences, de leurs métrique, des tags utilisés, etc.

Neon

- id
- cc_num
- amt
- trans_date_trans_time
- …

Stocke des données PostgreSQL relationnelles de taille variable avec un accès rapide et optimisé pour les applications nécessitant une latence faible

D'autre part, la création d'un modèle de machine learning génère deux catégories de données aux caractéristiques distinctes :



Neon

- experiments
- runs
- metrics
- params
- tags

Ces données sont de taille modeste (quelques Ko à Mo), elles nécessitent un accès rapide pour l'analyse comparative et le monitoring.

AWS S3

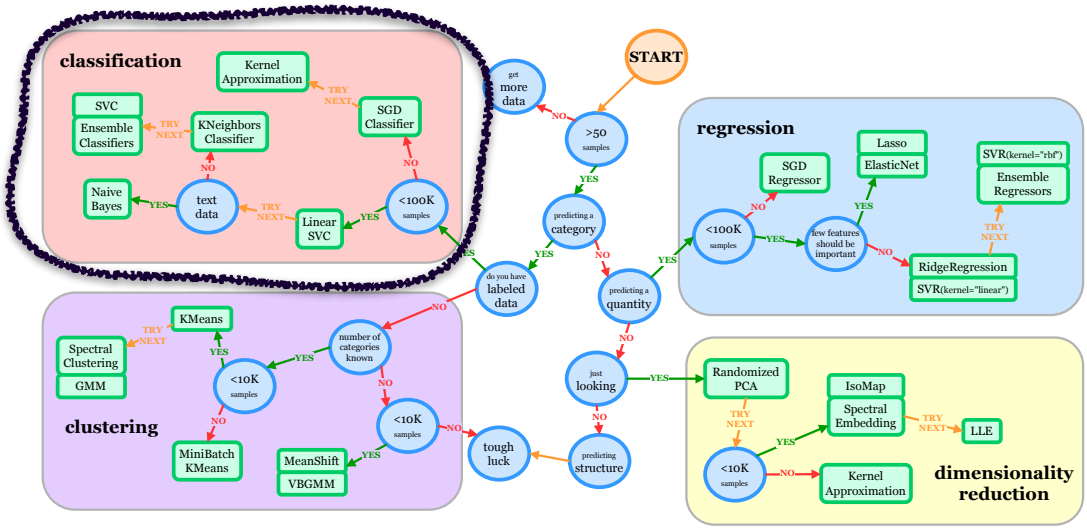
- conda.yaml
- input_example.json
- MLmodel
- model.pkl
- python_env.yaml
- requirements.txt
- serving_input_example.json

Modèles sérialisés pouvant atteindre plusieurs centaines de Mo, il peut également y avoir les visualisations KPIs, logs d'entraînement détaillés, etc.
Ces fichiers, potentiellement de grande taille, requièrent un stockage performant.

4. Modélisation et apprentissage

4.1. Méthodologie de présélection des modèles

Dans le contexte de la détection de fraude, la décision se résume à un choix binaire : « Fraude » ou « Pas Fraude ». Par conséquent, nous nous orienterons vers la sélection d'un modèle de classification approprié.



En s'inspirant de l'arbre de décision réalisé par sklearn[5], trois options distinctes se présentent à nous :

- *Support Vector Classifier*

Bien que ce modèle ne fournisse pas les meilleurs résultats sur un jeu de données déséquilibré, comme c'est souvent le cas en détection de fraude, il a été initialement envisagé pour l'évaluation de ses performances théoriques.

- *Régression logistique*

Ce modèle a été retenu en raison de son interprétabilité.
Il permet d'analyser clairement l'influence des variables sur la prédiction, ce qui est essentiel pour comprendre les facteurs associés à la fraude.

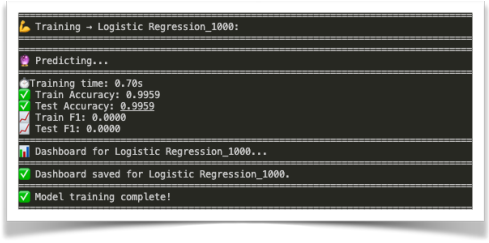
- *Arbre de décision*

Ce modèle a été sélectionné pour sa grande performance de capacité de traitement, une caractéristique importante dans le contexte de la détection de fraude.

En fonction des résultats que nous aurons réussi à obtenir, et à travers l'évaluation des modèles par le biais de métriques bien choisies, l'objectif sera de déterminer le modèle le plus adapté à notre problématique de détection de fraude.

5 skLearn est une bibliothèque Python dédiée au machine learning → https://scikit-learn.org

4.2. Entraînement des modèles



Pour construire le système d'apprentissage, il faut créer et entraîner des modèles qui reposent sur une approche méthodique combinant données historiques et des algorithmes adaptés.

Après le prétraitement (normalisation, encodage), trois modèles ont été testés : deux régressions logistiques, et un arbre de décision. Le modèle SVC n'ayant pas réussi à donner de réponse avec un échantillon de données trop important (*ce dernier point fera partie des axes d'améliorations*).

Chaque modèle a été entraîné sur un jeu de données partitionné (80% entraînement, 20% test), chacun des modèles ayant fait l'objet d'un test permettant l'évaluation postérieure.

4.3. Évaluation des modèles

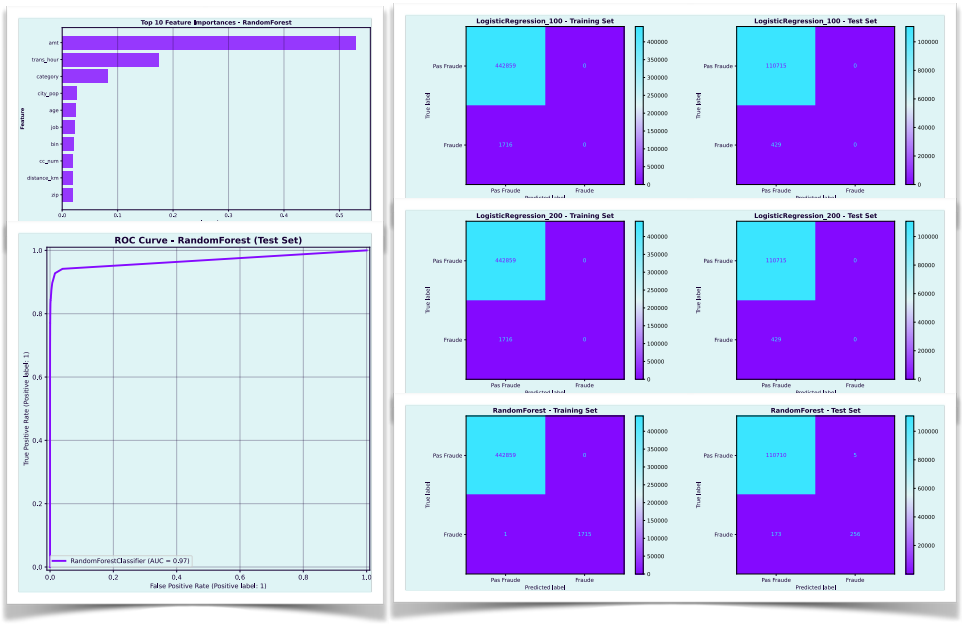
Afin de comparer les différents modèles, nous avons procédé à la mise en place d'indicateurs de performance essentiels à l'évaluation de chaque modèle dans l'objectif de venir choisir le plus adapté à notre problématique.

Parmi les indicateurs observés,

- *L'accuracy, qui quantifie la proportion de prédictions correctes parmi l'ensemble des prédictions effectuées par le modèle.*
- *Le F1-Score s'avère particulièrement pertinent pour l'évaluation des modèles sur des jeux de données déséquilibrés.*
- *La courbe ROC et l'aire sous la courbe (AUC).*

La matrice de confusion permet d'obtenir une représentation détaillée des prédictions du modèle, en distinguant les véritables positifs, les véritables négatifs, les faux positifs et les faux négatifs.

Cette phase a permis de sélectionner le modèle offrant le meilleur compromis entre performance et extensibilité, tout en garantissant une latence compatible avec un déploiement en temps réel.

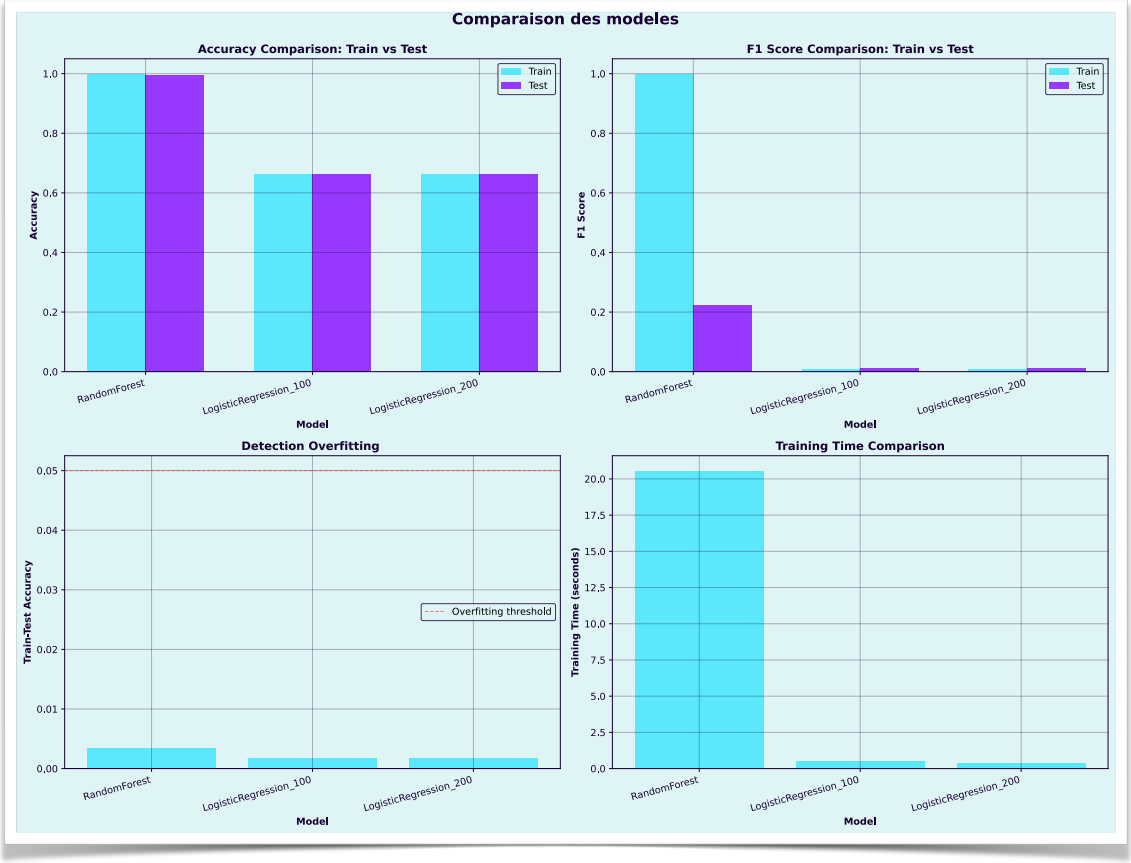


4.4. Résultats des modèles.

En conclusion, après comparaison des modèles, le plus pertinent est le modèle **random forest** qui présente les meilleurs résultats.

Cependant il est à noter qu'avec le jeu de données d'apprentissage qui présente un déséquilibre, il serait bon de continuer les travaux d'amélioration du modèle.

Par ailleurs, ce modèle présente l'inconvénient d'être moins « explicable » que des modèles plus simple. Il faut donc conserver une analyse quant au choix de la performance vs l'éthique ou la réglementation.



Modèle utilisé	Test F1 Score	Test Accuracy	Test ROC AUC	Training Time	Décision
Logistic Regression (100)	0 %	66,59 %	64 %	**1,9772s**	
Logistic Regression (200)	0 %	66 %	71 %	*1,9845s*	
👑 **Random Forest**	**97,94 %**	**99,78 %**	**97 %**	22,92s	

5. Déploiement et industrialisation.

La transition d'un modèle de machine learning depuis un environnement de développement local vers une plateforme de production accessible nécessite une stratégie de déploiement structurée, garantissant la reproductibilité, la portabilité et la maintenabilité du système.

Cette section détaille le workflow complet adopté pour le pipeline de détection de fraude, de l'expérimentation initiale jusqu'au déploiement sur HuggingFaceSpaces ainsi que la capture des transactions en temps réels.

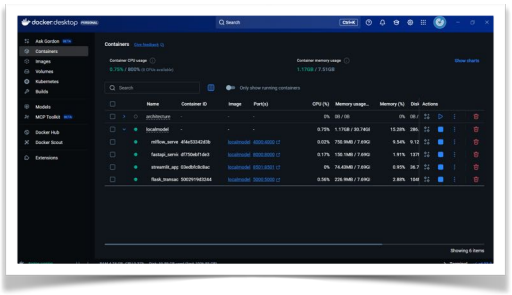
5.1. Conteneurisation

Pour passer d'un prototype développé localement (comme vu précédemment) à une solution déployable, les composants clés du pipeline sont « conteneurisés » via Docker©, technologie simple de mise en route et à l'avantage de pouvoir être portable de machine en machine.

Docker© résout donc le problème classique du « ça marche sur ma machine » en encapsulant l'intégralité des éléments techniques (*librairies Python, configurations, variable d'environnement, etc.*) dans une « image » transportable.

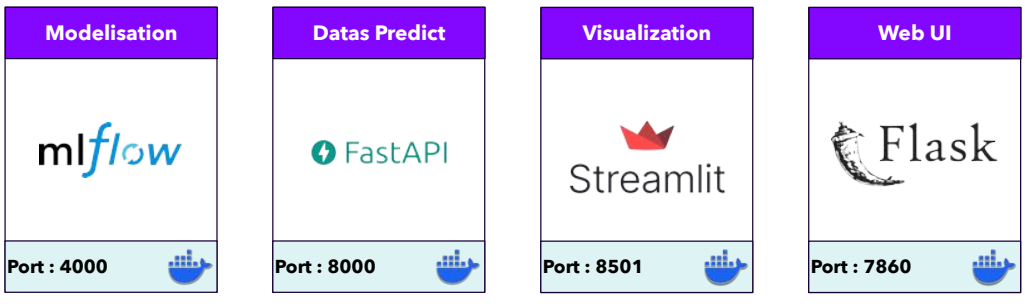
Cette approche garantit que l'environnement d'exécution reste strictement identique entre les tests locaux et la production sur des plateformes distantes.

Dans notre cas, le système emploie quatre services applicatifs indépendants :



- ***MLflow*** *: Serveur de tracking et model registry, stockant les métadonnées dans NeonDB et les artefacts sur S3.*
- ***Streamlit*** *: Interface utilisateur exposée publiquement, permettant de connaitre l'état des transactions.*
- ***FastAPI*** *: API dévaluation de fraude, alimentant Streamlit*
- ***Flask*** *: pour créer une interface utilisateur de simulation des transactions*

Cette séparation des responsabilités facilite la gestion et permet une extensibilité forte.



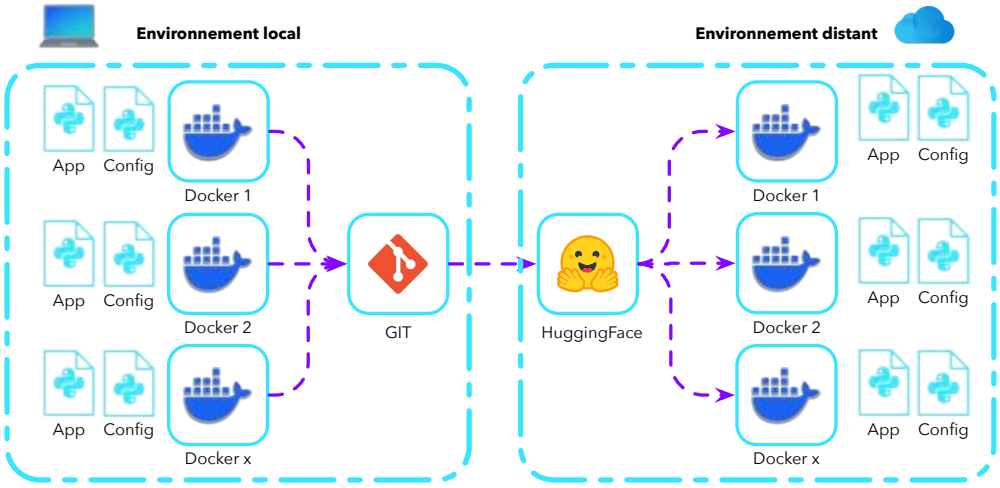
5.2. Déploiement.

Pour transformer une maquette fonctionnelle locale en une solution industrialisée et fonctionnelle, le déploiement des conteneurs Docker© vers HuggingFace© offre une approche efficace et intuitive.

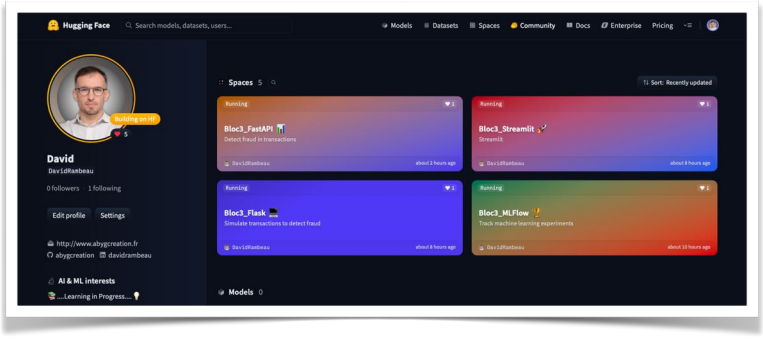
Une fois les conteneurs validés localement, ils sont poussés via un système de versionning vers HuggingFace©, une plateforme qui simplifie le déploiement d'applications web sans nécessiter de gestion d'infrastructure complexe.

Cela permet de centraliser les ressources et de les rendre accessibles aux équipes pour des tests ou des démonstrations.

⚠ *Cette plateforme est un environnement de test et permet de simuler un déploiement industriel à distance, il ne faut donc pas l'utiliser pour la mise en production.* ⚠



Cette approche facilite l'industrialisation du modèle, en permettant une intégration fluide avec des outils de CI/CD[6] (*voir livrable suivant Bloc4 - Construire, déployer et piloter des solutions d'IA*) pour automatiser les mises à jour et les déploiements.



[6] *Continuous Improvment/Continuous Delivery, methodes de travail en mode « Agile »*

6. Conception du système temps réel

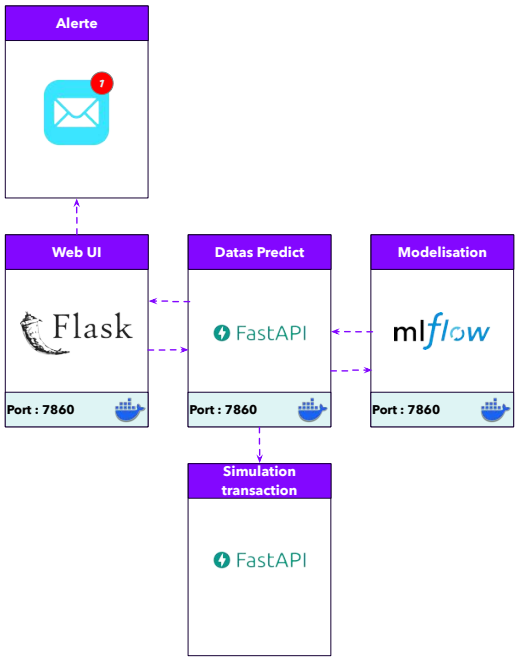
6.1. Architecture technique

Afin de concevoir un système de détection de fraude en temps réel, nous avons prolongé les travaux précédents en intégrant une API de simulation (*celle disponible dans le cas de cette étude*) de transactions, couplée à une interface web développée avec Flask.

Cette architecture repose sur une communication fluide entre les composants pour garantir une analyse instantanée des transactions, tout en offrant une visualisation interactive des résultats.

L'architecture technique combine :

- *Une API de simulation qui génère des transactions (conformes ou frauduleuses) et les envoie au système de détection via des requêtes HTTP.*
- *Un backend Flask qui expose une interface web (WebUI) pour simuler et visualiser les transactions en temps réel (statut, montant, heure).*
- *Un serveur mlFlow qui va nous permettre de récupérer le modèle de prédiction pour traiter la transaction en cours.*
- *Un système d'alerte pour effectuer l'envoi d'un mail lors de la détection d'une transaction frauduleuse*



Cette approche permet non seulement de valider le fonctionnement du modèle dans un environnement réaliste, mais aussi de faciliter l'utilisation par les équipes métiers grâce à une interface simple.

7. Conclusion.

Ce projet a permis de concevoir un pipeline complet de détection de fraude, depuis l'exploration des données jusqu'au déploiement temps réel, en passant par l'entraînement, l'optimisation et l'intégration des modèles.

Aujourd'hui il existe encore des axes d'améliorations possibles pour renforcer l'efficacité du système, notamment :

- *Cross-validation avancée pour éviter le surapprentissage et valider la robustesse du modèle.*
- *Optimisation des hyperparamètres en affinant les métriques.*
- *Exploration d'autres modèles pour améliorer la précision et la rapidité de détection.*
- *Mise en place d'un réapprentissage automatique et périodique pour adapter le modèle aux évolutions des schémas de fraude.*
- *Intégration de données externes pour enrichir les features et améliorer les prédictions.*

L'objectif ultime reste d'affiner la détection des fraudes pour la rendre plus précise et plus rapide, tout en garantissant une intégration fluide dans les systèmes existants.