

# RAPPORT DE STAGE DE FIN D'ÉTUDES

25 Rue Anatole France, 92300 Levallois-Perret

Sous le thème :

---

## Data Exploration & Machine Learning

---

Réalisé par : Mlle. Liuxin YANG

Encadré par : M. Yu ZHANG

01 Avril - 30 Septembre

Année universitaire 2024-2025

**Ce rapport est strictement confidentiel. Toute reproduction ou diffusion,  
même partielle, est interdite sans autorisation.**

## Remerciements

Je souhaite tout d’abord remercier mon manager pour son accompagnement exemplaire. Toujours bienveillant, il a pris le temps de répondre à nos questions, qu’elles portent sur des aspects techniques ou, plus largement, sur des points de langue française. Dès l’entretien, il a montré une maîtrise solide des fondements statistiques, qui m’a beaucoup inspirée pour la suite du stage. Au quotidien, son sens de l’humour a contribué à instaurer une atmosphère de travail agréable au sein de l’équipe. Attentif et compréhensif, il sait faire confiance et soutenir ses collaborateurs. En un mot, c’est un manager compétent et humain, dont l’appui a été déterminant pour la réussite de ce stage.

Je tiens à remercier tout particulièrement mon tuteur de stage, qui m’a initiée à plusieurs outils de la plateforme GCP et m’a encouragée à diversifier mes compétences techniques, notamment en explorant Dataflow et Airflow. Tout au long du stage, il m’a accompagnée dans mes apprentissages et m’a guidée pour progresser tant sur le plan technique que méthodologique.

Je remercie également chaleureusement mes collègues — Andy, Lorenzo, Robin, Sonia, Wentai et Xinyu — dont la gentillesse et la disponibilité m’ont beaucoup aidée au quotidien. Nous déjeunions ensemble, partagions des pauses et, à l’occasion, sortions en dehors du travail : des moments qui ont grandement facilité mon intégration. Leur sérieux et leur esprit d’entraide ont été une source d’inspiration. Merci à chacune et chacun pour cet accueil et cette collaboration exemplaires.

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>4</b>
<b>2</b>	<b>Présentation de l'Organisme d'Accueil</b>	<b>6</b>
2.1	LiveRamp Holdings, Inc. . . . .	6
2.2	Cadre de Travail . . . . .	6
<b>3</b>	<b>Environnement cloud : Google Cloud Platform</b>	<b>8</b>
3.1	BigQuery : Base de données . . . . .	8
3.2	Dataflow : pipeline distribué . . . . .	10
3.2.1	Architecture Dataflow . . . . .	10
3.2.2	Framework Apache beam . . . . .	10
3.2.3	Cloud service Dataflow . . . . .	12
3.3	Cloud Composer : Airflow . . . . .	13
3.3.1	Framework Apache Airflow . . . . .	13
3.3.2	Cloud service : Cloud Composer . . . . .	14
<b>4</b>	<b>Environnement interne : LiveRamp Safe Haven</b>	<b>15</b>
<b>5</b>	<b>Projets réalisés</b>	<b>17</b>
5.1	Mise en production d'outils analytiques . . . . .	17
5.1.1	Introduction à la notion de Clean Room . . . . .	17
5.1.2	Présentation de la plateforme Habu Clean Room . . . . .	17
5.1.3	Modèles de tableaux de bord . . . . .	18
5.2	Analyse de la qualité des données . . . . .	20
5.2.1	Introduction du jeu de données . . . . .	20
5.2.2	Problèmes de qualité des données identifiés . . . . .	20
5.2.3	Pipeline proposé . . . . .	21
5.3	Validation des données par NLP . . . . .	30
5.3.1	Contexte et objectif . . . . .	30
5.3.2	Construction de l'architecture . . . . .	30
5.3.3	Résultats . . . . .	34
<b>6</b>	<b>Conclusion</b>	<b>36</b>

7 Déroulement du Stage	38
Bibliographie	43

# 1 Introduction

Ce rapport présente une vue d'ensemble de mon stage de fin d'études, réalisé d'avril à septembre 2025 chez LiveRamp France, filiale de LiveRamp Holdings, Inc. J'ai intégré l'équipe Data Science France, dirigée par M. Sylvain Dupuis.

L'objectif principal de ce stage était de contribuer au développement de produits analytiques et à l'amélioration de la qualité des données. Les travaux se sont appuyés sur l'environnement Google Cloud Platform (BigQuery, Dataflow, Cloud Composer/Airflow) ainsi que sur les environnements internes de LiveRamp (Safe Haven et Habu Clean Room).

En tant que stagiaire data scientist, j'ai participé à plusieurs missions. Conformément aux objectifs décrits dans l'offre de stage, et en intégrant des volets plus innovants répondant directement aux besoins concrets de l'entreprise, mes travaux se sont articulés autour de trois grands axes :

- **Data assets et ingénierie** : développement de scripts Python/SQL dans GCP pour traiter, consolider et contrôler la qualité des données, avec documentation et automatisation des processus récurrents (orchestration via Airflow/Cloud Composer) ;
- **Data quality et machine learning** : conception d'un pipeline modulaire de nettoyage et validation en deux phases (format et sémantique) couvrant notamment codes-barres, marques, catégories et nomenclatures produit ; implémentations successives en `pandas` puis en mode distribué avec Apache Beam/Dataflow ; normalisation des marques (approche n-grams/similarité) et enrichissement par données publiques (Open \*Facts). Sur cette base, développement d'une brique ML/NLP pour la détection d'incohérences hiérarchiques : embeddings SentenceTransformers multilingues concaténés à des variables tabulaires, agrégés par un MLP léger pour la prédiction binaire `is_error` et, en option, la correction par niveau ;
- **Contribution aux activités de l'équipe** : validation et intégration des données entrantes, conception de tableaux de bord pour la restitution des insights<sup>1</sup>, ainsi que contrôle de la cohérence et de la fiabilité des insights délivrés au travers des plateformes LiveRamp.

Plusieurs livrables majeurs ont été produits :

1. la mise en production de tableaux de bord sur Habu Clean Room pour cinq comptes (le client Livraison (LATAM), le client Média (FR), le client Électronique, le client Cinéma (ES) et le client Assurance (UK)) ;
2. un pipeline de nettoyage/validation aligné sur les pratiques de gouvernance des données ;

---

1. Un insight désigne un résultat issu de l'analyse de données pertinentes, visant à comprendre, ajuster et améliorer les activités de l'entreprise.

3. une preuve de concept NLP multilingue, fondée sur des embeddings *Sentence-Transformers* et un MLP léger, pour détecter des incohérences hiérarchiques dans les variables produit ;
4. l'orchestration d'exécutions récurrentes et la réduction du travail redondant, améliorant la fraîcheur et la fiabilité des insights.

Au-delà des livrables, le stage a permis une montée en compétences sur GCP (Big-Query, Dataflow, Composer) et les bonnes pratiques de gestion de version (Git), tout en renforçant une approche rigoureuse de la qualité et de la gouvernance des données.

**Organisation du rapport.** La section suivante présente l'[organisme d'accueil](#) et le cadre de travail. Les sections consacrées à l'[environnement cloud \(GCP\)](#) et à l'[environnement interne \(Safe Haven\)](#) détaillent ensuite les outils mobilisés. La section [5.1](#) décrit la mise en production d'outils analytiques sur Habu Clean Room ; la section [5.2](#) expose le pipeline de qualité des données ; la section [5.3](#) présente l'approche NLP pour la validation des hiérarchies. Enfin, la section [7](#) retrace le déroulement hebdomadaire du stage.

## 2 Présentation de l'Organisme d'Accueil

Cette section est consacrée à la présentation de l'organisme d'accueil du stage : son implantation géographique, ses principaux services, ainsi que le cadre de travail et l'équipe dans laquelle je suis intégrée.

### 2.1 LiveRamp Holdings, Inc.

LiveRamp Holdings, Inc. (communément LiveRamp) est une entreprise américaine spécialisée dans les technologies SaaS (*Software as a Service* en anglais) de connectivité des données, fondée en 2011 à San Francisco. Elle possède des bureaux internationaux, l'un à Levallois-Perret, en France.

LiveRamp propose une plateforme complète de Data Collaboration permettant aux entreprises de relier, contrôler et activer les données de façon sécurisée. Ses services clés incluent :

- Onboarding des données : collecte et import de données clients en ligne et hors ligne, anonymisation, puis mapping vers un identifiant unique conçu par LiveRamp (*RampID*) pour faciliter l'analyse et l'activation marketing.
- Résolution d'identité : fusion de données multi-canal (CRM, web, offline) en profils unifiés, pour une vue client consolidée.
- Activation des données : convertir les données en segments activables et en tableaux de bord interactifs afin de maximiser la valeur des données marketing.
- Modélisation Lookalike : identification de nouvelles audiences présentant des comportements similaires aux segments existants, permettant d'élargir efficacement la portée marketing.
- Mesure cross-média : évaluation de l'impact des campagnes publicitaires via des audiences réconciliées sur TV, streaming, etc.

Au-delà de ces services technologiques, LiveRamp se distingue par ses atouts concurrentiels sur le marché. Grâce à son large réseau de partenaires (650+) et à ses plus de 900 clients globaux, LiveRamp offre une véritable force de diffusion et de performance marketing. Sa réputation repose aussi sur une infrastructure fiable et respectueuse des régulations, ainsi que sur un environnement favorable aux employés, reconnu par des certifications importantes.

### 2.2 Cadre de Travail

Ce stage s'est déroulé chez LiveRamp. L'entreprise occupe l'ensemble du deuxième étage d'un immeuble appelé Thaïs, situé à Levallois-Perret, en France.

J'ai intégré l'équipe DS (*Data Science*), qui compte au total six collaborateurs permanents : deux Senior Data Scientists, deux Data Analysts, un Data Engineer et un

manager. En parallèle, trois stagiaires – dont moi – ont rejoint l'équipe au cours de la même période, ce qui porte l'effectif global de l'équipe à neuf personnes.

Mon tuteur de stage occupe le poste de Data Engineer au sein de l'équipe Data Science. Son rôle consiste principalement à concevoir et à maintenir des pipelines de données automatisés sur la plateforme Google Cloud Platform, tout en assurant la supervision et la maintenance régulière des serveurs cloud. En complément de ces missions d'ingénierie, il développe également des scripts et du code destinés à être utilisés par ses collègues à des fins d'analyse commerciale.

Le cadre de travail est particulièrement agréable. Nous partageons un même bureau, ce qui facilite la collaboration et les échanges au quotidien. De plus, nous sommes entourés d'autres équipes avec lesquelles nous interagissons régulièrement, notamment l'équipe PO (*Product Operation*) et l'équipe CSM (*Customer Success Manager*). L'entreprise impose un minimum de trois jours de présence sur site chaque semaine. Afin de favoriser la cohésion, notre équipe s'efforce d'être présente les mêmes jours autant que possible.

Au cours de ce stage de six mois, j'ai également eu l'opportunité de participer à une activité de team building organisée par l'entreprise. Cette journée s'est déroulée sous la forme d'une randonnée urbaine à Paris, ponctuée de cinq points de passage comportant différents défis et missions. Cette expérience m'a permis de mieux connaître mes collègues dans un cadre informel et de renforcer la cohésion de l'équipe.



### 3 Environnement cloud : Google Cloud Platform

Pendant le stage, le travail est principalement effectué dans l'environnement cloud Google Cloud Platform (GCP). Il s'agit d'une plateforme de cloud computing proposée par Google. Elle fait partie de l'ensemble des solutions Google Cloud et fournit aux entreprises des services modulaires hébergés dans le cloud.

GCP regroupe une large famille de produits, accessibles via une interface web, une ligne de commande et des API. Au cours de mon stage, j'ai eu l'opportunité d'utiliser plusieurs de ces services, notamment : Cloud Storage (où LiveRamp stocke les bases de données clients), BigQuery, Dataflow et Composer. Dans les sous-sections suivantes, chacun de ces outils sera présenté plus en détail.

#### 3.1 BigQuery : Base de données

BigQuery est un service intégré à GCP qui permet l'analyse interactive massive de grands ensembles de données en collaboration avec le Cloud Storage.

La capture d'écran 1 montre la console BigQuery sur GCP.

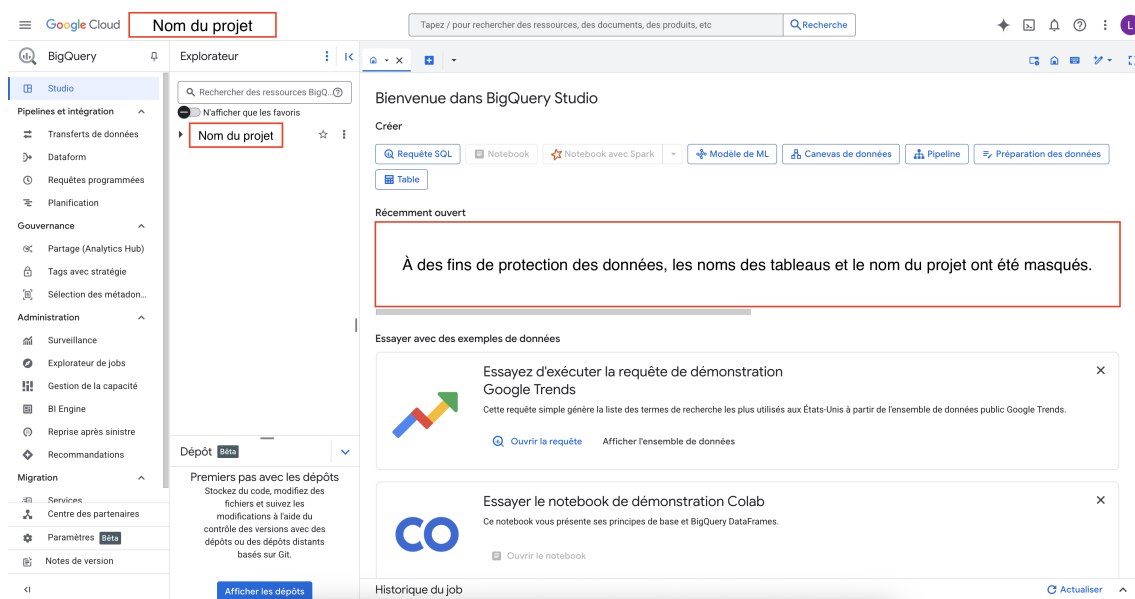


FIGURE 1 – Console BigQuery

BigQuery est également un entrepôt de données d'entreprise de Google, en mode sans serveur, donc sans infrastructure à gérer. Les requêtes sont écrites en SQL. Il dispose également de capacités d'apprentissage automatique intégrées.

Un entrepôt logique est constitué de bases de données, lesquelles contiennent à leur tour des tables ou des vues. Ainsi, une table est généralement identifiée sous la forme

suivante :

`projet.dataset.table.`

L'une des caractéristiques spécifiques de BigQuery est son modèle de stockage en colonnes.

**Stockage en colonnes** Les bases de données traditionnelles stockent les données dans un format orienté ligne, ce qui les rend performantes en matière de mises à jour transactionnelles. Par exemple, si l'on veut modifier l'état d'un produit, il suffit de lire une seule ligne. Cependant, pour effectuer des opérations telles que l'agrégation, il faut lire l'intégralité du tableau.

Dans ce contexte, le service BigQuery est optimisé pour stocker les données en colonnes dans un format nommé *Capacitor*. Un avantage supplémentaire des bases orientées colonnes est que les données d'une même colonne présentent souvent plus de redondance que celles d'une ligne. Cette caractéristique permet d'améliorer la compression des données en utilisant des techniques telles que l'encodage de longueur d'exécution.

#### Encodage de longueur d'exécution

L'encodage de longueur d'exécution (appelé en anglais *Run-Length Encoding*/RLE) est un algorithme de compression sans perte qui consiste à remplacer les suites de valeurs identiques par une paire (**valeur**, **nombre de répétitions**).

**Important :** RLE ne réorganise pas les données, il compresse uniquement les valeurs identiques consécutives, ce qui permet de préserver l'ordre et donc l'alignement entre colonnes.

##### Tableau d'origine

Q1	0
Q2	1
Q2	1
Q3	0
Q1	2
Q1	2

##### Séquence compressée (RLE)

Q1	(0, 1)
Q2	(1, 2)
Q3	(0, 1)
Q1	(2, 2)

##### Après décompression

Q1	0
Q2	1
Q2	1
Q3	0
Q1	2
Q1	2

Deux optimisations supplémentaires sont possibles pour le stockage :

- Les partitions sont conçues pour les cas où il y a une grande quantité de données et un faible nombre de valeurs distinctes, et elles sont créées à l'aide de la commande « **PARTITION BY** ». Une fois le tableau créé, la façon de partitionnement est fixée.

- Au contraire, les regroupements (ou « clustering ») sont conçus pour les cas où la cardinalité est élevée ou où le tableau est de petite taille. Ils sont créés à l'aide de la commande « `CLUSTER BY` ». Il est toutefois possible de modifier les clusters après la création du tableau.

## 3.2 Dataflow : pipeline distribué

### 3.2.1 Architecture Dataflow

L'architecture *dataflow* est un modèle de programmation dans lequel les données jouent un rôle actif : leur circulation à travers le programme déclenche directement l'exécution des opérations. Ce paradigme se distingue ainsi fondamentalement de l'architecture traditionnelle von Neumann, dans laquelle les données restent passivement stockées en mémoire et les instructions sont exécutées de manière strictement séquentielle selon un pointeur de programme.

Dans le modèle de *dataflow*, les traitements sont représentés sous la forme d'un graphe orienté, dont les nœuds correspondent aux opérations à effectuer. Les données circulent sur les arcs reliant ces nœuds, formant ainsi les entrées nécessaires à chaque opération. Une règle fondamentale de ce modèle stipule que dès qu'un nœud reçoit toutes les données en entrée nécessaires, il s'active, exécute son opération, produit une sortie et libère ensuite ses entrées.

Ce modèle est étroitement couplé aux langages de programmation fonctionnels. Il ne nécessite pas de mémoire partagée. En outre, les architectures *dataflow* facilitent naturellement le parallélisme : plusieurs opérations peuvent être exécutées simultanément. De ce fait, ces architectures exploitent pleinement les capacités des systèmes distribués ou des processeurs.

Afin de mettre en œuvre ce modèle dans des environnements concrets, des frameworks comme Apache Beam ont été développés.

### 3.2.2 Framework Apache beam

Apache Beam est un framework open-source définissant un modèle de programmation unifié pour les pipelines de traitement de données. Un pipeline Beam peut être exécuté sur divers environnements appelés *runners*, tels que [Google Cloud Dataflow](#), Apache Spark, Apache Flink ou même un exécuteur local (DirectRunner).

Apache Beam propose plusieurs SDK (*Software Development Kits*), notamment en Java, Python et Go, et supporte également des pipelines multilingues. Cela permet aux développeurs de choisir le langage le mieux adapté à leurs besoins et d'éviter d'avoir à réécrire le code du pipeline pour l'exécuter sur différentes plateformes. De plus, le modèle Beam isole des détails de bas niveau du traitement distribué, tels que la segmentation des données ou la coordination des nœuds de calcul.

Pour mieux comprendre l'approche d'Apache Beam, il est essentiel de présenter quelques concepts fondamentaux qui structurent la construction et l'exécution des pipelines.

### 1. Pipeline

Un pipeline est une description déclarative de l'ensemble des étapes de traitement. Il ne s'agit pas d'une structure mutable mais d'un graphe logique que l'on peut exécuter plusieurs fois avec différents paramètres d'entrée ou de sortie. Chaque pipeline représente donc une tâche complète de traitement de données.

### 2. PCollection

Une **PCollection** représente un ensemble (potentiellement distribué) d'éléments. Elle peut être bornée (données de lot finies) ou non bornée (flux de données continues), ce qui permet à Apache Beam de supporter à la fois le batch et le streaming.

### 3. Transformations

Une transformation représente une opération de traitement qui transforme les données, appelée **PTransform**.

### 4. ParDo et DoFn

**ParDo** est une **PTransform** permettant d'appliquer une fonction personnalisée à chaque élément d'une **PCollection**. Cette fonction personnalisée est définie par une classe dérivant de **DoFn** (*Do Function*).

La classe **DoFn** encapsule ainsi la logique métier spécifique à appliquer à chaque élément. Elle implémente notamment la méthode **process()** exécutée sur chaque élément reçu, capable de retourner un ou plusieurs résultats via l'instruction **yield**.

### 5. CombineFn

**CombineFn** permet de définir des opérations d'agrégation personnalisées sur une **PCollection** (par exemple : somme, moyenne, comptage). La logique d'agrégation est définie à travers quatre méthodes essentielles :

- **create\_accumulator()** : initialise une structure d'accumulation vide.
- **add\_input()** : ajoute un nouvel élément à l'accumulateur.
- **merge\_accumulators()** : fusionne plusieurs accumulateurs partiels en un seul.
- **extract\_output()** : extrait le résultat final de l'accumulateur.

Voici ainsi un exemple de pipeline Apache Beam :

```
1 with beam.Pipeline(options=options) as Pcollection:
2     data = (
3         Pcollection
4         | "Read Cleaned Dataset from BigQuery" >>
5           beam.io.ReadFromBigQuery(table = clean_table)
6         | "Handle missing local brand names" >>
7           beam.ParDo(handle_missing_brand("local_brand_name"))
8         | "Handle missing global brand names" >>
9           beam.ParDo(handle_missing_brand("global_brand_name"))
10        | "Standardize Contents of Columns" >>
11          StandardizeByFrequence(std_cols, all_columns)
12    )
```

### 3.2.3 Cloud service Dataflow

Dataflow est un service entièrement managé par Google Cloud, qui fournit un traitement unifié des données à grande échelle, qu'elles soient en flux continu (*streaming*) ou par lots (*batch*). Dataflow repose directement sur le framework [Apache Beam](#) et en constitue l'un des principaux moteurs d'exécution.

Lorsqu'un pipeline est soumis et que Cloud Dataflow est utilisé comme runner, Dataflow provisionne automatiquement les ressources nécessaires (machines virtuelles, workers). Il suffit de spécifier des paramètres d'exécution comme le nombre initial de workers (`num_workers`) et le nombre maximal de workers (`max_num_workers`).

De plus, Dataflow s'intègre nativement avec l'écosystème Google Cloud (BigQuery, Cloud Storage, etc.). Les jobs utilisent Cloud Storage pour stocker les fichiers temporaires (par ex. logs, états intermédiaires). Un exemple de ces fichiers temporaires est présenté dans la capture d'écran [2](#).

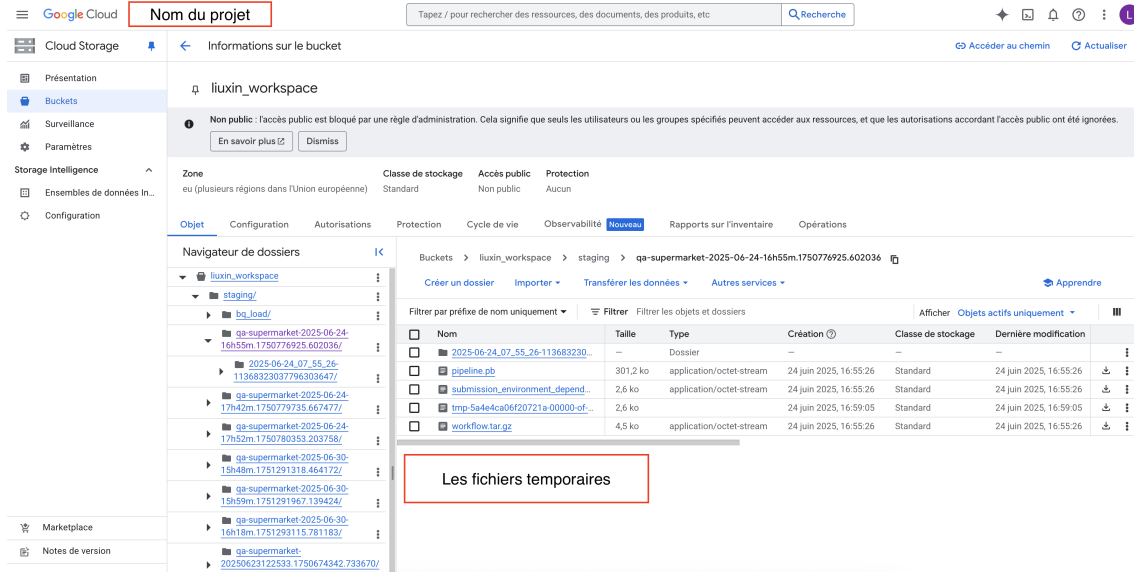


FIGURE 2 – Exemple de fichiers temporaires générés par Dataflow dans Cloud Storage

### 3.3 Cloud Composer : Airflow

#### 3.3.1 Framework Apache Airflow

Apache Airflow est une plateforme open source de gestion et d'orchestration de flux de travail (*workflows*)<sup>2</sup>. L'interface utilisateur d'Apache Airflow est présentée dans la capture d'écran 3.

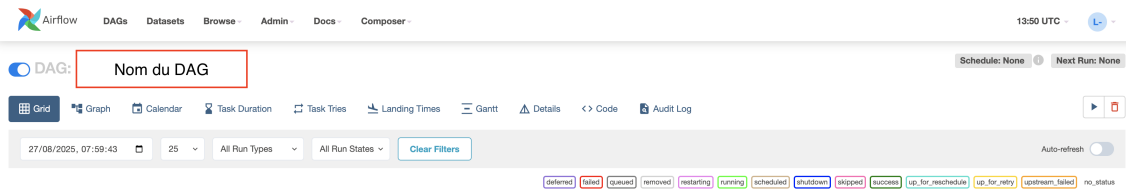


FIGURE 3 – UI Apache Airflow

Dans Airflow, les workflows sont créés à l'aide de graphes acycliques dirigés (DAG).

Un DAG est un ensemble de tâches à programmer et à exécuter, organisées de manière à refléter leurs relations et leurs dépendances. Une tâche peut effectuer l'une des fonctions suivantes :

- Préparer des données pour l'ingestion

2. Un workflow est une série de tâches, lancée selon un calendrier ou déclenchée par un événement, fréquemment utilisée pour gérer les pipelines de traitement des mégadonnées.

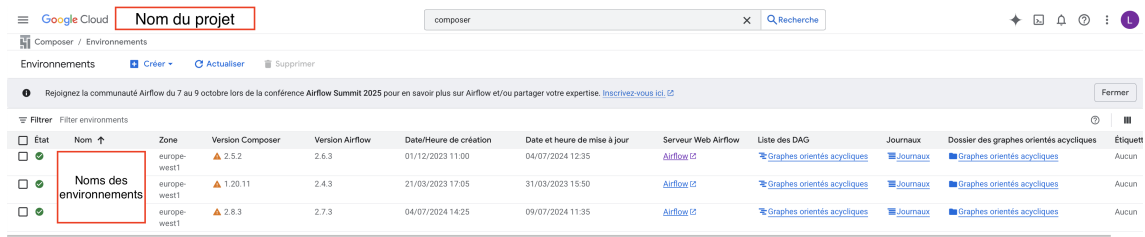
- Surveiller une API
- Envoyer un e-mail
- Exécuter un pipeline

De plus, les DAG peuvent être exécutés selon un calendrier défini (par exemple, tous les jours ou toutes les semaines) ou en fonction de déclencheurs d'événements externes, comme des modifications apportées à un bucket Cloud Storage.

### 3.3.2 Cloud service : Cloud Composer

Cloud Composer est un tel service d'orchestration de workflows entièrement géré par Google Cloud et s'intègre bien aux autres services GCP. Il est basé sur le projet Open Source Apache Airflow et fonctionne avec le langage de programmation Python.

Il est possible de créer un ou plusieurs environnements dans un même projet Google Cloud, comme illustré dans la capture d'écran 4.



État	Nom	Zone	Version Composer	Version Airflow	Date/Heure de création	Date et heure de mise à jour	Serveur Web Airflow	Liste des DAG	Journaux	Dossier des graphes orientés acycliques	Étiquettes
<input type="checkbox"/>	Noms des environnements	europa-west1	2.5.2	2.6.3	01/12/2023 11:00	04/07/2024 12:35	<a href="#">Airflow</a>	<a href="#">Graphes orientés acycliques</a>	<a href="#">Journaux</a>	<a href="#">Graphes orientés acycliques</a>	Aucun
<input type="checkbox"/>		europa-west1	1.20.11	2.4.3	21/03/2023 17:05	31/03/2023 15:50	<a href="#">Airflow</a>	<a href="#">Graphes orientés acycliques</a>	<a href="#">Journaux</a>	<a href="#">Graphes orientés acycliques</a>	Aucun
<input type="checkbox"/>		europa-west1	2.8.3	2.7.3	04/07/2024 14:25	09/07/2024 11:35	<a href="#">Airflow</a>	<a href="#">Graphes orientés acycliques</a>	<a href="#">Journaux</a>	<a href="#">Graphes orientés acycliques</a>	Aucun

FIGURE 4 – Page d'accueil du service Cloud Composer

## 4 Environnement interne : LiveRamp Safe Haven

Outre l'environnement cloud, notre équipe opère également dans un environnement interne chez LiveRamp.

LiveRamp Safe Haven est une plateforme sécurisée, conçue pour favoriser la collaboration autour de données pseudonymes (via RampID), tout en garantissant la confidentialité. Elle permet de créer des audiences, d'activer des données auprès de partenaires, d'analyser l'impact des campagnes publicitaires et de générer des insights avancés.

La capture d'écran 5 montre la page d'accueil de cette plateforme.

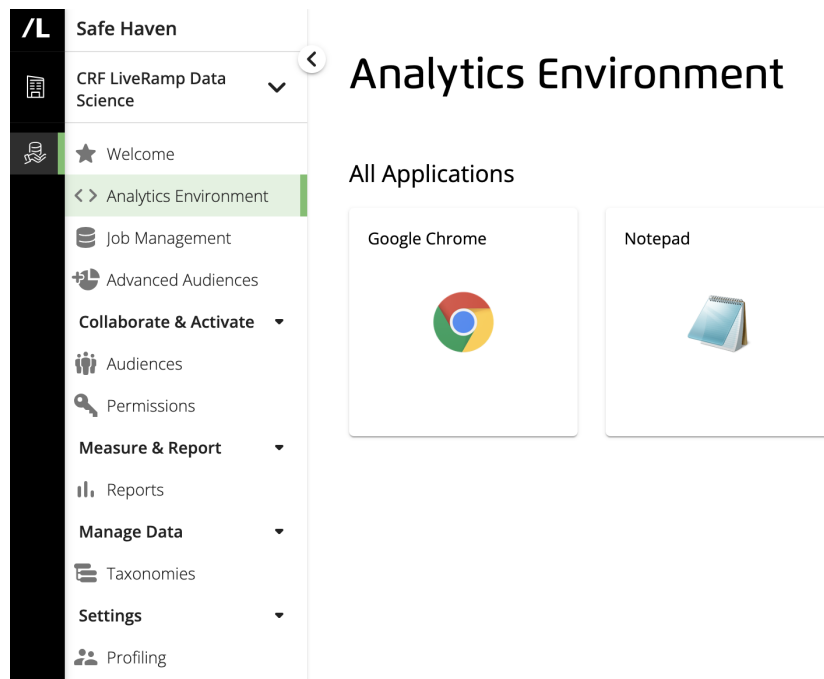


FIGURE 5 – Page d'accueil de la plateforme Safe Haven

Dans le cadre de mon stage, l'onglet **Analytics Environment** est principalement utilisé. Cet environnement se présente comme un poste de travail virtuel et sécurisé, destiné aux Data Scientists et aux Analysts. Il propose des outils tels que Jupyter, BigQuery, ou Dataproc Spark, comme illustré dans la capture d'écran 6. Ces outils permettent la construction de segments, la manipulation de données et l'analyse statistique en toute sécurité.



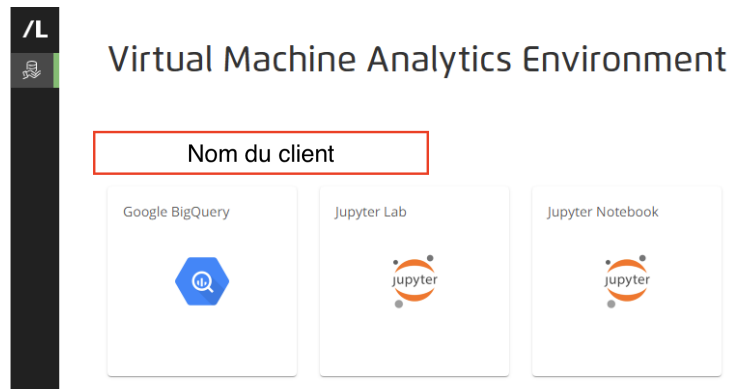


FIGURE 6 – Page d'accueil de l'onglet **Analytics Environment**

Au sein de cet environnement, l'accès à Internet est restreint et il n'est pas possible de copier les données vers l'extérieur. Les traitements sont effectués directement sur une machine virtuelle distante, ce qui garantit à la fois la confidentialité et la gouvernance des données.

## 5 Projets réalisés

Cette section est consacrée à la présentation des trois projets réalisés dans le cadre du stage : un projet opérationnel [5.1](#) et deux projets de recherche [5.2](#) et [5.3](#).

### 5.1 Mise en production d'outils analytiques

#### 5.1.1 Introduction à la notion de Clean Room

La protection des données constitue un enjeu majeur dans le traitement des données. Traditionnellement, les données clients sont exposées directement aux data analysts, ce qui dissuade les entreprises de faire appel à un tiers pour réaliser des analyses, car elles doivent alors partager leurs données. Grâce aux *Data Clean Rooms*, le croisement des données est désormais possible pour plusieurs acteurs, sans jamais exposer les données brutes à l'une des parties, garantissant ainsi un haut niveau de confidentialité. Ces environnements sécurisés permettent de partager, d'exploiter, de tracer, de stocker et de conserver les données de première partie en conformité avec les réglementations en vigueur.

Les *Data Clean Rooms* se sont imposées comme une solution technologique de référence pour permettre une collaboration sur des données réparties, tout en respectant les exigences liées à la vie privée. Elles offrent une réelle valeur ajoutée en permettant aux entreprises d'optimiser leurs chaînes d'approvisionnement, de détecter les fraudes, de réduire les risques et d'adapter leurs stratégies marketing afin d'améliorer l'expérience client.

En janvier 2024, LiveRamp a annoncé l'acquisition de Habu, une plateforme de *Data Clean Room* reconnue pour sa capacité à faciliter la collaboration sur des données décentralisées de manière sécurisée. Grâce à cette acquisition, LiveRamp dispose désormais d'une plateforme unique et simple pour mesurer la performance des campagnes publicitaires, et connecter les données de manière transparente à travers les clouds, les entrepôts de données et les *Clean Rooms*, y compris Amazon Web Services, Google Cloud, Snowflake, Azure et Databricks.

#### 5.1.2 Présentation de la plateforme Habu Clean Room

La capture d'écran [7](#) ci-dessous illustre la page d'accueil de la plateforme Habu Clean Room dédiée à l'équipe DS.

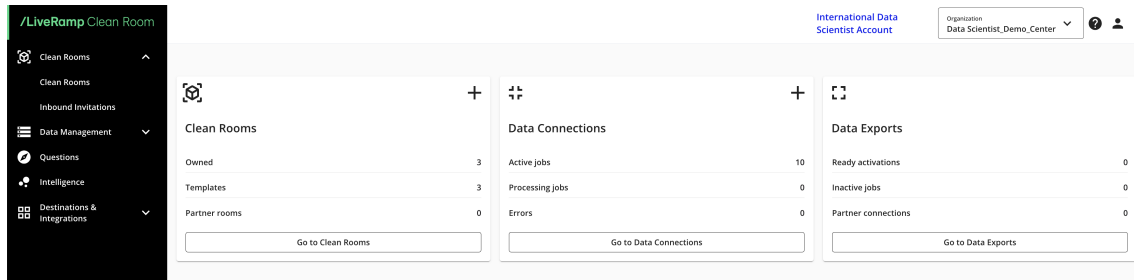


FIGURE 7 – Page d'accueil de la plateforme Habu Clean Room

Dans cette plateforme, les données elles-mêmes restent invisibles. En revanche, les champs des jeux de données peuvent être consultés dans l'onglet **Data Management**.

### 5.1.3 Modèles de tableaux de bord

Pour des raisons de protection des données clients, un modèle de démonstration très simplifié est présenté dans la capture d'écran 8.

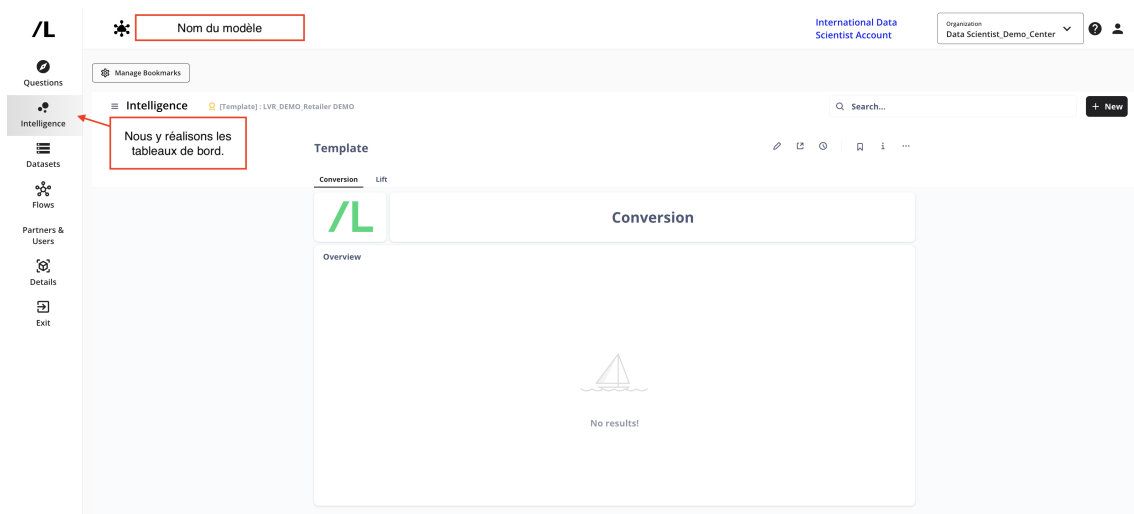


FIGURE 8 – Modèle de tableaux de bord

La construction d'un tableau de bord sur cette plateforme repose sur plusieurs étapes.

- D'abord, la formulation de *questions* dans l'onglet **Questions**, correspondant à des requêtes SQL permettant de calculer les indicateurs clés de performance (KPI).

- Ensuite, l'exécution coordonnée de ces requêtes, soit manuellement une par une, soit de manière simultanée grâce à un script Airflow, afin de garantir la cohérence et la mise à jour des résultats.
- Puis, les résultats peuvent être enrichis par des questions avancées dans l'onglet **Intelligence**, qui permettent d'appliquer davantage d'opérations que les questions de base.
- Enfin, l'ensemble de ces résultats alimente les tableaux de bord, qui constituent l'outil final de suivi et de visualisation des données.

Dans le cadre de mon stage, j'ai eu l'opportunité de contribuer à l'ensemble du processus, depuis son initiation jusqu'à son achèvement. Mes responsabilités ont inclus le développement et la correction de requêtes SQL (Snowflake et Hybride) en étroite collaboration avec Xinyu. J'ai également conçu, réalisé et validé des tableaux de bord pour cinq clients, à savoir le client Livraison (LATAM), le client Média (FR), le client Électronique, le client Cinéma (ES) et le client Assurance (UK). En outre, j'ai pris part à la mise en place de l'exécution automatisée des requêtes via Airflow aux côtés de Yu.

Ces travaux ont permis de garantir la cohérence des insights livrés, de réduire le travail redondant et d'assurer une actualisation illimitée et fiable des tableaux de bord.

L'utilisation directe de ces tableaux par les clients a constitué une expérience particulièrement enrichissante. Leurs retours m'ont permis de mieux comprendre les besoins opérationnels, d'affiner mes compétences en visualisation et en validation de données, et de me confronter aux enjeux concrets de la mise en production d'outils analytiques.

## 5.2 Analyse de la qualité des données

L'analyse de la qualité des données constitue une étape essentielle, car elle établit une base solide pour les analyses commerciales ultérieures et permet d'éviter un travail redondant lié à la découverte tardive de problèmes.

Ce processus vise à identifier et à corriger les erreurs, les lacunes et les incohérences présentes dans les jeux de données avant toute exploitation analytique ou modélisation.

Il consiste notamment à vérifier les erreurs manquantes, les doublons, les incohérences, les formats non conformes, les valeurs aberrantes, les informations erronées ainsi que la validité des métadonnées.

### 5.2.1 Introduction du jeu de données

L'analyse a principalement porté sur le jeu de données de référence des produits du client Retailer France. Ce tableau partitionné comptait environ cinq millions de lignes au 30 juin 2025 et regroupait 55 variables, réparties comme suit :

- **2 clés primaires** ;
- **1 variable supplémentaire relative au code-barres** ;
- 7 variables décrivant les informations du fournisseur ;
- 4 variables représentant l'état du produit ;
- 6 variables caractérisant le produit ;
- **11 variables hiérarchiques** ;
- 6 variables relatives à l'unité et à la capacité du produit ;
- **6 variables indiquant les marques du produit** ;
- 5 variables décrivant la typologie du produit ;
- 4 variables associées au code de sensibilité ;
- 3 autres variables.

En pratique, les plus déterminantes pour l'analyse sont les clés primaires, les variables relatives au code-barres, les variables hiérarchiques ainsi que celles indiquant les marques du produit.

Toutefois, j'ai eu la possibilité d'examiner deux jeux de données complémentaires, provenant du client Retailer (BE) et du client Cinéma (ES), afin de compléter l'étude d'analyse.

### 5.2.2 Problèmes de qualité des données identifiés

Après l'étude descriptive des jeux de données, nous examinerons les problèmes de qualité mis en évidence dans ces jeux de données.

**Contrôles sans anomalies** Parmi les contrôles effectués, aucun problème n'a été détecté concernant :

- la cohérence entre les métadonnées et les données réelles,
- la détection des lignes en doublon,
- la vérification de l'unicité des clés primaires.

**Contrôles avec anomalies** En revanche, l'analyse de deux jeux de données supplémentaires a mis en évidence plusieurs problèmes de qualité, que nous pouvons regrouper en trois catégories :

- **Problèmes de format**
  - \* format non conforme des codes-barres (la longueur doit être égale à 13),
  - \* hétérogénéité dans l'usage des majuscules et des minuscules,
  - \* présence d'accents dans certaines valeurs,
  - \* problèmes d'encodage lors des conversions : les caractères accentués en espagnol n'ont pas été correctement interprétés et apparaissent sous la forme de « ? ».
- **Problèmes liés aux valeurs textuelles**
  - \* multiples variantes pour désigner une même marque (par ex. « L'OREAL » vs « L OREAL »),
  - \* fautes d'orthographe dans certaines entrées textuelles (par ex. « LA COURONN » vs « LA COURONNE »).
- **Problèmes de valeurs manquantes**
  - \* multiples de variantes des valeurs manquantes (par ex. « », « N/A », « null » convertis en NULL),
  - \* multiples variantes textuelles employées pour désigner l'absence de marque (par ex. « AUTRES MARQUES », « SANS MARQUE », « NO NAME », « UNKNOWN », « 1 »).

### 5.2.3 Pipeline proposé

Afin de répondre aux problèmes listés dans la sous-section précédente, un pipeline d'analyse a été conçu.

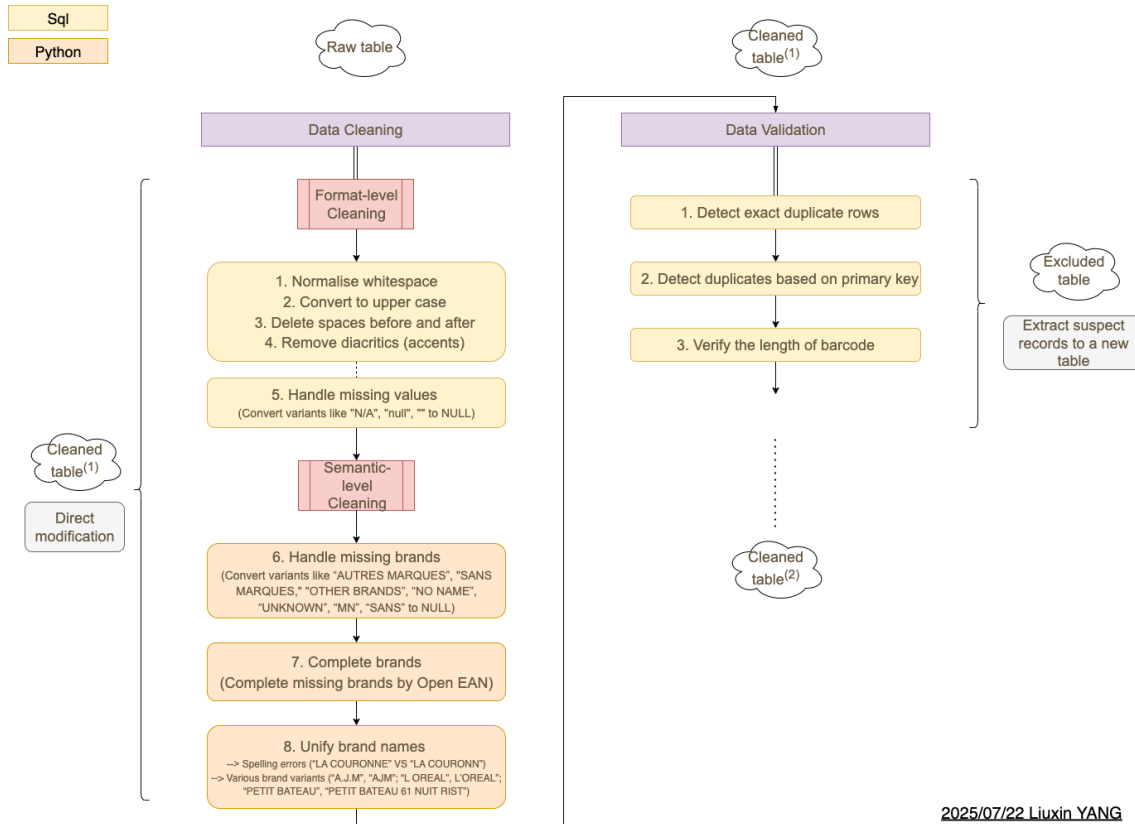


FIGURE 9 – Schéma global du pipeline de nettoyage et de validation

**Architecture globale** La figure 9 ci-dessus présente l’architecture globale du pipeline conçu. D’après le schéma, le pipeline se décompose en deux grandes étapes complémentaires : le nettoyage des données et la validation des données.

La phase de **Data Cleaning** commence à partir de la table brute.

Au niveau du format, les opérations suivantes ont été faites :

1. normalisation des espaces,
2. mise en majuscules,
3. suppression des espaces superflus,
4. remplacement des accents,
5. conversion de toutes les représentations techniques de valeurs absentes.

Au niveau sémantique, plusieurs actions sont effectuées :

6. harmonisation des multiples variantes textuelles employées pour désigner l’absence de marque,
7. complétion des marques à l’aide des produits publiques disponibles sur les sites [open PRODUCTS facts](#), [open FOOD facts](#), [open BEAUTY facts](#) et [open PET FOOD facts](#),

8. unification des noms de marques afin de corriger les fautes d'orthographe et regrouper les variantes d'un même libellé.

Le résultat produit une première table nettoyée **Cleaned table**<sup>(1)</sup>. Sur cette base, la phase de **Data Validation** exécute des contrôles de validation : détection des doublons exacts, détection des doublons selon la clé primaire et vérification de la longueur des codes-barres.

Les enregistrements suspects sont extraits dans la table distincte **Excluded table**, tandis que les données validées alimentent la table nettoyée finale **Cleaned table**<sup>(2)</sup>.

Ce pipeline de deux phases assure à la fois l'harmonisation du format et la cohérence sémantique des données, tout en garantissant la fiabilité structurelle des enregistrements conservés.

**Implémentation (Pandas)** Dans une première implémentation de la partie sémantique du pipeline, nous avons utilisé un enchaînement fonctionnel basé sur `DataFrame.pipe(...)` afin d'exprimer une suite de transformations :

```
1 clean_data = (  
2     obtain_dataframe(cfg, client, cfg.clean_table)  
3     .pipe(handle_missing_brand, brands)  
4     .pipe(complete_brand, open_ean_data)  
5     .pipe(unify_brand, brands, cfg, client, if_mapping=True,  
6           mapping_name="mapping_supermarket")  
7 )
```

Parmi les différents modules du pipeline, le dernier (**Unify brand names**) s'est révélé être le plus coûteux en temps de calcul lors de la première implémentation avec le package **pandas**. Cette étape vise à regrouper et harmoniser les marques en corrigeant les variantes orthographiques et les incohérences d'écriture.

Afin de réduire significativement le temps d'exécution, une procédure optimisée a été développée, dont le principe est présenté ci-dessous sous forme de pseudo-code.



---

**Algorithm 1** unify\_brand(df : DataFrame, columns : LIST, if\_mapping : BOOLEAN, mapping\_name : STRING)

---

```

1: mapping_similarity ← DICT[ ]
2: values_all ← SET[ ]
3: if if_mapping then
4:   mapping_similarity ← Read BigQuery table (mapping_name)
5:   for col in columns do
6:     df[col] ← extract_root(df[col])
7:     df[col] ← Apply mapping_similarity to df[col]
8:   end for
9: else
10:  for col in columns do
11:    df[col] ← extract_root(df[col])
12:    values_all ← Extract all the brands (df[col])
13:  end for
14:  frequency ← Count occurrences of values in values_all
15:  brands ← LIST (values_all)
16:  mapping_similarity ← generate_mapping_ngram(brands, frequency, n,
    min_share, threshold)
17:  for col in columns do
18:    df[col] ← df[col].apply(mapping_similarity)
19:  end for
20:  Save mapping_similarity to the BigQuery
21: end if
22: return df

```

---

Le principe général repose sur les étapes suivantes :

1. prétraitement des valeurs (normalisation, extraction des racines de chaînes) ;
2. génération d'un dictionnaire de similarité entre marques en utilisant une approche par *n-grams* ;
3. choix systématique d'une forme de référence pour chaque paire de marques similaires, en s'appuyant sur leur fréquence d'apparition.

Ces différentes fonctions auxiliaires

- extract\_root,
- get\_ngrams,
- choose\_most\_frequency,
- generate\_mapping\_ngram

permettent de construire un mapping efficace appliqué ensuite à l'ensemble du DataFrame. Leur description détaillée est également donnée ci-dessous.

---

**Algorithm 2** generate\_mapping\_ngram(brands : LIST, frequency : DICT, n, min\_share : INT, threshold : FLOAT) → DICT

---

```

1: mapping ← DICT[ ]
2: checked ← SET[ ]
3: brands_ngrams ← DICT[brand : set of grams]
4: ngram_to_brands ← DICT[ngram : set of brands]
5: for each brand b in brands do
6:   grams ← get_ngrams(b, n)
7:   brands_ngrams[b] ← grams
8:   for each gram in grams do
9:     Add (b) to ngram_to_brands[gram]
10:  end for
11: end for
12: for each brand b in brands do
13:   grams ← brands_ngrams[b]
14:   candidates_counts ← DICT[ ]
15:   for each gram in grams do
16:     for each brand another in ngram_to_brands[gram] do
17:       if another = b then
18:         break
19:       end if
20:       paire ← tuple(trier([b, another]))
21:       if paire in checked then
22:         break
23:       end if
24:       candidates_counts[autre] += 1
25:     end for
26:   end for
27:   candidates ← { another : c in candidates_counts if c ≥ min_share }
28:   for each another in candidates do
29:     paire ← tuple(trier([b, another]))
30:     Add paire to checked
31:     score ← Calculate similarity between b and autre
32:     if score ≥ threshold then
33:       std ← choose_most_frequency(frequency, b, autre)
34:       mapping[b] ← std
35:       mapping[autre] ← std
36:     end if
37:   end for
38: end for
39: return mapping

```

---

---

**Algorithm 3**  $\text{get\_ngrams}(\text{text} : \text{STRING}, n : \text{INT}) \rightarrow \text{SET}$

---

```

1: text  $\leftarrow$  text in uppercase
2: if length(text) > n then
3:   return SET(all substrings of length  $n$ )
4: else
5:   return SET(text)
6: end if

```

---



---

**Algorithm 4**  $\text{choose\_most\_frequency}(\text{frequency} : \text{DICT}, \text{val1}, \text{val2} : \text{STRING}) \rightarrow \text{STRING}$

---

```

1: freq1  $\leftarrow$  frequency.get(val1, 0)
2: freq2  $\leftarrow$  frequency.get(val2, 0)
3: if freq1 > freq2 then
4:   return val1
5: else if freq1 < freq2 then
6:   return val2
7: else
8:   return the value with the greatest length among val1, val2
9: end if

```

---

La capture d'écran [10](#) illustre les traces d'exécution du programme, montrant le temps de calcul associé à chaque étape du pipeline.

```

Step 1: Data Cleaning...
Running Format-level Cleaning...
  1. Normalise whitespace
  2. Convert to upper case
  3. Delete spaces before and after
  4. Remove diacritics (accents)
  5. Handling missing values

--> Format-level cleaning finished in 6.44 seconds.

Running Semantic-level Cleaning...
  6. Handle missing brand names
  7. Complete brands
  8. Unify brand names

Handle missing brands completed in 0.09 seconds.
Complete brands completed in 4.29 seconds.

      Unify N°1: Brand extraction completed in 3.07 seconds.

      Unify N°2: Mapping by similarity completed in 160.62 seconds.

      Unify N°3: Mapping saved in 2.37 seconds.

Unify brands completed in 166.06 seconds.
--> Semantic-level cleaning finished in 219.80 seconds.

Step 2: Data validation...

Pipeline finished.

```

FIGURE 10 – Traces d'exécution du programme

Afin de faciliter la lecture, le tableau 1 ci-dessous récapitule la durée d'exécution de chaque module.

TABLE 1 – Temps d'exécution par étape du pipeline

Étape	Temps (s)
Format-level Cleaning (étapes 1–5)	6,44
Handle missing brands (étape 6)	0,09
Complete brands (étape 7)	4,29
Unify brands (étape 8)	166,06
Brand extraction	3,07
Mapping by similarity	160,62
Mapping saved	2,37
Semantic-level Cleaning (étapes 6–8)	219,80

Nous pouvons constater que le temps d'exécution du module `Unify_brand_name` a été réduit de près de 70 %, passant d'environ 10 minutes lors de la première exécution à 166,06 secondes (soit environ 2,76 minutes).

Le tableau 2 compare deux exécutions : la première avec génération complète du mapping des marques, et la seconde avec simple lecture d'un mapping déjà existant.

*Remarque.* Bien que la durée exacte varie d'une exécution à l'autre, l'ordre de grandeur des temps de calcul reste stable.

TABLE 2 – Comparaison des temps d'exécution (génération du mapping vs. lecture du mapping)

Étape	Génération du mapping	Lecture du mapping
Format-level cleaning (1–5)	6,44 s	6,65 s
Handle missing brands (6)	0,09 s	0,08 s
Complete brands (7)	4,29 s	4,22 s
Unify brand names (8)	166,06 s	173,91 s
Semantic-level cleaning (6–8)	219,80 s	231,64 s

Nous constatons que le temps d'exécution n'a pas été significativement réduit ; au contraire, il a même légèrement augmenté.

Cela suggère que l'opération de lecture ou de génération du mapping ne représente pas la principale source de coût. En réalité, c'est l'application du mapping sur l'ensemble du jeu de données qui demeure l'étape la plus coûteuse en temps de calcul.

**Implémentation distribuée (Dataflow)** Pour intégrer l'apprentissage au projet, une solution équivalente a également été mise en place via Dataflow. Les principes d'Apache Beam et du moteur d'exécution Google Dataflow ont déjà été présentés dans la section 3.2.

La capture d'écran ci-dessous 11 illustre le résultat du travail de Dataflow, qui exécute le même pipeline de nettoyage et de validation des données. Le contenu du pipeline est équivalent à l'implémentation réalisée avec Pandas.

Nous pouvons constater que toutes les étapes se sont terminées avec succès, chacune en un temps d'exécution relativement court (de 0 à 17 secondes). La durée totale du travail est cependant de 5 minutes et 50 secondes.

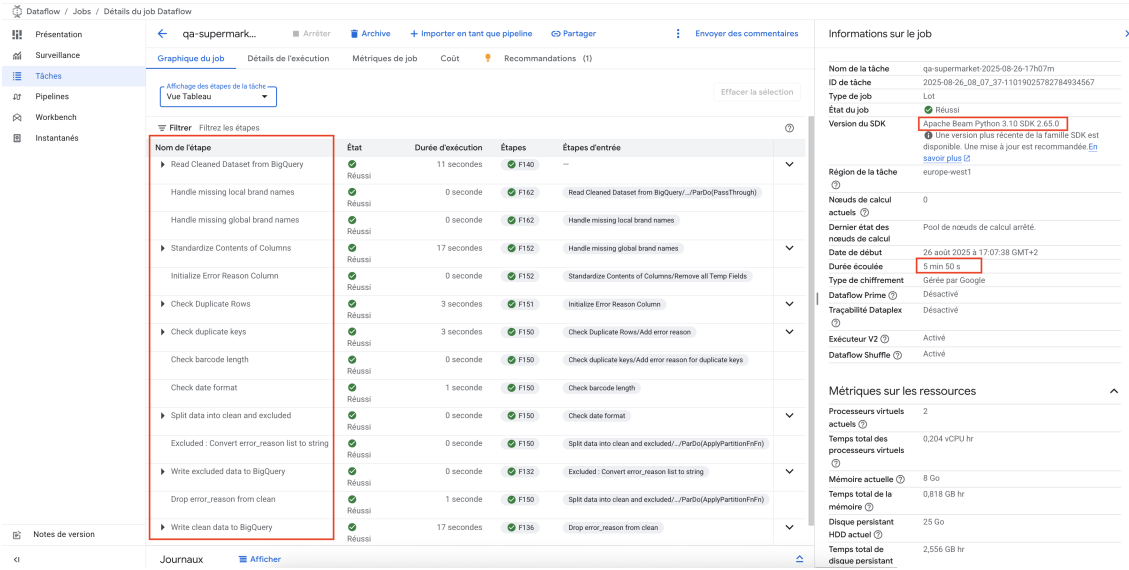


FIGURE 11 – Résultat du travail Dataflow exécutant le même pipeline

Ce écart s'explique par la nature même de Dataflow : bien que le temps de calcul unitaire par étape soit faible, la durée totale est dominée par les coûts fixes d'orchestration et d'initialisation. Ceux-ci incluent le déploiement du pipeline, la création et l'initialisation des *workers*, la sérialisation des données entre étapes, ainsi que les latences associées aux opérations d'E/S (Entrées/Sorties) avec BigQuery. Sur un volume de données relativement modeste, ces coûts fixes deviennent prépondérants et expliquent la différence entre le cumul apparent des durées d'étape et le temps total écoulé.

Néanmoins, cette approche distribuée prend tout son sens dans un contexte de volumes massifs de données ou de traitements récurrents, où l'autoscaling, la parallélisation et la tolérance aux pannes offerts par Dataflow constituent des atouts majeurs.

## 5.3 Validation des données par NLP

### 5.3.1 Contexte et objectif

Parallèlement au deuxième projet, une analyse approfondie a été conduite sur le même jeu de données, qui est présenté dans la sous-section 5.2.1.

Chaque produit est catégorisé par 11 variables hiérarchiques, dont 6 catégories locales et 6 catégories globales. Il est à noter que les deux groupes de variables partagent la première hiérarchie, d'où un total de 11 variables au lieu de 12.

En pratique, nous avons relevé plusieurs incohérences dans ces variables hiérarchiques, par exemple :

- « FOOD » → « TEXTILE »,
- « NO FOOD » → « TRADITIONAL FRESH FOOD ».

Lors de la phase de conception, le principal défi a résidé dans la difficulté d'annoter les hiérarchies pertinentes sans validation directe du client. Par ailleurs, l'examen approfondi du jeu de données n'a mis en évidence que peu d'erreurs, ce qui suggère un déséquilibre marqué entre les exemples positifs (`is_error = 1`) et négatifs (`is_error = 0`).

Nous avons d'abord formulé le problème comme une classification multi-classes visant à prédire le niveau hiérarchique. Toutefois, afin de tenir compte de la contrainte d'annotation, nous avons finalement retenu une approche binaire, plus simple et plus pragmatique : identifier et filtrer les lignes suspectes. Les corrections effectives relèvent ensuite des équipes clientes.

Ainsi, l'objectif principal de la détection exhaustive des anomalies dans les variables hiérarchiques, en amont de toute correction métier.

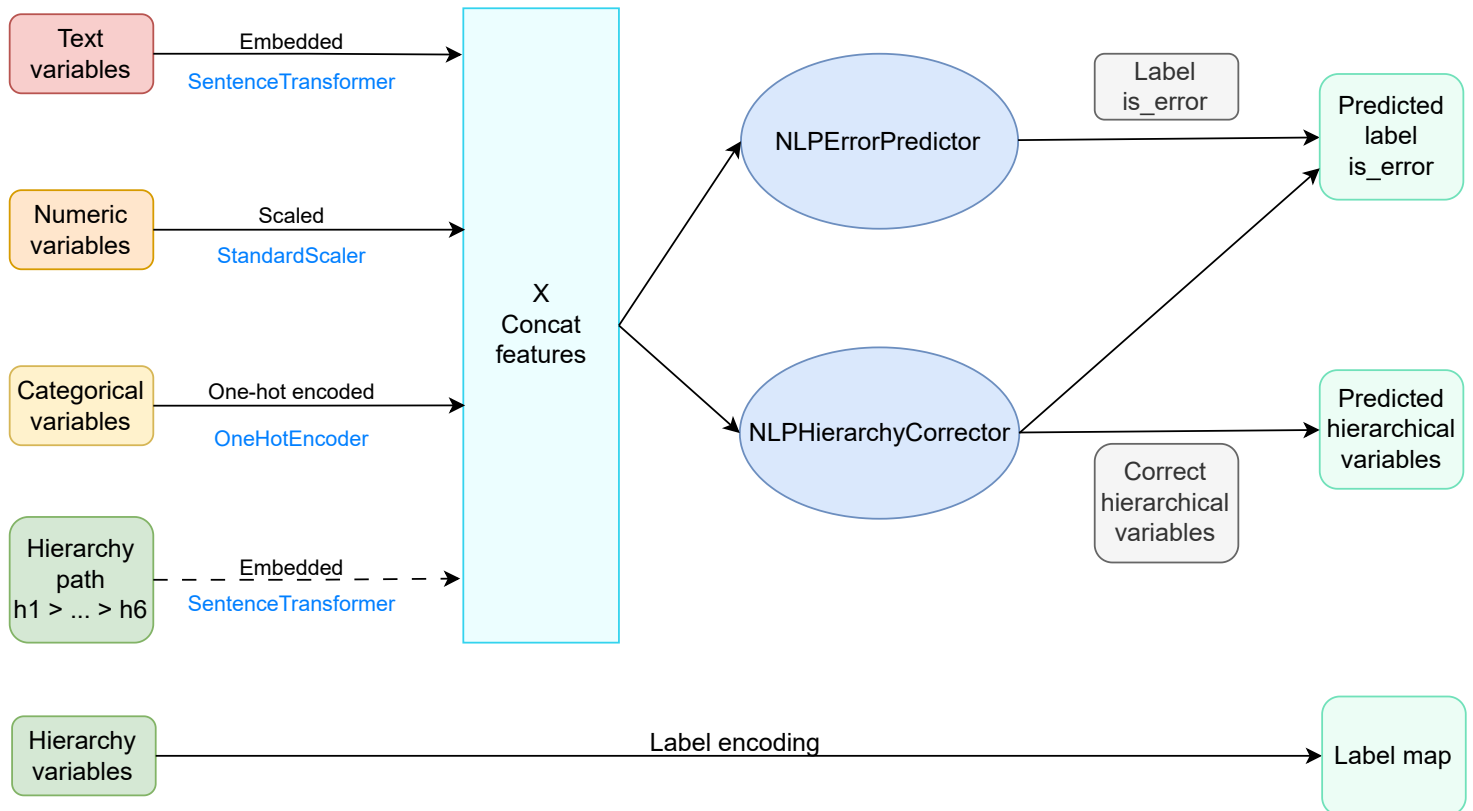
### 5.3.2 Construction de l'architecture

Le schéma de l'architecture est présenté en page suivante.

## 1. Features

## 2. Training / Inference

## 3. Output



Input

Model

Loss

Output

Optional  
-- -- -- -->

Embedding model: `SentenceTransformer('paraphrase-multilingual-MiniLM-L12-v2')`  
Metrics : Confusion Matrix / Precision / Recall / F1

2025/09/15 Liuxin YANG



**Choix des variables** Nous avons remarqué que pour des produits partageant la même description (`item_desc`), les hiérarchies ne sont pas toujours identiques. Il est donc nécessaire d’exploiter d’autres variables pertinentes. À l’issue de nos tests, en complément des variables hiérarchiques, nous avons retenu six variables pour la construction des entrées : `item_desc`, `color_desc`, `size_desc`, `pro_mod_desc`, `local_brand_name` et `global_brand_name`.

**Construction des entrées** Comme illustré par le schéma de l’architecture, le vecteur d’entrée  $X$  résulte de la concaténation de quatre blocs complémentaires :

1. Les textes sont encodés par le modèle d’embedding choisi ;
2. Les variables continues sont standardisées afin de limiter les effets d’échelle ;
3. Les variables catégorielles tabulaires sont transformées en vecteurs *one-hot* ;
4. Le chemin hiérarchique est sérialisé ( $h_1 \rightarrow \dots \rightarrow h_6$ ) puis projeté dans un espace vectoriel sémantique.

Cette conception s’inspire de l’approche proposée par Huang et al. [Hua+20]. Dans leur méthode, les embeddings des variables catégorielles sont enrichis par des couches Transformer afin de produire des *contextual embeddings*, ensuite concaténés avec les variables continues avant d’être passés à un MLP.

Dans notre cas, l’idée centrale est similaire : plutôt que de fusionner toutes les colonnes en un seul texte pour en extraire un unique embedding, nous préservons la nature hétérogène des données (texte, continu, catégoriel, hiérarchique), appliquons un encodage spécifique à chaque type, puis concaténons l’ensemble des représentations vectorielles obtenues avec les variables continues afin de former l’entrée du réseau.

**Choix du modèle d’embedding** Le modèle d’embedding : SentenceTransformer « paraphrase-multilingual-MiniLM-L12-v2 » a été choisi pour ce projet.

Ce modèle est une variante multilingue de la famille MiniLM, intégrée à la bibliothèque SentenceTransformers. Il produit des représentations vectorielles denses (embeddings) de dimension 384 pour des phrases ou courts paragraphes, tout en assurant un alignement sémantique entre plusieurs langues. Il est optimisé pour la détection de paraphrases et la similarité sémantique.

Ce modèle se distingue par sa robustesse multilingue : il fonctionne directement sur de nombreuses langues (français, anglais, etc.) et aligne sémantiquement des textes de langues différentes. Il offre en outre un excellent rapport qualité/performance : son architecture légère procure des temps d’inférence courts et une empreinte mémoire réduite, ce qui est pratique en environnement de production et/ou sans GPU.

**Détail du réseau de neurones** L’architecture choisie reste volontairement simple : un *MLP* à deux couches linéaires partagées, séparées par une non-linéarité ReLU et

un Dropout de 0,2. Ce choix se justifie par le fait que l'essentiel de la capacité de représentation provient des embeddings en entrée, tandis que le réseau doit simplement agréger et exploiter ces représentations.

À partir du vecteur d'entrée  $X \in \mathbb{R}^D$ , nous obtenons :

$$h = \text{ReLU}(W_2 \text{Dropout}(\text{ReLU}(W_1 X))) \in \mathbb{R}^H,$$

où  $H$  est la dimension cachée (`hidden_dim`).

Deux variantes sont ensuite utilisées selon la tâche :

- Détection d'erreur (`NLPErrorPredictor`) :  
une couche linéaire produisant deux logits ( $\mathbb{R}^2$ ) pour prédire l'étiquette `is_error`.
- Correction hiérarchique (`NLPHierarchyCorrector`) :  
 $K$  couches linéaires  $\{W_\ell h \in \mathbb{R}^{C_\ell}\}_{\ell=1..K}$ , une par niveau hiérarchique ( $C_\ell =$  nombre de modalités au niveau  $\ell$ ). Ce découpage reflète explicitement la structure hiérarchique des données.

Les têtes produisent des logits et l'entraînement s'effectue via une perte d'entropie croisée (`torch.nn.CrossEntropyLoss`), qui intègre la softmax en interne. En pratique, nous entraînons soit uniquement la tête binaire `is_error` (scénario « validation des données »), soit les têtes multi-classes lorsqu'une correction hiérarchique est visée.

Ainsi, cette architecture relativement simple se révèle suffisante : la complexité est portée principalement par les embeddings, tandis que le MLP assure une agrégation efficace et légère des représentations.

**Pondération de classe** Nous définissons un poids par classe  $c$  par

$$weight_c = \frac{N}{K count_c}, \quad (1)$$

où  $N$  est le nombre total d'exemples,  $K$  le nombre de classes et  $count_c$  le nombre d'exemples appartenant à la classe  $c$ . Cette formule rend la contribution totale de chaque classe ou loss approximativement égale : en effet,  $weight_c \times count_c = N/K$ .

Ainsi, les classes rares reçoivent un poids plus élevé, et les classes fréquentes un poids plus faible, ce qui compense le déséquilibre lors de l'apprentissage.

**Choix des métriques** Nous évaluons d'abord la tête `is_error` (problème binaire) à l'aide de la matrice de confusion et des métriques dérivées : exactitude, précision, rappel et F1.

Soit, en notant 1 la classe « erreur » et 0 la classe « non-erreur » :

$$\mathbb{M}_{confusion} = \begin{pmatrix} \text{TN} & \text{FP} \\ \text{FN} & \text{TP} \end{pmatrix}, N + P = \text{TP} + \text{TN} + \text{FP} + \text{FN},$$

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{N + P}, \quad \text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}},$$

$$\text{F1} = 2 \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.$$

Pour la tête de correction hiérarchique (multi-classe par niveau), nous mesurons l'exactitude par niveau uniquement sur les échantillons réellement non erronés, afin d'évaluer la capacité du modèle à reproduire les niveaux corrects quand l'instance n'est pas fautive.

Soit  $L$  le nombre de niveaux (ici  $L = 6$ ) et  $M = \{i : y_i^{(\text{err})} = 0\}$  l'ensemble des indices non erronés ; pour chaque niveau  $\ell \in \{1, \dots, L\}$  :

$$\text{Accuracy}_\ell = \frac{1}{|M|} \sum_{i \in M} \mathbf{1}\{\hat{y}_{i,\ell} = y_{i,\ell}\}.$$

*Remarques pratiques.* Dans un contexte potentiellement déséquilibré (peu d'erreurs), la F1 est plus informative que l'accuracy pure pour `is_error`.

### 5.3.3 Résultats

Dans un premier temps, afin de valider le fonctionnement du code en local, nous avons

- entraîné un modèle sur un petit échantillon constitué de 15 exemples :
  - 10 exemples étiquetés « non-erreur »,
  - 5 exemples étiquetés « erreurs ».
- évalué sur 6 exemples :
  - 4 exemples étiquetés « non-erreur »,
  - 2 exemples étiquetés « erreurs ».

Les résultats obtenus sont les suivants :

```
Inference: return_levels=False, only predict label (is_error).
[is_error] Accuracy: 1.0000, Precision: 1.0000,
           Recall: 1.0000, F1-score: 1.0000
[is_error] Confusion Matrix:
[[4 0]
 [0 2]]
```

Ce test exploratoire a confirmé que l'architecture était opérationnelle.

Dans un second temps, nous avons

- entraîné le modèle sur un ensemble de données plus important de 7 500 exemples, dont 7 étiquetés « erreurs ».
- évalué sur un jeu de test de 2 500 exemples, dont un seul était étiqueté « erreur ».

Les résultats obtenus sont les suivants :

```
Inference: return_levels=False, only predict label (is_error).
[is_error] Accuracy: 1.0000, Precision: 1.0000,
          Recall: 1.0000, F1-score: 1.0000
[is_error] Confusion Matrix:
[[2499 0]
 [0 1]]
```

Ces scores parfaits s'expliquent en partie par le déséquilibre extrême des classes. Ils montrent une bonne séparation sur l'annotation actuelle, mais ne garantissent pas la performance générale lorsque la prévalence ou la politique d'annotation évolueront.

Dans ce cas, nous affichons aussi un lot de candidats en fonction de leur incertitude. Les probabilités d'erreur prédites pour ces candidats sont présentées ci-dessous.

pred_is_error	pred_probs
1	1.000000e+00
0	1.978963e-19
0	2.709051e-20
0	1.916518e-20
0	1.084212e-20
0	9.676422e-21
0	9.030641e-21
0	8.480460e-21
0	7.505194e-21
0	5.369124e-21

Nous remarquons que le tri par  $p(\text{erreur})$  est peu informatif, les « dix premiers » présentant des probabilités très proches de zéro ( $\approx 10^{-21}$ ) du fait de la saturation du *softmax*.

En résumé, ces résultats doivent être interprétés avec prudence : forte dépendance à la prévalence observée, seuil implicite fixé par l'arg max, et sensibilité limitée aux cas ambigus. Une sélection complémentaire fondée sur l'incertitude (entropie/marge) reste nécessaire pour alimenter la révision manuelle.

## 6 Conclusion

Ce stage de fin d'études a constitué un complément essentiel à une première expérience de recherche effectuée en France. Alors que le stage de recherche permettait de se concentrer sur un sujet unique et approfondi, cette expérience en entreprise a imposé un rythme différent : concilier un projet « fil rouge » (machine learning) avec des livraisons opérationnelles récurrentes (Habu Clean Room). Dès le deuxième mois, des missions clients (réalisation de tableaux de bord et corrections de requêtes) ont été intégrées au quotidien afin de soutenir la charge de l'équipe. Cet équilibre, parfois exigeant, a progressivement appris à prioriser et à organiser le travail entre exploration méthodologique et résultats livrables.

L'objectif initial d'explorer le machine learning, en particulier le traitement automatique du langage naturel, a été maintenu : une brique NLP multilingue pour la détection d'incohérences hiérarchiques a été conçue et testée. Même si le tuteur n'était pas spécialisé en NLP, l'accompagnement a été solide sur l'ingénierie de données et l'industrialisation dans le cloud. Ce cadre a permis de développer des compétences concrètes et transférables : BigQuery, Dataflow (Apache Beam), Cloud Composer (Airflow), bonnes pratiques de gestion de version (Git), pipelines reproductibles et gouvernance de données. L'apprentissage s'est aussi appuyé sur une démarche proactive et autonome.

Au plan humain et organisationnel, cette expérience a réaffirmé l'importance de la communication régulière : partager l'avancement en temps utile, poser des questions tôt, clarifier les attentes et les hypothèses avant de démarrer, puis valider les jalons. Ces réflexes réduisent les retours en arrière et sécurisent les délais de livraison.

En synthèse, ce stage a permis (i) de contribuer à des actifs analytiques concrets (qualité des données, tableaux de bord, automatisation), (ii) de prototyper une approche NLP au service de la Data Quality, et (iii) d'acquérir des réflexes d'industrialisation dans un environnement Clean Room et GCP. À court terme, les perspectives d'amélioration portent sur la montée en charge du pipeline distribué, l'affinage des métriques et des seuils de détection, ainsi que l'intégration continue des modèles. Cette expérience fournit une base solide pour poursuivre des projets mêlant ingénierie de données, qualité et apprentissage statistique.

## 7 Déroulement du Stage

Grillet du déroulement du stage		
Semaine	Travail Réalisé	Activités des entreprises
N°1 : 01/04/25 - 04/04/25	Analyse des données (EDA) du tableau de référence du produit du client Retailer (FR).  Découverte de la plate-forme GCP ( <i>Google Cloud Platform</i> )	Prise de connaissances de l'entreprise.
N°2 : 07/04/25 - 11/04/25	Étude de la documentation du Big-Query (Premier pas et Migrer);  Détection des fautes de frappe et incohérences dans les variables hiérarchiques.	Rencontres d'équipe : Data Science groupe (DS), Stan et Thibault)
N°3 : 14/04/25 - 18/04/25	Implémentation d'un pipeline automatisé de nettoyage et de validation des données.  Prise en main de la plateforme Habu Clean Room;  Correction de bugs pour le client Livraison - Retail data.	Rencontres d'équipe : Sarah; Premier point d'avancement projet
N°4 : 22/04/25 - 25/04/25	Analyse des jeux de données du client Retailer (BE) & client E-commerce;  Amélioration du pipeline de nettoyage et de validation des données : - Remplacement des accents;  Étude de la cohérence entre les variables de description et les variables hiérarchiques.	
N°5 : 28/04/25 - 30/04/25, 02/05/25	Implémentation de requêtes SQL pour le client Livraison - Restaurant data.	Deuxième point d'avancement

Grillet du déroulement du stage		
Semaine	Travail Réalisé	Activités des entreprises
N°6 : 05/05/25 - 07/05/25, 09/05/25	Création de tableaux de bord pour le client Livraison - Retail data;  Étude de l'article scientifique [Liu+19].	client Livraison (LA-TAM) x LR/Habu - Appel hebdomadaire
N°7 : 12/05/25 - 15/05/25	Création de tableaux de bord pour le client Livraison - Retail data/Restaurant data.  Amélioration du pipeline de nettoyage et de validation des données : - standardisation des valeurs manquantes; - vérification de la longueur des codes-barres.	
N°8 : 20/05/25 - 22/05/25	Validation des tableaux de bord pour le client Média (FR).	Atelier de l'équipe DS (UK&FR)
N°9 : 26/05/25 - 28/05/25	Mise en pratique des outils de gestion de version avec Git : - création et gestion de branches; - résolution de conflits; - rebase et synchronisation avec un dépôt distant (GitHub).  Étude de l'article scientifique [Hua+20];  Analyse du jeu de données du client Cinéma (ES);  Améliorations du pipeline du nettoyage et de validation des données : - correction du schéma; - restauration des caractères corrompus (" ?").	Troisième point d'avancement

Grillet du déroulement du stage		
Semaine	Travail Réalisé	Activités des entreprises
N°10 : 02/06/25 - 06/06/25	Suite des améliorations du pipeline du nettoyage et de validation des données : - correction du schéma ; - restauration des caractères corrompus (" ?").	Suivi de stage avec M.Caputo  Client Cinéma (ES) x LR : Clean Room Proposal Walkthrough
N°11 : 09/06/25 - 13/06/25	Implémentation de l'architecture envisagée de NLP (Traitement automatique du langage naturel), et test du modèle ;  Améliorations des tableaux de bord pour le client Média (FR).	
N°12 : 16/06/25 - 20/06/25	Ajout de colonnes de variables hiérarchiques annotées manuellement, ainsi qu'une étiquette de validation de la stratification sur l'ensemble d'entraînement ; Adaptation du code en conséquence.  Apprentissage du modèle dataflow : - refactorisation du code existant sous forme de pipeline Apache Beam ; - exécution locale du pipeline ;  Réalisation des tableaux de bord pour le client Média (FR).	
N°13 : 23/06/25 - 27/06/25	Exécution du pipeline Apache Beam sur GCP.  Améliorations des tableaux de bord pour le client Livraison - Retail data/Restaurant data.	Quatrième point d'avancement



Grillet du déroulement du stage		
Semaine	Travail Réalisé	Activités des entreprises
N°14 : 30/06/25 - 03/07/25	<p>Étude approfondie de la documentation d'Apache Beam ;</p> <p>Test des différentes plateformes de recherche de codes-barres ;</p> <p>Téléchargement de jeux de données publics ; Sélection des variables pertinentes ; Premier filtrage des données ;</p> <p>Transfert vers GCP et normalisation des champs textuels clés.</p>	
N°15 : 07/07/25 - 11/07/25	<p>Analyse et diagnostic d'incohérences dans les noms de marques ;</p> <p>Proposition de pseudo codes pour traiter les problèmes identifiés, puis implémentation en Python.</p>	
N°16 : 15/07/25 - 18/07/25	Optimisation de l'algorithme.	Cinquième point d'avancement
N°17 : 21/07/25 - 25/07/25	Validation du bon fonctionnement du code.	
N°18-20 : 28/07/25 - 14/08/25	<p>Réalisation des tableaux de bord demo pour le client Électronique et le client Média (FR) ;</p> <p>Développement d'un workflow Airflow sur Google Cloud Composer qui intègre les données de Habu CleanRoom pour le client Média (FR).</p>	<p>Suivre les projets de mon collègue Xinyu pendant son absence.</p> <ul style="list-style-type: none"> <li>— le client Média (FR)</li> <li>— le client Livraison (LATAM)</li> <li>— le client Cinéma (ES)</li> </ul> <p>Réunions avec le client Média (FR) et le client Électronique.</p>

Grillet du déroulement du stage		
Semaine	Travail Réalisé	Activités des entreprises
N°21 : 18/08/25 - 22/08/25	Validation de la qualité des nouveaux jeu de données du client Cinéma (ES).	
N°22 : 25/08/25 - 29/08/25	Rédaction du rapport ;  Conception d'une solution Airflow pour intégrer les données des partenaires.	Semaine d'IA
N°23 : 01/09/25 - 05/09/25	Rédaction du rapport ;  Développement de requêtes SQL pour le client Média (FR).	
N°24 : 08/09/25 - 12/09/25	Rédaction du rapport ;  Amélioration de l'architecture envisagée de NLP, et test du modèle.	
N°25 : 15/09/25 - 19/09/25	Rédaction du rapport ;  Amélioration de l'architecture envisagée de NLP, et test du modèle.	
N°26 : 22/09/25 - 26/09/25	Rédaction du rapport ;  Réalisation des tableaux de bord demo pour le client Assurance (UK) ;	

## Bibliographie

- [Liu+19] Yinhan LIU et al. *RoBERTa : A Robustly Optimized BERT Pretraining Approach*. 2019. arXiv : [1907.11692 \[cs.CL\]](https://arxiv.org/abs/1907.11692). URL : <https://arxiv.org/abs/1907.11692>.
- [Hua+20] Xin HUANG et al. *TabTransformer : Tabular Data Modeling Using Contextual Embeddings*. 2020. arXiv : [2012.06678 \[cs.LG\]](https://arxiv.org/abs/2012.06678). URL : <https://arxiv.org/abs/2012.06678>.

Documentations :

LiveRamp :

[https://docs.liveramp.com/safe-haven/en/understanding-safe-haven.html?utm\\_source=chatgpt.com](https://docs.liveramp.com/safe-haven/en/understanding-safe-haven.html?utm_source=chatgpt.com)

BigQuery :

<https://cloud.google.com/bigquery/docs/reference/libraries?hl=fr>

Dataflow :

<https://cloud.google.com/dataflow/docs/quickstarts/create-pipeline-python?hl=fr>

Airflow :

[https://fr.wikipedia.org/wiki/Apache\\_Airflow](https://fr.wikipedia.org/wiki/Apache_Airflow)

<https://cloud.google.com/composer/docs/concepts/overview?hl=fr>